Universität Augsburg
University

Universität Augsburg
Fakultät für Angewandte Informatik
Institut für Informatik

DISSERTATION

Interpolation-Assisted Evolutionary Rule-Based Machine Learning

Strategies to Counter Knowledge Gaps in
XCS-Based Self-Learning Adaptive Systems

submitted by

Anthony Stein

in partial fulfillment of the requirements for the degree

Dr. rer. nat.

“Doktor der Naturwissenschaften” (dt.)
“Doctor rerum naturalium” (lat.)

Erstgutachter:	Prof. Dr. rer. nat. Jörg Hähner
Zweitgutachter:	Prof. Dr. rer. nat. Bernhard Bauer
Drittgutachter:	Prof. Dr. rer. nat. Gregor Schiele
Tag der mündlichen Prüfung:	6. November 2019

Abstract

Self-adaptive systems are increasingly endowed with *Artificial Intelligence* technology in order to enhance system autonomy. Most prominently, algorithms from the research field of *Machine Learning* are utilized to allow autonomous agents to continually strive for increasing the system's utility based on experiences made over time. The resulting *self-learning adaptive systems* are typically deployed in dynamic non-deterministic environments which are characterized by continuing change and stochasticity regarding condition observations and utility measurements. This often leads to circumstances where the inner learning mechanisms are exposed to so far unseen and unanticipated system states for which they lack sufficient knowledge about how to react appropriately. In this thesis a novel, technical notion of *Knowledge Gaps* in self-learning adaptive systems is developed in order to characterize exactly this challenge. Knowledge gaps are assumed to be existent within the continually growing but limited knowledge bases of these systems. This gap-centric perspective is transferred to the acquisition process of incrementally knowledge building systems. Accordingly, it is intended to (1) pave the way for the development of novel techniques which aim at countering such knowledge gaps, and, (2) to strengthen the self-reflective capability of self-learning adaptive systems with regard to their current knowledge. The former aspect constitutes the main topic of this thesis. On the basis of a well-known *Evolutionary Rule-based Machine Learning* technique, which has been applied several times in the context of Organic Computing research, the algorithmic structure of this type of algorithms is enhanced toward explicitly counter existing gaps in their incrementally evolving knowledge bases. It will be demonstrated how the exploitation of so far acquired knowledge elements can be improved by further incorporating raw experiences in a transductive manner. Transduction here means to immediately leverage already made and remembered experiences instead of inducing a model first and deducing to new situations afterward. Furthermore, the initialization of newly constructed knowledge elements is enhanced. Again, it will be explained how to make transductive use of already existing but not directly matching knowledge elements in the proximity of the currently queried problem space niche about which the algorithm is not confident yet. This knowledge transduction is realized by utilizing methods from the domain of *scattered data interpolation* within the *XCS Classifier System* – the most prominent representative from the class of Michigan-style evolutionary rule-based machine learning systems. Results on a range of conducted empirical validation studies are reported to corroborate the hypothesized benefits of transductive knowledge inference in XCS by means of interpolation. Additionally, plausibly applicable parts of the introduced methodologies are transferred to a realistic application scenario in the context of self-adaptive traffic light control. In order to capture the aforementioned second advantage of knowledge gap-centric learning – the advance of the self-reflection property of self-learning adaptive systems – a novel research direction for autonomous learning which is termed *Proactive Knowledge Construction* is proposed and first steps toward *Proactive Learning Classifier Systems* are taken. It is elaborated on how concepts from the domain of *Active Learning* can be incorporated within the Organic Computing Multi-layer Observer/Controller reference architecture in order to actively seek and bridge knowledge gaps within these systems. Furthermore, in order to substantiate the rationale behind the general concept of proactive knowledge construction, an initial formal proof is outlined and a first methodology to implement the envisioned proactive behavior is delineated in the last part of this thesis.

To my dear Julia.

Contents

Publications of the Author

Journal Articles and Book Chapters

[Ste17b] A. **Stein**. “Reaction Learning”. In: *Organic Computing – Technical Systems for Survival in the Real World*. Birkhäuser, 2017. Chap. Basic Methods, pp. 287–328.

[Ste+17a] A. **Stein**, D. Rauh, S. Tomforde, and J. Hähner. “Interpolation in the eXtended Classifier System: An architectural perspective”. In: *Journal of Systems Architecture* 75 (2017), pp. 79–94.

Conference and Workshop Papers

[Bar+19] C. M. Barnes, K. Bellman, J. Botev, A. Diaconescu, L. Esterle, C. Gruhl, C. Landauer, P. R. Lewis, P. R. Nelson, A. **Stein**, C. Stewart, and S. Tomforde. “CHARIOT – A Continuous High-level Adaptive Runtime Integration Testbed”. In: *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. June 2019, pp. 52–55.

[Bra+19] J. Brandl, N. Breinl, M. Demmler, L. Hartmann, J. Hähner, and A. **Stein**. “Reducing Search Space of Genetic Algorithms for Fast Black Box Attacks on Image Classifiers”. In: *KI 2019: Advances in Artificial Intelligence*. Kassel, Germany: Springer International Publishing, Sept. 2019, pp. 115–122.

[GSH19] M. Görlich, A. **Stein**, and J. Hähner. “Towards Physical Disturbance Robustness in Organic Computing Systems Using MOMDPs”. In: *2019 Intelligent Systems Workshop to appear in Workshop Proceedings of the 32nd International Conference on Architecture of Computing Systems (ARCS 2019)*. VDE Verlag GmbH, Berlin Offenbach, Mai 2019.

[PSH19] D. Pätzel, A. **Stein**, and J. Hähner. “A Survey on Formal Theoretical Advances Regarding XCS”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO ’19. Prague, Czech Republic: ACM, 2019, pp. 1295–1302.

[SSH19]	H. Stegherr, A. **Stein**, and J. Hähner. "Parallel Chemical Reaction Optimization for Utilization in Intelligent RNA Prediction Systems". In: *2019 Intelligent Systems Workshop to appear in Workshop Proceedings of the 32nd International Conference on Architecture of Computing Systems (ARCS 2019).* VDE Verlag GmbH, Berlin Offenbach, Mai 2019.
[ST19]	A. **Stein** and S. Tomforde. "Transfer Learning is a Crucial Capability for Intelligent Systems Self-Integrating at Runtime". In: *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W).* June 2019, pp. 32–35.
[Bel+18]	K. Bellman, J. Botev, A. Diaconescu, L. Esterle, C. Gruhl, C. Landauer, P. R. Lewis, A. **Stein**, S. Tomforde, and R. P. Würtz. "Self-Improving System Integration - Status and Challenges after Five Years of SISSY". In: *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W).* Sept. 2018, pp. 160–167.
[Ste+18a]	A. **Stein**, A. Margraf, J. Moroskow, S. Geinitz, and J. Hähner. "Toward an Organic Computing Approach to Automated Design of Processing Pipelines". In: *Workshop Proceedings of the 31st International Conference on Architecture of Computing Systems (ARCS 2018).* VDE Verlag GmbH, Berlin Offenbach, Apr. 2018, pp. 135–142.
[SMH18]	A. **Stein**, S. Menssen, and J. Hähner. "What About Interpolation? A Radial Basis Function Approach to Classifier Prediction Modeling in XCSF". In: *Proceedings of the Genetic and Evolutionary Computation Conference.* GECCO '18. **Best Paper Nominee**. Kyoto, Japan: ACM, 2018, pp. 537–544.
[Ste+18b]	A. **Stein**, S. Tomforde, A. Diaconescu, J. Hähner, and C. Müller-Schloer. "A Concept for Proactive Knowledge Construction in Self-Learning Autonomous Systems". In: *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W).* Sept. 2018, pp. 204–213.
[Ede+17]	S. Edenhofer, Y. Madkour, A. **Stein**, C. Stifter, and J. Hähner. "Bottom-Up Norm Creation in Open Distributed Computing Grids by Means of eXtended Classifier Systems". In: *Workshop Proceedings of the 30th International Conference on Architecture of Computing Systems (ARCS).* Vienna, Austria: VDE Verlag GmbH, Apr. 2017, pp. 55–62.
[Mar+17]	A. Margraf, A. **Stein**, L. Engstler, S. Geinitz, and J. Hähner. "An Evolutionary Learning Approach to Self-Configuring Image Pipelines in the Context of Carbon Fiber Fault Detection". In: *Proceedings of 16th IEEE International Conference on Machine Learning and Applications (ICMLA).* Cancún, Mexico: IEEE, Dec. 2017, pp. 147–154.

[SMH17] A. **Stein**, R. Maier, and J. Hähner. “Toward Curious Learning Classifier Systems: Combining XCS with Active Learning Concepts”. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Companion.* Berlin, Germany: ACM, July 2017, pp. 1349–1356.

[Ste+17b] A. **Stein**, S. Rudolph, S. Tomforde, and J. Hähner. “Self-Learning Smart Cameras – Harnessing the Generalization Capability of XCS”. In: *Proceedings of International Joint Conference on Computational Intelligence (IJCCI).* Funchal, Madeira-Portugal: SciTePress, Nov. 2017, pp. 129–140.

[SSH16a] M. Sommer, A. **Stein**, and J. Hähner. “Ensemble Time Series Forecasting with XCSF”. In: *Proceedings of 2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO).* Augsburg, Germany: IEEE, Sept. 2016, pp. 150–151.

[SSH16b] M. Sommer, A. **Stein**, and J. Hähner. “Local Ensemble Weighting in the Context of Time Series Forecasting using XCSF”. In: *Proceedings of 2016 IEEE Symposium Series on Computational Intelligence (SSCI).* Athens, Greece: IEEE, Dec. 2016, pp. 1–8.

[Ste+16a] A. **Stein**, C. Eymüller, D. Rauh, S. Tomforde, and J. Hähner. “Interpolation-based Classifier Generation in XCSF”. In: *Proceedings of 2016 IEEE Congress on Evolutionary Computation (CEC).* Vancouver, Canada: IEEE, July 2016, pp. 3990–3998.

[Ste+16b] A. **Stein**, D. Rauh, S. Tomforde, and J. Hähner. “Augmenting the Algorithmic Structure of XCS by Means of Interpolation”. In: *Proceedings of 29th International Conference on Architecture of Computing Systems (ARCS).* Nuremberg, Germany: Springer International Publishing, Apr. 2016, pp. 348–360.

[Ste+16c] A. **Stein**, S. Tomforde, D. Rauh, and J. Hähner. “Dealing with Unforeseen Situations in the Context of Self-Adaptive Urban Traffic Control: How to Bridge the Gap?” In: *Proceedings of 2016 IEEE International Conference on Autonomic Computing (ICAC).* Würzburg, Germany: IEEE, July 2016, pp. 167–172.

[Tom+16] S. Tomforde, D. Meier, A. **Stein**, and S. von Mammen. “Distributed Resource Allocation as Co-Evolution Problem”. In: *Proceedings of 2016 IEEE Congress on Evolutionary Computation (CEC).* Vancouver, Canada: IEEE, July 2016, pp. 1815–1822.

[Jän+15] U. Jänen, C. Grenz, S. Edenhofer, A. **Stein**, J. Brehm, and J. Hähner. “Task Execution in Distributed Smart Systems”. In: *Proceedings of the 8th International Conference on Internet and Distributed Computing Systems (IDCS).* Windsor, UK: Springer-Verlag New York, Inc., Sept. 2015, pp. 103–117.

[Mär+14] C. Märtin, A. **Stein**, B. Prell, and A. Kesper. “HCI-Patterns for Developing Mobile Apps and Digital Video-Assist-Technology for the Film Set”. In: *Proceedings of Human-Computer Interaction. Theories, Methods, and Tools: 16th International Conference, HCI International 2014.* Heraklion, Crete, Greece: Springer International Publishing, June 2014, pp. 320–330.

[Mär+13] C. Märtin, A. **Stein**, B. Prell, and A. Kesper. “Mobile App-Support for Advanced Digital Video-Assist Systems in Computer-Supported Film Sets”. In: *6th Forum Medientechnik, Fachhochschule St. Pölten.* 2013, pp. 171–182.

Doctoral Symposia

[Bie+17] M. Bieshaar, A. Calma, C. Gruhl, S. Rudolph, and A. **Stein**. “Machine Learning in Organic Computing: A Brief Clarification of Terms and Concepts”. In: *Organic Computing Doctoral Dissertation Colloquium 2016, June 2–3.* Duisburg, Germany: kassel university press GmbH, June 2017, pp. 113–130.

[Ste17a] A. **Stein**. “Evolutionary Online Learning in Organic Computing Systems”. In: *Organic Computing Doctoral Dissertation Colloquium 2016, June 2–3.* Duisburg, Germany: kassel university press GmbH, June 2017, pp. 17–33.

[Ste16] A. **Stein**. “Evolutionary Online Machine Learning from Imbalanced Data”. In: *Proceedings of IEEE 1st International Workshops on Foundations and Applications of Self-* Systems (FAS*).* Augsburg, Germany: IEEE, Sept. 2016, pp. 281–286.

[Ste14] A. **Stein**. “Neighborhood-based Interpolation for XCS Improvements”. In: *Organic Computing: Doctoral Dissertation Colloquium 2014, May 22–23.* Kassel, Germany: kassel university press GmbH, May 2014, pp. 71–83.

List of Figures

List of Tables

List of Abbreviations

AC Autonomic Computing

AI Artificial Intelligence

AL Active Learning

ANN Artificial Neural Network

ANOVA Analysis of Variance

ASI Action Selection Integration

CAS Complex Adaptive Systems

CBP Checkerboard Problem

CBR Case-based Reasoning

CII Covering Intialization Integration

CIL Collaborative Interactive Learning

DL Deep Learning

DNN Deep Neural Network

EA Evolutionary Algorithm

EC Evolutionary Computation

EML Evolutionary Machine Learning

ERBML Evolutionary Rule-based Machine Learning

FIFO First-In-First-Out

GA Genetic Algorithm

GPGPU General Purpose Graphical Processing Unit

i.i.d.	independent, identically distributed
IC	Interpolation Component
ICT	Information and Communication Technology
IDW	Inverse Distance Weighting
IPO	Input-Process-Output
KDE	Kernel Density Estimation
KG	Knowledge Gap
LCS	Learning Classifier System
MDP	Markov Decision Process
ML	Machine Learning
MLA	IC-attached Machine Learning Algorithm
MLI	Machine Learning System Interface
MLOC	Multi-Layer Observer/Controller
MSM	Modified Shepard's Method
NaNe	Natural Neighbor
NeNe	Nearest Neighbor
NSE	Non-Stationary Environment
O/C	Observer/Controller
OBR	Ordered Bound Representation
OC	Organic Computing
OII	Offspring Initialization Integration
OL	Online Learning
OML	Online Machine Learning
OTC	Organic Traffic Control

PAL	Proactive Learning
IPI	Interpolated Prediction Integration
QQ-plot	Quantile-Quantile Plot
RBF	Radial Basis Function
RL	Reinforcement Learning
RLS	Recursive Least Squares
SAS	Self-Adaptive System
SASO	Self-Adaptive and Self-Organizing
SCN	Smart Camera Network
SCPD	Strictly Conditional Positive Definite
SL	Supervised Learning
SLAS	Self-Learning Adaptive System
SOS	System-of-Systems
SSL	Semi-Supervised Learning
SuOC	System under Observation and Control
TL	Transductive Learning
TPS	Thin-Plate-Spline
UBR	Unordered Bound Representation
UCS	Supervised Classifier System
UL	Unsupervised Learning
XAI	Explainable AI
XCS	XCS Classifier System
XCS-O/C	XCS Classifier System for the Observer/Controller architecture
XCSF	XCS Classifier System for Function Approximation
XCSI	XCS Classifier System for Integer-valued Input
XCSR	XCS Classifier System for Real-valued Input

Chapter 1.

Introduction

During the past decades, a continuing rise in system complexity has been observed. Modern *Information and Communication Technology* (ICT) systems are comprised of a large number of subsystems, each coming with their own set of configurable parameters. These subsystems can interact with each other and, thus, need to be interconnected to a certain degree. Additionally, such systems are not deployed in stationary environments, free of uncertainties and other influencing conditions. Actually, ICT systems are typically deployed in the real world, i.e., surrounded by an environment that exhibits challenging characteristics. Examples are non-determinism, coincidence, time-dependent influences, or other external factors that are not part of the systems themselves. Such surroundings are typically referred to as *Non-Stationary Environment* (NSE) [Dit+15].

The resulting system complexes, which can also be understood as *System-of-Systems* (SOS) [Tom+14; BTW14], are virtually impossible to manage by a single human administrator anymore. In order to allow for an appropriate functioning even in the presence of these highly complex system structures and their changing environments, one viable way is to increase the systems' autonomy. Decisions that used to be felt at the design time by the system engineers, are now postponed to the actual runtime of the systems. This comes at the cost of hardly predictable system behavior. On the other hand, during the design time it is nearly impossible to anticipate all possible system states for which the engineer might "hard code" an appropriate response a priori. In order to achieve autonomy, the systems need to be equipped with certain mechanisms that pave the way for on-demand decision making and the initiation of reconfiguration processes at runtime. One solution for achieving such a desirable self-adaptive behavior is provided by the field of *Machine Learning* (ML) [Mit97; Alp10].

ML is a branch of computer science, as well as one of the most important subfields of *Artificial Intelligence* (AI) [RN95] today. Research in ML is concerned with the understanding of techniques that, roughly speaking, allow machines, i.e., computers, to learn from data. One of the most common definitions of ML was coined by Tom M. Mitchell in [Mit97], who states that:

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*

The integral components according to Mitchell's definition are *experience*, *performance measures* and *tasks*. From a systems engineering perspective, each technical system is developed for a certain purpose which constitutes its task. A system is monitored and assessed in terms of certain utility measures, which can be mapped to the performance measures of Mitchell's definition. Finally, a running system experiences new situations every time it is exposed to some input that has to be processed further and subsequently results in some sort of output. A more detailed introduction to the field of ML will be given in Chapter 2.

ML is not the only field concerned with building autonomously learning systems – which therefore might become self-adaptive and self-organizing. Related research directions which share common aspirations are for instance: (1) Methods from the domain of *numerical optimization* [WN99], a field which is strongly related to ML, and its subarea of *metaheuristics*. (2) *System theory*, which includes attempts to gain an understanding of how to cope with complexity. (3) Concepts from *social sciences*, e.g., ways to establish a notion of trust and reputation in distributed computing systems [Rei+16]. Methods from all these domains have been investigated in order to accomplish increasing system autonomy.

Research initiatives have been founded to establish communities that cooperatively advance the development of intelligent systems. Among them, *Organic Computing* (OC) [MT17a], as well as *Autonomic Computing* (AC) [KC03], are two prominent representatives. Both are concerned with building *Self-Adaptive and Self-Organizing* (SASO) systems that relieve the human administrators but at the same time allow for viable and robust system functioning in spite of changing conditions and goals, as well as other disturbances such as component failures.

The scenario of *self-adaptive traffic light control* at urban intersections (as already thoroughly investigated in e.g., [Tom12; Pro11]) shall serve as a tangible running example for a realistic application in the course of the present thesis: Consider a number of interconnected embedded technical systems, here traffic lights, equipped with self-learning computation units (the traffic light controllers running the learning *program*). These subsystems are deployed in a highly dynamic environment, i.e., the traffic system with all its autonomously acting participants. The *Self-Learning Adaptive System* (SLAS) has to fulfill the *task* of preserving an appropriate degree of traffic flow (*performance measure*) by autonomously adapting the green phases of all traffic lights installed at the observed intersection (the *environment*). At arbitrary points in time, disturbances can challenge the self-learning system with unforeseen situations and system states. For example, road blocks can be established due to lacking water pipes, or traffic accidents might happen occasionally. When the system was (1) not explicitly designed to handle such exceptional situations a priori, and, (2) also has no rule or other knowledge acquired by itself via its learning mechanisms, it faces a circumstance regarded as *Knowledge Gap* (KG) in this thesis. For instance, such a KG could prevent the system to adequately reconfigure the green times in spite of severe unexpected changes in the traffic load of a particular intersection leading to jams and increasing delays.

The development of first principles and building blocks for a general methodology based on *scattered data interpolation* to 'bridge such knowledge gaps' in SLAS constitutes the main objective of the present thesis. Therefore, a family of incrementally knowledge constructing learning algorithms is focused. In the context of OC, these algorithms have proven successful in endowing complex technical systems with the capability of self-learning robust adaptation strategies. The subsequent section outlines this thesis' scientific contribution in a more detailed way.

1.1. Scientific Contribution

In this thesis, it is focused on ML methods which have been successfully utilized several times in OC systems. These methods, called *Learning Classifier Systems* (LCS), belong to the class of *Evolutionary Rule-based Machine Learning* (ERBML). Throughout this thesis, only *Michigan-style Learning Classifier System* (LCS) are considered (cf. Sect. 2.4 for more details). ERBML algorithms following this learning style incrementally build up a knowledge base. Each knowledge element targets a certain subspace of the entire learning problem. These techniques are capable of *Online Learning* (OL), which means that they adapt their solution (i.e., knowledge base, hypothesis or model) each time a new observation is available. Especially in the case of *Self-Adaptive Systems* (SAS) which are asked to learn continually at runtime, OL plays an important role. Furthermore, ERBML algorithms make use of computational evolution, which endows them with the inherent capability of adapting to change in NSEs. The rule-based knowledge representation constitutes another advantage of these techniques. Encoded in an IF-THEN rule style, these knowledge elements are directly interpretable and understandable for humans (at least in theory). *Explainable AI* (XAI) [GA19], or *Interpretable ML* [Mol19] attracts huge research attention these days. This is due to the outstanding progress made in solving vastly complex tasks which is clearly dominated by *Deep Neural Network* (DNN) and general *Deep Learning* (DL) approaches [GBC16]. However, these achievements come to the expense of explainability. This, however, constitutes a crucial aspect in guaranteeing autonomous systems to comply with *exploration boundaries* in safety-critical tasks, e.g., where humans are involved. A topical example of such a safety-critical scenario is *Autonomous Driving*. Erroneously interpreted sensor perceptions at the time when the employed algorithms are updating their internal world model regarding the traffic context might lead to dangerous situations. For instance, inappropriate control or steering signals can have fatal consequences for the occupants or others. For the sake of transparency and traceability of the decisions an autonomous system continually fells, the self-learned internal models need to facilitate a "view into the blackbox". So-called *symbolic* methods of ML or rule-based systems as considered in this thesis fulfill this requirement. They learn knowledge representations which are naturally human comprehensible, such as production rules or decision trees.

This thesis aims at extending the competence of a certain representative of the ERBML field – the *XCS Classifier System* (XCS) – for the purpose of a more robust use in SLAS. Secondarily, the awareness about the general class of LCS algorithms as viable candidate solutions for the utilization in autonomous systems is to be increased. This work is furthermore intended to push the research field concerned with ML in SAS a step toward a unifying understanding regarding the aspects and challenges specific to SLAS.

Accordingly, a system model will be derived in Chapter 3. It introduces a unified notion and basic definitions to provide a common terminology for this thesis. Whenever we think about 'learning', the goal is to acquire knowledge about a certain task by making experiences with this task (recall Mitchell's definition).

Looking at this notion from a different angle, a common model that illustrates the relationships between raw (sensory) data, information and knowledge can be found in Figure 1.1. It shows the well-known concept of the *Wisdom Pyramid* [Row07].

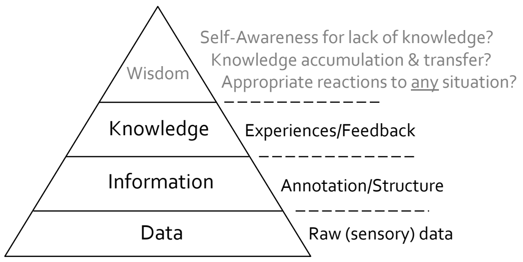

Wisdom
Self-Awareness for lack of knowledge?
Knowledge accumulation & transfer?
Appropriate reactions to any situation?
Knowledge
Experiences/Feedback
Information
Annotation/Structure
Data
Raw (sensory) data

FIGURE 1.1.: The pyramid of wisdom, adapted from [Row07]

The tip of the pyramid is wisdom – a goal every learning entity is pursuing. However, until wisdom can be reached, a couple of intermediate steps have to be done. Starting from raw data, one has to add some semantics to gain information. This could be the scale and domain of a sensory measurement, coupled with some implication. For instance, a temperature sensor that measures -5 degree Celsius could be annotated with the implication "cold". If a learner repeatedly receives such an annotated information together with some sort of feedback, it will gain experience regarding this information, which results in knowledge. Whenever we are able to further assess the acquired knowledge in terms of some criteria, this leads to the wisdom each learning entity is (more or less urgently) striving for.

This thesis introduces a novel concept of thinking beyond the strictly technical means of inducing models from data and assessing resulting hypothesis as a whole – the concept of *Knowledge Gaps* (KGs). Learning systems should gain awareness of their own KGs and consequently the ability to steer the learning process in order to close them. This KG-centric view on learning algorithms is deemed to be promising and viable, especially in the context of SAS. However, in this thesis only the tip of the iceberg can be touched upon, since it is at its very beginning, yet.

The major contribution of this thesis, is to develop first principles and techniques to implicitly overcome KGs in the course of online learning at runtime. More precisely, a methodology will be developed that takes advantage of already existing knowledge that could be acquired via *similar* experiences previously made. This methodology is termed *Transductive Knowledge Inference*. The overall goal is to improve the knowledge exploitation and construction processes by interpolating novel knowledge based on collected raw experiences with the environment (the task to solve) or already existing knowledge elements learned for similar situations.

To break the above stated down to a concise research statement, the main objective of this thesis can be summarized as providing a scientific confirmation of the following hypothesis:

Consider an incremental ERBML algorithm $L(K)$ as briefly sketched above. $L(K)$ incrementally builds up a knowledge base K, which is comprised of single knowledge elements $k \in K$. $L(K)$ is employed within a SLAS which is deployed in an NSE facilitating the occurrence of knowledge gaps kg which are not covered in K of L.

Hypothesis. *The* ***transductive exploitation*** *of* ***collected experiences*** *as well as the* ***transductive construction*** *of new knowledge elements $k^* \notin K$ on the basis of* ***existing*** *$k \in K$ by means of* ***scattered data interpolation*** *yield an improved learning (or sample) efficiency of L in terms of faster decreasing initial prediction errors.*

In what follows, different derivatives of the *XCS Classifier System* (XCS) [Wil95] introduced by Stewart W. Wilson in 1995, will serve as representatives for the class of ERBML algorithms which follow the Michigan-style. XCS constitutes the most thoroughly investigated system today. Several variants and further descendants have been proposed in the course of the past two decades. Due to this fact, nearly all extensions and insights gained in the last ten years can be fed back to the original XCS system, which renders it an ideal candidate for serving as case study in this work.

As a further innovation, novel approaches for creating knowledge in a *(pro)active* manner are initially elaborated during a discussion of future research directions at the end of this thesis. This proactive means of constructing novel knowledge elements stays in contrast to the reactive way most systems rely on. Therefore, concepts from the domain of *Active Learning* (AL) [Set09] have been adopted for the use within

XCS for the first time. The main idea behind AL is to bring a so-called *oracle* into the learning loop which is supposed to answer posed queries about the learning task. Most often, a human expert is assumed to take on the role of the oracle. In this thesis, however, this assumption is loosened by also considering simulations, heuristics, or even the interpolation methodology to be introduced in the first part. A picture of a novel research direction for *Proactive Knowledge Construction* is drawn and corroborated by informal as well as formal definitions that pave the way for future research activities.

As a last objective, this thesis is intended to promote the visibility of LCS in general and to convey an understanding beyond what these systems are sometimes deemed to be – yet simply another machine learning algorithm. Michigan-style LCS such as XCS should be more generally seen as a rich framework for interpretable and online machine learning based on an evolving ensemble of individual models that collectively solve the underlying learning problem. This alternative view is supposed to open a wide variety of promising research activities devoted to the incorporation of LCS in modern AI applications.

1.2. Structure of this Thesis

The present work is organized into the following parts:

- The first three introductory chapters sketch the scientific context, provide necessary background information and pose a corresponding problem statement.
- In the main part of this doctoral thesis comprising six chapters, it is elaborated on the development and the evaluation of several novel methodologies to attempt the identified problem.
- An elaborate discussion of future research directions points toward two concrete methodologies for advancing and carrying on the results of this work.
- The thesis is then closed with a summary of the results and an outlook on related scientific fields where the outcomes of this thesis might further contribute.

Subsequent to this first introduction chapter, the thesis proceeds as follows: Chapter 2 provides a comprehensive introduction of the basic concepts and methods used in this thesis. Section 2.1 starts with a brief review of the *Organic Computing* (OC) research initiative which was started back in 2002 and focuses on engineering technical, self-adaptive and self-organizing systems for survival in the real world. Since this work focuses on the self-learning aspects of SAS, an overview of the field of *Machine Learning* is given in Section 2.2. A formal definition of the *interpolation problem* as well as a short description of selected interpolation methods can be found in Section 2.3. Afterward, a comprehensive introduction into the field of *Learning Classifier Systems* (LCS) and the *XCS Classifier System* (XCS) is provided in Section 2.4.

The succeeding Chapter 3 revisits the challenges SLAS face when they are deployed in NSEs. A detailed problem statement is formulated descriptively and formally. The chapter begins with the introduction of a general notion of *Learning in Organic Computing Systems* in Section 3.1. On that basis, Section 3.2 derives a unifying system model and integrates the most challenging aspects which are deemed catalysts for knowledge gaps to occur. A formal notion of the aforementioned challenge of *knowledge gaps* is developed in Section 3.3. Subsequently in Section 3.4, *LCS-based learning* is brought in line with the just derived system model, the potentials and advantages, but also the drawbacks of LCS are discussed.

After the introduction of all basic concepts, a unified system model and the problem statement, the main part of this thesis starts – the augmentation of XCS by means of integrating interpolation into its algorithmic structure. Chapter 4 starts out with an architectural view on XCS. A generic *Interpolation Component* (IC) is introduced in Section 4.1. The generic IC can be used in combination with nearly any ML algorithm (here XCS) in its *loose coupling* approach (cf. Sect. 4.2). A *tightly interwoven* variant especially designed for XCS is presented in Section 4.3. Potentials and shortcomings of both variants are further discussed in Section 4.4 followed by an appreciation of related work.

The succeeding Chapters 5, 6, 8 and 9 introducing the interpolation-based strategies to counter KGs share a common structural blueprint:

1. A recap of the *basic mechanisms* used in the conventional XCS variants.
2. The introduction of the newly devised *interpolation incorporating strategies*.
3. A dedicated *empirical validation study* that demonstrates the beneficial effects on XCS's learning behavior.
4. A *discussion* section that further discusses the obtained results along with current limitations and aspects of future work.
5. A dedicated appreciation of *related work* is presented at the end of each chapter.

In Chapter 5, a novel *interpolation-assisted action selection regime* is developed. Chapter 6 focuses on an XCS-specific reactive knowledge construction mechanism, also known as *Covering* in the LCS literature. Specifically, the initialization of newly constructed knowledge elements is enhanced by transductive knowledge inference by means of interpolating between existing rules. The so far developed techniques are applied to a realistic scenario in Chapter 7. A case study utilizing a special-purpose XCS modification within the *Organic Traffic Control* (OTC) system is subject of investigation. Due to the specific adaptations regarding the substantially adapted

XCS, a transfer of the novel approaches developed in the succeeding Chapters 8 and 9 is not plausible. The approach developed in Chapter 6 is then taken a step further and transferred to the initialization of *offspring* rules as created by the evolutionary component of an LCS in Chapter 8. As a last interpolation-based methodology, in Chapter 9 one of the most crucial steps in nearly any ML technique – the actual *prediction* step – is considered and a novel interpolation-based prediction modeling technique is devised.

The subsequent Chapter 10 is intended to provide an elaborate discussion about two concrete directions of future research. The outlined research directions are inspired by the main insights as will be obtained throughout this thesis and take the respective research a step further toward an entirely novel methodology in the domain of LCS as well as SLAS – *Proactive Knowledge Construction*. Therefore, in Section 10.1 the concept of *Active LCS* is briefly introduced. It starts with a clarification of why its current knowledge construction process is deemed to be mostly reactive. Afterward, ways to adopt specific AL techniques in order to extend the knowledge construction process are presented. Preliminary results on a novel multi-class toy problem are reported which corroborate the expected benefits and are then further discussed. As a next reasonable step toward *Proactive Learning Classifier Systems*, Section 10.2 starts with introducing the notion of *Proactive Knowledge Construction*. The reasonableness of this concept is then formally proved for the first time. A possible solution to proactively construct new knowledge elements for insufficiently explored problem space niches based on *Kernel Density Estimation* (KDE) is outlined. Finally, the findings are discussed. The chapter ends with another literature review on existing and recent research aspirations to introduce concepts similar to proactivity into ML algorithms.

The last Chapter 11 briefly summarizes the main results of this thesis (Sect. 11.1) and finally spends a few concluding thoughts on possibly impacted research fields in the last Section 11.2.

1.3. Integration of the Author's Published Works

Parts of this doctoral thesis have been previously published in several scientific contributions where this thesis' author has taken the role of the leading investigator. The following paragraphs briefly indicate which publications integrate into which chapter of this doctoral thesis.[1]

Chapter 2 introduces the theoretical backgrounds and prerequisites necessary for this work. Among these prerequisites, a comprehensive introduction to *Learning Classifier Systems* (LCS) is provided which is mainly based on [Ste17b].

For the problem statement described in Chapter 3, core challenges of learning in *Organic Computing* (OC) systems and a formal notion of *Knowledge Gaps* (KGs) are introduced. Parts of the thoughts on these challenges have been presented in [Ste16; Ste17a]. A first formal description of KGs has been recently introduced in [Ste+18].

Chapter 4 introduces an architectural extension of XCS to allow for incorporating interpolation. The architectural variant has been initially presented at a doctoral symposium [Ste14]. It was later published as a conference contribution [Ste+16b], and finally as an extended journal version [Ste+17]. The same contributions also form the basis for the interpolation-based approaches *Action Selection Integration* (ASI), *Covering Intialization Integration* (CII) and *Offspring Initialization Integration* (OII) which are going to be introduced in Chapters 5, 6, and 8.

Parts of the details, insights and results regarding the novel CII and OII strategies as will be introduced in Chapters 6 and 8 are furthermore based on [Ste+16a].

In Chapter 7, particular aspects of the introduced techniques are adapted and evaluated in the context of a real world self-adaptive urban traffic light control scenario. This chapter is based on the work published in [Ste+16c].

The idea of the interpolation-based prediction modeling has been initially published in [SMH18], on which the corresponding Chapter 9 is mainly based.

At the end of this thesis, Chapter 10 provides an elaborate introduction to two future research directions: (1) An integration of *Active Learning* (AL) concepts with *XCS Classifier System* (XCS) (Sect. 10.1). (2) The proposition of a new methodology called of *Proactive Knowledge Construction* in Section 10.2. This chapter's contents are based on the author's publications [SMH17] and [Ste+18].

[1]For the detailed references, please refer to the author's publication list at the beginning of this document.

Chapter 2.

Background & Prerequisites

The following sections introduce the theoretical as well as the contextual background of the present thesis. Starting with the aims and scope of the research initiative *Organic Computing*, basic paradigms and concepts from the domain of *Machine Learning* are subject of discussion. Subsequently, the discipline of *scattered data interpolation* which originates from the more general field of approximation theory is briefly introduced, followed by an elaboration of selected interpolation methodologies. The biggest part of this chapter constitutes the thorough introduction of Michigan-style *Learning Classifier Systems* in general, and of the *XCS classifier system* in particular. This chapter lays the necessary foundation in order to allow for the adequate comprehension of the extensions that will be developed throughout this work.

2.1. Organic Computing

Back in 2002, researchers from Germany started thinking about a new research initiative to meet the predicted increasing demands for the steadily rising complexity in computing systems which could be observed over the last decades. *Organic Computing* (OC) [MT17a] was born and the ball was set rolling for a major research project which has shaped today's profile of OC. Originating mainly from the *Systems Engineering* domain, OC was deemed a *paradigm shift* for the engineering of complex systems which are largely interconnected and purposefully designed to fulfill critical tasks in real world environments [MSU11]. The continuing rise in complexity was recognized as the "nightmare" of system engineers and administrators [KC03], since the multitude of interacting (sub-)systems would make it nearly impossible to control the overall system complex by the same means as done in the past. On the one hand, the envisaged paradigm shift and, thus, the overarching goal of OC can therefore be found in moving decisions that have been conventionally felt at design-time to the systems' runtime in order to allow for robust and flexible systems. This, however, involves the acceptance of the circumstance that not all possible environmental condition and resulting system states can be anticipated at the time of system development (the *design time*) by the human engineers. Accordingly, the autonomy of the individual subsystems needs to be increased by endowing them with the ability to learn from experience made with the task environment. This comprises the observation of system states and a subsequent decision making regarding appropriate reactions to be realized during the system's runtime. On the other hand, this paradigm shift demands for a deeper understanding of the collective behavior of the interacting subsystems as a whole – which is expected to be governed by a single central authority any longer. The autonomous entities shall be able to organize themselves via local interactions and on the basis of their own local knowledge about the tasks for which they are deployed. In order to achieve the envisaged shift, a strong focus was set on leveraging the potentials of *self-organization* as can be observed in *nature*. Inspired by social and natural sciences, OC mainly postulates that complex tasks can be attempted most effectively by collectives of (semi-)autonomous systems (or agents), in such a way as has been demonstrated by nature myriad times. This demands for different angles to look at the involved systems, i.e., from macroscopic and microscopic perspectives which define the respective contexts. An overarching OC system which is observed from a macroscopic viewpoint has to be aware of its own environmental and situational context. However, it still comprises several sub-systems or lower level hierarchies of them, each bearing its individual microscopic (or mesoscopic) context. More details on organizational aspects of OC systems can be found in [MT17b]. Nonetheless, what each abstraction view of an OC system has in common is an individual context which shapes the problem space to which a learning entity is exposed. Following [MT17b], it is distinguished between *single context-aware adaptive systems* and *multi context-aware adaptive systems*, which can be be further divided into *open collective* and *goal-oriented holonic* systems. For the sake of simplicity, the focus is set on the first category of single context-aware adaptive systems. However, the techniques and concepts developed in this thesis are not deemed to be restricted to this category. As stated before, each abstraction level brings its individual context which spans the problem space that is to be learned. This context may change over time and, thus, feeds the risk of occurring *knowledge gaps*, simply at another level of abstraction.

OC systems have been envisioned to exhibit *life-like* characteristics making them viable in dynamic real world environments – more generally referred to as NSEs. A number of so-called *self-x properties* have been mentioned to obtain the pursued life-like behavior, among which the capabilities of

- Self-*organization*
- Self-*adaptation*
- Self-*optimization*
- Self-*configuration*
- Self-*protection*
- Self-*healing*
- Self-*explanation*, and naturally
- Self-*learning*

have been explicitly formulated as expected attributes with which full-fledged OC systems should be endowed. These properties are not necessarily mutually exclusive but rather overlap. For instance, self-configuration can be seen as part of self-adaptivity both of which realizable via self-learning mechanisms. The methodology introduced in this thesis can be deemed as being mainly targeted at the self-learning property, though, but also affects the self-explanatory possibilities of OC systems.

A very basic OC system usually follows the so-called generic *Observer/Controller* (O/C) architecture as introduced in [Ric+06; Tom+11b]. The architectural blueprint of this reference model is detailed in Chapter 3.1.

Over the years, a variety of OC systems have appeared in several application domains. For example, the self-adaptive traffic management system referred to as *Organic Traffic Control* (OTC) [Pro+11] was among the first systems developed in the scope of OC research. An *Organic Robot Control Architecture* (ORCA) [ALM08] which equips a six-legged robot with self-healing mechanisms to make it fault-tolerant and acting more life-like is another example. Bernauer and Zeppenfeld et al. proposed a two-stage method implementing a lightweight LCS to combine the advantages of software and hardware solutions in [Ber+11; ZH11]. During the first stage, the approach harnesses the adaptivity of a simplified LCS to learn a basic control strategy for a given problem based on simulation. In a next step, the learned rule set is transferred to a piece of high-performing hardware called "Learning Classifier Table". At this point, the learned production rules are not updated any longer. This concept has been further refined and applied to an *Autonomic System on a Chip* (ASoC) to achieve a self-management of CPU workload [Zep+11]. For more examples and a summary of the achievements the Organic Computing initiative has spawned until 2011, the reader is referred to [MSU11].

Nearly at the same time when OC emerged, *Autonomic Computing* (AC) [KC03] was founded by IBM to deal with similar problems. Here, the motivation originated from a more application-oriented and industry-driven view such that a first focus was set on the autonomic management of large-scale data and computing centers. AC was mainly inspired by the autonomic nervous system. The so-called *self-CHOP* (i.e., *-configuration, -healing, -optimization, -protection*) properties were initially defined. Today, the OC and AC communities share common scientific conferences and mostly pursue similar research goals.

Beside OC and AC, the *Self-Adaptive and Self-Organizing* (SASO) systems community has established as an umbrella for general research in this area [BS07]. Annual conferences on this overarching topic have been organized since 2007. During the the 2019 edition it was announced that the international conferences on SASO and on AC will be finally merged into a single conference on *Autonomic Computing and Self-Organizing Systems* (ACSOS)[1]. The focus of SASO is to bring in line the concepts and insights of *Self-Organization* in biological systems [Cam+03] with research

[1]http://acsos.org/ (last accessed August 7, 2019)

on technical SAS in order to cope with the explicit and implicit interconnectedness of (sub-)systems and the resulting complexity. Also the principles and foundations of collective system behavior are in the spotlight. Thus, the SASO community seeks for foundations and a principled approach to understanding and building complex systems – a perfect match for OC research.

The study of *Complex Adaptive Systems* (CAS) [Hol92] is another strongly related field strongly propelled by John H. Holland and colleagues since the 1960s [Hol62]. Holland's initial attempt to the challenge of CAS was the invention of the today well-known *Genetic Algorithm* (GA) [Hol75]. He saw a viable solution to complexity due to highly changing system structures in Darwin's theory of evolution and the principle of the *survival of the fittest*. Holland was also the researcher who set up the stage for decades of LCS research. Together with Reitman, he was the one who proposed the first ERBML system – the *Cognitive System* (CS-1) [HR78]. Thus, also LCS themselves have their roots in research on complex adaptive systems, what in turn underpins the reasonableness of choosing a LCS technique as the subject of investigation in this thesis.

Beside AC, SASO, CAS, further overlapping research fields such as *Cybernetics* preceded OC, or rather emerged at nearly the same time as OC did as is the case for *Proactive Computing* [Ten00]. For more details in this regard, the interested reader is referred to the chapter on the "Major Context" in [MT17b].

Obviously, the demand for building autonomous SAS in order to cope with the complexity of upcoming computer-supported technical systems has been recognized nearly two decades ago – and still there exist a lot of aspirations in that respect. This thesis is elaborated sharing the mindset of OC and, thus, adheres to the definitions and insights spawned since its foundation. Please note that this is not about denying the right to exist of any of the complementary initiatives mentioned before. Rather, it is about mainly sticking to the terminology and generic architectures stemming from OC, which will be picked up again in Section 3.1.

In OC, AI technology is considered a key enabler and essential ingredient to build SAS with an increasing degree of autonomy. Intelligence is often connected with the *ability to learn*, as is also done in this thesis. It is differentiated between (1) *Self-Adaptive Systems* (SAS) which are rendered adaptive by means of integrating a priori defined production rules that enable the system to reconfigure as a response to changing conditions anticipated at design time, and, (2) SLAS which build up and further evolve their knowledge bases (i.e., self-learned production rules) which determine their adaptation strategies during the runtime.

Most research aspirations that attempt to build in intelligence into SAS make strong use of methods from the AI domain, more specifically of its probably most important subfield these days – *Machine Learning* (ML). In the next section, basic terms and concepts of ML are briefly sketched and the methods and approaches used in this thesis are brought in line with the overarching field.

2.2. Machine Learning

The field of Machine Learning has emerged as the study of design and analysis of algorithms that allow machines, or programs, to learn from data. It is considered as a branch of AI. Historically, the investigations mainly focused on intelligent agents that learn to act in their living environments by continuing interaction, i.e., it was attempted to mimic natural intelligence observed in animals or humans. Over time, however, ML evolved into a much broader research field mainly concerned with the improvement of some learning entity in terms of a performance measure on a specified task. The assumption is, that when exposed to more and more past data, the learner will get more experienced with regard to its task reflected by higher performance measures. Due to achieved advances over the last decades, the focus has shifted from mimicking human or animal intelligence toward exploiting the revealed potentials of ML algorithms on tasks where the cognitive capabilities of humans reach their bounds. The exploration of large amounts of structured or unstructured data collected over a certain time period constitutes an example. Automatically recognizing patterns in complex data or leveraging the prevalent correlations to induce a predictive model can be done much more efficiently – or even achieved at all – by ML and *data mining* algorithms. *Computational Learning Theory* is concerned with understanding the fundamental working principles of ML algorithms and whether a given problem can be provably solved (cf. e.g., [MRT12]). Bounding the expected errors of ML algorithms given a certain amount of data on the other hand is subject of *Statistical Learning Theory* [Vap98].

As already briefly discussed in Chapter1, a common and concise definition of ML is provided by Mitchell [Mit97]:

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*

This definition is chosen because it is abstract enough to comprise all paradigms of ML that have emerged over the years, but also sufficiently simple to provide an intuitive understanding of the core principle.

In the following paragraphs, the most prominent subfields of ML which are deemed relevant for this work will be briefly discussed. Subsequently, the scope of the techniques and concepts that will be developed in this thesis is brought in line with these categories. The categorization and the according descriptions are mainly adopted from seminal introductory books on ML [Mar09; SB14; Alp10; SB98].

Supervised Learning

When reading about the latest advances in ML, the articles often refer to achievements with regard to the category of *Supervised Learning* (SL). This ML paradigm is about learning from a vast number of training examples. These are data instances which are annotated, i.e., provided with the correct target value, and made available by a "supervising" entity a priori. In the era of *Big Data*, large collections of digital records do not constitute an issue anymore for many problems. This fact accompanied by the achievements in computing efficiency paved to way for the rise of *Artificial Neural Networks* (ANNs) and their descending *Deep Neural Networks* (DNNs). The term *supervised* is inspired by assuming a *teacher* (or domain expert) that provides annotated data comprising the *ground truth* upon which the algorithms shall build a *hypothesis*. In that way, the teacher is supervising the algorithm in a certain sense. This is also referred to the *inductive* way of learning, i.e., inferring from the particular (i.e., available training cases) to a general rule (the hypothesis or model). Advances in large-scale computing due to *General Purpose Graphical Processing Units* (GPGPUs) and the leveraging of *single instruction multiple data* instruction sets have opened the possibility of highly efficient computation of statistical learning models which is often referred to as *vectorization*. ANNs are the most prominent example of algorithms that highly profit from this technological development.

In SL the task can be formally described as follows: Given a set D of m labeled (or annotated) training instances of the form

$$D = \{(\vec{x}_i, y_i)\}_{i=1\ldots m}$$

where $\vec{x}_i \in X$ constitutes a vector from a *feature space* X comprising descriptive features[2] and $y_i \in Y$ is a target value out of a *target space* Y assigned to the i-th training instance. The task of a SL algorithm is then to find a hypothesis, or else a predictive model based on the training data

$$h(\vec{x}) : X \to Y,\ \vec{x} \mapsto y,$$

which maps the feature space onto the target space.

This trained model is then supposed to be applied to new, unseen data instances which are not contained in D. To train such an algorithm, the available training instances in D are typically split up into a dedicated *training set* D_{train} and a hold out or *test set* D_{test}. The latter is used to evaluate the predictive performance of the obtained model $h(\vec{x})$. Usually, the feature space X is spanned over n dimensions,

[2]The space of inputs has various notations in the ML literature. Sometimes it is referred to as *state* or *situation* space S, *feature* space X or simply called *input* space. In this thesis the notations X and S are used interchangeably in order to adhere to the typical notations used in the current context's literature. See Appendix A for an overview of notations used in this work.

where each dimension $j = 1 \dots n$ refers to another descriptive feature of a data instance $d \in D$ in the training sample. For example, the salary, the age and the civil status would be descriptive features for an automated credit status decision. The entirety of features is also referred to as *covariates* or *independent variables.* To induce the desired hypothesis, the algorithms often have to learn, or rather optimize, a number of *model parameters* θ. These learning parameters are sometimes also referred to as *weights* or *coefficients* in the literature. However, they should not be confused with an algorithm's *hyperparameters*, which are – at least in the usual sense – not subject to learning. Rather they need to be configured a priori by the human expert applying the ML algorithm. Nevertheless, research branches such as *hyperparameter optimization* or *Automated Algorithm Configuration* [HNT+17] exactly deal with this subproblem.

The most prominent tasks in SL are known as *classification* and *regression.* In classification, the *target* variable (or *dependent* variable) is categorical. This means that there is no inherent order in the possible targets $y \in Y$. It also implies discreteness. Technically, one would usually encode $Y \subset \mathbb{N}$ to represent the classes in a program. Subcategories of classification are: (1) *Binary classification*, where only two classes $Y = \{0, 1\}$ exist (e.g., a tumor can be *benign*$= 0$ or *malignant*$= 1$). (2) *Multi-class classification*, where $|Y| > 2$, meaning that there are more than two categories or classes. (3) *Multi-label classification*, where each data instance can be assigned multiple classes (or labels), i.e., $Y = \mathcal{P}(Y) \setminus \emptyset$ (e.g., genres of a song in a music library). Many of the theoretical insights assume the first subcategory of binary classification, however, the transfer to the other categories is often straightforward. Typical applications of SL techniques are for instance, credit card fraud detection, spam filters, medical diagnosis, or customer churn prediction. Further prominent results reported since the rise of DNNs come from the application to the task of *object recognition* in images.

Regression, on the other hand, refers to predicting a scalar (continuous) target value $Y \subseteq \mathbb{R}$. As for classification, the applied models typically need to capture non-linear relationships between the covariates and the target variables. Exemplary application scenarios would be the prediction of house prices, forecasting of energy consumption, or even stock trading. Another application domain, which also serves as benchmark scenario in this thesis, is the more abstract task of *function approximation.* Here, an unknown function $f(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ is to be reconstructed on the basis of a number of available *samples* $(\vec{x}_i, f(\vec{x}_i))$ of that function.

Unsupervised Learning

In contrast to the SL task, in *Unsupervised Learning* (UL) scenarios the data instances are not annotated, i.e., no ground truth $y_i \in Y$ exists. This means that the task shifts from learning a mapping of an input $\vec{x} \in X$ onto a target variable $y \in Y$, to finding regularities, correlations or similarities in the available data. This is accomplished only on the basis of the n features $x_i^{(j)}$ $(j = 1 \dots n)$ of the available data instances in $D \coloneqq \{\vec{x}_i\}$. Revealed structures can then be used to reduce the dimensionality of the feature space, a task called (*dimensionality reduction*), or to figure out the most influencing or irrelevant features by applying *feature selection*. *Clustering*, i.e., automatically grouping instances in order to form "clusters" of similar data instances constitutes another task achievable with this paradigm. UL techniques for instance can be incorporated to preprocess the available data in order to reveal certain patterns or to reduce the data complexity for a more efficient processing by the applied algorithms. In this thesis, UL takes a back seat. Nevertheless, in Section 10.2, it will be made use (at least conceptually) of Kernel Density Estimation, which can be regarded as kind of a Statistics means to UL.

Semi-Supervised & Transductive Learning

Between SL and UL, a further category called *Semi-Supervised Learning* (SSL) [HS13] can be found in the literature. Here, it is assumed that the available data can be split up into a set of labeled instances $\mathcal{L}$ and a set of remaining unlabeled data instances $\mathcal{U}$ such that $D = \mathcal{L} \cup \mathcal{U}$. The underlying task is still a SL one. However, with the implicit assumption that it is beneficial for the algorithms' predictive performance to further incorporate the unlabeled data instances in $\mathcal{U}$ during the construction of the model. At this point, the term *Transductive Learning* (TL) is introduced, which stays in contrast to the *inductive* means of learning as mentioned before. The term TL or transductive inference is due to Vapnik [Vap98; GVV98]. Vapnik's motivation is to omit the intermediate step of learning a general model via induction which he deems as solving a more general problem in the first place. His transductive approach is to directly solve the actual problem, i.e., assigning a target value to a formerly unseen data instance (i.e., not present in $\mathcal{L}$), via using only the particular data examples available in D – regardless of being labeled or not. *Transduction* can therefore be regarded as a third category complementing induction (inferring a general rule from particular cases) and deduction (concluding from a general rule to a specific case). TL and SSL are closely related. However, the motivations differ quite a bit [CSZ06].

Case-based Reasoning

In analogy to human reasoning, the field of *Case-based Reasoning* (CBR) deals with analogies found in previous cases, i.e., experiences regarding certain problems and their corresponding solutions [AP94]. Also known under the name *instance-based reasoning*, CBR approaches are centered around a so-called *case base* (or case memory) which can be queried in case of new situations for which an ML algorithm is asked to obtain an appropriate output. A typical CBR cycle comprises four phases: (1) *retrieve* the most similar case from the case memory. (2) *reuse* the determined 'most similar' case for solving the current problem instance at hand. (3) *revise*, i.e., adapt the memorized case to better fit to current and future situations. Finally, (4) *retain* the adapted case or parts of it by storing it as new memory in the case base if it is deemed useful for the future. As becomes apparent, CBR strongly relies on the assumption of *similarity* in its reasoning process. Instead of building a model in an inductive manner and deducing solutions for new cases from the model, in CBR conclusions are drawn directly from memorized experiences. This, in turn, is reminiscent of transductive inference as discussed above and as stated by Gammerman et al. in [GVV98]. In general, the term CBR does not refer a certain set of algorithms directly, but can be understood as a general working principle for reasoning based on past experiences. It is not clearly prescribed how cases have to be represented. This can be either by simple data instances in terms of feature vectors and target variables, or by rather rich knowledge representations such as subsumptions of similar cases or ontologies. However, this thesis shares the common mindset with CBR in that the direct utilization of past experiences are deemed highly beneficial in terms of accelerating early phases of learning where an accurate model can usually not be assumed so far.

Active Learning

Conventional SL algorithms can be thought of learning in a *passive* manner. They process incoming data instances either provided online or by existing training sets and attempt to learn a corresponding hypothesis. They take the data as it is and do not reflect on whether some data instances might be of higher value than others. In the field of AL [Set09], the learning algorithm is extended with the capability to query a so-called *oracle* for providing the correct label of a selected data instance, or at least a confident guess about it. The oracles are typically expected to be human experts which are now explicitly taken into the learning loop. According to Settles' survey [Set09], three main categories of AL can be found in the literature: (1) *Pool-based AL* strongly adheres to the SSL mindset. Also a small set of labeled $\mathcal{L}$ and a disjoint set of numerous unlabeled instances $\mathcal{U}$ is assumed. The rationale behind considering unlabeled data is the assumption that the acquisition of unlabeled examples from a given learning task comes at far lower costs. In contrast, the annotation by a human expert is expected to bind a huge amount of monetary resources. Thus, an AL algorithm is enabled to actively seek data instances from $\mathcal{U}$ that contribute most to effectively reducing the prediction error of the SL algorithm. (2) *Stream-based AL* differs from the former approach in terms of the decision when to query the oracle. Here the decision is felt for each incoming data instance individually at the time it appears for the first time. Thus, in this setting, the SL algorithm is expected to learn in an incremental or online manner. That is, roughly speaking, building a model *instance by instance* from a continuing stream of incoming data (cf. to the corresponding section below). (3) *Query Synthesis* constitutes the third category. It is concerned with generating artificial data instances from the feature space X

and to subsequently query the oracle for the target value. This methodology was found to bear the risk of potentially obscuring human oracles, since the synthesized data instances might be either irrelevant or, even worse, implausible [BL92]. As an extension to the AL framework, in [DC08] Donmez and Carbonell introduced *Proactive Learning* (PAL). With PAL, a new ML branch is proposed that aims to overcome certain limitations of the original AL setting. Among these, the restriction to only one oracle, the omniscience and absence of oracle reluctance, as well as the assumption of equal costs for all queries are explicitly focused. Even if the name conveys the impression of a "proactive" learning behavior, Donmez's and Carbonell's perspective on proactive learning is rather different from the perspective which is to be conveyed in this thesis' Chapter 10.

Reinforcement Learning

Besides SL and UL, usually a third overarching category of ML appears in most overview articles and books – the field of *Reinforcement Learning* (RL). In this paradigm, a learning *agent* exists and acts within an *environment* in which it has to learn a certain task via *interactions*. Following the principle of *trial-and-error*, this agent learns how to "survive" by successively performing *actions* from an action space $a \in A$ to move within or modify the environment in other ways. This leads to a transition of the current *state* s_t to the next state s_{t+1}, where all possible states are from a *state space* S. The agent continually perceives the current state in each step. I then always decides on the next action based on the currently observed state s retrieved from the environment. Depending on the appropriateness of the realized action, the agent receives a so-called *reward* which can be positive (payoff) or negative (penalty). It reinforces or discourages the agent to perform this particular action again with respect to the observed state, respectively. A conventional RL agent always acts goal-oriented. That is, it tries to maximize the received cumulative reward over time. Thus, the *reward signal* implies the degree of the agent's goal achievement due to its actions. Typical tasks for RL agents are (sequential) control problems. Consider the previously introduced running example: An intelligent traffic light controller (agent) monitors the incoming traffic flows (states) and adapts the green phases of the traffic lights (actions) installed at an urban intersection (environment). The task can either be modeled as single-step mode-switching task, i.e., a simple selection of another signal plan as a response to the current traffic situation. Or as sequential control task, where the green times of the traffic light are successively increased or decreased instead of exchanging the entire signal plan (cf. Ch. 7 for a brief introduction to traffic terminology).

From an RL perspective, a control problem is modeled by means of a *Markov Decision Process* (MDP). A deterministic MDP can be defined by:

1. A state (or situation) space S,

2. An action (or configuration) space A,
3. A state transition function $\tau : S \times A \rightarrow S$,
4. A reward function $r : S \times A \rightarrow \mathbb{R}$.

It is called "Markov" since it is assumed that the environments dynamics, i.e., state transitions and rewards, depend only on the immediately preceding state and action. A state is regarded as to fulfill the *markov property* if it includes all information about past agent-environment interactions that make a difference for the future. [SB98] For the sake of simplicity, the explanations are restricted to the deterministic case here. This implies that the state transitions and rewards are not subject to stochastic influences and always evaluate to the same values. In non-deterministic settings, the transition function would be defined as probability distribution $\tau(s_t, s_{t+1}, a_t) = P(s_{t+1}|s_t, a_t)$ over the possible states s_{t+1}, given the current state s_t as well as the executed action a_t. In the same fashion, the reward function would be defined as $r(s_t, a_t) = P(r_t|s_t, a_t)$ from which the expected values can then be calculated. The task of an RL algorithm (here called agent) is to figure out an *optimal policy* $\pi^* : S \rightarrow A$ on the basis of subsequent interactions with the environment. A policy determines an agent's action selection strategy. Depending on its internal working principle, it tries to select those action that maximizes the expected return in the long run. An often pursued approach is then that each state-action pair gets assigned a quality value $Q(s, a)$ which has to be estimated over time.

Generally, it can be differentiated between *single-step* problems with only one single reward received after deciding for a particular action. Afterward the problem ends, and a new instance is presented to the agent. This mostly resembles SL tasks, where no sequence of state-action decision is necessary to solve a given problem. For instance, a classification can be learned by providing a reward of 1000 if the selected action (e.g., a class) was correct for a given state (e.g., a patient data record) and 0 if not. The more common type are *multi-step problems* (or sequential decision problems) modeled by the abovementioned MDPs. Here the difficulty arises due to the so-called *credit assignment problem.* Until a certain goal is reached (e.g., an agent attempting to exit a maze), possibly only rewards of 0 are payed out. In case of reaching the goal, finally a positive reward is received. For the latter category, the Q values are estimates of the expected value of the cumulative reward obtainable until the end of the learning task. If tasks have a particular goal to be achieved (e.g., maze exit), have a specified end due to a maximum time limit (e.g., restricted battery capacity), or where an agent can fail (e.g., a pole balancing scenario where the pole falls over) this implies a so-called *episodic* task. In the case of infinite duration (or at least undefined or unknown endings) tasks are regarded as being *continuous.* An example was already given above. The traffic light control scenario is a continuous task with no predefined ending. The concept of *discounting* by a factor $\gamma \in (0, 1)$, eventually realizing a geometric series, needs to be incorporated to guarantee finite cumulative reward sums.

With these two ingredients, the agent can iteratively learn a policy $\pi(s)$ to decide which action to choose to maximize the expected cumulative reward in the long run, on the basis of the current Q value estimates $\hat{Q}(s, a)$. The possibly most prominent algorithm to achieve this is Q-learning [WD92].

RL approaches can be further divided into *model-free* approaches such as Q-learning or SARSA [SB98], which try to find a policy without building explicit models of τ and r. A second category are the *model-based* approaches, as realized by the DYNA approach [Sut91], for instance. Explicit models of the environment and its dynamics are build via function approximation. Afterward, based on the learned models the optimal policy is determined via e.g., planning approaches. At this place, the mixture of the RL with the SL paradigm becomes apparent.

As Sutton and Barto state in [SB98], another key aspect that distinguishes RL from other ML paradigms is the existence of the so-called *exploration vs. exploitation dilemma*. This dilemma captures the fact that the agent has to learn the quality of actions in certain states by means of *trial and error*, i.e., by exploring the effects of different actions in the same state. This naturally involves performing inadequate actions which results in penalties from the environment. This trade-off is also of very high relevance for OC and thus for this work. OC systems are expected to comply to exploration boundaries guaranteeing functional safety, which stays in contrast to freely exploring the problem space via trial-and-error.

Offline, Incremental and Online Learning

A further distinction can be made with regard to the time when the data becomes available to the ML algorithms. In a typical SL setting, the data is assumed to be available a priori in form of a labeled data set D, which is then split up into a training portion and a test portion (and possibly a further validation portion) for evaluation purposes. This kind of learning is also called *batch learning*, because the training data is available as a whole batch and can be feed at once to the algorithms. As the data is already available at the design time, it is also called *offline learning*. The applied algorithms can be trained with data on which preceding preprocessing operations were performed, and only the final trained model is deployed at runtime when the system acts *online*, i.e., with actual influence on the productive system.

At the other end of the spectrum, the *Online Learning* (OL) setting can be found (cf. [CC06; OR01; NCK11]). In this setting, the data arrives *instance-by-instance*. Accordingly, the algorithms have to update their models on-the-fly. There exist variants which further require that each data instance must only be considered once and then thrown away. With that, the memory costs for storing samples shall be kept at a minimum. This is called *one-pass learning*. In *Online Machine Learning* (OML), there is no clear separation between training and testing instances. Therefore, a so-called *test-the-train* or *prequential* learning strategy is pursued in order to allow the algorithms to learn from made experiences. This means that each time a new data instances is presented to the algorithm, it first tests its current hypothesis on it, then compares the output with the true target value, and afterward incrementally adapts (i.e., trains) the model for this particular instance. It should be noted that not the entire model is supposed to be completely retrained based on all data instances collected over time. This would rather be regarded as *lazy learning* or *instance-based learning* in the literature, where a ML algorithm postpones the model building step until a new data item upon which a prediction is to be made arrives. The *k-nearest neighbor* algorithm constitutes an instance of this learning approach. Clearly, the lazy learning notion is reminiscent of what previously was introduced as transductive inference. However, it depends on the working principle of the lazily performed prediction step. When a model is induced at the postponed time of prediction, this again contradicts the TL philosophy.

A synonym for OL which can be found in the *Data Mining* domain is *Data Stream Mining.* OML algorithms reveal their power in settings where an a priori acquisition of training data is too expensive or simply unfeasible – maybe because the system is too complex to anticipate all possible situations offline at the design time. This kind of ML approach is also viable for learning in NSEs. Due to the inherent dynamics of the underlying concepts and data generating processes the ground truth might gradually drift or abruptly shift over time. This circumstance gave rise to the formulation of the *stability-plasticity dilemma* [Dit+15]. It captures the trade-off between the *forgetting* of older and possibly obsolete knowledge in favor of newly constructed knowledge as a response to the most recent observations. Forgetting older knowledge might turn out to be detrimental as soon as the situations which where covered by the deleted knowledge parts occur again. A related term that mainly focuses on the plasticity side of the aforementioned dilemma can be found under the term *catastrophic forgetting.* This issues becomes even more important when the memory for realizing an OML algorithm is limited. In the era of the *Internet of Things* this might often be the case in view of the small and special-purpose embedded systems brought out.

When facing Big Data scenarios, the vast amount of available records is too large to be computed at once within the limited random access memory of most computers. Therefore, a further category between the extremes of batch and online learning exists which is called *incremental learning.* Here, the strict requirement for one-pass and instance-by-instance learning is attenuated. Incrementally learning algorithms are assumed to learn from small increments of data, also referred to as *mini-batches*, which are provided via streams of incoming data. The size of these mini-batches can be fixed or else variable. The employed algorithms are typically batch learning algorithms enhanced with the capability to adapt to new data without the necessity to be completely retrained from scratch. Accordingly, the learning appears to be incremental. ML algorithms that make extensive use of data increments in order to overcome the aforementioned stability-plasticity dilemma can be found in the branch of *Ensemble Learning* (cf. e.g., [Pol+01; DP13].

In the scope of this thesis, a learning technique that builds its knowledge base via OL is in the spotlight. However, the strict requirement for one-pass learning will be attenuated for the method introduced in Chapter 9, since naturally interpolation needs a set of sampling points. Nevertheless, always only a small subsample of all the data instances presented to the algorithm so far is stored and gets continually updated in order to also consider the stability-plasticity dilemma.

Imbalanced Learning

The difficulty of learning problems exhibiting drifting data distributions which in turn feed the stability-plasticity dilemma can be further increased by prevalent imbalances in the data. A well-recognized issue for the classical SL task of classification is the so-called *class imbalance problem*. The existence of class imbalances is a well-studied issue in the domain of ML [Cha05; HG09; Wei04]. Class imbalances occur when the available data set for training an SL algorithm can be divided into so-called *majority* and *minority* classes. A majority class is represented by far more data instances than the minority class what causes sparseness of samples from the latter. This leads to overestimates in the model regarding the majority class and consequently negligence of instances from the other. Most learning algorithms assume balanced training data which naturally leads to accuracy losses in the presence of class imbalance. A lot of research has been conducted to cope with this issue. (Re-)Sampling approaches such as *oversampling* of the minority class and *undersampling* of the majority class are popular methods to recreate a balance in the available data set (see [HG09] for a comprehensive survey).

Weiss [Wei04] classifies imbalanced data into the following categories: (1) *rare classes*, and (2) *rare cases*. The former issue corresponds to the well-understood but still important class imbalance problem. The latter issue on the other hand points to the sparseness resulting from the distribution of the data instances over the input space. That is, some feature vectors $\vec{x}$ appear seldom so that the algorithm has difficulties to learn and thus to generalize appropriately – clearly a cause for knowledge gaps. A sort of combination of both issues is termed *small disjuncts* [Lóp+13]. Those can be understood as rare cases that are surrounded by instances from the majority class. There exist a variety of approaches to alleviate the negative effects caused by imbalanced data. For instance, the aforementioned (re-)sampling methods [WMY15; Cha05]. A more sophisticated approach, however, is introduced by Chawla et al. in [Cha+02]. The *Synthetic Minority Oversampling Technique* (SMOTE) synthesizes new minority class instances on a straight line between two actually received instances from that underrepresented class. A comprehensive survey on dealing with imbalanced data is provided by Lopez et al. in [Lóp+13]. Open challenges such as regression in imbalanced scenarios, multi-class problems bearing imbalances as well as online learning from imbalanced data streams are reviewed by Krawczyk in [Kra16]. A rather recent research issue is the challenge of *online class imbalance learning* (OCIL) [NCK11; WMY13; WMY15]. It addresses the issue of how to cope with class imbalances when no finite set of training data is available at design time which precludes a preprocessing by means of resampling. When data instances arrive one-by-one, no statistical distribution analysis can be conducted at the design time. This renders it difficult for the OML algorithm to generalize appropriately without neglecting the minority class. Thus, OCIL combines the challenges of class imbalances with the specific issues OL bears.

Since this thesis focuses on dealing with KGs at the system's runtime, for which data imbalances constitute a possible cause (cf. Sect. 3.1), the techniques presented in the following chapters also constitute countermeasures against the OCIL problem.

Evolutionary Machine Learning

As a last category of ML algorithms, the class of *Evolutionary Machine Learning* (EML) is mentioned in order to reflect its high relevance in this thesis. EML studies ML algorithms that generally leverage the power of *Evolutionary Algorithms* (EAs). Kovacs in [Kov12] reviews the field and provides a comprehensive introduction to *Genetics-based Machine Learning* – a designation that is superseded by the term EML today. EAs have proven very efficient in complex black-box optimization tasks. These are tasks where no information about the structure or gradients of the underlying function under consideration exists. As population-based *metaheuristics*, EAs rely on probabilistic genetic operators such as selection, recombination and mutation. With those operators, specific pressures are exerted to the population of individuals (i.e., possible solution candidates) that guide the evolution over generations. *Selective pressure*, *genetic drift*, the *Baldwin effect*, etc. All are concepts inspired by evolution theory that need to be considered and can be leveraged to obtain effective and efficient optimization performance. In contrast, non-evolutionary ML algorithms mainly make use of gradient-based optimization techniques, such as the popular *gradient descent* algorithm, in order to find optimal coefficients for the predictive models to be learned. A downside of EAs and probabilistic metaheuristics in general is the fact that it cannot be guaranteed that the global optimum will be found. In many practical applications, however, a near optimal solution is often sufficient. Especially in highly complex objective functions, e.g., with a high degree of multi-modality, algorithms that solely rely on gradient information show deficiencies in terms of escaping local optima. The underlying objective functions are also referred to as *fitness functions* in the *Evolutionary Computation* (EC) context. This is where EAs reveal their strengths. They do not rely on gradients but rather on fitness values of individuals alive in the current generation which can then be captured by the evolutionary process following the *survival of the fittest* principle. The population of individuals can be considered as moving through the search space. Furthermore, under the presence of concept drift affecting the surface of the fitness function, evolution is deemed a viable candidate to adequately deal with that issue. Leveraging the aforementioned strengths of EAs and EC in general in the context of designing competent learning algorithms constitutes the research goal of EML. In EML, EC techniques can be applied to nearly any subproblem of ML, such as coefficient optimization, hyperparameter configuration, ensemble formation, etc. A huge variety of algorithms emerged in the last decades and they have found to perform competitively compared to conventional, non-evolutionary approaches [OCB08]. Evolutionary variants of many algorithms have been devised, among them: (1) Evolved ANNs, also found under the umbrella of *Neuroevolution* [Yao99; Sta+19]. (2) *Genetic Fuzzy Systems* [Her08]. (3) *Evolutionary Ensemble Machines* [SC06]. But also idiosyncratic techniques emerged. For instance *Genetic Programming* [Ban+98] and Learning Classifier Systems [LSW00]. The latter initiated a thoroughly investigated subcategory of EML which is still under active research and also is in the spotlight of the present thesis – *Evolutionary Rule-based Machine Learning* (ERBML).

Integration of this Thesis into the Scientific Landscape of ML With this classification of ML algorithms at hand, eventually the methods that will be developed in this thesis can be ranged in:

It will be demonstrated how XCS, an ERBML algorithm, can be extended with interpolation and AL techniques. XCS working in a single-step RL manner, which can be mapped to a SL setting, constitutes the main subject of investigation. Due to the fact that XCS is an OML technique, a priori available batches of data instances are not assumed. Empirical results corroborating the beneficial effects of the enhanced XCS will mostly be obtained by the application to the tasks of online classification as well as online regression for function approximation purposes.

Furthermore, the idea of transductive learning plays an important role in this thesis. Even if the concepts and approaches introduced by Vapnik et al. (cf. e.g., [GVV98]) are not adopted directly, techniques that interpolate new knowledge elements from collected experiences or already existing elements in the knowledge base are going to be developed. This way of knowledge construction can be regarded as transductive. New knowledge, which is aimed to predict on particular, yet unseen cases, is directly inferred from existing knowledge elements for similar cases without inducing a model first. For instance, instead of making a detour over assessing the overall solution to infer the target value for a new case (deduction), particular cases accommodated within existing knowledge elements are directly used for the prediction (cf. Ch. 9).

In the last part of this work, Chapter 10, the concept of AL takes a central role. XCS is extended toward an *Active Learning Classifier System* via adopting pool-based and query synthesis techniques.

Last but not least, XCS belongs to the class of Michigan-style LCS, an ERBML technique that makes strong use of *Genetic Algorithms* (GAs). As will become clear in Section 3.4, the inherent evolutionary dynamics of XCS, and EML systems in general, is considered a key ingredient to deal with non-stationary nature of SLAS.

2.3. Multivariate Scattered Data Interpolation

The general problem formulation for *interpolation* can be defined as follows:

Definition (Interpolation problem)**.** *Given a limited set of so-called sampling points $SP := \{s_i = (\vec{x}_i, f(\vec{x}_i))\}_{i=1 \ldots m}$, find a continuous function $\tilde{f} : X \to \mathbb{R}$ that maps a so-called query point $\vec{x}_q = (x_q^{(1)}, \ldots, x_q^{(n)}) \in X$ from an unknown function's domain (or input space) $X \subseteq \mathbb{R}^n$, to a function value $\tilde{f}(\vec{x}_q)$ that ideally equals the real function value $f(\vec{x})$ of the unknown function f such that $\tilde{f}(\vec{x}_i) = f(\vec{x}_i)$ holds.*

The desired continuous interpolation function is sometimes referred to as *interpolant*. The known sampling points s_i are represented by 2-tuples $s_i = (\vec{x}_i, f(\vec{x}_i))$ each comprising a *sampling point coordinate* $\vec{x}_i = s_i^{(1)}$, as well as an associated true function value $f(\vec{x}_i) = s_i^{(2)}$. The latter is abbreviated by f_i in the following. In contrast to the more general task of *approximation*, previously observed or otherwise available sampling points $s_i \in SP$ are not just taken into account for fitting a parametric model which attempts to minimize a particular error between the approximated and the true function value. As stated in the definition above, for interpolation, the available sampling points must be exactly passed by $\tilde{f}(\vec{x})$. Therefore, the additional constraint $\tilde{f}(\vec{x}_i) = f(\vec{x}_i)$, $\forall s_i \in SP$ must hold.

Figure 2.1 depicts this distinction:

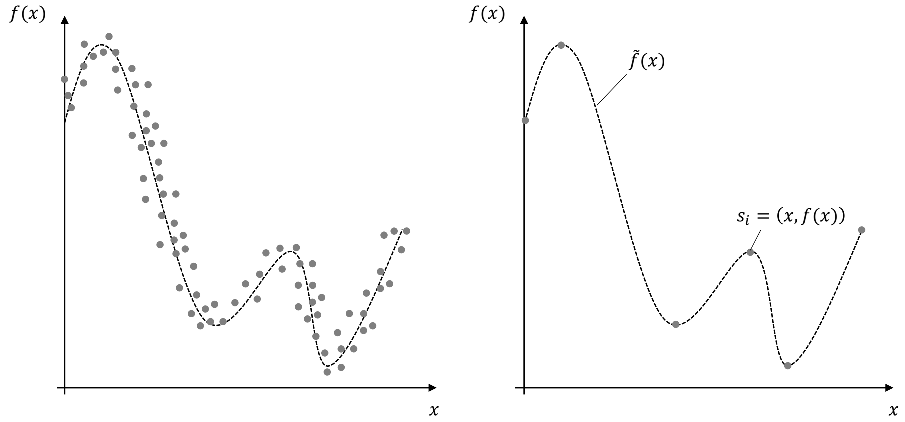


FIGURE 2.1.: Intuition of the difference between approximation (left) and interpolation (right), following [SK11]

In this thesis, the focus is set on *multivariate interpolation*, where the domain of the functions are of dimensionality $n > 1$, i.e., $X \subseteq \mathbb{R}^n$. This naturally restricts the available methods that can be considered in this work. For the interpolation of a query point's function value, not any available sampling point needs to be taken into consideration. Accordingly, interpolation methods can be roughly divided into two categories [Fra79]: (1) *Global methods*, that consider each of the available sampling points for interpolating the function value $\tilde{f}$ of the query point $\vec{x}_q$. (2) *Local approaches*, that only incorporate sampling points in a certain proximity of $\vec{x}_q$.

Another distinction can be made between *grid-based* and *scattered data* interpolation. The former category assumes that the sampling points are structurally organized in a regular grid what allows for the exploitation of mathematical conveniences. In this work, however, only the second category can be applied reasonably. This is because the acquisition of the sampling points constitutes a part of the self-learning system which is invariant and thus can hardly be actively biased. This aspect will become more clear in Chapter 4.

All methods considered in this thesis have in common that they are (at least implicitly) distance-based. More precisely, the influence of each considered sampling point is directly related to the distance to the query point $\vec{x}_q$ within the usually real-valued input space X. This is a common approach to deal with scattered sampling points. Nevertheless, these techniques are usually straight-forwardly applicable to sampling points that are organized in a grid.

In the following, selected interpolation techniques from different classes which are used in this thesis will be described. Some are based on triangulation, others simply need to calculate Euclidean distances between the sampling points and the query points The most sophisticated interpolation method considered here relies on *Radial Basis Functions* (RBFs). At the end of this section, a discussion about the suitability of the introduced techniques is given.

2.3.1. Direct Neighborhood-based Interpolation

Neighborhood-based interpolation techniques belong to the class of local methods, since they rely solely on sampling points s_i in the direct proximity of the query point $\vec{x}_q$. In contrast, global methods use all available sampling points what results in an increased (re-)calculation complexity. Global methods are thus in general more sensitive to an increasing number of sampling points, where this is not directly the case for neighborhood-based variants [LH10]. Therefore, two representatives of the category of neighborhood-based interpolation methods – *Nearest Neighbor* (NeNe) interpolation and *Natural Neighbor* (NaNe) interpolation – are first described.

Voronoi Diagrams

The basis of both NeNe and NaNe forms a so-called *Voronoi tessellation* and its geometric dual the *Delaunay triangulation.* Figure 2.2 illustrates this duality relationship. Voronoi tessellation is a technique for partitioning the entire input space X into a unique graph which consists of a finite number of *Voronoi cells* determined on the basis of the sampling points $\vec{x}_i = s_i^1$. The resulting graph is called *Voronoi diagram.* Each Voronoi cell appears as a convex polyhedron within an n-dimensional metric space. For instance, the Euclidean space with its Euclidean metric derived from the 2-norm.

According to de Berg, a Voronoi cell can be defined as follows [Ber+08]:

Definition (Voronoi cell)**.** *For each sampling point s_i, its associated Voronoi cell $\mathcal{V}_{s_i}$ comprises all points $\vec{x} \in X$ that are closer to its coordinate $\vec{x}_i$, than to any other sampling point's coordinate $s_j^{(1)} = \vec{x}_j$.*

$$\mathcal{V}_{s_i} := \left\{ \vec{x} \in X \;\middle|\; \mathbf{d}\left(\vec{x}, s_i^1\right)_2 \leq \mathbf{d}\left(\vec{x}, s_j^1\right)_2, \forall j \neq i \right\} \tag{2.1}$$

Here, $\mathbf{d}\,(x, y)_2 = \|x - y\|_2$ denotes the *Euclidean distance* between two arbitrary vectors x and y. Each time a new site s_i^* is added to a Voronoi diagram, a new cell $\mathcal{V}_{s_i}^*$ appears. Thus, a Voronoi diagram can be expressed as the set of all Voronoi cells build upon a set of sampling points SP:

$$\mathrm{VD}_{SP} := \{\mathcal{V}_{s_i}\}_{i=1\ldots|SP|} \tag{2.2}$$

For the two-dimensional case and provided that all m sampling points are known a priori, a complete Voronoi diagram can be constructed in $O(m \log m)$ time and $O(m)$ space by using Fortune's algorithm [For86]. Due to the inherent dynamics of NSEs which results in the necessity for OL or at least incrementally working algorithms, this assumption does usually not hold for the application within SLAS. To cope with such a continually changing set of sampling points SP, knowledge about the duality relationship (depicted in Fig. 2.2) is essential for the algorithmic implementation of the concrete interpolation techniques which will be described in the next two paragraphs. For more details about this relationship, please refer to [Ber+08; LH10].

Naïve Approach: Nearest Neighbor Interpolation

The NeNe interpolation method searches for the coordinate $\vec{x}_i$ of the i-th sampling point s_i that has the shortest distance to the current query point $\vec{x}_q$. It then assigns the associated function value f_i to the queried data point $\vec{x}_q$. More formally, this can

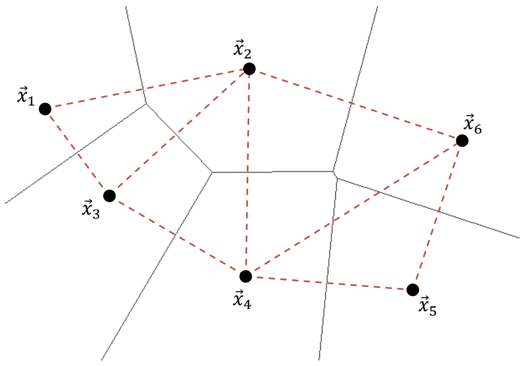


FIGURE 2.2.: Duality relation between Voronoi tessellation (black solid lines) and Delaunay triangulation (red dashed lines)

be expressed as follows (remember that the first component of a sampling point's tuple is its coordinate, i.e., $s_i^{(1)} = \vec{x}_i$):

$$s_{nearest} = \underset{s_i \in SP}{\operatorname{argmin}} \; \mathbf{d}\left(\vec{x}_q, \vec{x}_i\right)_2 \tag{2.3}$$

Thus, $s_{nearest}$ is the sampling point whose coordinate $\vec{x}_i$ minimizes the distance to the query point $\vec{x}_q$. The actual interpolation is then performed by assigning

$$\tilde{f}(\vec{x}_q) := f_{nearest} = f(\vec{x}_{nearest}) = s_{nearest}^{(2)}. \tag{2.4}$$

Given a Voronoi diagram of the available sampling points $s_i \in SP$, the NeNe interpolation would first identify the Voronoi cell which contains the query point $\mathbf{x}_q$. In second step, exactly the function value of the site s_i that determines the cell $\mathcal{V}_{s_i}$ would be assigned as stated above. In the planar, or 2-dimensional case, the identification of the $\mathcal{V}_{s_i}$ encompassing the query point can be achieved in $O(logm)$ time, where m is still denoting the number of sites in the sampling point set SP (cf. [Ber+08]). However, this efficiency holds only true for an already constructed static Voronoi diagram and for input dimensions up to $n = 2$.

This naïve approach results in a step-wise and non-smooth interpolant $\tilde{f}$. Intuitively, the function value assigned by Equation 2.4 does not change as long as a query point $\vec{x}_q$ finds its minimal distance to another sampling point. The computational complexity of this interpolation method depends on the realization of the point location challenge. This is also known as the *nearest neighbor search problem.* An exhaustive search would calculate the distance to any available sampling point s_i and returns the one with the smallest distance to $\vec{x}_q$. This results in a linear computation complexity, i.e., $O(m)$, assuming a constant query time for each dimension. However, more efficient algorithms for exact nearest neighbor search, even for dimensions higher than $n = 2$, exist. Examples are *BSP-trees* or *kd-trees* which can solve the nearest neighbor location problem with sub-linear computational time [Ber+08]. However most often these methods come at the expense of super-linear space complexity which grows exponentially with the number of dimensions [AI18].

Natural Neighbor Interpolation

In contrast to NeNe, the NaNe technique not only considers the closest sampling point $s_{nearest}$ for the interpolation of a query point $\vec{x}_q$, but particular surrounding s_i. More precisely, in a given Voronoi diagram VD_{SP}, all sampling points $s_j \in SP$ sharing an edge with the Voronoi cell $\mathcal{V}_{s_i}$ of a specified sampling point $s_i \in SP$ are taken into account. These s_j are called *natural neighbors* of s_i. For example, in Figure 2.2, the natural neighbors of site $\vec{x}_4$ are $\vec{x}_2, \vec{x}_3, \vec{x}_5$ and $\vec{x}_6$.

In terms of the dual Delaunay tessellation, the natural neighbors can be defined as all vertices which share a common edge with a reference site s_i [LH10].

Definition (Natural neighbors)**.** *Let s_i be a sampling point under consideration and $\mathcal{V}_{s_i}$ the corresponding Voronoi cell, then according to [BU08], the set of natural neighbors is given by:*

$$SP_{s_i}^{NaNe} := \left\{ s_j \in SP \;\middle|\; \mathcal{V}_{s_j} \cap \mathcal{V}_{s_i} \neq \emptyset \right\} \subset SP \tag{2.5}$$

Based on the notion of natural neighbors, Sibson in the 1980's devised the so-called *Sibson weights* or else *natural neighbor coordinates* which he used for smooth interpolation purposes [Sib81]. The interpolation based on the natural neighbors works as follows:

As a first step, a new so-called *second-order Voronoi cell* $\mathcal{V}_{\vec{x}_q}^{+}$ is virtually added to the Voronoi diagram VD_{SP}. This second-order cell can be interpreted as a new cell that would appear, if $\vec{x}_q$ became a new sampling point s^{+}. By means of this virtual addition, potentially overlapping areas with the first-order Voronoi cells $\mathcal{V}_{s_i} \in \mathrm{VD}_{SP}$ can be identified. The volumes of these overlapping areas, normalized by the volume of the second-order cell $\mathcal{V}_{\vec{x}_q}^{+}$ exactly determines the Sibson weights or natural neighbor coordinates (cf. [Sib81; LG04]).

Definition (Sibson weight)**.** *Let $Vol(\mathcal{V})$ denote the volume of a Voronoi cell, or a polytope that resulting from the intersection $\mathcal{V}_{s_i} \cap \mathcal{V}_{\vec{x}_q}^{+}$ of a first- and the second-order Voronoi cell. Then a Sibson weight can be calculated by:*

$$w_{s_j}^{\vec{x}_q} = \frac{Vol(\mathcal{V}_{s_j} \cap \mathcal{V}_{\vec{x}_q}^{+})}{Vol(\mathcal{V}_{\vec{x}_q}^{+})} \tag{2.6}$$

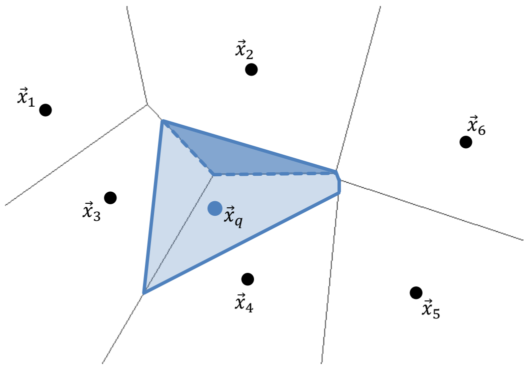


FIGURE 2.3.: Natural Neighbor Interpolation – Formation of overlapping areas (also Sibson Weights) by virtual insertion of second-order Voronoi cell (blue-rimmed area)

In Figure 2.3 this relationship is illustrated. The blue-rimmed convex polygon shows the second-order cell $\mathcal{V}^{+}_{\vec{x}_q}$. The area highlighted in dark blue depicts the specific overlap area for the natural neighbor s_2 with its coordinate $\vec{x}_2$.

Having determined the Sibson weights $w^{\vec{x}_q}_{s_j}$, in a second step the interpolation is realized by calculating the weighted sum of the function values f_j corresponding to the natural neighbors of $\vec{x}_q$. Equation 2.7 summarizes this calculation step. It essentially consists of a convex combination of the function values corresponding to the potential natural neighbors of $\vec{x}_q$.

$$f(\vec{x}_q) = \frac{\sum_{s_j \in SP^{NaNe}_{\vec{x}_q}} w^{\vec{x}_q}_{s_j} \cdot f(s_j)}{\sum_{s_j \in SP^{NaNe}_{\vec{x}_q}} w^{\vec{x}_q}_{s_j}} \tag{2.7}$$

The NaNe technique leads to a smooth interpolant that is continuous except at the given sampling points [Bob+09]. Another advantage is the implicit adaptation to different sampling point densities within the domain (or input space) X, which is directly captured by the Voronoi diagram. Most of the Voronoi diagram calculations can be carried out on the dual Delaunay graph which allows for a more efficient computation. For implementation details of this algorithm the reader is referred to [LG04; LH10; SBM95; Sib81].

2.3.2. Shepard's Methods

The next class of interpolation methods is due to Shepard [She68]. Both NeNe as well as NaNe interpolation rely on the neighborhood of sampling points in SP. Thus they only implicitly consider distances among the known and the queried data points. Back in 1968, Shepard has already proposed an approach similar to NaNe interpolation. The similarity becomes apparent by having a look at the general way of formulating the interpolant as a linear combination of the sampling points' function values and corresponding weights that here explicitly consider the distances to the query point.

Inverse Distance Weighting

Shepard's first approach falls in the class of global interpolation techniques. Thus, all available sampling points $s_i \in SP$ are taken into account. He defined the basic form, also called *Inverse Distance Weighting* (IDW), as follows [She68]:

$$\tilde{f}(\vec{x}_q) = \frac{\sum_i w_i \cdot f_i}{\sum_i w_i} \tag{2.8}$$

with $i = 1 \ldots |SP|$ and the weights w_i defined by

$$w_i = \left(\frac{1}{\mathbf{d}\left(\vec{x}_q, \vec{x}_i\right)_2} \right)^p \tag{2.9}$$

Thus, the weights constitute the inverse distance of the query point $\vec{x}_q$ to the i-th sampling point's coordinate $\vec{x}_i$. The power parameter p controls to which degree the weights w_i decrease when the distance to $\vec{x}_q$ increases and vice versa. Essentially, the IDW method interpolates any query point $\vec{x}_q \in X$ by calculating a weighted average of the already known function values f_i. Hence, it is theoretically not restricted to any dimensionality as is the case for the NaNe technique which requires the calculation of n-dimensional polytope volumes.

The method is therefore straightforward to implement. Considering the time complexity of IDW, the calculation effort grows linear with the number m of available sampling points. It has been recognized that too much influence is assigned to sampling points located relatively far away from the actual query point (cf. e.g., [DD16]). In order to overcome this issue, modifications have been proposed in the literature [She68; FN80]. The most essential one will be presented in the subsequent section. Another disadvantage directly identified by Shepard in his original article is the so-called *flat-spot* problem, also mentioned in related studies [GW78; MT03; DD16]. In the succeeding section this aspect is revisited.

Modified Shepard's Method

The name *Modified Shepard's Method* (MSM) is an umbrella term that summarizes various extensions of Shepard's original method. One of the most distinguishing aspects is the enhancement toward a local interpolation scheme. To transform Shepard's original method into a local approach, the weight calculation has to be modified. Franke and Nielson comprehensively developed such a modification in [FN80].

In order to overcome the aforementioned issue of IDW that too much influence is attributed to sampling points far away from the query point, according to Shepard [She68], simply not all available sampling points $s_i \in SP$ should be considered. To restrict the set of considered s_i, a radius R_w is defined, which serves as a spherical boundary in a sense that only sampling points within the resulting hypersphere are assigned non-zero weights. More precisely, only those sampling points s_i satisfying $\mathbf{d}\left(\vec{x}_q, \vec{x}_i\right)_2 \leq R_w$ are taken into account and actually impact the interpolated value $\tilde{f}(\vec{x}_q)$.

Accordingly, the weights for local Shepard interpolation are calculated following Equation 2.10 (adopting the notation of [Ren88]):

$$W_i(\vec{x}_q) = \left[\frac{\left(R_w - \mathbf{d}\left(\vec{x}_q, \vec{x}_i\right)_2\right)_+}{R_w \cdot \mathbf{d}\left(\vec{x}_q, \vec{x}_i\right)_2} \right]^2 \tag{2.10}$$

$R_w > 0$ defines the radius within which sampling points are considered for interpolation. Formally, this is guaranteed by applying the function $(\cdot)_+ := \max\{0, \cdot\}$. With this modification, Shepard's interpolation now constitutes a *local-support* method.

An adequate choice of R_w is crucial for an effective application of MSM. If the radius is chosen too large, MSM possibly degenerates to a global method reminiscent of IDW. On the other hand, if it is chosen too small, maybe too few sampling points are considered for an accurate interpolation. To relieve the user from choosing an appropriate radius, Franke and Nielson [FN80] proposed to automatically determine the radius R_w dependent on a hyperparameter N_w. This parameter determines the desired number of sampling points that should actually be used for the interpolation. Assuming sampling points that are uniformly distributed over X, a suitable radius is given by:

$$R_w = \frac{D}{2}\sqrt{\frac{N_w}{m}}, \tag{2.11}$$

where $D = \max_{i,j} \mathbf{d}\left(\vec{x}_i, \vec{x}_j\right)_2$ and m still denotes the number of all available sampling points. The size of N_w is, however, still a hyperparameter which needs to be carefully configured and it is highly dependent on the dimensionality of the function's domain. Thacker et al. provide a summary of suggestions for the right choice of N_w in [Tha+09]. The adaptive choice of R_w is considered valuable for the aims pursued in this thesis – especially for problem spaces where the bounds are not known a priori. An additional advantage is that the restriction to a certain subset of sampling points reduces the computational effort for the interpolation.

The second modification thoroughly investigated in the literature, is the replacement of the constant terms f_i from Equation 2.8 by *inverse distance weighted least square approximations*, also called *nodal functions*, denoted $Q_i(\vec{x})$ in the following. This treatment shall wipe out the recognized shortcoming referred to as the *flat-spot* problem the interpolant bears at the sampling points s_i. Shepard stated that at these locations the interpolant has zero directional derivatives, which he deemed an undesirable and arbitrary constraint on the interpolation surface [She68]. To retain the interpolation property of $\tilde{f}$ the constraint $Q_i(\vec{x}_i) = f_i$ must hold. For instance, considering the $n = 2$ dimensional (i.e., bivariate) case, Franke and Nielson defined $Q_i(\vec{x}_q)$ as a quadratic polynomial:

$$\begin{aligned} Q_i\left(x_q^{(1)}, x_q^{(2)}\right) = a_{i,1}\left(x_q^{(1)} - x_i^{(1)}\right) + a_{i,2}\left(x_q^{(2)} - x_i^{(2)}\right) + a_{i,3}\left(x_q^{(1)} - x_i^{(1)}\right)^2 + \\ a_{i,4}\left(x_q^{(1)} - x_i^{(1)}\right)\left(x_q^{(2)} - x_i^{(2)}\right) + a_{i,5}\left(x_q^{(2)} - x_i^{(2)}\right)^2 + f_i \end{aligned} \tag{2.12}$$

The above notation can be interpreted as roughly resembling a Taylor series up to the second order about the point $\vec{x}_i$, where the coefficients $a_{i,j}, j = 1 \ldots 5$ resemble the partial derivatives with respect to the independent variables $x_q^{(1)}$ and $x_q^{(2)}$ (cf. [Tha+09; DD16]). The constant term f_i at the end of the quadratic given in Equation 2.12 ensures the fulfillment of the constraint mentioned above. This is true since all differences within the expression evaluate to zero whenever the query point equals the i-th sampling point's coordinate.

To figure out a nodal function that reconstructs the underlying function f as good as possible, Franke and Nielson proposed to again select a subset of the available sampling points to retain the local-support property of the overall interpolant. Analogously to the selection of the sampling points that are considered for the linear combination in Equation 2.8, a radius R_p and the corresponding weights W_i^p are determined as follows:

$$R_p = \frac{D}{2}\sqrt{\frac{N_p}{N}} \tag{2.13}$$

$$W_i^p(\vec{x}_k) = \left[\frac{\left(R_p - \mathbf{d}\left(\vec{x}_k, \vec{x}_i\right)_2\right)_+}{R_p \cdot \mathbf{d}\left(\vec{x}_k, \vec{x}_i\right)_2}\right]^2 \tag{2.14}$$

The index k refers to those sampling points that are considered for the optimization. Again, N_p is the desired number of sampling points that shall take part in the optimization step. Based on the sampling points that fall within the sphere determined by R_p, the following optimization problem needs to be solved to obtain the inverse distance weighted least square approximation, Q_i, of f around $\vec{x}_i$.

For all $k = 1 \ldots m$

$$\min_{\vec{a}_k} \sum_{\substack{i=1 \\ i \neq k}}^{m} W_i^p(\vec{x}_k)[Q_k(\vec{x}_i) - f_i]^2 \tag{2.15}$$

Thus, coefficients $\vec{a}_k = (a_{k,1}, a_{k,2}, \ldots, a_{k,l})$ are seeked that minimize the squared difference between the local approximation around $\vec{x}_i$, i.e., $Q_i(\vec{x}_k)$, and the actual function value f_i, for all sampling points $\vec{x}_k$ that fall inside the sphere (determined by the radius R_p) multiplied with resulting weights $W_i^p(\vec{x}_k)$. The number l of coefficients $a_{k,j}$ depends on the degree of the polynomial d which was chosen to approximate f locally, as well as on the dimensionality n of the sampling point coordinates. More precisely $l = \left(\binom{n+d}{d} - 1\right)$. The reduction by one is due to the constant term fixed at f_i in Q_i.

Several implementations of MSM exist and have been published [Ren88; BM99; Tha+09]. Each of these methods describes the use of the local inverse distance weighted approximations $Q_i(\vec{x})$ of the underlying function f. Variations for $Q_i(\vec{x})$ realized as polynomials of the second or third degree for two or three-dimensional sampling points have been proposed. Berry and Minser extend Renka's implementation to work with up to five-dimensional data in [BM99]. For this thesis, methods are searched which are not restricted to the dimensionality of the input space X in general. MSM interpolants indeed can be used for arbitrary-dimensional data when the use of the local approximations are omitted. An alternative pointed out by Thacker et al. in [Tha+09] would be to apply $Q_i(\vec{x})$ to be a first-degree polynomial. Regardless of which local approximation is used, the number of available sampling points plays an important role. In general, to find an approximation via a d-degree polynomial for n-dimensional data, the optimization problem of Equation 2.15 involves finding $\binom{n+d}{d}$ coefficients. To achieve this, at least the same number of sampling points need to be available. For instance, to interpolate an $n = 10$ dimensional function with MSM using nodal functions $Q_i(\vec{x})$ realized by a polynomial of degree $d = 3$, a total of $\binom{10+3}{3} = 286$ coefficients have to be fitted and 286 sampling points (supposed to be "nearby" to still remain a local method) are needed. This is often infeasible and the computational minimization costs are considerably high (depending on the employed optimizer). Therefore, Thacker et al. propose to either choose the constant values of f_i as initially proposed by Shepard. Or to revert to linear functions which only demand $n+1$ samples for the coefficient optimization step. For the scope of this thesis, these suggestions of Thacker et al. appear to be plausible and will be followed.

2.3.3. Radial Basis Function Interpolation

The use of *Radial Basis Function* (RBF) interpolation [Buh03] is another prominent means for approaching the multivariate scattered data interpolation problem. The assumption is that the function to be interpolated can be reconstructed by a linear combination of certain *basis functions* $\phi(\cdot)$ that exhibit certain characteristics. More formally, this can be expressed by the following equation:

$$\tilde{f}(\vec{x}_q) = \sum_{i=1}^{m} \alpha_i \Phi_i(\vec{x}_q) \tag{2.16}$$

with $\vec{x}_q \in \mathbb{R}^n$ and α_i being scalar coefficients from the reals. Thus, the basis of the linear space of all interpolants is chosen to be by $\{\Phi_i\}_{i=1\ldots m}$, where each Φ_i constitutes a shifted variant of a *basic* function Φ. More precisely, $\Phi_i(\vec{x}_q)$ can be written as $\Phi(\vec{x}_q - \vec{x}_i)$ which makes the "shift" more obvious. With that definition, the reason why it is called *basis* function should be clarified. This technique, however, also contains the term "radial" in its name.

A basis function Φ is said to be radial, when it satisfies

$$\Phi(\vec{x}) = \phi(||\vec{x}||), \qquad \vec{x} \in X \subseteq \mathbb{R}^n,$$

where $\phi : [0, \infty) \to \mathbb{R}$ is univariate and $|| \cdot ||$ constitutes a norm on a vector space such as the Euclidean norm $|| \cdot ||_2$ on $\mathbb{R}^n$.[3] Accordingly, the value of $\Phi(\vec{x})$ solely depends on the radius. Put another way, it is spherically symmetric about its center which is given by its argument $\vec{x}$.

According to the more general notion of calculating a linear combination of certain basis functions in order to build an interpolant (see Eq. 2.16), the most basic RBF interpolation model can be formulated as follows:

$$\tilde{f}(\vec{x}_q) = \sum_{i=1}^{m} \alpha_i \phi(||\vec{x}_q - \vec{x}_i||_2) \tag{2.17}$$

Again, $\vec{x}_q$ constitutes the unknown query point for which the function value shall be interpolated. And the i-th sampling point's coordinates are denoted by $\vec{x}_i$. So far, only the sampling points' coordinates $\vec{x}_i$ have been used in the RBF interpolation model, but not the corresponding function values $y_i = f(\vec{x}_i)$. At this point, the coefficients α_i come into play. These coefficients are determined by enforcing the interpolation constraints $\tilde{f}(\vec{x}_i) = f(\vec{x}_i)(= y_i), i = 1 \ldots |SP|$ and then solving the resulting system of linear equations $D\vec{\alpha} = \vec{y}$. This linear equation system can be written in matrix form as follows:

$$\underbrace{\begin{pmatrix} \phi(||\vec{x}_1 - \vec{x}_1||_2) & \phi(||\vec{x}_1 - \vec{x}_2||_2) & \cdots & \phi(||\vec{x}_1 - \vec{x}_m||_2) \\ \phi(||\vec{x}_2 - \vec{x}_1||_2) & \phi(||\vec{x}_2 - \vec{x}_2||_2) & \cdots & \phi(||\vec{x}_2 - \vec{x}_m||_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(||\vec{x}_m - \vec{x}_1||_2) & \phi(||\vec{x}_m - \vec{x}_2||_2) & \cdots & \phi(||\vec{x}_m - \vec{x}_m||_2) \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{pmatrix}}_{\vec{\alpha}} = \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}}_{\vec{y}} \tag{2.18}$$

[3]In the remainder, the value of the norm will sometimes be simply written as r denoting the "radius", i.e., $\phi(|| \cdot ||) = \phi(r)$.

Here D turns out to be a symmetric $m \times m$ (recall $m = |SP|$) matrix, which can be interpreted as *distance matrix* (also called kernel matrix in the ML context). It is defined by $D := (\phi(||\vec{x}_i - \vec{x}_j||_2))_{1 \leq i,j \leq m}$ and $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)^T$ is a vector representing the coefficients to solve for. The right hand side of the linear system, $\vec{y}$, is a vector containing the actual function values $f(\vec{x}_i)$ of all m available sampling points $s_i \in SP$. In that way, the linear equation system is uniquely solved for the coefficients $\vec{\alpha}$ which, incorporated in the interpolation equation, perfectly reconstruct all sampling points $s_i \in SP$.

One obvious fact is that the quality in terms of the interpolation error strongly depends on the choice of the (radial) basis functions. The solvability of the linear equation system given in Equation 2.18 is a crucial issue when the choice for the RBF to be used is made.

There exist a variety of often used RBFs. In this background chapter at least the two important classes of *(strictly) positive definite (radial) functions* and *(strictly) conditional positive definite (radial) functions* will be briefly introduced.

It is well-known that a real symmetric $N \times N$ matrix A is called *positive semi-definite* if its quadratic form is non-negative for its coefficients $\vec{c} \in \mathbb{R}^N$ (cf. [Fas07]), i.e.,

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j A_{i,j} \geq 0.$$

This property can be extended to positive definiteness if the aforementioned quadratic form becomes zero only when the coefficient vector $\vec{c}$ is the null vector. If the distance matrix D is positive definite, then it follows that it is non-singular, i.e., uniquely solvable. This follows from the fact that for positive definite matrices their eigenvalues are positive. [Fas07]

One can obtain a positive definite distance matrix D when the selected RBF $\phi(\cdot)$ belongs to the class of strictly positive definite functions.[4] With this in mind, one could argue that it is straightforward to only incorporate RBFs of this particular type into the Equation 2.17 in order to yield a well-posed interpolation problem. However, there exist a variety of very useful functions that unfortunately do not meet this requirement. These functions belong to the second class of *conditionally positive definite (radial) functions*.

[4]As can be seen, the property of *positive definiteness* occurs at two interrelated sites – at the distance matrix site, and the RBF function site. Here, only the importance and intuition of why the aspect of positive definiteness is relevant for RBF interpolation is outlined, since a complete review of the mathematical details would clearly go far beyond the scope of this thesis. For the exact mathematical definitions of (strictly) positive (semi-)definite functions and further theory regarding scattered data interpolation by means of RBFs, the reader is referred to e.g., [Fas07; CHS05; Buh03].

When functions of this kind shall be used, the system of linear equations from 2.18 has to be extended to assure non-singularity. The extension consists of adding a polynomial term to Equation 2.17 as follows:

$$\tilde{f}(\vec{x}_q) = \sum_{i=1}^{m} \alpha_i \phi(||\vec{x}_q - \vec{x}_i||_2) + \sum_{j=1}^{M} \beta_j p_j(\vec{x}_q) \tag{2.19}$$

The number M is the number of monomial terms of the polynomial to be added. It calculates as $M = \binom{d-1+n}{d-1}$. Thus, M also constitutes the dimension of the linear space of n-variate polynomials of degree less than d, denoted Π^n_{d-1}. The basis B of Π^n_{d-1} is given by $B = \{p_1, p_2, \ldots, p_M\}$. By extending the interpolation model $\tilde{f}$ in that way, not only the prerequisite for non-singularity of the resulting linear equation system is obtained. Also another mathematical finesse can be recognized. Given that the set X of sampling point coordinates from SP is $d-1$-unisolvent, then polynomial precision on X is guaranteed. This means that when the sampling points as well as the unknown query points stem from an unknown polynomial of degree at most $d-1$, they will be exactly fitted by the interpolant [Fas07].

Since the extended system now has $|SP| + M$ degrees of freedom, but so far only $m = |SP|$ conditions, namely $\tilde{f}(\vec{x}_i) = y_i,\ i = 1, \ldots, m$, additional M side constraints have to be included in the formulation of the system. Therefore, the coefficients α_i are further constrained by the subsequent conditions:

$$\sum_{i=1}^{m} \alpha_i p_j(\vec{x}_i) = 0, \quad j = 1, \ldots, M \tag{2.20}$$

Accordingly, the linear equation system now reads

$$\begin{pmatrix} D & P \\ P^T & O \end{pmatrix} \begin{pmatrix} \vec{\alpha} \\ \vec{\beta} \end{pmatrix} = \begin{pmatrix} \vec{y} \\ \mathbf{0} \end{pmatrix}, \tag{2.21}$$

where $A, \vec{\alpha}, \vec{y}$ are the same as for the previous formulation. P is a $m \times M$ matrix comprising the M basis polynomials p_j for each sampling point $\vec{x}_i$ with entries $P_{i,j} = p_j(\vec{x}_i)$ with $i = 1, \ldots, m$ and $j = 1, \ldots, M$. P^T is the transpose of P and O is a $M \times M$ zero matrix. $\vec{\beta} = (\beta_1, \ldots, \beta_M)^T$ is the vector of the polynomial coefficients. As can be seen in the system, $\vec{\alpha}$ is multiplied with P^T and solved for the zero vector $\mathbf{0}$ which ensures the incorporation of the additional M side conditions of Equation 2.20.

Unique solvability of the resulting system of linear equations can be guaranteed when the following requirements are met:

1. The applied RBF $\phi(\cdot)$ is strictly conditionally positive definite of order d on $\mathbb{R}^n$.

2. The sampling point coordinates $\vec{x}_i$ are distinct and together they form a $(d-1)$-unisolvent set.

An example of such a required *Strictly Conditional Positive Definite* (SCPD) RBF is the so-called *Thin-Plate-Spline* (TPS) basis function defined by

$$\phi_{TPS}(r) = r^2 \log r \tag{2.22}$$

which is of order $d = 2$. According to [Du 08] and [Fas07], the need for unisolvency is a rather weak requirement. For the case of the TPS which has the order $d = 2$ and a bivariate interpolation problem (i.e., $n = 2$), the coordinates $\vec{x}_i$ of the sampling points $s_i \in SP$ must not lie on a straight line, since otherwise a polynomial of degree $(d-1) = 1$ interpolating these collinear points can not be uniquely determined. d-unisolvency with regard to a set X requires that the only polynomial p of degree at most d which interpolates zero data on X is the zero polynomial itself. Put another way, given unisolvence, this means that there must exist a unique polynomial $p \in \Pi^n_{d-1}$ of lowest possible degree that interpolates the sampling points. $(d-1)$-unisolvency can also be assumed on the $\vec{x}_i$ when the $m \times M$ matrix P has full column rank, i.e., the columns are all linearly independent.[Fas07]

An example for a strictly positive definite radial basis function is the *Gaussian* RBF, which is defined by

$$\phi_{GAUSS}(r) = e^{-(\epsilon r)^2} \tag{2.23}$$

with ϵ denoting a so-called *shape* parameter where lower or higher values determine how flat or peaked the bell-shape will appear, respectively. An even simpler case where it is known that the resulting distance matrix D becomes non-singular would be to let ϕ be simply the Euclidean norm $||\vec{x}_q||_2$. Reducing further to the absolute value function $|\vec{x}_q|$ and substitute this function into Equation 2.16 yields a piece-wise linear interpolation.[Fas07]

2.3.4. Discussion on Suitability

In this section, various types of scattered data interpolation techniques have been introduced. It was explicitly focused on methods that allow for an interpolation of multivariate functions $f : \mathbb{R}^n \rightarrow \mathbb{R}$ from a set of sampling points with coordinates $\vec{x} \in X \subseteq \mathbb{R}^n$ without any regularity in their positions. This methods are called *mesh-free* or *scattered data* interpolation techniques.

For the methods as will be developed in this thesis, especially local interpolation techniques are considered valuable. They can reduce the additional computational effort in comparison to global methods.

In general, the introduced interpolation methods vary quite a lot regarding their computational complexity. NeNe interpolation essentially requires to deal with the point location problem of nearest neighbor identification, which can be accomplished in logarithmic time on average, i.e., $O(\log m)$. Despite being very efficient, the interpolation surface appears to be rough and staircase-shaped. This might cause higher errors when the query point falls between two available sampling points. Smoothing can be achieved by incorporating not only the single nearest neighbor, but also the k-nearest neighbors in order to average their function values.

The NaNe technique constitutes another methodology that takes more than one sampling point into account. It is based on the direct neighborhood, introduced as the set of natural neighbors. This technique is based on the Voronoi diagram and most of the necessary calculations are based on its dual form, i.e., the Delaunay tessellation. This fact imposes limitations related to computational geometry. In dimensions higher than $n = 3$, the computations of convex polytopes or n-simplexes become extremely complicated [LG04] and computationally expensive. Since general methodologies are seeked which can be applied to problem spaces of arbitrary dimension, in this thesis the NaNe method is mostly neglected, except of the very first experimental evaluation conducted in Chapter 5.

The next type of interpolation techniques that have been discussed – Shepard's methods – provide a well-applicable alternative free of the abovementioned limitation on the problem dimensionality. Furthermore, they allow for both global and local interpolation schemes. Shepard's initial approach, also called IDW, considers any available sampling point. Although, this leads to higher computational demands compared to naïve nearest neighbor methods, it still fall in the rather efficient class of $O(m)$. However, the IDW method was found to attribute too much influence to sampling points relatively far away from the queried position. This deficiency can be alleviated with the more sophisticated MSM approach, which constitutes one of the first local interpolation approaches. Here, only a subset of the available sampling points is used for interpolation, eventually leading to sub-linear computational complexity for the actual interpolation step. Therefore, however, a predefined number of surrounding samples which lie within a dynamically calculated radius has to be identified from SP. This in turn increases the overall computational complexity depending on the incorporated identification techniques and data structures (such as space partitioning trees for instance). In order to overcome further limitations of the original method such as the flat spot problem, it has been proposed to add polynomials of certain degrees to the interpolation function. This however comes at the expense of solving an approximation problem via least square methods. Such an approximation can be expensive especially when matrix inversion operations (complexity up to $O(N^3)$ when N denotes number of rows/columns) are involved. More specialized methodologies that store and remember previous intermediate results (such as matrix inversions) are possible when a fixed SP set can be assumed. Since in this thesis an online learning setting, where continuing changes due to the non-stationary nature of the environment are set in the spotlight (cf. Ch. 3), the aforementioned assumption unfortunately does not hold true in the general case.

Even the nodal function's coefficients that need to be optimized, basically would have to be recalculated whenever SP changes. This limits the applicability of these improvements to the techniques that will be developed in the remainder of this work.

The last class of mesh-free interpolation techniques that has been described is RBF interpolation. Next to the MSM approach with added polynomials that increase the interpolation accuracy and counter the flat spot problem, this RBF interpolation constitutes the most sophisticated technique considered in this work. However, a high degree of "sophisticatedness" most often comes at the cost of increased computational complexity. Even the basic model as presented in Equation 2.17 involves the necessity of matrix inversion for solving the resulting system of linear equations. This system needs to be even more extended when SCPD RBFs shall be reliably utilized. Fortunately, Skala proposed an incremental version of RBF interpolation in [Ska13]. Here, the computational complexity is reduced to the quadratic class by means of remembering the previous inverse matrix. The interpolation-strategy as presented in Chapter 9 can make use of this enhancement, since only a single sampling point is replaced at a time.

Another aspect that should be considered is the number of hyperparameters the individual interpolation techniques add to the configuration space of the SLAS. While for all of the methods an appropriate number of sampling points to be stored in SP needs to be selected, which constitutes a rather simple decision mainly affected by the available memory, particular techniques require further configurations. For example, the MSM technique needs an additional configuration for the number of sampling points N_p that should be involved for the additional polynomial fitting. Thacker et al. [Tha+09] provide valuable guidelines for the selection of these parameters. With regard to the RBF interpolation, on the one hand, the RBF itself has to be selected. On the other hand, depending on whether the choice yields a strictly or else a conditionally positive definite radial function, in the first case a shape parameter ϵ needs to be carefully determined. The choice of an appropriate value for ϵ is crucial for the interpolation quality, since the "optimal" choice depends strongly on the underlying function to be interpolated. As will be shown in Chapter 9, the parameter-free TPS RBF yields promising results without adding a further hyperparameter.

2.4. Learning Classifier Systems

Learning Classifier Systems (LCS) comprise a family of ERBML techniques. Their invention dates back to 1976 when John H. Holland firstly presented the notion of *Cognitive Systems* in [Hol76]. Since that time, two main branches of LCS emerged in the literature: Firstly, the so-called *Michigan-style* LCS due to Holland appeared. As a second, alternative paradigm the *Pittsburgh-style* (or simply Pitt) approach has established since its introduction by Smith [Smi80] in 1980. The latter style is not subject of discussion in this thesis, since these systems demand for a higher extent of computational resources and lack the OL capability [UM09]. This capability, however, constitutes an essential requirement as will be described in Chapter 3. Pitt-style systems usually learn in a batch-wise (or offline) manner. They are evolving a population of entire *rule sets* (which constitute the candidate solutions) once per generation instead of incrementally evolving individual rules. Thus, Pitt-systems are far better suited to data mining tasks with fixed data sets where quick reaction times not constitute a critical issue.

In contrast, Michigan-style LCS, actually evolve one single solution in an iterative fashion following the principle of OL. This single solution is represented by a population of rules. In the context of LCS, these rules are also called *classifiers*, what is the main reason for this algorithm family's name. In each iteration, selected individual rules are updated as a response to the data instance presented to the system and the gained reward after performing a selected action (test-then-train). Michigan-style LCS received the most research attention over the past four decades. The invention of the XCS classifier system in 1995 probably constitutes the most influential stepping stone. In this section, a generic Michigan-style LCS will be described at first. Afterward, the focus is set on the XCS derivative. Necessary adaptations that allow for an application to real-value encoded feature (or input) spaces be further discussed. At the end, a special-purpose XCS variant for the task of function approximation is briefly introduced.

2.4.1. A Generic Modern Michigan-style LCS

This section provides a thorough introduction of the working mechanisms behind modern Michigan-style LCS.[5] Accordingly, in the remainder Michigan-style LCS are simply denoted by LCS. For the sake of brevity, a comprehensive review of the historical development of LCS is omitted. The interested reader is referred to various surveys available in this regard, e.g., [WG89; LR00; UM09; Bul15].

Modern LCS are mostly inspired by Stewart W. Wilson's work. He recognized that the rather complicated architecture proposed by Holland could be one of the reasons of stagnating progress in LCS research at that time. With his *Zeroth-level Classifier System* (ZCS) [Wil94], Wilson proposed to get rid of the rather complicated message-list mechanism at the core of Holland's primary system. This and further modifications have laid the foundation for a fresh impetus regarding research on LCS. But a major stepping stone for LCS history was reached only one year later in 1995. Again due to Wilson, a further extension toward the probably most important and most investigated LCS today was introduced – the XCS Classifier System [Wil95].

[5]LCS that appeared after Wilson's revolutionary modifications in the mid 90s are regarded as "modern" here

With XCS, Wilson paved the way for a more deeper understanding and insightful theoretical advances regarding the complex interactions of the system's main components. For instance, the synergy of the incorporated RL techniques for local rule refinement with the globally acting GA for input space coverage optimization.

More precisely, a modern LCS can be subdivided into three main components: (1) The *performance component* that manages the interaction with the learning environment (or the *System under Observation and Control* (SuOC) in OC terminology, cf. Sect. 3.1). (2) The *discovery component* which is responsible for optimizing the classifier coverage of the input space as well as for constructing new knowledge on demand by means of a steady-state GA and a so-called *covering* mechanism, respectively. (3) The *reinforcement component* which updates the inner parameters of the evolved classifiers (or *production rules*) in order to judge on their accuracy in predicting a reasonable response (*action*) for a particular set of situations (*condition*). In the following, each of these three components is described in more detail.

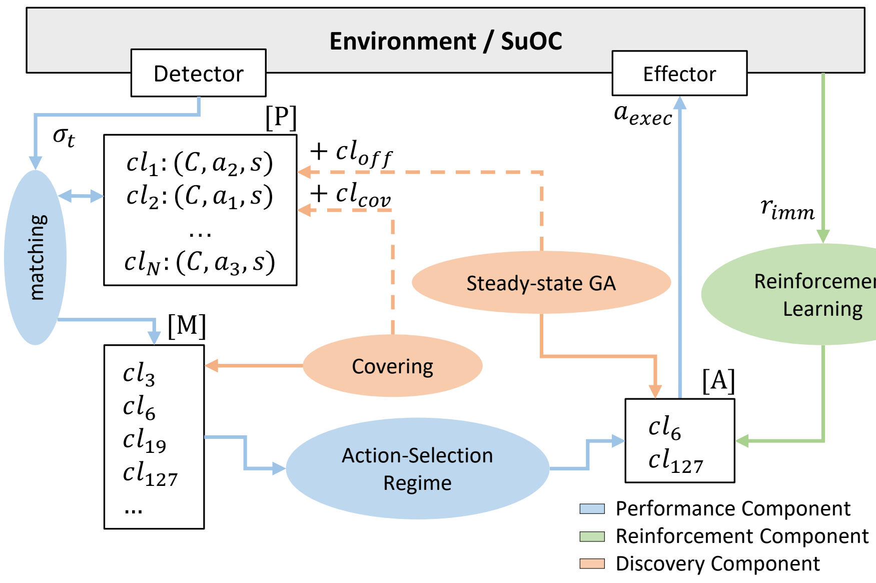


Figure 2.4.: Schematic of a generic modern LCS. The colored ellipses illustrate the mechanisms which belong to one of the three main components. The analogously colored arriving and outgoing arrows show their interactions with other components and integral parts of the system depicted as rectangles.

Figure 2.4 depicts a schematic of a generic modern LCS and illustrates a basic iteration through the main loop. All classifiers are stored in the population $[P]$. Classifiers that match the current incoming situation σ_t form a so-called match set $[M]$. The action set $[A] \subseteq [M]$ contains all classifiers from $[M]$ which advocate the same action as selected by the action-selection-regime. Subsequent to action execution and a reward signal is retrieved, classifiers in $[A]$ are going to be reinforced. Additionally, $[A]$ serves as mating pool for the steady-state GA.

In this thesis, the following notation is used (cf. Appendix A): The current *situation* (or system state, data instance, feature vector) which arrives at the learning system at time t is denoted by σ_t.[6] The predicted target variable with respect to the learning task at hand is denoted by the *action* a_{exec} here, which stands for the "action to be executed". The numerical feedback, or *reward*, which is delivered immediately by the *environment* after executing a_{exec} is denoted by r_{imm}. The environment here is defined analogously to the RL terminology. It comprises the learning task and delivers the situations perceived via the *detector*, takes the actions realized by the *effector*, and eventually provides the rewards. Whenever a particular classifier is referred to, cl_* serves as a reference, where here the $*$ is a placeholder for an index or another descriptor clearly identifying a specified rule. Additionally, an overview of any notations used throughout this thesis, including a dedicated section for LCS-related designations, is provided in Appendix A.

Performance Component

Once an environmental situation σ_t is perceived by the LCS's *detector* interface, the performance component initiates an iteration through the system's main cycle. The first procedure is called *matching*. It is responsible for checking whether the *condition* part of any classifier cl contained in the rule-base $[P]$ at that time, is fulfilled by the current σ_t. All $cl \in [P]$ for which this check evaluates true form the so-called *match set* $[M]$. To entirely comprehend how this can be achieved, first a closer look at the structure of an individual classifier has to be taken.

Classifier In its very simplest form, a classifier is defined by a triple $cl := (C, a, s)$. In the following, the dot-notation $cl.*$ is used to refer to the components or else learning parameters (specified by $*$) of a classifier. Within the aforementioned basic triple describing an individual classifier, $cl.C$ encodes the *condition* that has to be satisfied for "classifier activation". Thus, if the condition is satisfied the classifier cl matches the current situation σ_t. A condition determines a certain subspace of the input space (denoted by X in the following). Depending on the domain of the input space, the concrete encoding of $cl.C$ can differ significantly. For instance, in a binary domain, each bit of a binary input string is matched against a corresponding string of symbols from the ternary alphabet $\{0, 1, \#\}$. Thereby, only congruence of the entire bit string with a particular classifier's condition is sufficient to positive matching. The only exception where differences in the strings are permitted is at those places within the conditions where the *wildcard* or "don't care" symbol #

[6]At some places where the subject of discussion is not immediately related to LCS, σ_t can be used interchangeably with the notation for a feature vector $\vec{x}$.

is present. This allows for the crucial capability of LCS to *generalize* over similar inputs. Two of the most important condition representations will be introduced below in the section explaining XCS.

As for the general RL setting, $cl.a \in A$ denotes the *action* from a specified action space A that a certain classifier cl advocates. At this point, the analogy to production rules becomes apparent. Accordingly, a classifier can be interpreted as a

$$\text{IF}(cl.C \text{ matches } \sigma_t) : \text{THEN}(\text{execute } cl.a) \text{ rule.}$$

Finally, each classifier stores an estimate of quality, termed *strength $cl.s$*. The intuition is again similar to that of Q-values in general RL. It estimates the expected return when executing action $cl.a$ in the current situation σ_t, or in the case of LCS, any other situation which satisfies the classifier's condition $cl.C$. The last statement is important, since it again points to one of the major differences between LCS and standard tabular temporal difference learning techniques, such as Q-learning [WD92]. Namely, the ability of LCS to generalize over more than one single state. The strength value $cl.s$ also serves as *fitness* for the involved GA.

Population The entirety of classifiers cl constitute the so-called *population* $[P]$, also simply called rule set. With regard to the terminology which will be introduced in the succeeding chapter, the population constitutes the *knowledge base* of LCS. The size of $[P]$ is limited to a maximum number of N classifiers. This essentially exerts a sort of competition for survival among existing classifiers $cl \in [P]$, since each time a new classifier is to be added to an already filled up population, another classifier has to be removed. This kind of *set pressure* is due to the interplay of the involved GA of the discovery component and a *replacement* mechanism that ensures to satisfy the limit of $[P]$, i.e., $|[P]| \leq N$. The maximum size of $[P]$ is strongly application dependent. Generally, it should be chosen large enough to allow for a complete coverage of the input space during the initial learning phase when the population is initialized by means *covering* (cf. the paragraph on the discovery component below). This prevents a so-called *covering-deletion cycle* which results in continuing *detrimental forgetting*. That is, classifiers covering already explored niches are replaced in favor of newly "covered" classifiers before a reasonable number of updates can apply to increase their fitness and, thus, reduce their probability of being deleted. This aspect is briefly revisited again in the next section.

Now with the established notion of the general classifier structure and of the population $[P]$ at hand, it can be proceeded with a more detailed description of a single iteration through the system main loop cycle.

Main loop iteration As outlined above, the matching procedure checks which $cl \in [P]$ match the current situation σ_t. Each classifier whose condition is satisfied becomes a member of the match set $[M] \subseteq [P]$. The match set $[M]$ typically contains classifiers that advocate different actions at the same time. This is where the *action-selection regime* comes in. It is responsible for resolving this conflicting situation. This is accomplished by determining the action which promises the highest expected reward when the evolved knowledge should be *exploited*. On the other hand, the action-selection procedure can also decide probabilistically which action to be executed next in order to *explore* the problem space and to gain new, but perhaps negative experience. This well-known *exploration vs. exploitation* trade-off is one of the most challenging issues in RL as already discussed before. It is usually dealt with by the policy π the system or agent follows. In terms of LCS, often an interleaving strategy is applied. That is, switching between randomly and greedily chosen actions after each detected incoming situation. Once a decision on a_{exec} has been made, another subset of $[P]$, or rather $[M]$ is formed: The *action set* $[A]$. As the name indicates, $[A]$ contains all classifiers from $[M]$ whose advocated action $cl.a$ equals a_{exec}. More formally, $[A] := \{cl \in [M] \mid cl.a = a_{exec}\}$. Eventually, a_{exec} is executed on the environment via the *effector* interface which translates the abstract encoded a_{exec} into applicable actuator signals. Subsequently, the immediate reward r_{imm} for performing the selected action in the current situation is received, what completes the performance cycle and triggers the reinforcement mechanisms to be described next.

Reinforcement Component

The immediate reward r_{imm} is then further used to update all classifiers $cl \in [A]$. Accordingly, the strength values for all classifiers in $[A]$ are incrementally adjusted in the course of interaction with the environment, or else cycles through the main loop. Two scenarios of this *credit assignment* procedure have to be distinguished – the *single-step* and the *multi-step* update.

Single-step The former case of single-step updates can be understood as a sequential control task with an episode length of exactly one (what essentially resembles an online classification task). Each learning iteration (environment interaction) ends after executing a single action a_{exec} for the current state σ_t. Thus, the immediate reward r_{imm} serves as direct feedback signal whether the selected action was appropriate or not. The succeeding problem instance σ_{t+1} is assumed to be entirely independent of the preceding situations. This means that the *Markov* property is not assumed. Thus, the challenge of delayed reward, which requires techniques that allow for a successive distribution of payoff among all classifiers that established the action chain leading to the reward, can be neglected here.

Multi-step This changes when multi-step problems are considered. Here, the *credit assignment problem* comes into play. In primary LCS, the *Bucket Brigade Algorithm* (BBA) [Hol85] was conceived to approach this problem. Holland drew an analogy to an economy, where he interpreted classifiers as both customers and suppliers in a market. A classifier trades with its preceding active classifiers and the potentially succeeding ones. The strength values serve as budget. Roughly speaking, classifiers bid for being active in the current step and pay a fraction of their strength to the preceding, i.e., stage setting classifiers, when they win. Later in 1994, Dorigo and Bersini drew an analogy between an LCS using BBA and the, at that time recent Q-learning algorithm in [DB94]. This was also the year when Wilson introduced ZCS, where another credit assignment mechanism, the so-called *implicit bucket brigade* was introduced. This implicit version differs to Holland's initial BBA in certain respects [Wil94]. Finally, for XCS the multi-step credit assignment mechanism was again modified toward an even stronger relation to the temporal difference mechanics of Q-learning. Lanzi conducted a formal analysis in this respect in [Lan02]. The exact update rules for each of the two scenarios are explained in detail for XCS below.

Discovery Component

As described above, the performance component of a generic modern LCS is responsible for "sensing" the environmental state σ_t, for filtering all matching classifiers into a match set, and finally for selecting an appropriate action a_{exec} on the basis of a certain action-selection regime. Afterward, the reinforcement component updates the rules that made it in the action set $[A]$ on the basis of the reward r_{imm} received after realizing the selected action a_{exec}. What remains is the third component of the introduced generic modern LCS – the *discovery component*. This component is responsible for the system's exploratory behavior. Therefore, it creates new classifiers via its "creative" component – the involved *steady-state niche GA*. Additionally, in order to ensure an always non-empty match set $[M]$ what in turn allows for even responding to situations the system has never been exposed to so far, a so-called *covering* routine constitutes the second part of an LCS's discovery component.

Covering Following their order of appearance in Figure 2.4, the covering mechanism is explained first. Each time an entirely new situation $\sigma_t = (x_1, \ldots, x_n)$ arrives for which the LCS would not have any matching $cl \in [P]$, the system would simply stall when there was no fallback mechanism. In order to prevent that, the covering mechanism has been introduced as a first reactive classifier creating component. It creates at least one new classifier cl_{cov} at each invocation. The actual number of classifiers to be generated during one covering execution depends on the hyperparameter θ_{mna} (see Appendix B for a complete list and explanations). This hyperparameter should be configured in view of the size of the action space A for the learning task at hand. For example, if the system only permits two distinct actions, e.g., *turn left* and *move forward*, then it is most often sensible to create one classifier for each of them in order to cover the entire problem space more quickly. Accordingly, θ_{mna} would be set to 2. However, when A is large, covering one classifier for each action could quickly guide the system into the abovementioned covering-deletion trap. Lack of matching classifiers is not the only case when covering applies. It is often also activated when $[M]$ does actually contain classifiers, but the represented diversity of actions in $[M]$ is smaller than the configured minimum number of actions θ_{mna} required to be represented within each $[M]$. Naturally, selecting an appropriate covering policy requires knowledge about the number of possible actions $|A|$. This becomes impossible when $|A| \to \infty$, or put another way, a continuous action space is assumed. Conventional LCS derivatives are not capable of dealing with continuous actions off-the-shelf. However, a few approaches to deal with continuous actions have been introduced already, confer for instance [Wil07; IBZ12]. Assuming a discrete action space as typical for LCS, covering creates at least one novel classifier cl_{cov} by initializing it with a predefined strength value $cl_{cov}.s = s_{ini}$. It further assigns a random action which is not already represented by any other classifier in $[M]$. The condition $cl_{cov}.C$ is initialized to the current situation σ_t and then generalized to match *similar* situations in a restricted probabilistic manner (see the details for XCS below)

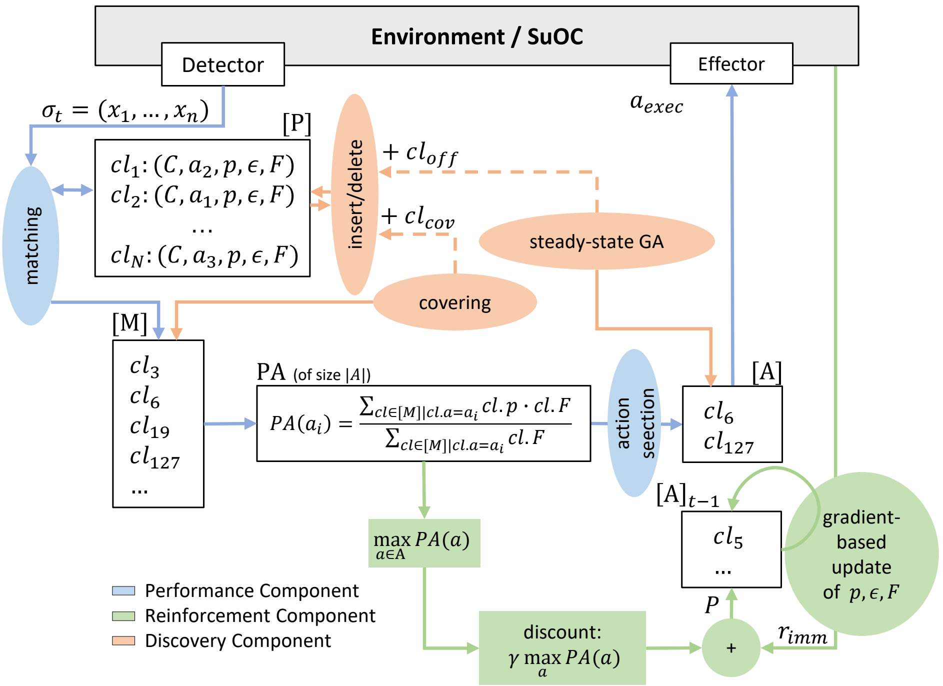
Niching GA As a second ingredient of an LCS's discovery component, a steady-state niche GA is brought into operation. A steady-state GA differs from a (standard) *generational* GA. Here not the entire population of individuals (recall that in the context of Michigan-style LCS these are the classifiers in $[P]$) is replaced by newly created offspring classifiers in each generation. Instead, only a certain subset of the entire population is replaced while the remaining classifiers are taken over to the next generation. However, the usual genetic operators *selection*, *recombination* and *mutation* are still performed in each iteration. In an LCS the GA creates two new *offspring classifiers* cl_{off} by selecting two *parental classifiers* cl_{par} on the basis of their strength estimates $cl_{par}.s$. The parents are deeply copied to produce the offspring rules in a first step. Next, a recombination (or *crossover*) of the offsprings' copied conditions are conducted by chance (dependent on the crossover probability χ). As a last operation, the condition, and maybe the actions as well, are altered by applying the *mutation* operator. Again the actual application is probabilistic and controlled by the mutation probability μ. Finally, the modified offspring classifiers are added to $[P]$, provided that they are not subject to *subsumption* as detailed shortly. The replacement mechanism as briefly outlined in the description of the performance component ensures the satisfaction of the population size limit. It further checks whether exactly congruent classifiers are already present in $[P]$ what would again result in subsuming the offspring. Instead of simply keeping the copied values for the strength estimates $cl_{off}.s$, those are set to the parents' average values and typically are reduced by a predefined discount factor. This is supposed to require the new classifiers to prove themselves in view of their modified input space coverage $cl_{off}.C$ and possibly modified actions $cl_{off}.a$. In the schematic illustration depicted in Figure 2.4, the GA is applied on the action set $[A]$. Accordingly, it directly acts on classifiers that all match at least a small fraction of similar situations $\sigma_i \in X$ and advocate the same action $a \in A$. The action set $[A]$ can also be understood as an approximation of a particular environmental niche in the problem space. Applying the GA to either $[A]$ as initially proposed by Wilson in [Wil95], or more commonly to $[A]$ [Wil98] results what is called *niche GA*. In early variants of LCS, the GA was applied *panmictically*, i.e., to the overall population $[P]$. The rationale behind the shift from a panmictic to a niche GA is revisited in the following sections regarding XCS. The invocation of the GA is usually controlled by the hyperparameter θ_{GA}. It determines the required minimum mean time since the last GA invocation. Hence, it is not necessarily triggered at every learning iteration but periodically.

The preceding paragraphs are intended as a brief introduction to the general concept of modern Michigan-style LCS. The general framework has been introduced as a subdivision into three main components and their respective tasks have been outlined. On that basis, in the next sections the details of the probably most prominent and today mostly investigated descendant of LCS – the *XCS Classifier System* (XCS) – is subject of discussion.

2.4.2. The XCS Classifier System

Back in 1995, a major milestone in the field of LCS research was reached by Stewart W. Wilson. He introduced a novel derivative of Michigan-style LCS, namely the *XCS Classifier System* (XCS). This variant paved to way for a major category within the LCS domain – *accuracy-based* LCS. Wilson proposed to base the fitness of individual classifiers not only on the strength estimate any longer but on the *accuracy* regarding the payoff prediction. Following his idea, he separated the quality related learning parameters of a single classifier into a triple consisting of p, ϵ, F. Thereby, $cl.p$ is an estimate of the reward or payoff an individual classifier expects when its advocated action is executed. The $cl.p$ parameter thereby has the same semantics as the strength attribute $cl.s$ as introduced before. $cl.\epsilon$ defines the estimated absolute *prediction error*, i.e., the absolute difference between the reward prediction $cl.p$ and the actually received payoff quantity P. The P quantity can be the immediate reward r_{imm} for single-step problems, or the sum of the immediate and the expected discounted future reward for multi-step problems. The most essential innovation, however, was the explicit *fitness* attribute $cl.F$ that estimates a classifier's *relative accuracy* in predicting the reward. Roughly speaking, $cl.F$ can be seen as a sort of inverse of the prediction error $cl.\epsilon$. It is niche-relative, i.e., estimated based on the prediction errors of partially overlapping classifiers cl_j sharing the same environmental niche (i.e., $cl_j \in [A]$). Instead of the payoff prediction $cl.p$ or strength $cl.s$, from now on the designated fitness attribute $cl.F$ is used by the selection operator of the involved steady-state niche GA and other calculations.

Before diving deeper into Wilson's fundamental thoughts leading to the invention of XCS, the extended architecture of the system is analyzed first. Figure 2.5 shows an augmented schematic of the XCS, adapted from the original illustration that can be found in [Wil98].



Environment / SuOC
Detector
Effector
$\sigma_t = (x_1, \ldots, x_n)$
[P]
$cl_1: (C, a_2, p, \epsilon, F)$
$cl_2: (C, a_1, p, \epsilon, F)$
…
$cl_N: (C, a_3, p, \epsilon, F)$
matching
insert/delete
$+ cl_{off}$
$+ cl_{cov}$
steady-state GA
covering
a_{exec}
[M]
cl_3 cl_6 cl_{19} cl_{127} …
PA (of size |A|)
$PA(a_i) = \frac{\sum_{cl \in [M] | cl.a = a_i} cl.p \cdot cl.F}{\sum_{cl \in [M] | cl.a = a_i} cl.F}$
action seection
[A]
cl_6 cl_{127}
$[A]_{t-1}$
cl_5 …
gradient-based update of p, ϵ, F
$\max_{a \in A} PA(a)$
discount: $\gamma \max_a PA(a)$
+
P
r_{imm}
Performance Component
Reinforcement Component
Discovery Component

FIGURE 2.5.: Schematic of the XCS classifier system

The main loop depicted in the schematic only reveals marginal changes in comparison to the generic LCS as shown in Figure 2.4. The reinforcement component is refined to indicate more explicit operations. As can be seen, these operations are performed on the action set $[A]_{t-1}$ of the previous time step $t-1$. Having a closer look, a certain similarity to Q-learning becomes apparent, which will be more clearly stated below. The classifiers' parameters $cl.p, cl.\epsilon, cl.F$ are indicated to be updated via gradient-based techniques. The standard update rule incorporated in XCS is the *Widrow-Hoff* update procedure [WH88], also referred to as *delta rule* according to [Mit97]. This technique adjusts the classifier parameters toward the direction of the steepest decent (negative gradient) in the underlying error surface. The gradient is simply approximated by the current deviation of the old estimate and the current value. Urbanowicz and Browne more intuitively introduce this approach as *time-weighted recency averages* (TWRA) in [UB17]. Furthermore, the action-selection mechanism as part of the performance component is now based on a new integral part of the system called *prediction array* (PA). The PA contains one scalar entry for each possible action $a \in A$. The entries calculate as the *fitness-weighted sum of the reward predictions* of those classifiers in $[M]$ that advocate the equal actions. More formally, one particular PA entry for action a_i which is denoted by $PA(a_i)$ calculates as follows:

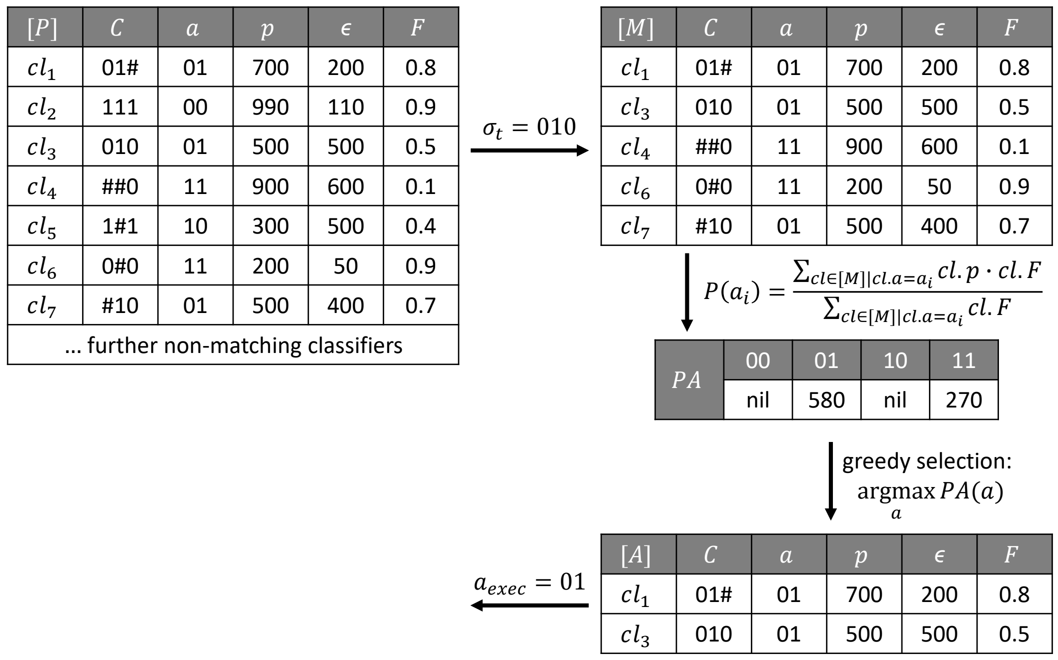
$$PA(a_i) = \frac{\sum_{cl \in [M] | cl.a = a_i} cl.p \cdot cl.F}{\sum_{cl \in [M] | cl.a = a_i} cl.F} \tag{2.24}$$

Accordingly, the PA provides accuracy-weighted estimates of the expected rewards for each action that is a potential candidate for execution in the current time step. The PA entries are thus also referred to as *system predictions.* At this point, the cooperation of individual rules becomes apparent. Overlapping classifiers in $[M]$ that propose to execute the same action are "mixed" based on their niche-relative fitness values. By means of this *classifier mixing* a collective system prediction for each possible action is modeled. Basing the action-selection on the fitness- or rather accuracy-weighted system predictions has the following effect: Classifiers which inaccurately predict high rewards via thier $cl.p$ parameter are selected with lower probability except for purely random choices during exploration. If not every action $a \in A$ is advocated by at least one classifier in $[M]$, the PA contains a *nil* entry. Thus, these actions are neglected for participation in the action-selection competition. At this point it is useful to revisit the *exploration vs. exploitation trade-off* issue once again. When the system is supposed to explore, the PA calculation essentially has no effect on the actual choice since the action to be executed is usually selected purely by chance. This has found to be necessary for learning agents to not get stuck in local optima. For example, when an agent learns action sequences (movements) that indeed lead to a desired target position, but maybe including unnecessary cyclic trajectories or overly long movement paths. In case of exploitation, a greedy strategy such as simply selecting the action with the highest PA entry seems reasonable. The alternation between those two selection strategies appears to be the default case for standard XCS implementations. This is plausible when these systems are applied to non-critical sequential or single-step (classification) tasks. However, it is not always legitimate to choose such an alternation strategy which essentially results in a 50% exploration rate. Considering safety-critical control tasks with humans involved, the system cannot freely discover action effects by numerous *trial-and-error* attempts. Thus, it should be restricted in its exploratory behavior. OC systems are often confronted with such a type of tasks. To deal with these cases, there exist mixed policies such as *roulette-wheel selection* [Gol89] or *ϵ-greedy* [SB98] policies, for instance. The former policy sets the selection probability of a particular action a_i proportional to the calculated system prediction $PA(a_i)$ [Wil95]. The latter regime has a fixed or decaying probability ϵ for random selection assigned.

Based on these informal explanations regarding the structural changes of the extended version of the previously introduced generic LCS, the subsequent paragraphs will focus on the more technical and algorithmic parts of XCS.

XCS for Binary Input Domains

In primary LCS, the input spaces X have typically defined to be binary, i.e., $X \subseteq \{0,1\}^n$. This is due to the fact that LCS were initially conceived around Holland's probably most prominent invention – the GA and the corresponding theory of *schemata* [Hol75]. In the binary case, the conditions $cl.C$ are encoded as strings composed by symbols out of a ternary alphabet $\Sigma := \{0, 1, \#\}$. More precisely, each condition is a concatenation of n symbols from Σ, where # defines a so-called *wildcard* or "do not care" symbol. The semantic of this symbol is that it evaluates true when matched against both 0 and 1. For example, an incoming situation $\sigma_t = 101$ would theoretically match seven different conditions, for instance 101 (the fully specified case), ### (the fully general or non-specified case) or one of the remaining intermediate cases such as 1#1.



$[P]$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_2	111	00	990	110	0.9
cl_3	010	01	500	500	0.5
cl_4	##0	11	900	600	0.1
cl_5	1#1	10	300	500	0.4
cl_6	0#0	11	200	50	0.9
cl_7	#10	01	500	400	0.7
… further non-matching classifiers					

$\sigma_t = 010$

$[M]$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_3	010	01	500	500	0.5
cl_4	##0	11	900	600	0.1
cl_6	0#0	11	200	50	0.9
cl_7	#10	01	500	400	0.7

$$P(a_i) = \frac{\sum_{cl \in [M] | cl.a = a_i} cl.p \cdot cl.F}{\sum_{cl \in [M] | cl.a = a_i} cl.F}$$

PA	00	01	10	11
	nil	580	nil	270

greedy selection: $\operatorname{argmax}_a PA(a)$

$[A]$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_3	010	01	500	500	0.5

$a_{exec} = 01$

FIGURE 2.6.: A handcrafted example to illustrate the matching process and the calculation of the prediction array

Figure 2.6 illustrates an exemplary matching process on a handcrafted classifier population and the resulting PA. According to Equation 2.24, the PA entry for the specified action 01 for instance calculates as follows:

$$PA(a = 01) = \frac{700 \cdot 0.8 + 500 \cdot 0.5 + 500 \cdot 0.7}{0.8 + 0.5 + 0.7} = 580 \tag{2.25}$$

In the depicted example, the action-selection is conducted deterministically, or rather greedily. More formally, $a_{exec} = \text{argmax}_{a \in A} PA(a)$. As can be seen, the action set $[A]$ contains only those classifiers from $[M]$ that advocate the action a_{exec} which was selected for execution.

Reinforcement After a_{exec} was applied to the environment, the reward r_{imm} is received. At this point, it is important to distinguish between single-step and multi-step problems again. First, it is concentrated on the more complicated multi-step task. In this case, the learning parameters $cl.p, cl.\epsilon, cl.F, \ldots$ of the classifiers contained in the previous action set $[A]_{t-1}$ are subject to reinforcement. Thus, the previous action set is temporarily stored and a state transition happens from t to $t+1$. It is explicitly noted here, that the immediate reward denoted by r_{imm} for the action a_{exec} realized at the preceding time step t as a response to the system state σ_t is assumed to be received after the transition, i.e., at time $t+1$. This detail is important for the correct interpretation of the update formulas stated below.

For all $cl \in [A]_{t-1}$ first the estimates for $cl.\epsilon, cl.p$ and $cl.F$ are updated in the given order. Please note that there are a few more so-called *book-keeping* parameters of which a classifier keeps track. For the sake of simplicity, these are neglected at this point and only introduced when necessary.

The prediction error estimates $cl.\epsilon$ for all classifiers in the previous action set are adjusted following the Widrow-Hoff (or simply delta) rule:

$$cl.\epsilon \leftarrow cl.\epsilon + \beta(|P - cl.p| - cl.\epsilon) \tag{2.26}$$

with

$$P = r_{imm} + \gamma \max_{a \in A} PA(a) \tag{2.27}$$

Here, the hyperparameters β and γ are learning rates. β determines the speed or step size of adjustment by weighting the old estimate, e.g., $cl.p$, and the new observation, e.g., P, in a convex combination. On the other hand, γ, also called the *discount factor*, controls the impact of rewards expected in the future. It is thus sometimes referred to as 'urgency of life' in the RL literature referring to the time left for an artificial agent to reach the goal before the episode ends. Thus, higher values of γ attribute higher influence to future reward estimates so that the agent becomes more far-sighted, and vice versa. A value of $\gamma = 0$, on the other hand, completely neglects the future and results in an agent behavior strictly pursuing the goal to maximize immediate rewards.

After the adjustment of the prediction errors, the payoff prediction estimates $cl.p$ are updated:

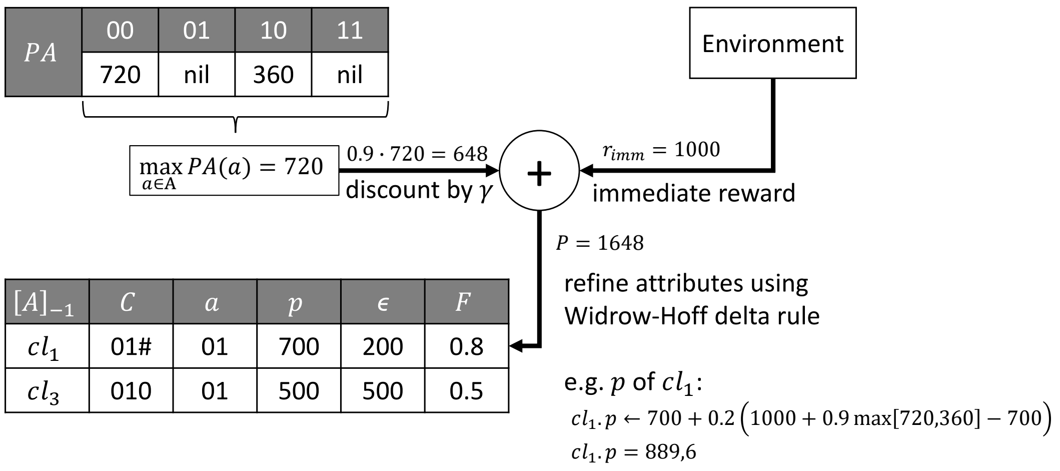
$$cl.p \leftarrow cl.p + \beta\ (P - cl.p) \tag{2.28}$$

Rewriting Equation 2.28 by substituting Equation 2.27 yields:

$$cl.p \leftarrow cl.p + \beta\ (r_{imm} + \gamma\ \max_{a \in A} PA(a) - cl.p) \tag{2.29}$$

Again, r_{imm} denotes the reward received after executing action a_{exec} in state σ_t. It is, however, received with a delay of 1, i.e., at time $t+1$. Also only in this current time step, the PA included in Equation 2.29 can be calculated. By then updating the classifiers in the previous action set $[A]_{t-1}$ (or simply $[A]{-1}$) information "from the future" can be incorporated to *backup* preceding rules. By means of this mechanism, the credit assignment problem is handled in XCS.

The complete update formula for $cl.p$ is reminiscent of the well-known Q-value update rule. This insight highlights the obvious relation to conventional temporal difference learning approaches such as Q-learning. The distinguishing aspect is, however, that instead of updating just one single entry of a Q-table, i.e., the $Q(s, a)$-value for a particular state-action pair (s, a), in XCS the update affects multiple state values or situations comprised by a classifier's condition $cl.C$. Furthermore, instead of the maximum Q-value for the resulting state after executing a_{exec}, the maximum entry of the next PA, i.e., the system prediction that promises the highest expected reward in the succeeding learning step, is used. Figure 2.7 depicts this algorithmic step and shows an exemplary adjustment of the payoff prediction estimate $cl.p$ of cl_1.



PA	00	01	10	11
	720	nil	360	nil

$[A]_{-1}$	C	a	p	ϵ	F
cl_1	01#	01	700	200	0.8
cl_3	010	01	500	500	0.5

FIGURE 2.7.: Illustrative example for updating the $cl.p$ parameter of an exemplary classifier (here cl_1) from the previous action set $[A]_{-1}$. In this example, the learning rate β is set to 0.2 and the discount factor γ is 0.9.

In the more simple case of single-step problems, the quantity of P is directly set to r_{imm} and the updates of $cl.p, cl.\epsilon$ and $cl.F$ are performed for all classifiers within the action set of the current time step t. Thus, the temporal dependence between succeeding situations of sequential decision tasks can be neglected for single-step tasks.

For the reinforcement of the fitness estimate $cl.F$, a few more steps are necessary. First, the *absolute accuracy* $cl.\kappa$ for each classifier in the respective action set needs to be temporarily calculated:

$$cl.\kappa = \begin{cases} \alpha \left(\frac{\epsilon}{\epsilon_0}\right)^{-\nu} & \text{if } cl.\epsilon \geq \epsilon_0 \\ 1 & \text{otherwise} \end{cases} \tag{2.30}$$

This function establishes an inverse relationship between a classifier's absolute accuracy in predicting the reward and the current estimate of the prediction error $cl.\epsilon$. Here, ϵ_0 serves as a *tolerance* hyperparameter that defines the level of error that is acceptable to assume a classifier as being fully accurate. It is therefore also regarded as *target error.* ϵ_0 is a critical hyperparameter. It controls not only the sensitivity to noise, but also the even more important fitness pressure that applies. Setting ϵ_0 too low might let the XCS struggle in identifying accurate rules and thus prevents the distinguishing between accurate and inaccurate rules. This, in the worst case, leads to random selections, and thus undirected innovations by the GA. Setting ϵ_0 too high on the other hand causes similar effects. A rule of thumb is to let ϵ_0 be 1% of the maximum reward that can be receive in single-step problems. For multi-step problems, however, this choice is not that obvious, and needs some sort of expert knowledge. Nevertheless, current research is concerned with analytically deriving optimal values for ϵ_0 and other hyperparameters [Nak+17].

The additional hyperparameters α and ν control the impact of larger prediction errors in terms of steeper descents in accuracy. These hyperparameters are typically set to default values $\alpha = 0.1$ and $\nu = 5$ and are only rarely adjusted.

Since the fitness estimate was introduced as a measure of *relative* accuracy, in a second step, the absolute accuracy values $cl.\kappa$ are normalized to obtain action set, or else *niche relative accuracy* values κ':

$$cl.\kappa' = \frac{cl.\kappa \cdot cl.num}{\sum_{cl \in [A]} cl.\kappa \cdot cl.num} \tag{2.31}$$

According to Equation 2.31, the relative accuracy $cl.\kappa'$ sets the absolute accuracy of an individual classifier cl in relation to the sum of all classifiers covering the same environmental niche. At this point, the first book-keeping parameter $cl.num$ called the *numerosity* of a classifier needs to be briefly introduced. As will be described more thoroughly below, classifiers can subsume others. *Subsumption* here means that instead of storing an identical physical copy of an existing classifier, simply the numerosity of the existing one can be increased. This is also done when the classifier to be added can be replaced by a more general and fully accurate classifier. In both cases, the subsumed classifier is dropped in favor of increasing the numerosity parameter of the subsuming one. Accordingly, classifiers with a higher numerosity should also gain a larger fraction during the normalization of the absolute accuracy estimates. As might be recognized, this normalization implements the concept of *fitness sharing* into XCS, which in general is a mechanism to maintain niches in populations. More detailed information on that advanced topic can be found in [BGL07].

Eventually, the fitness estimates $cl.F$ of all classifiers in the respective $[A]$ are updated by means of the usual Widrow-Hoff update rule using the previously calculated relative accuracy values κ':

$$cl.F \leftarrow cl.F + \beta(cl.\kappa' - cl.F) \tag{2.32}$$

Beside the use of the Widrow-Hoff rule, another concept called *Moyenne Adaptive Modifiée* (MAM) [Ven94] is applied for the update of $cl.\epsilon$ and $cl.p$ (and $cl.as$ to be defined below). In essence, MAM applies an incremental averaging update for the first $1/\beta$ steps, where β denotes the learning rate as usual. The motivation behind MAM is that during the first updates, the parameter approximations are expected to reach their expectation values more quickly. This can be understood as an adaptive learning rate for the first $1/\beta$ updates which would be $\beta = 1/n$, where $n = 1 \dots 1/\beta$ constitutes a counter of the updates performed so far. After this initial MAM phase, the Widrow-Hoff rule takes over to account for the presumed non-deterministic characteristics of the learning environment. At this place, another book-keeping parameter is introduced. The so-called *experience* $cl.exp$ of a classifier. A classifier's experience is initialized with 0 when it is newly created. It gets incremented each time the reinforcement component acts on it. It is thus a measure of how often this particular classifier has been active so far throughout the course of learning. Accordingly, classifiers with low experience have not had enough opportunities to prove themselves. Thus, convergence of their inner learning parameters cannot be assumed.

Micro- vs. Macroclassifiers The aforementioned *numerosity* parameter $cl.num$ is only one of the additional book-keeping parameters of which classifiers keep track. It determines how many syntactically equivalent rules has been subsumed by this classifier. Syntactically equivalent means that two classifiers have equal conditions and advocate identical actions. There are different places within XCS's algorithmic structure where classifiers are checked for being subsumable: (1) Whenever a new classifier is about to be inserted into $[P]$. (2) During the application of the GA, offspring classifiers are checked against their parents or even against all classifiers in their niche determined by $[A]$. Subsumption is an essential ingredient for the generalization capability of XCS. This aspect will become more clear, when Wilson's *generalization hypothesis* is subject of discussion. Classifiers with a numerosity greater than 1 are called *macroclassifiers*. Otherwise they are referred to as *microclassifiers*. Important to note is that the satisfaction of the population size limit is based on the sum of the numerosity values of all classifiers in $[P]$. Accordingly, $N \leq \sum_{cl \in [P]} cl.num$ is evaluated and not the number of physically stored micro- or macroclassifiers is considered in this regard. Thus, a single classifier with $cl.num > 1$ actually represents $cl.num$ microclassifiers that would advocate an identical action $a \in A$ for the same or else a smaller but entirely comprised set of situations $\sigma \in X$.

The next algorithmic steps to be described belong to the discovery component of XCS. According to Figure 2.5, there are three mechanisms that are responsible for the exploratory behavior of XCS: Covering, the steady-state niche GA, and the deletion/insertion mechanism which as of yet deliberately only took a back seat at the introduction of the generic LCS.

Covering As already discussed, covering is triggered whenever a situation σ_t occurs that is not covered by any classifier in $[P]$, or whenever $[M]$ contains less than θ_{mna} distinct actions. Thus, it guarantees an instant reaction of the system at any time. However, at the expense of a probably high prediction error due to the probabilistic and partially arbitrary initialization of the newly created classifiers cl_{cov}. For the ternary case, the hyperparameter $P_{\#}$ determines the initial generality of a classifier generated by covering. The higher $P_{\#}$ is set, the more general the covered classifiers are expected to be initially. Consider the following example which illustrates the general covering process presented in Figure 2.8.

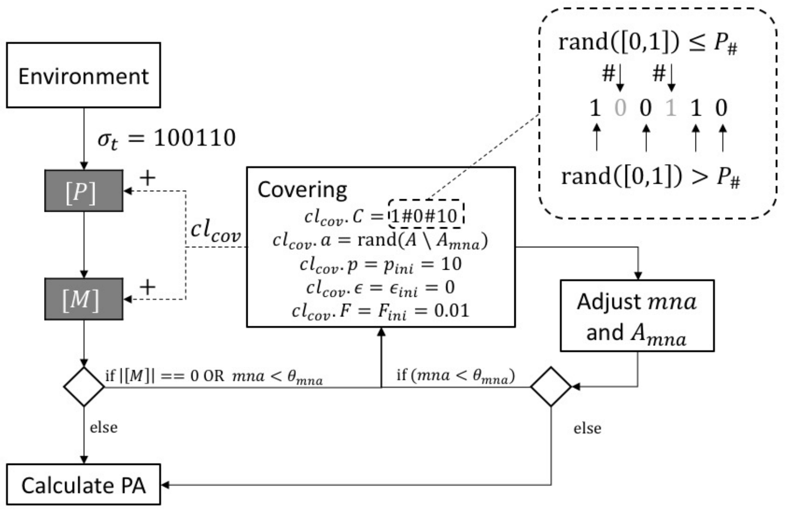

Environment
$\sigma_t = 100110$
$[P]$
$[M]$
+
+
cl_{cov}
Covering
$cl_{cov}.C = 1\#0\#10$
$cl_{cov}.a = \text{rand}(A \setminus A_{mna})$
$cl_{cov}.p = p_{ini} = 10$
$cl_{cov}.\epsilon = \epsilon_{ini} = 0$
$cl_{cov}.F = F_{ini} = 0.01$
$\text{rand}([0,1]) \leq P_{\#}$
# #
1 0 0 1 1 0
$\text{rand}([0,1]) > P_{\#}$
Adjust mna and A_{mna}
if $|[M]| == 0$ OR $mna < \theta_{mna}$
if $(mna < \theta_{mna})$
else
else
Calculate PA

FIGURE 2.8.: The covering mechanism of XCS illustrated

A situation $\sigma_t = 100110$ is detected by the system. Let $\theta_{mna} = 2$. The set of possible actions is given by $A \coloneqq \{00, 01, 10, 11\}$. The wildcard probability is set to $P_{\#} = 0.5$. Further, let mna be the number of distinct actions in $[M]$ and $A_{mna} \subseteq A$ denote the set of actions that are already present in $[M]$. Accordingly, $mna = |A_{mna}|$. Performing covering creates a new classifier cl_{cov} and temporarily sets its condition to the current situation. For each bit in $cl.C$, the covering routine individually decides to replace it by a wildcard # symbol with probability $P_{\#}$. In the figure, the second and fourth value are replaced by a #. The values for $cl_{cov}.p, cl_{cov}.\epsilon, cl_{cov}.F$ are set to predefined initial values $p_{ini}, \epsilon_{ini}, F_{ini}$. The action parameter $cl_{cov}.a$ is assigned an action which is chosen randomly from $A \setminus A_{mna}$, denoted by $rand(A \setminus A_{mna})$ in the example. The experience $cl.exp$ is set to 0. Another book-keeping parameter that keeps track of the time of creation and the time of the last GA operation is called the *time stamp $cl.ts$*. It is initialized with the current time step t. Finally, cl_{cov} is added to both $[M]$ and $[P]$. Depending on the value of θ_{mna} the covering process is repeated until $mna = \theta_{mna}$ is satisfied.

Covering occurs mainly at the beginning of a learning task. An important rule of thumb is to choose $P_{\#}$ sufficiently large such that the input space can be entirely covered in view of the population size restriction. Setting it too high, the population is initialized with classifiers belonging to the class of *over-general* rules. It might be necessary to choose $P_{\#}$ small enough to allow the GA to distinguish between accurate and inaccurate classifiers, i.e., to provide a sufficient *fitness signal* as discussed above. The choice depends strongly on the complexity of the underlying problem space.

XCS then has to automatically discover classifiers with an appropriate degree of generality or else specificity. The appropriateness is subject to a trade-off relationship between generality and accuracy. The overarching goal of XCS is to construct a state-action-payoff mapping $X \times A \to P$ which fulfills the desired properties of being (1) *maximally accurate*, (2) *maximally general*, (3) *compact* and (4) *complete*. These properties are considered with regard to the rule set $[P]$ [Wil95; Kov98]. This objective can be interpreted as a multi-objective optimization problem. In XCS, a solution to this problem can be achieved by exerting several *learning pressures* to the system. A great amount of this pressure is due to the steady-state GA, which is explained in the next paragraph.

Niche Genetic Algorithm For the purpose of discovering new rules and, therefore, exploring the underlying problem space, XCS incorporates a steady-state niche GA. The GA is regarded as *niching* since it acts on environmental niches which are approximated by a set of matching and, thus, partially overlapping classifiers. The GA is periodically invoked whenever the following criterion is fulfilled:

$$t - \left(\frac{\sum_{cl \in [A]} cl.ts \cdot cl.num}{\sum_{cl \in [A]} cl.num} \right) > \theta_{GA} \tag{2.33}$$

Accordingly, the GA is invoked when the average time since its last activation (given by the *time stamp* book-keeping parameter $cl.ts$) over all classifiers in $[A]$ exceeds a predefined threshold θ_{GA}. Again, this calculation is performed on the level of microclassifiers, since the numerosity values are factored in. In essence, three genetic operators are used in standard XCS: (1) *Selection* of the two parental classifiers (cl^1_{par}, cl^2_{par}). (2) *Crossover* of the conditions of the two generated (i.e., copied) offspring classifiers (cl^1_{off}, cl^2_{off}) (3) *Mutation* of the offsprings' conditions and possible their actions.

Parental selection is traditionally performed by means of *fitness proportionate selection*, often implemented as *roulette-wheel selection*. That is, the probability of a classifier $cl \in [A]$ to be selected as a parent for reproduction increases proportionally with increasing fitness $cl.F$. In [BSG03], the alternative *tournament selection* variant has been investigated. It was found to yield better properties in terms of an improved differentiation between accurate and inaccurate classifiers. This eventually leads to more stable fitness pressure. Tournament selection can be viewed as today's standard technique in the context of XCS. Each time the GA is invoked, two parental classifiers are selected for reproduction.

Subsequent to the selection of the parents for mating, two offspring classifiers $cl^1_{off}.C$ and $cl^2_{off}.C$ are created by simply copying the parents. The conditions $cl^1_{off}.C$ and $cl^2_{off}.C$ are then altered by means of *recombination*. That is, by crossing a certain subset of the bits/symbols between the offsprings' conditions by chance. The probability of crossover happening at all again is determined by the XCS hyperparameter χ which is usually set to a high value ($\chi = 0.8$). It can be distinguished between different crossover variants: (1) If each symbol of the respective conditions is considered individually and exchanged with the a probability $p = 0.5$, this variant is called *uniform crossover*. (2) Other approaches are so-called *n-point crossovers*. Usually $n = 1$ or $n = 2$ crossing points are determined at random or at design time in advance. The latter can be done when it is assumed that specific bits at specific locations are interrelated. Then only the resulting substrings, e.g., the right, or the inner parts of the offsprings' conditions, are exchanged [But05a]. See Figure 2.9 for an illustrative example of an $n = 2$-point crossover.

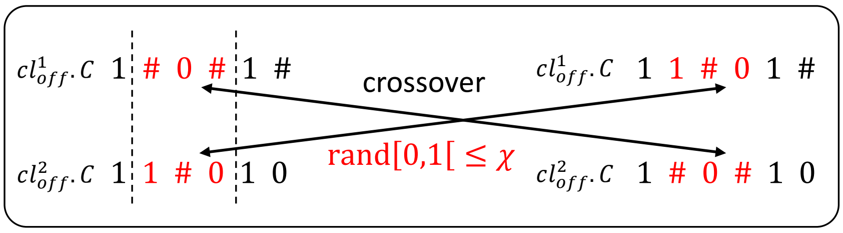

$cl^1_{off}.C$ 1 # 0 # 1 #
$cl^2_{off}.C$ 1 1 # 0 1 0
crossover
rand[0,1[≤ χ
$cl^1_{off}.C$ 1 1 # 0 1 #
$cl^2_{off}.C$ 1 # 0 # 1 0

FIGURE 2.9.: An illustrative example of an $n = 2$ point crossover

The last genetic operator applied is called *mutation*. As for crossover, mutation acts on the offsprings' conditions $cl^1_{off}.C$ and $cl^2_{off}.C$. However, it works individually on both. Mutation permutes each symbol in the corresponding condition with probability μ (typically set to to a small value, $\mu = 0.04$). If a certain bit (*gene*) within the condition $cl_{off}.C$ (*chromosome*) is subject to mutation, the value (*allele*) of this particular position (*locus*) is flipped from 0 to 1 and vice versa. In case of a wildcard # being the current allele, this gene is mutated to 0 or 1 with equal probability. The terms in the parentheses shall denote the terms from the genetics domain sometimes used in the literature.

For binary input spaces, mutation exerts a specific form of evolutionary pressure to the classifiers, which is called *mutation pressure*. Considering the 6 possible permutations it becomes apparent that the probability of specifying a bit is twice as high as for generalizing it. More precisely, *general-to-specific* mutations (e.g., $\# \rightarrow 1$) have a probability of $2/6 = 1/3$. Indeed, *specific-to-general* mutations, e.g., $1 \rightarrow \#$ are equally likely. However, there remains the *specific-to-specific* case, which also adds a one-third chance toward specificity. Accordingly, the overall mutation probability of resulting in a more specified condition is $p(specify) = 2/3$. In contrast, the generalization probability is only $p(generalize) = 1/3$. This is an important property of the applied GA, since this is the only algorithmic part which explicitly pushes toward specification of classifiers, except of the condition initialization during covering. Figure 2.10 shows an exemplary mutation operation.

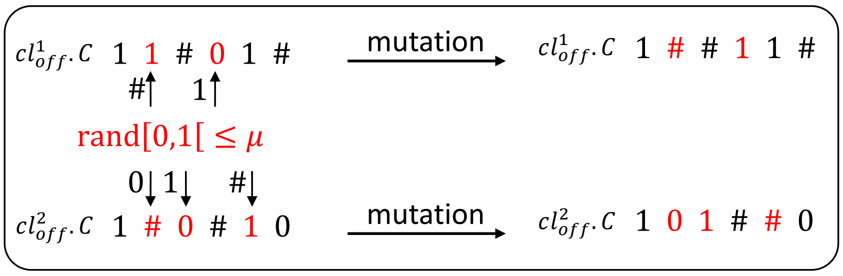

$cl^1_{off}.C$ 1 1 # 0 1 #
#↑ 1↑
mutation
$cl^1_{off}.C$ 1 # # 1 1 #
rand[0,1[≤ μ
0↓ 1↓ #↓
$cl^2_{off}.C$ 1 # 0 # 1 0
mutation
$cl^2_{off}.C$ 1 0 1 # # 0

FIGURE 2.10.: An example for the genetic operator mutation

Another possibility for mutation is on the actions $cl_{off}.a$. When *action mutation* is activated, the corresponding action of an offspring classifier cl_{off} is exchanged with one of the other possible actions $a \in A \setminus \{cl_{off}.a\}$. The probability is the same as for the condition alleles, i.e., μ. Action mutation increases the exploratory behavior of the system and facilitates a faster coverage of the entire problem space. However, in most of the studies reported in the literature it is turned off.

Subsequent to crossover and mutation, two novel classifiers have been created with a certain probability. Sometimes, however, due to the probabilistic nature of the GA, exact copies of the parents constitute the result. Either way, as of yet the parameter values for $cl.p, cl.\epsilon$ and $cl.F$ (as well as for the other book-keeping parameters) are identical to the immediate parent, e.g., $cl^1_{off}.p = cl^1_{par}.p$. When crossover has been applied to the offspring classifiers, all aforementioned values are set to the average of both parents, e.g.,

$$cl^1_{off}.p = cl^2_{off}.p = \frac{cl^1_{par}.p + cl^2_{par}.p}{2}.$$

This is done analogously for $cl.\epsilon$ and $cl.F$. Independent on whether crossover was performed or not, the estimates for fitness and prediction error are then discounted by so-called reduction factors $F_{reduction}$ and $\epsilon_{reduction}$. Typically, these are set to 0.1 and 1.0, respectively. With a fitness reduction of 0.1, the offspring classifiers have their fitness estimates reduced by 90 %. Thus, the new classifiers have to prove themselves again to be still accurate or maybe even more accurate with regard to their new conditions $cl_{off}.C$ and possibly their new actions $cl_{off}.a$. Due to the applied fitness reduction, new classifiers just produced by the GA will usually not be selected immediately for reproduction. However, if the "inherited" parameters for the payoff prediction $cl_{off}.p$ turn out to be correct, the corresponding fitness values will increase quickly. The prediction error is usually directly adopted (due to the default error reduction factor of 1). The numerosity parameter $cl.num$ is set to 1 and the time step parameter $cl.ts$ is set to t. The experience of a newly created classifier is set to 0, since it has not been applied so far. Additionally, all classifiers in the current mating pool, i.e., $[A]$, get their time stamp parameters set to the current time step t.

In the previous paragraphs, the concept of subsumption has been mentioned a couple of times. There are several places, where a classifier ready to be inserted in $[P]$ can be checked to be subsumable by a more general and also fully accurate one. One way is to check, whether the generated offspring classifiers cl^1_{off} and cl^2_{off} are directly subsumable by one of their direct parents cl^1_{par} and cl^2_{par}. If possible, the numerostiy of the respective parent is increased by one and the subsumed offspring is discarded. More precisely, a classifier cl_1 subsumes another classifier cl_2 if all of the following requirements hold:

1. The number of wildcards in $cl_1.C$ is greater than the number in $cl_2.C$ and the remaining symbols at those loci where both classifiers are specified are equal.
2. The prediction error of the potential subsumer cl_1 is sufficiently small, i.e. $cl_1.\epsilon < \epsilon_0$.
3. cl_1 has been updated sufficiently often, i.e., $cl_1.exp > \theta_{sub}$.

Recall that $cl.exp$ denotes the number of performed reinforcement updates, and θ_{sub} is another threshold a potential subsumer has to exceed before subsumption is permitted.

An even more radical form of subsumption is *action set subsumption.* Here, the offspring classifiers are not only compared against their direct parents, but also against all classifiers of the mating pool, here $[A]$.

Subsumption is an essential concept in XCS, since it supports generalization and compactness of the final problem solution which is represented by the classifiers in $[P]$ [Wil98] after learning is completed.

Classifier Insertion and Replacement Whenever a new classifier is to be inserted into $[P]$, regardless of whether generated by covering or the GA, a few preconditions have to be met. First of all, the size of $[P]$ must not exceed the maximum number of (micro-)classifiers N. Thus, if after the insertion of a new classifier the assertion $|[P]| = \sum_{cl \in [P]} cl.num > N$ evaluates true, $|[P]| - N$ classifiers have to be deleted in favor of the new one(s). It is important to note that although the GA generates new classifiers within a certain niche, the deletion candidates are chosen from the set of all classifiers $[P]$. This combination again stresses the population towards consisting predominantly of general but accurate classifiers, as will be explained more thoroughly below. To decide on candidates for removal, a *deletion vote* is calculated for each $cl \in [P]$. This vote depends on three factors:

1. The average *action set size* of a certain classifier: $cl.as$ [7].
2. The mean fitness within the entire population $[P]$, denoted by $\bar{F} = \frac{\sum_{cl \in [P]} cl.F}{\sum_{cl \in [P]} cl.num}$
3. A classifier's experience $cl.exp$

Based on the aforementioned factors, a classifier's *deletion vote* is calculated as follows:

$$vote(cl) = \begin{cases} cl.as \cdot cl.num \cdot \frac{\bar{F}}{cl.F/cl.num} & \text{if } cl.exp > \theta_{del} \wedge \frac{cl.F}{cl.num} < \delta \cdot \bar{F} \\ cl.as \cdot cl.num & \text{otherwise,} \end{cases} \tag{2.34}$$

Here δ (often set to 0.1) determines the fraction of the population's mean fitness $\bar{F}$ a classifier's *microfitness* $cl.F/cl.num$ has to exceed. Otherwise, its deletion vote is increased by a factor $\bar{F}/\frac{cl.F}{cl.num}$. The threshold θ_{del} is another hyperparameter which defines the minimum number of updates a classifier is required to experience before its deletion vote can be increased based on its microfitness. After all classifiers in $[P]$ have their deletion votes calculated, the deletion candidate is determined by means of roulette-wheel selection and eventually "removed". That said, if the deletion candidate's numerosity is greater than 1, its numerosity is simply decremented instead

[7] $cl.as$ is the last book-keeping parameter to be introduced. Its estimate is initialized with 0 and incrementally adjusted by the reinforcement component in a similar fashion as for $cl.p$ or $cl.\epsilon$, including the MAM bootstrapping procedure.

of an actual removal of it. In case of a selected microclassifier, i.e., $cl.num = 1$, it is completely removed from $[P]$.

As briefly mentioned above, the insertion of new classifiers is the second site where subsumption can take place. If a new classifier cl^* is about to enter the population, $[P]$ is first checked for a congruent classifier, i.e., with identical condition and action. More formally, if the assertion $\exists cl \in [P] : cl.C = cl^*.C \wedge cl.a = cl^*.a$ evaluates true, then cl^* is discarded and the numerostiy of the existing classifier is incremented.

With all the concepts and algorithmic steps formally described in the previous paragraphs, the basic understanding of the general functioning is now established. In the next section, the focus is shifted to considerations regarding the real-world applicability of XCS. Therefore, necessary modifications that allow XCS for dealing with real-valued input domains are explained in the next paragraphs.

XCS for Real-valued Input Domains

So far, only binary input spaces have been assumed for the introduction of XCS's working mechanisms. Naturally, in real-world applications the situations and state representations of systems and their surrounding productive environments are often represented by real-valued numbers. An encoding in binary strings is possible, but seems not plausible since this would lead to tremendously large input spaces and involves other problems such as the preservation of order (depending on the applied binary encoding). It is clearly desirable allow for a straightforward utilization of XCS in the presence of numerical, and even continuous input spaces. Therefore, σ_t is now represented by a vector $\vec{x}_t = (x_1, \ldots, x_n)^T$, where n denotes the number of different features or the dimensionality of the input space $X \subseteq \mathbb{R}^n$. In order to allow for the application of XCS in nominal- and real-valued input domains, Wilson introduced modifications to XCS, which he called simply *XCS Classifier System for Integer-valued Input* (XCSI) [Wil01] and *XCS Classifier System for Real-valued Input* (XCSR) [Wil00], respectively. It is important to note that these extensions only provide a solution for the discrete- or continuous-valued input spaces. The action representation still remains categorical, thus the action spaces are still required to be discrete.

Most of the XCS mechanisms discussed so far remain unaffected by Wilson's modifications. However, particular changes are necessary for the condition representation, as well as for all mechanisms that work with a classifier's condition $cl.C$. In particular, these are the matching and covering mechanism as well as the GA. In the following paragraphs, only the most common modifications are discussed. This, however, suffices to provide a solid basis for comprehension of the techniques which will be developed in this thesis, as well as for a further self-study. Furthermore, it is focused on the more general case of real-valued inputs and, thus, on XCSR.

Conditions with Interval Predicates In a real world learning environment, where e.g., sensors constitute the source of system state descriptions, it is hardly feasible to encode the current state σ_t only in binary form. Hence, a classifier's condition $cl.C$ cannot be represented by a string of symbols from the ternary alphabet $\{0, 1, \#\}$ anymore. Instead, several encoding schemes have been proposed in the literature that deal with continuous-valued inputs. Among them, simple interval-based representations [Wil00; SB03], but also complex hyper-ellipsoidal forms [BLW06] and a few more. To provide a basic intuition, in the following the most straightforward approach – the interval-based *hyper-rectangular representation* – is introduced.

The designation of this representation originates from the geometric interpretation of a condition $cl.C$ when visualized in a real-valued input space $X \subseteq \mathbb{R}^n$. For each dimension $i = 1 \ldots n$ in X, a classifier encodes a so-called *interval predicate* (l_i, u_i). Each interval predicate contains a lower bound l_i as well as an upper bound u_i, with $l_i, u_i \in \mathbb{R}$. Without loss of generality, in the following it is assumed that all input space dimensions are normalized to $[0, 1]$. Accordingly, $l_i, u_i \in [0, 1] \subseteq \mathbb{R}$ holds. The condition now is represented as a concatenation of interval predicates, and not as a string of symbols from the ternary alphabet as done for the binary case. Figure 2.11 provides an illustrative intuition.

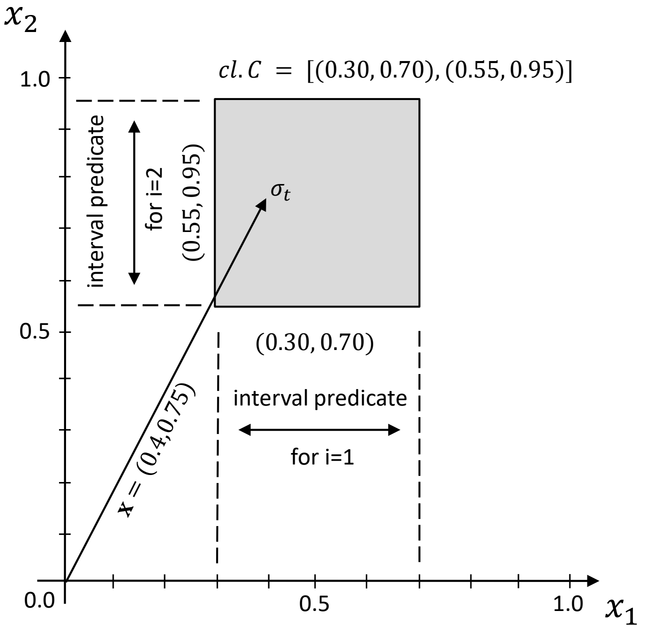


FIGURE 2.11.: Visualization of a hyper-rectangular condition in a two-dimensional input space $X := [0, 1]^2$

The condition appears as a rectangle within the two-dimensional input space $X := [0, 1] \times [0, 1] \subseteq \mathbb{R}^2$. In higher-dimensional input spaces, the geometric shapes are referred to as hyperrectangles. The current σ_t is represented by a vector $\vec{x} =$

$(0.4, 0.75)^T$ which is matched by the depicted classifier's condition

$$cl.C = [(0.30, 0.70), (0.55, 0.95)].$$

Modifications to Matching and Covering According to the illustrated example in Figure 2.11, generally a classifier cl matches a certain input $\sigma_t = (x_1, \ldots, x_n)^T$ if and only if

$$l_i \leq x_i \leq u_i, \quad \forall i = 1 \ldots n.$$

The so far introduced interval predicate encoding is also referred to as *Ordered Bound Representation* (OBR) [SB03]. It is called "ordered" since it implies that $l_i \leq u_i$. Since it was shown that this introduces an unintended bias to the exploration performance of XCSR for specific problem domains, Stone and Bull introduced the so-called *Unordered Bound Representation* (UBR) to alleviate the identified deficiencies. With the UBR the ordering of the lower and upper bounds within the interval predicates are neglected. Instead, the matching procedure has to take care of temporarily establishing the correct order. This allows for a less biased exploratory behavior during the recombination and permutation of the conditions during GA operation. As for the conventional XCS, all classifiers that fulfill the above matching criterion become members of the match set $[M]$.

In case of not fully covered input spaces, still situations might occur where the population does not contain matching classifiers and covering needs to be activated. Accordingly, also the covering operator needs slight modifications. As for the binary case, covering begins with creating a new classifier cl_{cov} whose condition is initialized maximally specific, i.e., it exactly matches σ_t. This is realized by setting both the lower and the upper bound to the value of the corresponding input dimension. In order to generalize from the completely specified condition $(l_i = x_i, u_i = x_i), \forall i = 1 \ldots n$, a new hyperparameter r_0 is introduced that replaces (or complements) $P_\#$. This *default spread* r_0 determines the maximum deviation of l_i and u_i from the current x_i. Thus, the interval width for each dimension is at most $2r_0$. The condition of cl_{cov} is then altered by

$$cl_{cov}.C = \Big[(x_i - \text{rand}[0, r_0], x_i + \text{rand}[0, r_0]) \Big], \quad \forall i = 1 \ldots n, \tag{2.35}$$

where $\text{rand}[0, r_0]$ yields a uniformly distributed random number between 0 and r_0. When using the UBR, the lower and upper bounds can be shuffled afterward to prevent bias. Any further steps of the covering routine remain unchanged for XCSR.

The choice of an adequate value for r_0 is crucial for ensuring adequate learning success. It directly controls the initial generality of classifiers as is done by $P_\#$ for the ternary case. If it is set too large, mainly overgeneral classifiers occur at the beginning and the GA has to push the population toward an appropriate degree of specificity. On the other hand, if it is set too small and the maximum population size N is set too small as well, then XCSR might not able be to construct a complete state-action-reward mapping $X \times A \rightarrow P$ for large input spaces. Under the assumption of uniformly distributed input data, his phenomenon appears when covering is continually active because the input space cannot be completely covered. This might lead to the situation that new classifiers are continually inserted into $[P]$ although N is already reached. This in turn results in a continuing replacement of other classifiers covering different niches in the problem space. This trap is also regarded as the *covering-deletion cycle* which in the end causes *detrimental forgetting.* On the other hand, for learning problems with frequent changes in the payoff levels within the corresponding payoff surface $X \times A \rightarrow P$, however, starting from the overspecific side might be advantageous. This is because otherwise XCSR might not be able to capture the complexity of the underlying problem in terms of distinguishing the diverse payoff levels present in the possibly numerous niches. This is also called the problem's *modality* [UB17]. It becomes intuitive, that complex problems require a population size that is chosen sufficiently large to prevent the pitfall of the so-called *covering challenge.*

One might think, that the obvious solution is to set r_0 to a large value. Clearly, this would ensure full coverage of the input space, while keeping the necessary population size small. In this case, XCSR faces the second well-known trap called the *schema* or more expressively the *representation challenge.* Recall the discussion of the *mutation pressure* above. It was stated that mutation stresses the offspring classifiers toward specification in binary domains. This explicit specification pressure, however, does not apply for real-valued input spaces any more. Thus, specification purely occurs by chance. If the population is filled up with only overgeneral classifiers, XCSR will likely take a long time to evolve classifiers with conditions that identify the underlying structure correctly, i.e., specify toward the problem-dependent appropriate level. This is because no sufficient *fitness signal* can be provided to the GA. Without a sufficient fitness signal that distinguishes accurate from inaccurate classifiers, the GA mostly acts randomly and, thus, the learning progress stalls. The provision of a sufficient fitness signal is crucial to ensure effective solution growth and sustenance. The resulting stress toward accuracy is referred to as *fitness pressure.*

Modifications to the Genetic Algorithm Beside the covering mechanism, also the incorporated GA affects classifier conditions. Recombination (or crossover) now acts on the level of interval predicates instead of a string from the ternary alphabet. Theoretically, n-point crossover can split up the conditions of the offspring classifiers between interval predicates or even in-between them. The latter would imply that possibly not the entire interval predicate for a corresponding dimension is exchanged, but only one bound. This depends on the interpretation of what is actually encoded as a single *gene* within the *chromosome.* If it is desired to allow crossing the conditions between closed interval predicates, a gene would be defined as the entire interval predicate (l_i, u_i). Alternatively, any lower bound l_i and any upper bound u_i can constitute an individual gene. This results in the number of genes in the chromosome being doubled. Thus, the only noticeable change for crossing the offsprings' conditions is that a gene is not represented by a single symbol anymore, but rather by (1) one scalar value that represents a lower or upper bound from the interval predicate, or, (2) the entire predicate (l_i, u_i). However, if crossover in-between interval predicates is desired and OBR is used then the crossover mechanism has to ensure that $l_i \leq u_i$ is satisfied thereafter. If not the case, the bounds need to be switched. The combination of UBR with in-between predicate crossover appears to be the most exploratory variant. It arguably might happen that the recombination of two offspring conditions results in a new condition that covers a very different part of the input space which even not matches the current input σ_t anymore. It is to be assessed individually whether this behavior is desirable or not for the problem task at hand. The same rationale applies to uniform crossover, where for each predicate or else bound it is decided individually and with equal chance whether or not it is exchanged.

Mutation in XCS for binary inputs flips certain specified bits to one of the other two possibilities, e.g., 1 can be changed to 0 or #. Here, the individual genes of the chromosome (condition) are represented by the particular bounds. Thus, the number of possible alleles is theoretically infinite. To still allow the GA to permute the conditions of the offspring classifiers, another hyperparameter has to be introduced: m_0 defines the maximum deviation from the starting value before mutation. Accordingly, for each gene in the chromosome the following mutation rule is applied with probability μ:

$$l_i, u_i \coloneqq l_i, u_i \pm \text{rand}[0, m_0], \quad \forall i \tag{2.36}$$

Again, $\text{rand}[0, m_0]$ yields a uniformly distributed random number between 0 and m_0. The decision whether the quantity is added or subtracted is also felt probabilistically with equal chance. As for crossover, under the utilization of OBR, $l_i \leq u_i$ has to be satisfied for all $i = 1 \ldots n$. Accordingly, for UBR, this validity check can be omitted.

As a last place, the GA subsumption is affected by the real-value representation of classifier conditions. The prerequisite that a subsuming cl_1 has to be more general than a potentially subsumable cl_2 is now given by

$$cl_1.C.l_i \leq cl_2.C.l_i \land cl_1.C.u_i \geq cl_2.C.u_i, \quad \forall i = 1 \ldots n.$$

This completes the discussion of necessary modifications to allow XCS to be applied to real-valued input spaces. The intuition of the inner working principles of XCS have now been introduced. Before discussing another special-purpose derivative designed for the task of function approximation, a brief overview of the theoretical advances in the field of XCS shall be provided.

2.4.3. Theory of Generalization in XCS

The two fundamental changes to prior versions of LCS, i.e., basing the fitness of classifiers on their prediction accuracy and the shift from a panmictic to a niche GA have paved the way for a new era of LCS research that is still progressing. But which rationales have led Wilson to make these changes? What was his fundamental hypothesis in this regard?

In [Wil95], Wilson elaborates that the goal of designing XCS is to learn accurate and complete state-action-payoff mapping $X \times A \rightarrow P$ (also termed *payoff landscapes*). At the same time the system should be prevented from creating overgeneral rules that eventually take-over the population due to reward sparsity in starving niches. It has been found, however, that XCS is not only able to evolve maximally accurate but at the same time maximally general rules. These two objectives, that seemed somewhat contradictory at that time, can be achieved due to interacting *learning pressures* exerted by the newly introduced concepts.

Wilson summarized that observation qualitatively in his *generalization hypothesis*:

He suggested to consider two classifiers cl_1 and cl_2 with identical actions. cl_2 is assumed to be a generalization of cl_1, meaning that cl_2's condition matches more inputs than the counterpart of cl_1. Further it is assumed that both classifiers have an equal prediction error, i.e., $cl_1.\epsilon = cl_2.\epsilon$. Each time both classifiers are members of the same action set $[A]$, their fitness values $cl.F$ are updated by the same amount according to 2.32. Wilson further realized that, due to the more general condition of cl_2, it will appear in more match sets than cl_1. Thus, cl_2 is supposed to get more *reproductive opportunities* due to the applied niche GA (at that time, the GA was applied on $[M]$ rather than $[A]$, but this was revised three years later [Wil98]). In conjunction with GA subsumption, the increase in cl_2's numerosity is expected to outnumber the numerosity of cl_1 over time. This, again referring to 2.32, would lead to a higher fitness share for cl_2 than for cl_1 when both are members of the same $[A]$. Speaking in microclassifier terms, cl_2 is assumed to have more offspring classifiers subsumed in the course of learning than cl_1. Thus, a higher portion of the shared accuracy κ is allocated to cl_2. This in turn increases the chance for being selected and reproduced by the GA compared to the probability of cl_1. According to Wilson, this is assumed to finally results in cl_1 being superseded in $[P]$ by cl_2 itself or its offspring classifiers.

The process of generalization is expected to continue until a further increase in generality of the classifiers stemming from cl_2 would result in an accuracy drop, or else to an increase of their prediction errors. The criterion of maximum accuracy is controlled by the parameter ϵ_0 (cf. 2.30). These interactions stress the population toward comprising classifiers that are maximally accurate and maximally general at the same time.

Shortly after the formulation of Wilson's generalization hypothesis, Kovacs extended it toward the so-called *optimality hypothesis* [Kov98]. He showed for a certain class of single-step problems (i.e., the multiplexer problem) that XCS is not only able to evolve an accurate and maximally general solution, but also a *complete* and *minimal* one, denoted by $[O]$.

A few years later, the qualitative hypotheses around XCS's generalization capability have been analytically examined by Butz et al. in [But+04a]. In their work, the learning biases of XCS were separated into several *evolutionary pressures* whose interaction forms the basis of the learning process. Wilson's initial generalization hypothesis has carried on and was termed *set pressure*.

Butz in [But05a] analytically derives several bounds that need to be satisfied in order to ensure XCS to learn appropriately. Initially, these bounds have been derived following the assumption of binary input domains. Parts of the facetwise theoretical results have been transferred to real-valued problem domains by Stalph et al. in [SB10a; Sta+12a].

Further Theoretical Advances in XCS Research

A further XCS derivative specifically introduced for supervised learning tasks, termed *Supervised Classifier System* (UCS) [BG03], has been investigated formally regarding its competence on learning in imbalanced domains [OB06; Orr+07]. Drugowitsch pursued a rigorous formal approach to model LCS from first principles following a probabilistic ML perspective [Dru08; DB08]. Recent theoretic work aims at figuring out optimal hyperparameter settings in binary problem domains [Nak+17; NBH18] enabling XCS to cope with very complex instances of the well-investigated multiplexer problem.

Since the advancement of theoretical insights about the underlying working principles of XCS is not the main goal of this thesis, a more thorough elaboration on the current state of theory is included here. A sensible starting point to become familiar with the theory behind LCS can be found in [But05a]. A recent comprehensive survey on formal theoretical advances in XCS research is provided in [PSH19].

2.4.4. XCS for Function Approximation

So far, both XCS and its real-valued extension to XCSR have been introduced as techniques for single- and multi-step RL problems. As already outlined at the beginning of this thesis, XCS is a very flexible system that allows to be applied to various problem domains with only minor changes in its algorithmic structure.

Accordingly, in 2002 Wilson proposed a modification to XCSR in order to allow for a piecewise online approximation of functions defined over the reals [Wil02] – *XCS Classifier System for Function Approximation* (XCSF). Since XCSF serves as reference system later in this thesis, the following paragraphs describe the necessary algorithmic and architectural modifications to XCSR in order to yield a learning algorithm capable of regression.

Algorithmic and Architectural Modifications to Reach XCSF

XCSF, in contrast to conventional XCS variants, strives to approximate functions in a piecewise, i.e., local fashion. No action space is assumed which reduces the payoff landscape to $X \rightarrow P$. As before, a single classifier cl comprises all of the ordinary parameters. However, for XCSF, only one so-called *dummy action* a_d is defined, yielding a pseudo action space $A = \{a_d\}$. Of course, the action attribute could also be omitted, but it is often still mentioned in the literature to adhere to the conventional XCS structure. Another important modification that was introduced concerns the payoff prediction scalar $cl.p$. It is replaced by a prediction model, initially defined to be a first-order polynomial $cl.p(\vec{x})$, i.e., a linear function

$$cl.p(\vec{x}_t) = (\vec{x}_t^* - \vec{c}^*)^T \vec{w}. \tag{2.37}$$

Here, $\vec{w}$ is defined as an $n+1$ dimensional weight vector $(w_0, w_1 \ldots, w_n)$. w_0 determines an offset (or bias term). $\vec{x}_t^*$ and $cl.\vec{c}^*$ represent an extended input vector and the extended center point of a classifier's condition $cl.C$, respectively. The extension constitutes a leading 1 and 0, respectively. The subtraction of $cl.\vec{c}^*$ constitutes a translation of the input (vector) space. It sets the origin to the center point $cl.\vec{c}$ of a classifiers condition $cl.C$. This translation is not mandatory but has been found to yield beneficial effects.

Instead of estimating a scalar value for the predicted payoff $cl.p$ as before, in XCSF a weight (or coefficient) vector $cl.\vec{w}$ is stored and updated as follows [BLW08]:

$$\vec{w} \leftarrow \vec{w} + \eta \left(y - p(\vec{x}_t)\right) p(\vec{x}_t) \tag{2.38}$$

According to Equation 2.38, the weights are updated using a slight modification of the Widrow-Hoff rule. Since the resulting search space for finding optimal weights for a linear model and given a certain loss function yields a uni-modal optimization surface, gradient-based techniques were found to be the adequate choice [Wil02]. Accordingly, the η hyperparameter serves as step size (or learning rate) for the gradient-descent procedure. The remaining classifier parameters, however, are still updated with the conventional update rule using the β learning rate.

The number of weights to be learned varies dependent on the degree of the polynomial used for the local approximation of the underlying target function (cf. [Lan+05b]).

For the sake of simplicity, the following elaborations focus on the linear case. The semantics of the remaining classifier attributes is analogous to standard XCS. Since now, only one single dummy action a_d can be selected, the prediction array reduces to a single variable. Thus, the match set $[M]$ always equals the action set $[A]$. Eventually, this makes $[A]$ obsolete. The obsoleteness of $[A]$ also implies that an environmental niche is now approximated by the according match sets, and the GA is therefore applied on $[M]$.

The system prediction $PA(a)$ still calculates as the fitness-weighted sum of the matching classifiers predictions. The predictions, however, are now so-called *computed predictions* that are obtained by passing the current input vector $\vec{x}_t$ as an argument to the locally defined polynomial (Eq. 2.37) which is determined by the current *model parameters* (or coefficients) stored in $cl.\vec{w}$.

$$PA(a_d) = \frac{\sum_{cl \in [M]} cl.p(\vec{x}_t) \cdot cl.F}{\sum_{cl \in [M]} cl.F} \tag{2.39}$$

It is locally defined, since it only accepts input vectors that fall in the range of matching classifiers' conditions $cl.C$. Accordingly, the underlying function is approximated in a piecewise fashion.

Function approximation falls under the (supervised) learning task of regression. Thus, the output of XCSF needs to be a continuous value (a scalar) $y \in \mathbb{R}$ from the co-domain of an unknown function $f(\vec{x})$ to be approximated. To realize that, for each incoming situation $\sigma_t = \vec{x}_t$, XCSF directly outputs the system prediction $PA(a_d)$ and receives an according feedback signal in return. More precisely, no designated reward function that delivers an immediate reward r_{imm} is needed anymore. Instead, simply the actual function value $f(\vec{x}_t)$ is returned. This allows for the calculation of the error measure $cl.\epsilon$ as usual. It is assumed that in the succeeding time step $t+1$ at the latest the actual function value of the preceding input vector $\vec{x}_t$ becomes available.[8] With the actual $f(\vec{x}_t)$ at hand, XCSF calculates the absolute error of the predicted payoff $P(a_d)$. This absolute error is then used to update the matching classifiers' parameters $cl.\epsilon$ and $cl.\vec{w}$.

With that, the most important modifications to XCSR have been outlined. The aspect of prediction modeling will be revisited again in Chapter 9 when it is replaced by a interpolation-based approach.

[8]In the context of SLAS this assumption is indeed reasonable. Consider learning task where the utility surface should be modeled and the assessment of just realized adaptations takes a certain amount of time (an evaluation cycle) to capture the impacts on the system utility.

Major Extensions to the Initial XCSF

For the sake of providing a comprehensible understanding of the inner workings of XCSF, so far the elaborations on the algorithmic part have been narrowed down mostly to the rudimentary form proposed in [Wil02]. However, since its first proposal, plenty of research was concerned with improving the performance and capabilities of XCSF. In the following paragraphs, an overview of the current state of the art regarding XCSF will be provided along with the corresponding references.

In [But05b], Butz presents a sophisticated approach for classifier condition representation, which impacts the partitioning of the input space. Instead of the conventional hyperrectangular representation as already described before, he proposes to modify the geometric shape of classifier conditions to hyperspheres. Especially in smooth functions with a certain degree of curvature, the corners of hyperrectangles might cause higher prediction errors. In the same breath, Butz extends the hyperspherical conditions to hyperellipsoidal conditions. This allows for distinct stretches in each dimension of the input space. So far, however, these hyperellipsoidal conditions are limited in their orientations. More precisely, only axis-parallel conditions can be encoded. In order to overcome this limitation and to deal with the issue of oblique functions, Butz et al. further enhanced the hyperellipsoidal condition representation to allow for explicit rotations of the evolved hyperellipsoids in [BLW06]. Figure 2.12 shows an exemplary population of classifiers using this general, rotating hyper-ellipsoidal condition representation.

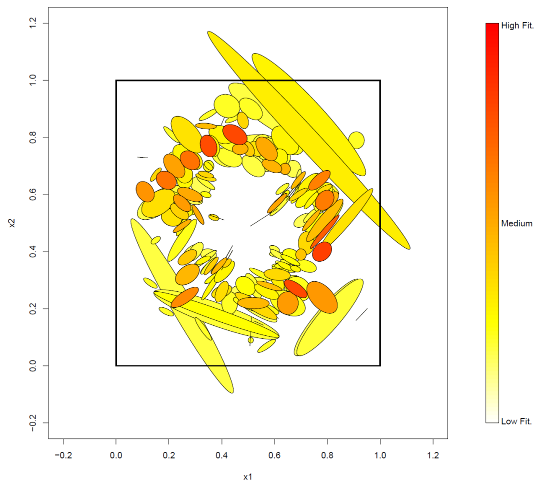


FIGURE 2.12.: Exemplary population of classifiers evolved by XCSF employing rotating hyperellipsoidal conditions

As can be seen, the ellipses are not restricted to the input space bounds, since otherwise, the bounding regions can not be captured appropriately by the classifiers.

In [BLW08], this condition scheme is further enhanced. The so far utilized multivariate Gaussian kernel and the according covariance matrix used to encode the hyperellipsoidal condition is replaced by an explicit angular representation. This avoids disruptive effects during the GA-driven evolution. Various investigations confirmed the beneficial effects of the general, rotating hyperellipsoidal condition representation. The revealed advantages are attributed to the ability of XCSF to much better capture the complexity of the underlying functions to be approximated.

Another major modification to Wilson's initially proposed XCSF is the replacement of the incremental update procedure for the coefficient vector $\vec{w}$. In [BLW08], the authors incorporated the *Recursive Least Squares* (RLS) method which, among others, has been initially applied to XCSF in [Lan+05a]. The RLS update procedure, which can be regarded as an incremental version of *linear least squares* technique, yields highly significant improvements in terms of learning speed and prediction accuracy. Further update algorithms have been examined: For instance *gain adaptation* and the utilization of the *Kalman filter* in [Lan+06]. *Neural Networks* have been applied in [LL06]. [LML07] reports on experiments with *Support Vector Regression.* And also *Evolution Strategies* [TSD08] have been explored within XCSF. All of these methods have been found to exhibit individual strengths and weaknesses with regard to several functions with different degrees of complexity. From a comprehensive literature review it becomes apparent that RLS has been most often used in reported experiments. This is due to the overall competitive or rather superior performance of this technique compared to alternative approaches that have been tried out over time. For details on RLS and comparative studies, the interested reader is referred to [Lan+05a; Lan+06; BLW08].

In [Lan+05b], Lanzi et al. investigated polynomials beyond linear functions to accomplish the local approximation of functions with XCSF. For complex functions, higher-order polynomials up to a degree of three have been investigated. They have been found to improve both the accuracy and the generalization capability what in turn results in more compact final solutions. In Chapter 9 a novel concept for approaching local predictions based on interpolation is presented and compared to polynomials up to the third degree.

In [BLW08], the authors further introduced a novel compaction strategy. This *greedy compaction* methodology in conjunction with a *closest classifier matching* (CCM) procedure can drastically decrease the size of the final population. With this approach, XCSF has been found to be able to decrease its population size by 90% on average. This is accompanied by only a marginal increase regarding the system error. Approaches to *condense* the final population have already been proposed by Wilson in [Wil95; Wil98]. Those early variants of compaction essentially apply the GA frequently at the end but with mutation and crossover inactive. This is intended to strictly reproduce the fittest classifiers in their corresponding niches in order to allow for subsumption and niche takeover. Since the focus of this thesis is not directly set on the comprehensibility of the final population in terms of their compactness but rather on the robustness against gaps in the rule-base, the impacts of post-hoc compaction are not investigated further.

Another extension introduced by Butz et al. in [BS12] is intended to counter the problem of detrimental forgetting effects when the function domain is sampled non-uniformly. The niche-based reproduction coupled with population-wide (panmictic) classifier deletion can cause sparsely sampled niches to be “forgotten” due to a lack of reproductive opportunity and niche support. The *local deletion* mechanism is proposed to alleviate this effect. As concluded by the authors, neither significant improvements nor adverse effects could be observed during the online learning phase. In view of the application of compaction techniques, however, local deletion was found to yield more stable final solutions in terms of accuracy.

In their work reported in [SB12], Stalph and Butz introduced the concept *guided evolution* to XCSF. Each classifier is extended to hold a set of past samples it has matched. On the basis of an accuracy-weighted covariance matrix, the devised *guided mutation* operator is able to identify dimensions with lower, or else higher impact. In consequence, XCSF is able to steer the evolution of the condition structure accordingly. Presented results show that guided evolution leads to a strongly increased learning speed. Additionally, this technique enables XCSF to approximate higher-dimensional functions that used to be rather intractable for standard XCSF so far.

In this section, the fundamental ideas and the algorithmic structure of XCS have been thoroughly discussed. Necessary modifications to “vanilla” XCS that allow for handling real-valued inputs have been explained. Afterward, a basic theoretic understanding what causes the generalization in the system has been conveyed. Lastly, a further extension to XCSR that enables the system to accomplish regression was the subject of discussion. The conducted modifications to reach XCSF as well as an overview of the most essential extensions have been mentioned. In Section 3.4 of the next chapter, the potentials and drawbacks of XCS when incorporated in SLAS will be discussed. Furthermore, a strongly modified XCS variant, called *XCS Classifier System for the Observer/Controller architecture* (XCS-O/C) will be outlined that has been designed to overcome most of the present drawbacks in this regard.

2.5. Chapter Summary

Throughout this chapter, the necessary background information regarding concepts and techniques used in the remainder of this thesis have been introduced. Starting from a brief introduction and motivation of the OC initiative, fundamental paradigms and terms of the field of ML have been explained. Subsequently, the domain of multivariate scattered data interpolation was the subject of discussion. Concrete approaches from the classes of global and local interpolation have been described along with a discussion on their suitability. The last part of this chapter was then dedicated to the field of Michigan-style LCS in general and the XCS algorithm in particular – the central ERBML approach under investigation in this thesis. Based on the now provided prerequisites, the next chapter brings these concepts in line with each other and eventually derives the central problem statement.

Chapter 3.

Problem Statement

This chapter is intended to delineate the problem domain into which the research topic of this thesis integrates. Starting from an OC point of view on SLAS, a system model is derived afterward in order to provide a basis for the actual problem statement – the existence of *Knowledge Gaps* (KGs) in *Self-Learning Adaptive Systems* (SLAS) systems. This chapter closes with a discussion of why LCS constitute viable candidates for dealing with the stated research problem.

3.1. Learning in Organic Computing Systems

OC systems approach the complexity challenge by means of transferring design-time decision to the runtime in order to achieve a flexible and robust operation in spite of continuing change. Therefore, the systems need to be self-organizing and self-adaptive. This latter property can be achieved by equipping OC systems with the runtime ability to self-learn from experiences, e.g., control decisions and the resulting utility changes. During its runtime, an OC system is continually required to adapt to occurring changes in its typically *Non-Stationary Environment* (NSE). These systems therefore steadily observe the current system state. State information is derived through gathering raw sensory data in order to perceive the environmental conditions as well as considering further internal parameters of the observed system. As a response to detected changes, OC systems have to react appropriately and within a certain time. One possibility how these *reactions* can be brought into the systems is by the designers which rely on their expert knowledge. However, this necessitates for a sort of “omniscience” regarding all possible system states that might appear in the course of the system’s lifetime. Unfortunately, this assumption is usually not plausible for complex interconnected systems deployed in dynamic real world environments. Therefore, a second way of acquiring appropriate reactions for changing conditions is *reaction learning* [Ste17].

The OC way to build systems with the desired self-adaptation capability is to equip the productive systems (SuOC) with a *control mechanism* that provides a complexity reduction for the human operators. They shall be relieved by means of only specifying what is to be achieved and not how exactly this should be done.

For the purpose of facilitating self-adaptation through learning in OC systems, a generic architectural blueprint has been devised [Ric+06; Tom+11b] – the O/C architecture. Figure 3.1 illustrates the most generic variant of this pattern.

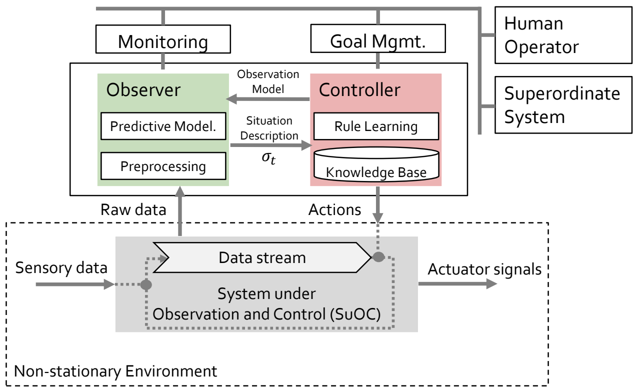


FIGURE 3.1.: The generic Observer/Controller architecture.

At the bottom level, the underlying *System under Observation and Control* (SuOC) is encapsulated. It is expected to be instrumented with particular *sensors* to *perceive* the current conditions of the surrounding *productive environment*[1]. Furthermore the SuOC is typically expected to be equipped with *actuators* (or effectors) in order to allow for alterations within its productive environment. In order to facilitate data exchange with the desired control mechanism, here realized by an *observer/controller* tandem, the encapsulated SuOC itself further has to provide interfaces for: (1) Granting access to its internal parameters and the gathered raw sensor data reflecting the environmental conditions. (2) Passing control signals (actions) from the control mechanism back to the SuOC that initiate a reconfiguration of its controllable parameters or actuation of its effectors. Together the internal and the environmental conditions constitute the *raw data* which can be provided by the SuOC and which can be directly observed by the control mechanism's layer set on top.

The continual perception of internal states and externally measured environmental conditions, enriched by the applied control actions is illustrated a *data stream DS*. A *feedback loop* is established from which the applied learning algorithms have to build their (predictive) models. Since the SuOC is assumed to be applied in an NSE, certain challenges are imposed on this data stream which will be defined more clearly in the system model derivation (cf. Sect. 3.2).

[1]With "productive environment" the relevant and measurable part of the environment in which the technical system is deployed is meant. Typically, the surrounding environments are closed and there exist hidden conditions that cannot be observed through sensors.

The control mechanism is designed as follows: An *observer* is set on top of the SuOC to continually perceive the current conditions through the raw data provided. The observer is intended to generate an appropriate abstracted situation description at certain points in time t. The terms (system) *state* and *situation* are used interchangeably in the following and will be denoted by σ_t. This abstraction might comprise further measures derived from the incoming stream of data. For example, system sate forecasts can be conducted [SSH16a; SSH16b] or dimensionality reduction techniques can be applied to select the most expressive data features. Also anomaly detection algorithms are imaginable that attempt to detect changes in the underlying data generating processes [GS16]. As becomes apparent, the observer part of the O/C pattern constitutes a place to accommodate *Unsupervised Learning* (UL) techniques.

The observer then reports σ_t to the *controller* component which on that basis decides on the necessity for a reactions or else control *actions* a_t. In case that the controller decides to intervene, how does the it know which control actions are most appropriate to be applied? A crucial ingredient for ML in general is an adequate performance measure (recall Mitchell's definition as stated in the introduction). In OC terminology, such a measure of performance is provided by a *utility function* u. It provides a feedback (or learning) signal based on which the system can evaluate the "appropriateness" of the action applied at a current state. This utility function comprises the system's goals, which in turn are passed to the control mechanism via a *goal management* interface by the *human operators* in the simplest case. Assuming system hierarchies, also a *superordinate system* working on a higher abstraction level can cause goal changes. [2]. The controller is responsible to evaluate the utility of its performed adaptations. It therefore takes the next delivered situation description into account and compares the current utility level to the previous one. With these ingredients, the feedback loop is finally established.

A further designated component in the generic O/C architecture is the *monitoring* interface. It is designed for the sake of allowing an inspection of the current system performance and operation. It therefore facilitates system transparency in view of the important self-explanation property of OC and general AI-based systems.

Another question that should be answered is how the observer knows which type of system state abstraction to build. Considering a vast amount of raw sensory data due to high sampling of the observed data stream, the observer has to select an appropriate window of sensory data which is to be included. Another decision would be on which part of the incoming data to concentrate. Different goals might render particular sensor data irrelevant. A so-called *observation model* which is determined

[2]Although beyond the scope of this thesis, it should be noted that this basic O/C pattern can be applied in a hierarchical fashion to build even more complex OC systems. Goal-oriented Holonic Architectures [Dia+16] are an example where such a setting can be imagined. In this thesis, however, it is focused on a single-context SLAS. Nevertheless, the developed techniques can operate on any level of abstraction given a well-defined context and learning task.

by the controller is responsible for felling such decisions. Such an observation model should be flexible and, thus might be replaced by a refined one depending on changed goals for instance.

Example. *An imaginable scenario in the introduced traffic management system example would be the compliance to newly enacted legal constraints. For instance, instead of striving for reducing the traffic noise in housing areas by means of lower speed limits, now minimum pollution in the same area is pursued. These are contradicting goals which demand for different sensory data on which the system should set a stronger focus. Also the evaluation window for assessing the satisfaction of the prescribed goals (the system's utility) might appear different.*

As of yet, the learning task of an OC system involves the following ingredients:

1. Continually observed *system states* σ_t at discrete points in time t.
2. Selected *control actions* a_t to adapt when necessary.
3. A *utility function* u which assesses the degree of appropriateness with regard to the fulfillment of the system's goals.

In Section 2.2, the notion of RL has already been introduced. The aforementioned learning setting of controlling the SuOC via actions learned and selected by controller on the basis of perceived states perfectly fits the RL paradigm of ML. As briefly outlined above, ML can also be applied directly within the observer, e.g., SL for predictive model building (i.e., utility function approximation) or for unsupervised dimensionality reduction purposes. Later in this chapter, it will be abstracted from specific learning tasks. The RL formulation will then be reduced to a SL notion in order to derive the unifying system model for this thesis. For now it is focused on the probably most intuitive system control task in view of applying ML in OC systems.

In the following, the control task of an OC system is formulated by a *Markov Decision Process* (MDP):

- The system state abstractions σ_t stem from a space of possible states S
- The realizable control actions a_t are determined by an action space A
- A transition function $\tau : S \times A \to S$ determines the system state transitions after realizing actions
- A utility function $u : S \times A \to \mathbb{R}$ serves as reward function to provide the learning entity with necessary feedback

Parts of the state space S are determined by the value ranges of the sensors for perceiving productive environment with which the system is equipped. The spanned space might be modeled discrete up to a certain degree, e.g., due to the numerical precision of values the sensors deliver. However, it can also be expected continuous which demands for learning algorithms capable of that. The state space can also be enriched with observer derived measures or current conditions from within the system. The final system state description σ_t on which the applied learning algorithm is eventually making decisions is thus determined by the observer component. It has the ability to analyze and preprocess the sensory data, to extract more descriptive or reduce to relevant features on that basis, and even to forecast future states to extend the abstract system state description σ_t.

Based on the actuators with which the SuOC is equipped and the internal control parameters which can be directly accessed and adjusted by the controller, the action space A can be derived. The controller's tasks are to self-learn and to decide on the most adequate control actions at any given state σ_t delivered by the observer. These actions can be encoded as abstract control signals that might need to be translated into applicable actuator signals. This translation, however, is not considered to be a part of the learning problem here. Depending on the number of the SuOC's control variables and actuators, the size of the action space can become large or even theoretically infinite. Often predefined discretizations are necessary in order to successfully apply RL algorithms. One alternative way to deal with that issue would be to employ an offline learning layer which optimizes particular control action solutions out of a possibly large or continuous action space by means of simulation. The extension of the O/C architecture toward a variant capable of that will be introduced shortly. In Chapter 7 a concrete example of that methodology is outlined.

Usually, realized control actions for alterations of the controlled system might yield differing sensor measurements afterward. This in turn might yield a changed system state $\sigma_t + 1$ at the next time of measurement $t + 1$ due to the observer. This transition from a current state σ_t to a subsequent state σ_{t+1} after performing action a_t is usually unknown. The transitions also might be subject to stochastic influences which cannot be determined a priori at design time. An explicit modeling of such an unknown state transition function τ, more precisely of an estimated probability distribution over the succeeding states σ_{t+1}, is a designated subtask of model-based RL algorithms. Such approaches can be valuable in real world scenarios, where the exploration of the system needs to be restricted. With models of the environment, i.e., for the transition τ and the reward function u, the trial-and-error behavior can be replaced by performing planning. Simulation-based or *Anytime Learning* [GR92] is a related concept which is followed by the O/C architecture extension which will be introduced below.

As already described, the utility function u represents the system goals to be achieved. These goals can be multifaceted in terms of involving a number of subgoals. These subgoals can be contradictory what leads to the problem of multi-criteria optimization. The utility function is assumed to be composed of the individual subgoals here. Priorities among those subgoals could be reflected by applying different weights for instance. When the system goals change, the current weighting might change or even further subgoals might be added. This results in changes regarding the utility surface (also called *fitness landscape* in the OC context). Again model-building algorithms could be applied to approach this issue. However, they need to be capable of OL in order to detect and quickly adapt to the occurring changes. The relevance of the ML branch of *concept drift adaptation* becomes apparent in this regard.

Remark In the preceding paragraphs, essential elements and aspects of modeling an OC system's learning task have been discussed. For instance, the importance of the utility function u and the meaning of a state transition function τ. The latter component maps certain states $\sigma \in S$ and a correspondingly selected actions $a \in A$ to a succeeding state $\sigma_{t+1} \in S$. With such a deterministic state transition function as defined above, it is implicitly assumed that the succeeding state σ_{t+1} only depends on the previous state σ_t and the executed action a_t. Since the observer continually creates system state descriptions σ_t for periodic adaptation cycles of the control mechanism, the time domain is assumed to be discrete. For example, in the self-adaptive traffic light control system, each action, i.e., signal plan adaptation, necessitates a certain activation interval. This could be two complete cycles through the traffic light phases and interphases for instance for instance in order to allow for an appropriate evaluation of the utility gain (in this example a decreased average delay at the observed intersection) or utility drop. The more general assumption that σ_{t+1} depends only on σ_t is known as *Markov property* in the literature [SB98]. More precisely, σ_t is actually required to comprises all necessary information of past state transitions which might become relevant in the future. When furthermore the action is considered in the state transition, as assumed for the OC learning scenario as introduced above, this is referred to as MDP or *controlled Markov process*.

3.1.1. The Multi-Layer O/C-Architecture

In order to continually self-optimize, SLAS need to *explore*. At least at early stages or whenever changes occur, new knowledge elements which sufficiently cover the current problem space have to be acquired by the systems. Without sufficient exploration, the systems probably get stuck in local optima of the fitness landscape determined by the utility function u. Unfortunately, exploration implies *trial-and-error*. This refers to the unrestricted realization of random actions which never have been tried before in the same situation. Such an exploratory behavior might either yield low system utility or can even cause damage regarding the system itself or other involved parties. In the worst case, free exploration can lead to injury of human participants. Consider just an industrial robot having a heavy object attached and which freely discovers different kinematic trajectories in its operation area. In order to cope with this exploration-exploitation dilemma (cf. Sect. 2.2), the generic O/C architecture discussed before can be extended toward the *Multi-Layer Observer/Controller* (MLOC) architecture as depicted in Figure 3.2.

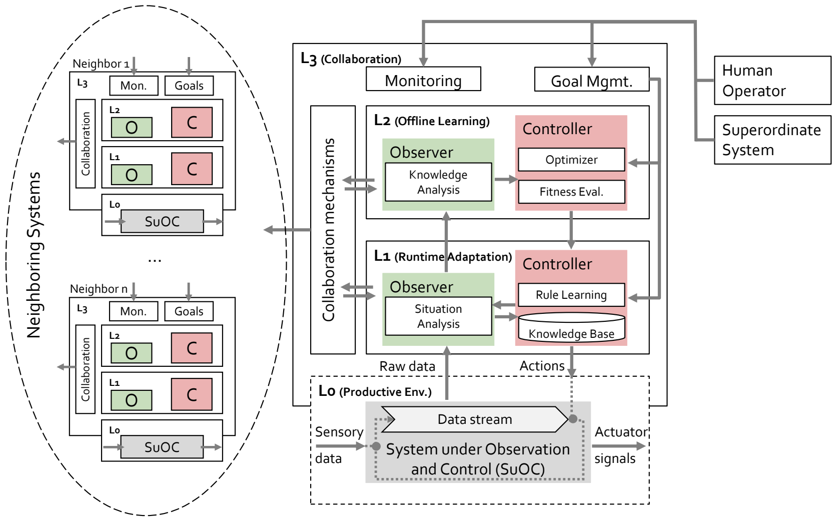


FIGURE 3.2.: Schematic of the Multi-layer Observer/Controller architecture

This multi-layered variant stacks two distinct layers each with a dedicated observer and controller component on top of the SuOC. It additionally defines an overarching third layer for collaboration and communication with external authorities and neighboring systems. The individual tasks of each layer can be summarized as follows:

Layer 0 - Productive System At the lowest zeroth layer, the SuOC placed within its productive environment is encapsulated. The layer comprises the adaptable managed resource, which might be either a physical system or a pure software-based system, as well as the surrounding environment. Examples for the letter are weather conditions or participating humans in cyber-physical systems, or the software ecosystem in which an self-adaptive software component is embedded. As before, the communication paths to the superordinate O/C tandem are realized by means of well-defined interfaces which allow for gathering sensory data (observation) and realizing control actions (control). Together the SuOC and the surrounding environment constitute an NSE. Such a dynamic learning environment is expected to continually challenge the involved self-learning mechanisms and, thus, the entire SLAS with unforeseen situations. These might be caused by internal issues (e.g., component failure) or external disturbances (e.g., weather conditions or traffic accidents). Again, the periodically conducted observations, enhanced with the realized adaptation reactions (or control actions), constitute a data stream. The *runtime adaptation* layer (L1) is asked to learn from the data provided by this data stream. It directly follows the requirement for online or incremental ML algorithms as explained in Section 2.2.

Layer 1 - Runtime Adaptation Layer This *online learning* layer establishes the feedback control loop. The task of the observer L1-O is the same as before, i.e., analyzing the incoming raw data and deriving an abstracted situation description σ_t. The controller's (L1-C) task is still the determination of appropriate reactions. It conducts this decision making process based on the delivered situation description σ_t and on the basis of the so far acquired production rules stored in a dedicated *knowledge base*. The controller has often been realized by online rule learning systems, such as provided by XCS. The applied algorithm is expected to incrementally learn IF(system state)-THEN(control action) rules and to maintain a corresponding rule or knowledge base. As already described in more detail in Section 2.4, these rules constitute the system's knowledge and are incrementally refined toward higher predictive accuracy and maximum generalization over the system's state space.

Layer 2 - Offline Learning Layer The second O/C layer is designed to observe and analyze the behavior of the lower runtime adaptation layer and to enhance the knowledge base of the reactive learning component situated at L1-C. In particular, it is responsible for reactive on-demand construction of novel rules whenever L1-C has no knowledge for the current system state delivered by L1-O. Therefore, the second-layer controller (L2-C) is equipped with (1) an arbitrary black-box optimization heuristic, and, (2) with a computational (simulation) model that simulates the behavior of the productive system (L0). It thus provides a "sandboxed" environment without direct impact on the productive SuOC. With this layer 2, the MLOC-based SLAS is enabled to learn *offline* by means of creating optimized (not necessarily optimal) rules via simulation for instance. Depending on the incorporated optimization method, this reactive knowledge construction can demand for a vast amount of computation time. Thus, depending on the time-criticality the optimized rules are usually not expected to be immediately available but only after a certain optimization time. These knowledge elements can then be utilized at one of the next occurrences of at least a similar situation for the first time. As initially outlined by Tomforde et al. in [TCH09], one particular aspect for improvement would be to leverage stand-by times of the offline learning layer in order to optimize the coverage of the problem space. In the same work, another aspect has been mentioned as part of the presented research roadmap, namely the use of "[...] intelligent inter- and extrapolation mechanism(s)[...]" in order to *increase the efficiency* of the rule creation process. In this thesis, both aspects are explicitly targeted. Initial methods for *interpolation-assistance* and *proactive knowledge construction* for LCS will be developed in the following chapters.

Layer 3 - Collaboration Layer Layers 0 to 2 define an autonomously learning SAS with a single-context, i.e., local, scope. In order to allow for cooperation and collaboration among neighboring SLAS as well as to provide a clearly defined interface for user access, a third, so-called *collaboration layer* exists. The dedicated interfaces for *monitoring* and *goal management* have the same meaning as before.

Again, enabling external authorities to change system goals at runtime requires what is called *flexibility* in the OC context. This refers to the capability of the system to automatically maintain or at least effectively recover to acceptable utility levels in spite of structural changes in the system's underlying fitness landscape due to goal modification. Additionally, a further *collaboration* component is included for communication and negotiation purposes between neighboring systems. This shall allow for a self-organizing collective achievement of the system's goals which a single entity can not or only hardly manage. An example would be the establishment of progressive signal systems in order to pursue green waves and accordingly to achieve far reaching continual traffic flows.

The MLOC architecture as depicted in Figure 3.2 again constitutes a generic blueprint. It can be easily extended to incorporate further O/C layers and individually equipped with capable learning and optimization techniques for which the problem at hand demands.

As follows from the previous elaborations on the individual layers' responsibilities, the exploration-exploitation dilemma can be approached by moving the exploratory knowledge discovery behavior to the offline learning layer L2. The runtime adaptation layer L1 can then simply keep exploiting sufficiently reinforced knowledge. It is also reminiscent of Grefenstette's *anytime learning* approach [GR92].

This type of restricted knowledge exploration [Tom+11a] has already been applied a couple of times and in combination with XCS as particular rule learning component at L1-C. The modifications to XCS that have been found necessary in this regard will be discussed in the following.

3.1.2. XCS in Organic Computing Systems

In the following, one particular modification of XCS toward an OC variant denoted by XCS-O/C will be presented. This modification allows for restricted online learning and exploration has been especially designed to fit the MLOC scheme presented above. In essence, two crucial modifications have been introduced (cf. e.g., [Tom+11a; Pro+08]):

1. The steady-state niche GA is removed from the conventional main-loop and replaced by an evolution strategy [BS02] which serves as the optimization heuristic situated at L2-C. It is utilized to figure out optimized actions by using the sandboxed offline simulation or other models of the environment.

2. The covering operator was modified to prevent the purely probabilistic creation of conditions and the exploratory random assignment of actions. Instead, it considers classifiers in a restricted proximity and builds upon the nearest neighbor's condition. It then creates a copy of this condition and widens it to encompass the so far uncovered situation.

Figure 3.3 integrates the three main components of XCS (see Sect. 2.4) with the generic MLOC architecture.

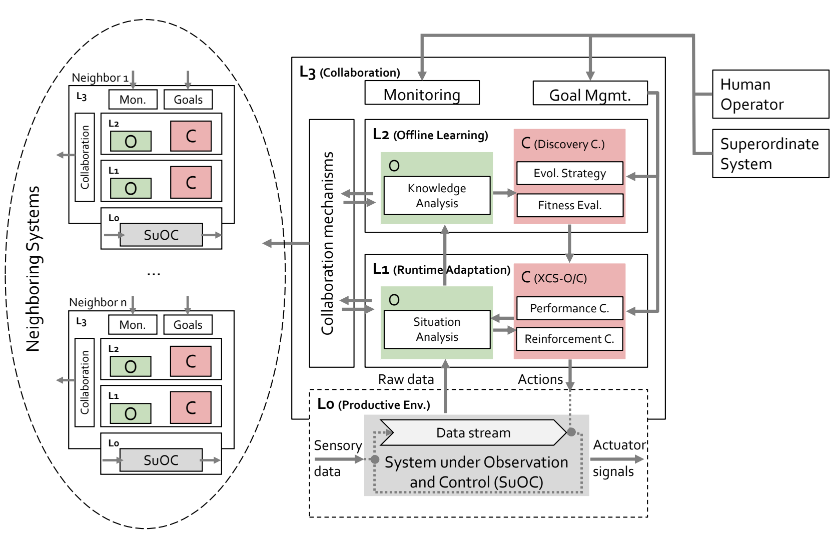


FIGURE 3.3.: XCS-O/C integrated with the Multi-layer Observer/Controller architecture

The performance, reinforcement and discovery components are spread among the controllers of layers 1 and 2. XCS's performance and reinforcement component are positioned at the runtime adaptation layer 1 on top of the SuOC. Thus, whenever a reward signal is evaluated by the controller, the active classifiers in the action set are adjusted as usual. Also the conventional main-loop of XCS (performance component) is situated at L1-C. Matching, the calculation of the prediction array as well as the action-selection regime are applied as for standard XCS. The discovery component, however, is outsourced to the controller of the offline learning layer L2. The standard steady-state niche GA is replaced by an generational evolution strategy which serves as the optimization heuristic at L2-C. The evolutionary discovery process is now invoked whenever the rule-base situated at L1-C contains no classifier that matches the current situation σ_t. Another reason for invocation is given when the modified covering operator cannot find a neighboring classifier within an acceptable distance. The operation of the evolutionary component of XCS is thus not periodical anymore, but rather occasionally happens on demand.

The exploration mechanism of XCS-O/C outsourced to L2-C essentially needs two components:

1. An optimization heuristic, such as the evolution strategy utilized here
2. A fitness evaluation method

The latter ingredient might be realized by a computational simulation as mentioned before, for instance. In the simplest, and least likely, case it can be realized by means of an a priori known model of the fitness landscape. Another option might be to use existing domain-specific mathematical or physical models or else approximation formulas which capture at least parts of the behavior of the underlying SuOC. Naturally, appropriate and available options for fitness evaluation highly depend on the specific application domain. In contrast to the GA involved in standard XCS which optimizes the input space coverage, the optimization component at L2-C seeks optimized actions. The final all-time best action candidate is then put into a new rule and evaluated against the same acceptance metrics as for the widening routine. If positively evaluated, the new rule is eventually passed to the knowledge base at the runtime adaptation layer, i.e., XCS-O/C's population. From this time on, the offline generated rule can be applied and directly affect the SuOC.

As can be imagined, a generational (or iterative) optimization algorithm with a fitness evaluation based on a computational simulation model can hardly produce solutions promptly at the time of request. Since the optimization is conducted offline, the productive system at layer 0 as well as the (online) runtime adaption layer remain responsive. To still assure immediate real-time reactions in spite of missing rules, the so-called *widening* covering mechanism has been developed. Its intuition is illustrated in Figure 3.4.

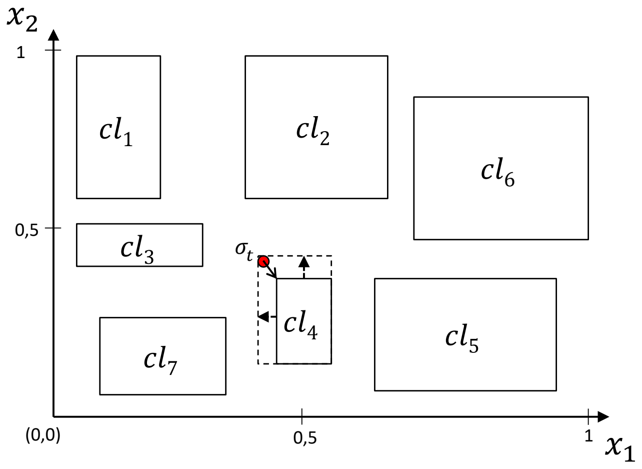


FIGURE 3.4.: The rule widening operation illustrated

It is activated as conventional covering, i.e., whenever the population does not contain any classifier matching the current situation. Within a certain radius determined by a predefined distance threshold, it seeks for nearby classifiers covering similar situations. The condition of the nearest classifier within the acceptance area is then copied and widened to barely include the current situation. In the case that no classifier falls within the accepted distance, layer 2 is triggered to optimize a rule offline. In the simplified setting depicted in Figure 3.4, the Euclidean distance is utilized as similarity metric. The action of the nearest neighbor is retained. The remaining classifier parameters can then be initialized by different means (for a concrete examples cf. Ch. 7). Ideally, if there exist approximation formulas that model the behavior of the underlying SuOC, the widened rules can be assessed for meeting predefined acceptance criteria before added to the knowledge base. This is to address the aspect of safety in terms of bounded exploration again. Such acceptance metrics, however, are non-trivial to define and depend strongly on the learning task.

Example. *For the running example of an intelligent traffic control system, such an acceptance criterion could be the compliance to known maximum capacity of a road lane which leads into the observed intersection. Further concrete examples of such acceptance metrics and situation descriptions as well as for a domain-specific utility function can be found in [Pro11; Tom12].*

For the indeed relevant case that widening cannot satisfy the predefined constraints and the evolution strategy on L2-C has not finished the optimization process yet, the rule base on L1-C should be equipped with at least one default rule to serve as a viable fallback solution. This could be a human-engineered default signal plan for the traffic light control scenario. Alternatively, the system could decide to just keep the previous configuration. Except for severe abrupt changes in the detected conditions, this can be plausible since the system should currently follow an optimized configuration for a similar situation that appeared before. The interplay between the new rule widening covering operator, and the outsourced evolutionary optimization mechanism results in a situation in which only safely exploitable rules are created and applied by the learning system. These rules should not be expected to be optimal, but at least fulfill minimum criteria as defined by the users. However, this circumstance does not prevent the occurrence of knowledge gaps during the systems runtime. Whenever a system is exposed to a entirely unforeseen situation, neither rule widening, nor the rule exploration on L2 can be expected to deliver high-quality rules just in time. Even more capable mechanisms are needed to compensate for such situations. The interpolation-based techniques as will be developed in this thesis constitute a possible solution.

Remark One important aspect on which it is worth to spend a couple of further thoughts is that the steady-state GA, conventionally used for global optimization of the state space coverage, is replaced by a generational evolution strategy that does not consider the coverage at all. This means that it does not select from a mating pool of existing classifiers sharing a common environmental niche. Instead, the evolution strategy initializes a certain number of random actions from the action space A and applies mutation over a predefined number of generations. Accordingly, it seeks an action $a \in A$ for the current situation σ_t which is nearly optimal in terms of the utility (or fitness) that was returned by the applied fitness evaluation component. This utility value can then be further used for the initialization of the prediction parameter $cl.p$ of the classifier to be created. Thus, the generalization pressure of XCS-O/C is exerted only by the widening mechanism. As a consequence, Wilson's generalization hypothesis does not apply directly, since reproduction is not based on the action set anymore. Conversely, the covering mechanism now stresses the classifiers to generalize but without considering their current fitness estimates. In summary, however, the pursued aim to constrain the exploratory behavior of XCS-O/C is indeed met with this approach. The consequence of

1. feeding the population with one or more handcrafted default rules which serve as fallback,
2. only creating new classifiers with optimized actions at the offline learning layer L2 on demand, and
3. handling so far uncovered situations by widening existing (and thus optimized) rules,

leads to the situation that the rule base on layer 1 consists only of classifiers that satisfy predefined and domain-specific acceptance criteria. In this way, safety-critical trial-and-error situations are mitigated and the system can be assumed to act within adequate exploration bounds. Nonetheless, the removal of the steady-state niche GA, results in limited generalization pressure. Furthermore, the application of a generational evolution strategy instead of the conventional steady-state approach weakens XCS's ability to remain responsive to NSEs that are subject to change. While the steady-state GA steadily considers the fitness of already existing classifiers and pushes fitter rules toward a maximum level of generality, the offline optimization process has to explicitly reflect detected changes in the environment in the fitness evaluation components (e.g., in the simulation model). Of course, this can be achieved by the system operator who monitors the system's behavior. Another means would be to apply novelty detection methods [GS16] to let the recognize such changes by itself. However, this does not feedback the detected changes into the fitness evaluation components at L2. The acceptance criteria as well as the current fitness evaluation methods still need to be adjusted in order to capture the changed conditions.

3.1.3. Four Core Challenges of Learning in SAS

As becomes apparent, the OC-related task of online learning bears several challenges. Many of them have already been addressed, others remain unsolved and are under current investigation. The research field of ML in OC systems still provides a variety of open questions. These open research questions should be approached in order to meet the rising demand of SLAS which will appear the future. In the context of this thesis, identified challenges which are deemed most critical are summarized as follows:

I. *Knowledge gaps* which appear due to prevalent sparsity and imbalances in the incoming data

II. *Non-stationarity* of the learning environments in which the systems are asked to act flexibly and robustly

III. *Behavioral guarantees* by means of assuring explainability, interpretability and exploration boundaries

IV. *Problem space complexity* in terms of (the curse of) dimensionality, uncertainty, continuity, obliqueness and curvature

This thesis addresses the first challenge of dealing with KGs. A more detailed notion will be defined shortly in Section 3.3. Nevertheless, also the aspect of NSEs is partly touched upon in the last part of this thesis. It is also already captured in the formal system model which will be derived in the subsequent section. The need for behavioral guarantees constitutes a huge topic in the AI community these days. This seamlessly applies also to research concerning SLAS which are supposed to act autonomously based on AI technology. The aspect of interpretability of the evolved knowledge bases is implicitly seized by the utilization of ERBML techniques. These algorithms represent their knowledge in form of IF-THEN rules that, depending on the condition representation used, are far better human comprehensible than subsymbolic methods such as ANNs. Methods to further compact the resulting rule bases [TMU13] and visualization techniques which allow for post-hoc analysis already exist. Examples are *feature dependency trees* [But+04b], *attribute tracking* [Urb+18] and more [LXB17], each of which increasing the interpretability property of LCS. Despite not falling in the scope of this thesis, they are definitely worth to receive more research attention. The same is true for establishing and proving guarantees regarding the exploratory behavior of autonomously learning systems. Apparently, interpretability and explainability can be seen as first stepping stones in this direction. Last but not least, the complexity of the underlying problem spaces is not exclusive to SLAS. The outlined properties also challenge ML and optimization algorithms since the year one.

3.2. System Model Derivation

In this section, the scope of technical systems which are deemed to facilitate the problem of KGs which this thesis' centrally approaches is narrowed down to a concise system model. Starting with an informal description of the targeted type of systems, a formal notion related to the ML point of view is presented.

There exist various architectural approaches of how SAS can be designed [Kru+15]. Two of the probably most prominent architectures are: (1) The *Monitor-Analyze-Plan-Execute* (MAPE) cycle [KC03] originating from IBM's AC initiative. (2) The generic O/C architecture from the OC domain [MT17a], and its more sophisticated MLOC version, as introduced in the previous section. Both architectures share very similar ways to structure single components dedicated to specific tasks. However, in this thesis the MLOC architecture serves as reference case for an architectural model, since it clearly defines the incorporation of rule-based online learning mechanisms already in its generic blueprint.

In the remainder of this chapter, SLAS with a local scope, i.e., single-context systems (cf. Ch. 2.1), composed by adhering to the MLOC scheme (see Fig. 3.2) are considered. The SuOC of such a SLAS is assumed to be deployed in an NSE [Dit+15] which is subject to ongoing change. This affects the encapsulated data stream in a sense that *drifting input distributions* $P(X)$[3] affecting the derived system states $\sigma \in S$ are presumed. This means also that the systems are subject to *uncertainties* (e.g., noisy sensor data) and unforeseen internal or external *disturbances*. Furthermore, occasional *system goal changes* which demand for ongoing reconfiguration at runtime are expected. As a result, the initially unknown data distributions are expected to be non-uniform. All the aforementioned assumptions are deemed to eventually facilitate sparsity and imbalances regarding particular regions of the situation space. In technical terms, these assumptions can be abstracted to a notion of *data generating processes* (encapsulated by the NSE datastream, cf. Fig. 3.2) that follow an *unknown probability distribution function*. Whenever at least one of these processes changes over time or due to other influencing aspects, those changes are regarded as *gradual drifts* or *abrupt shifts* in the underlying concepts. The technical notion of *Concept Drift* [Web+16] provides technical models in this regard. In the literature it is usually differentiated between *real* concept drift and *virtual* concept drift (cf. e.g., [Gam+14]). In general, SLAS are presumably subject to both forms of concept drift.

As of yet, in own previous work regarding the KGs, only the former kind of *virtual concept drift* has been taken into account as a major cause. This implies that the main factors leading to the occurrence of KGs are expected to be due to changes in the underlying input data distributions, i.e., the prior or *marginal probability* $P(X)$. Real concept drift in contrast affects the a posteriori probabilities $P(Y|X)$, roughly speaking changes regarding the correct targets given particular situations. Input distributions changes (virtual drifts) are in turn assumed to arise due to unexpected disturbances (e.g., internal component failures) or more subtle changes over time (e.g., gradual changes due to ongoing sensor drift) that eventually move the currently derived system state abstractions (represented as state vectors $\sigma_t \in S$) to other niches within the system's state space S. Different niches within a system's state space are assigned different levels of *system utility.* These sort of "state

[3] X here denotes the domain of raw values as delivered by the data stream DS

preferences" are usually specified by an external authority such as a human administrator in charge. By specifying objectives instead of entire solutions, and letting the systems themselves learn how to solve the task at hand, a complexity reduction is pursued. The consequence of considering only the first kind of concept drift for the following KG definitions is that these preferences, i.e., utility levels, regarding the system states remain unchanged over the entire learning period. The incorporation of *real concept drift* would also involve changes regarding the externally specified utility function u, which essentially determines the typically unknown (i.e., black box) *fitness landscape* (or problem function surface) which in turn is approximated by the learning algorithms during runtime (see more detailed explanations below). This even worse kind of real concept drift would render so far learned state-utility mapping approximations invalid to a certain degree, eventually decreasing their predictive quality. This, in turn, opens a second reason for KGs to occur which will be introduced as type-2 KGs in the next Section 3.3.

Please note that for the sake of a general formal problem description as will be introduced below, both types of concept drift are incorporated. In the evaluation sections of this work, however, the consideration of concept drift is excluded. The focus of the experiments is restricted to the achievable benefits of using interpolation in order to enhance the learning capabilities of SLAS under the presence of KGs. The actual occurrence is provoked by utilizing uniform input distributions applied to selected theoretical problems which have been found to be challenging for a learning mechanism that alreay proved successfully utilizable in self-learning OC systems – the XCS classifier system. Therefore, the focus is mainly set on the algorithm's progress at early stages of learning, where no appropriate knowledge can be assumed. By following this methodology, the pure beneficial effects of the developed KG-countering strategies on the learning behavior of XCS can be fathomed. Possibly biased insights due to incorporation of specifically designed concept drifts that occur repeatedly are therefore avoided. Nonetheless, virtual concept drift is implicitly reflected in Chapter 7 where a real world application in the context of self-adaptive traffic control serves as evaluation scenario. Furthermore, the influence of virtual concept drift realized by using different sampling techniques has already been investigated in preliminary studies (cf. e.g., [Ber18; Mei17]) where the benefits of interpolation-assisted XCS could be confirmed. A thorough elaboration on that issue, however, falls beyond the scope of this thesis. Further including real concept drift in the experimental setups would amplify the aforementioned plasticity-stability dilemma prevalent in OML settings (cf. Sect. 2.2). The circumstance that so far learned knowledge might become entirely obsolete would have to be taken into account. It would then need to be decided whether it is more efficient to adapt the existing knowledge elements to the changed underlying problem concepts, or if it would be better to entirely learn from scratch. This decision would have strong impacts on the interpolation-based methodologies as developed throughout the subsequent chapters. An analysis of those impacts with respect to varying degrees of both types of concept drift goes far beyond the scope of this thesis but certainly constitutes an important step of future research.

With the formerly introduced assumptions, the first two identified core challenges that affect the actual modeling of a general learning task – *knowledge gaps* and *non-stationary learning environments* – are explicitly incorporated into the system model. To cope with these challenges, a learning algorithm is considered that is capable of building models in an online or reinforcement learning manner. This means that the current knowledge base is updated each time a new situation is encountered. Therefore, an initial data set that can be used for a priori analysis and hyperparameter tuning is not expected to be available. The algorithm family of LCS is selected to represent as such a learning component in this thesis. More precisely, it is focused on the XCS classifier system. Variants of XCS have already been applied in several autonomous systems deployed in various real-world domains, for instance: Self-adaptive traffic control [Pro+09; STH16], self-adaptive network protocols [THH11; THH11], hardware-software co-design of a System-on-a-Chip [ZH11; Zep+11], as well as self-configuring smart camera networks [Ste+17b]. A more detailed rationale for this choice is given in the last section before the chapter summary.

Formal Problem Description

As discussed in the previous section, the SuOC encapsulates the system to be adaptively controlled by a superordinate layer that constitutes the active control mechanism. The continuing cycle of sensory data and control signals can be interpreted as a data stream (or sequence)

$$DS \coloneqq ((s_1, a_1), \ldots, (s_t, a_t), \ldots, (s_n, a_n)) \tag{3.1}$$

of situations (or system states) σ_t from the systems state space S combined with corresponding control actions $a_t \in A$ over a period of time $t = 1 \ldots n$. Assume that the situations σ_t as well as the possible actions a_t are d- and c-dimensional vectors of the corresponding state space $S \subset \mathbb{R}^d$ and action space $A \subset \mathbb{R}^c$, respectively.

Remark By relying on such a real-valued encoding for the state and actions spaces, the formal model should be generic enough to comprise a wide variety of real world problems. On the other hand, it should be expressive enough to also capture simpler problems using integer encoding (holds since $\mathbb{N} \subset \mathbb{R}$) or even binary problem domains. In the end, possible limitations are related to the distance metrics which are applicable on the respective normed vector spaces defined over the reals. Without a natural order, it is hardly possible to calculate distances in order to yield a measure of similarity between two elements of a state space. For instance, how should a similarity between system state “code red” and system state “code blue” be calculated, if those are encoded categorically, i.e., by arbitrary integer numbers or other symbols? The notion of distance or at least another quantifiable type of similarity is necessary for a lot of ML techniques applied to real world data in general and for the interpolation-based approaches developed here in particular. Some basic learning algorithms, such as Q-learning [WD92], only require and countable number of states without any order between them to work appropriately. However, these techniques usually struggle with very large input or state spaces and cannot be applied to continuous spaces at all without substantial modifications – a restriction that shall be avoided by the applied real-valued encoding.

To assess the system's performance, a mapping onto a *utility space*[4] $U \subseteq \mathbb{R}$ is needed which provides a utility measure for state-action pairs (σ_t, a_t). This mapping is given by a utility function $u : S \times A \rightarrow U$. The actual *problem space* PS is given by the Cartesian product of the situation space $S \subset \mathbb{R}^d$ and the action space $A \subset \mathbb{R}^c$, i.e., $PS \coloneqq S \times A$. For the techniques developed in this thesis, the state and action spaces as well as the combined problem space PS are interpreted as *vector space* over the reals, which is further assumed to be Euclidean. This is a common assumption made in the field of ML, since many methods heavily rely on the field of linear algebra of which vector spaces and linear combinations constitute essential parts. [5] As already discussed above, this assumption is necessary for this work, since the interpolation techniques applied in the next chapters, will use the state vectors $\sigma_t \in S$ as sampling point coordinates $\vec{x}$, whereas actions from A or any other numerical value (the *output* or *target* values) will be used as the function values $f(\vec{x})$ of the unknown functions interpolated.

In the following, at first the actual learning task is modeled from two different ML angles – from the *Supervised Learning* (SL) and the *Reinforcement Learning* (RL) perspectives. Afterward, a more general notion is modeled in order to abstract from concrete learning tasks.

Supervised Learning From a SL point of view, the task is to figure out and model the relationship between a number of *features*, here the state vectors $\sigma_t \in S$, and a *target* variable (independent variable), e.g., an adequate action $a_t \in A$ or an utility estimate $u_t \in U$. For the training step, examples comprising features and the desired targets are presented to an SL algorithm. Usually, features are represented as d-dimensional vectors $\vec{x} \in X \subseteq \mathbb{R}^d$, whereas the target can be a categorical value $t \in T \subset \mathbb{N}$ (e.g., one out of a few possible control actions), or a scalar $t \in T \subset \mathbb{R}$ (e.g., an utility estimate). Thus, the task of an online SL algorithm L is to create a hypothesis h that takes features $\vec{x}$ as input and approximates a

[4]Please note that the term "space" is not necessarily intended to refer to the strict mathematical definition of spaces here if not explicitly stated otherwise. This "abuse of notation" seems to have somehow established in the literature.

[5]See also
`https://machinelearningmastery.com/examples-of-linear-algebra-in-machine-learning/`
(last accessed August 5, 2019)

mapping $h : \mathbb{R}^n \rightarrow T$ onto the targets based on the training examples made available so far. For classification tasks, this modeling often involves the calculation of a set of continuous output variables $t \in [0, 1]$ in a first step. Then a certain decision mechanism fells a decision for one (ordinary multi-class) or many (multi-label) of the possible categorical values on the basis of these scalar *scores*. This approach is also called *scoring* and is used within the *logistic regression* algorithm or ANNs for instance (cf. e.g., [PF13]).

Reinforcement Learning From an RL perspective, a control problem can be modeled by means of an MDP as done before. Again, the components of an MDP include a state (or situation) space S and an action (or configuration) space A among others. In contrast to the SL approach, the learner L is asked to figure out an optimal policy π^* on the basis of an unknown state transition function $\tau : S \times A \rightarrow S$ (here deterministic) in combination with a utility (usually called reward) function $u : S \times A \rightarrow U$. For a stochastic problem, the transition function can also be defined as $\tau(s_t, a_t, s_{t+1}) = P(s_{t+1}|s_t, a_t)$. This yields a probability distribution over the possible states $s_{t+1} \in S$, given the current state s_t as well as the selected action a_t. With these two ingredients, the learning agent L can learn a policy $\pi : S \rightarrow A$ to decide which action to choose in order to maximize the expected cumulative reward over time t. In fact, this policy is another function that can be formulated by a probability distribution or rather a parameterized model (see e.g., the *policy gradient* approach [SB98]). In that fashion, model-free RL approaches try to find a policy without building explicit models of the environment dynamics. On the other hand, model-learning RL approaches strive for explicitly approximate the unknown functions τ and u to improve the present or find the optimal policy [SB98].

As can be recognized, both approaches involve the necessity to find a particular mapping from states vectors $s \in S$, and sometimes combined with actions $a \in A$, to certain targets. These targets might be the correct class or action to be executed, a scalar utility estimate, or the probabilities for the succeeding states (transition function) or actions (policy). Bearing this in mind for the system model formulation, the learning task is now abstracted toward a unified notion which abstracts from particular (sub-)tasks or (sub-)models that are build by different learning approaches to approach the overarching problem at hand. This abstracted notion is then used for motivation of the method developments and evaluations conducted in the remainder of this thesis.

The learning task abstraction is therefore defined as the induction of an approximation $\tilde{f}$ of an unknown *problem function* f,

$$\tilde{f} \approx f, \quad f : PS \rightarrow U, \tag{3.2}$$

where $PS \coloneqq S \times A$ substitutes the feature space $X \subseteq \mathbb{R}^d$ of the SL notion, and the utility space U substitutes the target space T. According to this reformulation, the learning task of a SLAS is abstracted from the particular definitions of SL and RL toward a unifying *function approximation* task.

To bring the simplified learning task of Equation 3.2 in line with the formerly introduced notion of learning in NSEs, the factor *time* needs to be added to the model. As outlined above a couple of times already, SLAS are subject to change caused by e.g., changing high-level goals and unforeseen situations or disturbances. When the system goals change from one point in time t to another point $t + x$, the knowledge (hypothesis or models) built by the SLAS (e.g., state-action-utility mappings) also need to be adjusted properly in order to assure goal compliance. Or in other words, the knowledge a learning algorithm L has acquired until time t, is not guaranteed to be appropriate at time $t+x$ anymore. When the underlying goals change, the actual mapping f of the problem space PS onto the utility space U, which essentially can be interpreted as fitness landscape or utility surface, also changes. Furthermore, the problem space PS is also assumed to be subject to alterations over time. For example, *mutual influences* between interfering subsystems might be detected [Rud+15] and, thus, corresponding additional information (new state vector components) would need to be integrated into an individual agent's situation space. Regarding the notion of inherent dynamics in NSEs, also the utility space U is allowed to change. For instance, the utility space's scale and dimensionality could increase or decrease when novel goals are added or obsolete goals are removed, respectively. The following statement summarizes these thoughts by stating an inequality regarding the abstracted learning task with the factor time incorporated:

$$\tilde{f}_t \approx f_t, \quad f_t : PS_t \to U_t \neq f_{t+x} : PS_{t+x} \to U_{t+x} \tag{3.3}$$

The aforementioned occurrences of possible change can be brought in line with the notion of *real* concept drift, where all components of a learning task can be subject to change over time t. This, however, does not imply that only the factor time causes the components to change. Due to the stochastic nature of NSEs, abrupt changes can occur without any indication at any point in time.

So far, the learning problem is modeled as the task of automatically inducing a model or approximation $\tilde{f}$ of an underlying non-stationary problem function f_t on the basis of abstracted situation observations $\sigma_t \in S$ and corresponding control actions $a_t \in A$ gathered from a continuing data stream DS as defined before. In NSEs, a typical assumption is that the system state observations stem from a data generating process that follows an unknown probability distribution:[6]

$$P_t(S) = \mathbf{Pr}(S_{t+1} = \sigma_{t+1} | S_t = \sigma_t, A_t = a_t) \tag{3.4}$$

Naturally, this data generating process is directly influenced by the control actions realized on the SuOC in order to improve the system's utility.

[6]With a slight abuse of notation, the symbols for the state and action spaces are temporarily used as random variables in a stochastic process.

Thus, with changing goals corresponding to *real* concept drifts, that also directly impact the utility mapping (i.e., the problem function f), also the aforementioned system state distribution might change. This is because those niches within the problem space which yield higher utility values are shifted to other locations as a consequence to the changed structure of the fitness landscape. The task of the SLAS is then to successively steer the system to exactly those high utility niches again. The system state trajectories through the problem space in turn impact the probability distributions $P_t(S)$ over time.

Further reasons for *virtual* (or covariate) concept shift which causes abrupt changes in the system state distributions, i.e., $P_t(S) \neq P_{t+x}(S)$ are internal or external disturbances. Disturbances are unforeseen events that abruptly shift the system's state σ_t to another region within the state space S. This *abrupt shift* differs from a *gradual drift* in that no successive steps (movement trajectory) through the problem space is assumed. The runtime adaptation mechanism at L1-C of the MLOC-based SLAS is then responsible to guide the system state back to regions within the problem space PS that yield the targeted level of utility. The difference to the real concept drift case is that not the entire problem function f_t is changed, since the utility mapping remains the same. But only the system state generating process, i.e., the probability distribution $P_t(S)$ again changes.

At this point, the system model is now informally as well as formally defined. A unified notion for the task of learning in SLAS as considered in this thesis has been developed. The developed system model now serves as common basis and motivation for the actual problem statement which is derived in the next section – the definition of *Knowledge Gaps* (KGs).

3.3. Knowledge Gaps in Self-Learning Adaptive Systems

The purpose of this section is the introduction of the notion of *Knowledge Gaps* (KGs) and related terms such as knowledge base and knowledge elements in the context of knowledge-based SLAS.

Therefore, first the general term of *knowledge* shall be revisited on the basis of the well-known Wisdom pyramid [Row07] which is depicted in Figure 3.5. This hierarchical model defines *knowledge* as a result of the repeated use of *information*. That is the system has made experiences with the information, e.g., by feedback after its application. Information in turn constitutes *raw data* which is structured and annotated with a certain form of semantics. For instance, sensor data provided with a certain encoding and annotated with the actual meaning of the values. The tip of the pyramid is given by *wisdom*, which again takes knowledge to a higher level. According to Rowley [Row07], there seems to be no consensus of the concept

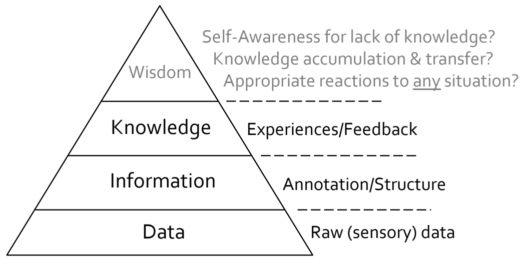

Wisdom
Knowledge
Information
Data
Self-Awareness for lack of knowledge?
Knowledge accumulation & transfer?
Appropriate reactions to any situation?
Experiences/Feedback
Annotation/Structure
Raw (sensory) data

FIGURE 3.5.: The pyramid of wisdom model, adapted from [Row07] to fit the introduced system model

of wisdom, especially when considered in the context of knowledge-based information systems. However, Rowley outlines the interpretations of a few authors that relate wisdom with: (1) The ability to accumulate knowledge and understand how to apply it from the original domain to new situations or problems. (2) The highest level of abstraction and vision foresight. (3) "(...) the ability to act critically or practically in any given situation." [Row07].

In order to bring in line the above notion with and to provide a more intuitive understanding of knowledge in technical systems, consider a simplified version of the introduced running example of *self-adaptive traffic light controllers* (cf. Chapter 7 for a more detailed introduction):

Example. *Detectors situated at a road in front of a* single traffic light *deliver simple integer values counting the number of vehicles passing this road. Augmenting this* raw data *with the semantics that this detected integer represents the counted number of vehicles that used this particular road and annotating it with a unit such as* $\frac{vec}{h}$ *yields* information. *Let this information be further related with (1) an applied control signal, here the adaptation of the green phase by applying an alternative signal plan, and, (2) with a corresponding utility feedback obtained, here e.g., the average waiting time of cars at this traffic light. "Experiencing" this extended information a certain number of times increases the confidence of the utility estimate and thus yields* applicable knowledge *in form of an IF(x* $\frac{vec}{h}$*)-THEN(signal plan B) production rule.*

To reach a level of *wisdom*, however, would require the SLAS to be able to: (1) Respond to any new situation with an appropriate control action. (2) Foresee upcoming system states far in the future. (3) Critically assess each decision, and thus its current set of production rules which represent its knowledge base. In more complex scenarios, this state of wisdom is hardly reachable, because not any situation is encountered often enough to reach the necessary level for begin regarded as safely applicable knowledge. In fact, the opposite is the case. Typical situations occur far more frequently than even more important special cases that might impose severe utility degradation when not handled appropriately. Exactly this missing knowledge in a system's knowledge base constitutes what will be defined as KG in the following.

According to this intuitive model, technical representations of the terms *knowledge* and *knowledge gap* are successively defined formally as follows. The defined subcategories of KGs are further underpinned with examples from the traffic control scenario sketched above.

Definition (Knowledge element). *Let k denote a single knowledge element which is part of a finite* knowledge space KS. [7] *The space of any describable k is given by the problem space PS extended with the natural numbers (assumed to be bounded to a maximum number) and the system's utility space U.*

$$KS \subset PS \times \mathbb{N}_0 \times U \tag{3.5}$$

Thus, k represents a four-tuple as defined by:

$$k \coloneqq (D_i, a_i, exp_i, q_i) \in KS \tag{3.6}$$

In the above definition, $D_i \subseteq S$ constitutes a finite subset of possible states σ of the state space S. The incoming data as gathered by the system's observer from the data stream DS provided by the SuOC can be understood as *raw data.* If this raw data is preprocessed or (re)structured by the observer, descriptive meta-information is added (*semantics*), and eventually an abstracted situation description σ_t is derived the level of *information* is reached. $a_i \in A$ in Definition 3.6 is the action, e.g., a possible control action to adapt the SuOC's configuration. The action enriches the derived situation description such that it becomes applicable *information*, i.e., in this particular system state, apply this particular action. However, to judge on the quality of this information, it has to be further enriched by some sort of feedback. With increasing application of this information the learning system gains experience $exp_i \in \mathbb{N}_0$ which, in combination with the aforementioned quality feedback $q_i \in U$ eventually yields *knowledge.* The successively received quality feedbacks provided by utility measures from U allow for an estimation of the expected utility of this particular information.

To sum up, knowledge elements sort of encode *rules* that allow for a derivation of policies that determine whether or not to apply them in certain situations. Consider the following example to gain a deeper intuition.

[7]For simplicity, in this thesis, KS is deemed to comprise only a finite number of knowledge elements. This assumption can be justified by considering the fact that computational representations are typically discretized and limited in some sense. Thus, this also holds for the potential knowledge which can be sensibly represented in such a computational model. For example, the state space can be discretized to represent values up to a precision of five decimals. Analogously, the action and utility spaces A and U might be restricted. Experience is countable and can thus be restricted to a maximum number of experiences necessary to be regarded as "fully-experienced".

Example. *In the self-adaptive traffic light control scenario, a single knowledge element $k_i \in KS$ could comprise an adaptation rule which comprises a specific signal plan B as action a_i which is valid for detected incoming traffic flows in the range $D_i := [20, 50]\frac{vec}{min}$. This rule has evaluated to an average utility measure q_i of $5\frac{sec}{veh}$ delay averaged over the last $exp_i = 10$ experienced applications.*

Having defined the term knowledge in a technical sense, next the notation of a *knowledge base* K can be defined by:

Definition (Knowledge base). *A knowledge base K is a finite collection with a maximum number of N knowledge elements k_i that are continually acquired by the utilized learning algorithm $L(K)$ over time.*

$$K := \{k_i\}, \quad i \leq N, N \in \mathbb{N} \tag{3.7}$$

It is to be noted explicitly that K might contain k_i that are of low quality and low experience. This means that they might be insufficiently experienced, just constructed and arbitrarily initialized, or transient knowledge elements. However, over time, insufficient k_i might gain experience and, thus, might become more useful.[8]

According to the definition of the learning task in the previous section, the knowledge base K can also be subject to change over time t. Again, this is the main reason why ML algorithms $L(K)$ capable of online learning, i.e., working by incrementally building up and continually adjust their knowledge base K, are considered in this thesis.

With the definition of knowledge elements k_i stemming from a large knowledge space KS and the notion of a knowledge base K that is subject to continual improvement over time t by means of a learning algorithm $L(K)$, the theoretical concept of a *knowledge gap* can eventually be defined.

Whenever it attempted to *learn* an entirely new task, that is a task the system has not been exposed to in the past, the existing knowledge about this task is expected to be quite low and thus K is nearly empty. Approaching a learning task with a completely empty knowledge base is also sometimes referred to as learning *tabula rasa*. Accordingly, in the following *gaps* are always defined in relation to the current knowledge base K with respect to some point in time t. For the sake of readability, an index t indicating the point in time is omitted throughout the following definitions.

In a first step, a set of *type-1 knowledge gaps* KG_1 is defined to comprise all knowledge elements k_i from the knowledge space KS that are not part of the knowledge base K yet, i.e.,

$$KG_1 := KS \setminus K.$$

[8]From this time on, it is not distinguished between inexperienced and experienced knowledge elements to be just applicable information or actual knowledge. Since inexperienced k_i can improve over time, all elements of K are assumed to be knowledge elements.

With this first definition, the size of KG_1 will be enormous, depending on the cardinality of the state space S, A, and U, as well as the lifetime horizon of the system which bounds $exp \in \mathbb{N}_0$. Indeed, each $k_i \in K$ reduces $|KG_1|$, but each $D \subset D_i$ and the empty set $\emptyset$, combined with all possible experience and utility values, again would constitute a gap $kg_j in KG_1$. Thus, type-1 knowledge gaps are redefined as follows:

Definition (**Type-1 Knowledge Gaps**). *Regions D_j within the state space S, regardless of their experiences exp_j, qualities q_j, as well as the advocated actions $a_j \in A$, which are not already covered by any knowledge element in the learning system's knowledge base K so far are type-1 KGs kg_j.*

$$KG_1 \coloneqq \{kg_j \in KS \mid \nexists k_i \in K : D_j \subseteq D_i\} \tag{3.8}$$

With this redefinition, the theoretical magnitude of the set of type-1 knowledge gaps $|KG_1|$ can be reduced by $|\mathcal{P}(D_i)| - 2$ gaps for each $k_i \in K$ (under the simplifying assumption of non-overlapping k_i). $\mathcal{P}(X)$ denotes the power set of a set X including both, X as well as the empty set $\emptyset$.

In Figure 3.6, a schematic of an exemplary two-dimensional state space is depicted that provides an intuitive understanding of KGs. Therefore, the subspaces D_i of existing k_i as well as for the exemplarily shown knowledge gaps kg_j (appearing shaded) are chosen to be represented by rectangles. Naturally, in higher-dimensional state spaces the comprehensibility of such an intuitive geometric representation vanishes, however, the formal definitions still apply.

Example. *In the context of the self-adaptive traffic light scenario, a type-1 KG might occur whenever a completely unexpected traffic condition is perceived by the system's sensory equipment. Such a rising road congestion could be caused by unexpected accidents or water-pipe bursts that require blockages of parallel roads. In such unanticipated situations, the rule base initially contains no knowledge element at all and does not know how to react appropriately.*

Each k_i is assigned a quality estimate q_i. This allows to judge on whether knowledge is either of higher or lower quality. To also include low quality knowledge elements in the definition of KGs, the former definition is complemented by the notion of type-2 KGs:

Definition (**Type-2 Knowledge Gaps**). *Already existing knowledge elements $k_i \in K$ which exhibit only poor quality q_i and an insufficient degree of experience exp_i are regarded as type-2 KGs.*

$$KG_2 \coloneqq \{k_i \in K \mid q_i \leq \theta_q \wedge exp_i \leq \theta_{exp}\}, \tag{3.9}$$

In the above Definition 3.9, θ_q and θ_{exp} constitute thresholds which determine sufficiency.

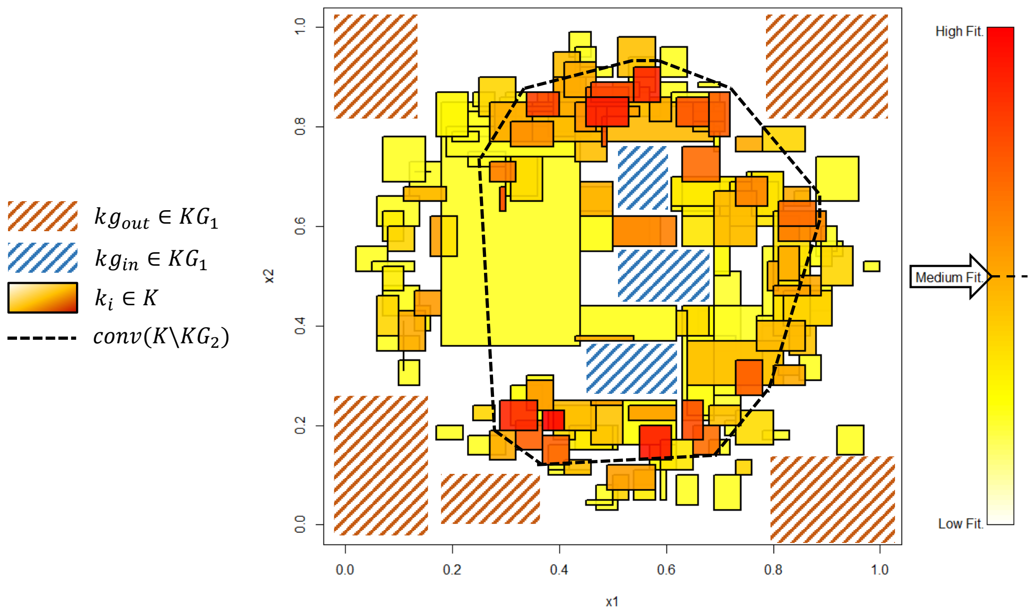


FIGURE 3.6.: A schematic illustration of the different types of knowledge gaps in an $n = 2$ dimensional input space. The color scheme indicates the quality or *fitness* (abbreviated as "Fit.") of the plotted knowledge elements, with highly to low fit elements colored in a range from red to white. Exemplary knowledge gaps of the defined types are depicted as shaded rectangles.

The thresholds θ_q and θ_{exp} involved in the above definition can be set to fixed values at design time or rather self-adaptively set during runtime, e.g., by assign them to the corresponding mean values of K at time t. Having a look at Figure 3.6 again, also knowledge gaps of type 2 can be found. A collection of knowledge elements $k_i \in K$ is shown, colored with their according quality estimates, called fitness (Fit.) in the schematic. Again, rectangles represent the scope D_i of the individual knowledge elements k_i, which are colored from white (low fitness), over orange (medium fitness) to dark red (high fitness). All elements appearing in the color range of light orange toward white can be considered of insufficient quality (the threshold is indicated by the arrow pointing to the temperature bar on the right hand side).

Example. *In the outlined traffic control system, a type-2 KG would be an adaptation rule that was reactively created to handle a rarely occurring situation that the system was exposed to only once in the past. For example, a demonstration near the city hall for which a couple of roads need to be blocked on demand has caused a circumnavigation over the observed and controlled road. There might indeed exist a rule that captures this rare situation. However, the signal plan adaptation does not yield an adequate compensation of the increased traffic demand yet and needs to be further optimized in terms of more fine-grained situation intervals D_i or better control actions a_i.*

Finally, the union of both types constitutes the entirety of KGs denoted KG, i.e.,

$$KG \coloneqq KG_1 \cup KG_2 \tag{3.10}$$

As follows from the previous definitions, KG is not necessarily disjoint from the knowledge base K, i.e., $KG \cap K \neq \emptyset$ might hold.

Additional to the previous definitions of type-1 and type-2 KGs, the following lines further introduce the concepts of *inner* and *outer* knowledge gaps, denoted KG_{in} and KG_{out}, respectively.

Inner knowledge gaps are surrounded by existing knowledge elements $k_i \in K$, thus, intuitively they more probably belong to the second type of gaps which are due to insufficient experience and quality. In Figure 3.6 inner KGs of type 2 are the yellow rectangles within the indicated black dashed convex hull. More formally, inner knowledge gaps are defined as follows:

Definition (Inner knowledge gaps). *Knowledge gaps kg_j whose finite state space subset D_j is entirely enclosed by the convex hull which is built from the subset of knowledge elements k_i in the knowledge base K which have a sufficient degree of experience and estimated quality and thus do not constitute type-2 KGs.*

$$KG_{in} \coloneqq \Big\{ kg_j \in KG \mid \forall \sigma \in D_j : \sigma \in conv(K \setminus KG_2) \Big\}, \tag{3.11}$$

with,

$$conv(K \setminus KG_2) = conv\left(\bigcup_{i=1}^{|K|} D_i \setminus \bigcup_{j=1}^{|KG_2|} D_j \right), \tag{3.12}$$

In the above definition, $conv(\cdot)$ denotes the *convex hull* of a set of vectors. According to de Berg [Ber08] it can be defined by $conv(X) \coloneqq \bigcap_{X \subseteq K \subseteq V} K$, where K denotes all convex supersets of X, which are at the same time convex subsets of a vector space V. An alternative definition is given by the set of all possible convex combinations of a finite set of vectors $x \in X$, i.e., $conv(X) \coloneqq \{\sum_{i=1}^{|X|} \lambda_i x_i \mid (\forall i : \lambda_i \geq 0) \wedge \sum_i \lambda_i = 1\}$. Thus, inner knowledge gaps are $kg_j \in KG$ surrounded by existing knowledge elements k_i from within the system's knowledge base K without all contained type-2 KGs KG_2. Although intuitively inner KGs seem to be more likely of type-2, also type-1 KGs are possible as depicted by the bluely shaded rectangles in Figure 3.6.

Inner knowledge gaps are expected to occur mainly due to the rule discovery mechanisms of an ERBML-based SLAS which introduce type-2 KGs due to arbitrary or non-informed rule initialization.

Example. *For example, when the self-adaptive traffic control system attempts to self-optimize, it could explore the problem space by discovering novel rules that focus on a more specified subset of observable situations accompanied with further adjusted* *signal plans. Such an exploratory behavior often follows the trial-and-error principle and yields inferior utility until the right parameterization, for instance slightly increased green times in a more specific range of incoming vehicles per minute, has been found in the course of optimization.*

As follows from the definition of inner KGs, *outer knowledge gaps* can be straightforwardly defined as follows:

Definition (Outer Knowledge Gaps). *The set difference between KG and KG_{in} is defined to contain all outer KGs since they essentially constitute the opposite of inner KGs $kg \in KG_{in}$.*

$$KG_{out} \coloneqq KG \setminus KG_{in}. \tag{3.13}$$

Example. *Outer knowledge gaps can occur due to extreme cases in each dimension of the state space. For instance, consider a world championship soccer game which takes place only once in a number of decades. This situation might not be foreseen by the system engineers at design time. A couple of hours before the games starts, the traffic situation in the city will dramatically increase in terms of road congestion. This leads to extreme values measured through the sensory equipment of the self-adaptive traffic light controller. This moves the system state vector away from the usually observed state space niches toward outer regions where no knowledge elements might be present within the system's knowledge base so far.*

This particular distinction between inner and outer knowledge gaps is introduced in order to decide the means of how to close identified gaps later on. For example, if an $kg_j \in KG_{in}$ is identified, maybe an interpolation (clearly distinguished from an extrapolation here) can be favored over asking the human expert for assistance by actively posing a query. For the case of an outer knowledge gap on the other hand, the lack of surrounding high-quality knowledge might result in poor extrapolation results. Thus, triggering an optimization process at L2-C could be more sensible in this situation. Furthermore, presenting such outer knowledge gaps as a query might obscure human experts and, thus, should be prevented to not making them reluctant and distrustful of the system. Hence, this convex hull modeling approach enables the learning algorithm $L(K)$ to self-decide on which strategy to follow, or even whether to pose a query to a human expert in order to let her close the encountered knowledge gap at all. Figure 3.6 again can be used for illustration purposes. The dotted line encompassing certain knowledge elements illustrates a rough approximation of the convex hull $conv(K \setminus KG_2)$ around all knowledge elements k_i for which $k_i \in K \wedge k_i \notin KG_2$ holds true. Accordingly, the exemplary knowledge gaps outside this convex hull are outer knowledge gaps (shaded in dark red), whereas the bluely colored gaps surrounded by several knowledge elements of sufficient fitness can be seen as inner knowledge gaps.

At this point, again, the necessity of the input space defined as a normed vector space becomes apparent. Without a given order and thus the possibility for a notion of similarity defined on distance, the geometric intuition of KGs as introduced here is not straightforwardly applicable. The possibility to generalize over more than one particular vector from the state space is a necessary criterion to allow for the application of the investigated LCS-based algorithms on very large or continuous input spaces. Since the notion of *gaps* in the knowledge bases of such incrementally knowledge building algorithms builds upon their *problem space partitioning* working principle, the applicability of this concept on problem spaces modeled by different representations needs more research. A first attempt might be to embed an arbitrarily represented state space into another space where similarity metrics can be defined, for example by using *self-organizing maps*.

At this point, also the concepts behind KGs in SLAS has been established by means of an intuitive informal description as well as a formal definition. With that, the central problem statement of this thesis is provided.

3.4. Potentials and Drawbacks of LCS-based Learning

This section is intended to shed light on the potentials and various advantages but also on the drawbacks of using LCS-based learning mechanisms in order to endow SAS with the self-learning capability.

3.4.1. Ideal Candidate for KG-Centric Learning

One aspect might already have become apparent in the previous section where a technical notion for KGs has been developed. The knowledge representation of LCS in form of classifiers cl_i constitutes a perfect match for a KG-centric learning intuition as envisioned in this thesis.

Clearly, the condition $cl_i.C$ suits the understanding of the state subspace D_i of a knowledge element k_i. There already exist various approaches to explicitly represent such a continuous state or input subspace ranging from rather naive hyperrectangles [Wil00; SB03], over hyperspheres and their extension to general hyperellipsoids [But05b; BLW08], to convex hulls [LW06], and more. Each classifier advocates one particular action (vector) from A in its action parameter $cl.a$, which directly maps to a_i from the knowledge element definition. It further maintains an experience count, so that $cl_i.exp \equiv exp_i$. The last component of an abstract knowledge element is defined to be a quality estimate q_i. A single classifier in the XCS has several parameters, namely $cl_i.p, cl_i.\epsilon$ and $cl_i.F$ as well as some sort of confidence value $cl_i.num$, what can be interpreted as the number of supporters for that classifier. These parameters can be used to estimate the quality of a classifier. Either they can be combined in a meaningful functional relationship to calculate a *quality estimate* $q(cl.p, cl.\epsilon, cl.F, cl.num)$. Or simply the predicted payoff $cl.p$ can be used which is, however, only expressive in combination with the fitness attribute. Accordingly, brought in line with LCS terminology, the knowledge elements are contained in a population of IF-THEN rules, i.e., classifiers $cl \in [P]$.

Thus, the following similes apply:

$$K \equiv [P] \quad \text{and} \quad k_i \equiv cl_i.$$

Accordingly, type-1 KGs can be formalized by

$$kg \in KG_1 \coloneqq \{cl\}_{cl \notin [P]}$$

and KGs of the second type are denoted by

$$kg \in KG_2 \coloneqq \{cl \mid cl.exp < \theta_{exp} \wedge q(p, \epsilon, F, num) < \theta_q\}_{cl \in [P]}.$$

Apart from that mappings, XCS's knowledge base, i.e., the population $[P]$, is limited in size and built up (and improved) incrementally. All these fulfilled requirements facilitate the development of implicit KG closing strategies (see Ch. 4 to 9) as well as the conception of explicit KG identification mechanisms (see Ch. 10).

3.4.2. Human-Interpretable Knowledge Construction and Extraction

As already outlined before, the use of rule-based learning algorithms bears distinctive advantages. Rules of the generic form IF(condition)-THEN(action) are in principal directly interpretable and comprehensible by human operators. XCS evolves such rules. Even if the input dimensionality is high, the interval-based (hyperrectangular) condition representation is easy to understand and to represent (e.g., in tabular form). On the other hand, however, if the condition representation schemes become more powerful (e.g., general hyperellipsoids), then the comprehensibility naturally suffers. Nevertheless, the representation in form of rules has another advantage: It allows the injection of initial or default rules, optimized by domain experts and framed in the intuitive IF-THEN shape. In the scenario of self-adaptive traffic light control, this is indeed done by using a human-engineered standard signal plan that serves as backup solution whenever the system is unable to immediately respond with a self-learned rule.

As already mentioned in Section 3.1, various ways to further increase the interpretability of the evolved knowledge bases have been proposed in the literature. Condensation [Wil95] and compaction techniques [BLW08; TMU13] serve the purpose of distilling the most expressive and accurate rules from a non-filtered final population which usually also comprises transient rules. Feature dependency trees [But+04b], so far only presented for binary-coded input domains, provide a means to create a tree-based structure which allows insights regarding the dependency and the importance of single features from the state space. A related endeavor is pursued with the attribute tracking (and feedback) mechanism proposed by Urbanowicz et al. in [UBM14; Urb+18]. It can can be used to reveal epistatic relations and heterogeneity [UM10] in the input data. Liu et al. recently proposed another way to extract knowledge from the combination of evolved solutions to specific problems from multiply set-up XCS instances in [LXB17].

3.4.3. Evolutionary Online Learning

Besides the interpretability of the evolved knowledge, XCS strongly relies on the powerful method of computational evolution. The evolutionary part of the discovery component, the incorporated steady-state GA, continually strives for classifiers within the numerous environmental niches, that maximize the generality and accuracy at the same time. Thus, it is inherently geared up to deal with the second identified core challenge of learning in SLAS – the non-stationarity of real world learning environments. Due to the evolutionary pressures that guide the learning process in XCS, formerly accurate rules may become inaccurate, when e.g., the fitness landscape changes due to changes in the system's goals. Consequently, over time these outdated classifiers will be removed from the population and eventually replaced by novel classifiers created by covering and further evolved by the GA. The speed of such a recovering phase after a change, strongly depends on a variety of XCS's hyperparameters that will be subject of discussion below. When this implicit adaption, driven by the evolutionary pressure exerted by the GA, shall be transferred to the modified XCS-O/C variant, the question of applicability arises. Since the conventional GA is replaced by a generational evolution strategy situated at the offline learning layer L2, the generalization theory as outlined in Section 2.4 does not apply anymore. Can the XCS-O/C still be deemed responsive to change? With the only reactive activation criterion of the offline optimization process described so far, no. In contrast to standard XCS, the discovery of new rules in XCS-O/C is not invoked periodically, but only on-demand. For instance, when the population does not contain a matching rule at all. This coincides with the introduced notion of type-1 knowledge gaps in a sense that it constitutes a possible solution to handle them. In standard XCS, the involved GA is periodically invoked regardless of how well a particular niche is already covered by the match set $[M]$. Thus, considering that a similar situation occurs various times repeatedly, the GA will continue to create new classifiers what overpopulates this niche and eventually causes subsumption or the deletion mechanisms to remove knowledge. At the same time, the niche-relative fitness estimates of the matching classifiers will drop due to fitness sharing – possibly leading to type-2 KGs. This again increases the deletion probability further. Final removal (not only decreasing the numerosity) from the population might result in new type-1 gaps, which will be implicitly handled by the covering mechanism with a predefined number of new classifier advocating different actions each. Hopefully (depending on θ_{mna}) these comprise the now appropriate one which yields high utility (reward) again and satisfies the changed goals.

Applying the new theory of KGs to XCS or else XCS-O/C allows for new mechanisms to detect such changes more quickly and to initiate countermeasures. This could be done by (proactively) triggering e.g., the optimization component at layer 2. First concepts for this conjecture are presented in Chapter 10.

3.4.4. LCS are More than just Yet Another ML Technique

In Section 2.4, it was outlined that LCS have been proposed incorporating various extensions realized by means of other ML techniques such as ANNs, Support Vector Regression, RBF interpolation, and more. This fact gives rise toward an alternative perspective on LCS. Considering the problem space partitioning learning intuition in conjunction with the various possibilities of utilizing existing ML techniques within the locally defined models is to a certain degree reminiscent of the notion of *ensemble learning*. Each rule can then be interpreted as individual (weak) learner, each possibly utilizing a different ML technique for predictive modeling. Since the classifiers are partly overlapping, this necessitates some mechanisms for conflict-resolving. In XCS, this is accomplished by the applied classifier mixing strategy, most commonly realized by the PA-based action-selection regime. Of course, other ways are imaginable [DB07]. The evolutionary nature of LCS even more extends the hybrid ensemble view to an *evolutionary ensemble learning* technique. Internalizing this alternative view on the learning paradigm of LCS opens a variety of possibilities to leverage it. For instance, the aforementioned possibility to automatically select the most appropriate ML technique to model a classifier's reward prediction in dependence on the underlying part of the problem surface, for which a classifier is responsible.

To sum up, LCS should be understood as a capable *evolutionary machine learning framework* instead of being regarded just yet another old-fashioned ML algorithm.

3.4.5. Hyperparameter Configuration

Despite the various aforementioned advantages and potentials of using LCS-based learning mechanisms, hyperparameter configuration constitutes a clear downside. With around 20 configurable hyperparameters (see Table B.1), finding the optimal configuration for XCS is an intensive non-trivial task. To be fair, though, not every parameter has to be adjusted every time XCS is deployed to a new learning problem. The most crucial hyperparameters are to be adapted are:

- The maximum population size N

- The values for determining the initial classifier generality $P_{\#}$ and r_0
- The mutation step size for real-valued condition structures m_0
- The error tolerance ϵ_0 determining the fitness pressure.

As a second shot, the learning rate β, the mutation and cross-over probabilities μ and χ, as well as the thresholds for GA invocation θ_{GA}, subsumption θ_{sub} and deletion θ_{del} should be revisited whenever a lack of convergence is encountered. Urbanowicz and Browne provide a helpful guide how to parameterize accuracy-based LCS in [UB17].

Parameterization of an LCS algorithm demands for a certain degree of experience as well as a deep understanding of the inner working principles. An exhaustive hyperparameter study before deploying an LCS to the actual problem at hand, requires a vast amount of time and computational resources as well as the existence of a representative training set reflecting the real environmental dynamics. In the era of DL and DNN such a high number of hyperparameters is not exceptional anymore. Of course, recent advances from the field of automated algorithm configuration [HNT+17] and AutoML [Tho+13] can be directly applied to tackle this issue. Current research aims at deriving optimal parameter settings theoretically [Nak+17; NBH18], however, yet the theory is limited to the use in binary problem domains. For a learning task as defined in Section 3.2, however, techniques that allow for hyperparameter self-configuration, such as the self-adaptive learning rate approach of Dam et al. [DLA07], might be preferable. Even if not the focus of this work, with the interpolation-based classifier generation technique developed in Chapter 6, a few hyperparameters for initializing new classifiers will become nearly obsolete, which constitutes a first step in that direction.

3.4.6. Learning in High-Dimensional Input Spaces

It is common sense that traditional forms of LCS struggle with high-dimensional input spaces. A recent study captures the implications of the curse of dimensionality on LCS in great detail [DS17]. As follows from the theoretical insights of learning bounds and generalization in XCS and its supervised derivative, the UCS, convergence of these systems can only be assured when certain challenges are met. Among them, the covering challenge as well as the challenge that classifiers receive reproduction opportunities are most essential. A direct implication is that the maximum size of the classifier population directly depends on the dimensionality of the input space in an exponential relationship. Larger population sizes, in turn have a direct impact on the computational demands, since the computational effort of the matching procedure only scales linearly with the number of classifiers. The expected time to convergence also increases tremendously when the dimensionality gets higher. Several approaches exist that overcome this problem with either injecting domain-knowledge [UM15; UGM12a], or by using dimensionality reduction techniques such as *Principal Component Analysis* [Beh+12] or *Deep Auto Encoders* [Mat+18]. Recent advances regarding computational capacities, for instance using GPGPUs, allow for an increased learning efficiency when measured in real time and, thus, yield faster training times for higher-dimensional data.

However, often it cannot be assumed that such powerful machines are available in SLAS deployed in real world scenarios, such as an embedded traffic light controller. On the other hand, those scenarios often do not exhibit state spaces of several hundreds of different state vector components. For example, in the self-adaptive traffic light controller scenario, typical values of situation space dimensionality for realistic urban intersections situated e.g., at metropolitan cities such as Hamburg, Germany, appeared to be less than 20.

For the system model assumed in this thesis, severely high-dimensional state spaces are not considered to constitute an issue.

3.5. Chapter Summary

The purpose of this chapter was to propose a uniform notion of aspects regarding learning in OC systems. A special-purpose modification of XCS has been introduced along with four identified core challenges of learning in SAS. From that viewpoint, a system model was derived which comprises also a formal problem description of an abstracted learning task as considered in this thesis. The identified problem of KGs in SLAS was explicitly stated and substantiated with a formal definition. At the end of this chapter, it was elaborated on the unique potentials but also on the drawbacks of endowing SAS with self-learning capabilities by means of LCS technology. With the established theoretical backgrounds as well as the cleary formulated problem statement at hand, the introductory part of this thesis is complete. In the following chapters, methodologies to attempt the identified research problem will be developed. Therefore, the succeeding chapter starts with presenting architectural approaches to augment the algorithmic structure of XCS to allow for the integration of interpolation.

Chapter 4.

Architecture for Interpolating Learning Classifier Systems

In order to alleviate the negative impacts of KGs of both types in XCS-based SLAS, in the main part of this thesis starting with this chapter, the stated problem will be approached by utilizing techniques from the domain of scattered data interpolation.

As initially stated in the research hypothesis, it is assumed that the application of scattered data interpolation techniques on memorized experiences or existing knowledge elements k in the learning algorithm's knowledge base K leads to an improved learning progress in terms of decreasing initial system errors and, thus, faster convergence to the reachable error level.

This *transductive* means of constructing and initializing novel knowledge elements constitutes the focus of the following chapters. Therefore, in this chapter a novel designated component to augment the algorithmic structure of LCS in general and XCS in particular will be designed to allow for the incorporation of various interpolation techniques during the online learning process. The subsequent Chapters 5 to 9 are concerned with the development of various *interpolation integration strategies.* These strategies enhance algorithmic steps where interpolation can be plausibly applied in order to support (1) the knowledge construction or else KG closing process, and, (2) the actual prediction step of XCS. It should be noted that even the secondly mentioned integration strategies which shall support the decision and prediction steps of XCS are related to the existence of KGs. Whenever an incrementally knowledge acquiring learning algorithm $L(K)$ has insufficient knowledge for certain situations, also the prediction step will implicitly be affected negatively. Interpolation can also help to increase the reliability of autonomously made decisions for this type of learning algorithms by guiding the internal calculations through the incorporation of interpolation. Throughout the subsequent chapters, four integration strategies will be developed and empirically validated in order to fathom their unique impacts on the learning performance of XCS.

In the following sections of this chapter, a novel dedicated architectural extension called the *Interpolation Component* (IC) is introduced. The IC is intended to complement the generic algorithmic structure of LCS as discussed in Chapter 2. Two separate modes of integration are described. The first allows for the incorporation of the IC in nearly any type of OML algorithm. And a second for the specific use with XCS. The advantages and shortcomings of both modes are then discussed.

4.1. Generic Design of the Interpolation Component

In order to develop an independent software component that provides the functionality to perform interpolations on the basis of sampling points which successively arrive in an instance-by-instance manner, the following functional requirements have to be met:

- To be independent, a clearly defined interface for data exchange is required.
- To cope with the online learning nature of the *IC-attached Machine Learning Algorithms* (MLAs), a dynamic storage of sampling points needs to be provided.
- To ensure space and time requirements, the capacity of the sampling point storage have to be limitable.
- The according storage limitation demands for a decision logic in order to decide which sampling points to keep or remove in the case of capacity exceedance.
- The component have to keep track on their reliability by continually measuring the interpolation errors, which also needs to be accessible by the MLA.
- To perform interpolations, the components need to comprise at least one applicable interpolation algorithm to be applied.

Figure 4.1 illustrates the proposed generic architecture of the IC.

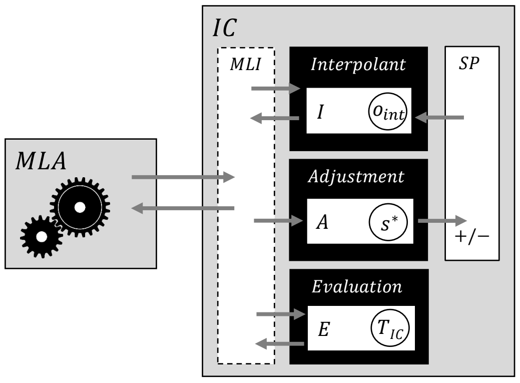


FIGURE 4.1.: Schematic illustration of the generic Interpolation Component (IC) with its five core components and indicated data flows

In order to meet the aforementioned functional requirements, it comprises five subcomponents as described below:

1. The *storage* or *set of sampling points* SP. It is more formally defined by $SP := \{s_1, \ldots, s_m\}$ and its capacity is limited by $m \leq N_{SP}$. SP serves as the internal *memory* of already made *experiences* of the MLA with the learning task of at hand.
2. The *interpolant* part encapsulates at least one scattered data *interpolation technique* I and calculates the *output values* o_{int}. The interpolated outputs are subsequently fed back to the MLA algorithm via the *Machine Learning System Interface* (MLI).
3. The *adjustment* component provides a decision logic A to manipulate the SP. Decisions that are felt here comprise: (1) Whether or not to add a new sampling point s^*. (2) When and which obsolete sampling point $s \in SP$ have to be deleted in favor of a new s^*. The adjustment component further takes care of preventing duplicates within SP or any other prerequisites to guarantee a valid interpolation solution.
4. The *evaluation* component provides an *evaluation function* E which assesses the level of interpolation quality. It is used in order to continually calculate a scalar metric called the IC's *trust-level* T_{IC}. This trust metric reflects the current reliability of the interpolation component IC. It needs to be accessible by the MLA via the MLI. This allows the MLA to decide whether or not the IC should be utilized or else to which degree its output shall affect the internal calculations (i.e., for weighting purposes).
5. Finally, the *Machine Learning System Interface* (MLI) is responsible for communication and data transfer between the MLA and the IC. The minimal required data flow between the IC and the extended algorithm is:
   a) Reporting of the feature vector $\vec{x}$ or current situation σ_t used as query point for the interpolant subcomponent.
   b) Reporting of the selected action a_{exec} or any other target variable as predicted by the MLA.
   c) Provision of the interpolated value o_{int} back to the MLA.
   d) Provision of the current trust level T_{IC} from the evaluation subcomponent to the MLA.
   e) Pass-through of any kind of numerical feedback from the MLA to the evaluation subcomponent to allow for an ongoing update of the trust level by E.

f) Pass-through of any kind of numerical feedback from the MLA to the adjustment component to create new sampling points s^*. The feedback signal allows for a decision regarding addition to SP or better discarding it. Here, the numerical feedback could a reward, the observed utility gain/drop, or simply the actually correct target value if available. The latter one might be made available with a delay by the offline optimization component of an MLOC-based SLAS).

The generic IC is principally designed to extend any OML algorithm. The implementation details of the sketched subcomponents depend on the particular algorithm to be extended as well as on the targeted learning task. These tasks can be entire learning problems, or rather subproblems to be solved by the particular MLA, e.g., building a model of the environment dynamics for a model-based RL setting. If, for example, the augmented algorithm aims at solving a binary classification task, the IC can be used to interpolate the membership score of the positive class. The evaluation component could track whether the interpolated membership was correct or not. If it was, a new sampling point with the most recent data instance $\vec{x}$ and a membership probability (i.e., a score) of 1.0 as its "true" function value can be added to SP. In the contrary case, the same can be done by setting the according score to 0. Even if this seems to be a realization of a simple binary classification algorithm by its own, it should be clearly noted that this is not the intended purpose of the IC. Rather, it should be understood as a *surrogate model* which can be quickly obtained and accessed due to straightforward interpolation between already gained experiences in form of purposefully collected sampling points. This is intended to allow for an improved initial learning progress by sort of bootstrapping the succeeding phase of model or rule convergence. Moreover, the capacity of the experience memory (or sampling point set) SP is limited which also bounds the achievable level of accuracy of the IC alone. Following Vapnik's thoughts on *learning by transduction* [GVV98], the IC omits the step of inducing a model in order to be applied to novel data instances via deduction. Rather, it directly interpolates between incrementally collected and steadily updated sampling points in order to support an arbitrary inductive algorithm during its knowledge acquisition process (i.e., while incrementally building up its model) and in its actual prediction step (i.e., deduction from the current model to a novel case).

The general idea of the IC is thus to provide an *independent complementary component* that can be utilized for *obtaining transductive information* to be further processed by the MLA to *support its individual learning (sub)tasks.*

As of yet, the IC was introduced as a more or less standalone software component with its purpose to augment existing OML algorithms and to support them by means of serving as a quickly available surrogate for a certain learning (sub-)task. At the end of this chapter, possible extensions and further ways to use the generic blueprint of the IC will be discussed.

In the following sections, a more detailed picture regarding the actual integration of the introduced IC with XCS as a concrete instance of an attached OML algorithm (the MLA) is drawn. Two architectural extensions of XCS are presented along with their corresponding advantages and disadvantages.

4.2. XCS-IC: A Loosely Coupled Extension

This first variant, denoted XCS-IC, follows the initial ideas of the generic IC. It is schematically illustrated in Figure 4.2 and depicts the XCS on the left-hand side, and the IC on the right. Both parts are connected via the MLI.

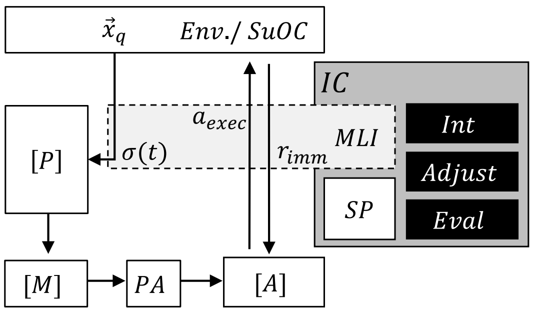


FIGURE 4.2.: Schematic illustration of the XCS-IC variant for a loosely coupled integration of the IC and XCS

Whenever a new situation σ_t is detected by XCS, this data vector is reported as a new query point $\vec{x}_q$ to the IC via the MLI. The IC in turn interpolates a corresponding output value o_{int} which is fed back to XCS immediately. XCS then follows its (possibly interpolation-assisted, cf. Ch. 5) action-selection regime and realizes the selected action a_{exec} on the environment. As soon as the reward is available, it is also passed-through to the IC which further forwards it to the evaluation and the adjustment subcomponents. One advantage of this loosely coupled approach becomes apparent here: The evaluation component can continuously determine the interpolant's quality in terms of T_{IC}. This possibility is due to the small variety of interpolatable values. These values, such as an action a_{exec} or the system prediction $PA(a_j)$, are system-wide numerical values which result from the internal calculations of XCS based on its current knowledge base. This allows for a straightforward evaluation by simply considering the received reward r_{imm} and comparing the interpolated values o_{int} with those values XCS has actually calculated. By means of the evaluation metric E, the IC is enabled to decide whether or not the interpolated values would have been more appropriate than the values actually determined by XCS. In a subsequent step, this also allows the adjustment component to adequately create new s^* for updating SP. This automated decision provides sort of a preselection of suitable sampling point candidates s^*, which in turn assures a certain level of quality regarding all $s_i \in SP$. Essentially, o_{int} can be any value which is system-wide and does not belong to a single rule. Accordingly, the action to be executed a_{exec}, or the corresponding system prediction $PA(a)$ of the reward can be subject of interpolation with regard to XCS. Also a tuple of the current input σ_t and the selected action a_{exec}, i.e., (σ_t, a_{exec}), can serve as the sampling point coordinate. The sampling point's function value could then comprise the received reward r_{imm}, which is possibly extended by the discounted maximum value from the current PA in order to yield a state-action, or Q-value estimate. The aforementioned approach of interpolating the system prediction is clearly plausible in the context of function approximation tasks, i.e., *value function approximation.* The other approach outlined, however, is sensible in single- and multi-step RL settings, ideally with continuous state-action spaces.

In Chapter 5 a particular integration strategy for using the IC in the way that particular actions are interpolated is presented. Also for the outlined integration approach that proposes to interpolate the system prediction a methodology will be developed in Chapter 9.

4.3. XCS-CIC: A Tightly Integrated Extension

The second extension variant to be introduced in the following paragraphs interweaves the subcomponents of XCS and the IC more strongly.

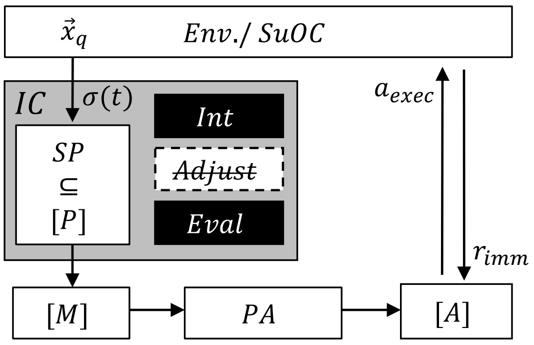


FIGURE 4.3.: Schematic illustration of the tightly integrated variant for integrating XCS and the IC

As Figure 4.3 illustrates, from an architectural point of view, this results in a stronger coupling of the two components. However, this opens a wider variety of possibilities for incorporating interpolation into the algorithmic structure of XCS.

The most obvious change in comparison to the former XCS-IC approach is that now the population of classifiers $[P]$ serves as the set of available sampling points SP. This innovation implies that the base algorithm, here XCS, is responsible for managing SP by its own. This in turn makes the adjustment component of the generic IC design obsolete as depicted in Figure 4.3.

At each time step t, the population yields an up-to-date set of possible sampling points $s_i \in SP$ which can be extracted from the currently existing classifiers $cl_i \in [P]$. Accordingly, with a slight abuse of mathematical notation, the sampling points can be redefined as $s_i := cl_i$. This further implies an increased variety of possible function values to be interpolated.

Let the i-th sampling point's coordinate $\vec{x}_i \in S \subseteq \mathbb{R}^n$ be represented by an arbitrary point extracted from a classifier's condition $cl_i.C$, i.e., $\vec{x}_i \in cl_i.C \subseteq S$. Consider this coordinate to be the condition's center point, calculated from the interval predicates (l_j, u_j), i.e.,

$$x_i^{(j)} = l_j + (u_j - l_j)/2, \qquad j = 1 \dots n.$$

Of course, more than one sampling point can be extracted from an individual classifier. However, depending on the interpolated function values this might cause unwanted effects regarding the interpolant's function surface comparable to noise. For instance, plateaus or a high degree of ruggedness might occur when classifiers with different experiences overlap.

With this second way of XCS extension, it is possible to introduce further integration strategies into the internal calculations of XCS. Interpolation can now be incorporated within the covering routine (cf. Ch. 6). The initialization of offspring classifiers cl_{off} when the GA is invoked constitutes another possible place for integration (see Ch. 8). With the XCS-CIC extension, nearly at any algorithmic step where classifiers cl_i are involved sampling points can be constructed in an ad-hoc fashion and used for interpolating different types of function values. For example, the initial values for classifier parameters $cl.p, cl.\epsilon, cl.F, cl.exp, \dots$ can be interpolated from surrounding classifiers. Since for the sake of exploring the problem space, the population contains transient knowledge during the learning process. This refers to classifiers that have been constructed but will evaluate to poor accuracy/fitness (type-2 KGs). This needs to be considered when sampling points shall be extracted from classifiers. Therefore, the task of the already deemed obsolete adjustment component can be reduced to provide a straightforward filter mechanism which is responsible for considering only classifiers (knowledge elements) of sufficient quality. The definition of an appropriate quality measure is not a trivial task. It needs to be carefully selected with regard to the integration strategy to be applied. This aspect will be picked up again at the corresponding places in this work.

4.4. Discussion

In the previous paragraphs, a generic component to extend existing OML algorithms with interpolation capabilities have been introduced. The aim is to provide a sort of blueprint instead of a concrete implementation by stating the minimum number of subcomponents deemed necessary as well as the interplay between them. An intuitive interpretation has been given. Namely, that the IC can be understood as an independent complementary software component that provides the functionality to quickly provide a so-called surrogate model which is intended to support the attached algorithm via a transductive inference mechanism. Again, it is clearly noted that the predictive accuracy of the IC itself is limited by the number of considered sampling points in SP. However, the solution of the MLA's overarching learning task is not the aim of the complementary IC. Rather, it is to support the extended algorithm at the initial learning, or knowledge acquisition phase, its actual prediction steps, or even in idiosyncratic subtasks of the MLA. A further use in later phases still seems plausible, considering the occurrence of concept drifts in the course of the systems runtime.

Two architectural variants to extend the XCS with this novel component have been presented. The first one follows the loose coupling principle. It adheres more strictly to the generic blueprint. The second variant is a more conflating approach, tightly integrating XCS and the IC with each other. This conflation, however, allows for further, more sophisticated integration strategies where interpolation can support various algorithmic steps of XCS.

An interesting way to further enhance the IC would be to allow multiple interpolants I at the same time. Each individual interpolation technique I_j with $j \in \mathbb{N}$, could then interpolate the same target variable. Subsequently, the different interpolations could be combined into one aggregate value. On the other hand, each single I_j might be assigned a separate trust level. Based on that, a meta selection algorithm could then decide which interpolation output to return to the MLA over the MLI. This is reminiscent of a hybrid ensemble approach, which are well-known to outperform individual, so-called *strong predictors* in certain cases.

When applied with the proposed loosely coupled XCS-IC extension approach, for instance because the MLA offers no algorithmic steps where a tighter integration is possible, another way to extend the generic IC concept would be to allow for multiple sampling point storages SP_j. In this case, several ML-related subproblems, necessary to solve the overarching learning problem, could be supported with one instance of the IC at the same time. This might be more advantageous in comparison to an instantiation of multiple IC's, especially in systems that are not capable of multi-threading.

As already suggested in the XCS-CIC extension variant, the notion of sampling points might slightly differ from its strictly mathematical meaning. They can be replaced by objects that only need to allow for an extraction of sampling point coordinates and at least one target function value. In the case of XCS this could be the replacement of s_i by the classifiers themselves cl_i which opens the possibility to extract several sampling points from them. For example, in the case of XCS, a single classifier contains several candidates to serve as function values to be interpolated. The sampling points' coordinates do not need to change and can be kept even if the target function values are exchanged. For instance, the fitness $cl_i.F$ could be replaced with the prediction $cl_i.p$. Thinking in geometric terms, what would change is the interpolants function surface, since the domain remains the same, but the co-domain changes. Since the classifiers comprising the population $[P]$ are subject to continual evolution, naturally the sampling points change steadily. This, however, does not constitute an issue for the IC due to its transductive working principle. Since no model is inductively built from the current sampling points in SP, no model has to be adapted and kept up-to-date by the IC. Each time the IC is activated it simply interpolates on the basis of the current composition of sampling points in SP.

A last point of discussion is the selection of appropriate evaluation functions E. As for the loss functions to train a classifier, also at this place a plausible choice should be made. However, it is not necessarily obvious what is the best choice. And sometimes a feedback signal which lets even the MLA barely judge on its quality is only sparsely available. For supervised learning tasks, similar metrics as for the evaluation of the predictive accuracy of the MLA can be used to determine the reliability of the IC, too. In the case of reinforcement learning tasks, the reward signal might serve as the figure of merit to derive the proposed trust-level T_{IC}. In the simplest case, the definition of an appropriate evaluation metric can be omitted and it is only relied on the precision guarantees of the interpolants themselves (remember the polynomial precision of RBF interpolants). Accordingly, the maintenance of SP is a highly problem-dependent and thus non-trivial task, since both, the evaluation metric E as well as the adjustment logic A need to be purposefully defined for different problems. Again, this last aspect corroborates the potential of the integrated XCS-CIC variant.

4.5. Related Work

The introduction of the IC and its integration with XCS is reminiscent of other concepts from within and beyond the domain of LCS research. The following paragraphs indicate the, to the best of the author's knowledge, only existing work that proposes an interpolation concept to be used with XCS. Additionally, works farther related in terms of integrating interpolation with other RL algorithms are summarized. At the end of this section, recent but only barely related works from the domain of ANNs are mentioned.

Interpolation in an LCS

To the best of the author's knowledge, the only work which mentions to utilize *interpolation* in the scope of LCS is due to Wilson, who investigates the question how

XCS in general can be extended to cope with continuous action spaces. In [Wil07], he presented three architectures to approach this problem. One of these is called *Interpolating Action Learner* (IAL) which clearly suggests a relation. However, what Wilson actually proposes is to apply XCSF in a hierarchical fashion. With this technique, the system on the second layer actually *approximates* between the best found discrete action values from the first layer, and therefore learns an approximation of the mapping form an input to the optimal action value. These approximations can then be applied in the environments that are observed by the immediately interacting XCSF instance. Wilson's approach differs clearly from the techniques which will be proposed in this thesis, since the idea is to incorporate explicit *interpolation* techniques as an integral part of XCS's algorithmic structure in order to achieve different goals beyond continuous actions.

Interpolation in other RL algorithms

Wire fitting as introduced in [BK93] is another technique for allowing RL in high-dimensional and continuous action spaces. It achieves fast calculations of actions that yield maximum reward given particular states. In wire fitting, so-called *wires* are learned by means of function approximation which, depending on a chosen smoothing coefficient, can also be regarded as being interpolated. However, the wires can be interpreted as high-dimensional sampling points or so-called *control points* which are subject to training (i.e., the wires are *fitted*) and, thus, do not reflect actual experiences with the environment. The interpolation property as introduced in Section 2.3 is further used to proof the property that the maximum control point is always the maximum of the approximated function, which allows it to be quickly evaluated and maximized. Wire fitting can indeed be understood as bringing interpolation into RL systems, however again with a completely different objective, namely allowing for learning with continuous actions.

Davies introduced interpolation to RL tasks with continuous actions spaces in [Dav97]. He extended model-based value iteration, Q-learning and model-learning value iteration by incorporating grid-based multilinear and simplex-based interpolation. These extensions are evaluated on two widely investigated continuous state space problems – Mountain Car and Acrobot. Results revealed that the necessary grid-size which was typically used at that time next to neural networks can be significantly reduced by interpolation. This work shares common ideas with this thesis and can be seen as primary attempt to exploit the benefits of using interpolation in RL systems. However, the techniques presented in the course of this thesis take this concept further by adopting mesh-free and local interpolation techniques in model-free and generalizing RL systems (i.e., LCS) for their utilization in real-valued input domains.

Interpolation-based Q-learning has been introduced by Szepesv́ari and Smart in [SS04]. Although the name clearly suggests a strong relation to the topic of this thesis, studying the contents reveals significant differences. The overall objective of this work is to apply Q-learning (again) to continuous state spaces and prove its convergence. More precisely, a function approximator is derived which provably converges to the optimal Q-values under the problem of continuous state spaces. The approach is denoted *interpolation-based* since particular assumptions for the derived function approximators are made. However, neither of the interpolation techniques and integration strategies which are developed here have been utilized, nor similar concepts. Again, the objectives show strong differences to those pursued in this thesis.

In a more recent work [Gu+17], another RL technique called *policy gradient* is extended to an interpolating variant for Deep RL. With their *interpolated policy gradient*, the authors propose to *mix* on-policy likelihood ratio policy gradient with a deterministic gradient based on an off-policy fitted critic. This mixing is essentially achieved by a convex combination of the aforementioned approaches. Accordingly, this technique does not constitute an interpolation-based extension in a similar sense as proposed in this thesis. Sampling points in terms of actually made experience tuples (s, a, r) are neither collected, nor created out of existing knowledge elements and then further used for interpolation as done in this work, e.g., for the interpolation of several classifier parameters which are subject to incremental updates via RL.

Looking Farther Away: Interpolation in Artificial Neural Networks

While not being directly related to the research reported in this thesis, the conducted literature review revealed recent further utilization of interpolation techniques in ANNs. For the sake of brevity and due to the lack of direct relation, these works are only very briefly mentioned here.

Scardapane et al. propose to learn activation functions by means of *cubic spline interpolation* in [Sca+19]. They argue that in most of today's neural network architectures, the neurons' activation functions are selected from a small set of well-investigated alternatives. The presented approach attempts to mitigate this shortcoming by incorporating interpolation to learn the activation functions for each neuron individually and, thus, obtain data-dependent adaptations.

In [Wan+18], *harmonic extensions* are used to interpolate the output activations in deep neural networks in order to improve the generalization accuracy and the ability to deal with insufficient training data. The approach is not directly applicable to online learning settings.

Williams introduced *Shepard Interpolation Neural Networks* (SINN) in [Wil16]. The proposed architecture is reminiscent of RBF networks. Essentially, SINN networks map Shepard's interpolation methods to a ANN architecture. As a result, memory and computation consumption can be reduced in comparison to fully connected networks while maintaining competitive accuracy.

4.6. Chapter Summary

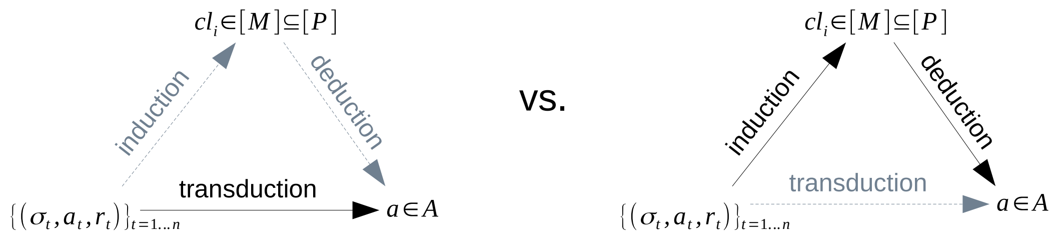
This chapter laid the foundation for facilitating the integration of interpolation techniques in XCS. First, a generic IC has been designed based on clearly identified requirements. Afterward, two architectural variants have been discussed: (1) XCS-IC, a loosely-coupled extension variant, where not only XCS but basically any OML algorithm can be attached to the IC via a so-called MLI. (2) XCS-CIC, a special-purpose variant for a tighter integration with XCS. Resulting possibilities and design decisions as well as ideas for further extensions of the IC were then thoroughly discussed. Related research aspirations that aim at integrating interpolation facilities in ML techniques have been appreciated in the last part of this chapter. With the developed IC at hand, the next chapter will introduce the first strategy to integrate interpolation in the algorithmic structure of XCS. More precisely, the action-selection step in the case of knowledge exploitation is supported by means of using the IC as transductive surrogate model.

Chapter 5.

Interpolation-Assisted Action Selection

The first strategic point for the integration of interpolation into XCS's algorithmic structure to be described in the following is the action-selection mechanism. This mechanism (also called *action-selection regime* in the literature), is responsible to decide for the next action to be executed a_{exec}. The decision is felt on the basis of the classifiers that match the current situation σ_t and, thus, can be interpreted as a mapping $S \to A$. In the context of RL, this mechanism is often called an agent's *policy* and denoted by π. Put in the context of SAS, it can also be understood as the *adaptation strategy*, which determines an adaptation in terms of control actions in response to the currently derived system state abstraction.

This chapter introduces a novel integration strategy that allows for making use of interpolation to facilitate adequate action-selection decisions – also in the presence of type-2 knowledge gaps. The IC is utilized to serve as a *surrogate* model for the aforementioned state-action mapping. It can be quickly accessed to retrieve transductively derived information from the sampling points collected thus far. Figure 5.1 illustrates this aspect and contrasts transductive with deductive inference.



$cl_i \in [M] \subseteq [P]$
induction
deduction
transduction
$\{(\sigma_t, a_t, r_t)\}_{t=1\ldots n}$
$a \in A$
vs.
$cl_i \in [M] \subseteq [P]$
induction
deduction
transduction
$\{(\sigma_t, a_t, r_t)\}_{t=1\ldots n}$
$a \in A$

FIGURE 5.1.: The difference between transductive inference and the conventional two-step induction/deduction way

It should be clearly noted that this technique does not aim at closing KGs in the sense of creating new or refining existing knowledge elements (i.e., classifiers) in order to deal with KGs of both types. Rather it is intended to support the SLAS to select and realize adequate adaptations in spite of missing or inexperienced knowledge elements. This integration strategy therefore strengthens the robustness against KGs. Figure 5.1 also illustrates the resulting difference in operation mode.

In the following sections, first the basic action-selection regimes are briefly recapitulated. Subsequently, the so-called *Action Selection Integration* (ASI) strategy is developed. *Action Selection Integration* (ASI) leverages interpolation weights derived from the distances of (neighboring) sampling points to the current situation in order to temporarily increase the prediction array value (the system prediction $PA(a)$) of those entries whose action is also transductively inferred by the IC. This technique will be introduced for both architectural extension variants of XCS by the IC, i.e., XCS-IC and XCS-CIC. An empirical validation of the introduced techniques on a common benchmark problem underpins the increased learning performance in comparison to a non-interpolation-assisted version of XCS. The chapter closes with a discussion about advantages and limitations of the ASI strategy for both manifestations and an appreciation of related work.

5.1. Basic Action-Selection Regimes

As already roughly described in Section 2.4, the basic action-selection regime of XCS depends on the current execution mode. This means, whether it currently explores the problem space or rather exploits the so far acquired knowledge base by greedily selecting the expected highest-rewarding action. The actual decision is based on the current prediction array PA. It contains the *system prediction* $PA(a_j)$ for each possible action $a_j \in A$. There exist a variety of action-selection regimes in the literature. The most frequently applied versions are:

- *Roulette wheel selection*, also more generally known as *fitness proportionate selection* in the GA domain
- *Greedy selection*, which chooses the *best* action with the *highest* system prediction
- *Random selection*, which ignores the system prediction and chooses an action purely by chance
- ϵ-*greedy selection*, a mixture of greedy and random selection

Where the concrete workings of greedy and random selection trivially follows from their designation, the first and the last one need to be clarified.

Roulette wheel selection This regime assigns each action $a_j \in [M]$ a share of an imaginary roulette wheel that is directly proportionate to its system prediction $PA(a_j)$. The system then "throws the ball", i.e., it draws a random number between 0 and the sum of all entries in PA. Given that the action shares are sorted in ascending order, the point where the "ball hits" a share, i.e., the action where the accumulated system predictions is only just greater than the random number, is eventually selected. This selection strategy implies that actions having a greater system prediction have a higher probability to get chosen. Nevertheless, by chance also actions with a smaller amount of share can be hit.

ϵ-greedy selection This technique constitutes a mixture of a pure greedy action-selection and a completely randomized choice. It is a well-known strategy to handle the exploration vs. exploitation trade-off in RL. With probability $\epsilon \in [0, 1]$ it selects an action by chance which corresponds to pure exploration of the problem at hand. Therefore, again a uniformly distributed random number between 0 and 1 is drawn. In case it is smaller than the selected ϵ, the agent explores its environment by discovering the effects of a randomly selected action. In the other case, the action is selected purely greedy, i.e.,

$$a_{exec} \coloneqq \operatorname*{argmax}_{a \in [M]} PA(a). \tag{5.1}$$

Naturally, when the system is in exploitation mode, the greedy selection is typically used. However, it also depends on the activated *switching mode* which determines when the system explores and when it is asked to exploit. In online learning settings with which XCS is typically confronted, an interleaving strategy between explore and exploit operations is often applied. Depending on the criticality of the learning problem at hand, this might result in unwanted *trial-and-error* situations. That is, when the system tries out a random action which might yield negative impacts on the system performance. In the context of real-world systems, in the worst case this could lead to human injury. Therefore, a 50% exploration rate might not be an adequate choice. In such cases, the ϵ-greedy strategy with a small ϵ alleviates the chance of such unwanted circumstances. One could also imagine to decay the value of ϵ with a growing number of situations the system has been exposed to. Of course, in situations where training data is available a priori, a *training phase* of full exploration could precede the exploitation phase which then only applies exploration with little or zero probability anymore (e.g., via roulette wheel selection).

The way OC proposes to handle this severe trade-off is to make use of *simulation-based machine learning* [Tom+11a; GR92]. The idea is to let the system explore *offline* on the second O/C layer without any effect on the productive SuOC at layer 0. As detailed in Section 3.1, this can be accomplished by relying on available simulation tools in combination with a capable optimization heuristic which seeks appropriate actions for a given simulated situation. The most adequate choice is, however, highly dependent on the problem characteristics and, thus, has to be made with caution – as is often the case in engineering SLAS.

5.2. Interpolation-Guided Action-Selection Regime

What all the conventional action-selection mechanisms have in common is that they only depend on the current prediction array, which can be interpreted as a summary of the currently available knowledge about a particular situation σ_t. The PA contains information about what the system expects to return (in terms of reward), when a particular action is realized. These values are calculated based on the several predictions $cl.p$ from classifiers that match the current situation and are further weighted by their niche-relative accuracy estimates $cl.F$. However, these classifiers might not have reached a sufficient level of accuracy. This would render them knowledge gaps of type 2. Therefore, one enhancement that is proposed in this thesis is to also incorporate information gained from past experiences. Here, these experiences are given by (σ, a)-pair sampling points, the system was able to collect so far. Two different means of how XCS can gain and further use such experiences are discussed below.

5.2.1. XCS-IC Approach

For the loosely coupled architectural variant of utilizing the IC as introduced in the previous chapter, the ASI strategy works as follows: Whenever a new σ_t is detected by XCS, the IC is triggered to calculate the interpolation weights w_i for any available sampling point $s_i \in SP$. Therefore, for the realization of this strategy, only interpolation methods of the class of distance-based or neighborhood-based approaches that explicitly calculate their interpolations on the basis of some sort of *weights* (cf. Sect. 2.3) constitute plausible candidates. The IC is configured to store sampling points of the form $s_i = (\sigma_t, a_{exec})$. Since actions (or classes) $a \in A$ are usually discrete categorical values, it is not sensible (but nevertheless possible) to interpolate between them, since this would imply that the actions are scalars from continuous domain. In order to still make use of transductively obtained values, for the ASI strategy only the interpolation weights are calculated and further used to guide the action-selection decision. The interpolation weights reflect the similarity (most often in terms of distance) between the sampling points' coordinates $\vec{x}_i$ and the query point $\vec{x}_q$. This information is deemed valuable to support the action-selection decision of XCS transductively.

Let the set of all weights w_i of corresponding sampling points s_i whose function values $f(\vec{x}_i)$ (here encoding an action $a \in A$) are equivalent to a_j be denoted by

$$W_{a_j} := \{w_i | f(\vec{x}_i) = a_j\}\,. \tag{5.2}$$

Then the *accumulated weight* of all sampling points that advocate action a_j is be defined as follows:

$$w_{acc}^{a_j} = \sum_{w_i \in W_{a_j}} w_i \tag{5.3}$$

Without loss of generality all $w_{acc}^{a_j}$ are normalized to the range $[0, 1]$.

Based on the normalized accumulated weights, the action to be suggested by the IC is determined by

$$a_{int} = \underset{a}{\operatorname{argmax}}\, w_{avg}^{a}. \tag{5.4}$$

In order to reflect the transductive information obtained via the IC within the action-selection step, the prediction array calculation is modified as follows:

$$PA'(a_j) = \min \left[PA(a_j) \cdot (1 + w_{acc}^{a_j} \cdot T_{IC}), R_{max}\right] \tag{5.5}$$

In the above equation and in the following, the constants R_{min} and R_{max} denote the lowest and highest possible rewards $r_{imm} \in R$ to be retrieved from the environment, respectively. As can be seen, the initial system prediction value is increased by $w_{acc}^{a_j} \cdot 100\%$, discounted by the trust-level $T_{IC} \in [0, 1]$.

The trust-level T_{IC} tracks the current interpolation quality of the IC. In the simplest form, T_{IC} is determined by the *average accuracy* determined by a sliding window over the latest interpolations. For instance on the basis of the last t_{window} rewards r_{imm} reported to the IC over the MLI. Combined with the accessible information of the actually executed action a_{exec} and the action as suggested by the IC (a_{int}), theIC's evaluation metric E can straightforwardly infer whether or not its guess was adequate. Beside such an accuracy measure, other metrics are imaginable. Examples are action (or class) sensitive measures from the classification / SL learning context such as precision or recall. However, a more thorough investigation of this particular aspect is left for future work.

With the ASI strategy, XCS's internal calculation to select an appropriate action a_{exec} is enhanced by transductive information obtained from the IC. More precisely, actions which more prominently appear among the current experiences $s_i \in SP$ in the proximity of the current situation σ_t gain higher system prediction values in the PA. Paired with a greedy selection, the ASI strategy constitutes a novel interpolation-based action-selection regime.

As for the evaluation metric E, based on the retrieved reward r_{imm} and the actually selected action a_{exec}, the IC's adjustment component can automatically decide on whether or not to create a new sampling point s^* and add it to SP. It is explicitly stated that this *sharing of experiences* between the XCS and the IC can be done in every step, regardless of whether it is an exploration or an exploitation step. Especially when the system explores the problem space by means of randomly selecting actions in situations where the classifiers are not experienced (i.e., are type-2 KGs), surprisingly successful actions should be added to SP. Thus, in the next exploitation step the IC can guide the action-selection toward this recent positive experience. One plausible way to implement this function is to check whether $r_{imm} = R_{max}$. If this assertion holds, a new sample $s^* = (\vec{x}_q, a_{exec})$ will be created. In realistic problems, the extremes of the reward signal are sometimes unknown. Then it might become necessary to define a threshold of minimum reward (or system utility) which needs to be surpassed in order to add a new sample s^*. The concrete design of a decision function A again strongly depends on the learning task at hand and typically requires a certain degree of domain knowledge. As a simple self-adaptive heuristic, the threshold could be set to the average reward the IC has seen so far.

As discussed in Chapter 4.2, the size of sampling point set SP is restricted to $|SP| \leq s_{max}$. The choice of this limit is not always straightforward and, again, dependent on the learning problem's complexity.

Remark If some sort of *expert knowledge* exists about how many environmental niches the problem at hand bears, a very simplistic rule-of-thumb could be: Let n denote the dimensionality of the input space $S \subseteq \mathbb{R}^n$ and $\tilde{N}$ be an estimate of the number of environmental niches characterizing the problem space. When furthermore a determines the desired number of sampling points for each environmental niche, then the maximum number of sampling points can be assigned the following value:

$$s_{max} := a^n \cdot \tilde{N} \tag{5.6}$$

This heuristic implies that if the problem consists of many niches (demanding for many specific classifiers), the number and the density of sampling points as well increases proportionally, and vice versa. Furthermore, the size of SP is exponentially increased with the number of input dimensions in order to pay tribute to the *curse of dimensionality.* As might be quickly recognized, this rule-of-thumb is only applicable for problem spaces with moderate input dimensionality n when more than one sampling point per niche is desired. Furthermore, the assumption of having a clue about the number of niches is probably not reasonable in most cases. Another implicit assumption is that any niche is expected to be of the same size.

As becomes apparent, the first introduced XCS-IC approach for realizing ASI comes at the cost of a number of design choices. An entirely different means to implement the ASI approach which completely omits part of those design choices, such as the size of SP or the decision function A, is introduced in the subsequent section.

5.2.2. XCS-CIC Approach

One of the obvious shortcomings of the previously introduced loosely coupled IC approach, is the moderate number of decisions that have to be made to incorporate ASI effectively and obtain the desired benefits. Two important IC-related hyperparameters to be set include: (1) the maximum number of sampling points to be collected s_{max}, and, (2) the choice of an appropriate decision function A (within IC's adjustment component) to control the maintenance the experience storage SP.

So far, a suggestion for s_{max} has been provided to serve as a first attempt. With that rule-of-thumb, the size of SP grows exponentially with the input space dimensions of the problem, as well as linearly with the estimated number of distinct environmental niches. However, this initial suggestion is not plausibly applicable in higher dimensional problem spaces. Nevertheless, for the use in SLAS as considered here the dimensionality of the system's state space (usually constructed out of certain sensor measurements and additionally derived metrics) is assumed to stay in lower ranges as discussed in Chapter 3. But even for problems of moderate dimensionality, it is not always possible to estimate the number of different niches a priori. Instead, the learning system itself is required to figure out an appropriate state-action mapping and the corresponding environmental niches during its runtime.[1]

A second SP-related issue is the applied policy to decide when new sampling points are added and, due to the limited size, which ones should be deleted. But at the same time without loosing useful experiences (*detrimental forgetting*). Depending on whether the learning agent is faced with single-step (typical in classification and regression tasks) or multi-step (sequential problems where the agent influences the state space sampling through its action sequences) problems, the aforementioned decisions have to be felt differently. The firstly mentioned single-step case can be implemented according to the methodology outlined above. More generally, this holds for all problems where a binary rewarding scheme (e.g., 0/1000 for false/correct) which clearly indicates the correctness of action decisions can be assumed. Typical episodic multi-step RL problems are often characterized by delayed reward signals. That is, a single high reward is only payed out when a predefined goal state is reached (e.g., the exit of a maze). The intermediate steps are rewarded with zero or small negative values. The second scheme stresses the search for shortest trajectories through the state space (e.g., to reflect limited battery resource of a robot). For the multi-step case, a sampling point of the form $s_i = (\sigma_t, a_t)$ inserted on the basis of

[1]Another related factor is of course the representational power of the individual XCS classifiers. As briefly mentioned in Chapter 2.4, various possibilities to encode a classifiers condition and, thus, the subspace for which it is responsible exist. Furthermore, the predictive capabilities of single classifiers strongly depend on the selected prediction model. Highly sophisticated techniques for condition representation and prediction modeling usually reduce the number of necessary classifiers. But on the other hand, they increase the algorithmic complexity and decrease the interpretability of the evolved solutions.

the received immediate reward r_{imm} is not sufficient anymore. Entire episode trajectories of the form $(\sigma_1, a_1, r_1, \sigma_2), (\sigma_2, a_2, r_2, \sigma_3) \ldots, (\sigma_{n-1}, a_{n-1}, r_{n-1}, \sigma_n)$ are needed in order to estimate the value (expected long-term reward or *return*) of a certain state-action pair (also Q-value [SB98]). The adaptation of the ASI technique based on the loosely coupled IC variant to allow for the use in multi-step problems with delayed rewards is not part of this thesis and therefore left for future work.

Remark Storing sampling points of this type is reminiscent of a technique which recently has become known as *experience replay.* It has gained a lot of interest and plays an essential role in reaching human-level performance in various games with (Deep) RL agents [Mni+15].

The aforementioned thoughts motivate the second methodology of realizing the ASI strategy as will be introduced now. This approach is based upon the tightly integrated architectural variant of incorporating the IC into XCS – XCS-CIC. It allows for moving the decisions regarding the sampling point management from the IC to the responsibility of XCS itself. Furthermore, this variant eradicates the non-applicability to multi-step problem, which will become obvious in the next paragraphs.

As already described in the preceding chapter, the sampling point set SP is replaced by the current rule base $[P]$, or at least by particular subsets of it. More precisely, each classifier cl_i serves as (at least) one sampling point s_i and provides a variety of possible function values to be interpolated. For instance, $cl_i.p, cl_i.\epsilon, cl_i.F$, etc.

In order to being used with the ASI technique, the sampling points are extracted from the classifiers as follows:

Let $cl_i.\vec{c} \in cl_i.C \subseteq S \subseteq \mathbb{R}^n$ denote the center point of the interval-based condition of a classifier cl_i defined by

$$cl_i.\vec{c} = \frac{1}{2}\left((u_1 - l_1), (u_2 - l_2), \ldots, (u_n - l_n)\right)^T. \tag{5.7}$$

Then an individual sampling point s_i extracted from the respective cl_i is given by the tuple:

$$s_i \coloneqq (cl_i.\vec{c}, cl.a) \tag{5.8}$$

In contrast to the former approach, no sampling points (experiences) are actively collected anymore. Instead, they are extracted from the existing classifiers cl_i in an ad-hoc fashion. Thus, the sampling point store (experience memory) now continually changes with progressing evolution of $[P]$. XCS is thus enabled to completely handle the insertion and deletion of sampling points (i.e., classifiers) by means of its internal replacement mechanisms (cf. Ch. 2.4). This, in turn, renders the IC's adjustment component and consequently also the decision function A obsolete. Also the size of the sampling point set SP is now directly controlled by the XCS hyperparameter N, which limits the maximum number of microclassifiers kept in the population. It should be emphasized that the sampling points are constructed from *macroclassifiers* which are actually stored in the data structure $[P]$. In learning problems where the prevalent niches demand for more general classifiers, the number of available sampling points decreases and vice versa. In order to counteract this dependence, more than one sampling point can be extracted from one particular classifier as discussed in Chapter 4 before. One possibility to reach a number of sampling points equal to N is to draw exactly $cl_i.num$ sampling points from each classifier $cl_i \in [P]$. A thorough investigation of this aspect is another aspect of future work.

The size of $[P]$ must be chosen adequately in order to overcome several *learning challenges* of XCS [But05a]. It needs to be chosen large enough to guarantee a complete input space coverage, schema supply, sufficient opportunities for reproduction as well as solution sustenance. To master all the aforementioned challenges, LCS populations naturally do not contain only accurate and maximally general rules. Instead, they also bear a great share of so-called *transient* rules. These transient rules are necessary for exploring the problem space, i.e., to discover niche representations that should be optimized further by the GA. If such a transient rules proves to be successful, it will get promoted further and is likely to survive. Otherwise, after some time it will be selected for deletion in favor of a novel rule which has been constructed by the discovery component. Transient rules clearly bear the risk of being *type-2 knowledge gaps* and should be excluded from interpolation. Therefore, a classifier filter which omits weak classifiers which cannot surpass a certain quality[2] threshold θ_q is defined below:

Let $q(cl)$ be a function $q : [P] \to \mathbb{R}, cl \mapsto q$ which fulfills the following properties:

1. Quality estimates are positive including zero
2. Classifiers with greater experience gain increasing quality
3. Classifiers predicting high rewards $cl.p$ with high fitness (niche-relative accuracy) $cl.F$ gain higher quality

A reasonably simple function which entirely reuses existing classifier parameters and completely meets the above requirements is given by:

$$q(cl) = cl.exp \cdot cl.F \cdot cl.p \tag{5.9}$$

[2]The term "quality" in this specific context solely refers to the *value* of a classifier for being considered as a sampling point for the upcoming interpolation.

Accordingly, the filtered sampling point set SP' in view of its utilization with the ASI strategy can be defined as follows:

$$SP' := \left\{ (cl_i.\vec{c}, cl_i.a) \;\middle|\; q(cl_i) > \frac{1}{|[M]|} \sum_{cl_i \in [M]} q(cl_i) \right\}_{i=1\ldots|[P]|} \tag{5.10}$$

As can be noticed, instead of involving the entire population, only the current match set $[M]$ is considered for constructing sampling points. The rationale behind this decision is to exclude specialized (i.e., non-general) classifiers situated too distant from the current situation σ_t within the state space S, even if they might be highly accurate. It is assumed that the valuable knowledge for transductively modifying the system prediction is found in the immediate input space niches which are represented by the current classifiers in $[M]$. The rest of the ASI approach works analogously to the XCS-IC variant introduced in the previous section.

Since the active maintenance of an adequately collected set of sampling points SP is now obsolete, certain shortcomings of previously introduced XCS-IC approach to realize ASI are eradicated as well. With the just defined methodology, the ASI technique is now straightforwardly applicable to (multi-step) RL tasks even without the simplifying binary reward scheme. This is due to the fact that the classifiers in $[M]$ are filtered by their fitness weighted prediction values. When a classifier is not able to predict the expected long-term reward correctly (estimated by $cl.p$), its fitness decreases exponentially, what in turn yields lower quality estimates. By applying the designed filter (Eq. 5.10), only more accurate and higher reward predicting rules will be considered for the sampling point extraction. Accordingly, when ASI is applied, those actions promising higher long-term rewards are supposed to be higher influence on the prediction array modification as given by Equation 5.5.

In general, the decision which sampling points are considered for interpolation, is now implicitly felt by the applied classifier filter. By additionally considering only $[M]$, more situation-dependent sampling point sets SP' are created. This stays in contrast to one dedicated central SP which contains experiences from all niches that have been visited so far. The effect is that misleading sampling points from weak classifiers are discarded and the IC-based surrogate is enabled to better support the decision on appropriate actions.

Regarding the *transductive nature* of the ASI approach, the XCS-CIC variant follows a similar but slightly different way. Instead of reusing raw experiences collected within SP, here the additional transductive information is directly inferred from already existing knowledge elements (i.e., $cl \in [P]$), again without the intermediate steps of induction and deduction.

5.3. Evaluation

In order to validate the benefits of the ASI technique for both architectural variants, the following paragraphs are concerned with the results of an empirical evaluation regarding the impacts of interpolation-guided action-selection in XCSR.[3]

5.3.1. Checkerboard Problem

For the purpose of evaluation, it is relied on a well-known benchmark problem for LCS applied to real-valued input domains – the *Checkerboard Problem* (CBP) as introduced in [SB03]. According to the theoretical analysis of Butz in [But05a], the CBP constitutes one of the maximal challenging problems for XCS, since its general structure can prevent it from overcoming the challenge of guaranteeing a sufficient fitness signal and thus niche supply, growth and finally sustenance. The general form of a CBP instance is depicted in Figure 5.2.

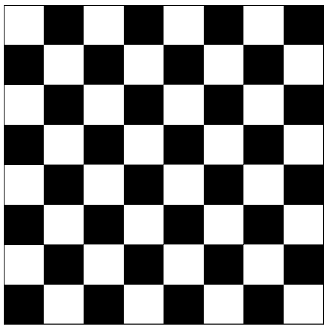


FIGURE 5.2.: An exemplary CBP instance with two dimensions and eight divisions each

The illustrated CBP instance has $n = 2$ dimensions and $n_d = 8$ divisions in each dimension. This results in a typical checkers or chess board (from which its name hails). It is denoted by CBP$(n = 2, n_d = 8)$. The task of a learning agent is to decide for any possible state $\sigma_t \in S \coloneqq [0, 1]^n$ of which color the field is which encompasses the situation vector σ_t. Thus, the action space A is $A \coloneqq \{\text{white}, \text{black}\}$. The CBP constitutes a single-step (or classification) problem with binary reward scheme, i.e., $r_{imm} \in \{0, 1000\}$. A reward of 1000 is payed out for correct answers, and a 0 reward for incorrect guesses. In order to control the problem complexity, the parameters n and n_d can be modified to arbitrarily high-dimensional and partitioned hypercubes. The problem difficulty results from the numerous, i.e., $n \cdot n_d$ input space niches that need to be identified and sustained. XCS has to achieve this without (1) steering

[3]Please note that the incorporation within XCSF [Wil02] is not plausible, since it is targeted at approximating functions what refers to a regression task, rather than a control problem. In later chapters, more interpolation-based XCS extensions will be developed and evaluated within XCSF's algorithmic structure.

toward a covering-deletion cycle, or else, (2) population take-over by overgeneral classifiers due to a lack of fitness signal. The CBP was devised to eradicate the weaknesses of the formerly often used *real multiplexer problem* [Wil00], which was found to respond to biases of early real-valued condition representations (cf. [SB03] for a detailed analysis). It also has been noted in the study that CBP resembles real-world problem characteristics more strongly.

Remark on the Relation to Realistic Problems In order to bring the choice of this particular "toy" problem and the general class of single-step problems in line with the formulated learning tasks of SLAS (cf. Ch. 3), the exemplary traffic management scenario shall again be adduced for clarification purposes. Different traffic situations, that is input-output vehicular flows between the possible turnings of an urban intersection, demand for different traffic signal timings, which are provided via specific signal plans. Assuming a somewhat simplified view in the following, a signal plan can be modeled as follows: For each of n traffic signal groups (individual traffic lights electrically wired to show the same light signal), the duration d of the green phase (in sec.) can be adapted freely within certain bounds. This yields a large (theoretically continuous), multi-dimensional action space $A := [d_{min}, d_{max}]^n$. Since XCSR is not designed to handle *continuous action spaces*, consider further, that an optimization procedure obtains a number of solution candidates $a_i \in A$ that have to compete against each other and against a predefined human-engineered signal plan as well. A conceivable learning task for a SLAS would then be to learn which signal plan to apply in which situation. Therefore, XCSR would approximate the complete $S \times A \to P$ payoff map. This is reminiscent of classical *mode switching* in technical systems. As a feedback signal, the learning agent receives a positive reward when the mode switch leads to an increase in system utility. A negative or zero reward is received otherwise. Assuming more than two distinct modes (here signal plans), also more fine-grained reward levels would support the learning of a capable adaptation strategy.

5.3.2. Experimental Setup

For the sake of evaluating both ASI methodologies and in order to gain insights of XCS's learning behavior, benchmark problems such as the CBP are often selected. Such synthetic problems allow for a straightforward analysis of the problem complexity, for derivable hyperparameters and for an intuition regarding the expected behavior of the learning algorithm under investigation.

Standard XCSR [Wil00] serves as a baseline here.[4] All reported experiments are conducted for 30 *independent, identically distributed* (i.i.d.) repetitions with individual random seeds. All depicted plots show the learning curves averaged over

[4]In the following XCS and XCSR are used interchangeably.

the 30 runs over a predefined learning period of alternating explore/exploit trials. Only the exploit trials are plotted and further assessed for pure system performance evaluation.[5] The plotted points of all figures of merit show aggregated values over a predefined number of exploit trials each. The error bars depicted for each point represent the standard deviation ($\pm$1SD).

Any evaluation metric is assessed beforehand with respect to whether it follows a normal distribution. Therefore, *Quantile-Quantile Plots* (QQ-plots) and *Shapiro-Wilk tests* have been conducted. If not stated differently a *paired t-test* is used for hypothesis testing of statistical significance between two configurations. Otherwise, if the requirements for using parametric significance tests are not fulfilled, the non-parametric *Wilcoxon's signed-rank test* will be used instead. The significance results are shown in the corresponding tables, where a single asterisk (*) indicates a significant differences with a significance level of $\alpha = 0.05$, and two asterisks (**) denote highly significant differences ($\alpha = 0.01$) between the particular configuration and the baseline.

For the CBP, two instances with different complexity are selected – CBP(3,3) and CBP(3,6). They have found to be sufficiently challenging for XCS to learn, but not too complex to run into computational time issues due to slow convergence. XCSR with hyperrectangular conditions by virtue of its problem space partitioning learning intuition seems to be a perfect fit to solve this problem. However, as already stated above, it is really difficult to capture all perfectly disjoint environmental niches and to evolve an optimal solution $[O]$ as defined by Kovacs in [Kov98]. An optimal solution for CBP(3,6), would comprise exactly $n_d^n \cdot |A| = 6^3 \cdot 2 = 432$ rules, for instance. As can be seen below, a maximum number of classifiers with a factor of approximately 20 is needed to get a nearly accurate result. The sampling of the input space follows a uniform distribution since for the case of CBP this constitutes the most challenging configuration.

Evaluation Metrics The performance of XCS is evaluated by means of observing the progress of three figures of merits as is typically done in the literature: (1) the fraction of correct classifications (*Fraction Correct*) which reflects a ratio of correct and incorrect actions over the last 100 steps when XCS decided in its exploitation mode. For instance, when XCS guesses correct for 30 out of 100 problem instances σ_t, than the metric evaluates to 0.3. This metric is equivalent to the *accuracy* measure which is typically used in the supervised learning domain. (2) A *mean absolute error* (MAE) measure which determines the difference of the system prediction $PA(a_{exec})$ and the actually received reward r_{imm}, again averaged over the last 100 exploit trials. For the utilization of the ASI strategy, the interpolation-assisted prediction array calculation $PA'(a_{exec})$ is used instead. (3) The average number of macroclassifiers

[5]This is a standard methodology to account for the exploration/exploitation trade-off and which is mostly applied in the literature studying LCS.

present in $[P]$ to indicate the generalization success of the system, again averaged over the last 100 exploit trials. The plots depicted in the following thus show the progressions of these metrics over the observed learning periods.

5.3.3. Configuration of ASI Strategy

For the XCS-IC implementation variant of ASI, the maximum number of sampling points to collect is set to $s_{max} = 2000$ for the CBP(3,3) and to $s_{max} = 20000$ for the CBP(3,6), respectively. This yields a comparability with the XCS-CIC approach which will be evaluated below, since here the maximum number of sampling points is limited by the maximum number N of microclassifiers allowed in the population. XCS shares experience triplets of the form $(\sigma_t, a_{exec}, r_{imm})$ in each exploitation step with the IC, which in turn simply decides on the basis of the reward r_{imm} whether to create and insert a new sampling point $s^* = (\sigma_t, a_{exec})$ into SP.

The both IC-extended XCS variants using ASI are compared with standard XCS as a baseline.

Three different interpolation methods, namely

- *Nearest Neighbor* (NeNe)
- *Inverse Distance Weighting* (IDW)
- *Modified Shepard's Method* (MSM),

are applied to fathom their individual potentials of being used within the IC. Those particular interpolation methods have been selected, since they constitute representatives from different categories of interpolation techniques: (1) NeNe serves as naïve baseline method, (2) IDW represents the class of global interpolation, and, (3) MSM covers the local techniques. The NaNe interpolation technique is restricted in being plausibly applicable to higher input dimensions with regard to stable and efficient computations [LH10]. MSM does not exhibit comparable restrictions. Thus, it was decided for using MSM interpolation as representative of the local-support methods.

Experiment 1 - XCS on CBP(3,3) In the first experiment, XCS is asked to learn a 3-dimensional, 3-fold-divided hypercube (CBP(3,3)). This configuration hardly challenges a standard XCSR. It is selected to demonstrate that even for non-challenging problems, an increase in learning efficiency can be achieved. Based on suggestions from the literature [SB03; But05a], XCS is configured as follows: $N = 2000, \alpha = 0.1, \beta = 0.2, \delta = 0.1, \epsilon_0 = 10, \nu = 5, \theta_{mna} = 2, \theta_{GA} = 12, \theta_{sub} = 20, \theta_{sub} = 20, p_{ini} = 10, \epsilon_{ini} = 0, F_{ini} = 0.1, F_{reduce} = 1.0, \epsilon_{reduce} = 1.0, \mu = 0.04, \chi = 0.8, r_0 = 1/n_d = 0.33, m_0 = 0.1$. GA subsumption is active. Action are not mutated. Hyperrectangular condition representation with UBR encoding is used. Uniform crossover and niche-relative tournament selection with a proportion of $\tau = 0.4$ was used in the GA. An interleaving strategy with alternating explore and exploit trials for handling the exploration vs. exploitation trade-off is applied. For a detailed description of the standard parameters please refer to Appendix B. As can be noted, the maximum number N of microclassifiers is restricted to 2000. Accordingly, the maximum number of sampling points s_{max} is set to N, as this allows for a comparison between the XCS-IC and the XCS-CIC concept. This restriction only affects the former approach, since for the latter the sampling points are determined by the current population, i.e., $SP' \subseteq [P]$. For this first experiment on CBP(3,3), XCS learns for $100k$ explore/exploit trials for each repetition.

Experiment 2 - XCS on CBP(3,6) In the second experiment the number of divisions n_d is doubled to increase the task complexity. This causes a challenging rise of necessary learning steps for XCS. More precisely, $400K$ explore/exploit trials are needed to observe a convergence of the learning progress. Parameters are set as for the first experiment, except of: $N = s_{max} = 20000$, $r_0 = 1/n_d = 0.167$.

5.3.4. Results

The following paragraphs discuss the impacts of ASI on XCS's learning behavior and describe general findings. Tables 5.1, 5.2 and 5.3 summarize the results for each interpolation technique on both CBP instances. Bold values indicate statistically significant improvements in comparison with standard XCS. The table entries are the mean values and standard deviations obtained from the conducted 30 i.i.d. experiment repetitions. The * (**) symbols indicate statistically (highly) significant deviations of the reported metrics compared to standard XCS. This means that for the p-values of paired t-tests the assertion $p < \alpha = 0.05$ (0.01) is true. Up and down arrows indicate whether the value has increased ($\uparrow$) or decreased ($\downarrow$) in comparison to standard XCS.

Performance of ASI

By means of using the ASI strategy clear differences in XCS's learning performance can be noticed. When utilized in the loose-coupling variant (XCS-IC), the ASI strategy always leads to an increased average fraction of correct predictions as well as to a substantially decreased averaged system error compared to standard XCS. Since the ASI strategy is intended as an exploitation action-selection regime, it has no direct effect on the creation of classifiers and accordingly on the average number of macroclassifiers. If ASI would also be used in the exploration phase, the discovery of the problem space would be restricted to already seen sampling points, what contradicts the idea of exploring the problem space by allowing *trial-and-error*

Table 5.1.: Summary of ASI results on CBP(3,3) and CBP(3,6) using Nearest Neighbor interpolation

CBP(3,3)	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
XCS-IC w/ **ASI**	$\mathbf{951.53}^{\uparrow **} \pm 2.93$	$\mathbf{64.15}^{\downarrow **} \pm 3.30$	$669.51^{=} \pm 11.99$
XCS-CIC w/ **ASI**	$945.75^{\downarrow **} \pm 4.19$	$\mathbf{64.02}^{\downarrow **} \pm 4.44$	$669.51^{=} \pm 11.99$
Standard XCS	947.97 ± 3.89	105.56 ± 5.30	669.51 ± 11.99
CBP(3,6)	*Fraction Correct*	*System Error*	*Macro-Classifiers*
XCS-IC w/ **ASI**	$\mathbf{898.79}^{\uparrow **} \pm 3.05$	$\mathbf{130.04}^{\downarrow **} \pm 4.64$	$8761.07^{=} \pm 76.05$
XCS-CIC w/ **ASI**	$867.14^{\downarrow **} \pm 5.17$	$\mathbf{154.98}^{\downarrow **} \pm 7.27$	$8761.07^{=} \pm 76.05$
Standard XCS	876.62 ± 4.67	220.97 ± 8.30	8761.07 ± 76.05

Table 5.2.: Summary of ASI results on CBP(3,3) and CBP(3,6) using Inverse Distance Weighting interpolation

CBP(3,3)	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
XCS-IC w/ **ASI**	$\mathbf{950.72}^{\uparrow **} \pm 3.79$	$\mathbf{62.42}^{\downarrow **} \pm 4.06$	$669.51^{=} \pm 11.99$
XCS-CIC w/ **ASI**	$947.96^{\downarrow} \pm 4.03$	$\mathbf{61.23}^{\downarrow **} \pm 4.24$	$669.51^{=} \pm 11.99$
Standard XCS	947.97 ± 3.89	105.56 ± 5.30	669.51 ± 11.99
CBP(3,6)	*Fraction Correct*	*System Error*	*Macro-Classifiers*
XCS-IC w/ **ASI**	$\mathbf{880.68}^{\uparrow **} \pm 4.36$	$\mathbf{149.68}^{\downarrow **} \pm 6.91$	$8761.07^{=} \pm 76.05$
XCS-CIC w/ **ASI**	$874.87^{\downarrow **} \pm 4.80$	$\mathbf{146.80}^{\downarrow **} \pm 6.42$	$8761.07^{=} \pm 76.05$
Standard XCS	876.62 ± 4.67	220.97 ± 8.30	8761.07 ± 76.05

Table 5.3.: Summary of ASI results on CBP(3,3) and CBP(3,6) using Modified Shepard Method interpolation

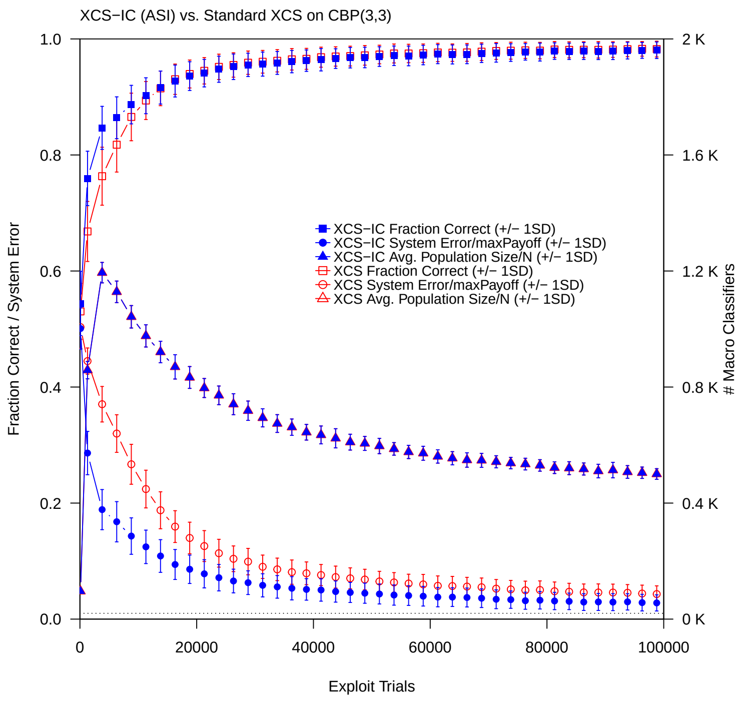
CBP(3,3)	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
XCS-IC w/ **ASI**	$\mathbf{955.18}^{\uparrow **} \pm 3.04$	$\mathbf{56.80}^{\downarrow **} \pm 3.26$	$669.51^{=} \pm 11.99$
XCS-CIC w/ **ASI**	$947.85^{\downarrow} \pm 4.04$	$\mathbf{61.26}^{\downarrow **} \pm 4.25$	$669.51^{=} \pm 11.99$
Standard XCS	947.97 ± 3.89	105.56 ± 5.30	669.51 ± 11.99
CBP(3,6)	*Fraction Correct*	*System Error*	*Macro-Classifiers*
XCS-IC w/ **ASI**	$\mathbf{902.76}^{\uparrow **} \pm 2.78$	$\mathbf{122.79}^{\downarrow **} \pm 4.43$	$8761.07^{=} \pm 76.05$
XCS-CIC w/ **ASI**	$874.61^{\downarrow **} \pm 4.86$	$\mathbf{146.96}^{\downarrow **} \pm 6.64$	$8761.07^{=} \pm 76.05$
Standard XCS	876.62 ± 4.67	220.97 ± 8.30	8761.07 ± 76.05

experiences. The MSM technique for local interpolation yields superior performance compared to both NeNe and IDW. The global IDW approach obtains the smallest improvements. Figures 5.3 to 5.8 corroborate this observation visually for both CBP instances. In summary, using XCS-IC with ASI increases the fraction correct metric slightly (up to 0.76% for MSM) over the entire learning steps for the CBP(3,3) scenario. Nevertheless, the improvements have been found statistically significant for each of the applied interpolation techniques. More prominently appearing improvements are observable for the more complex CBP(3,6) scenario, where up to 2.98% for the MSM technique have been obtained. It has to be noted that the stated percentages cover the entire learning periods of $100k$ and $400k$ exploit trials for CBP(3,3) and CBP(3,6), respectively. However, the most distinct differences between the interpolation-assisted XCS-IC and standard XCS appear at the early learning phases (until $\approx 20k$ for CBP(3,3) and $\approx 200k$ for CBP(3,6)). Especially in these initial phases, knowledge gaps of both types are highly present. Accordingly, much higher improvements regarding the fraction correct metric become apparent in these learning periods (cf. the corresponding plots).

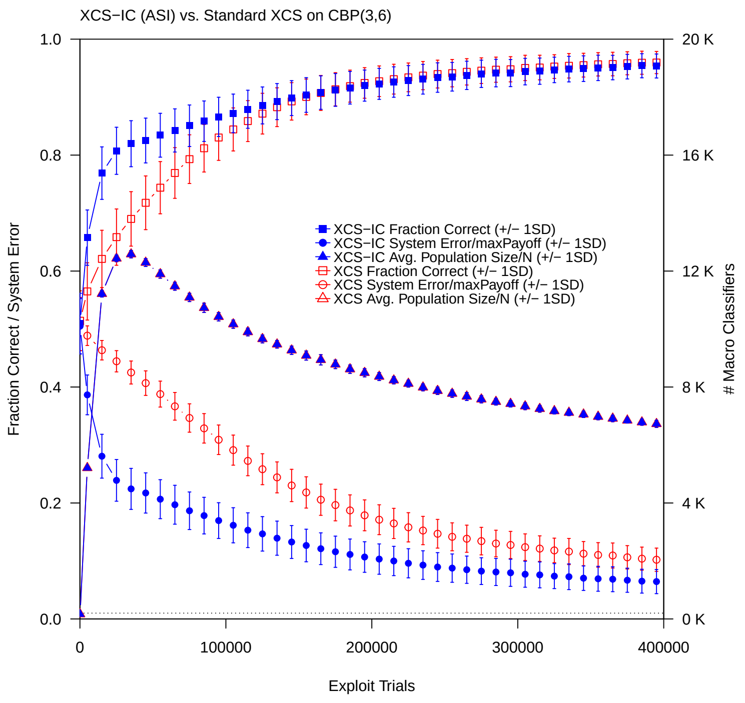
Inspecting the second metric, the MAE or system error, strong improvements up to 46.19% for MSM interpolation on CBP(3,3) and up to 44.43% on CBP(3,6) can be observed. This indicates a more accurate approximation of the underlying $S \times A \rightarrow P$ payoff landscape XCS strives to learn. Again, these relatively stated improvements involve the entire learning process and even higher differences can be observed during the initial learning phases.

Another insight is that for each figure of merit, the standard deviations turned out to be smaller in contrast to standard XCS. This points toward a slightly increased robustness with regard to the stochastic nature of the system, which is attributed to the deterministic way of obtaining tranductively inferred information in order to the guide the action-selection step.

As for XCS-IC and throughout all experiments, the ASI strategy leads to a substantially reduced average system error when utilized in the XCS-CIC variant. However, in contrast to the results obtained with XCS-IC, slight and partially statistically significant degradations in terms of the average fraction of correct predictions can be recognized. The revealed reductions are up to 0.23% on CBP(3,3) and 1.08% on CBP(3,6) for the NeNe interpolation technique which has been found to perform worst. The occurrence of this effect is attributed to the reliability of the classifier-extracted sampling points. Especially at the beginning of a learning task, the classifiers are less accurate what in turn leads to a lower mean quality for the preselection filter of classifiers to appear in SP'. Furthermore, most of the niches are initially covered by different classifiers advocating both possible actions of the action space A (controlled by θ_{mna}). This in turn leads to a higher accumulated weight $w_{acc}^{a_j}$ for the actually incorrect action at early stages of learning. The observations are further substantiated by slightly increasing standard deviations with regard to the obtained values for the fraction correct metric. In order to eradicate this limitation,

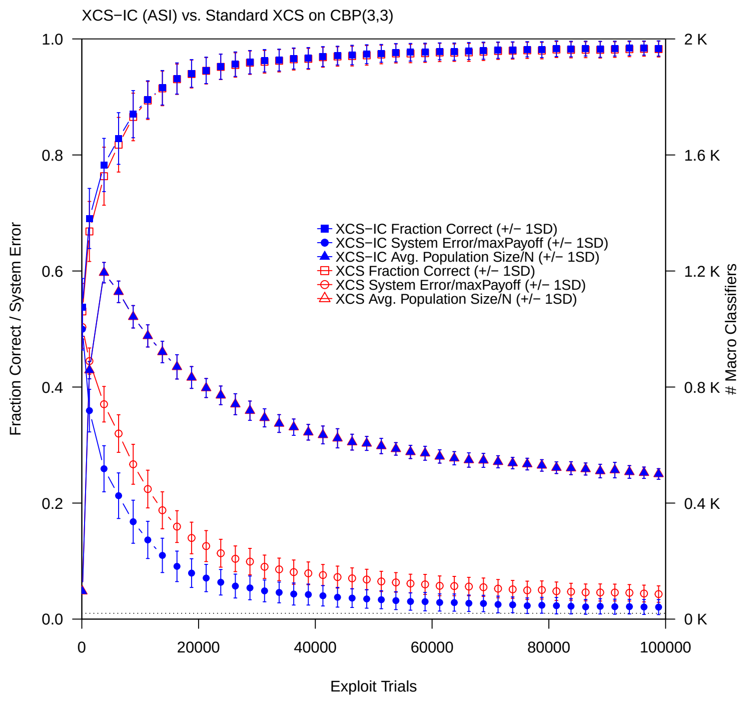


(A) CBP(3,3): NeNe

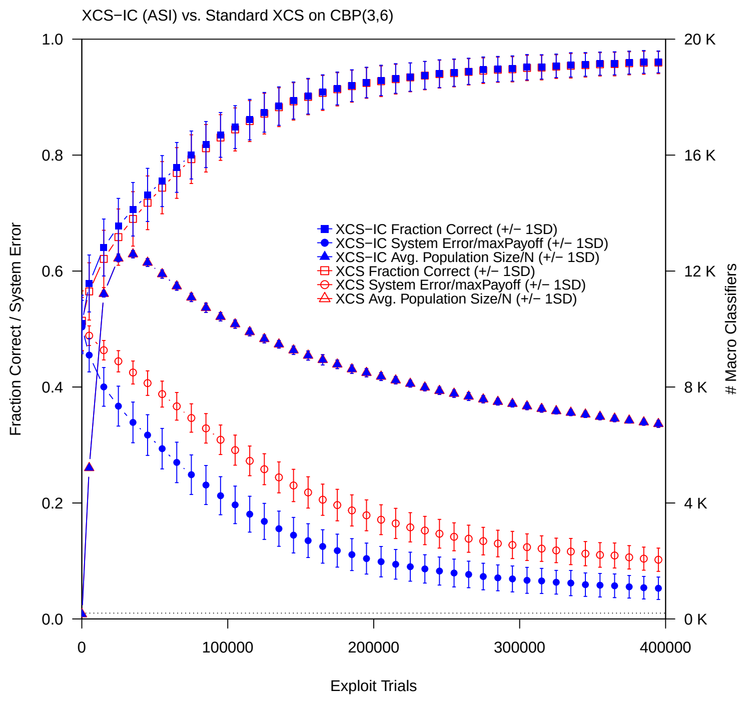


(B) CBP(3,6): NeNe

FIGURE 5.3.: Comparison of XCS against XCS-IC using ASI with naïve NeNe interpolation on CBP(3,3) (top) and CBP(3,6) (bottom)

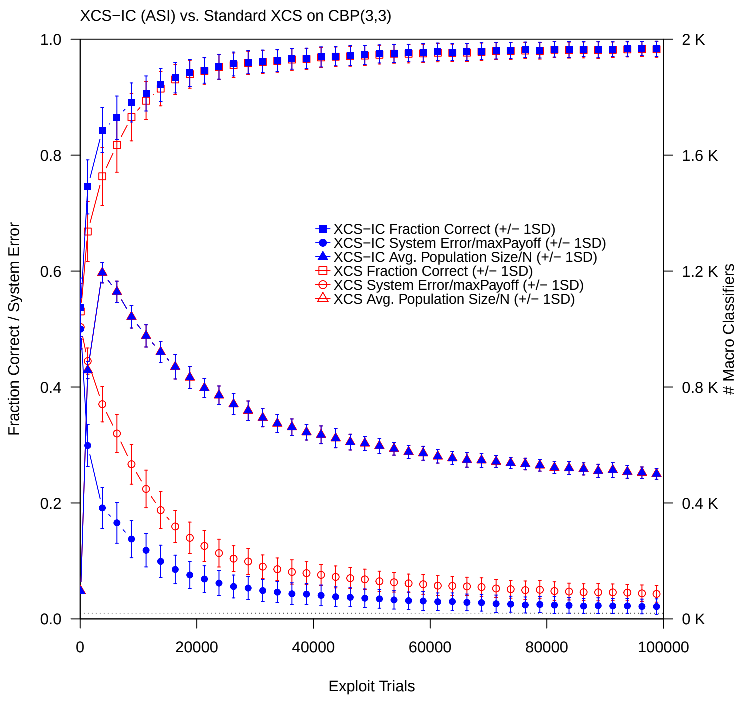


(A) CBP(3,3): IDW

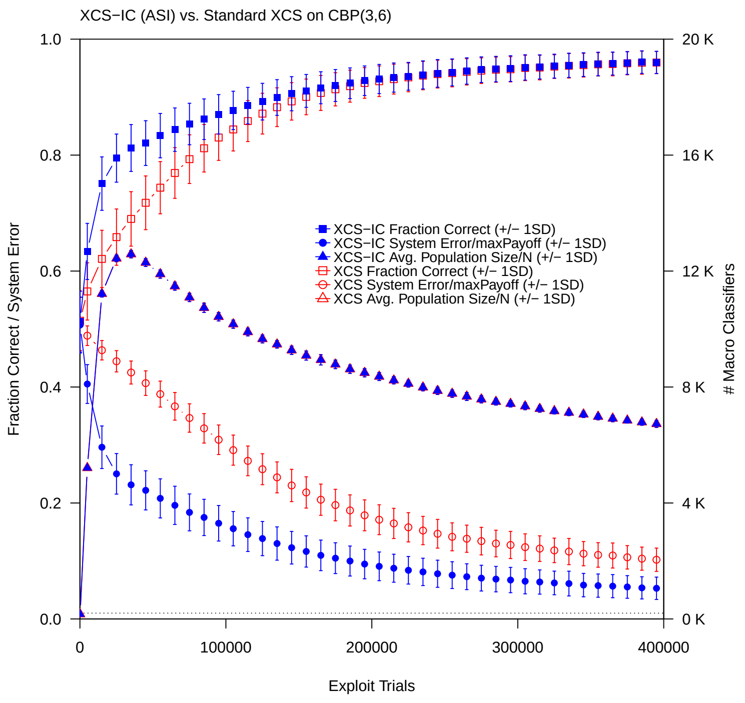


(B) CBP(3,6): IDW

FIGURE 5.4.: Comparison of XCS against XCS-IC using ASI with global IDW interpolation on CBP(3,3) (top) and CBP(3,6) (bottom)



(A) CBP(3,3): MSM



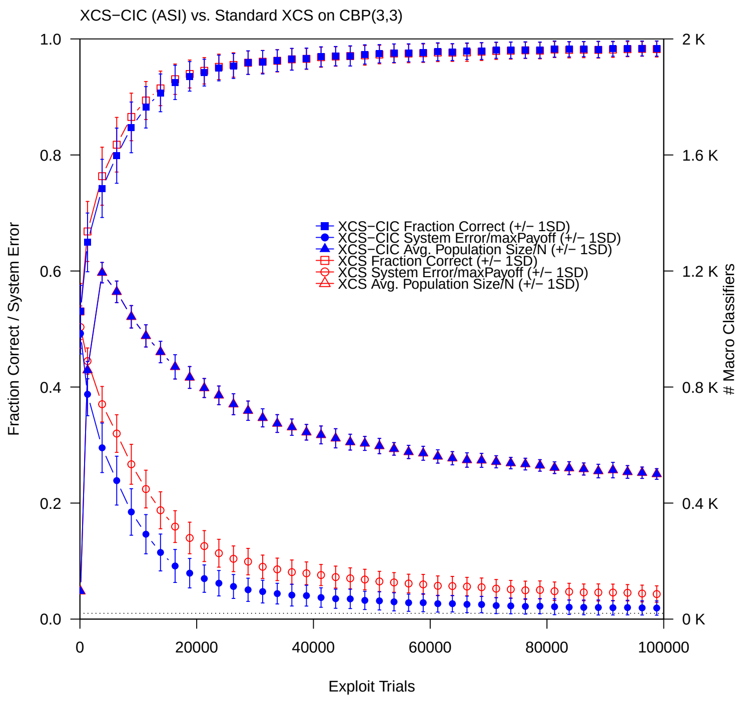
(B) CBP(3,6): MSM

FIGURE 5.5.: Comparison of XCS against XCS-IC using ASI with local MSM interpolation on CBP(3,3) (top) and CBP(3,6) (bottom)

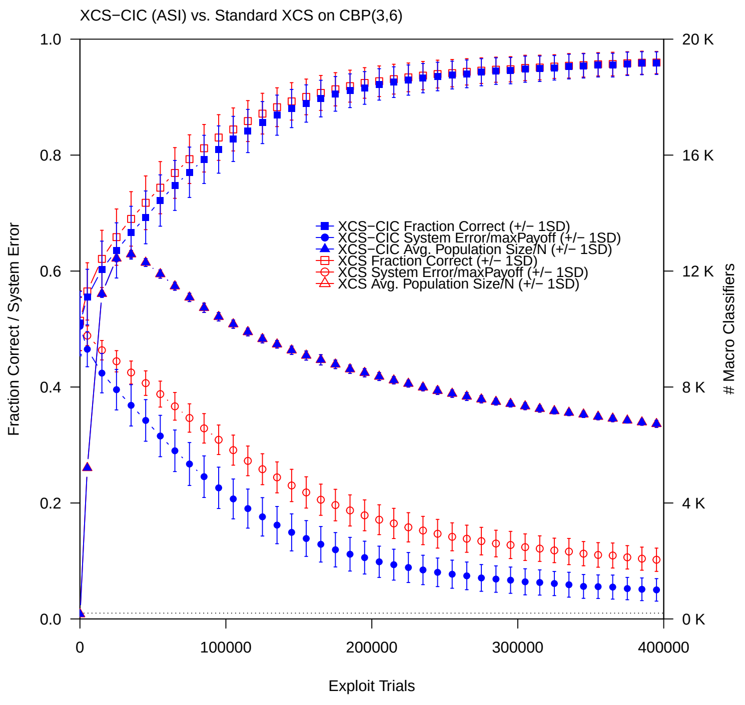
one approach would be to define a trade-off function that controls the impact of the transductively (i.e., interpolation-assisted) derived prediction array modification of Equation 5.5. The derivation of such a functional relationship that trades off the reliability of the interpolation-assisted transductive means against the conventional deductive action selection step (cf. again Fig. 5.1) is, however, left for future work on the ASI strategy.

Regarding the system prediction error (MAE), still substantially improved results can be observed. Interestingly, the IDW technique yields the best result in terms of numbers, reducing the system error by 42% and 33.57% on average for CBP(3,3) and CBP(3,6), respectively. Nevertheless, the actual differences between IDW and MSM can be considered neglectable. On both CBP instances, the naïve NeNe technique yields the worst results regarding the MAE. Nevertheless, the reported improvements regarding the system error metric for all interpolation methods have been found to be statistically significant.

A further relevant insight from the reported experimental results is that MSM constitutes the most promising technique to be incorporated for ASI in XCS-IC and in XCS-CIC as well. In most of the reported cases MSM has been found to outperform the contending interpolation techniques NeNe and IDW. This observation is attributed to the fact that MSM only uses a local neighborhood of the current situation out of all the available sampling points. This seems to be beneficial in highly multi-modal problem domains such as the CBP where frequent changes of the correct actions and, thus, reward levels characterize the problem landscape (i.e., $S \times A \to P$). Incorporating too many sampling points in the actual determination of the accumulated weight $w_{acc}^{a_j}$, might lead to detrimental effects which lessen the possible benefits of utilizing transductively obtained information on past experiences. However, this is expected to be highly dependent on the characteristic of the underlying problem space – as is the case for nearly any decisions that have to be made a priori at design time. On the other hand, using the naïve NeNe approach turns out to incorporate too few knowledge from already experienced situations stored in SP or rather encoded within the currently available classifiers in $[M]$. Local interpolation techniques such as MSM turn out to constitute a plausible middle way with the possibility to change the size of the considered neighborhood. Such a locality adaptation could also be achieved in a self-adaptive manner at runtime. However, concrete techniques to accomplish this behavior is out of the scope of this work.

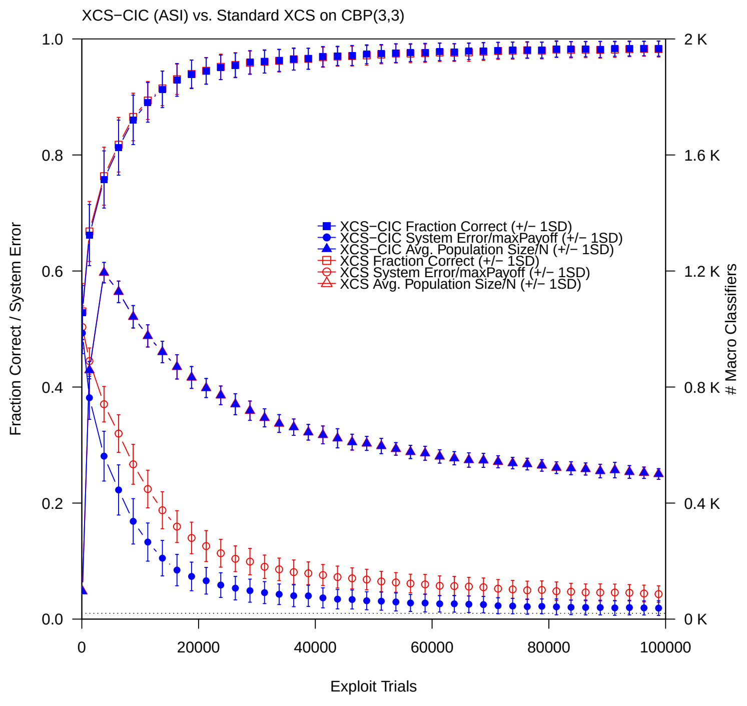


(A) CBP(3,3): NeNe

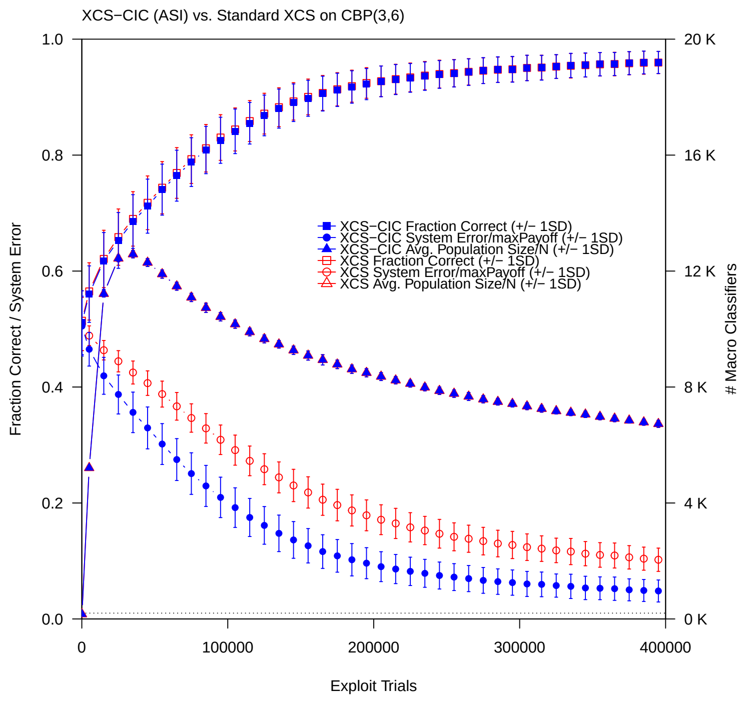


(B) CBP(3,6): NeNe

FIGURE 5.6.: Comparison of XCS against XCS-CIC using ASI with naïve NeNe interpolation on CBP(3,3) (top) and CBPs(3,6) (bottom)

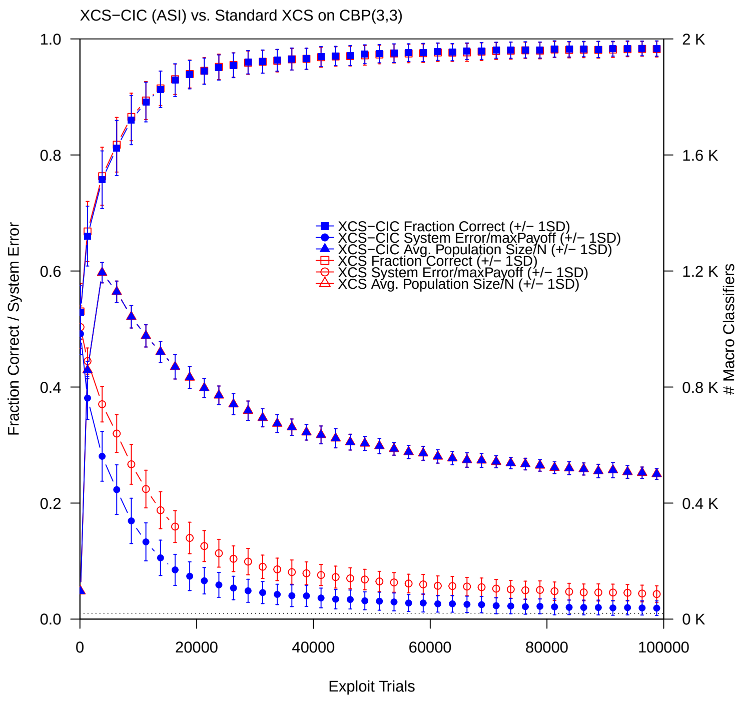


(A) CBP(3,3): IDW

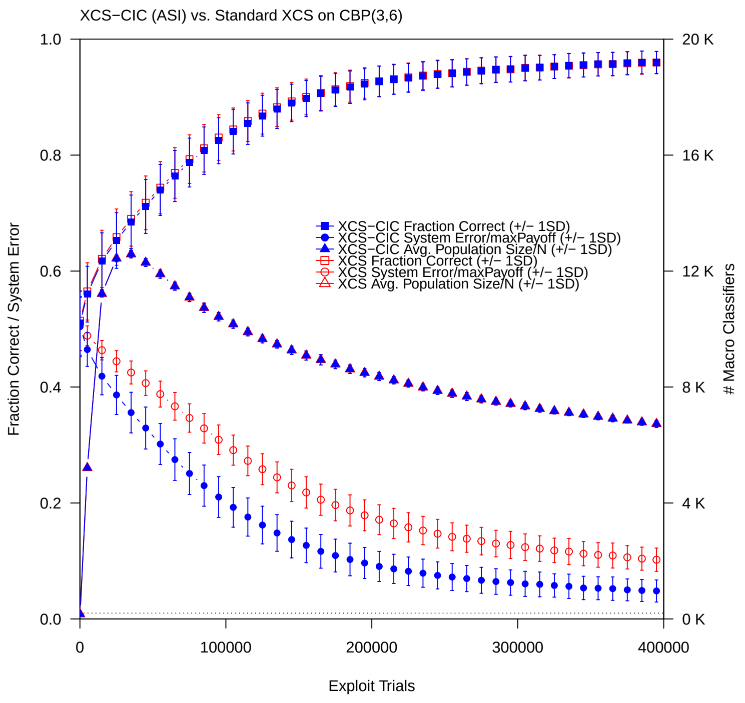


(B) CBP(3,6): IDW

FIGURE 5.7.: Comparison of XCS against XCS-CIC using ASI with global IDW interpolation on CBP(3,3) (top) and CBPs(3,6) (bottom)



(A) CBP(3,3): MSM



(B) CBP(3,6): MSM

FIGURE 5.8.: Comparison of XCS against XCS-CIC using ASI with local MSM interpolation on CBP(3,3) (top) and CBPs(3,6) (bottom)

5.4. Discussion

The preceding sections confirm, that the presented interpolation-assisted action-selection regime, the ASI strategy, achieves an increased learning efficiency. Especially at the beginning of a learning task, where a SLAS usually faces KGs of both types, the application of an action-selection surrogate that allows for transductive inference by means of scattered data interpolation techniques yields superior performance over the conventional two-step induction-deduction mechanism of XCS. By making transductive use of previously made experiences, the decision component of the learning agent can be guided to choose more adequate actions faster. These insights support the initially posed research hypothesis that the integration of interpolation techniques into the algorithmic structure of online learning algorithms decreases their prediction errors in situations where their knowledge bases lack sufficiently reinforced knowledge elements.

Two approaches to implement the ASI strategy have been presented. The first one is based on a loosely coupled extension of XCS by the novel IC. Following this approach, a set of sampling points is maintained within the IC that reflects collected experiences the environment. The management of the stored sampling points constitutes a key task to successfully employ the IC. The evaluation of the IC's current predictive quality in terms of a certain evaluation metric is another important aspect. So far, this approach is thoroughly investigated on a well-adopted and challenging benchmark problem, the so-called *Checkerboard Problem* (CBP). This task is characterized by configurable dimensionality and specificity of the resulting environmental niches (or problem subspaces). CBP is a two-action single-step RL task, which can also be interpreted as a binary classification problem. Thus, a binary reward scheme is applied and discounting of possibly deferred future rewards is not necessary. For such problems, the ASI strategy has been well-defined and solutions for any decision to be felt in order to obtain an appropriately composed IC have been presented. Among the aforementioned decisions, the most important one is clearly an appropriate choice for the adjustment components decision function A. It is responsible for managing the insertion (and removal) of new sampling points in the experience memory SP. Accordingly, the question might appear how other problem types, such as multi-class classification, possibly with layered reward schemes (e.g., assuming different misclassification costs for several classes), or multi-step RL problems bearing the delayed credit assignment challenge, can be handled by the proposed methodology. How should the adjustment component A decide when to add a new sample $s^* = (\sigma_t, a_{exec})$ when there might exist another action a' that would yield a higher reward in the same situation? What if there is only zero reward until a certain goal state is reached, which is the only state rewarded with high payoff?

One possible way has already been investigated in a more detailed fashion – the more tightly integrated XCS-CIC variant for implementing ASI. It constructs the sampling points out of the current set of classifiers $[P]$ and, thus, renders decisions on which adjustment strategy to choose for managing the sampling points in SP obsolete. However, due to the necessity of applying an appropriate quality filter to omit type-2 KGs for the sampling point extraction, the beneficial effects of the ASI method naturally depends more strongly on the learning success of XCS itself. Empirically obtained results also revealed performance gains in comparison to standard XCS. However the overall performance increase is inferior in contrast to the independently hold experience store SP as used in the XCS-IC approach. Nevertheless, the secondly introduced XCS-CIC approach to deploy ASI has further advantages. These will become apparent in the subsequent chapters. XCS-CIC in general allows for the integration of interpolation techniques at other places of XCS's algorithmic structure such as the covering routine and the involved GA.

A further solution for coping with more realistic and harder problem types based on the loosely coupled XCS-IC variant can be achieved as follows. Instead of maintaining only exactly one set of sampling points SP, a distinct set SP_a is to be hold for each possible action $a \in A$. These SP_a sets would then comprise state-reward pairs $s_i^a = (\sigma_t, r_{imm})$ instead of a state-action pairs (σ_t, a_{exec}). Intuitively speaking, the action would be "cut out" of the problem space $S \times A$ by maintaining separate SP's. Thus, in essence, the IC approximates parts of the overall problem space, i.e., $S \rightarrow P$ mappings, by slicing it at the possible actions $a \in A$. In an extreme case of continuous action spaces over the reals, a more sensible way would be to extend the sampling point coordinate $\sigma_t \in S \subseteq \mathbb{R}^n$ by the applied action $a_{exec} \in A \subseteq \mathbb{R}$. This would yield sampling point coordinates of the form $\vec{x}' = (\sigma_1, \sigma_2, \ldots, \sigma_d, a_{exec})$. Accordingly, the SP would then comprise experiences of the form $s_i = (\vec{x}'_i, r_i)$, with r_i denoting the reward received. This would also impact the actual modification of the PA that effectively guides the action-selection (Eq. 5.5). Instead of relying on the discrete values for R_{max} and R_{min} for the decision of creating new sampling points s^* and the resulting calculation of $PA'(a_j)$, it would also be possible to directly interpolate the expected immediate reward for each possible action (class), i.e., $o_{int}^a \approx r(\sigma_t, a), a \in A$.

For sequential RL tasks, which face the credit assignment problem, a further modification regarding the SP store becomes necessary. Instead of considering each sampling point $s_i \in SP$ individually, chains of (s, a, r) triplets need to be established. These chains allow for the maximization of the IC interpolated long-term reward over entire *problem space trajectories* which in turn can be used to modify the PA entries accordingly. This form of interpolative *fore-tracking* constitutes an interesting point for further investigation, but nevertheless goes beyond the scope of this thesis.

The general idea of using a loosely coupled IC instance for supporting the learning mechanism in a SLAS reveals further potentials. The collected experiences, stored in SP, can be used for further offline (or "mental") exploration of the problem space. Especially in sequential RL problems, the collected experiences can be used to implement a form of *experience replay* [Mni+15] in order to boost the convergence of a classifier's learnable parameters $cl.p, cl.\epsilon, cl.F$. Another possibility would be to interpolate the utility function u. Based on this approximation, optimization algorithms can be employed to seek system configurations which yield high utility. The previously introduced MLOC architecture already defines a designated layer for offline learning purposes.

Another important aspect is how an adequate maintenance of the sampling point set SP can be accomplished with regard to different requirements. Depending on the characteristic of the problem domain at hand, it might be better to seek a distribution of the collected experiences that is closer to a uniform distribution (e.g., equidistant samples s_i) A purposeful collection of experiences and the decision which sampling points can be considered obsolete and therefore removed from SP are the tasks of the IC's adjustment component. Considering a highly dynamic NSE which challenges the learning agent with frequently changing input data distributions, a simple time-recency based strategy that keeps the latest experiences and drops the oldest ones might be the right choice. This behavior is straightforwardly achieved by realizing SP as a *First-In-First-Out* (FIFO) queue. For problems that are characterized by seasonal and, thus, recurrent situation occurrences, the former strategy can result in a loss of useful experiences. Therefore, a replacement strategy which actively steers the sampling point collection toward a uniform distribution within SP should be preferred. For obtaining such an SP update behavior, methods based on the notion of *entropy* and according estimators might be utilized in the future.

One particular limitation of the current approach is that it is so far designed for application in deterministic MDPs only. This means that non-stochastic reward functions r are assumed. In the case that for a state-action pair (σ, a) it is not guaranteed that always the same reward is payed out, this would also needed to be considered in the collected experiences, at least for the case that $s_i^a = (\sigma, r_{imm})$ tuples are remembered. For enumerable (discrete) state spaces S this can be achieved by simply calculating the mean value of all rewards seen so far for a particular state-action pair. This would essentially "average out" the noisy reward signals. This can also be achieved by an estimation in a time-recency weighted manner using e.g., the Widrow-Hoff rule. This would lead to recently arrived measures gaining higher influence than older ones. However, for continuous state spaces $S \subseteq \mathbb{R}^n$ this turns out to be more complicated. The currently applied solution is to define a very small similarity threshold ϵ such that whenever the Euclidean distance between two sampling point coordinates $\vec{x}_1$ and $\vec{x}_2$ is smaller than ϵ, the vector defining the center on a straight line between $\vec{x}_1, \vec{x}_2$ serves as new sampling point coordinate and the function value is set to the average.

Finally, it should be explicitly noted that ASI strategy has no explicit effect on the evolution of XCS's rule base. Rather, it supports the learning agent to choose appropriate actions in situations where the knowledge base has not converged to an accurate approximation of the problem space, yet. This makes a SLAS more robust whenever a context change appears gradually or else abruptly. Such circumstances confront the system with perpetually novel state descriptions σ_t. Even if the IC might not have exact experiences for the so far unknown incoming situations σ_t in its SP, due to its transductive learning manner it is still expected to provide reliable "hints" to alleviate a potential utility degradation of the system.

5.5. Related Work

5.5.1. Memory in an LCS

With the introduction of a novel IC in Chapter 4, a form of memory is added to XCS. Primary endeavors in equipping LCS with temporary memory in order to better cope with *Non-Markov* environments can be found in [CR94]. Lanzi introduced further memory-based mechanisms to XCS in [Lan98a; Lan98b] and together with Wilson demonstrated their superiority on aliased learning environments in [LW00]. A further XCS-based system called TP-XCS, which is also capable of learning *partially observable markov decision processes* (POMDPs), is presented by Pickering and Kovacs in [PK15]. The primary works mainly approach the problem of *perceptual aliasing* via adding a register-bit to the systems in order to disambiguate aliased states. In contrast, the latter TP-XCS maintains a window of the last n situations perceived and extends the input perception as well as the condition of each classifier accordingly. This inhibits scalability when large memories would be needed. All aforementioned approaches are only evaluated on binary problem domains which constitutes a first major distinguishing aspect to the work presented in this thesis. Additionally, in contrast to the kind of *experience memory SP* as introduced in this chapter, none of the aforementioned approaches make use of their memories in terms of interpolating between previously encountered states in order to transductively infer additional knowledge for the current situation σ_t.

5.5.2. Interpolation and CBR in RL

Emigh et al. propose a methodology for utilizing NeNe interpolation in Q-learning for playing an arcade game called *Frogger* in [Emi+16]. Their approach is to determine the nearest neighbors within the Q-table and then use the NeNe-interpolated Q-value for (1) the actual action selection step, and, (2) for the update in cases where no entry exists so far. Additionally, an approach to metric learning is presented, that constitutes an interesting means to control the emphasis of particular state features when nearest neighbors are to be determined. Promising results are reported in terms of speeding up convergence when their presented extensions to Q-learning are applied. In contrast to the ASI scheme as introduced in this chapter, no trade-off between inductively acquired Q estimations and transductively determined values is considered. Furthermore, in the proposed NeNe-based Q-learning approach interpolation only takes place when no entry in the current Q-table is present.

Another branch that can be found in the context of RL is the combination of *case-based reasoning* (CBR) with RL methods. Gabel and Riedmiller [GR05] use function approximation based on CBR to increase learning efficiency for state value approximation in RL. They apply k-nearest neighbor regression for the use in high-dimensional continuous state spaces. The authors describe promising data management techniques to maintain the case memory. These techniques might also be adopted for utilization within the adjustment component which is responsible to update SP.

Considering the second approach as presented in this chapter, XCS-CIC, where the set of sampling points SP is replaced by the current rule base $[P]$ of XCS, further similarities to the work of Glatt et al. reported in [GSC17] become apparent. In their paper, the *case-based policy inference* (CBPI) is presented. CBPI aims at transferring knowledge from previously learned tasks (in terms of policies) to related and similar tasks. Therefore, a case base comprising previously solved RL tasks with their corresponding policies is maintained. As can be seen, the authors of this work introduce CBR in the context of RL in a different way. More precisely, entire problems defined by an MDPs constitute cases with their solutions given by the policies solving these MDPs. CBPI then works by selecting the most similar cases from the case base and by evaluating the importance for each of the selected ones. This evaluation is done by conducting a limited number of training episodes. Based on these importance values, the selected cases are *blended* to yield a policy for the new target task. Furthermore, a trade-off mechanism is proposed for successively letting the actually learned policy take over the control. Even if not directly related to the approaches as developed in the previous sections, interpreting the population of classifiers as case base, and the interpolation between the filtered classifiers as sort of blending, at least rudimentary similes can be drawn.

5.5.3. Heuristically Accelerated RL

Bianchi et al. in [BRC04; BRC08] present the concept of *heuristically accelerated RL* (HARL). The objective of their HARL approach shares clear similarities with the ASI strategy. A *heuristic* is to be formed, either by incorporating domain knowledge, or in an online fashion during early learning stages, in order to *guide* the action-selection mechanism of RL agents. The authors apply their concept by proposing HAQL, an heuristically accelerated variant of Q-learning, which is evaluated on robot navigation scenarios. Two main phases are mentioned: (1) *Structure extraction* which attempts to build a *map sketch* of the environment. (2) *Heuristic backpropagation* which uses that map to compose a heuristic. Thus, the approach essentially approximates a model of the underlying problem and subsequently applies planning to figure out adequate action choices. This is reminiscent of model-learning

RL techniques and contrasts with the transductive, i.e., no model building, approach presented in this chapter. However, in the context of LCS-based RL, ASI can be understood as a HARL approach extracting a heuristic online during the learning progress by means of interpolating between a set of previous experiences stored in SP.

5.5.4. Experience Replay in Deep RL

As already discussed in the previous section, the IC comprises all components that are required to perform experience replay [Mni+15]. Experience replay is used to prevent overfitting in DNNs which are used to approximate the Q-function in DQN. Therefore, previous experiences in the form (s, a, r, s) are stored in a so-called *replay memory*. For each update of the Q-network, a small minibatch is sampled uniformly at random from that memory. This minibatch is used, together with the just experienced transition, to conduct backpropagation. The means of how the stored experiences are used differs clearly from the ASI techniques as introduced here. However, in future work it is planned to combine these two techniques. More sophisticated approaches that take the idea of experience replay even further can be found in [Han+18] and [Sch+15], for instance.

5.5.5. Memory-based and Model-learning RL approaches

In the seminal book of Sutton and Barto [SB98], a rather short section is dedicated to *Memory-based RL approaches*. The potentials of memory-based approaches, such as the attenuating effect regarding the curse of dimensionality by using directly stored experiences and making only use of instances in the proximity of a current state or state-action pair in contrast to fitting a global parametric model are indicated. They further provide a short list of relevant work in this domain, which mainly narrows to *Memory-based Dynamic Programming* (MBDP) by Peng [Pen95] and the work of Schaal and Atkeson et al. on *Locally Weighted Regression* [AMS97; SA94]. Where the former is an early approach applied to dynamic programming techniques and not to temporal difference learning, the latter approach indeed uses local instances. However, it does this for learning a local but parametric regression model which again constitutes an approximation in the end. Thus, it can be concluded that the attention on the explicit use of interpolation techniques remains sparse in the literature on RL.

Beside, model-learning and value function approximating techniques from the domain of RL can be regarded as related farther away. Examples are Sutton's Dyna-architecture [Sut91] or Actor-Critic models [SB98]. These methods have in common that they build explicit models of at least parts of the underlying problem by approximation. These models are then combined with RL methods. Even if these approaches often use DNNs as approximators and, thus, do not rely on interpolation directly, the approach of having a sort of surrogate model for improving the performance of learning algorithms at different aspects of the learning task indeed is reminiscent of the loosely coupled structural extension of XCS by the IC.

5.6. Chapter Summary

In this chapter, an interpolation integration strategy has been developed that supports the action-selection step of XCS during exploitation. The IC was used as a transductive surrogate model to support the actual selection by modifying the prediction array entries. This modification was realized by transductively calculating accumulated interpolation weights from collected experiences. Both architectural variants, i.e., XCS-IC and XCS-CIC, have been considered for implementing this so-called ASI strategy. An empirical study was conducted on two instances of a challenging benchmark problem known as the CBP. Results indicated superior performance for both variants regarding the system error for expected reward prediction. However, the fraction of correct classifications (average accuracy) was only found to be superior for the loosely-coupled XCS-IC approach. More research has to be done for the XCS-CIC variant in this regard, since slightly decreased average accuracy values have been observed so far. A thorough discussion on possible improvements followed the evaluation section. As before, the chapter was closed with the appreciation of related work comprising memory-based approaches and model-learning in LCS and RL in general, interpolation-based concepts from the domain of RL, heuristically accelerated RL, as well as experience replay in deep RL. While this chapter mainly focused on making transductive use of so far collected experiences given by situation-action pairs, the next chapter pursues a different objective. XCS's reactive fallback mechanism to handle type-1 KGs is enhanced by the concept of transductive knowledge inference in order to improve the initialization of a novel classifier's learning parameters.

Chapter 6.

Interpolation-Based Covering

The covering routine is a fallback solution for situations where the rule base of XCS does not contain any classifier that matches the current state vector σ_t. Thus, XCS per default provides a reactive solution for type-1 KGs. Novel knowledge elements, i.e., classifiers, are created almost, however, not entirely at random. The means how a newly covered classifier cl_{cov} is initialized can be controlled via several hyperparameters which determine the initial generality of the condition as well as the initial values for the predicted payoff $cl.p$, the error estimate $cl.\epsilon$, as well as the fitness $cl.F$. Except for the first parameter, most often denoted by r_0 or s_0 in the literature, the configurations for the initial values of the remaining classifier parameters are changed seldom if ever. Thus, the initialization just covered rules' learning parameters is mainly arbitrary and "blind". Initializing a novel rule in that manner bears the potential for type-2 KGs being created. In order to at least attenuate this circumstance by more radically updating the parameter estimates during the very first reinforcement updates, the MAM technique has been introduced to XCS in [Wil95] (cf. Sect.2.4). It will be briefly revisited below.

The interpolation-based covering operator developed in this chapter constitutes another means of integrating interpolation with XCS' algorithmic structure. The so far rather arbitrarily initialized covered classifiers cl_{cov}, will undergo a more informed initialization procedure which takes the already learned parameters of adjacent classifiers from neighboring problem space niches into consideration. This second strategy for interpolation integration is therefore named *Covering Intialization Integration* (CII).

The hypothesis behind this methodology, i.e., of incorporating values from neighboring but non-matching classifiers for novel classifier initialization purposes, is that high-quality classifiers from adjacent niches presumably possess valuable insights regarding the expectation values of the parameters to be estimated. For instance, the achievable error level or the amount of fitness, which is shared among the classifiers defining the niches, might also – at least approximately – apply to the currently considered but yet uncovered niche. This assumption however implies a certain degree of smoothness of the underlying problem structure, i.e., similar actions in similar situations lead to similar rewards, errors and consequently fitness estimates. Basing the design of learning algorithms on this assumption is also known as *nearest neighbor inductive bias*). However, for the abstracted general learning problem as defined in Chapter 3, i.e., the online approximation of continuous utility functions over a problem space $\tilde{f} \approx f : PS \to U$, this assumption turns out to be indeed reasonable. Even if the resulting utility surface is not continuous is a mathematical sense, an LCS is still very capable of learning such problems. This can be expected whenever discrete action spaces A with nominal, i.e., non-ordered, possibilities factor into the problem space PS, for instance. As a consequence, the utility or reward might change quite abruptly for different actions. However, this issue can be circumvented by applying an appropriate classifier filter that restricts the considered classifiers out of which the sampling points are constructed to those that advocate only identical actions.

Accordingly, in contrast to begin the classifier parameter estimation completely from scratch for each classifier newly created by covering, the *Covering Intialization Integration* (CII) technique provides an informed means of classifier construction which relies more strongly on the immediate classifier neighborhood.

Additionally, and in line with the overarching research hypothesis1.1 this thesis is posing, the interpolation-based covering technique is supposed to increase the learning efficiency, especially at the early phases of learning, by counteracting KGs of type 1 in a more sophisticated manner. The incorporation of learned knowledge from neighboring problem space niches can be understood as a form of *transfer learning* in terms of effectively transferring learned parameters from one or more particular problem space niches to another, so far unexplored one in the same problem space.

The next paragraphs briefly recapitulate on the basic covering scheme that is applied in XCS. Afterwards, the novel CII strategy is introduced. The benefits of this interpolation-based covering strategy is then evaluated on the task of continuous function approximation applied to three different functions of varying complexity in terms of dimensionality and curvature from the numerical optimization domain. Finally, the results and further insights are discussed and an overview of related work is given.

6.1. Basic Covering

As previously explained in Section 2.4, the covering process occurs when any of the following cases is true:

1. Whenever $[P]$ does not contain any matching classifier for the current situation $\sigma_t = \vec{x}_t$ and thus $[M]$ is empty.
2. Whenever the number of distinct actions represented by classifiers in $[M]$ is less than θ_{mna}.

3. Whenever the average quality of $[M]$ is poor, i.e., the average fitness $\bar{F}_{[M]}$ of $cl \in [M]$ is less than the mean fitness of the population $\bar{F}_{[P]}$.

In consequence, XCS generates at least one novel classifier cl_{cov} to cover the yet unexplored environmental niche. The condition of the novel classifiers $cl_{cov}.C$ is initialized with $\vec{x_t}$ and for each dimension $i = 1 \dots d$ a so-called interval predicate (l_i, u_i) is calculated as follows: $l_i = \max\{l_i^*, x_i - U[0, r_0]\}$ and $u_i = \min\{u_i^*, x_i + U[0, r_0]\}$. $U_{[0,r_0)}$ returns a uniformly distributed random number between 0 and r_0 excluded, where r_0 is a predefined default spread parameter. l_i^* and u_i^* denote the minimal and maximal bounds of the problem space for the i-th dimension, respectively. For a more detailed description of covering please refer to Section 2.4. Classifier parameters for $cl.p, cl.\epsilon, cl.F, cl.exp$ are initialized with predefined values (hyperparameters) $p_{ini}, \epsilon_{ini}, F_{ini}$ and 0, respectively. In the case of XCSF, the weight vector $\vec{w}$ for the polynomial prediction approximation is initialized with $w_i = U[-1, 1], i = (1 \dots n)$. If known, the offset weight w_0 is initialized randomly in between the value range of the function f to be approximated. Otherwise, it is also initialized randomly with $w_0 = U[-1, 1]$. In classification and RL tasks, the action is selected by chance among those which are not already present in $[M]$. Other works on XCS assume that the actual target value, e.g., the correct class a or the function value $y = f(\vec{x}_t)$, is immediately known when the situation vector $\vec{x}_t$ arrives, i.e., each trial XCS receives a complete sample (cf. e.g., [Sta14]). Accordingly, in the case of XCSF e.g., the offset weight can be set to the actual output value, i.e. $w_0 = f(\vec{x}_t)$, and the initial prediction error drops. The same holds for the correct action during the covering phase in standard XCS in RL or classification tasks. This is a plausible assumption, when XCS is asked accomplish a typical supervised learning task based on a data set available a priori. In contrast, this work focuses on the challenge of learning some unknown target concepts or functions at the runtime of SLAS. Thus, at time t, XCS gets not presented a complete sample, but only the situation vector $\vec{x}_t$. The correctness of the prediction which is deduced can only be determined on the basis of the retrieved immediate reward r_{imm}. Nonetheless, the interpolation-based classifier initialization techniques introduced in the following sections as well as in Chapter 8 are straightforwardly applicable to LCS variants utilized to solve supervised learning problems.

6.2. Interpolation-Based Covering Initialization Strategy

As already discussed, the problem with predefined initial values is that they are often set arbitrarily. More precisely, the initialization depends on particular hyperparameters that are rarely changed if ever. Following the usually applied MAM technique (cf. Ch. 2.4), some of the arbitrarily set initial values are quickly eliminated by applying a simple average over the first $1/\beta$ updates. This procedure is applied for the initial prediction value $cl.p$, as well as for the absolute error estimate $cl.\epsilon$. However, for the fitness estimate, this procedure is usually not applied. Also for the case for XCSF using computed predictions, where the scalar prediction parameter $cl.p$ is replaced by a vector of weights $cl.\vec{w}$ which constitutes the coefficients for a polynomial approximation (cf. [Wil02; BLW08]). Furthermore, the influence of MAM and its expected effect that the parameter estimates would converge more quickly to their expected values, entirely depends on the choice of the learning rate β. Here, lower learning rates lead to more frequent applications of the running average calculation instead of the gradient-approximating Widrow-Hoff update.

Approaching the expected error estimates $cl.\epsilon$ quickly is specifically crucial to supply an adequate fitness signal to the evolutionary component of XCS. Lower absolute prediction errors lead to exponentially higher absolute accuracy values (cf. Sect. 2.4 for details regarding the fitness calculation and update). The absolute accuracy values determine the share of fitness that is attributed to a particular rule within its environmental niche. Thus, the faster the prediction error drops, the quicker a higher fitness can be assumed. This in turn leads to a more distinct fitness signal on which the applied GA bases its selection, resulting in a higher reproductive opportunity and, thus, niche exploration in the end. When it is possible to speed up the time needed to figure out the correct estimates, strong classifiers will be promoted and poor classifiers will be 'sorted out' faster. The CII approach introduced next, therefore, takes the estimates of classifiers from adjacent niches into account when a new classifier in a so far unexplored niche is to be created by means of the covering operator. Therefore, it is explicitly made use of already existing knowledge elements in the rule base in order to estimate appropriate initial values for a newly created one. Following the previously introduced notion of transductive knowledge inference, the already learned parameters (e.g., $cl.p, cl.\epsilon, cl.F$) of classifiers in the direct proximity of the currently unknown situation are used to transductively infer initial values for the new classifier to be created via covering. Figure 6.1 is intended to convey the intuition of this approach. As for XCS-CIC, the sampling points are constructed from available classifiers $cl_i \in [P]$. Again, the center points of the conditions serve as sampling point coordinates $\vec{x}_i$. In contrast to ASI, however, the sampling points' function values to be interpolated y_i are different as will be explained next. It is explicitly noted here, that the shape of the condition is not interpolated so far. However, first thoughts in this direction have already been spent and will be investigated for deeply in the future.

The influence of classifiers that lie farther away from the newly created one decreases depending on the distance metric utilized for the particular interpolation technique. Consider a large problem space with a high dimensionality and non-uniformly distributed instances from the problem input domain at hand. At the beginning, the covering mechanism acts frequently but a complete coverage of the entire problem space is unlikely due to the non-uniform sampling and the high-dimensional space. When a situation occurs for the first time, the population contains no classifier which

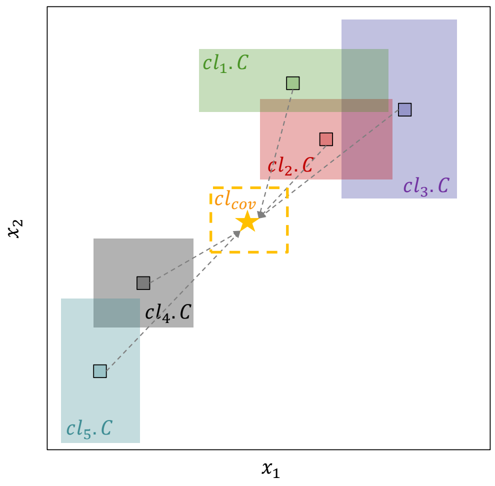


FIGURE 6.1.: Intuition of the interpolation-based covering operator using CII

yields adequate predictions about this environmental niche. However, nearby classifiers from surrounding niches that are sampled more frequently and, thus, can be expected to predict with higher accuracy might be present. With this transductive incorporation of such available knowledge, a more appropriate initialization of newly generated classifiers is pursued resulting in decreased initial prediction errors and, thus, higher robustness against unforeseen situations.

Technically, the CII lessens the influence of XCS' hyperparameters for classifier initialization. Instead of assigning new classifiers' parameters with predefined initial values such as p_{ini}, ϵ_{ini} or F_{ini}, using the CII approach the IC as introduced in Chapter 4 is employed to interpolate these values. Essentially, any classifier parameter which is represented by a numerical value can be transductively initialized by means of interpolation. Preliminary experiments have revealed that specifically the interpolation of $cl.p$ (in case of XCSF $cl.\vec{w}$), $cl.F$ and $cl.exp$ leads to increased learning efficiency in terms of faster prediction error drops at early stages of learning with empty rule bases, i.e., high presence of type-1 KGs. The CII strategy can be understood as an explicit countermeasure against type-1 KGs, since it is only activated during the covering routine. Depending on the used triggers for activating covering itself, it is also imaginable that CII supports effective reduction of negative impacts due to type-2 KGs (see the abovementioned covering activation case 3). However, since traditionally covering in XCS is only activated when the match set $[M]$ is empty, an experimental evaluation with alternative covering activation schemes is not reflected in the experiments reported in this thesis.

Equation 6.1 exemplarily formalizes the interpolation procedure based on local Shepard interpolation (MSM) for the more complex case of XCSF which replaces scalar reward predictions $cl.p$ by computed predictions where the learnable classifier parameter is represented by a weight (or coefficient) vector $cl.\vec{w}$. A classifier's parameters subject to interpolation in the following study can be represented by a function value vector as follows:

$$\vec{f}_{cov} = \begin{pmatrix} cl_{cov}.\vec{w}^{(0)} \\ cl_{cov}.\vec{w}^{(1)} \\ \cdots \\ cl_{cov}.\vec{w}^{(d)} \\ cl_{cov}.F \\ cl_{cov}.exp \end{pmatrix}$$

The function value vectors $\vec{f_i}$ (defined analogously to $\vec{f}_{cov}$) of the considered classifiers cl_i out of which the sampling points are constructed are multiplied component-wise with a corresponding interpolation weight W_i (cf. Ch. 2.3). Further, $SP' \subseteq SP$ defines the filtered set of sampling points constructed from $[P]$. It was decided to not apply a quality filter, since especially during early stages of learning, it was expected that any available knowledge elements positively contribute to classifier initialization. However, it is explicitly noted that the application of an appropriate filter, e.g., as proposed in the previous chapter might be meaningful in later phases of runtime learning. For the CII strategy, the entire population $[P]$ instead of the current input space niche $[M]$ serves as candidate pool of classifiers. Since it is focused on a function approximation (regression) task here, a further filtering for classifiers with respect to the actions they advocate is not necessary. In the case of typical RL or online supervised learning tasks, such as the CBP classification problem, the sampling points need to constructed out of a set of classifiers with equal actions. Empirical results that reflect this aspect in the experimental setup have also been conducted in [Ste+16b; Ste+17a].

Equation 6.1 shows the calculation of the function value vector $\vec{f}_{cov}$ for each classifier cl_{cov} that is created throughout one activation of the covering routine. The shape of the condition $cl_{cov}.C$ as well as the action $cl_{cov}.a$ are determined probabilistically as usual.

$$\vec{f}_{cov} = \frac{\sum_{i=1}^{|SP'|} W_i \cdot \vec{f_i}}{\sum_{i=1}^{|SP'|} W_i}, \text{ with } cl_i \in SP' \tag{6.1}$$

The CII strategy can be regarded as sort of knowledge transfer and bootstrapping for newly created classifiers in a sense that already available knowledge (learned parameters $cl.\vec{w}, cl.F, cl.exp$) of adjacent classifiers is transductively inferred.

In case of an initially empty population where no sampling points an be constructed out of neighboring rules, standard covering serves as fallback solution.

6.3. Evaluation

For the sake of validating the benefits of interpolation-based covering, in this chapter three benchmark scenarios from the domain of function approximation are investigated. Thus, the XCSF derivative is employed here. The learning task is to reconstruct, i.e., approximate, a particular function surface online instance by instance – or in terms of SLAS, at runtime. It is explicitly noted that the learning i.e., an approximate reconstruction, of the underlying functions is the objective here and not the search for a global minimum. Further results have been obtained in a study that investigates the potential of CII and further interpolation strategies within XCSR applied to the previously introduced CBP task. For the sake of brevity, the results are not included in this chapter, but can be found in Appendix C. For details please confer [Ste+16b; Ste+17a].

6.3.1. Function Approximation

As discussed earlier in Chapter 3, the approximation of unknown functions during a system's runtime constitutes a generic learning task. Possible functions which are sensible to be learned at runtime are for instance: 1) the *overall* or *partial utility function(s)* which determine the appropriateness of certain systems states in terms of fulfilling the objectives provided by an external authority (e.g., for triggering self-adaptation processes via the control mechanism to achieve robustness and flexibility). 2) The *value function* of an employed RL agent which is asked to solve a sequential control problem such as successively adapt a control variable of interest (e.g., for using model-free policy gradient approaches). 3) Any other unknown functional relationships where only input-output pairs can be observed and on which a gradient-based optimization technique should be applied to figure out the most appropriate inputs with as few steps as possible (model-based learning or planning).

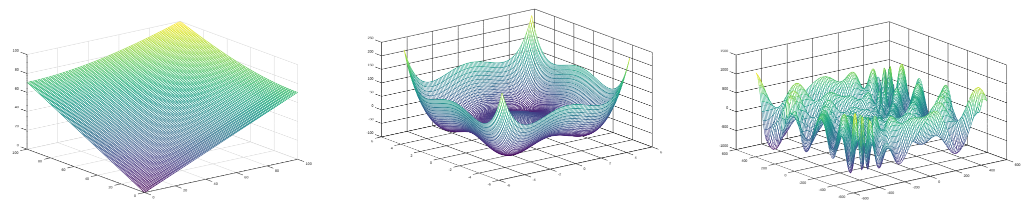
The function surfaces of the aforementioned examples are usually not known a priori and, thus, constitute black-box problems. To reflect this aspect, in the following study, functions of different shapes are investigated.

The following paragraphs present the results from the experimental evaluation of the proposed techniques. Experiments on three test functions with different degrees of complexity in terms of dimensionality and curvature have been conducted. Figure 6.2 depicts the 3-dimensional surface plots of the considered functions for 2-dimensional input domains.

6.3.2. Experimental Setup

If not stated differently, for the empirical study presented next, XCSF has been configured following the suggestions by Lanzi et al. [Lan+05a]: $\alpha = 0.1$, $\beta = 0.2$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 50$, $\epsilon_0 = 0.01$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\chi = 0.8$, $\mu = 0.04$, $\epsilon_I = 0.0$, $F_I = 0.01$, $r_0 = 0.1$, $m_0 = 0.2$, $fitnessreduction = 0.1$, $\delta_{rls} = 1000$, $\lambda = 1$. Linear approximation in conjunction with the RLS method [Lan+05a; BLW08] has been used to incrementally adjust the prediction coefficients stored in the classifiers as weight vectors $cl.\vec{w}$ – effectively replacing the prediction scalar $cl.p$ by computed predictions. Furthermore, *tournament selection* as introduced by Butz et al. in [BSG03] is utilized for parental selection within the GA. Based on the insights regarding the performance of different interpolation techniques obtained in the previous chapter, it has been decided to set the focus on the use of local interpolation via the MSM technique. Preliminary investigations in the scope of a master's thesis [Rau16] supervised by the author corroborate this decision. The number of considered sampling points for local interpolation was set to $N_w = 19$, following the suggestion of Thacker et al. [Tha+09]. This hyperparameter is kept constant throughout all the experiments on functions of different complexity in order to gain an intuition about the importance regarding the choice of N_w. It turns out that even with such a small number of sampling points, significant improvements can be achieved. It is clearly noted that this decision is not meant to replace a sensitivity study of this hyperparameter. However, the conduction of an elaborate and computational highly expensive sensitivity analysis is beyond the scope of this thesis. It is expected, though, that a higher value for N_w will result in even larger improvements regarding the initial prediction errors. This hypothesis could also be corroborated with preliminary results conducted in the scope of student's theses supervised by the author, cf. e.g., [Rau16; Mei17; Wag17].

All reported results are averages over 30 i.i.d. runs with different random seeds. Each repetition is performed for $200k$ learning trials. At each trial, XCSF is presented an input vector $\vec{x}_t$ which is uniformly sampled from the domains of the benchmark functions. To assess the obtained results on statistical significance, paired t-tests



(A) f_1: RMS-like function [Wil02] (B) f_2: Styblinski-Tang function [ST90; JY13] (C) f_3: Eggholder function [JY13]

FIGURE 6.2.: Surface plots of the considered test functions to be approximated for dimensionality $n = 2$

(see Table 6.1) have been conducted. For this reason, the results of all 30 runs have been confirmed to follow a normal distribution beforehand using Shaprio-Wilk test as well as by visual inspection of QQ-plots. The plotted results depicted in the Figures 6.3 to 6.6 compare the normalized system error (i.e., normalized MAE) and the number of evolved macroclassifiers of standard XCSF with the interpolation-assisted XCSF-CIC using CII. Points show the averages over the last 100 trials and the error bars indicate the standard deviation ($\pm$1SD). The y-axes of the plots, are log-scaled in order to better illustrate the interesting ranges of the system error. The number of macro-classifiers is also depicted on a log-scale (right-hand y-axis of each plot). This representation is adopted from Stalph and Butz (cf. e.g. [SB12; SB10b]). Since covering – and therefore CII – almost exclusively acts at the beginning of a learning task, the plots concentrate on the initial learning phases up to $10k$ trials to allow for a better illustration of the potential benefits. However, for the results reported in Table 6.1 as well as for the conducted t-tests, still the entirety of learning steps ($200k$) is actually used.

6.3.3. Results

The following paragraphs describe the unique characteristics of the particular functions used to validate the CII strategy as well as the results obtained when testing both standard XCSF and XCSF-CIC on them. Table 6.1 summarizes the results obtained from the conducted empirical studies. * (**) indicates statistically (highly) significant differences regarding the reported metrics compared to standard XCSF, i.e., that for the p-values of paired t-tests holds $p < \alpha = 0.05$ (0.01). Bold values indicate significant improvements compared to standard XCSF and the arrows, $\uparrow$ or $\downarrow$, indicate increased or decreased values, respectively.

Results for Simple RMS-like Function f_1

In [Wil02], Wilson proposed a rather simple, but easily scalable function to test XCSF on higher-dimensional problems. Figure 6.2a depicts the surface of this function for $n = 2$ dimensions. Wilson describes it as a 'RMS-like' function resulting in a flat but rising hyper-plane. It is defined as follows:

$$f_1(\vec{x}) = [(x_1^2 + \cdots + x_n^2)/n]^{1/2},\ 0 \leq x_i < 100 \tag{6.2}$$

Wilson has shown in his article [Wil02] that XCSF is very capable of learning this function even in a decent number of input dimensions ($n = 6$). Accordingly, the parameterization of XCSF is based on Wilson's previous work for this first experiment which only slightly differs from the abovementioned default settings for the following hyperparameters: $N = 3200$, $\beta = 0.1$, $\theta_{sub} = 200$. However, instead of the

TABLE 6.1.: Summary of results for XCSF-CIC using the CII strategy on the selected test functions f_{1-3}

f_1 **RMS-like 6D** (target error $\epsilon_0 = 1$)	*System Error* mean±1SD	*Macro Classifiers* mean±1SD
XCSF-CIC w/ **CII**	**.5394** $\pm$ .005 $\downarrow$**	1680.62 $\pm$ 12.80 $\uparrow$
XCSF	.6690 $\pm$ .013	1676.59 $\pm$ 16.95
f_2 **Styblinski-Tang 3D**	*System Error*	*Macro Classifiers*
XCSF-CIC w/ **CII**	**.0227** $\pm$ .0002 $\downarrow$**	3280.61 $\pm$ 17.09 $\uparrow$**
XCSF	.0260 $\pm$ .0003	3250.96 $\pm$ 16.75
f_2 **Styblinski-Tang 6D**	*System Error*	*Macro Classifiers*
XCSF-CIC w/ **CII**	**.0689** $\pm$ .0006 $\downarrow$**	18288.88 $\pm$ 79.30 $\uparrow$**
XCSF	.0699 $\pm$ .0005	18201.20 $\pm$ 125.49
f_3 **Eggholder 2D**	*System Error*	*Macro Classifiers*
XCSF-CIC w/ **CII**	**.0520** $\pm$ .0006 $\downarrow$**	2954.93 $\pm$ 12.57 $\downarrow$
XCSF	.0553 $\pm$ .0006	2955.01 $\pm$ 9.37

modified Widrow-Hoff update rule as proposed in his article, here the more powerful RLS update rule is used which generally has been found to perform significantly better (cf. e.g., [Lan+06; BLW08]) in terms of increased system error decrease and convergence.

Table 6.1 summarizes the results obtained for f_1.[1] Figure 6.3 furthermore shows the effect of CII which can be clearly observed at the beginning of the learning phase where per definition KGs of type 1 are predominantly present what causes covering to act frequently. With CII the system error drops significantly faster during the first $10k$ trials. The population size develops similar but marginally increases during the initial $2k$ steps in comparison to standard XCSF.

Although the $n = 6$ dimensional RMS-like function f_1 does not constitute a challenge for standard XCSF, the utilization of the CII strategy yields beneficial effects in terms of a significantly reduced system error (19.37% over the entire experiment) which is additionally reached more quickly and within a tighter standard deviation. The average number of classifiers over the entire $200k$ learning steps, however, marginally increases in terms of absolute measures (0.24%). The difference has not been found to be statistically significant and thus can be considered as caused by chance.

[1] Please note that the value range of the RMS-like function is $[0, 100)$. Thus, the targeted error level of 1% corresponds to an absolute error of 1 and the values reported in Table 6.1 are above the results of the other test functions.

Overall, it can be noticed that for a rather low complex function with moderate dimensionality such as the RMS-like f_1, where neither high curvature nor obliqueness characterizes the function surface, the transductive inference of a covered classifier's parameters from neighboring rules yields a faster decrease regarding the initial prediction errors while the average population size nearly stays the same.

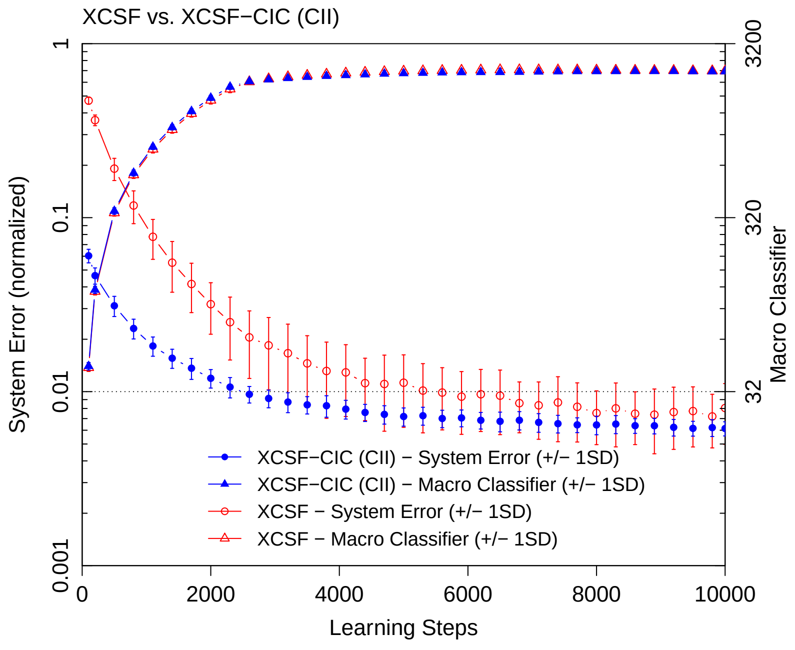


FIGURE 6.3.: Learning curves of XCSF-CIC with CII and XCSF on 6D RMS (f_1).

Results for Curved Styblinski-Tang Function f_2

The second test case, the *Styblinski-Tang* function[2] [ST90], is a common benchmark function from the domain of numerical optimization [JY13]. It is characterized to be non-convex, smooth and multimodal, i.e., by having multiple local optima. Thus, it shows a moderate degree of curvature. Figure 6.2b shows a surface plot for $n = 2$ input dimensions. For the general case of n dimensions, i.e., $\vec{x} \in [-5, 5]^n$, the Styblinksi-Tang function is defined as follows:

$$f_2(\vec{x}) = \frac{\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i}{2}, \quad -5 \leq x_i \leq 5 \tag{6.3}$$

[2]W.l.o.g., both the domain and co-domain are normalized to the range $[0, 1]$.

n=3-dimensional case The approximation capabilities of XCSF is evaluated for $n = 3$ and $n = 6$ dimensions. For the 3-dimensional case, XCSF parameters were set as described before in the experimental setup following Lanzi's suggestions [Lan+05a] except for the maximum population size which is $N = 6400$.

As for the previous experiment, the CII-extended XCSF outperforms the conventional XCSF in terms of a distinctly steeper descent of the system prediction error at the beginning of the learning phase during the first $10k$ learning steps (see Fig. 6.4) as well as a decreased overall system error over the entire $200k$ learning steps by -12.69%. Similar as for the RMS function (f_1), a slight but statistically significant increase in the average magnitude of $[P]$ ($+0.91\%$) throughout the entire $200k$ step experiment (cf. Tab. 6.1) can be observed. Overall, XCSF is not able to reach the target system error of $\epsilon_0 = 1\%$ which corroborates the Styblinski-Tang function's complexity even for the rather low 3-dimensional case. However, the interpolation-based covering operator yields beneficial effects which are mainly attributed to the interpolation of the coefficient vector components $w_i \in \vec{w}$ used for computing the classifiers' predictions. Instead of initializing the coefficients with random numbers from a predefined range, the already gained experiences of surrounding classifiers in close niches are taken into account and allow for a more informed initialization. On that basis, the stochastic gradient-descent RLS update is expected to start from a more appropriate initial position within the search space spanned by the coefficient vector $cl.\vec{w}.'$

n=6-dimensional case Next, the dimensionality has been doubled to $n = 6$. XCSF is configured as for the previous experiment, except for the maximum population size $N = 25600$ and the expected initial condition spread $r_0 = 0.5$. This has been decided in order to allow XCSF to explore the problem space effectively and prevent it from being trapped in a covering-deletion cycle due to initializing the classifier population process from the over-specific side during covering (cf. [Sta+12a]). However, a higher value for r_0 usually results in a reduced number of covering operations since less initial classifiers are sufficient to at least cover the entire input space – effectively reducing KGs of type 1 but in the same breath creating those of the second type.

As summarized in Table 6.1, XCSF struggles to reach the desired target error level $\epsilon_0 = 0.01$ and to find a suitable approximation for the test function f_2 with a 6-dimensional input domain. This limited approximation capability is mainly attributed to the linear model utilized for the prediction computation. A way to improve on this will be demonstrated in Chapter 9, where the 6-dimensional Styblinski-Tang function is revisited again and XCSF is evaluated using different polynomial degrees for the prediction modeling step as well as with a new interpolation-based approach. For now, however, with the CII interpolation strategy activated, a marginal but nonetheless statistically significant reduction by -1.43% on average can already

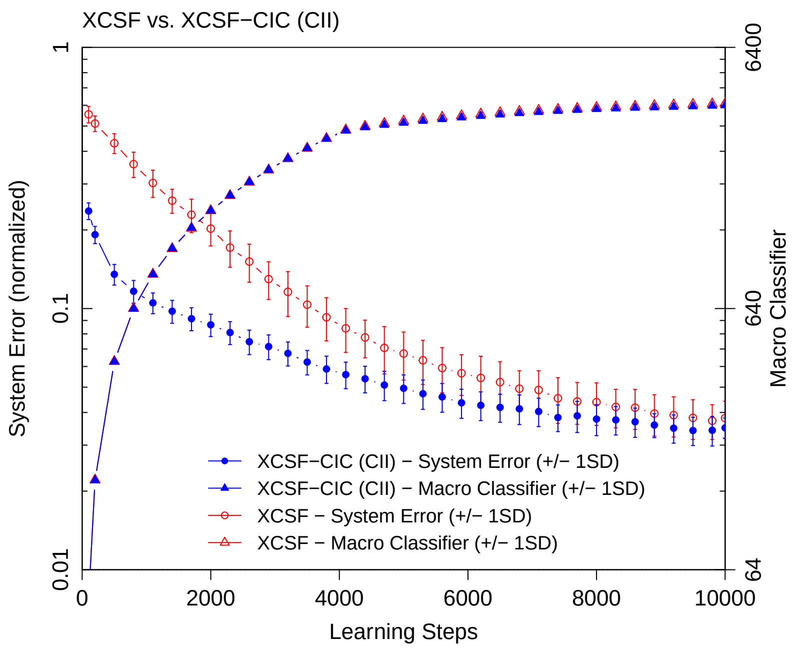


FIGURE 6.4.: Learning curves of XCSF-CIC with CII and XCSF on 3D Styblinski-Tang function f_2.

be achieved with regard to the system prediction error. However, the number of necessary classifiers to reach this error level again has marginally increased by $+0.48\%$, which is still found to be statistically significant. Having a look at the standard deviation, a slight increase for XCSF with CII can be observed for the system error, while a strong reduction with regard to the population size has appeared. Figure 6.5 reveals that the activation of the CII strategy still yields benefits during the very early stages of learning when covering is utilized. As noted before, the choice of a larger value for r_0 causes a reduced number of covering operations which in turn lessens the positive impacts of CII in this scenario. It also becomes apparent that the room for improvements through interpolation-based strategies seems to strongly depend on the principal ability of XCSF to learn a certain problem.

Results for the Highly Multimodal Eggholder Function f_3

The *Eggholder function* [JY13] is well-known as another benchmark function within the domain of numerical optimization. It is defined only for two dimensions and for

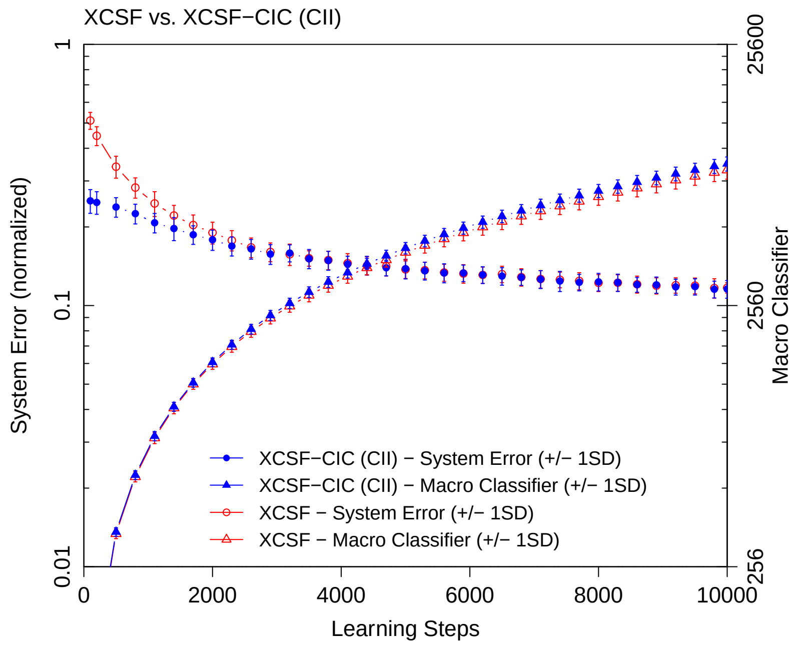


FIGURE 6.5.: Learning curves of XCSF-CIC with CII and XCSF on 6D Styblinski-Tang function f_2.

the input domain ranges $x, y \in [-512, 512]$ as follows:[2]

$$\begin{aligned} f_3(x,y) = &-(y+47)\sin\left(\sqrt{\left|\frac{x}{2}+(y+47)\right|}\right) \\ &- x\sin(\sqrt{|x-(y+47)|}),\ -512 \leq x, y \leq 512 \end{aligned} \tag{6.4}$$

Figure 6.2c illustrates the Eggholder function's characteristics. As can be seen the function surface appears to be continuous and highly multimodal with a large number of local optima. The repeating curvatures in both dimensions render this function a challenging approximation task for XCSF. XCSF is parameterized as for previous experiments except: $N = 6400, r_0 = 0.05$ and $m_0 = 0.02$. Due to the complex shape of function f_3, XCSF would not able to receive a sufficient fitness signal when initially generated classifiers would cover a too large fraction of the input space. Thus, following the recommendation of Butz et al. [BLW08; SB10a], r_0 and m_0 are set to small enough values in order to guarantee a sufficient fitness pressure toward accurate classifiers. Previously conducted parameter studies confirmed that smaller spread values are most suitable. Consequently, smaller initial condition hyperrectangles are created by covering which cover only small parts of the variety of valleys and peaks. Since here a linear approximation is used for the prediction computation, XCSF needs to divide peaks into several classifiers to capture the underlying function surface complexity. As for the previous function f_2, also the Eggholder function is revisited in Chapter 9, where an interpolation-based prediction modeling approach is introduced that is capable of approximating this challenging function with an accuracy far below the target error level ϵ_0. At this point, it clearly becomes apparent that hyperparameters are partially dependent what makes the configuration of XCSF a challenging and time consuming task – a hurdle that many modern ML approaches bear.

Having a look at the results given in Table 6.1, again the CII strategy turns out to accelerate the learning speed especially at the beginning (see Fig. 6.6) what results in a decrease of the system error by −5.97% while showing a similar standard deviation. Also the evolved number of macroclassifiers decreases non-significantly by −0.003% which can be considered to be due to chance.

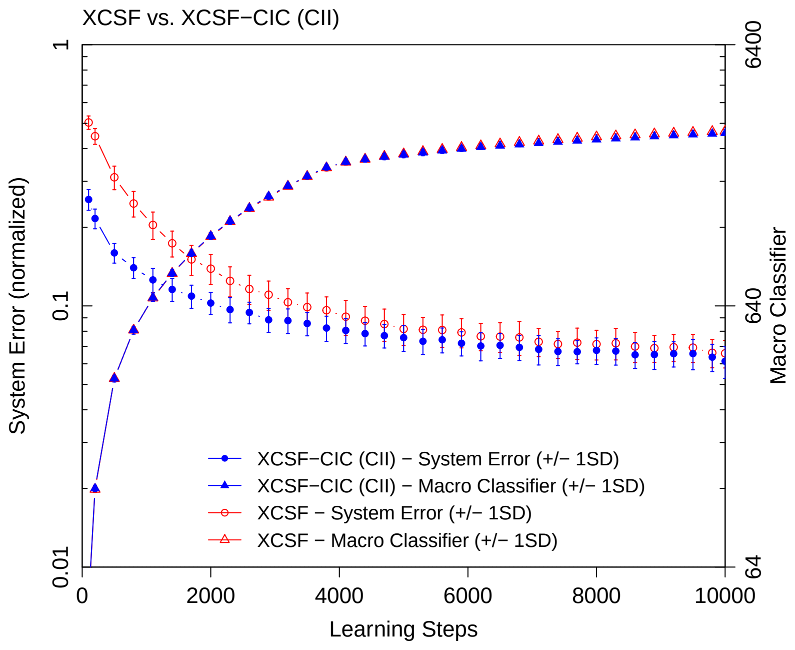


FIGURE 6.6.: Learning curves of XCSF-CIC with CII and XCSF on 2D Eggholder (f_3).

6.4. Discussion

Overall, it can be summarized that the interpolation-based covering strategy introduced in this chapter yields faster error declines at the early stages of learning when the covering operator is activated to counter KGs of type 1. For all investigated functions with their varying characteristics, XCSF-CIC using CII has obtained statistically significant reductions of the overall system prediction error over the entire learning task and averaged over the conducted 30 i.i.d. repetitions. For the simpler functions f_1 and f_2 in 3 dimensions, the observed standard deviations have also decreased in contrast to standard XCSF. This points toward a higher robustness against different sequences of situations to which XCSF is exposed. For the second metric investigated, the average number of macroclassifiers present in the rule base $[P]$, the use of CII leads to only slight increases, except for function f_3. However, the statistical significance has only been confirmed for the Styblinski-Tang function instances, where the relative amount of increase always stays below 1% which is deemed neglectable. Thus, CII can be considered an effective means to counteract completely uncovered regions of the input space and, therefore, the knowledge space, i.e., $kg \in KG_1 \subset KS$. It turns out to never have considerable detrimental effects on the learning performance, at least for the problem functions investigated in this chapter and for the supplemental results on the CBP provided in Appendix C. Accordingly, this novel interpolation-based covering operator constitutes a candidate for default utilization in XCS and related systems such as *Supervised Classifier System* (UCS) [BG03] and further derivatives.

The success however is dependent on the number of covering operations conducted during the learning phase of XCSF. Choosing high values for the initial spread hyperparameter r_0 leads to a quick coverage of the entire input space what prevents covering to be activated in conventional XCS implementations. An option to increase the number of covering steps would be to also analyze the formed match sets $[M]$ for their average qualities. If this quantity would fall below a predefined threshold then again covering could be activated in order to benefit from the CII interpolation strategy.

Further room for improvement is expected when not only the learning parameters are interpolated as introduced in this chapter, e.g., $cl.\vec{w}, cl.F$ and $cl.exp$, but also the conditions themselves. A first approach in this direction would be to interpolate the volume of the selected geometric representation of the conditions (e.g., hyperrectangles or hyperellipsoids). The interpolated volume could then be used to initialize an axes-parallel condition with equal stretches in each dimension, effectively forming hypercubes or hyperspheres as a first step. The involved GA would then be asked to act upon this informed condition initialization in order to form offspring with more appropriate orientations and stretches. With this extension to the CII strategy firstly explored in this thesis a reduction of XCS' sensitivity to the r_0 hyperparameter is deemed to be possible. In a later attempt, also the orientation of hyperellipsoidal conditions (which is encoded by explicit angles) could be subject to interpolation to take even more advantage of the interpolation-based covering approach.

By enabling XCS to transductively infer entire classifiers from neighboring ones it is also expected that the negative impacts of phenomena such as detrimental forgetting sometimes occurring in online learning settings (cf. e.g., [BS12]) can be attenuated to a certain degree. XCS never removes knowledge, i.e., rules, from more than one niche at a time when being trapped in a covering-deletion cycle. In the case that a just deleted rule was the only representative of a particular problem space niche and a situation occurs that exactly hits this now uncovered region, the CII strategy should be able to close this knowledge gap much more appropriately than standard covering.

Such situations also arise in problem domains which are sampled non-uniformly and are subject to covariate drift. Preliminary experiments already confirmed the hypothesis that interpolation-based classifier initialization strategies such as CII and the strategy which is introduced in Chapter 8 are especially valuable in such situations (cf. e.g., [Mei17]). However, the aim of this thesis is to develop the initial techniques for interpolation-based classifier systems and to provide first insights on their benefits on different problems. Therefore, an in-depth investigation of the discussed aspects for improvements would clearly exceed the scope and is thus left for future research.

6.5. Related Work

6.5.1. Rule Covering Schemes

Urbanowicz et al. propose to guide the covering of new rules by the incorporation of *expert knowledge* in [UGM12a]. By means of using *Spatially Uniform RelieF* (SURF) scores as a measure of feature quality, probabilities for further use in the covering and GA routines are derived using the logistic function. SURF works on the feature level of all instances in the available training set and estimates feature importance based on genotypic contradictions (feature manifestation) among the nearest neighbor instances. On the basis of the derived probabilities, the covering mechanism is guided in the sense of increasing or decreasing the chance of specification or generalization of the condition counterpart for the respective feature. For the SL scenario investigated in this study, the EK-guided covering operator yields superior accuracy performance in contrast to standard UCS [BG03].

Another version of covering is presented in the context of OC. To account for safety requirements in real-world environments despite using non-deterministic learning algorithms such as XCS, Tomforde [Tom11] and Prothmann [Pro11] introduced a strongly modified variant of XCS called XCS-O/C (cf. also Sect. 3.1 and Ch. 7). In this specific variant, the conventional covering routine was replaced by a *widening* approach [Pro+08]. In case of an empty match set, instead of creating a novel rule with more or less random initialization for the condition structure, the nearest neighbor classifier within the population is determined, copied and eventually its condition is widened to barely match the so far uncovered situation. If the distance to the closest classifier exceeds a certain threshold, a default action is realized. Anyways, an offline learning component equipped with an evolution strategy is activated that creates a novel rule and initializes it with optimized parameters and a reasonably general condition that conforms to domain specific expert knowledge. A concept to take this approach a step further toward proactive knowledge construction is presented in Chapter 10.2.

For endeavors regarding theoretical insights, populations in XCS have also been proposed to be *seeded* with random rule initially (cf. e.g., [But05a] and the statement of Kovacs and Bull in [KB07]), instead of incrementally creating new rules whenever $[M]$ is empty. However, this methodology has been quickly replaced with the standard covering routine as introduced by Wilson in [Wil95] for vanilla XCS and as recapitulated in this chapter when applied to more practical problems beyond purely theoretical investigations.

6.5.2. Further approaches to deal with sparse input domains in LCS

Since the interpolation of the classifiers' initial parameter values during covering is intended as a means to deal with KGs due to sparsity in the input domains (or data imbalances), related works on dealing with this issue have been researched.

Orriols-Puig et al. in [Orr+07] analyze the scalability of XCS with regard to the number of classifiers needed to appropriately learn imbalanced data. The derived theoretical models are partially empirically validated, however deviations are also reported when evaluating one of the presented evaluation scenarios. Furthermore, the theory is derived for binary input spaces, a fact that limits the transferability on real world data. Nonetheless, the insights provided by this study are definitely important for the utilization of XCS in problem domains exhibiting data imbalances.

In [OB05] Orriols and Bernadó Mansilla presented an algorithm-level adaptation of UCS, a derivative of XCS for supervised learning tasks. The authors propose to make the fitness adaptation *class-sensitive* to attribute more value to the learning steps when the minority class is to be classified. They discuss possible limitations when the training data is noisy or contains misclassified samples what possibly results in a limited generalization due to classification boundaries that are tailored too strongly to the training data. Furthermore, as the name of the algorithm suggests, an initial training set is necessary that is re-sampled certain times what contradicts the *online learning* capability OC systems usually require. Another point is that the techniques are clearly fitted to classification tasks. This is not the case for the CII method which is also applicable to function approximation tasks without further modification.

The question on the learning capabilities of XCS and UCS on imbalanced data is also thoroughly investigated and theoretically modeled by Orriols-Puig et al. in [OB08; Orr+09]. In their work, mainly the issues of *rare classes* and *rare cases* are focused and alleviated by online adaptation of XCS parameters such as the learning rate β and the threshold determining the GA activation θ_{GA}. This is an entirely complementary way to deal with this issue. Combinations of the techniques presented in this thesis and the aforementioned hyperparameter self-adaptation approach constitutes an interesting direction for future research.

6.5.3. Interpolative Fuzzy Rule Inference Systems

Looking in a entirely different direction as LCS research, another rule-base approach can be found in *Fuzzy Rule-based Systems* (FRS). These systems make use of *fuzzy rule inference* mechanisms based on fuzzy rules encoding *antecedents* (or premises) and *consequents* (or conclusions) which can be understood as the counterparts of conditions and actions in an LCS. A fuzzy rule base comprises several IF-THEN rules which evaluate a so-called *crisp* instance (e.g., a real-valued feature vector) after a *fuzzification* via the degree of membership to fuzzy sets which are typically encoded as *linguistic terms*. In the same manner, the consequents are decoded fuzzy and eventually need to be *defuzzified* to crisp (e.g., nominal) outputs again. More than one fuzzy rule can be active at a time, which makes a sort of mixing of the conclusions based on the membership degrees necessary. One the other hand, in *sparse* rule bases situations might occur where no fuzzy rule fires and no consequent can be calculated. This is where *interpolative fuzzy reasoning* [KH93] comes into operation. Regions in the *universe of discourse* which are not covered by any fuzzy rule are bridged via interpolation. A generalized concept is presented in [BKG04]. Fuzzy rule interpolation has been combined with RL, e.g., in the FRIQ-learning approach [Vin17]. Here, the consequents constitute the Q-value for certain crisp state action pairs which are interpolated. This allows for compact knowledge bases preventing overpopulation by unnecessary fuzzy rules. Clearly, the idea is related with the CII approach as developed in this chapter. New knowledge is obtained by making use of interpolation techniques applied to adjacent rules in the knowledge base. However, fuzzy rule interpolation works in the ‘fuzzy world’ whereas LCS directly deal with the crisp or real-valued data. The methodology of interpolative fuzzy reasoning have thus been found to not being directly transferable to interpolation-assisted LCS. Furthermore, whereas fuzzy rule interpolation mostly aims at interpolating the consequent (i.e., the action in LCS terminology), the approaches presented in this thesis can be seen more general. Besides the ASI technique that also interpolates the action for further use in a PA modification, also the learnable classifier parameters are subject to interpolation. Nevertheless, adopting the concept of only interpolating consequents without creating new rules in order to obtain more compact knowledge bases, and thus increase the comprehensibility of the overall solution for humans, is an interesting aspect of further research.

6.6. Chapter Summary

This chapter introduced an interpolation-based approach to covering in XCS. It was demonstrated how interpolation can be utilized to initialize selected learning parameters of a newly constructed classifier, i.e., $cl.p$ or $cl.\vec{w}$, $cl.F$ and $cl.exp$ during covering. Adjacent classifiers from neighboring niches serve as basis for transductive knowledge construction. The center points of each classifier served as sampling point coordinate. The current learning parameter estimates of the classifier considered for the interpolation were used as function values to be interpolated. Based on an empirical study targeted at the task of function approximation, the XCSF derivative was shown to yield superior performance in terms of strongly reduced system (or approximation) errors at early stages of learning where type-1 KGs are highly prevalent. This, however, was found coming at the cost of marginally increased average population sizes – a bearable shortcoming that usually not compensates the benefits. Current limitations and room for further advances have been discussed and related work targeting similar aspects was appreciated. Throughout the next chapter, the so far devised techniques ASI and CII are adopted within the so-called OTC system. This realistic scenario of self-adaptive traffic light management has been approached by using a strongly modified version of XCS by previous work in the context of OC. The modified XCS will therefore serve as testbed for the newly developed interpolation-based approaches to action selection and covering in the next chapter.

Chapter 7.

Application to Self-Adaptive Traffic Light Control

Autonomously adapting signaling strategies to changing traffic demands in urban areas have been frequently used as application scenario for SLAS [Kru+15]. The *Organic Traffic Control* (OTC) system [Pro11; Tom11] is one of the most prominent representatives in this domain. OTC implements the MLOC architecture and utilizes the purposefully adapted XCS-O/C modification as introduced in Chapter 3 for self-adaptation purposes. In this section, XCS-O/C is extended by the interpolation-based transductive inference techniques ASI and CII which have been developed in the preceding Chapters 5 and 6, respectively. The positive impacts in terms of reduced delay times are demonstrated by using a near-to-reality simulation of realistic traffic conditions obtained from a census in Hamburg, Germany.

Although a remarkable effort has been spent on developing SLAS that utilize efficient and safety-oriented solutions (cf. e.g., [Pro+11]), another important aspect has received only sparse research attention so far – How to instantaneously and at the same time robustly react to knowledge gaps when the offline generation of appropriate behavior by means of simulation-based learning (e.g., as proposed in [GR92; Tom+11a]) exceeds required time constraints?

This chapter starts with a short introduction to basic terminology in the domain of traffic management. Afterward, the OTC system is described in more detail. The proposed approach to incorporate purposeful adaptations of ASI and CII to counteract knowledge gaps in an ad-hoc fashion, constitutes the main focus of this chapter. The term ad-hoc refers to omitting the need to wait for an optimized solution received from a sandboxed offline-learning layer on top of the self-adaptive feedback loop system (the online learning layer). More details in this regard can be found in Chapter 3. The results of a conducted empirical study based on a commercial microscopic traffic simulator are then described and further discussed.

7.1. Traffic terminology

Urban traffic networks usually consist of a large number of intersections where two or more incoming and outgoing sections cross each other. Consider the case of a simple four-armed intersection: Traffic participants are able to enter and leave the intersection from every direction which clearly bears the risk of traffic jams or accidents due to vehicles or pedestrians passing it simultaneously. Therefore, traffic light signals indicate the permission of traffic participants to proceed for each possible *turning*. Each turning is related to a specific class of traffic participants, e.g., motorized vehicles or pedestrians. Of course not every traffic light is controlled separately. Traffic lights that consistently show the same signal form so-called *signal groups*. Figure 7.1 depicts four signal groups with their corresponding junctions, each surrounded by a circle.

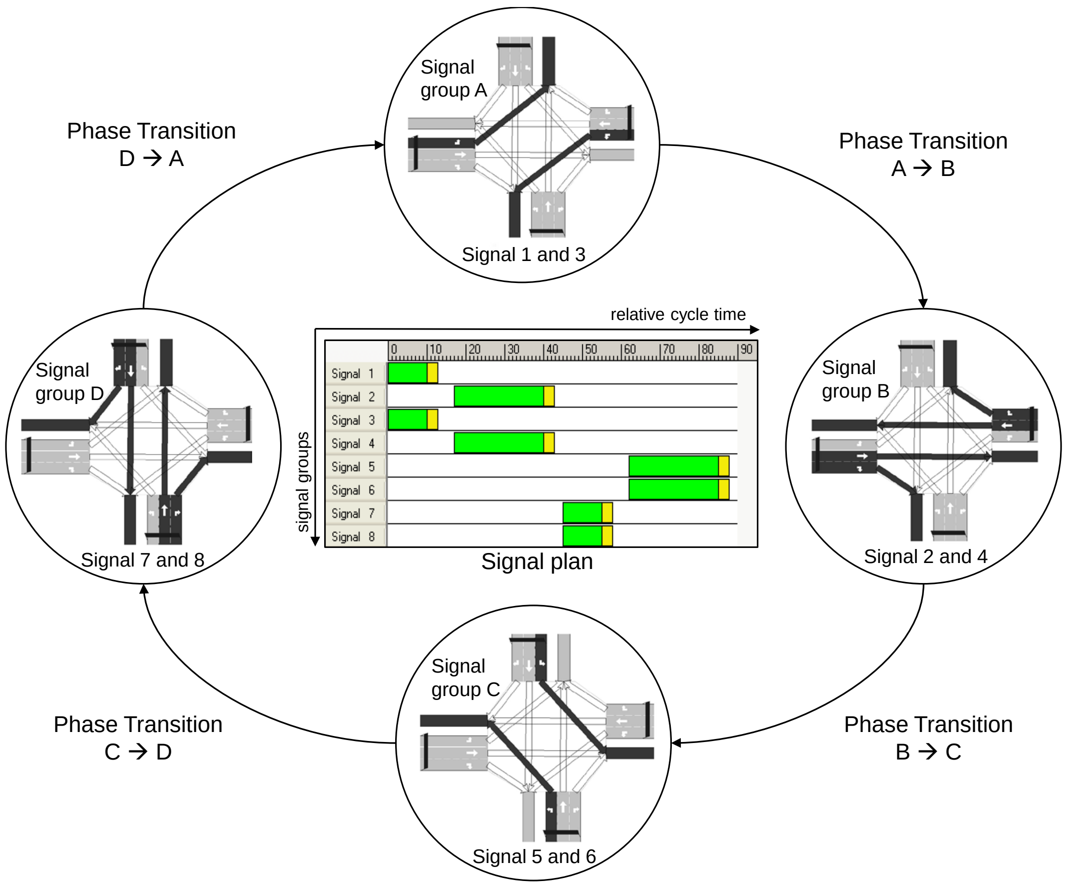


FIGURE 7.1.: An example of four signal groups with their corresponding signal plans

It also depicts the current *signal plan* for controlling the junctions. A signal plan determines the transitions between different phases that control the traffic flows into an intersection. This includes the periods of time for phases of green, yellow, and red signals. *Signal phases* comprise those periods that show green or else red signals concurrently. The elapsed time for one entire iteration through a signal plan is called the *cycle time*. Its duration is denoted by t_C. Each phase has a predefined activation time, which is called *phase duration* t_d. The time slot between activating two different phases is called *interphase*. Interphases are important for clearing purposes, i.e., to allow traffic participants which got stuck in the middle of a junction to leave it without causing traffic jams or accidents. The length of such interphases is usually regulated by a public authority. The number of maximally allowed vehicles on a turning is referred to as *saturation flow* S and it is measured in vehicles per hour ($\frac{veh}{h}$). The currently observable traffic flow at a turning is denoted by M.

In general, it is differentiated between two types of *traffic light controllers* (TLC) whose task is to manage the traffic light signaling.

Fixed-time controllers strictly follow a predefined signal plan that is estimated by traffic engineers. Once a signal plan is activated, it does not change. Hence, using fixed-time controllers, the only way to adapt to changing traffic demands is to deploy an alternative signal plan (mode-switch). Accordingly, sequences of signal plans are engineered that are supposed to closely match observed traffic demands. Variables that are considered in this process include the demand throughout the day, as well as the distinction between work days, weekends and holidays.

Traffic-actuated controllers allow for a different approach to the optimization of signal plans as a response to changing conditions. Using this type of TLC, signaling is not based on predefined plans, but rather on logical and temporal constraints observed and fine-tuned by traffic engineers in advance. Because of temporal restrictions that must not be violated, a traffic-actuated controller's ability to adapt to unforeseen situations is limited.

In the next section, the OC approach to face the challenge of missing adaptivity in the context of traffic light control is presented.

7.2. Organic Traffic Control

OTC [Pro+11] has been developed as one of the first OC systems which make extensive use of XCS-based online learning. The overall goal of OTC is to achieve a robust and optimized vehicular flow in urban traffic networks.

A majority of actually installed traffic control solutions in urban areas relies on fixed-time signal plans. OTC is built upon such a state-of-the-art solution and allows for a situation-dependent adaptation of the its configuration. More precisely, in self-adapts the green phases and strives for a continuing self-improvement regarding this adaptation policy.

Figure 7.2 illustrates the architectural design of OTC and emphasizes the integration of learning components.

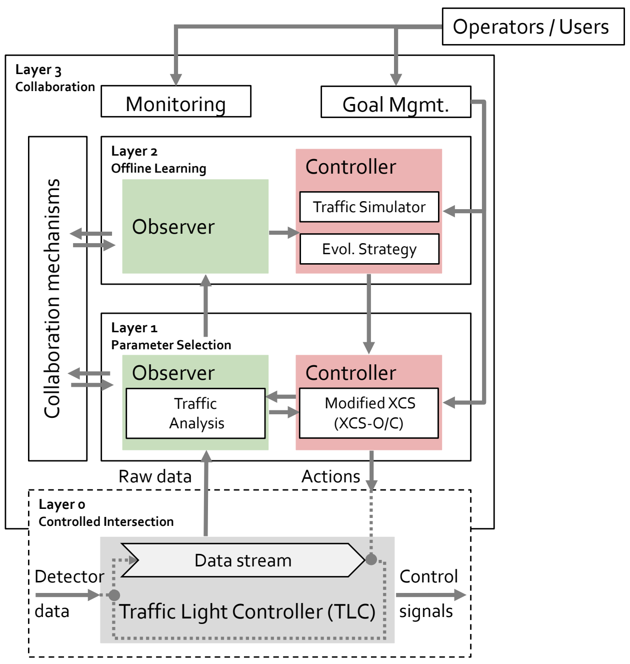


FIGURE 7.2.: Architecture of the OTC system, adapted from [Tom12]

As can be seen, the MLOC scheme serves as basis. On the bottom layer, a standard fixed-time traffic controller is encapsulated as the SuOC and equipped with interfaces to sense environmental conditions via detectors and manipulate the behavior in terms of switching the current mode, i.e., the *traffic light controller configuration* (TLC). The SuOC's control parameters are adapted by an online adaptation loop realized as an O/C instance as introduced in Section 3.1.

Figure 7.3 illustrates how such a TLC is composed by considering an exemplary three-armed intersection.

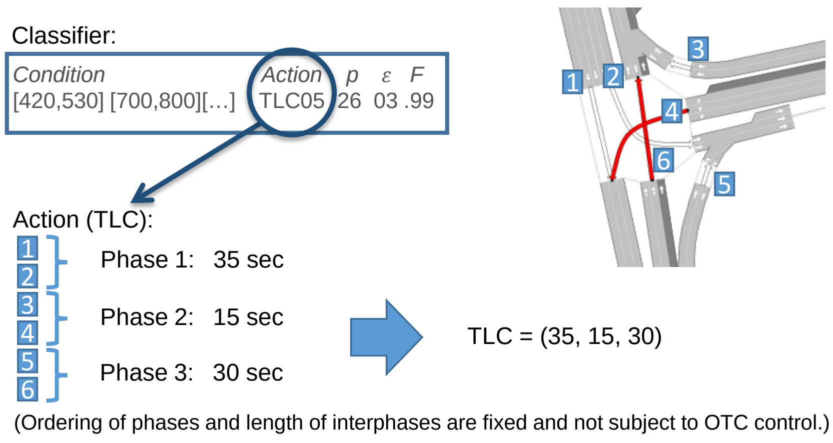


FIGURE 7.3.: Illustration of an exemplary action (here $TLC05$) as contained in an XCS-O/C classifier, adapted from [Ste+16a]

The green durations are specified for situated traffic lights (depicted by numbers 1 to 6 in the example intersection on the right) which are grouped to signalization phases. Both the phase ordering and phase lengths of so-called interphases (for clearing purposes, not depicted) are static. The lengths of the green periods are stored into a vector that encodes the TLC. The set of all possible TLC vectors constitutes the action space A of the learning task.

The self-adaptive control mechanism is separated into two layers according to the different tasks performed by the MLOC instance. Layer 1 is responsible for the *online selection* of a suitable control action, here a TLC, and for performing the reinforcement updates of matching classifiers which advocate the eventually selected TLC.

The observer component monitors the traffic flows arising at the controlled intersection. It processes the determined flow values into a situation vector representing the local traffic conditions at time t (denoted by σ_t). This information is periodically reported to the right-hand side controller. On that basis, XCS-O/C selects suitable classifiers from its knowledge or rule base $[P]$.

New classifiers for so far unseen traffic conditions are not created by covering as in usual XCS implementations. Rather novel rules are safely created by invoking the offline learning layer 2. In this second O/C layer, an *Evolution Strategy* (ES) [BS02] seeks TLCs for a specified traffic situation as observed from the first layer. The quality, i.e., fitness, of the computed TLCs is evaluated using either a computational simulation or else a specialized heuristic. This separation of concerns between feedback learning and offline rule generation follows Grefenstette's and Ramsey's concept of *anytime learning* [GR92].

In the following paragraphs, the operational process of OTC and XCS-O/C is described.

Figure 7.4 illustrates online decision and learning process of XCS-O/C.

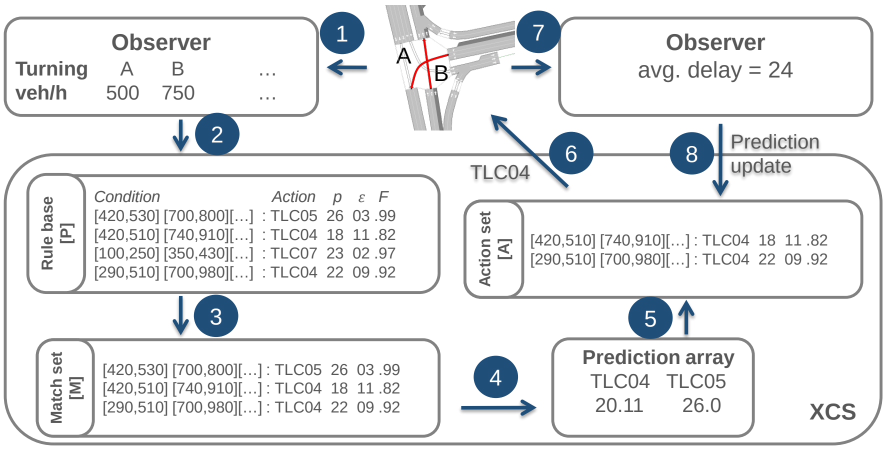

Observer
Turning A B …
veh/h 500 750 …
1
A
B
7
Observer
avg. delay = 24
2
6
8
Prediction update
TLC04
Rule base [P]
Condition Action p ε F
[420,530] [700,800][…] : TLC05 26 03 .99
[420,510] [740,910][…] : TLC04 18 11 .82
[100,250] [350,430][…] : TLC07 23 02 .97
[290,510] [700,980][…] : TLC04 22 09 .92
Action set [A]
[420,510] [740,910][…] : TLC04 18 11 .82
[290,510] [700,980][…] : TLC04 22 09 .92
3
5
Match set [M]
[420,530] [700,800][…] : TLC05 26 03 .99
[420,510] [740,910][…] : TLC04 18 11 .82
[290,510] [700,980][…] : TLC04 22 09 .92
4
Prediction array
TLC04 TLC05
20.11 26.0
XCS

FIGURE 7.4.: Illustration of OTC's online learning process at layer 1, adapted from [Ste+16a]

This adaptation cycle can be described by an 8-step process as follows:

1. In a first step, the current traffic conditions at the underlying intersection are perceived using detectors and then processed into an abstracted situation description vector σ_t. The situation vector represents the accumulated traffic flows of each signal group. Accordingly, the dimensionality of σ_t depends on the number of signal groups present at the controlled intersection.
2. Next, σ_t is matched against all classifiers contained in $[P]$ as usual (Step 2).
3. In the succeeding step, all matching classifiers are collected to form the match set $[M]$.
4. The prediction array (PA) is constructed as described in Section 2.4 in Step 4.
5. From the PA, one action is selected based on a greedy selection regime in the fifth step. In this scenario, a smaller system prediction value $PA(a)$ "wins", since the objective is to minimize delays.
6. Afterward, the selected action (here a TLC) alters the signal plan and, thus, the green duration of the traffic light controller (Step 6).
7. Subsequent to this adaptation process, and after a certain amount of time, a feedback signal has to be provided in order to complete the learning cycle. After running through all phases and interphases[1] for at least two cycles,

[1]Interphases denote the periods where the traffic lights show yellow or red.

the average delay for all traffic participants at the intersection is calculated according to Equation 7.1 (Step 7).

8. Finally, the payoff prediction estimate $cl.p$, which in this case estimates the expected flow-weighted delays, the prediction error $cl.\epsilon$ and the fitness $cl.F$ for all classifiers from the last action set $[A]$ (not to be confused with $[A]_{t-1}$ in multi-step tasks) are updated based the feedback signal obtained in the previous step.

The utility (or reward) function as required in Step 7 is given by the *average delay* for the *entire intersection* as determined by Equation 7.1.

$$t_D = \frac{\sum_i (M_i \cdot t_{d,i})}{\sum_i M_i} \tag{7.1}$$

M_i corresponds to the current traffic flow at the i-th turning (or junction) of the observed intersection. $t_{d,i}$ denotes the average waiting time with respect to a single turning t_i which is also measured by the system. This metric is also referred to as *Level of Service* (LoS) [Tra00].

Figure 7.5 illustrates the interplay of both O/C learning layers.

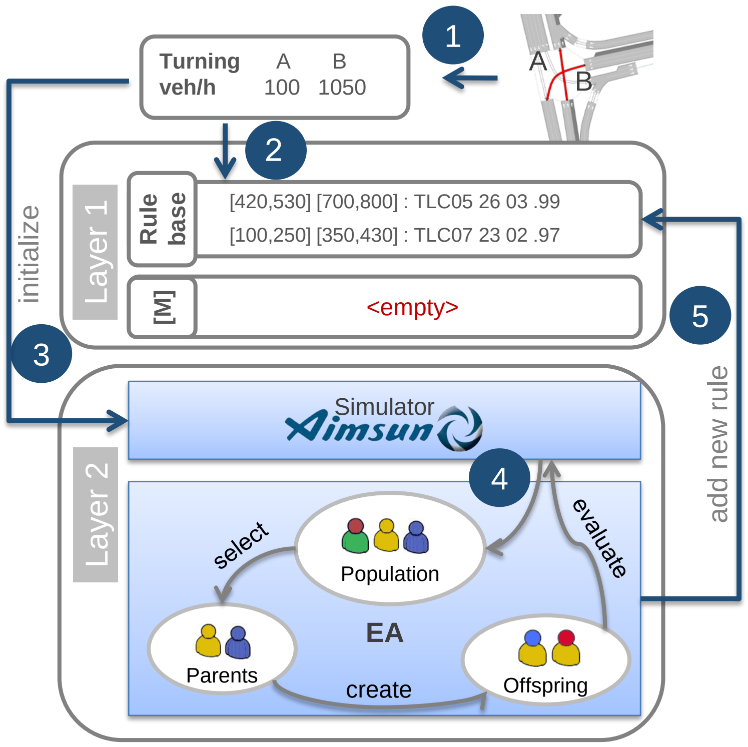


FIGURE 7.5.: Illustration of OTC's offline rule-generation process at layer 2, adapted from [Ste+16a]

As before, the process can be best described as a 5-step operation sequence:

1. Initially, Layer 1 reactively responds to the observed situation as described before.
2. In the case of missing knowledge, i.e., if the knowledge base $[P]$ does not contain an appropriate rule (type 1 KG), OTC applies a so-called *rule widening* operation.
    a) It selects the classifier from the population with the smallest Euclidean distance to the current situation σ_t (as described in more detail below).
    b) Given, that the distance does not exceed a predefined similarity threshold, the TLC action of the nearest neighbor classifier is applied to the system.
3. Simultaneously, OTC triggers a reactive knowledge construction process for the current situation by activating Layer 2. It performs a simulation[2] initialized to the currently observed traffic flows to obtain an appropriate TLC for the next occurrence this particular, or sufficiently similar situation.
4. Based on Equation 7.1, the ES optimizes the green phases for a predefined number of generations. Therefore, the fitness is determined either on the basis of simulations or by means of analytical models (heuristics) such as Webster's approximation formulas [Web58]. The best configuration of green phases is then used to generate a new classifier in the last step.
5. Finally, a new classifier is created and added to the knowledge base at Layer 1.
    a) The new classifier gets assigned an appropriately selected interval condition which encompasses the current situation description σ_t.
    b) The evolved signal plan TLC retrieved by the optimization process of layer 2 serves as its action.
    c) The obtained approximated flow-weighted delays are set as its initial reward prediction $cl.p$.
    d) The error estimate $cl.\epsilon$ is initialized with 25 and the fitness $cl.F$ with 0.01.

    This newly constructed rule is thus available for productive online adaptation from now on.

As can be recognized, the OTC system entirly adheres to the extended MLOC references architecture as introduced in Section 3.1 before.

[2] In this work, the proprietary traffic simulation tool Aimsun [Bar+05] which features a topological model of the underlying intersection is used.

7.3. Approach

In this section, the interpolation-based techniques ASI and CII are adapted to be used within the XCS-O/C and in the realistic context of self-adaptive traffic light control. Due to the removal of the standard GA from the modified XCS structure introduced elsewhere, the *Offspring Initialization Integration* (OII) technique is not applicable here. The initialization of the payoff prediction values is also modified in contrast to standard XCS. As will already outlined above, these values stem from an alternative evolutionary process that returns already appropriate estimates alongside. For that reason, also the *Interpolated Prediction Integration* (IPI) technique (to be introduced in Ch. 9) is not adopted in this scenario.

As discussed in the preceding paragraphs, the action space within OTC is determined by a set of multidimensional vectors that represent the possible green phase duration of the controlled signal groups. This action space is thus not only multi-dimensional, but also near to continuous. The boundaries of the action space are determined by legal constraints which prescribe minimum and maximum green phases. With regard to the CII technique, this circumstance opens a possibility that has already been discussed in Chapter 5 – a direct interpolation of actions instead of a modified argmax selection based on the cumulative interpolation weights. Therefore, in addition to the interpolation of a classifier's quality parameters (such as the fitness $cl.F$), here also the TLC action vectors, i.e., $cl.a$ are interpolated during the covering process. Since, the CII approach is applied, the second architectural variant of integrating XCS with the IC is chosen – the tightly integrated XCS-CIC approach (cf. Sect. 4.3.

For the case of XCS-O/C applied to the OTC scenario, a classifier maps a certain traffic situation σ_t, which comprises the current traffic flows in $\frac{vec}{h}$ per turning, to a traffic light configuration (TLC). Each TLC that can be selected by the adaptation logic of the system (layer 1) is evolutionary optimized by the offline learning layer 2. For exceptional cases and formerly unknown situations, a human-engineered standard signal plan serves as fallback solution. This standard configuration, however, is only used when XCS-O/C cannot respond directly. This can be due to long completion times of the offline optimization process.

In OTC, a classifier action $cl.a$ is always set to one of the optimized and validated TLCs as obtained by Layer 2. In the remainder, a distinct action for XCS-O/C is denoted by $TLCk$. Again, a $TLCk$ is represented by a vector with $j = 1 \dots p$ values, each encoding the green duration of one of the p signal group phases. Accordingly, the duration of the j-th phase of $TLCk$ is given by $TLCk_j$.

In order to construct the sampling points s_i from the classifiers $cl_i \in [P]$ the rudimentary approach as introduced in Section 6 is chosen. Accordingly, the center point of a classifier's condition is applied as sampling point coordinate, i.e.,

$$\vec{x}_i = cl_i.C.\vec{l} + (cl_i.C.\vec{u} - cl_i.C.\vec{l})/2,$$

where $cl_i.C.\vec{l}$ and $cl_i.C.\vec{u}$ define the n-dimensional vectors encoding the lower and upper bounds of the traffic flows which are accepted to meet the condition of cl_i.

For the utilization in OTC, the sampling point function values are selected as follows: The duration of each signal group phase in $cl_i.a = TLCk$ and the quality parameters $cl_i.p, cl_i.\epsilon$ and $cl_i.F$ serve as individual function value f_i. Accordingly, all function values f_i together make up a function value vector $\vec{f_i}$. For instance, in case of a $p = 3$-phased TLC, $\vec{f_i}$ is of the form

$$\vec{f_i} = (cl_i.TLCk_1, cl_i.TLCk_2, cl_i.TLCk_3, cl_i.p, cl_i.\epsilon, cl_i.F)^T. \tag{7.2}$$

As introduced for the CIC architecture in this thesis, the population of classifiers $[P]$ serves as set of sampling points SP. The weights W_i are calculated as defined for IDW in Equation 2.8.

With all the necessary ingredients defined, the interpolation-assisted XCS-O/C is enabled for performing interpolations. For this study, the IDW interpolation technique as defined in Section 2.8 has been applied since the population size within OTC is typically set very low such that local-support methods would presumably reduce to global techniques.

The following paragraphs shed light on how the selected strategies for interpolation integration, ASI and CII, are adapted for the application to the OTC scenario.

Interpolation-assisted Signal Plan Selection with ASI

In order to support the selection of appropriate signal plans transductively, the ASI strategy is adopted in the following way. To account for the additional information obtained from existing knowledge elements in $[P]$, the system prediction $PA(a)$ is again increased to influence the action-selection process. Therefore, as a first step the accumulated weight is calculated as follows:

$$W^{acc}_{TLCk} = \sum_i W_i \qquad \forall i : cl_i.a = TLCk \tag{7.3}$$

This accumulation implicitly reflects the presence of sampling points in the proximity of the current situation σ_t which advocate exactly $TLCk$. To account for a possible poor quality of the interpolation at initial learning phases, a trust-level T_{IC} is continually estimated to control the influence of W^{acc}_{TLCk}, as proposed earlier in this thesis (cf. Ch. 5). T_{IC} here is defined as the ratio (thus $\in [0,1]$) of the last ten situations at which the interpolated classifier yielded more appropriate signaling adaptations in terms of a LoS approximation using Webster's formulas [Web58].

As a next step, the ordinary system prediction $PA(a)$ (for simplicity directly denoted by $PA(TLC)$ in the following) needs to calculated as usual:

$$PA(TLCk) = \frac{\sum_{cl_i \in [M] | cl_i.a = TLCk} cl_i.p \cdot cl_i.F}{\sum_{cl_i \in [M] | cl_i.a = a} cl_i.F} \tag{7.4}$$

Based on the ordinarily calculated system prediction values, ASI can act by modifying the PA:

$$PA'(TLCk) = PA(TLCk) \cdot (1 + (W^{acc}_{TLCk} \cdot T_{IC})) \tag{7.5}$$

Since negative delays are not possible and IDW is utilized, the application of a max operator in Equation 7.5 in order to satisfy the utility bounds is not necessary here.

Interpolation-based Covering for Unknown Situations using CII

As briefly outlined above, covering in XCS-O/C selects the closest non-matching classifier from $[P]$, creates a copy of its condition, and widens the copy to barely match the current input. This widening approach generalizes the subspace that is covered by the new classifier. However, this generalization is done in a rather arbitrary manner, i.e., without being controlled via the accuracy criterion as done by the GA in standard XCS. In consequence, the covered classifier indeed matches more possible inputs, but likely at the expense of its accuracy.[3]

The condition $cl.C$ of a CII interpolated classifier is initialized analogously to those created by Layer 2. Interval predicates encoding a lower (l_i) and upper (u_i) bound for each dimension $i = 1 \dots n$ of a certain traffic condition $\sigma_t = (\sigma_1 \cdots \sigma_n)$ at time t are created. $(\vec{l}, \vec{u})$ is a vector notation for the lower and upper bounds. More formally, the intervals are calculated as follows:

$$(l_i, u_i) = \left(\min[\sigma_i - \tau, 0], \min[\sigma_i + \tau, \kappa_i]\right), \quad \forall i = 1 \dots n, \tag{7.6}$$

Here, τ denotes a *tolerance range* (or allowed spread) for the actually measured flow σ_i. κ_i denotes the maximum capacity of all turnings that belong to the i-th signal group. Thus, as typical for XCS, a classifier generalizes over similar situations – in this case traffic conditions observed at the controlled intersection. Still, the GA

[3]Activation of the standard steady-state niche GA would alleviate this limitation, however, the designers of OTC decided to reduce the degree of XCS's non-determinism for safety considerations. For the sake of comparability of results, this design choice has been kept for the experiments reported here.

does not optimize the input space coverage further as usual for XCS. However, the condition creation process introduced above can be deemed to directly incorporate expert knowledge in terms of a priori defined reasonable interval ranges.

As the last step to bring the CII strategy to XCS-O/C, the learning parameters of the newly constructed classifier are interpolated. The function values $\vec{f_i}$ that comprise the desired classifier parameters to be interpolated are chosen as defined by Equation 7.2 above.

The actual interpolation is then performed by building the sum of the component-wise multiplications of the weights W_i with all $\vec{f_i}$ over all considered sampling points $s_i \in SP$. Accordingly, the interpolation is calculated as follows:[4]

$$\vec{f}(\vec{x}_q) = \frac{\sum_{i=1}^{|[P]|} W_i \cdot \vec{f_i}}{\sum_{i=1}^{|[P]|} W_i} \tag{7.7}$$

To further account for the current level of interpolation quality, a probabilistic approach for deciding between a just interpolated and the standard signal plan is applied. Whenever the condition $U[0,1] \leq T_{IC}$ is satisfied, the interpolated classifier is selected. Otherwise, the human-engineered standard signal plan for the observed intersection comes into operation. This allows for a legitimate traffic signaling even at very initial stages of the system's runtime. Since the trust-level increases with system experience, the probability of selecting the interpolated classifier grows as well.

7.4. Evaluation

This section reports on the results obtained from an empirical case study where near-to-reality simulations of the application scenario have been applied.

7.4.1. Experimental Setup

An intersection situated at Hamburg, Germany[5] (also denoted as K3) serves as evaluation scenario. Figure 7.6 depicts the topology of this intersection.

[4]W.l.o.g., the weights W_i have been normalized to the range $[0,1]$. Thus, the denominator of Equation 7.7 can be neglected since the values of all weights sum up to 1.

[5]The K3 intersection is defined by the crossing of the following streets: Borsteler Chaussee, Alsterkrugchaussee, Deelböge, and Rosenbrook.

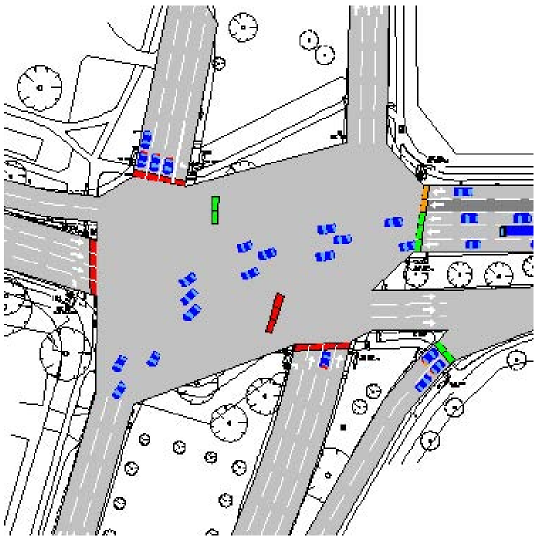


FIGURE 7.6.: Topology of a realistic intersection situated in Hamburg, Germany (K3) which is used to evaluate XCS-O/C with ASI and CII within the OTC system

The illustration shows the topological mapping of this intersection modeled with the Aimsun microscopic traffic simulator [Bar+05]. The Aimsun simulation model is configured with realistic and actually applied signal plans for this intersection (provided by the local authorities in charge). Finally, traffic data, which has been made available from a census performed by the local authorities on May 4, 2004, is used to simulate realistic traffic flows. The experiment covers the main part of the day, i.e., it is conducted for a simulation period from 5.30 am to 7 pm. This particular intersection was subject of investigation in more extensive studies of OTC [Pro11; Tom12]- It constitutes a more complex scenario compared to artificial intersection models, such as regular Manhattan-style typologies.

As already mentioned, the simulations are carried out by using the Aimsun simulator. Three approaches are subject of comparison:

1. The reference solution, i.e., the fixed-time control strategy actually installed in reality
2. An instance of the existing OTC system as introduced before, i.e., utilizing XCS-O/C without interpolation
3. The novel interpolation-assisted OTC system based on XCS-O/C with ASI and CII integrated

The last configuration is also denoted by XCS-O/C-CIC in the following. The performance of the three considered solutions is assessed in terms of typical traffic metrics which have a local scope (i.e., consider an individual intersection):

1. The *Level of Service* (LoS) as defined Equation 7.1

2. The averaged waiting time over all simulated vehicles

The presented experimental results are given as the averages over 10 i.i.d. runs for each solution. Additionally, in order to judge for statistical significance regarding obtained differences in performance, a paired t-test with significance level $\alpha = 0.05$ is performed.

XCS-O/C is configured as follows: $N = 200, \alpha = 0.1, \beta = 0.2, \epsilon_0 = 2, \nu = 5, \delta = 0.1, \theta_{del} = 20, \epsilon_{ini} = 25, F_{ini} = 0.01, \tau = 120$. Interval-based conditions using the OBR are utilized. The GA and respective mechanisms are set inactive. Payoff predictions are initialized based on the fitness value for the winning solution as obtained by the optimization process situated at Layer 2. XCS-O/C starts with an empty knowledge base, i.e., tabula rasa. This facilitates the occurrence of knowledge gaps especially at early stages of learning and simulates the occurrence of entirely unforeseen traffic conditions during the systems runtime.

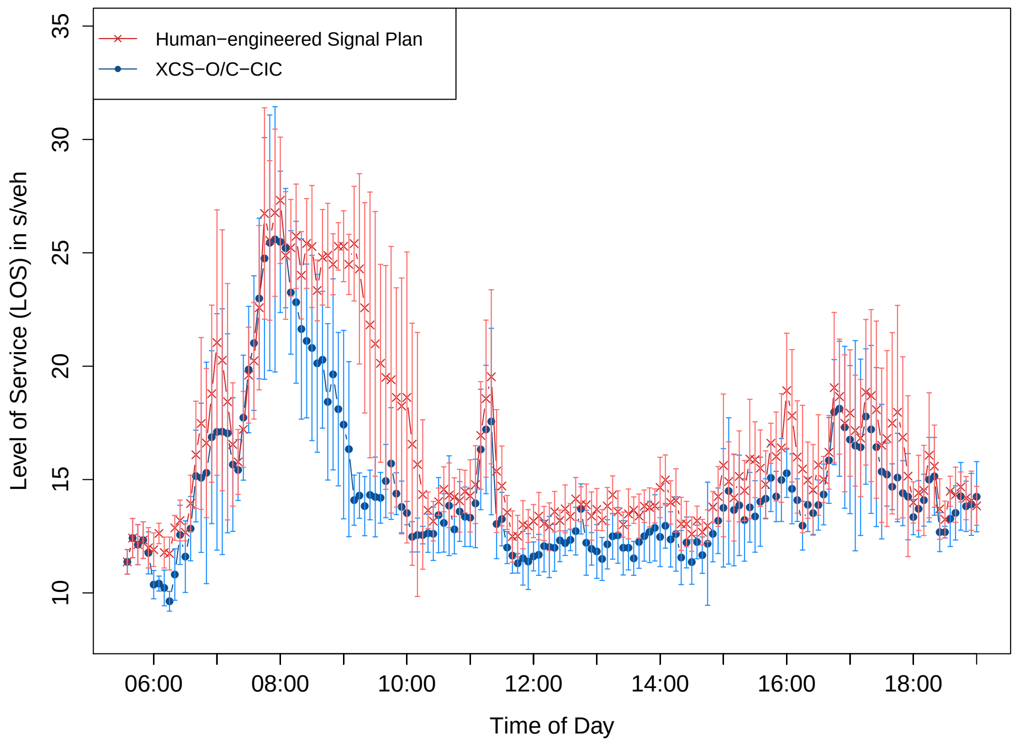
The ES at Layer 2 is applied with the following configuration: 64 generations are evolved. The Webster approximation formula [Web58] is used as fitness evaluation function. In each generation $\lambda = 24$ children are created, where the best $\mu = 16$ individuals are potential parental candidates for the next generation. The probabilities for mutation and crossover are set to 1. Crossover is performed as a discrete recombination for each allele, i.e., the offspring's final value is randomly chosen from one of the two parents (uniform crossover). Mutation adds noise to each allele following a Gaussian distribution $N(0; \sigma)$, where the standard deviation σ itself is included in an individual's chromosome and accordingly directly affected by the evolutionary process (*self-adaptation*). The initial value of σ for new individuals is set to $\sigma = 0.2$. This values are adopted from [Pro11], where a thorough sensitivity analysis has been conducted.

7.4.2. Results

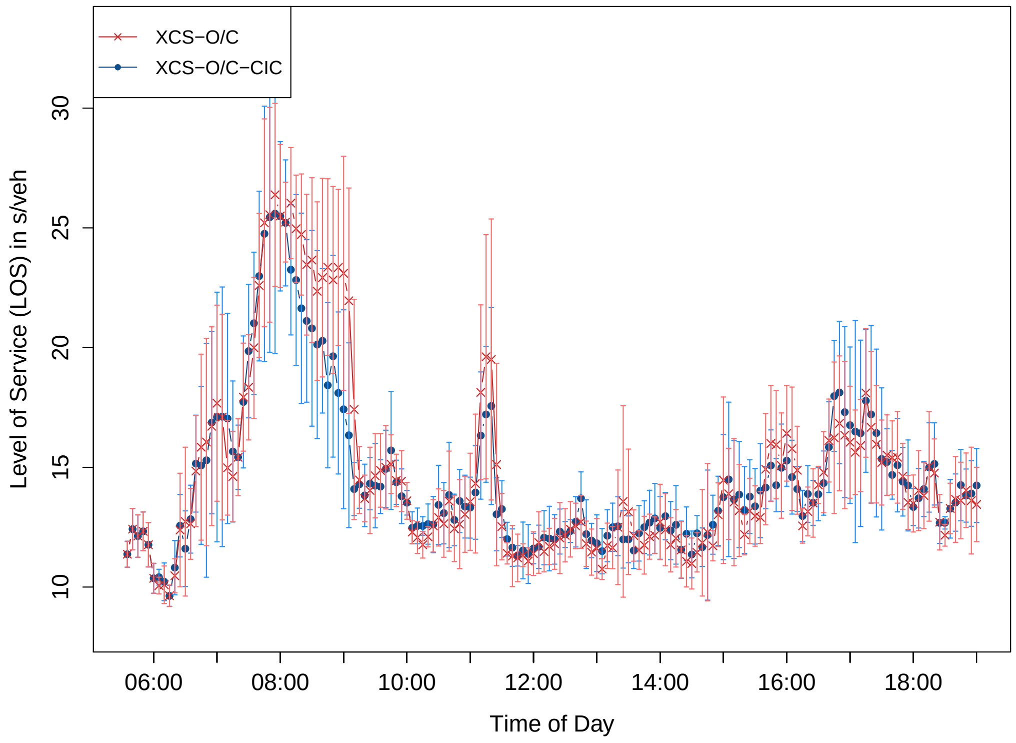
The approach is evaluated by means of two metrics, the LoS value and a similar time-delay metric that reflects the network-wide delay with respect to the considered intersection (in $\frac{sec}{km}$). The results of all variants are presented in Table 7.1.

Figure 7.7a depicts the average LoS values along with their standard deviations over a simulation horizon from 5.30 am until 7.00 pm.

Since the performance metrics of the human-engineered solution are already outperformed by the standard OTC approach without interpolation-assistance, in the following the focus is set on the comparison between the two OTC based solutions. The corresponding performance graphs are depicted in Figure 7.7b. As becomes apparent when having a look at the performance plots of the second experiment, the interpolation-extended XCS-O/C is able to slightly depress the steeply rising peak in the morning. More precisely, this effect can be observed starting from 6



(A) LoS performance of OTC with XCS-O/C-CIC vs. a human-engineered solution on the case study intersection K3



(B) LoS performance of OTC with XCS-O/C-CIC vs. standard XCS-O/C on the case study intersection K3

FIGURE 7.7.: LoS performance of the OTC system compared to a human-engineered signal plan on the Hamburg intersection K3

am, reaching its maximum at 8 am, and recovering to a normal level again until 10 am.

More importantly, the recovery time required until a normal level of delay is reached can be noticeably reduced. Similar effects can be observed whenever the detected traffic conditions rise quickly, e.g., from 11 am until 11.30 am and even to a higher extend when compared to the human-optimized signal plan. Since the population is empty at the beginning of the system's runtime, the traffic conditions arising at this time can be interpreted as knowledge gaps of type 1 (or entirely unforeseen situations). Just before the quick rise of the average delay as a consequence of an increasing traffic demand, a few classifiers are generated offline by means of the evolutionary component situated at Layer 2. Since this operation requires a certain amount of time, covering, and thus CII, is performed in order to deal with unforeseen situations.

The distinct improvements in terms of reduced average delay are attributed to the ability of the interpolation-assisted XCS-O/C-CIC to immediately make use of the early optimized signal plans in terms of a transductive on-demand inference of novel classifiers.

Compared to standard XCS-O/C, the interpolation-extended variant decreases the average delay by 2.17 $\frac{sec}{km}$ (15.15 $\frac{sec}{km}$ in comparison to the reference signal plan) which underpins the potential benefits more quantitatively. Table 7.1 summarizes the results of the experiments and further confirms the statistical significance. Results presented in bold indicate statistically significant ($p < 0.05$) reductions in comparison to both XCS-O/C and the human-optimized solution.

Table 7.1.: Summary of the experimental results on the case study intersection K3

	LoS (sec/veh)	**Avg. Delay (sec/km)**
	Mean (SD min|∅|max)	Mean (SD min|∅|max)
XCS-O/C-CIC	**14.60** (0.26|1.86|5.85)	**131.90** (2.60|17.27|62.74)
XCS-O/C	14.79 (0.35|1.90|5.87)	134.07 (3.08|18.03|55.42)
Human	16.47 (0.37|2.06|6.42)	147.05 (4.00|18.95|64.52)

Figure 7.8 depicts the progress of the second performance metric, i.e., the average network-wide delay in $\frac{s}{km}$ for the investigated intersection for all three considered solutions. Again a superiority of the interpolation-assisted solution with regard to robustness (i.e., recovery time) after unexpected and so far never seen situations becomes apparent.

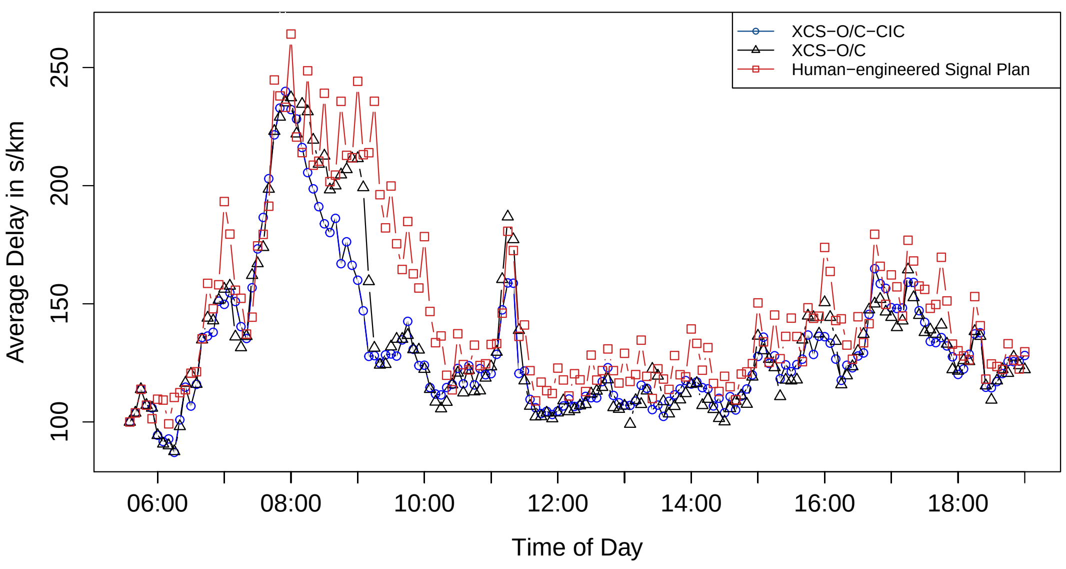


FIGURE 7.8.: Average delay performance of the OTC system compared to a human-engineered signal plan on the case study intersection K3

7.5. Discussion

Experimental results on a realistic case study intersection K3 that is actually implemented in Hamburg, Germany indicated that the integration of the ASI and CII techniques as developed in this thesis prove successfully applicable. The interpolation-assisted XCS-O/C-CIC yields superior performance in contrast to standard OTC as well as to a reference solution engineered by human experts. Especially in highly demanding conditions such as morning rush-hours the positive effects became apparent. This insight is relevant, since peak loads pose the most severe challenges to adaptive traffic control system. However, the focus of this experiment was set on the reaction of the involved online learning system, XCS-O/C, to unanticipated and unforeseen situations. Such situations constitute knowledge gaps within the system's knowledge base. Therefore, the system was assumed to learn *tabula rasa*, i.e., without an already initialized knowledge base. The only expert knowledge that has been assumed is the presence of a standard signal plan as developed by traffic engineers. This serves as fallback solution in situations where OTC can not yield an appropriate reaction – regardless of with or without interpolation assistance.

The objective of the reported study is to validate whether the benefits revealed for the transductive knowledge inferring techniques ASI and CII, which as of yet have been validated on artificial problem domains, also apply to realistic applications. This can be confirmed at least for the present case study of traffic flow optimization at an urban intersection situated at Hamburg, Germany.

The capabilities of the interpolation-based approaches have even taken a step further in this chapter. The CII strategy was extended to allow for dealing with large (or even continuous) actions spaces by means of interpolating between continuous actions (vector of green phase durations) as advocated by existing classifiers in the population. This constitutes a promising contribution to the field of RL in continuous domains and will be subject to more thorough investigations in the future.

7.6. Chapter Summary

In this chapter, it was demonstrated how the interpolation-based action-selection strategy called ASI and the interpolation-based covering scheme introduced as CII can be adopted to one of the most thoroughly investigated OC applications thus far – the OTC system. First, basic terms of traffic management have been briefly introduced along with the necessary details of the OTC system. In order to fit the modified structure of the specifically designed XCS version called XCS-O/C, a couple of adaptations have been described. Experimental results on a realistic intersection situated at Hamburg, Germany, which was simulated by means of the Aimsun microscopic traffic simulator, have been reported. The results further corroborate the benefits of using interpolation-assisted XCS for the sake of runtime learning in real-world scenarios. Due to the purposeful modifications to XCS-O/C necessary to account for safety requirements in such a safety-critical scenario where humans are immediately involved, only the novel approaches ASI and CII can be plausibly adopted in the OTC system. In the next two chapters, however, further strategies to incorporate interpolation in the XCS are going to be presented.

Chapter 8.

Interpolation-Based Offspring Initialization

In this chapter, another step toward an interpolation-assisted classifier system is taken. Having extended the action-selection mechanism in Chapter 5 as well as the covering routine in Chapter 6, the involved steady-state niche GA is now the subject of investigation. One of the major shortcomings of the interpolation-based covering operator is that typically it is only performed at early stages of learning until the population $[P]$ entirely covers the underlying input space. After this phase, covering usually occurs only sporadically. Possible reasons are e.g., detrimental forgetting or the occurrence of rare or unforeseen situations caused by non-uniform input distributions or, even worse, covariate drift. Accordingly, another part of XCS's algorithmic structure is to be found which acts throughout the entire learning process. The involved steady-state GA constitutes exactly such a component. It is activated periodically depending on the configured activation threshold hyperparameter θ_{GA}. In order to extend the GA to also take advantage of transductive inference from neighboring classifiers by means of interpolation, the novel *Offspring Initialization Integration* (OII) strategy is introduced in this chapter.

Therefore, it will be recapitulated how novel classifiers produced by the GA are created and initialized in conventional XCS implementations first. The novel OII strategy for extending XCS's GA by means interpolation is then introduced. As for the previously developed CII technique, the initialization values for the learning parameters of newly created classifiers serve as subjects for interpolation. The benefits of interpolation-based offspring initialization will again be validated on the basis of empirical experiments. The same benchmark problems as for the CII approach are considered. Additionally, the combination of CII and OII will be subject of discussion. The results and their statistical significance in comparison with the conventional approach is assessed subsequently. A discussion on current limitations and potential extensions complements the empirical results. At the end of this chapter, a dedicated section on related work is presented.

8.1. Basic Offspring Creation

XCSF applies a niche GA on $[M]$ whenever the mean time since the last GA invocation surpasses a predefined threshold θ_{GA}. Therefore, each classifier maintains another book-keeping parameter $cl.ts$ which serves as timestamp when the last GA was applied on the match set to which this particular classifier belongs. If the GA is activated, two parental classifiers from $[M]$ are chosen by means of *tournament selection* as described by Butz et al. in [BSG03]. For the standard offspring creation, the selected parents cl^1_{par} and cl^2_{par} are copied to construct two offspring classifiers cl^1_{off} and cl^2_{off}. Afterward, crossover is applied to the children by chance with rather high probability χ (typically $\chi = 0.8$). Crossover can be performed *uniformly*, i.e., each allele is switched between the two children with equal probability, or via so-called *n-point* variants. For the latter, n crossing points are selected randomly, and the resulting parts in between are switched among the offspring rules. Eventually, crossover alters the geometrical condition structure in order to explore more suitable partitions within the current environmental niche. Mutation modifies the conditions probabilistically as described in [Wil02]. Whereas in conventional XCS for binary inputs the mutation applied exerts pressure toward more specific conditions [But05a], for real-valued XCS variants, this pressure does not apply anymore. Thus, in standard XCSR and XCSF specialization purely occurs by chance. The only exception is when mutation alters the lower (upper) condition bounds to exceed the input space bounds. In this case, *clipping* at the boundaries implicitly leads to more specific classifiers, i.e., conditions with smaller volumes.

In XCS(F)'s standard GA, values for the parameters $cl.\epsilon$ and $cl.F$ are set to be the mean of both parents. Subsequently, these means are discounted by predefined factors $\epsilon_{reduction}$ and $F_{reduction}$ (see Appendix B for an overview of all hyperparameters). Regarding standard XCS, the prediction scalar $cl.p$ is set to the average of both parents. In contrast, for XCSF or any XCS that utilized computed predictions, the weight vectors are directly inherited by the parents according to [Wil02; Lan+05a]. Assuming a true supervised learning task as done in [SB12; Sta14] for instance, the offset weight w_0 could be set to the actual function value $y = f(\vec{x})$, given that it is passed to XCSF along with the input vector $\vec{x}$. However, as previously stated, for the stated learning problem as defined in this thesis, this assumption does not hold. Thus, it is expected that the actual value is received after the actual prediction step in form of the immediate reward $r_{imm} = y$. The bookkeeping parameters $cl.exp$ and $cl.num$ are set to 0 and 1, respectively, following the conventional approach.

After this typical application of the genetic operators selection, crossover and mutation and the initialization and adjustment of the offsprings' learning parameters, a GA subsumption attempt is performed before the novel rules are finally inserted into the population $[P]$ (cf. Sect. 2.4 for more details).

8.2. Interpolation-Based Offspring Initialization

The actual interpolation of the offspring classifiers' parameter vectors $\vec{f}_{off}$ works analogously to the CII procedure, as Equation 8.1 suggests.

$$\vec{f}_{off} = \frac{\sum_{i=1}^{|[A]|} W_i \cdot \vec{f}_i}{\sum_{i=1}^{|[A]|} W_i}, \text{ with } cl_i \in [A] \tag{8.1}$$

A major difference between OII and CII is that for OII the set of considered sampling points is defined to be the action set $[A]$[1], which assures to interpolate within the environmental niche of interest. Furthermore, for the OII strategy, the interpolation of the weight vectors $\vec{w}_i$ is omitted, since a range of preliminary experiments indicated inferior performance. This can be attributed to the fact that the fittest rules within a certain action set $[A]$ have gained their higher fitness estimates through being accurate in their predictions. Thus, their prediction scalars $cl.p$ or their coefficient vectors $cl.\vec{w}$ should already be most appropriate. Including the coefficient vectors of further less fit rules in the same niche during interpolation is thus deemed to be detrimental at this place. Accordingly, the function value vector for the interpolation step as used in Equation 8.1 is made up of only two components:

$$\vec{f}_{off} = \begin{pmatrix} cl_{off}.F \\ cl_{off}.exp \end{pmatrix}$$

A third difference is the selection of the query point $\vec{x}_q$. In contrast to CII, not the current situation vector $\sigma_t = \vec{x}_t$, but the center point of the individual offspring classifier $cl_{off}.C.\vec{c}$ is chosen. With that, the parent whose center point is closer to the offspring's center in terms of the Euclidean distance gains a higher interpolation weight and, thus, a stronger influence regarding the initialization of its child. The created two offspring classifiers are then initialized with the interpolated values for their fitness $cl_{off}.F$ and experience $cl_{off}.exp$. Subsequently, the fitness correction is applied by multiplying the $F_{reduction}$ factor to the interpolated fitness values. This is done to account for the fact that the offsprings numerosity values are initialized to 1 what impacts the receivable fitness share during reinforcement (see Sect. 8.4 for more details).

As usual, after the parameter initialization and adjustments of the created offspring, subsumption is performed to check whether one of the immediate parents or any classifier from the environmental niche $[A]$ is a valid subsumer (cf. Sect. 2.4 for details).

Figure 8.1 is intended to convey the intuition of the OII technique.

[1]In the case of XCSF, the action set $[A]$ is equivalent to the match set $[M]$.

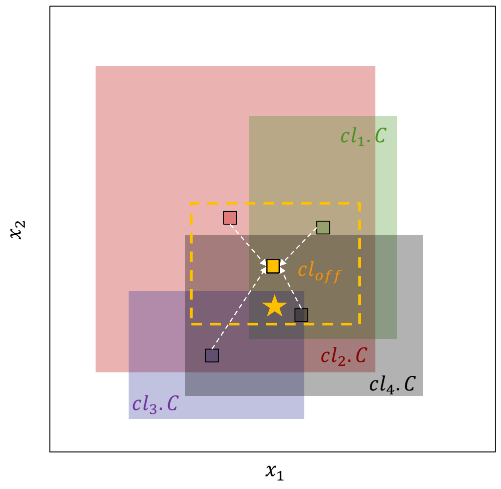


FIGURE 8.1.: Intuition of the interpolation-based offspring initialization technique OII

The schematic illustrates an exemplary niche of an arbitrary problem space currently covered by four classifiers (depicted in different colors). The dashed yellow rectangle shows an offspring's condition shape after the application of the genetic operators crossover and mutation. The star symbol indicates the current situation σ_t based on which the action set is formed. The small squares in the center of each classifier correspond to the center points of the conditions, which make up the sampling point coordinates for the interpolation. The yellow square in the middle of the offspring classifier cl_{cov} serves as the query point $\vec{x}_q$.

The main idea of OII is to take advantage of the knowledge present in the direct proximity, i.e., the same problem space niche, when novel rules are created by means of the involved GA. This can be interpreted as the introduction of some sort of influence from the "social surrounding" on young classifiers. More precisely, at this primal stage of the OII strategy, newly created classifiers are supposed to benefit from niche-wide experience $cl.exp$ and fitness values $cl.F$ (which is by definition a niche-relative estimate of a rule's accuracy) of all $cl \in [A]$.

In terms of the introduced notion of knowledge gaps, the OII strategy is intended to alleviate type-2 KGs. Since the GA acts on the action set in case of standard XCS(R) and on the match set in case of XCSF, type-1 KGs are by definition excluded. Nevertheless, the application of the GA bears the risk to produce KGs of type 2 due to inappropriate recombinations of probabilistically selected parental classifiers by chance. Interpolating the experience and fitness estimates between the values of classifiers sharing the same niche is hypothesized to enable the deletion mechanism to act more against such poor quality offspring rules. As is described more detailed in the corresponding Section 2.4, the rule deletion mechanism favors classifiers which (1) exceed a predefined experience threshold θ_{del}, and, (2) show smaller microfitness values $cl.F/cl.num$ than a predefined fraction δ of the population's average (micro)fitness $\bar{F} = \sum_{cl \in [P]} cl.F / \sum_{cl \in [P] cl.num}$.

If an offspring classifier is now initialized by means of OII, exactly its experience and fitness values are more strongly influenced by closer rules in a common niche. In the case that the genetic operators produced an inappropriate condition structure $cl_{off}.C$, its center point is expected to lie closer to other inappropriate rules (i.e., type-2 KGs). Therefore, the fitness value is modified to reflect the insuffcent quality more strongly in contrast to only consider the immediate parents' high fitness estimates. This pushes weak offspring rules toward the calculation of higher deletion votes. This effect is even more amplified by not simply initializing the offspring experience to 0 as usual, but instead assigning a value reflecting the distance-weighted niche average. In consequence, due to a more effective removal of GA-created type-2 KGs, the effective size of $[P]$ in terms of actually stored classifiers is expected to shrink.

8.3. Evaluation

In order to validate the effect of the OII strategy to be utilized in the IC-extended XCSF-CIC, the same procedure as for the previously introduced CII technique is pursued. The same three functions to be approximated by XCSF serve as benchmark for the evaluation.

Since OII is also intended to eradicate the shortcoming of CII which only applies at early stages of the learning process, the combination of both is additionally investigated in this chapter in order to fathom the potential synergistic benefits.

For the sake of brevity, again only the results of the study focusing on XCSF are presented in this chapter. Supplemental results obtained from integrating OII and the combination with CII in XCSR applied to the CBP scenario can be found in Appendix C and together with more detailed discussions are provided in [Ste+17a].

8.3.1. Experimental Setup

XCSF is configured exactly identical as before in Chapter 6. Again, MSM interpolation is used throughout the experiments. In the case that the action set $[A]$ contains less than the desired number of sampling points $N_w = 19$ for the local interpolation performed by MSM, the global IDW technique automatically serves as fallback solution by simply taking all available sampling points, i.e., classifier in $[A]$, into consideration for interpolation. 30 i.i.d. repetitions with $200k$ learning steps each have been conducted. The outcomes of paired t-tests which have been conducted in order to assess the statistical significance of the results are indicated in Table 8.1. To make sure that t-tests are appropriate, the experimental results have been positively evaluated to follow a normal distribution using the Shaprio-Wilk test and visual inspection of QQ-plots. As before, the approximation results on three different functions of varying complexity are reported and further discussed in the next paragraphs. The test functions comprise: (1) the $n = 6$-dimensional convex RMS-like function f_1, (2) the multimodal Styblinski-Tang function f_2 in $n = 3$ and $n = 6$ dimensions, as well as, (3) the complex $n = 2$-dimensional Eggholder function f_3 with its high degree of curvature. For a graphical representation of the investigated test functions confer Figure 6.2.

8.3.2. Results

Table 8.1 summarizes the results obtained from the conducted empirical studies. * (**) indicates statistically (highly) significant differences regarding the reported metrics compared to standard XCSF. This means that for the p-values of paired two-sided t-tests the assertion $p < \alpha = 0.05$ (0.01) evaluates true. In case that only one requirement for using parametric t-tests is violated, a non-parametric Wilcoxon signed-rank test is used instead in order to double-check the correctness of the null-hypothesis rejection. Bold values indicate the highest significant improvements compared to standard XCSF. The arrows, $\uparrow$ or $\downarrow$, indicate increased or decreased values, respectively. Plots illustrating the prediction error curves as well as the population size development are presented for both configurations, i.e., OII and OII+CII for each benchmark function. The points show the aggregated values over 5000 learning steps each. The corresponding bars indicate the measured standard deviations of each point over the 30 i.i.d. experiment repetitions.

Results on the Simple RMS-like Function f_1

On the relatively easy to approximate function f_1, XCSF-CIC using the OII strategy only reveals positive effects on the average number of macroclassifiers in the population, but negative influences regarding the achieved system error. In comparison to standard XCSF, the interpolation-based offspring initialization reduces the population size by 9.87% on average. This observation underpins the hypothesized impact of OII on the population development, i.e., that type-2 KGs created by the GA can be more effectively identified and consequently removed from $[P]$. However, the reduced average population size comes at the cost of a slightly higher (0.9%) overall system error on average. This small difference has still been found to be statistically significant, but only with a significance level of $\alpha = 0.05$. As Figure 8.2a reveals, the OII scheme impacts the learning progress over the entire $200k$ learning steps.

Since it was intended to complement the revealed positive effects of the interpolation-based covering operator with the CII strategy, the combination of both has also been investigated.

TABLE 8.1.: Summary of results for XCSF-CIC using the OII strategy individually as well as in combination with CII on the selected test functions f_{1-3}

f_1 **RMS-like 6D** (target error $\epsilon_0 = 1$)	*System Error* mean±1SD	*Macro Classifiers* mean±1SD
XCSF-CIC w/ **OII**	.6750±.014 ↑*	**1511.06**±12.28 ↓**
XCSF-CIC w/ **OII+CII**	**.5387**±.005 ↓**	1547.73±10.92 ↓**
XCSF	.6690±.013	1676.59±16.95
f_2 **Styblinski-Tang 3D**	*System Error*	*Macro Classifiers*
XCSF-CIC w/ **OII**	.0269±$2.5 \cdot 10^{-4}$ ↑**	**2659.01**±18.96 ↓**
XCSF-CIC w/ **OII+CII**	**.0223**±$1.9 \cdot 10^{-4}$ ↓**	2766.32±17.06 ↓**
XCSF	.0260±$2.8 \cdot 10^{-4}$	3250.96±16.75
f_2 **Styblinski-Tang 6D**	*System Error*	*Macro Classifiers*
XCSF-CIC w/ **OII**	.0696±$6.9 \cdot 10^{-4}$ ↓**	**14765.20**±97.07 ↓**
XCSF-CIC w/ **OII+CII**	**.0683**±$5.8 \cdot 10^{-4}$ ↓**	14805.32±77.77 ↓**
XCSF	.0699±$5.4 \cdot 10^{-4}$	18201.20±125.49
f_3 **Eggholder 2D**	*System Error*	*Macro Classifiers*
XCSF-CIC w/ **OII**	.0540±$4.5 \cdot 10^{-4}$ ↓**	**2518.66**±12.24 ↓**
XCSF-CIC w/ **OII+CII**	**.0491**±$6.3 \cdot 10^{-4}$ ↓**	2540.76±12.10 ↓**
XCSF	.0553±$5.7 \cdot 10^{-4}$	2955.01±9.37

Figure 8.2b depicts the learning curves of XCSF with and without both interpolation integration strategies activated. As can be recognized, a smaller population size can still be maintained while additionally a smaller overall system error is achieved. In numbers, this means that XCSF-CIC with both interpolation strategies activated can reduce the overall system error by 19.48% on average. The average population size is again decreased by 7.69%. Thus, the combination of OII and CII allows to retain the positive effects of both methods. That is: (1) A significantly reduced system error at early learning stages as well as over the entire experiment for CII. (2) A clearly reduced population size as achieved by OII.

Considering also the observed standard deviations, it can be noted that the combination obtains the improved results with lower variability and, therefore, more stably.

Results on the Curved Styblinski-Tang Function f_2

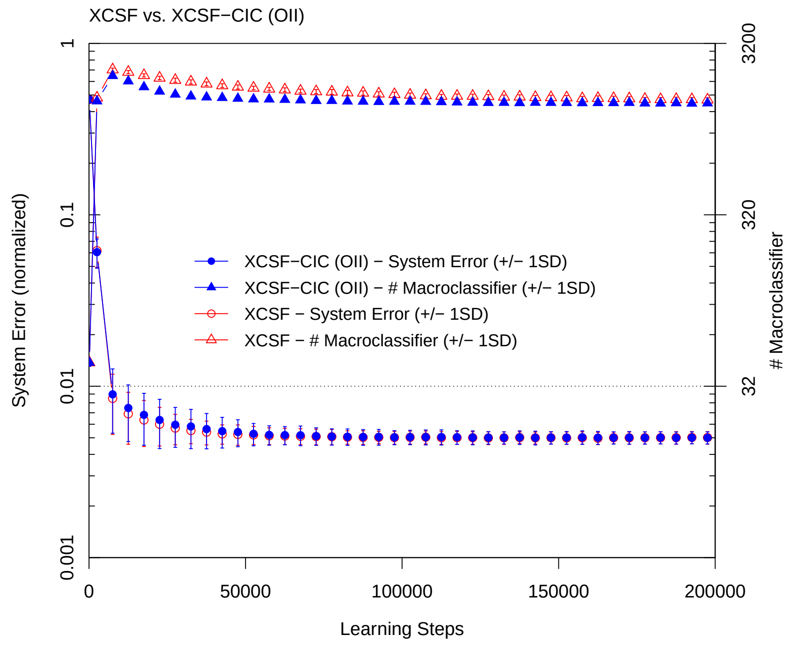
Analogous positive effects as obtained for the RMS-like function f_1 also can be observed for the application of XCSF-CIC on the Styblinksi-Tang function f_2. Again XCSF-CIC with both OII and CII active produces smaller approximation errors as well as decreased population sizes on average. This holds true for the $n = 3$ and the more complex $n = 6$ dimensional case.

n=3-dimensional case Using OII in isolation yields an by 3.46% increased approximation error. The difference has been found to be statistically significant. The upside, however, is the drastically reduced population size by ≈ 591 macroclassifiers. This evaluates to a relative reduction of 18.21%.

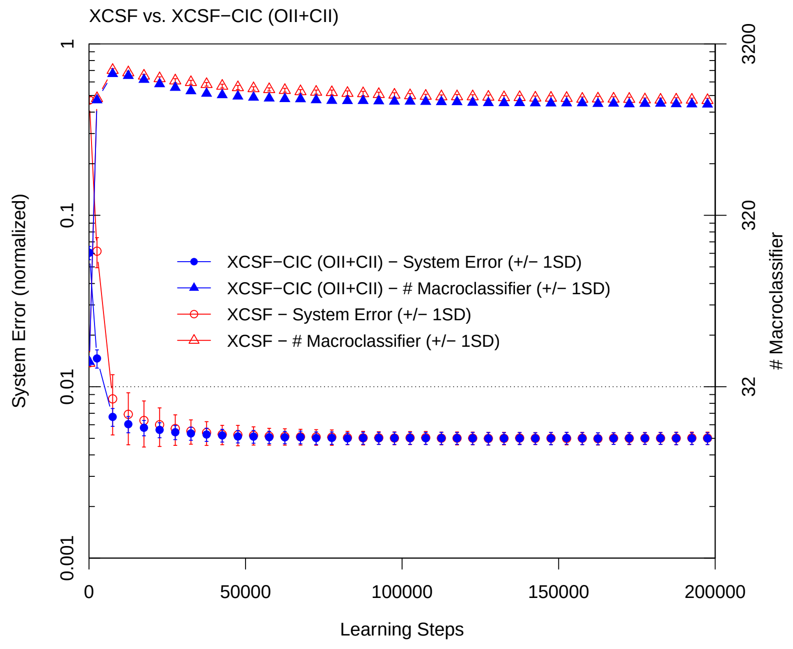
Combined with CII, the overall approximation error can eventually be reduced by 14.23% on average and with a decreased variance. Although slightly worse than for OII alone, the combination with CII still allows an effective population size reduction by 14.91%.

n=6-dimensional case Even for the 6-dimensional instance of the non-convex and multimodal function f_2, the so far made observations are observable. This time, the OII strategy alone in fact improves the approximation accuracy which is reflected by a marginal but still statistically significant decrease in the system error by 0.43% on average. The average population size can be decreased by ≈ 3436 macroclassifiers, i.e. by 18.88%.

XCSF-CIC with OII and CII involved further improves the obtained results. The approximation error drops by 2.3% and the average number of macroclassifiers in the rule base can be observed to decrease by 18.66% and with a significantly lower

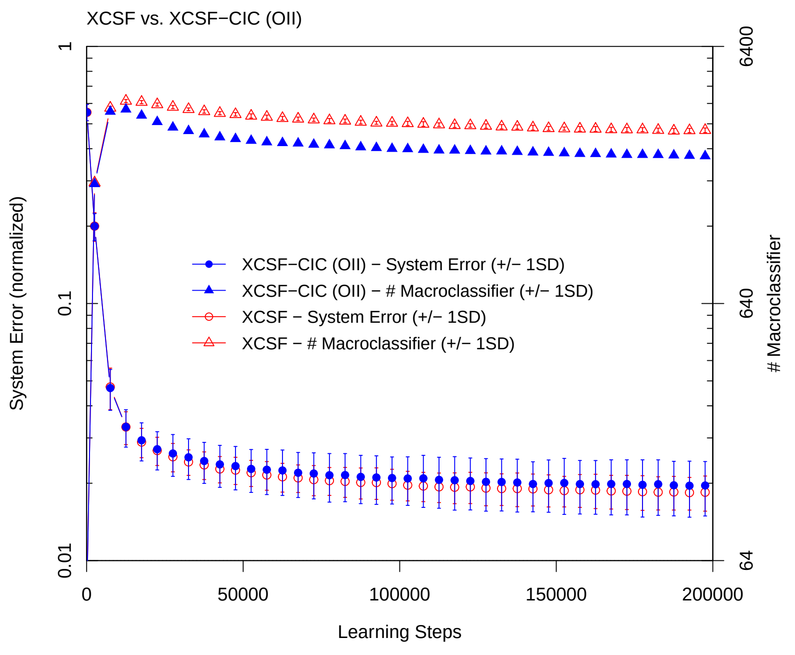


(A) XCSF-CIC with OII on 6D f_1

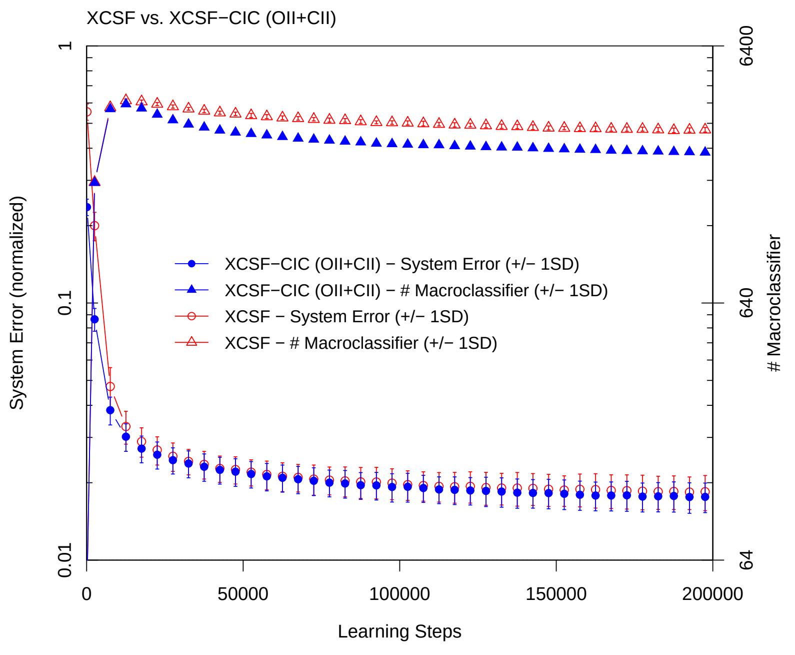


(B) XCSF-CIC with OII+CII on 6D f_1

FIGURE 8.2.: Learning curves of XCSF-CIC applying OII + CII and XCSF on 6D RMS-like function (f_1)



(A) XCSF-CIC with OII on 3D f_2



(B) XCSF-CIC with OII+CII on 3D f_2

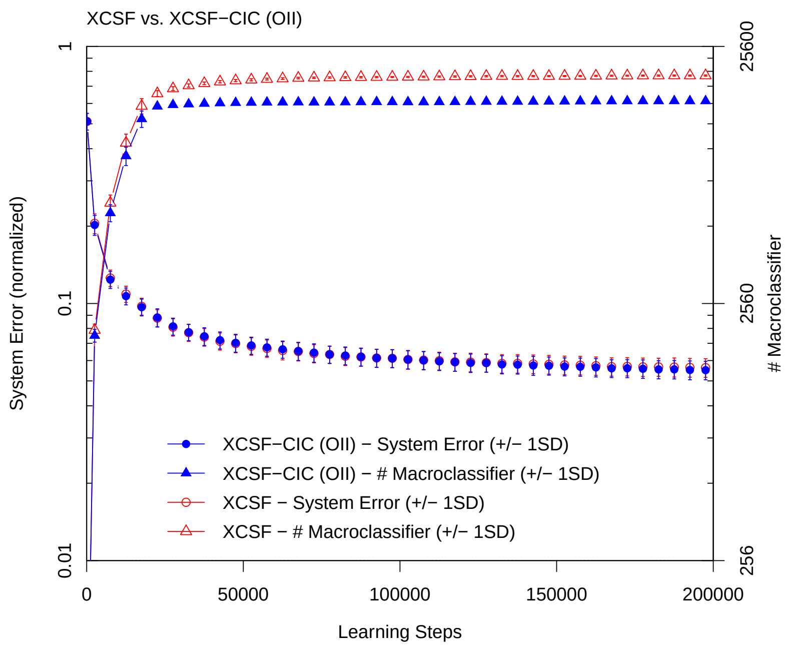
FIGURE 8.3.: Learning curves of XCSF-CIC applying OII + CII and XCSF on 3D Styblinski-Tang function (f_2)

standard deviation. All reported improvements revealed to be statistically highly significant.

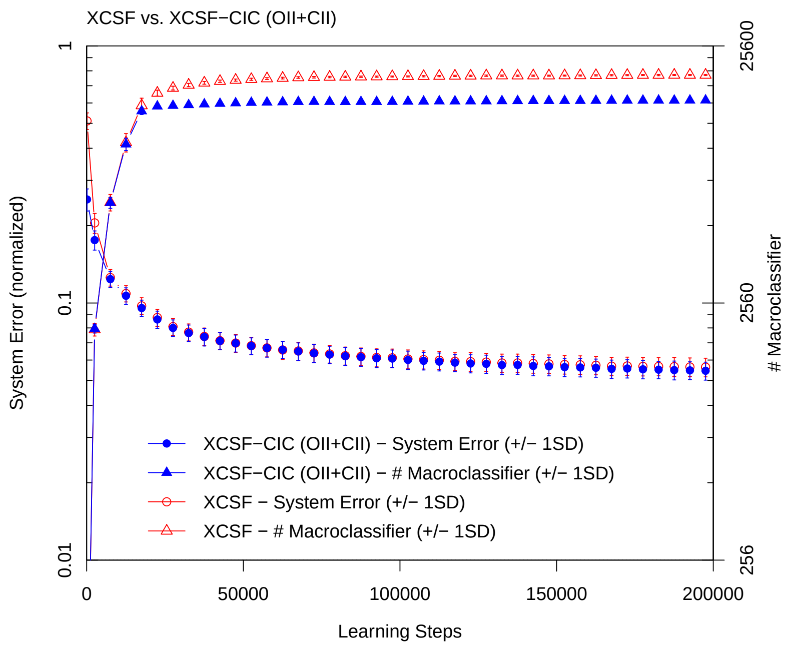
Results on the Highly Multimodal Eggholder Function f_3

For the Eggholder function f_3, whose main complexity lies in the high degree of curvature, again improvements in all figures of merit can be noted when having a closer look to Figures 8.5a and 8.5b plotting the learning curves.

The system prediction error decreases by 2.35% for OII alone and by 11.21% for the combined OII+CII case. Regarding the average population size, distinct reductions have been observed again. 14.77% ($\approx$ 437 classifiers) when OII is used individually, and 14.02% ($\approx$ 415 classifiers) when combined with the CII strategy. As before, all reported improvements have been confirmed to be highly statistically significant (i.e., calculated p-values are all $< \alpha = 0.01$).

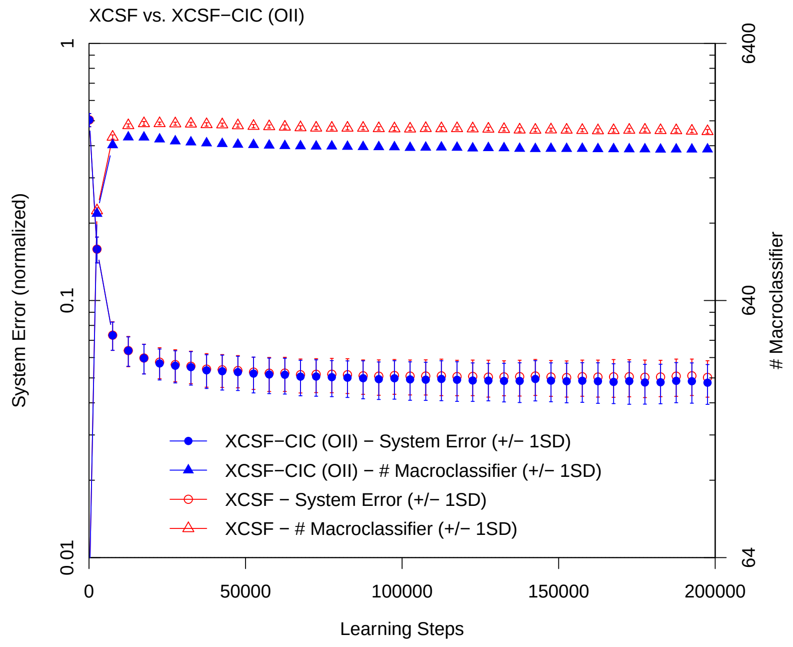


(A) XCSF-CIC with OII on 6D f_2

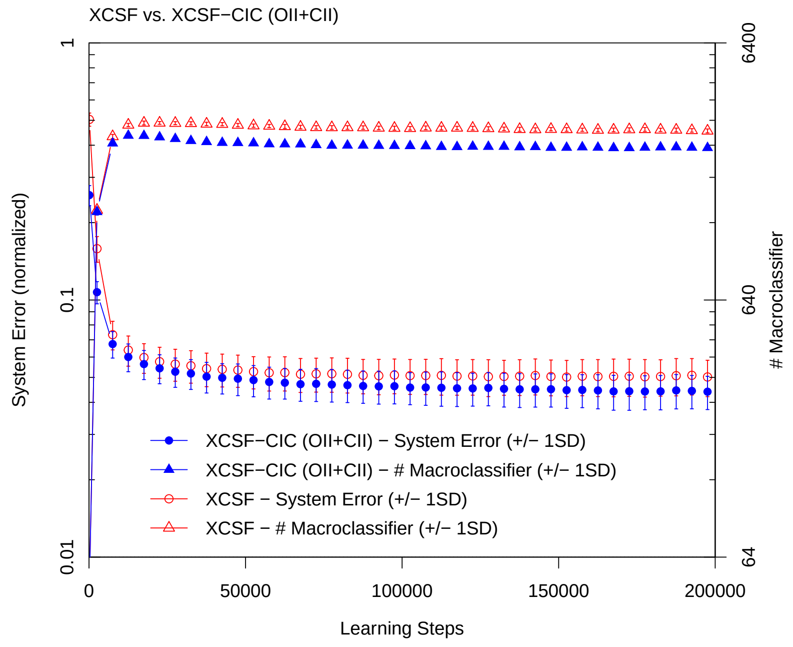


(B) XCSF-CIC with OII+CII on 6D f_2

FIGURE 8.4.: Learning curves of XCSF-CIC applying OII + CII and XCSF on 6D Styblinski-Tang function (f_2)



(A) XCSF-CIC with OII on f_3



(B) XCSF-CIC with OII+CII on f_3

FIGURE 8.5.: Learning curves of XCSF-CIC applying OII and CII on 2D Eggholder function (f_3).

8.4. Discussion

The reported results reveal that the introduced interpolation-based offspring initialization of the classifier parameters $cl.F$ and $cl.exp$ results in significant reductions of the average number of classifiers within the rule base $[P]$. This observation supports the hypothesis as stated at the beginning of this chapter. Namely, that by taking the experience and fitness estimates of surrounding classifiers within a common environmental niche into account, the deletion mechanism is enabled to more effectively remove poor quality offspring rules just created by the GA. This in turn results in an effective reduction of rules physically stored in $[P]$. Such a reduction of the overall population size is clearly desirable for efficiency reasons. Since XCS needs to scan any rule for matching the current situation vector σ_t, the computational effort scales linearly with the current population size. Furthermore, reducing the amount of transient rules introduced by the GA yields more room for innovative rules to let their numerosities increase. Eventually this leads to higher fitness shares received during the reinforcement steps.

However, for the test functions f_1 and the 3-dimensional f_2, slightly increased measures for the approximation error have been observed. Having a look at Figure 8.2a again, it can be noted that the main differences between the error curves of XCSF (red) and XCSF-CIC (blue) occur during the first $50k$ learning steps. Afterward, both curves converge. XCSF is able to very quickly reach the targeted error level of $\epsilon_0 = 0.01$ (normalized). Whenever this threshold is undershot, the evolutionary pressure toward accuracy hardly applies anymore. One possible reason for the slightly increased approximation errors might be that this circumstance hinders the interpolation from calculating proper initial values for the offspring classifiers. This might stall the learning progress instead of accelerating it. For the same observations made with $f2$, however, this rationale does not apply and so far no explanations have been found. However, a more detailed investigation of this aspect would require a close inspection of a number of classifiers to be identified by hand what involves numerous experiments of carefully selected representative configurations. The corresponding effort would have gone beyond the scope of this thesis and is thus postponed to future research endeavors in this regard.

In spite of this missing rationale, one solution to this particular limitation has already been presented in this chapter. By means of a combination of the previously introduced CII strategy and the just developed OII scheme, the deficiency of increasing approximation errors can be completely eradicated for the function approximation scenarios considered in this chapter. Therefore, at this particular state of the OII strategy, it is recommended to always use it in combination with the interpolation-based covering scheme CII. With this combined utilization the observable slight increases regarding the approximation errors can be prevented but it can still be profited from the significantly reduced population sizes.

The applied mindset of restricting the genetic material which is passed to the offspring to that of the direct parents is absolutely legitimate from the genetics perspective. However, it is yet hypothesized that the incorporation of more than the genetic material provided by the immediate parents yields beneficial effects on the learning efficiency. This is exactly what future work on the OII strategy is supposed to investigate more deeply. By means of interpolating the initial values for the offspring classifiers' parameters on the basis of all classifiers residing in the same environmental niche, the *social surrounding* of the newly created offspring is taken into account. Instead of incorporating the entire niche, another means to increase the gene pool is imaginable. In order to extend the set of "high-fitness" sampling points for the interpolation of the offsprings' initial parameter values, a higher number of selection tournaments during GA application could be conducted.

Another limitation of the current state of the OII can be identified in the way the sampling points are extracted from the classifiers in $[A]$. As of yet, neither the generality (i.e., the volumes) of the classifiers is taken into account, nor is their relative quality. The latter aspect can be straightforwardly approached by simply conducting a quality filter similar as the one defined in Chapter 5. However, the quality threshold should not be set to high in order to guarantee a sufficient number of sampling points for the interpolation process. One way to self-adapt such a threshold would be to consider the current average over the action set size estimates $cl.as$ of all classifiers in the population. The same indicator could be used to approach a self-adaptive configuration of the N_w hyperparameter, which determines how many sampling points are considered during a local interpolation. The formerly mentioned limitation that the actual generality of the classifiers is not taken into account yet, is expected to bear larger potential for improvements, though. Overgeneralized classifiers might not reflect the situation of the actually relevant problem space niche properly, since they can be expected to take part in further action sets. Therefore, the information of the condition volumes can be immediately used to design a purposeful filter. Furthermore, the impact of overgeneral classifiers within the interpolation should be decreased for application with OII, since more general classifiers are likely to be part of more action sets and, thus, get their experiences increased more frequently. This might bias the interpolations toward too high experience initializations for newly created offspring rules.

Further room for improving the novel interpolation-based offspring initialization scheme realized by the OII strategy is expected by enlarging the vector of classifier parameters subject to interpolation: As already outlined in Section 6.4, the rationale for interpolation of the $cl_{off}.C$ condition volumes (or generalities) and the geometric orientations also applies here. However, in order to not entirely ignore the condition optimization result obtained in the course of evolution over previous generations, a third offspring classifier with fully interpolated learning parameters (including the condition structure) should be introduced along with the ordinary created offspring.

Furthermore, having a look at the formula for the initialization of an offspring's fitness value

$$cl^1_{off}.F = cl^2_{off}.F = \frac{(cl^1_{par}.F + cl^2_{par}.F)}{2} \cdot F_{reduction} \tag{8.2}$$

and having in mind that the offsprings' numerosity values are initialized with 1, whereas the parents usually are macroclassifiers having a value $num > 1$, another point for deeper investigation appears. Recalling the fitness update formula as introduced in Section 2.4, it can be seen that higher numerosities lead to higher accuracy (and consequently fitness) shares. By not correcting the fitness inherited from the parents by the offspring's numerosities, an overestimate or underestimate regarding the fitness values might appear. This needs then to be corrected by the RL component of XCS over time, which lowers the learning and sample efficiency. The reduction discount $F_{reduction}$ is typically set to 0.1 and changed rarely. A value of 0.1 sort of implies a fitness correction for an assumed parent's average numerosity of 10. Regardless of which value is set for $F_{reduction}$, since it is a hyperparameter set at design time, the aforementioned correction is static over the entire learning process. The eradication of this issue can be accomplished by the OII interpolation strategy by also interpolating $cl_{off}.num$. Furthermore, the assumption of equal influence of both parents for the initial values would be relaxed by determining the weights using distance-weighted interpolation technique as introduced in Chapter 2.

The objective of this thesis is to devise first principles of interpolation-based classifier systems by fathoming the fundamental potentials of integrating interpolation into various sites of XCS's algorithmic structure. Thus, a holistic investigation of all remaining aspects and current limitations unfortunately goes beyond the scope of this work and thus needs to be left for future work.

As a last thought regarding the just introduced interpolation-based offspring initialization strategy, OII can also be understood as follows: By means of involving learning parameters in the genetic recombination process, the involved steady-state niche GA can be regarded as being *Lamarckian*. That is, not only the "immutable" genetic material (i.e., condition $cl.C$ and the action $cl.a$) of the parent classifiers is passed to their created offspring, but also "traits" that are learned over time – in this case the learning parameters $cl.p$ or $cl.\vec{w}$, $cl.\epsilon$ and $cl.F$. To a certain extent the GA conventionally applied in XCS already shows this characteristic. However, with the newly introduced OII strategy, this behavior is more strongly enforced.

8.5. Related Work

8.5.1. Offspring Initialization Schemes

In [KB07] Kovacs and Bull face the question of how to initialize and delete classifiers most appropriately. They reviewed the most common and established initialization and corresponding discounting schemes as well as deletion vote mechanisms found in the literature at that time. Three initialization schemes, three deletion vote calculations and further three fitness discounting approaches have been evaluated on the well-known 6-multiplexer function (a binary problem). Results demonstrate that the most established methods as used in reference implementations such as Butz's XCSJava (described algorithmically in [BW02]) appear superior in reaching a higher percentage of the optimal population denoted $\%[O]$ as well as yielding smaller population sizes. In this thesis, the best performing techniques are adopted as described in Chapter 2.4. Both the OII and the CII methodologies are related in the sense of modifying the initialization of classifiers during rule discovery. It substantiates the need for further research in this direction, since this is the only (short) paper which could be found in the conducted literature review and which is directly concerned with this topic.

8.5.2. Extensions to XCS's Rule Discovery

In their work reported in [SB12; Sta14], Stalph and Butz introduced the concept of *guided evolution*. Based on the *covariance matrix adaptation* method, a novel mutation operator is presented that aims at purposefully guiding the mutations of classifier conditions to yield the most suitable structures. Therefore, each classifier is extended to maintain a set of past input instances it has matched. On the basis of an *accuracy-like quality-weighted* covariance matrix, the guided mutation operator is able to identify dimensions with lower, or else higher impact. In consequence the amount of mutation for the respective condition part can be decreased or increased. Since it is deemed a heuristic, it is interleaved with standard mutation while both having an equal chance. Presented results show that guided evolution leads to a highly increased learning speed and facilitates XCSF to approximate more complex and higher-dimensional functions. In contrast to the modified offspring initialization technique introduced in this chapter, guided evolution works on each offspring classifier individually. Also, only the condition structure is affected, whereas OII interpolates the parts of the offspring classifiers' parameters.

In [FPS10] Fredivianus et al. proposed a novel rule discovery mechanism called *rule combining*. The GA is removed from XCS and the initial generalization probability ($P_{\#}$ in the binary case, and r_0 in the real-valued case [FKS12]) is set to zero, resulting in fully specified classifiers after covering. The rule combination routine is activated periodically and exerts strong generalization pressure to the system. It can also be interpreted as compaction technique, since more specific classifiers that satisfy certain criteria are deleted in favor of a newly constructed general one. The initial values for the combined classifiers' parameters $cl.F$ and $cl.\epsilon$ are set according to equations that are "derived experimentally" [FPS10]. These equations attempt to approximate a value similar as when updated iteratively by the reinforcement component. The payoff prediction values are initialized by the *numerosity-weighted* average of the combined or else subsumed classifiers. Experience $cl.exp$ and numerosity $cl.num$ are accumulated over all classifiers subsumed by the combined one. This approach has been evaluated on the multiplexer problem for both binary and real-valued inputs. It is found to yield superior performance in terms of smaller population size and increased accuracy. Since this modified discovery stresses no specification to the rule base anymore, it is expected to work inferior to the conventional XCS (with a steady-state niche GA activated) on multi-modal problems with a high degree of heterogeneity. These are problem spaces where a certain number of highly specified rules are needed to capture the underlying complexity.

The expert knowledge-guided discovery mechanisms of Urbanowicz et al. [UGM12a] as already mentioned in a related works section before also represent a methodology for guiding the mutation operator in a nearly analogous manner as for covering. Furthermore, the same authors apply *attribute feedback* to the crossover and mutation operators as described in [UGM12b; UM15; Urb+18]. To sum up, both extensions bias the probabilities of mutation and recombination and thus only influence the conditions. The initialization of the other offspring classifiers' parameters is not directly affected, which constitutes the main distinguishing aspect to the work presented here.

8.6. Chapter Summary

As the second mechanism of XCS's discovery component, the steady-state niche GA was subject of enhancement in this chapter. Akin to the previously presented CII strategy, here newly created offspring classifiers were initialized by means of interpolation. It was shown how surrounding classifiers that share a common environmental niche can be taken into consideration for setting the initial values of $cl.F$ and $cl.exp$ of GA-produced offspring rules. The application at this particular algorithmic part of XCS is intended to eradicate the shortcoming of the interpolation-based covering operation which mainly acts at early stages of learning. Results obtained from an empirical study revealed that this methodology leads to strongly decreased average population sizes compared to standard XCSF, but at the cost of slightly increased system errors. The latter negative result has been overcome by combining the strengths of CII and OII, what eventually yields superior performance regarding both figures of merit. Persisting limitations and further ideas to enhance this first attempt of an interpolation-based GA in XCS were discussed as directions for future work. Related attempts to improve the rule discovery mechanisms of XCS were outlined at the end of this chapter. As of yet, interpolation has been integrated into XCS's performance component and its rule discovery mechanisms. In the next chapter, a last methodology that integrates interpolation into the reward prediction calculation step is presented. This technique eventually directly affects (1) the reinforcement of a classifier's prediction parameter $cl.p$ (or prediction model $cl.p(\vec{x})$), and, (2) the way of mixing classifiers to obtain the collective system prediction.

Chapter 9.

Interpolation-Based Prediction Modeling

As a last strategy to counter knowledge gaps and therefore increase both the robustness and learning efficiency of SLAS, the interpolation-based prediction computation is introduced in this chapter. Each rule is extended to maintain a small window of already seen experiences, i.e., situation vectors with their corresponding rewards. These sort of experience memories are then used to transductively infer a prediction for new situations. Thus, the actual inference or prediction step of XCS, here XCSF, is replaced by a transductive, i.e., interpolation-based means. Additionally, the conventional way of classifier mixing is subject to modifications. Instead of letting each matching classifier calculate its prediction vote individually and calculating a fitness-weighted sum of all predictions afterwards, with the new *Interpolated Prediction Integration* (IPI) to be introduced in this chapter the union of all experiences collected by the matching rules is used as basis for the interpolation.

Before diving into the details of the novel IPI strategy, some thoughts will be spent on the traditional way of prediction modeling in XCS(F). The interpolation-based prediction modeling approach is then introduced and complemented with a short estimation of its complexity. Subsequently, the obtained results from conducted empirical studies on a variety of test functions to be approximated by XCSF are reported and assessed in terms of statistical significance. A summarizing discussion of the gained insights, current limitations and aspects of further research follows. As done in the previous parts of this work, the chapter is closed with the appreciation of related work on prediction modeling in XCSF.

9.1. Basic Prediction Modeling

Traditionally, XCS was introduced to let each classifier encode a scalar value for its reward prediction, denoted $cl.p$ in the literature. This implies a constant prediction over the entire subspace of X determined by a classifier's condition $cl.C$. The result is a piece-wise constant approximation that resembles a sort of non-continuous step function. Apparently, this means of "modeling" a classifier's prediction is not appropriate to reconstruct a smooth continuous function surface.

For this reason, Wilson extended his XCS classifier system, so far mainly applied to sequential control problems and classification tasks, toward XCSF in [Wil02]. He approached this problem by computing predictions that are functional depending on the current input vector $\vec{x}$. Thus, the prediction scalar $cl.p$ was replaced by a coefficient vector $\vec{w} \in \mathbb{R}^{n+1}$ which comprises n weights, one for each dimension, as well as one bias (or offset) term denoted w_0 (one could imagine this offset weight as the former prediction scalar $cl.p$). As a first attempt, a linear model was applied by calculating the linear combination of the input $\vec{x}$ and the current weight vector $\vec{w}$. Accordingly, a single classifier's prediction is modeled by a linear approximation as given by:

$$cl.p(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i \tag{9.1}$$

The weight vectors for each classifier $cl.\vec{w}$ are subject to stochastic optimization in the course of learning. Initially, a modified Widrow-Hoff delta rule has been applied [Wil02]. This was later revised by Lanzi et al. in [Lan+06] who investigated various algorithms to find the most appropriate coefficients online – among them *Kalman filter*, *gain adaptation* and RLS. Lanzi et al. also proposed to apply a batch gradient descent algorithm, i.e., *linear least squares* in [Lan+05a]. In summary, each classifier needs to solve its own *model fitting* task based solely on the data it has matched so far.

Figure 9.1 illustrates the idea of having piece-wise constant and piece-wise linear predictions used to locally model an exemplary one-dimensional function $f(x)$.

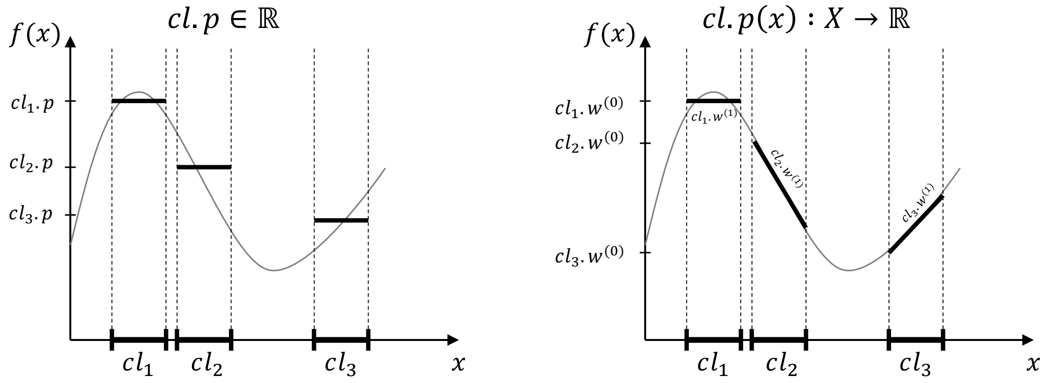


FIGURE 9.1.: A simplified illustration regarding the approximation capabilities of XCSF utilizing traditional scalar prediction (left) and linear model prediction (right)

As can be noted, using computed predictions bears mainly two advantages: (1) The complexity of the underlying function can be captured more accurately. (2) To yield such accurate predictions, classifier conditions evolve toward the most appropriate size in the course of learning.

Of course, more complex models than a first degree polynomial can be used within a single classifier. Lanzi et al. in [Lan+05b] proposed a way to easily extend the abovementioned means of a linear model to quadratic, cubic or any polynomial models of degree k. For example, in order to obtain a quadratic model (bivariate polynomial) of a $n = 2$-dimensional input vector $\vec{x} \in \mathbb{R}^2$, the input needs to be extended as follows:

$$\vec{x}' = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2, x_1^2 x_2, x_1 x_2^2, x_1^2 x_2^2) \tag{9.2}$$

As can be seen the input space is actually extended by adding the polynomial terms. For each of these terms one weight needs to be learned by XCS in an online fashion. An advantage, however, is that the approximation is linear in its weights what allows the application of powerful optimization techniques. Other ways are outlined in [Lan+05b] but are not considered in this thesis due to their inherent drawbacks.

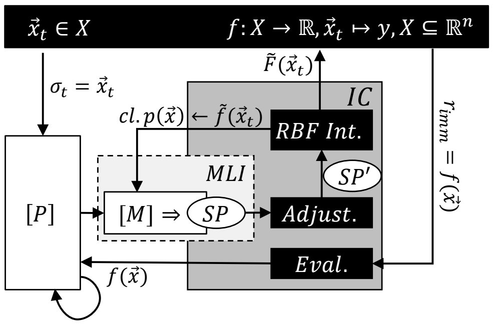
What should be noted at this point is the following: The selection of the polynomial degree to be applied for approximating a certain problem function at hand needs to be felt at the design time. Thus, by selecting the degree k, an a priori determined model bias is injected into the system. With the IPI approach as will be introduced next, one major objective is to free XCS from this sort of model bias.

9.2. Interpolation-Based Prediction Modeling

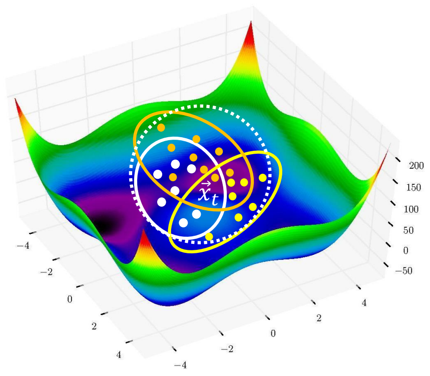
The novel interpolation-based approach to classifier prediction computation is based on the IC as introduced in Chapter 4. The main focus of modifications is set on the structure of individual classifiers as well as the means how a collective decision is felt (classifier mixing). Thus, each classifier needs access to the IC which is provided via the well-defined MLI. It is again focused on XCSF for function approximation to allow for a straight-forward experimental setup in terms of a variety of benchmark functions exhibiting idiosyncratic characteristics. In the remainder, this IC-extended XCSF is denoted XCSF-IC.

In XCSF-IC classifiers are extended to collect a limited number of previously encountered situations (or experiences) and store them in a sort of experience memory. This memory is then used as sampling point set on which the actual interpolation operations are based. Thus, each classifier is now enabled to access the global IC and pass its specific sampling point set over the MLI together with the current situation $\sigma_t = \vec{x}_t$. This is necessary to retain the competition among classifiers in the same match set (or action set in the case of standard XCSR). More precisely, each classifier individually interpolates an its prediction $cl.p(\vec{x})$ which is then compared to the actual function value $f(\vec{x})$ received as immediate reward r_{imm} (illustrated by the arrow from the RBF interpolant pointing to the match set in Fig. 9.2a). Based on the derived prediction accuracy from a classifier's prediction error estimate $cl.\epsilon$, the niche-relative fitness value $cl.F$ can be adjusted in the course of learning. The involved GA picks up these fitness estimates during parent selection which eventually fuels the condition shape optimization over the generations as usual. Assuming a limited experience memory, a beneficial side effect occurs – the classifier system self-controls the density of sampling points collected within individual niches. In niches which capture a very complex part of the underlying problem function surface (e.g., high curvature or ruggedness) the generality of covering classifiers usually appears to be smaller. Accordingly, the density of sampling points in that particular regions of the problem space increases. This effect could be even reinforced if the classifiers would be endowed with the capability to self-adaptively choose the memory size (cf. Sect. 9.4 for further discussion).

Although each classifier is now enabled to compute its individual prediction by means of using the interpolation facilities of the IC, another modification to the conventional XCSF is introduced in the following. Figure 9.2a depicts a schematic illustration of XCSF adapted to incorporate interpolation-based system predictions by means of using the IC. It focuses on the interplay between the IC and XCSF's conventional data flow for building an overall system prediction, i.e., a collective decision of all classifiers matching the current situation (classifier mixing).



(A) Schematic illustration of XCSF-IC using the IC to incorporate interpolation-based system predictions.



(B) Classifier mixing by forming the union (dotted circle) of experiences (dots) from matching classifiers (ellipses). Image adapted from [Wik]

FIGURE 9.2.: XCSF-IC working principle. Left: Algorithmic structure and data flow. Right: Intuition of novel classifier mixing scheme.

As the above figure shows, the utilization of the IC takes place after the formation of the match set. First, the union of all matching classifiers' experience memories is formed (see Fig. 9.2b). Next, this collective sampling point set SP is sent to the adjustment component of the IC. It is responsible for detecting and removing duplicates which can be introduced during offspring generation as described below. In noisy domains where the reward is non-deterministic, a preprocessing step is needed to account for that. Duplicated sampling points are simply averaged in order to center the noisy function values. Duplicates are identified by comparing their double values in a range of a very small ϵ. The filtered sampling point set SP' is then sent to the interpolant which is depicted as an RBF interpolation method. Throughout this chapter it is focused on RBF interpolation using the parameter free *thin plate spline* basis function. A comprehensive preliminary study revealed that RBF interpolation strongly outperforms alternative methods including MSM, IDW, NaNe and NeNe for the task of function approximation [Men17]. The interpolation result $\tilde{F}(\vec{x}_t)$, with the current input vector as query point ($\vec{x}_q = \vec{x}_t = \sigma_t$) and based on SP' (collective system prediction), eventually is used as the system output. Subsequently, and conforming to the system model described in Section 3.2, the actual function value $f(\vec{x}_t)$ is delivered as immediate reward r_{imm} to XCSF. The reinforcement operations are applied as usual.

In summary, the implementation of this interpolation-based classifier prediction technique essentially requires the following steps:

1. Add a limited set of sampling points $cl.SP := \{s_i\}$, where $s_i := (\vec{x}_t, r^t_{imm})$ as experience memory to each classifier cl and continually collect samples s_i during the classifier's lifetime. Thus, the experiences constitute situation-reward tuples of the observed input vectors $\vec{x}_t$ and the actual function value $f(\vec{x}_t)$ received as immediate reward r^t_{imm} at time t.

2. Each time a classifier cl_j matches the current situation $\vec{x}_t$, use interpolation to compute the individual classifier's prediction $\tilde{f}(\vec{x}_t) = cl_j.p(\vec{x}_t)$ by passing its set of sampling points $cl_j.SP$ via the MLI to the IC (repeat for all $cl_j \in [M]$).

3. The passed sampling point sets SP have to be adjusted to not contain duplicated sampling points. This is managed by the IC's *adjustment component*. Its purpose for this particular IPI strategy is to find and average duplicated sampling points. For the results reported in this chapter, this task was accomplished by calculating an equally-weighted mean of the different function values (e.g., due to noisy functions).

4. Build the union of all sampling point sets SP_j from $cl_j \in [M]$, i.e., $\bigcup_j cl_j.SP$, and pass it to the IC for further processing.

5. Use the interpolation result calculated on the basis of the union, denoted $\tilde{F}(\vec{x}_t)$, as system prediction for the current situation (or query point) $\sigma_t = \vec{x}_t$.

6. Alternatively: Combine all interpolated classifier predictions $cl_j.p(\vec{x}_t), \forall cl \in [M]$ by means of the conventional fitness-weighted sum calculation approach to classifier mixing.

For the results reported in this chapter, the former approach (step 4 and 5) to classifier mixing has been utilized. An insight obtained during preliminary experiments is that this approach yields superior performance compared to the conventional mixing procedure by means of calculating a prediction array. For the latter one, it could nonetheless be found that in comparison to the conventional model fitting approach, using linear, quadratic and cubic polynomials, the IPI approach using step 6 still yields improved performance [Men17]. In [DB07], Drugowitsch and Barry already pointed out that the conventional fitness-based mixing approach can be suboptimal in certain cases. This insight lead to the decision to further simplify the actual system prediction formation. The outlined IPI mixing strategy allows to fully omit the prediction array and, thus, an actual mixing based on weights to be determined carefully on the fly. Nevertheless, the niche-relative accuracy calculation for each classifier in $[M]$ is deliberately retained. As already discussed before, it implements a fitness sharing mechanism into XCS, which was found to be beneficial in various studies (cf. e.g., [BGL07]).

The advantage of XCSF and XCSF-IC in particular is that it provides a unique tandem of global system prediction formation and niche-relative competition between accurately predicting local models (i.e., classifiers). The involved evolutionary discovery component in form of a niche GA is supposed to support the evolution of optimized problem subspaces (i.e., environmental niches) that model the corresponding parts of the underlying problem most accurately. In XCSF-IC, each classifier attempts to achieve a best-possible interpolation of the input subspace $cl.C$ for which it is responsible. Therefore, a near-optimal generality (or else specificity) of the condition is subject to optimization. The condition size in turn immediately affects the resulting density of the stored sampling points. This self-adaptive density aspect is expected to improve the capability of capturing the underlying function complexity through individual rules, as will be discussed in more detail below.

9.2.1. Experience Memory Management

In order to take into account the online learning nature of XCSF, it should be kept in mind that the underlying functions may change over time. This might happen either due to changing input data distributions (virtual/covariate drift). Or, even more challenging, through real concept drifts affecting the actual surface of the underlying function to be approximated. In terms of the introduced KG notion in the context of SLAS, unforeseen or rarely occurring situations might appear that should be reflected in the experience memories to further take advantage of them in the future. Therefore, the experience memory of matching classifiers should be continually kept up-to-date in the course of runtime. The size of each classifier's sampling point set $cl.SP$ is limited to a predefined number N_{sp} of experiences (or sampling points s_i), i.e., $|SP| \leq N_{sp}$ must hold. Recalling the inherent generalization mechanism of XCS, one noticeable aspect can be leveraged for the IPI strategy: Due to the self-optimization of the condition structures through the GA, XCSF manages the density wit regard to the sampling points collected in SP by itself. Each classifier condition $cl.C$ represents a certain subspace of the input space X. Thus, the entire problem space is partitioned into a collective of partially overlapping local models (i.e., classifiers). The employed GA seeks near optimal condition structures by maintaining a maximum level of accuracy at the same time. This results in a global optimization of the problem space partitioning, essentially following the fundamental *divide-and-conquer* principle. Imagining a function with a high degree of curvature in certain parts of its domain, XCSF will automatically assign more resources, i.e., classifiers, to this environmental niche since more specialized models are needed to capture the complexity. Thus, in order to achieve high predictive accuracy, the classifiers have to shrink to a certain level (i.e., they will evolve to be more specialized). But still each of these classifiers shares the same sampling point restriction $|cl.SP| \leq N_{sp}$ with its more general counterparts. Accordingly, the density of collected sampling points in that region increases, which in turn is assumed to result in more accurate interpolations. On the other hand, it is hypothesized that the population-wide average generality of classifiers will increase due to the inherent ability (and claimed requirement) of interpolation techniques to more strictly reconstruct complex parts of the underlying target function. Figure 9.3 illustrates this assumption on a simplified one-dimensional target function as before.

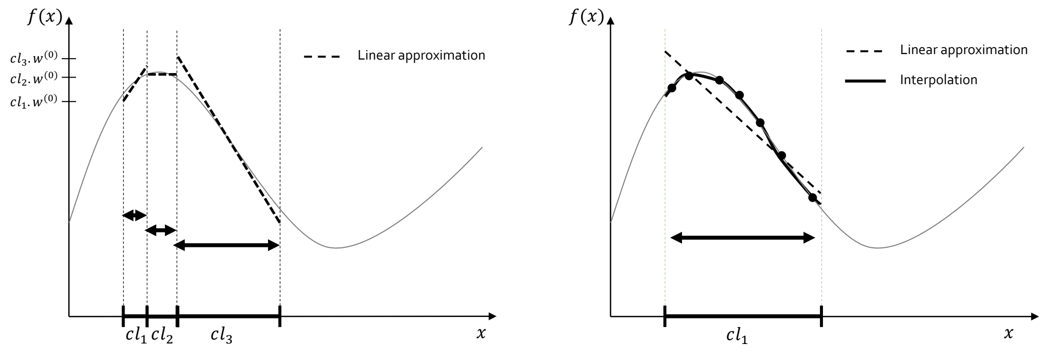


FIGURE 9.3.: Simplified illustration of the expected impact of IPI on the classifier generality. Left: Linear model approximation needs three specific classifiers. Right: IPI captures the same part of the function but with a single more general classifier and with less deviation from the ground truth compared to a single linear model.

As a result of both hypothesized effects, their combination is deemed to support the generalization/specialization capability of XCS in general and XCSF in particular. Section 9.3 will shed light on this.

Regarding the aspect of experience memory adjustment, two approaches are outlined: (1) The utilization of a FIFO queue. (2) A more sophisticated strategy which purposefully works toward a uniform distribution of collected experiences over the specific subspace a classifier covers. Although not investigated further so far, the latter concept can be realized based on an approximate entropy measure relying on nearest neighbors. For the scope of this work, however, the former approach of using a FIFO queue is realized. This resembles the approach proposed by Lanzi et al. in [Lan+05b; Lan+05a] who investigated the use of non-incremental *linear least squares* approach for determining optimal model coefficients $\vec{w}$. Further investigations of alternative sampling point replacement strategies are therefore left for future work. Each time a classifier cl_j matches the current input $\vec{x}_t$, it creates a new sampling point s_i as soon as the reward (the actual function value $f(\vec{x}_t)$) is received. Afterward, the classifier cl_j simply replaces the oldest $s_i \in cl_j.SP$ with the just created experience $(\vec{x}_t, f(\vec{x}_t))$. A novel data structure has been implemented that combines the advantages of both a queue and a set. That is, this data structure guarantees that sampling points can not be added twice – a necessary requirement to guarantee solvability of the RBF interpolation's linear equation system (cf. Sect. 2.3). Additionally, the efficient access operations of a FIFO queue are retained.

As the weight vectors or the simple prediction scalar are part of the genetic material which is passed to offspring rules, also the collected experiences should not get lost during the recombination process. As usual, the GA selects two parental classifiers, recombines them, and mutates the offspring rules individually by chance. When using the IPI strategy, the offspring rules inherit the sampling points from their parents as follows: Each cl_{off} firstly fills up its memory with the sampling points $s_i \in cl_{par}.SP$ from its immediate parent (technically, the parent from which it was deeply copied). But only those which fall inside the mutated offspring classifier's condition (input subspace). The remaining capacity is subsequently filled with matching sampling points from the other parent.

9.2.2. Complexity Considerations

Using IPI requires additional memory: A set of N_{sp} sampling points comprising $d+1$ scalar values each, i.e., the d-dimensional vector plus the scalar target function value $f(\vec{x})$ (assuming one-dimensional outputs). These experience sets are incrementally built up for each of the at maximum N classifiers in $[P]$. However, if it is possible for the generalization pressure to apply, the actual number of physically stored classifiers $cl \in [P]$ is typically far below N. Accordingly, the proposed interpolation-based prediction approach demands additional memory of $O(N_{sp}(d+1)N)$ scalar

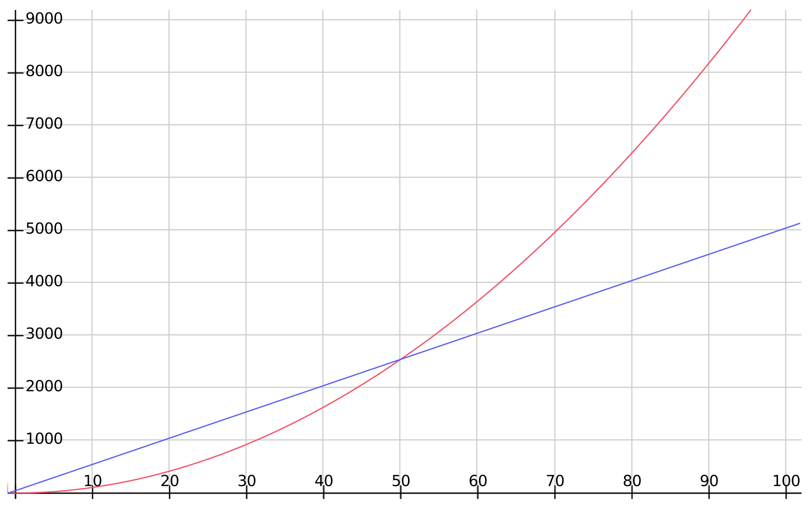


FIGURE 9.4.: Theoretical comparison of the memory requirements per classifier (y-axis) as a function of the dimensionality d (x-axis) of IPI with $N_{sp} = 50$ (blue line) vs. conventional linear model fitting using RLS (red line)

values in the worst case. In contrast, RLS only needs an $d \times d$ gain matrix plus a vector of $d+1$ coefficients for each classifier in the rule base. This results in a worst-case space complexity of $O((d^2 + d + 1)N)$.[1] Thus, for IPI the memory demand increases linearly in the number of dimensions d and the size of the experience memory N_{sp}. The conventional model fitting approach using RLS as coefficient optimization technique scales quadratic in the number of dimension d only. Please note that the maximum number of classifiers N was considered to be equal for both versions and is thus handled as a constant in this theoretical comparison. Figure 9.4 illustrates this relationship. As can be noted, as soon as the number of dimensions d surpasses the number of experiences N_{sp} to be stored per classifier (in the depicted case 50), the memory requirements of the linear model fitting approach using RLS overtakes the newly introduced IPI strategy. Whenever a higher-order polynomial approximation is used, the input vector $\vec{x}$ needs to be preprocessed as discussed before. This, in turn, also demands for more coefficients which also affects the gain matrix as stored for RLS. Accordingly, the intersection point appears earlier. Despite the theoretically increasing memory demand for IPI up to a dimensionality $d < N_{sp}$, the additional memory actually required turns out to be reasonably small in practice as the following example suggests: Considering an approximation problem with an $d = 6$ dimensional input space, a typical population size of $N = 6400$, and storage of the last $N_{sp} = 50$ samples in each classifier. Using C++ programming language and the double data type for storing the scalars, eventually less than $50 \cdot (6 + 1) \cdot 6400 \cdot 8$ Bytes ($\approx$ 17 MB in the worst case) of additional memory is needed. In view of today's microcomputers this should not constitute an issue in the vast majority of cases.

[1]Here d denotes the dimensionality of the input space X.

In terms of time complexity, for RLS $O(n^2)$ demanding matrix multiplications constitute the most time consuming task according to [Lan+05a]. In contrast, for the IPI strategy utilizing the computationally expensive RBF technique $O(n^3)$ demanding matrix inversions are required. However, there exist more efficient ways to solve the linear system of equations having the size $n \times n$ which only require $O(n^2)$ computation time. For instance, Skala in [Ska13] proposes an incremental approach that remembers previous inverse matrices and only replaces the necessary rows and columns. This approach is perfectly utilizable for the IPI technique, since at each timestep at most one experience is replaced by a newer one.

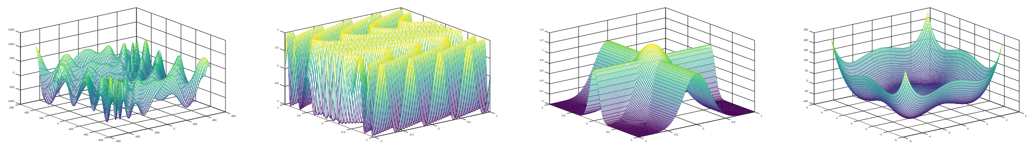
9.3. Evaluation

In order to confirm the hypothesized effects of the novel IPI strategy, empirical experiments have been conducted. As before, the regression problem of online function approximation has been chosen for validation purposes. Therefore, XCSF-IC is compared against standard XCSF on four carefully selected benchmark functions to be approximated. After revealing the general potential of IPI, results obtained from an additional experiment to fathom its sustainability under noisy conditions are reported.

9.3.1. Experimental Setup

The results presented in this chapter have been obtained by following the subsequent experimental setup: Each individual experiment has been conducted with ten i.i.d. repetitions, using differently seeded random generators each. The maximum number of learning problems is $200k$. Longer evaluations did not yield any noticeable learning progress thereafter. Each learning problem corresponds to a uniformly sampled vector $\vec{x}$ of the function's domain which is sent to XCSF for prediction of the function value $f(\vec{x})$. Following the suggestions from the literature, especially [Sta14] and [Sta+12b], XCSF is configured as follows: $N = 6400, \alpha = 1, \beta = 0.1, \delta = 0.1, \epsilon_0 = 0.01, \nu = 5, \theta_{GA} = 50, \theta_{sub} = 20, \theta_{del} = 20, \mu = 0.04, \chi = 1.0, F_{ini} = 0.01, \epsilon_{ini} = 0.0, F_{reduction} = 0.1, \epsilon_{reduction} = 1.0$. The general explicit hyperellipsoidal representation [BLW08] for condition encoding with self-determined mutation size as well as tournament selection within the GA and a tournament size of $\tau = 0.4$ is applied. GA subsumption is active. The allowable ranges for random selection of the initial spread values (configured by setting r_0 with an additional minimum value) for newly created hyperellipsoidal conditions are set as indicated for each experiment individually in Table 9.1.

The results table presents the mean values over the experiment repetitions accompanied with their standard deviation ($\pm$1SD). The novel IPI approach is compared against the, to the best of the author's knowledge, so far mostly investigated n-th



(A) f_1 'Eggholder' [JY13] (B) f_2 'Sine-in-Sine' [SB10a] (C) f_3 'Cross ridge' [Sta+12b] (D) f_4 'Styblinski-Tang' [ST90]

FIGURE 9.5.: Surface plots of the considered test functions to be approximated for input space dimensionality $d = 2$

degree polynomial approximation for classifier prediction combined with the RLS optimization technique. More precisely, XCSF-IC with RBF interpolation using the *thin plate spline* basis function, as well as standard XCSF with linear, quadratic and cubic approximation (i.e., $n \in \{1, 2, 3\}$) are subject to evaluation. To judge on the statistical significance, the non-parametric *Friedman's Rank Sum test* is conducted to decide whether there are significant differences among the several methods. A non-parametric hypothesis test is selected since the requirements for the application of parametric *Analysis of Variance* (ANOVA) (normality and homoscedasticity) could not be confirmed. If the Friedman test indicates Null-Hypothesis rejection, i.e., significant differences among the methods, pairwise *Wilcoxon Signed Rank* post-hoc tests are performed in order to determine significance groups showing which methods actually differ significantly from others. The significance level is set to $\alpha = 0.05$.

The following figures of merit are observed: (1) The *mean absolute error* (MAE), also denoted system (prediction) error in the LCS context, between the predicted and the actual function value. (2) The *average number of macroclassifiers* in $[P]$. (3) The population-wide average of *classifier generality*, i.e., the average volume of the hyper-ellipsoids.

Curves depicted in the following plots show the aggregated values of the aforementioned figures of merit. Each point shows an aggregation over the last $8k$ received learning problems (situations). Black point characters always indicate the learning progress of XCSF-IC regarding all figures of merit. For the sake of readability, error bars indicating the standard deviations at each point are omitted. The corresponding tables 9.1 and 9.2 summarize the mean values of the conducted repetitions over the entire experiment length of $200k$ learning steps as well as the corresponding standard deviations ($\pm$1SD).

The potential of the IPI approach is examined on the following four test functions[2] depicted in Figure 9.5. Each function has already been used for XCSF assessment in the literature:

[2]W.l.o.g., the domains of all test functions are normalized to the range $\forall i \; x_i, \in [0, 1]$. The co-domains of f_1 and f_4 are rescaled to $y \in [-1, 1]$.

$$f_1(x_1, x_2) = -(x_2 + 47) \sin\left(\sqrt{\left|\frac{x_1}{2} + (x_2 + 47)\right|}\right) - x_1 \sin(\sqrt{|x_1 - (x_2 + 47)|}),\ -512 \leq x_1, x_2 \leq 512 \tag{9.3}$$

$$f_2(x_1, x_2) = \sin(4\pi(x_1 + \sin(\pi x_2))), 0 \leq x_1, x_2 \leq 1 \tag{9.4}$$

$$f_3(\vec{x}) = \max\{\exp(-10a^2), \exp(-50b^2), 1.25 \exp(-5(a^2 + b^2))\}, \quad a = \frac{1}{\lfloor n/2 \rfloor} \sum_{i=1}^{\lfloor n/2 \rfloor} x_i, b = \frac{1}{\lceil n/2 \rceil} \sum_{i=\lfloor n/2 \rfloor + 1}^{n} x_i,\ 1 \leq x_i \leq 1 \tag{9.5}$$

$$f_4(\vec{x}) = \frac{\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i}{2}, -5 \leq x_i \leq 5 \tag{9.6}$$

As previously observed in Chapter 6, the first benchmark called *Eggholder* function f_1 has been found to be challenging for approximation with XCSF due to its high curvature and non-linearity in both dimensions. The *Sine-in-Sine* function f_2 shares equal characteristics with f_1 and has been used as test function for XCSF assessment in [SB10a; Sta+12b]. However, in contrast to f_1 it shows a higher regularity in terms of repeating structures in both dimensions. Such regularities can be exploited by the GA in XCSF. The so-called *Crossed-Ridge* function f_3 has its complexity originating from a mix of linear and non-linear subspaces forming crossing ridges with a centered single peak as illustrated in Figure 9.5c. It has been subject to investigations regarding XCSF's approximation capability and competitiveness against *Locally Weighted Projection Regression* (LWPR) in [Sta+12b]. In this chapter, the generalized version for $n > 2$ dimensions (here $n = 3$) as proposed by Stalph in [Sta14] is used. Finally, again the *Styblinski-Tang* function f_4 in its 6-dimensional form is taken into consideration. It already turned out to be a really challenging approximation task for XCSF in the previous Chapters 6 and 8. As Figure 9.5d suggests, its 2-dimensional version appears rather simple at the mid-area of its domain. In contrast, at the edges, the function values increase steeply which seems to let XCSF struggle with finding appropriate approximations. In higher dimensions, the severity of this characteristic is assumed to drastically increase.

9.3.2. Results

Table 9.1 summarizes the results for all investigated scenarios and provides insights regarding the statistical significance of the obtained differences between the investigated XCSF configurations. All entries present the means and their standard deviations (±1SD) over the entire learning task of $200k$ learning problems. Bold entries indicate statistically significant superior performance over any other configuration. After each value, the corresponding significance group (A–D) determined by the performed post-hoc tests, with A indicating best performance, is given. Furthermore, the p-values obtained from the conducted Friedman tests are presented for each scenario and each figure of merit.

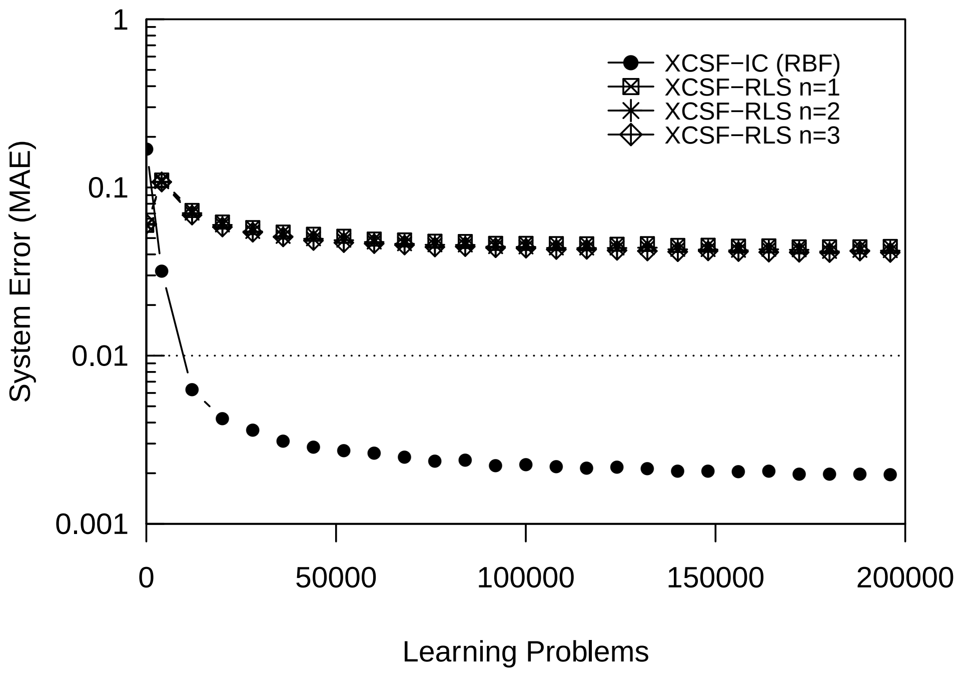
2-dimensional Eggholder function f_1

As can be noted by analyzing Figure 9.6, the RBF-based XCSF-IC variant employing the IPI methodology (denoted by XCSF-IC (RBF) in all plots) strongly outperforms all other XCSF variants which make use of polynomial approximation up to degree $n = 3$. For all three metrics considered, i.e., the average system error (MAE), the average number of macroclassifiers, as well as the average generality of the evolved rules, XCSF-IC has obtained the best results. The Friedman test suggested highly significant differences regarding all figures of merit among the investigated modeling techniques. The pairwise conducted Wilcoxon signed rank post-hoc tests revealed that XCSF-IC's performance is significantly superior in all considered metrics. XCSF-IC allows to approximate the highly non-linear Eggholder function (f_1) with an error level far below the target error $\epsilon_0 = 0.01$. Neither of the other approaches is able to achieve competitive prediction accuracy. In numbers, this corresponds to an overall error reduction of 92.31% with respect to the second best approach (XCSF-RLS with cubic approximation) for the entire experiment.[3] Furthermore, the classifiers' generality notably increases by 159.52% on average. This circumstance is further reflected in the average population size where XCSF-IC needs 13.33% less rules than the strongest opponent (XCSF-RLS with cubic approximation). This observation supports the initially posed hypothesis that interpolation-based predictions allow for more general rules since more complex parts of the underlying functions can be accurately reconstructed by the interpolant (i.e., without injecting model bias as required for the polynomial approach).

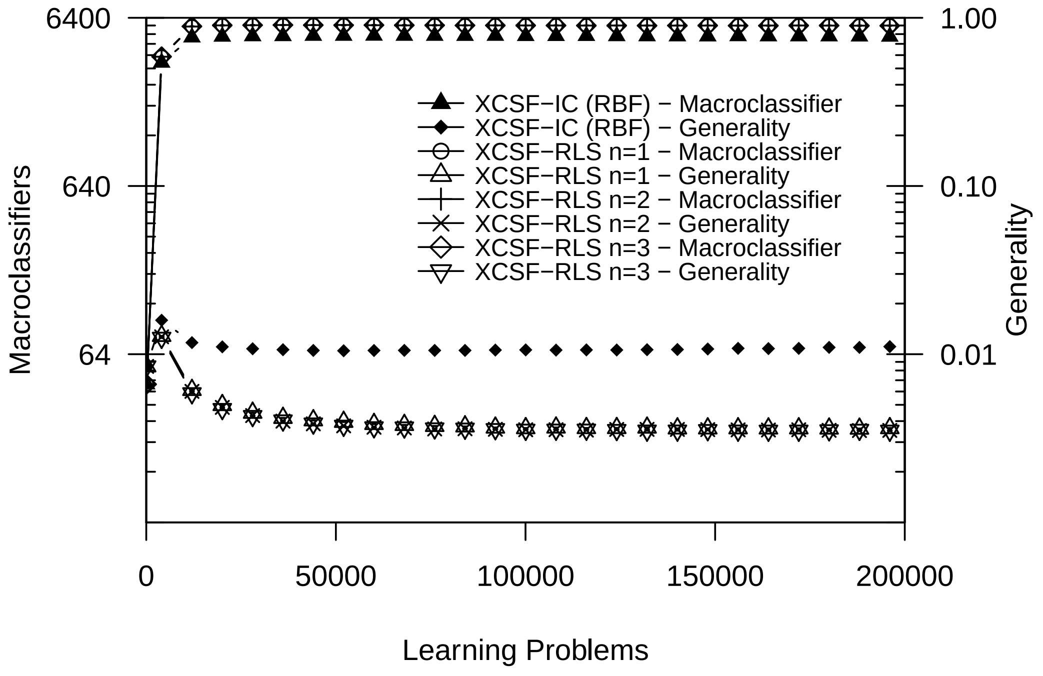
[3]Throughout this chapter, all percentage-wise improvement indications refer to the best alternative configuration of XCSF using RLS for polynomial approximation and are calculated based on the mean values obtained after conducting the entire experiments of length $200k$ learning steps.

TABLE 9.1.: Summary of results for XCSF-IC using RBF-based IPI vs. standard XCSF with RLS optimized n-th degree polynomial approximation for $n \in 1, 2, 3$ on f_1–f_4

f_1 **Eggholder 2D,** $r_0 \in [0.005; 0.1]$	p-values	XCSF-**IC (RBF)**	XCSF-**RLS (n=1)**	XCSF-**RLS (n=2)**	XCSF-**RLS (n=3)**
System Error (MAE)	$3.49 \cdot 10^{-6}$	**0.0037**±0.0002 (A)	0.0516±0.0010 (D)	0.0492±0.0010 (C)	0.0481±0.0001 (B)
Avg. Macroclassifiers	$3.50 \cdot 10^{-4}$	**4924.21**±12.16 (A)	5687.24±6.98 (B)	5683.31±8.25 (B)	5681.84±5.39 (B)
Avg. Generality	$5.88 \cdot 10^{-6}$	**0.0109**±0.0002 (A)	0.0042±0.0001 (B)	0.0041±0.0001 (C)	0.0041±0.0001 (C)
f_2 **Sine-in-Sine 2D,** $r_0 \in [0.005; 1.0]$	p-values	XCSF-IC (RBF)	XCSF-RLS (n=1)	XCSF-RLS (n=2)	XCSF-RLS (n=3)
System Error (MAE)	$4.71 \cdot 10^{-5}$	**0.0065**±0.0016 (A)	0.0842±0.0132 (C)	0.0667±0.0117 (B)	0.0739±0.0270 (C)
Avg. Macroclassifiers	$5.28 \cdot 10^{-5}$	2744.93±49.26 (A)	3072.38±42.01 (C)	2830.90±57.33 (B)	2743.88±73.03 (A)
Avg. Generality	$5.59 \cdot 10^{-3}$	0.0475±0.0174 (B)	0.0671±0.0245 (A)	0.0624±0.0184 (A)	0.0798±0.0491 (A)
f_3 **Cross 3D,** $r_0 \in [0.005; 1.0]$	p-values	XCSF-IC (RBF)	XCSF-RLS (n=1)	XCSF-RLS (n=2)	XCSF-RLS (n=3)
System Error (MAE)	$1.32 \cdot 10^{-5}$	**0.0059**±0.0006 (A)	0.0165±0.0021 (C)	0.0147±0.0019 (B)	0.0149±0.0022 (B)
Avg. Macroclassifiers	$3.49 \cdot 10^{-6}$	5613.67±53.36 (C)	5502.67±65.63 (B)	5411.97±49.82 (A)	5389.92±56.73 (A)
Avg. Generality	$4.15 \cdot 10^{-4}$	0.0804±0.0273 (B)	0.1563±0.0364 (A)	0.1567±0.0374 (A)	0.1580±0.0372 (A)
f_4 **Styblinski-Tang 6D,** $r_0 \in [0.005; 0.5]$	p-values	XCSF-IC (RBF)	XCSF-RLS (n=1)	XCSF-RLS (n=2)	XCSF-RLS (n=3)
System Error (MAE)	$1.38 \cdot 10^{-6}$	**0.0217**±0.0003 (A)	0.0891±0.0003 (D)	0.0529±0.0004 (C)	0.0268 ±0.0010 (B)
Avg. Macroclassifiers	$9.35 \cdot 10^{-6}$	5901.50±8.28 (B)	5919.05±6.96 (C)	5905.95±3.65 (B)	**5826.81**±8.93 (A)
Avg. Generality	$5.56 \cdot 10^{-6}$	**0.0912**±0.0041 (A)	0.0772±0.0025 (B)	0.0552±0.0031 (C)	0.0732±0.0046 (B)



(A) f_1: Avg. System Error (MAE)



(B) f_1: Avg. # Macroclassifiers and Generality

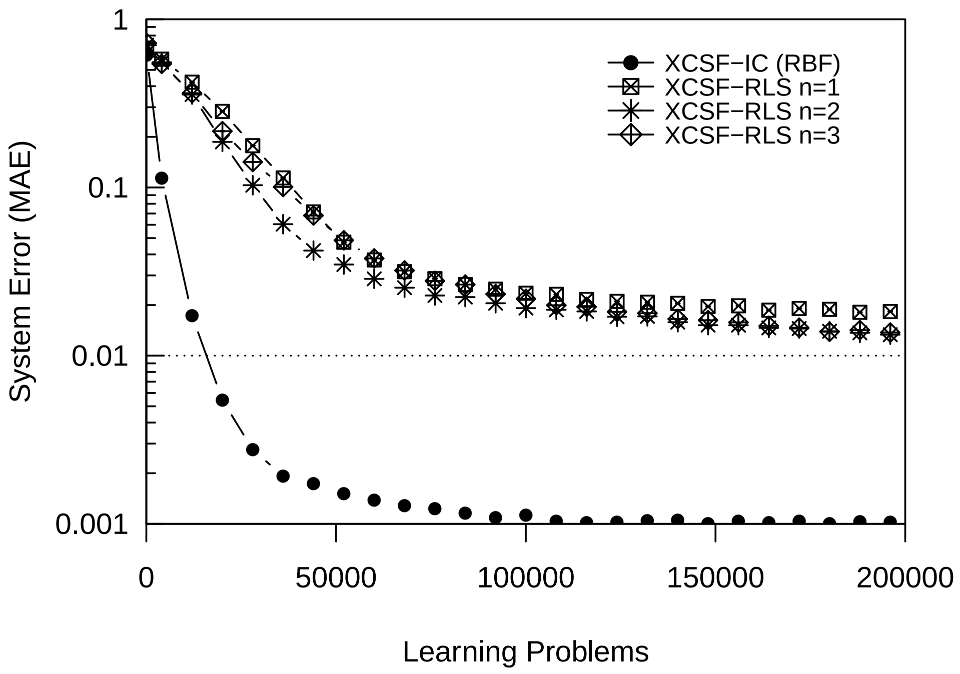
FIGURE 9.6.: Learning curves showing the average results for XCSF-IC with IPI and XCSF on the Eggholder test function f_1

2-dimensional Sine-in-Sine function f_2

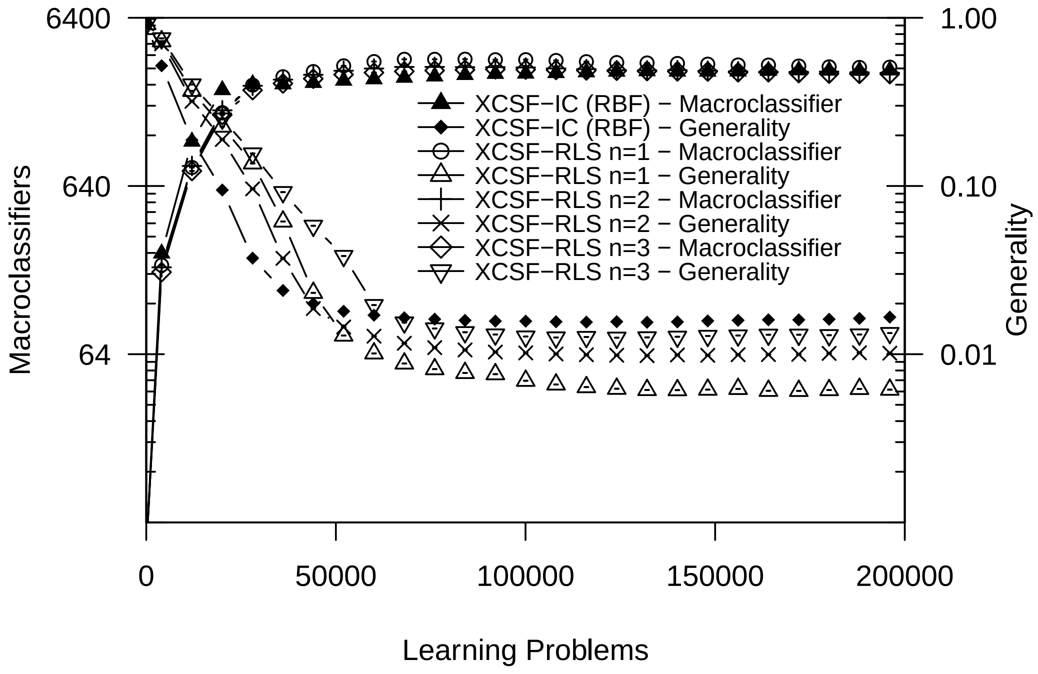
Very similar results as for f_1 can be observed for the second test function f_2 which has already been demonstrated to be difficult for XCSF to approximate in related studies [SB10a; Sta+12b]. The novel IPI technique leads to a significantly decreased MAE (-90.75%), reaching the target error already after less than $30k$ function samples (or learning problems). In contrast, the variants based on polynomial approximation all together fail to reach the target error of $\epsilon_0 = 0.01$ during the entire $200k$ trials. Starting with a lower average generality at the beginning (up to $\approx 50k$ learning problems), XCSF-IC proceeds with evolving classifiers of substantially higher generality until the end. However, on average, classifier generality appeared to be 40.48% smaller compared to the best alternative using cubic approximation. Although third-order polynomials seem to be able to well-approximate the regularities of the sinusoidal f_2 function, the RBF-based solution accomplishes to undercut the target error by one order of magnitude after approximately half of the experiment. The number of macroclassifiers is significantly lower than for XCSF-RLS with $n = 1$ and $n = 2$ polynomial degrees, but not significantly different from XCSF-RLS using cubic approximation. On average the population size of XCSF-IC is $3.83 \cdot 10^{-2}$ % larger which can be regarded as appearing by chance.

3-dimensional Cross function f_3

For the cross function f_3, the general learning capability of XCSF has already been demonstrated in [SB10a; Sta+12b]. Using the novel IPI strategy, however, it turns out that the learning speed is further increased. XCSF-IC is able to reach the target error of $\epsilon_0 = 0.01$ within only $20k$ learning steps on average. In contrast, XCSF-RLS with polynomial approximation only reached that level after approximately $50k$ steps. Apparently, in terms of average population size, XCSF-IC maintains a slightly but significantly higher number of macroclassifiers (+4.15%) over the entire learning period. This seems to be necessary to reach the superior prediction accuracy reflected by a strong decrease in the system error by 59.86%. This observation is further supported by the substantially lower average generality of classifiers evolved by XCSF-IC (-49.11%). It thus clearly contradicts the hypothesis that the use of interpolation instead of approximation always facilitates more general classifiers. Nonetheless, the RBF-based interpolation approach still enables the system to better capture the underlying function complexity. This appears most distinctly at the beginning. In the course of learning, however, the classifier generality continually increases for all configurations. Another effect worth mentioning is that the error of the model-fitting XCSF-RLS instances and that of XCSF-IC converge to the same level. Having a look at the average generality curve it can be noticed that as soon as the generality starts to increase, also the descent of the MAE ends. Interestingly, the MAE starts to increase again to a small extent. This observation is attributed



(A) f_2: Avg. System Error (MAE)



(B) f_2: Avg. # Macroclassifiers and Generality

FIGURE 9.7.: Learning curves showing the average results for XCSF-IC with IPI and XCSF on the Sine-in-Sine test function f_2

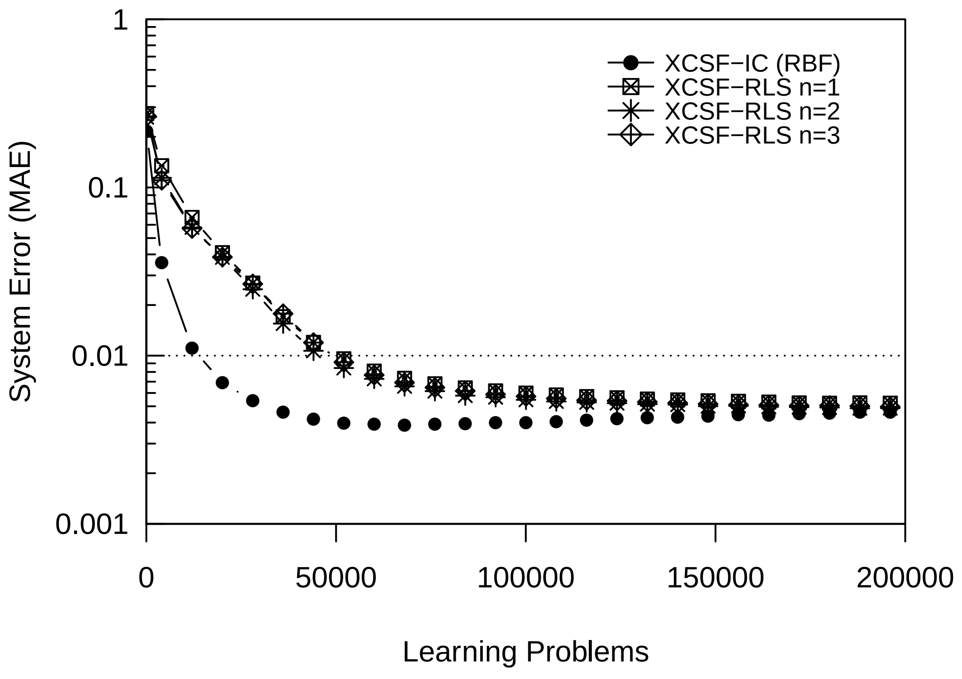
to the lack of accuracy pressure which stops to act in regions beneath the target error ϵ_0 (cf. Sect. 2.4 for details).

6-dimensional Styblinski-Tang function f_4

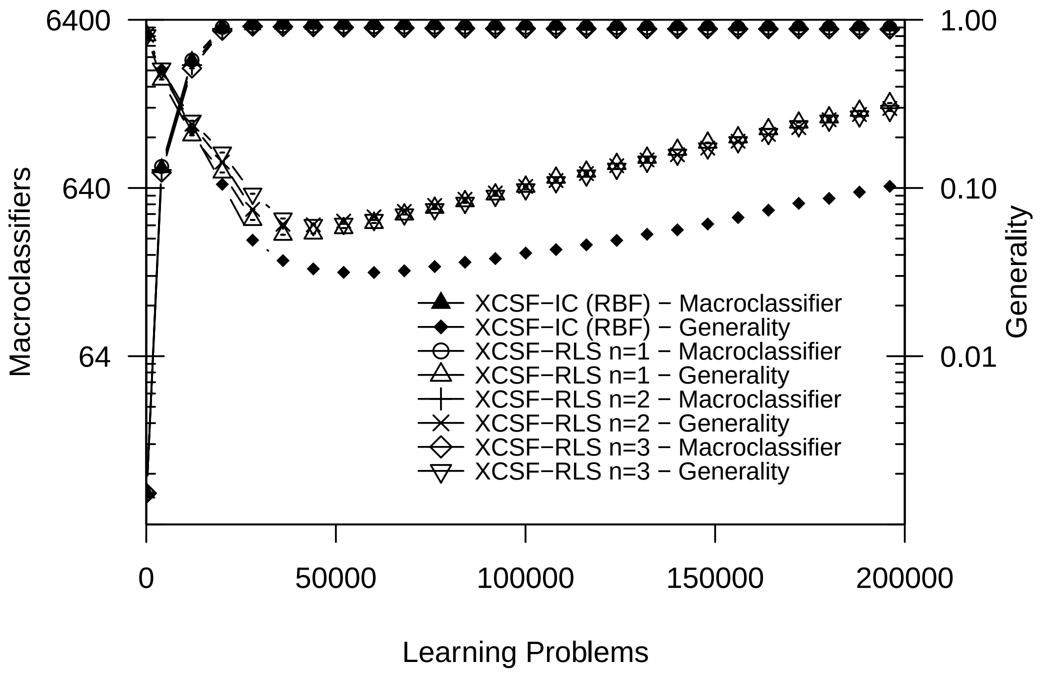
For the most complex 6-dimensional Styblinski-Tang function f_4, XCSF-IC again is able to outperform all XCSF variants using RLS in conjunction with polynomial approximation. Except for the average population size, RBF-based prediction interpolation yields superior performance in terms of a notable system error reduction of 19.03% on average. The population-wide average generality is significantly increased by 18.13% as well over the entire experiment. With regard to the average population size, XCSF with cubic approximation revealed to evolve the least number of macroclassifiers which corresponds, however, to an effective reduction of only 1.26% compared to XCSF-IC. The difference still can be found to be statistically significant. Further significant differences between the investigated configurations can not be reported. By having a closer look at Figure 9.9, it can be recognized that starting from the approximately $125k$-th problem instance, the system error level achieved by XCSF-IC is surpassed by the model-fitting XCSF-RLS variant with cubic approximation employed. At this point, XCSF-IC presumably would benefit from a larger experience memory per classifier ($N_{sp} > 50$) or from a higher number of classifiers maximally allowed in $[P]$. Regarding the latter aspect, inspecting the average population size as given in Table 9.1, all configurations make use of approximately 90% of the maximum allowed size of $[P]$ which is set to $N = 6400$ in this experiment. Still, at early stages XCSF-IC substantially reduces the approximation error, thus, increasing the learning efficiency in the presence of KGs due to lack of experiences.

Results for Noisy Eggholder Function

In order to gain initial insights for the applicability of the IPI approach on realistic applications, this section examines the behavior of XCSF under the presence of noise. Noise is a typical phenomenon in real-world settings where e.g., a variety of sensors deliver measurements which together span the domain of the situation space. The pursued approach to bridge the gap between synthetic and real-world scenarios is to simulate noise for the complex highly multimodal Eggholder function f_1. Therefore, uniform noise is added to the actual function values $f(\vec{x})$ as follows: $f(\vec{x}) + U_{[0,\sigma]}$, with $\sigma \in \{0.05, 0.1, 0.2\}$ and $U_{[0,\sigma]}$ returning a uniformly distributed random number between 0 and σ. In order to ensure solvability of the linear equation system of RBF interpolation, prior to each interpolation operation, a preprocessing step is performed. As already mentioned above, sampling points in SP are searched for duplicates within a distance range of a very small ϵ. If duplicates are found, the center of those duplicated sampling points s_j is used as new coordinate $\vec{x}'$ and the function

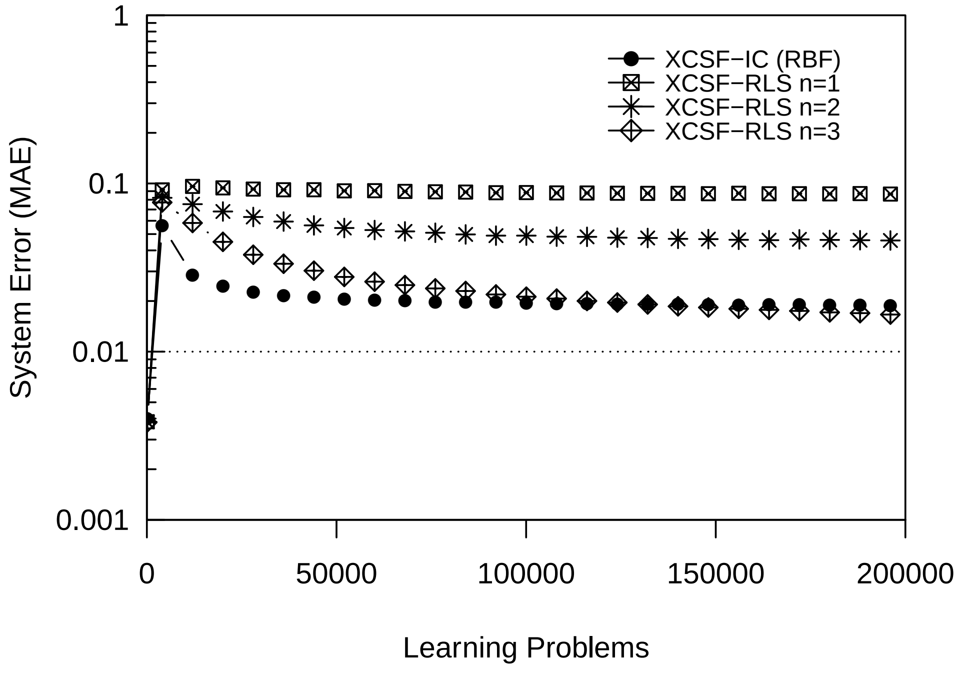


(A) f_3: Avg. System Error (MAE)

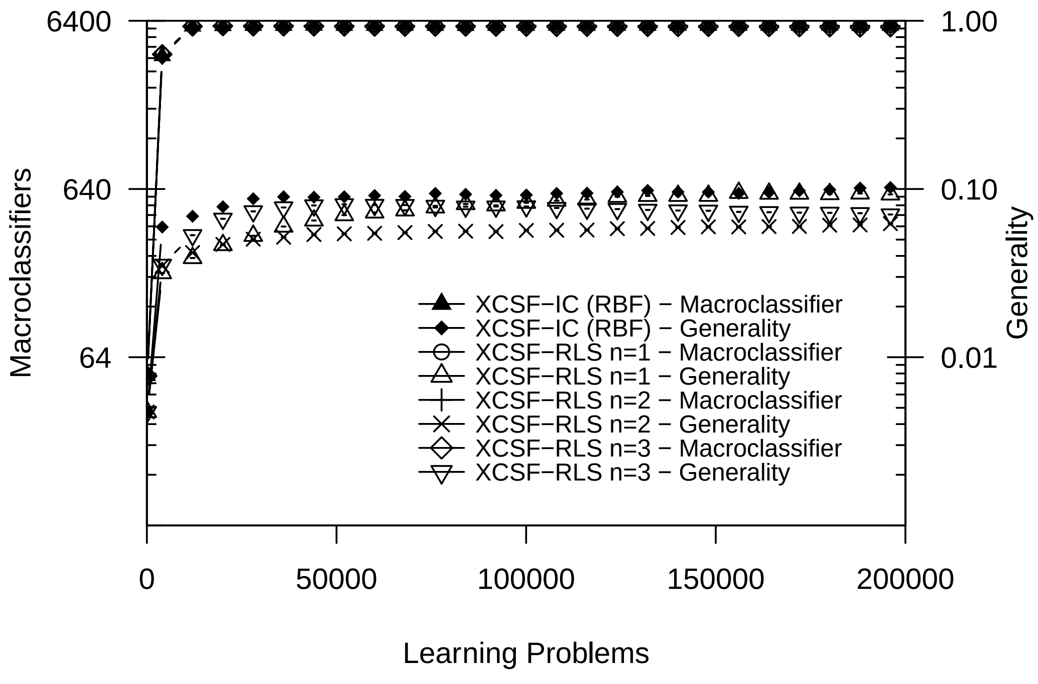


(B) f_3: Avg. # Macroclassifiers and Generality

FIGURE 9.8.: Learning curves showing the average results for XCSF-IC with IPI and XCSF on the Cross test function f_3



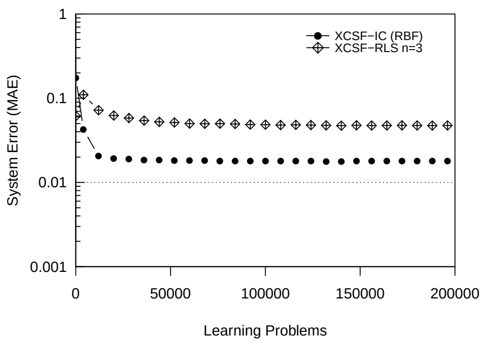
(A) f_4: Avg. System Error (MAE)



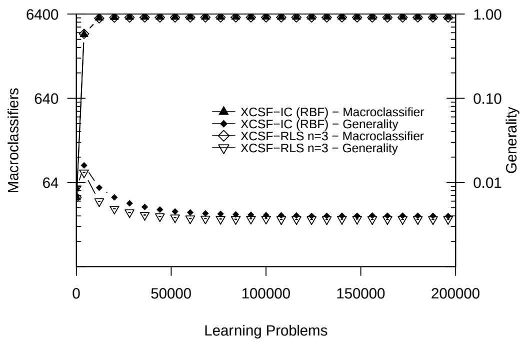
(B) f_4: Avg. # Macroclassifiers and Generality

FIGURE 9.9.: Learning curves showing the average results for XCSF-IC with IPI and XCSF on the Styblinski-Tang test function f_4

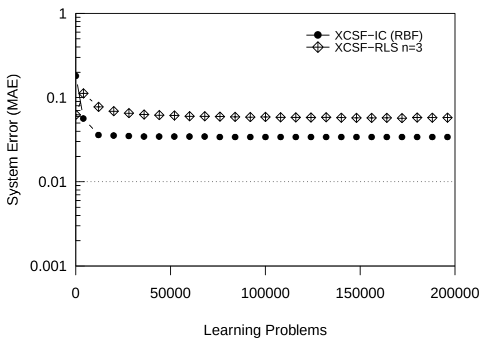
value is the equally-weighted average of all corresponding function values $f(\vec{x}_j)$. For the sake of brevity, it is focused on the particular test function f_1 where XCSF-IC shows the strongest improvements. This is done because insights regarding the sustainability of the achievable improvements are the subject of interest here. The presented IPI approach is compared with the most competitive configuration only, i.e., cubic approximation (which proved to be best in the previous experiments). Table 9.2 summarizes the results. In spite of varying degrees of noise, interpolating XCSF-IC is still able to significantly outperform the most strongest model-fitting configuration (XCSF-RLS with cubic polynomial approximation). For all investigated degrees of noise (ranging from 5% to 20%) both the average system error as well as the average classifier generality show significant improvements. Even for the case of $\sigma = 0.20$, XCSF-IC achieves a system error reduction of 20.26% on average while still evolving a classifier population with a by 24.49% increased generality. Only the average population size appears to be marginally smaller for XCSF-RLS with cubic approximation ($n = 3$). However, in numbers this corresponds to a neglectable difference of only $2.4 \cdot 10^{-2}$ % on average. Figure 9.10 reveals another interesting fact. For both configurations of XCSF, a higher degree of noise seems to result in more general classifiers on average. This is again attributed to the inner generalization pressure of XCS which acts until the accuracy of individual rules starts to deteriorate. In noisy domains usually the target error configured via ϵ_0 is slightly lifted to account for the prevalent uncertainty and to ensure an appropriate fitness signal on which the GA can adequately select parental classifiers. By means of using the IPI approach, the sensitivity regarding this hyperparameter seems to be attenuated to a certain extent, which increases the system's robustness against noise in the end.



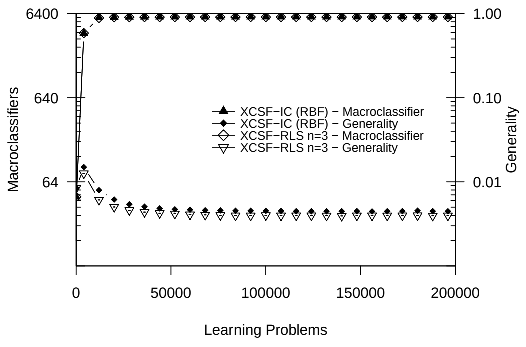
(A) System error for noise level $\sigma = 0.05$



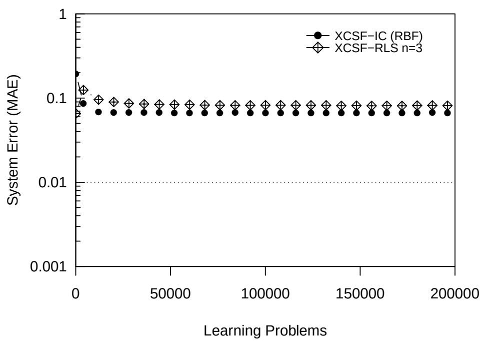
(B) Avg. # Macroclassifiers and Generality for noise level $\sigma = 0.05$



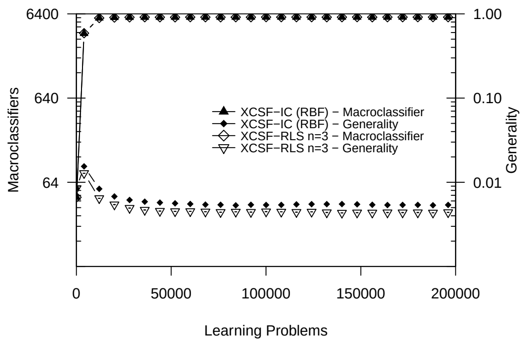
(C) System error for noise level $\sigma = 0.10$



(D) Avg. # Macroclassifiers and Generality for noise level $\sigma = 0.10$



(E) System error for noise level $\sigma = 0.20$



(F) Avg. # Macroclassifiers and Generality for noise level $\sigma = 0.20$

FIGURE 9.10.: Learning curves showing the system error (left) and average number of macroclassifiers / generality (right) comparing XCSF-IC with IPI and XCSF-RLS (n=3) on noisy Eggholder function with different levels of noise $\sigma \in \{0.05, 0.10, 0.20\}$

Table 9.2.: Summary of results on noisy eggholder function f_1. All entries present the mean values and their standard deviations (±1SD) over the entire experiment length of 200k learning problems. Bold entries indicate statistically significant superior performance over the other configuration, i.e., $p < 0.05$.

$\sigma = 0.05$	XCSF-**IC (RBF)**	XCSF-**RLS (n=3)**
System Error (MAE)	**0.0192**±0.0002	0.0531±0.0006
Avg. Macroclassifiers	5788.75±7.37	**5735.79**±5.76
Avg. Generality	**0.0050**±0.0001	0.0042±0.0001
$\sigma = 0.10$	XCSF-IC (RBF)	XCSF-RLS (n=3)
System Error (MAE)	**0.0353**±0.0002	0.0625±0.0009
Avg. Macroclassifiers	5767.69±5.87	**5741.31**±4.03
Avg. Generality	**0.0052**±0.0001	0.0045±0.0001
$\sigma = 0.20$	XCSF-IC (RBF)	XCSF-RLS (n=3)
System Error (MAE)	**0.0677**±0.0002	0.0849±0.0007
Avg. Macroclassifiers	5747.51±5.95	5748.88±7.87
Avg. Generality	**0.0061**±0.0001	0.0049±0.0001

9.4. Discussion

The initially posed hypothesis that interpolation-based classifier prediction leads to increased average generality is partially confirmed. However, it turns out that depending on the problem complexity, the opposite effect can also appear. For the 3-dimensional Cross function f_3 for instance, XCSF-IC evolves classifier populations with smaller average generality (cf. learning curves for f_3 in Fig. 9.8) but still uses those more specific rules effectively during the actual prediction step. It would therefore be really interesting to see the impact of condensation and compaction techniques [BLW08; TMU13] on the average generality of the final solution produced by the interpolation-based XCSF-IC. This aspect will be examined further in future work. Overall, it can be concluded that the use of RBF interpolation to model the prediction of individual classifiers leads to strongly decreased average system errors, especially at early learning periods where KGs are highly present. Especially for the investigated cases of the highly non-linear functions f_1 and f_2, the introduced IPI approach enables XCSF to quickly reach the targeted error level. This could not be achieved at all by any of the alternative polynomial model learning variants within the conducted learning time. Apparently, the flexibility of the utilized parameter-less thin plate spline basis function bears promising strengths with respect to the ability to capture the underlying function's complexity. The advantage of using interpolation-techniques instead of approximation approaches as almost exclusively done so far is that the former means does not require model hyperparameters such as the degree of the polynomial. By omitting such design time decisions, possibly injected model bias that might lead to underfitting effects can be prevented.

Sort of counterintuitively, these findings hold also true for noisy functions where uniform deviations up to 20% are added to the highly non-linear Eggholder function f_1. It might have been expected that interpolation easily falls into the trap of overfitting to noisy data where in contrast approximation would simply minimize the squared error as usual. However, such a behavior can not be observed in the experimental findings so far. Due to the deduplication process to ensure solvability of the linear equation system required to calculate the RBF interpolation, noise effects are averaged out. In the case of sticking to the traditional fitness-weighted classifier mixing procedure, a similar effect is expected to eventuate. Furthermore, since the experience memory is of small limited size, classifiers will continually replace older experience with newer ones. This last aspect is crucial in online-learning settings to account for the *stability-plasticity dilemma* causing phenomena such as *detrimental forgetting*.

To a certain extent, the observed benefits of employing the IPI strategy are attributed to the newly introduced classifier mixing technique. Here, a system prediction is calculated by means of taking the union of sampling points from all matching classifiers into consideration during the actual prediction interpolation step. Since still the classifiers' learning parameters are updated on the basis of their individual prediction errors, the generalization pressure applies as intended and the classifiers evolve toward their near optimal condition structure. This in turn has a direct effect on the density of sampling points which are collected within each classifier. This implicit self-adaptive sampling point density is deemed one of the most appealing factors of this new prediction modeling technique. By increasing the number of sampling points stored in each classifier (N_{sp}), classifiers might reconstruct the underlying problem subspace with the same level of accuracy, but with more generalized condition structures. This would eventually conform to the hypothesis posed at the beginning of this chapter. This aspect as well as ways to self-adaptively adjust N_{sp} as a response to currently identified accuracy levels during the system's runtime constitute important steps of future work.

Current limitations of this approach that also need more research effort comprise the following aspects: The adjustment of the experience memory (or sampling point sets) for each classifier needs more sophisticated ways to account for non-uniform data distributions. One idea is to select experiences for removal in order to maximize entropy measures. This would result in a steady pressure toward uniformly distributed experiences over a particular classifier's subspace of responsibility (as determined by its condition). So far, the approach accesses one global instance of the IC. This implies sequential interpolation steps which naturally reduce the computation efficiency. A solution would be to equip each single classifier with a lightweight instance of the IC in order to facilitate parallel interpolations. This would also make hybrid interpolation methods possible which in turn leverages the hybrid ensemble learning nature of XCS. In view of more automated ML, this clearly constitutes a promising direction of future work.

Having mentioned the possibility for hybridization before, an even more beneficial combination is expected when XCSF is endowed with the capability of self-deciding to switch between interpolation-based predictions and model-learning (i.e., approximation) techniques. In periods where uncertainty, covariate drifts and resulting KGs characterize the currently incoming situations, the novel IPI technique should be active. During stable phases, a model can be learned by either (1) employing the online RLS optimizer, or, (2) finding optimized weights by means of batch-wise gradient descent or linear least squares using the collected experiences in each classifier. This hybrid approach would also contribute to the system transparency, since linear regression still counts as one of the most human interpretable ML models these days.

9.5. Related Work

In the literature, various attempts to model the payoff prediction parameter of a classifier exist. However, all of them focus on approximation approaches or directly evolve the models. In [SMH18], the author has been the first who posed the question “What about interpolation?”. Nevertheless, the numerous approaches to computed predictions in classifiers definitely constitute related work with respect to the interpolation-based methodology presented in this chapter.

Lanzi et al. investigated higher-order polynomials up to a degree of three for the local prediction approximation within the classifiers of an XCSF in [Lan+05b]. The authors have demonstrated that polynomials beyond the first order can improve both the predictive accuracy and the generalization capabilities, and may result in a more compact solution in contrast to standard XCSF with linear approximation. For these polynomial models, several optimization algorithms to update the model coefficients have been studied by Lanzi et al. in [Lan+06]. In summary, their results indicate that RLS and the *Kalman filter* yield superior performance in contrast to *gain adaptation* algorithms in terms of a decreasing error rate and increased generality of the global solutions.

Another approach to classifier prediction modeling in XCSF is presented by Loiacono et al. in [LML07]. They introduced XCSF utilizing support vector regression (XCSFsvm) for for that purpose. XCSFsvm turns out to converge faster than XCSF with linear approximation in conjunction with the RLS technique, yields lower system errors for higher dimensional test functions and can result in even more generalized solutions. However, only comparisons to XCSF with linear polynomial approximation and RLS have been conducted. Furthermore, it seems that the axis-parallel hyperrectangular condition representation was used, which seems not to be suitable for capturing dependencies between the features of the input vectors (also called epistasis [UB17]) which appear in parts of their test functions.

Tran et al. examine the use of an evolution strategy for model coefficient optimization in XCSF in [TSD08]. Their approach is called XCSF-ES. Assuming a linear model for computing the classifier prediction, their findings indicate that on functions with one-dimensional input, XCSF-ES generates a significantly lower system error than the original XCSF with least mean square update (or modified delta rule), and slightly better results than XCSF with RLS. Also, the evolution strategy has been declared to have $O(n)$ complexity, while RLS requires $O(n^2)$ in terms of computation time.

Standard polynomial prediction approximation has been replaced by a feedforward neural network in [LL06] as well. The structure of this neural network is evolved by means of the GA in XCSF. It is shown that XCSF with neural network prediction outperforms XCSF with linear approximation for highly nonlinear functions. However, XCSF variants with higher-order polynomial approximation have not been taken into consideration. Bull and Hurst previously proposed a more incisive approach by replacing entire rules with *Multi-layer Perceptrons* (MLP) in [BO02]. Here, the prediction is still approximated by the neural network and calculated by a forward propagation through the neural net. However, also the condition representation is implicitly encoded. More precisely, whether or not a classifier is a member of $[M]$ is determined by an additional output neuron.

In summary, it appears that the most investigated and among the considered studies always at least equally effective approach for classifier prediction can be achieved by the combination of (1) the general hyper-ellipsoidal condition structure, and, (2) polynomial approximation in conjunction with the RLS technique. This has also been a reason why the experiments presented in this chapter compared the novel approach against XCSF using linear, quadratic and cubic polynomials for prediction modeling.

Regarding the aspect of storing a set of input vectors, Lanzi et al. in [Lan+05b; Lan+05a] have previously proposed the use of a non-incremental linear least square method for the model coefficient optimization. The applied procedure demands for a set of data samples which have been maintained by a FIFO queue. Ways for more sophisticated replacement of memorized samples, however, have not been part of their studies and therefore left for future research.

9.6. Chapter Summary

In this chapter, the prediction computation step of individual classifiers in XCSF has been replaced by an interpolation-based means. RBF interpolation using the *Thin-Plate-Spline* (TPS) basis function was shown to constitute a powerful method for computed predictions. Two effects caused by the ability of the novel IPI strategy to capture more complex characteristics of the underlying function have been hypothesized: (1) Higher approximation accuracy. (2) Increased average generality of the evolved rule sets. The obtained results of conducted validation studies partially confirmed these hypotheses. XCSF-IC using IPI strongly increases the approximation accuracy as well as the learning efficiency on each of the considered test functions. Increased classifier generality could only be reported for two of the four investigated functions. Additional results further confirmed the sustainability of this transductive means of prediction modeling on the highly multimodal Eggholder function under varying degrees of noise. A thorough discussion on current limitations and promising aspects for further enhancement of the IPI strategy as well as an appreciation of related work in this regard closed this chapter. With the end of this chapter, the main part of this thesis comes to an end. All approaches that have been developed throughout the previous chapters have one aspect in common – they incorporate interpolation as a means to transductive knowledge inference in Michigan-style LCS. This is done in order to *reactively* alleviate negative impacts caused by KGs which XCS-based SLAS might face in the course of learning. The next chapter discusses concrete directions of future research where the goal of approaching *Proactive LCS* is pursued.

Chapter 10.

Toward Proactive Learning Classifier Systems

All the techniques that have been developed so far, regardless of considered separately or in combination, provide reactive methodologies to overcome present KGs within a knowledge base at any point in time within the course of learning. These methods are regarded as *reactive* since the actual invocations of a knowledge-based learning agent's $L(K)$ knowledge construction processes are not directly affected.

With the methods and concepts that are going to be outlined in the following sections, an elaborate outlook on future research directions in terms of enhancing the *self-awareness* of SLAS is provided. The pursued objectives are to enable the involved learning agents $L(K)$ to *self-reflect* their own state of knowledge by means of proactively analyzing their current knowledge bases K evolved so far. More precisely, the $L(K)$ of a SLAS will be endowed with the capability to proactively seek for KGs in their knowledge bases K and, subsequently, to initiate the construction of new knowledge elements k^* in order to alleviate possibly negative impacts on the system's utility.

Therefore, a second research hypothesis, which is not confirmed comprehensively in this thesis, but constitutes the main conjecture motivating future research can be formulated as follows:

Hypothesis. *The* ***proactive construction*** *of new knowledge elements $k^* \notin K$ for covering* ***inexperienced***, ***under-explored*** *or* ***uncertain*** *regions of the problem space PS, as approximated by K, leads to an improved learning progress of a learning agent $L(K)$ under the assumption that k^* will actually become* ***relevant*** *during the lifetime of $L(K)$.*

In the following sections, first steps toward approaching *proactively knowledge constructing learning algorithms* are taken. In the first Section 10.1, a methodology recently proposed by the author in [SMH17] is briefly recapitulated to substantiate the general potential of using *Active Learning* (AL) within XCS's algorithmic structure. Subsequently in Section 10.2, a novel concept for *proactive knowledge construction* as recently proposed by the author in [Ste+18] is introduced formally and accompanied by a first proof confirming the general plausibility of the conjecture. In the same breath, the generic MLOC architecture is revisited to allow the incorporation of the proposed mechanisms in order to render SLAS self-reflecting proactive learners.

10.1. From Reactive to Active Learning Behavior

The objective of this section is to provide an intuition of how to accomplish a shift from a mostly reactive to a predominantly *active* construction process of knowledge elements, i.e., classifiers $cl \in [P]$. By means of adopting *Active Learning* (AL) concepts, LCS will yield a sort of *curiosity*[1] with respect to parts of their knowledge bases $[P]$ at which the knowledge is only insufficiently experienced so far (type-2 KGs, cf. Ch. 3.3).

As a short recapitulation of Section 3.4, in an LCS, the knowledge elements are contained in a population $[P]$ of IF-THEN rules, also called classifiers cl, that maintain certain learning parameters about their quality $(cl.p, cl.\epsilon, cl.F)$ and confidence $(cl.exp, cl.num)$. Thus, adopting the introduced notions of KGs, the following similes apply: $K \equiv [P]$ and $k_i \equiv cl_i$. Accordingly, type-1 KGs can be formalized by

$$kg \in KG_1 \coloneqq \{cl\}_{cl \notin [P]},$$

and KGs of the second type are denoted by

$$kg \in KG_2 \coloneqq \{cl \mid cl.exp < \theta_{exp} \;\wedge\; cl.q(p, \epsilon, F, num) < \theta_q\}_{cl \in [P]}.$$

10.1.1. Approach: Integrating XCS with AL

The pursued approach is to devise an active knowledge creation strategy by endowing XCS itself with the capability to decide which regions in its knowledge base are only insufficiently represented yet. Thus, it shall be enabled to autonomously identify KGs. Therefore, concepts from the domain of *Active Learning* (AL) [Set09] are adopted. More precisely, in a first attempt concrete AL techniques called *Least Confident Uncertainty Sampling* [LG94] as well as *Query-by-Committee* (QBC) [SOS92] will be adapted to fit XCS's algorithmic structure. Furthermore, first steps are taken to answer the question of whether the acquisition of knowledge in an even more exploratory fashion reveals additional benefits. It is thus examined, if it is valuable to construct classifiers prior to actually requiring these particular knowledge elements for the first time. Therefore, so-called *Query Synthesis* approaches [Ang88] are subject of preliminary investigation, even though these are restricted to a fully randomized approach, so far.

[1]Computational curiosity was recently reviewed more thoroughly elsewhere [WM13].

Adoption of Pool-based AL

AL [CAL94; Set09] shares strong similarities with the SSL paradigm which has proven itself to enhance the efficiency of supervised learning algorithms in various classification tasks (cf. e.g., [RCS15]). In both settings, it is assumed that only a small set of labeled training data $\mathfrak{L}$ is available (e.g., due to high labeling costs or other non-monetary efforts) comprising training examples of the form $(\vec{x}, y)$. In addition to $\mathfrak{L}$, there is also a large set of unlabeled data assumed, denoted by $\mathfrak{U}$, since obtaining unlabeled samples is supposed to be inexpensive. What is specific to AL, however, is the capability of the learning algorithm to *self-decide* which unlabeled instance $\vec{x} \in \mathfrak{U}$ is most beneficial to be selected and subsequently sent to a so-called *oracle* for obtaining a correct label in return. In the simplest scenario, the oracle is assumed to be omniscient and omnipresent and, thus, always yields the correct label for the queried instance. The queried instance together with the acquired target label is then added to the labeled set and the model is retrained with the augmented training data. This setting is a specific form of AL called *pool-based* AL. Beside pool-based AL, in essence two additional major paradigms can be found in the literature (cf. [Set09]) – *stream-based* AL and *membership query synthesis*. So far, the focus has been set on the former and the latter scenarios, i.e., pool-based AL and membership query synthesis. Here, the occurring match sets $[M]$ as part of XCS serve as *pool* of "unlabeled samples" $\mathfrak{U}$. As will be described in a later section, such a pool is not needed for the synthesis scenario, where in the most naive case arbitrary elements of the input space X are selected as queries for the oracle.

In the context of LCS, a *query instance* $\vec{x}_Q$ is represented by a particular classifier cl_Q, more precisely by its condition $cl_Q.C$. Thus, not only a single vector $\vec{x}_Q \in X$ is presented to the oracle, but rather a particular subspace of the input space, i.e., $cl_Q.C \subseteq X$. The task of the oracle is to assign the most appropriate label, or action, for the queried classifier cl_Q. If the oracle is not able to answer such a query with a predefined minimum level of confidence, the query is rejected and no answer can be retrieved by XCS.

In case of query rejection, it is assumed that the queried (macro-)classifier cl_Q negatively influences the overall learning process. Therefore, it is physically deleted from $[P]$ (not only has its numerosity decremented). On the other hand, when an oracle answer is received, a new classifier cl^* is constructed with the action $cl^*.a$ set to the oracle's suggestion. The condition is assigned as $cl^*.C := cl_Q.C$. The reward prediction value is set to $cl^*.p = 1000$. So far a binary rewarding scheme is assumed, i.e., XCS receives a reward of 1000 for correct decisions, and 0 otherwise. The fitness estimate is set to $cl^*.F = 0.9$ (the minimum confidence level of the oracle). The experience parameter is initialized with the hyperparameter value of θ_{sub} to enable cl^* to immediately act as subsumer. All remaining attributes are initialized as usual. cl^* is then added to $[P]$ and $[M]$ before XCS proceeds with the action-selection step.

The Omniscient Oracle The aforementioned assumptions about the oracle seem to be unrealistic and often have to be relaxed when human experts are involved. Humans are not necessarily omnipresent and certainly not omniscient. Accordingly, the uncertainty of humans as well as the interaction load with which it is confronted have to taken into account. The perspective postulated in this endeavor, however, is to not restrict the oracle to be realized as a human annotator. Rather, it is envisaged to take advantage of available simulations or existing mathematically expressed heuristics that model at least the crucial aspects of certain real-world problems. Such models can then be used alongside or instead of humans. It clearly depends on criticality of the learning task and the according time constraints whether to bring a fast reacting and non-fatiguing artificial oracle into the learning loop, or rather a human oracle with highly accurate domain competence but reduced availability into the learning loop.

Figure 10.1 proposes a generic two-layer architectural blueprint to render LCS AL algorithms.

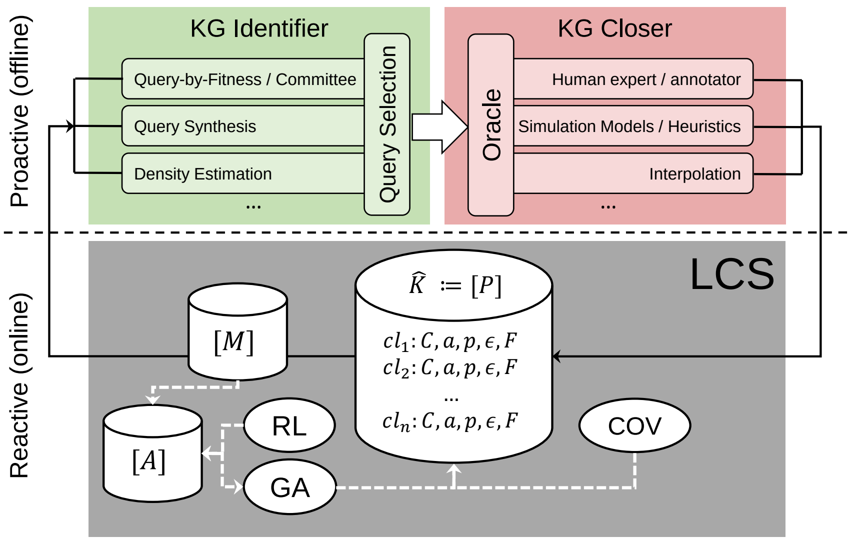


FIGURE 10.1.: Schematic for a two-layer architecture of Active LCS

The bottom layer constitutes the conventional LCS architecture, where the population $[P]$ serves as the knowledge base K (denoted $\hat{K}$ in the figure). The LCS works as usual, i.e., it *reacts* to environmental stimuli. On the top layer, two components are illustrated: (1) A *Knowledge Gap Identifier*, which is responsible for seeking KGs in the algorithm's knowledge base K by means of AL techniques, and (2) the *Knowledge Gap Closer*, which contains the oracle having access to several sources, e.g., a human expert, a simulation or the IC as introduced in Chapter 4. It is intended to provide the possibility that all oracle types can be queried in parallel or individually.

The next paragraphs each briefly outline the original concept from the AL domain and subsequently describe the adoptions of specific techniques in order to design an AL-extended XCS (ALXCS).

Uncertainty Sampling

Uncertainty Sampling as introduced by Lewis et al. in [LG94] is an AL technique that aims at querying instances from $\mathfrak{U}$ about which the current model trained solely on $\mathfrak{L}$ is most uncertain regarding the correct output.

Least Confident (LC) uncertainty sampling is a straightforward approach to be briefly introduced next. The formula (cf. [Set09]) for LC uncertainty sampling is given by:

$$x^*_{LC} \coloneqq \underset{x}{\operatorname{argmax}} \Big[1 - P_\theta(\hat{y}|x)\Big], \text{ where } \hat{y} \coloneqq \underset{y}{\operatorname{argmax}} \Big[P_\theta(y|x)\Big]. \tag{10.1}$$

Here, P_θ denotes the posterior probability of the current model (or hypothesis) $h(\theta)$ parameterized by the parameters to be optimized θ. $\hat{y}$ is the target label or class that maximizes the posterior probability for the currently considered unlabeled instance $x \in \mathfrak{U}$. Therefore, the formula yields an instance x^*_{LC} about which the model $h(\theta)$ is most uncertain. The above formula reveals that a posterior probability is necessary, i.e., the probability of a label $\hat{y}$ being correct given the currently considered instance x. However, learning approaches that are centered around probability distributions in a Bayesian fashion, such as *naïve bayes* classifiers, are fundamentally different form the learning intuition of an LCS. Because XCS is not a probabilistic model in a Bayesian sense, but rather an input space partitioning online learning system with an evolutionary optimization heuristic at its core, another measure similar to $P_\theta(\hat{y}|x)$ has to be determined. Intuitively, a classifier's fitness estimate $cl.F$ seems to constitute a legitimate candidate for expressing a notion of confidence regarding the subspace for which it is responsible. If the fitness is low, the classifier's predictive accuracy is poor. However, this does not necessarily indicate total incorrectness. Covering usually initializes fitness to a pessimistically low value. Novel classifiers, thus, first have to prove themselves. Accordingly, the accuracy of payoff prediction might be poor for the moment but also highly uncertain since it is possible that it turns out to be accurate in the future. In order to also consider a classifiers "age", its experience $cl.exp$ is also incorporated. Accordingly, the XCS adoption of the LC uncertainty sampling technique, referred to as *Query-by-Fitness* (QBF), is initially defined as follows:

$$cl_Q \coloneqq \underset{cl \in [M]}{\operatorname{argmax}} \Big[(1 - cl.F) \cdot cl.exp\Big] \tag{10.2}$$

cl_Q denotes the classifier $cl \in [M]$ to be selected for being presented to the oracle. The above selection strategy ensures to query classifiers having rather low fitness estimates but have already gained highest experience in their niches. This ensures that no classifiers are chosen that were just created.

With this primary approach, XCS is made curious about particular subspaces of X, i.e., about regions which it is rather uncertain, despite a certain number of experienced updates. Put another way, niches for which XCS has difficulties to figure out the "correct" action. This might be due to so-called *(strong) overgenerals* [Kov00b] which match in many situations where different actions might be correct. So, with the active KG identification technique as proposed above, XCS is enabled to uncover such disruptive classifiers and accordingly initiate certain countermeasures. For instance, if the oracle is not able to decide for the correct action with sufficient confidence, this overgeneral classifier can be simply removed from the knowledge base $[P]$. On the other hand, it is also possible to explicitly seek just created classifiers by simply inverting *cl.exp* in Equation 10.2. In this case, classifiers with the lowest fitness and at the same time the smallest experience are preferred. This can be done in order to bootstrap the initial learning phase by using the oracle for obtaining the correct actions already at an early stage of a learning.

The Committee of Matching Classifiers

Query-by-Committee (QBC) is a query selection strategy that uses a variety of independently trained models. The approach was first introduced by Seung et al. [SOS92]. The collection of models serves as committee

$$\mathfrak{C} = \left\{ h(\theta^{(1)}), h(\theta^{(2)}), \ldots, h(\theta^{(|\mathfrak{C}|)}) \right\}.$$

QBC reveals prediction conflicts by comparing the outputs of the committee members. A conflict's extent is determined by a measure of disagreement between the different models. One approach is to use *vote entropy* as reported in [Set09]. This technique essentially counts the number of times the committee members vote for a particular label, then derives a probability in terms of relative frequency, and, subsequently calculates the *Entropy* [Sha48].

In order to use QBC within the context of XCS one possibility is to train several XCS instances on different subsets of the labeled data at the same time. However, a less computationally expensive approach is chosen which interprets each classifier as a committee member. Let $A_{[M]}$ define the set of distinct actions in $[M]$. Then the vote entropy $H_{[M]}$ is calculated by:

$$H_{[M]} = \sum_{a \in A_{[M]}} V(a) \log_2 V(a) \tag{10.3}$$

with the actual vote for each action a given by

$$V(a) = \frac{\sum_{cl \in [M]^{(a)}} cl.num}{\sum_{cl \in [M]} cl.num}. \tag{10.4}$$

Thereby, a subset of $[M]$ which contains only those classifiers advocating a specified action a is given by:

$$[M]^{(a)} := \{cl \in [M] | cl.a = a\}.$$

After determining the vote entropy $H_{[M]}$ for the current match set, further the maximum entropy

$$H_{max} = \log_2 |A_{[M]}| \tag{10.5}$$

is calculated. H_{max} is equivalent to a situation of maximum conflict, i.e., fully uniform action vote distribution. The closer $H_{[M]}$ is with respect to H_{max}, the higher is the decision conflict about which action is most suitable. Considering both $H_{[M]}$ and H_{max}, a query is only sent to the oracle if

$$H_{[M]} \geq H_{max} \cdot (1 - \theta_{sim}) \tag{10.6}$$

is satisfied. θ_{sim} defines a tolerance factor to control the extent of the conflict required to send a query to the oracle. If this condition evaluates true, the query itself is selected by means of QBF again. Thus, QBC essentially narrows the application of QBF to match sets with high conflicts. This, in the end, is one approach to reduce the interaction load between the learning algorithm and the oracle. Even if not investigated further so far, it is hypothesized that the QBC technique reveals even higher benefits when a so-called *best action mapping* [Nak+15] is applied. This mapping, in contrast to standard XCS, does not strive to a learn *complete* $X \times A \rightarrow P$ mapping, also comprising classifiers which accurately predict rewards for incorrect actions with regard to a certain niche.

Knowledge Generation by Query Synthesis

Membership Query Synthesis [Ang88] is a special case of query selection where the queried instance $\vec{x}_Q$ is generated *de novo*. This means it is synthesized within the bounds of the problem's input space domain X. There are several approaches for synthesizing queries. Probably the most naive form is to generate an instance $\vec{x}_Q$ completely by chance as a randomly selected vector from X. A more sophisticated approach, i.e., iteratively estimating the distribution of the underlying data generating process by means of *kernel density estimation* (KDE), is outlined in the subsequent Section 10.2. For now, however, randomized query generation is subject of preliminary investigations.

Accordingly, the concept of *Query-Synthesis-at-Random* (QSR) is proposed. Here, a completely new classifier cl^* that also serves as the classifier cl_Q to be queried is constructed de novo. Its condition is created by random selection of a vector $\vec{x} \in X \subseteq \mathbb{R}^n$ and by choosing an interval predicate within predefined bounds, as similarly done by covering, subsequently. For determining these bounds, it is imaginable to use the default spread parameter r_0, so that each interval predicate is bounded by $[x_i - r_0, x_i + r_0]$. However, fist experiments have been conducted by using a different spread parameters s_{min} and s_{max}. Thus, the randomly constructed interval predicates are restricted to lie within $x_i \pm s \in [s_{min}, s_{max}], \forall i = 1 \dots n$.

Generating queries de novo has been found to not being suitable in any kind of learning task. For example, in handwritten character classification, completely undefined symbols can be produced [Set09; BL92]. However, in problem spaces where a uniform sampling of the entire domain can be assumed, QSR is hypothesized to be beneficial. On the other hand, considering very domain specific feature spaces such as for medical diagnosis tasks, randomly created instances bear the risk to easily confuse a human oracle since the attribute combination might be completely arbitrary.

10.1.2. Preliminary Evaluation

The following paragraphs report on the first results obtained from a conducted preliminary study.

Experimental Setup The outlined approaches have been evaluated on two different classification scenarios: (1) The well-known *Wisconsin Breast Cancer* (WBC) dataset [MSW95] and, (2) a novel toy problem similar to the CBP problem as introduced before but with a larger action space – the *Mario problem*. Since XCS is asked to solve classification tasks, the 0/1000 binary reward scheme is applied.

All experiments execute XCS for 100k alternating explore/exploit trials and have been repeated for 30 i.i.d. runs. Statistical significance of the differences between ALXCS and XCS is determined using pairwise t-tests. A prequential, i.e., *test-then-train* learning strategy for both scenarios has been applied to adhere to the online learning nature of XCS. Instances from the WBC data set are selected uniformly at random and with replacement. For the synthetic and continuous Mario toy problem, the input space is sampled uniformly. Thus, for both scenarios, the occurrence of KGs and, thus, the potential for improvements are again expected mostly at the beginning of the learning task. For the QBF and the QSR technique, the oracle is queried at each step. Using the QBC approach, the number of oracle interactions is self-decided.

For these preliminary experiments, the oracles are simulated by means of simple heuristics as described below.

Wisconsin Breast Cancer Data Set The WBC data set [MSW95] is available from the UCI Machine Learning Repository [Lic13] and can be found under the category of life sciences. It consists of 699 instances each having 9 attributes which show the results of a fine needle aspiration for the diagnosis of breast cancer. The class labels are "2" and "4" indicating a benign or malignant tumor, respectively. The attributes are encoded by integers in the range $[1, 10]$. Without loss of generality, the attributes are normalized to the range $[0, 1]$. Missing attribute values are marked as "?" in the data set are assumed to be matched in any case. This dataset has a class imbalance ratio of 0.655 : 0.345 (benign : malignant). The WBC data set has already been investigated in the LCS literature [Wil01; KOB08]. Thus, it is deemed a sensible candidate for preliminary experimentation, since it allows for straightforward validity comparisons.

For the present study, the oracle has been simulated by a rather simple heuristic: It is realized by a simple nearest neighbor search through the available instances. Therefore, the Euclidean distance between the center point $\vec{c}$ of a queried classifier's condition $cl_Q.C$ and the instance $(\vec{x}_i, y_i)$ from the set of labeled data $\mathfrak{L}$ is calculated. The label of the instance $\vec{x}_i$ with the minimal distance is then returned as the oracle's answer, i.e.,

$$cl_Q.a := \operatorname*{argmin}_{y_i} \mathbf{d}\left(cl_Q.C.\vec{c}, \vec{x}_i\right)_2 . \tag{10.7}$$

This heuristic implies a certain degree of oracle uncertainty. The nearest neighbor is not guaranteed to yield the correct classification. Therefore, the Euclidean distance is transformed to a similarity measure defined between $[0, 1]$, where 1 stands for the maximum distance. The normalized Euclidean distance is subtracted from 1. Whenever this similarity measure is < 0.9, the query is rejected by the oracle. For the WBC experiments, XCS is configured according to [Wil01]: $N = 6400$, $\alpha = 0.1$, $\beta = 0.2$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 48$, $\epsilon_0 = 1$, $\theta_{mna} = 2$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\chi = 0.8$, $\mu = 0.04$, $p_{ini} = 10.0$, $\epsilon_{ini} = 0.0$, $F_{ini} = 0.01$, $r_0 = 0.4$, $m_0 = 0.2$. The interval-based UBR condition representation is used. Uniform crossover and tournament selection is used with $\tau = 0.4$. Action mutation is not used. GA subsumption is active.

A Novel Toy Classification Problem: Mario Pixel Art The Mario environment consists of a 16x16 pixel art grid representing the famous video game character Super Mario [Nin85]. Figure 10.2 depicts the problem space.

The pixel art comprises seven different colors. The inner rectangle over the yellow knobs shows an exemplary condition of a queried classifier that is expected to be answered by the oracle. The learning objective is to learn the correct color for each possible position $\vec{x} \in \mathbb{R}^2$ in the picture. This toy problem is reminiscent of the CBP as introduced before. In contrast to CBP, however, the Mario problem facilitates the evolution of more general classifiers. See for instance the problem niche which represents the blue trousers of Mario in Figure 10.2. Additionally, it

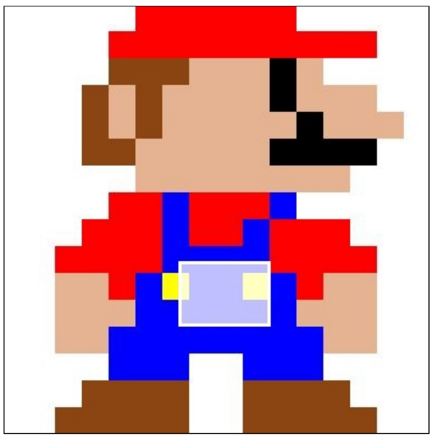


Figure 10.2.: Problem space of the novel Mario toy classification problem

entails a more complex action space A of size $|A| = 7$ and, thus, constitutes a multi-class classification problem. The input space $X \subseteq [0,1]^2$ is continuous. Thus, each "pixel" has a 1/16 share of the input space in each dimension.

The artificial oracle for this environment is realized as follows: The ratio of all colors (i.e., actions) covered by the queried classifier's condition in the problem space is determined. Thus, a relative frequency (probability) distribution over the colors is obtained. Whenever not a single color covers 90% of the requested area, the query is rejected by the oracle. Otherwise, the majority color is returned as answer. XCS is configured as follows: $N = 7000$, $\alpha = 0.1$, $\beta = 0.3$, $\delta = 0.1$, $\nu = 5$, $\theta_{GA} = 30$, $\epsilon_0 = 10$, $\theta_{mna} = 6$, $\theta_{del} = 50$, $\theta_{sub} = 50$, $\chi = 0.8$, $\mu = 0.04$, $p_{ini} = 10.0$, $\epsilon_{ini} = 0.0$, $F_{ini} = 0.01$, $r_0 = 0.1$, $m_0 = 0.1$. Again, the interval-based UBR condition representation is used. Uniform crossover and tournament selection is used with $\tau = 0.4$. Action mutation is not used. GA subsumption is active.

Short Discussion of Preliminary Results Examplary plots depicting the learning curves for both scenarios are shown in Figures 10.3 to 10.6. Additionally, the statistics (means and standard deviations) obtained after conducting the experiments are summarized in the Tables C.4 and C.5 which can be found in Appendix C. Since this chapter is mainly intended as an elaborate discussion on promising future research directions, a more thorough discussion of the results is omitted here. For a detailed comparison and a thorough report on the results, the interested reader is referred to [SMH17].

As is hypothesized, the main benefits of the outlined AL techniques appear at the beginning of the investigated learning tasks. QBF as well as the related QBC approach noticeably reduce the number of needed macroclassifiers. At the same time a significantly decreased system error can be obtained. This effect is observable

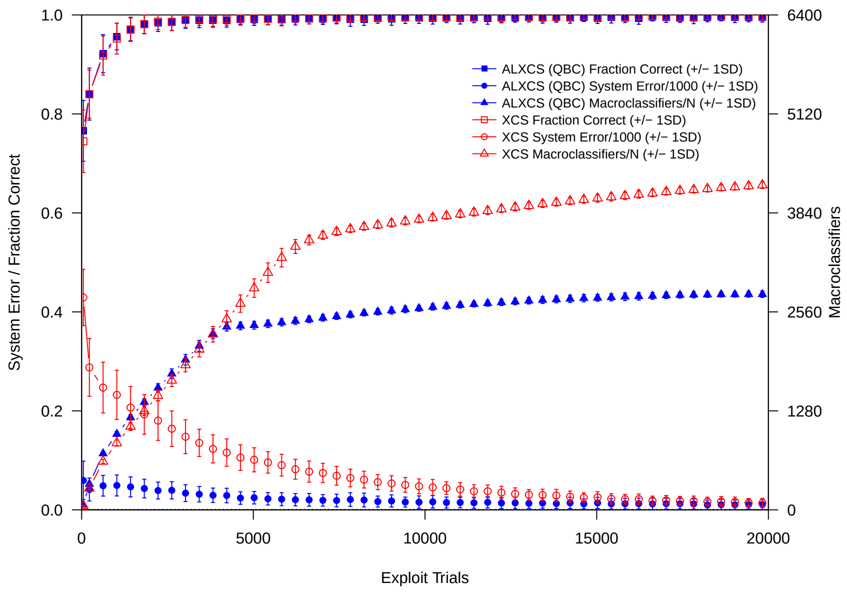


FIGURE 10.3.: Learning curves of ALXCS (blue lines) using QBC with $\theta_{sim} = 0.20$ and XCS (red lines) over the first $20k$ trials on the WBC classification problem

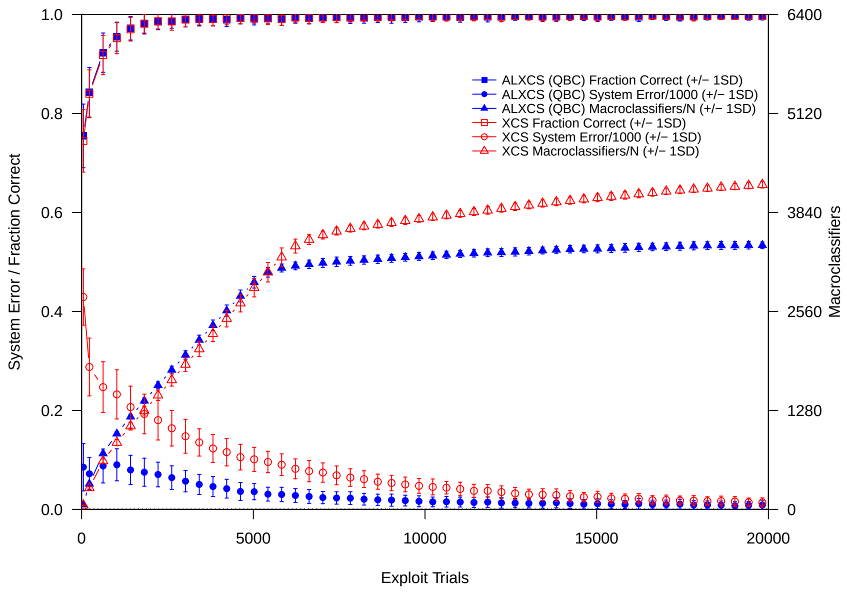


FIGURE 10.4.: Learning curves of ALXCS (blue lines) using QBC with $\theta_{sim} = 0.01$ and XCS (red lines) over the first $20k$ trials on the WBC classification problem

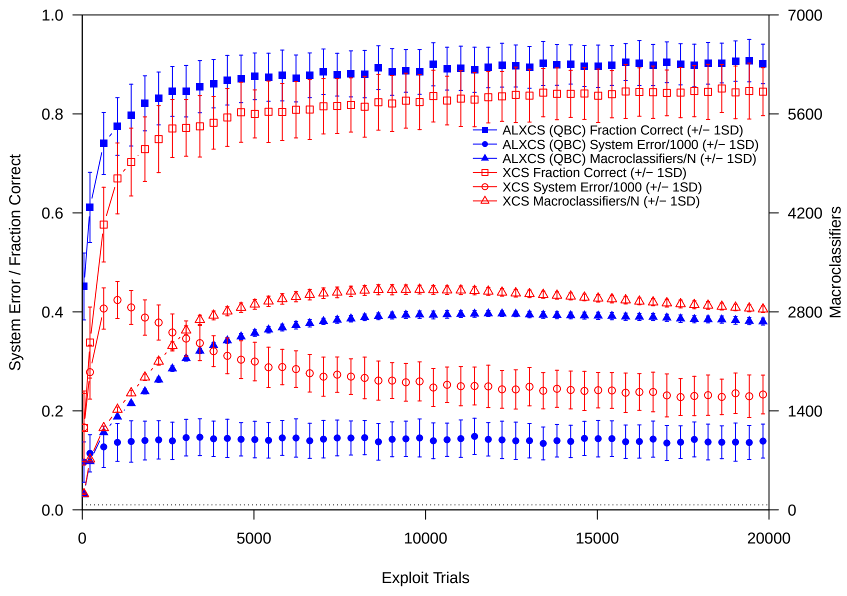


FIGURE 10.5.: Learning curves of ALXCS (blue lines) using QBC with $\theta_{sim} = 0.20$ and XCS (red lines) over the first $20k$ trials on the Mario toy problem

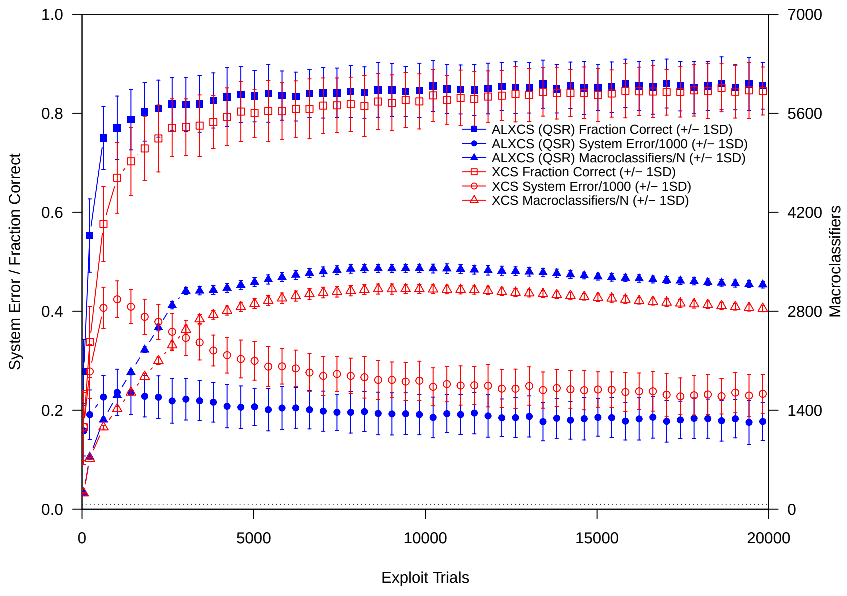


FIGURE 10.6.: Learning curves of ALXCS (blue lines) using QSR and XCS (red lines) over the first $20k$ trials on the Mario toy problem

for both scenarios when the entire learning process over $100k$ exploit trials is considered. The fraction of correct classifications (classification accuracy), however, is marginally reduced from a statistical perspective throughout all WBC experiments when QBF and QBC is incorporated. However, overall it can be found to stay on nearly the same level as can be seen in the corresponding plots.[2]

Regarding the QSR technique on the WBC problem, as of yet only negative impacts on all three figures of merit have been observed. This underpins the general disadvantage of query synthesis recognized in the literature as briefly discussed at the beginning of this chapter. The significantly increased average population size is deemed to result from the randomized queries (i.e., conditions). These seem to be selected too arbitrarily to yield a performance gain. The population is expected to be "swamped" with random classifiers which, due to the highly set initial values for fitness and predicted reward, exerts misleading fitness pressure to the evolutionary process. A more thorough investigation on the impacts on the actual classifier evolution is subject of future work.

For the Mario environment, not surprisingly, positive effects of the QSR technique can be observed. Since the problem domain is sampled uniformly it can be expected that any region of X is visited equally likely. Thus, as is hypothesized, the proactive construction of random knowledge turns out to be useful (see Fig. 10.6). The average accuracy as well as the system error measures have been found to significantly improve in contrast to standard XCS. However, at the expense of an increased average population size. Although the briefly discussed preliminary results on QSR appear to be a sort of double-edged sword with regard to the two investigated scenarios, proactive classifier construction is still deemed a powerful approach to overcome severe challenges such as drifting input distributions and highly imbalanced data streams. The following Section 10.2 outlines the idea for a more sophisticated method to decide in which region of the knowledge base $[P]$ classifiers are missing and a synthesis might be valuable.

In the exemplarily depicted learning curves, two different values for the similarity tolerance θ_{sim} are reported. θ_{sim} controls the committee voting conflict which is required to trigger an interaction with the oracle. Apparently, the higher value of $\theta_{sim} = 0.2$ yields better results since the oracle is queried more frequently. Additionally, this value is set to a reasonable small value, i.e, $\theta_{sim} = 0.01$, to obtain first insights on the effect of a drastically reduced number of oracle interactions. It turns out that the positive impacts persist for all three figures of merit, even if the number of oracle queries is significantly reduced by $\approx 76\%$ and $\approx 92\%$ for the WBC setting and the Mario environment, respectively. This constitutes an interesting result since it indicates that even with a small fraction of the learning steps at which an oracle is asked for assistance, still significant performance improvements can be obtained.

[2]Please note that in the Figures 10.3 and 10.4 the developments regarding the fractions of correct classifications for both XCS variants are nearly congruent what leads to a somewhat hidden (red) curve indicating standard XCS.

To consider the aspect of human uncertainty, another rudimentary experiment has been conducted with a human oracle taken into the learning loop. Again, improved performance over all configurations and for all metrics can be observed. Interestingly, the best configuration revealed to be QBC with a similarity tolerance $\theta_{sim} = 0.01$, which resulted only a low interaction load for the human oracle. For details, please refer to [SMH17].

10.1.3. Discussion

The objective of this section is to present the vision of (pro-)active learning classifier systems. It is outlined how XCS can be endowed with a form of *curiosity* resulting in an *intrinsic* desire to learn more about uncertain regions within the current knowledge base. Furthermore, it is demonstrated that proactive knowledge construction, i.e., building classifiers prior to the first request from the incoming data, can have beneficial effects. In particular, at the beginning of a learning task, the proposed technique called QSR which selects target regions for new classifiers completely at random, indicated positive effects under the assumption of uniform input space sampling. Although first promising results can be reported, it explicitly noted that the experiments are yet preliminary and, thus, limited in at least two aspects: (1) Only stationary, uniform sampling of the problem's input domain have been considered so far. This implies that KGs mainly occurred at the beginning of the respective learning tasks, what contradicts the idea of a proactive knowledge construction for later appearing KGs. (2) At least for the reported QBF and QSR query strategies, the AL component was activated in each learning step, what results in a noticeable interaction load with the oracle. The latter aspect mainly constitutes a problem when a human oracle is brought into the learning loop. However, as a first countermeasure, the QBC approach, which resulted in a significant reduction of the actual oracle load while still yielding significant performance improvements, has been adopted for utilization within XCS.

Future Work The promising preliminary results achieved so far steer future research efforts toward a more thorough investigation of the presented techniques with a specific attention on:

1. The number of actually necessary oracle interactions to achieve significant improvements.
2. The degree of oracle confidence required in order to expect beneficial effects
3. Idiosyncratic cognitive characteristics of human oracles, such as reluctance, overconfidence and volatile concentration
4. An in-depth inspection of the impacts on the classifier evolution in terms of e.g., generality, experience and mean lifetime.

The concept of the (pro)active knowledge construction will furthermore be combined with the transductive knowledge inference techniques as have been developed in the main part of this thesis.

So far, ALXCS was tested on classification tasks only. In a next step, it is planned to transfer the presented techniques to support the function approximation capability of XCSF. In general, all techniques introduced in this section are straightforwardly transferable to other LCS derivatives such as UCS [BG03] or ExSTraCS [UM15]. Another branch of investigation is planned to examine the use of ALXCS within sequential decision making tasks as presented in [NSC15a] or even on more complex scenarios such as coverage optimization in self-configuring smart camera networks [Ste+17b].

10.2. From Active to Proactive Knowledge Construction

In the preceding section, first insights regarding the envisaged future research aspiration to render LCS AL algorithms have been subject of discussion. Inspired by Angluin's *memberhship query synthesis* approach, one of the presented methodologies, QSR, already constitutes a first step ahead from a purely *active* toward a more *proactive* means of constructing and adding novel knowledge elements k^* to an algorithm's current knowledge base K at a certain point in time t (denoted by K_t in the following). However, the preliminary experiments revealed both positive and negative impacts on XCS's learning performance, depending on the characteristics of the learning problem at hand. For instance, in the WBC scenario arbitrarily created classifiers spread randomly over the entire input space lack the property of *relevance*. This, however, is a central assumption in the initially formulated hypothesis. The data distribution is completely determined by the available labeled data set. However, this data set is in turn randomly iterated over multiple epochs (as required by an online learning system such as XCS). Thus, classifiers created in regions within the input space X that are not hit by at least one instance from the labeled data set $\mathfrak{L}$ are essentially useless and waste population resources.

The main objective of this section is to push the idea of curious and self-reflecting LCS further by introducing the novel concept of *Proactive Knowledge Construction*. This concept is based upon the formerly discussed AL methodology. It is intended to endow OML algorithms with the ability to more purposefully construct knowledge elements, i.e., in the context of LCS, creating classifiers at insufficiently covered regions that are deemed to gain *relevance* in the near future. In the end, this idea is hypothesized to increase the *learning efficiency* as well as the *robustness* of SLAS. Under this crucial *relevance assumption*, $L(K)$ is expected to allow for more accurate responses as would be possible through arbitrarily initialized or even randomly created $k \in K$. Essentially, this decreases the initial prediction error and further leads to an improved discovery of these potentially relevant problem space niches.

In this thesis it is focused on incrementally learning algorithms $L(K)$ which update their knowledge bases K every time a new situation $\sigma = \vec{x} \in X$ arrives. Thus, it constitutes a non-trivial task to determine KGs, since the knowledge bases K evolve every time $L(K)$ is active. For the sake of introducing a generally applicable concept, in the following it is abstracted from concrete LCS implementations. Instead, a *generic incrementally knowledge acquiring learning algorithm* $L(K)$ serves as a basis for discussion. An exemplary generic $L(K)$ is outlined in Algorithm 1:

Algorithm 1 Incrementally knowledge acquiring learning algorithm $L(K)$

```
Input: $K_t, \vec{x}_t$                              ▷ Knowledge base and input situation at time $t$
Output: $K_{t+1}$                                    ▷ Updated knowledge base
 1: procedure LEARN($K_t, \vec{x}_t$) ▷ Take new input $\vec{x}_t$ and update knowledge base $K_t$
 2:     $K' \subseteq K_t \leftarrow$ANALYZE($K_t, \vec{x}_t$)          ▷ Find matching knowledge for $\vec{x}_t$
 3:     if $K' = \emptyset$ then                               ▷ Reactive Knowl. Constr.
 4:         $k_{new} \leftarrow$ CREATENEWKNOWLEDGE($\vec{x}_t$)
 5:         ADDTOKNOWLEDGEBASE($K_t, k_{new}$)
 6:         $K' = \{k_{new}\}$
 7:     end if
 8:     // Start of novel proactive knowledge construction part
 9:     if $t \mod \theta_P = 0$ then                         ▷ Proactive Knowl. Constr.
10:         $kg \leftarrow$ IDENTIFYKG($K_t$)
11:         $k_{pro} \leftarrow$ CLOSEKG($kg$)
12:         ADDTOKNOWLEDGEBASE($K_t, k_{pro}$)
13:         $K' \leftarrow$ REANALYZE($K', k_{pro}, \vec{x}_t$)        ▷ Check if $k_{pro}$ also matches $\vec{x}_t$
14:     end if
15:     // End of novel proactive knowledge construction part
16:     $a_t \leftarrow$ PREDICT($K', \vec{x}_t$)
17:     $u_t \leftarrow$ EVALUATEOUTPUT($a_t$)
18:     UPDATEEXISTINGKNOWLEDGE($K', a_t, u_t$)
19:     $K_{t+1} \leftarrow K_t$
20: end procedure
```

In the following, it is proposed to adapt existing algorithms which follow such an algorithmic blueprint as follows: Let the algorithms $L(K)$ (or shortly L) pursue the complementary goal of implicit or explicit minimization of the magnitude of the space of knowledge gaps $|KG|$.[3]

The achievement of the aforementioned goal involves proactive construction of knowledge in those areas of the problem space PS for which $L(K)$ has not acquired knowl-

[3]Since this proposition is quite formal in the beginning, the reader is referred to the definitions of knowledge gaps in Section 3.3.

edge elements k_i so far (type-1 KGs). Additionally, it involves the improvement of knowledge elements $k \in K$ that are already part of the knowledge base K, but have not yet reached the targeted quality and experience levels θ_q and θ_{exp} (type-2 KGs).

With regard to the introduced notion of incrementally learning algorithms and the definitions of knowledge and knowledge gaps in Section 3.3, the following proposition is posed:

Proposition 1. *Assume a base algorithm L that incrementally acquires and adjusts knowledge elements $k_i \in K$ in response to observed instances from a problem space PS. Extending L to a proactive knowledge constructing learning algorithm L^* which is able to self-reflect its knowledge base in order to identify and close knowledge gaps $kg \in KG$, leads to improved knowledge bases K^* in terms of (1) faster and higher coverage of the problem space PS, and, (2) knowledge $k \in K^*$ of increased average quality. Thus, the magnitude of knowledge gaps $|KG|$ decreases faster for L^* than for L and the average quality $\bar{q}$ of L^*'s knowledge base K^* increases more quickly with progressing time steps t.*

Sketch of proof for Proposition 1 It is now formally proved that this proposition holds true under clearly stated assumptions. Therefore, Proposition 1 is divided into Lemmata 1 and 2, where each targets one of the two main conjectures, i.e., (1) the faster coverage of PS, and (2) higher average quality of knowledge.

Lemma 1. *Under the assumptions of (1) an empty knowledge base K_t at $t = 0$, (2) a finite K, i.e., $|K| \leq N, N \in \mathbb{N}$, (3) a maximum number n of learning steps $t = 0 \ldots n$, as well as, (4) a finite knowledge space KS, the remaining magnitude of knowledge gaps for L^* after $t = n$ steps, denoted by $|KG^{L^*}_{t=n}|$, will be smaller than for L. Further consider that L^* additionally constructs $m \in \mathbb{N}$ knowledge elements k_{pro} proactively, each time L constructs a single new k_i in a reactive fashion. W.l.o.g. it is assumed that at each time step t, exactly 1 knowledge element is constructed by L.*

Proof. The number of knowledge elements in K^*_t of L^* at time $t > 0$ will always be $m + 1$ times higher than for K_t of L, i.e., $|K^*_t| = (m + 1) \cdot |K_t|$. According to the definition of type-1 knowledge gaps, i.e., $KG_1 = KS \setminus K$, it follows that the magnitude of KG_1 is given by $|KG_{1,t}| = |KS| - |K_t|$. According to the simplification stated above, $|K_t|$ can be replaced by t for L, and by $(t \cdot m) + t = (m + 1)t$ for L^*. Thus, if $t \to n$, $|KG^L_{1,t}| = |KS| - n$ whereas $|KG^{L^*}_{1,t}| = |KS| - (m + 1)n$. It clearly follows that $|KG^{L^*}_{1,t}| < |KG^L_{1,t}|$ holds true for $t > 0$. $\square$

Lemma 2. *The average quality $\bar{q}$ of the knowledge base K^* of L^* will always be higher than for L when knowledge gaps of type 2, $kg_j \in KG_2$, are proactively closed over time t. Additionally, it must hold that for the proactively generated knowledge* *element k_{pro}, which replaces the identified gap $kg_j \in KG_2 \subseteq K$, the quality condition $q_{pro} \geq q_j + \epsilon$ is satisfied. With $\epsilon \in \mathbb{R}^+$, q_{pro} denoting the initial quality value of the proactively created knowledge element k_{pro}, and q_j being the current quality estimate of the identified type-2 gap $kg_j \in K$.*

Proof. Following the definitions $KG \coloneqq KG_1 \cup KG_2$, and $KG_2 \coloneqq \{k_j \in K \mid q_j \leq \theta_q\}$, the average quality of K can be written as[4]

$$\bar{q} = \frac{1}{|K|} \sum_{k_i \in K} q_i.$$

As K can be rewritten as $K' \cup KG_2$, where $K' \coloneqq K \setminus KG_2$, $\bar{q}$ can be rewritten as:

$$\bar{q} = \frac{1}{|K'|} \sum_{k_i \in K'} q_i + \frac{1}{|KG_2|} \sum_{k_j \in KG_2} q_j,$$

Since $q_i > \theta_q$ and $q_j \leq \theta_q$ for each $k_i \in K'$ and $k_j \in KG_2$ holds, it follows that by proactively decreasing $|KG_2|$ by the same means as proven for Lemma 1, the average quality $\bar{q}$ of K will increase faster for L^* than for L. This holds true even if kg_j remains in KG_2 after the last proactive construction cycle, i.e., the $q_j > \theta_q$ criterion is not satisfied yet, since only $q_{pro} \geq q_j + \epsilon$ is required by assumption. □

The abovestated proof sketch corroborates the reasonableness of the outlined concept formally. However, the *initial quality* of the proactively constructed knowledge elements plays a central role. This initial quality is strongly dependent on the oracles in the learning loop. As already discussed in the preceding Section 10.1, different kinds of oracles can be taken into account ranging from human experts over existing simulation or formal models to the use of the IC for transductive knowledge inference as introduced in this thesis. Without the assumption of qualitative responses from at least either of those variants, also the benefits of the introduced concept of proactive knowledge construction are hardly accomplishable.

10.2.1. Enhancing the MLOC Architecture

Figure 10.7 integrates the two-layered blueprint for ALXCS as depicted in Figure 10.1 with the MLOC architecture from the OC domain (see Ch. 3). As can be seen, the first layer still contains an observer/controller tandem. The controller of layer 1 is simplified to comprise an online learning algorithm maintaining a knowledge base K_t over time t. This knowledge base serves as a basis for the newly introduced situation analysis mechanism located at the observer of the first layer.

[4]Please note that for the sake of simplicity and w.l.o.g. the second criterion of $exp_j \leq \theta_{exp}$ is omitted here.

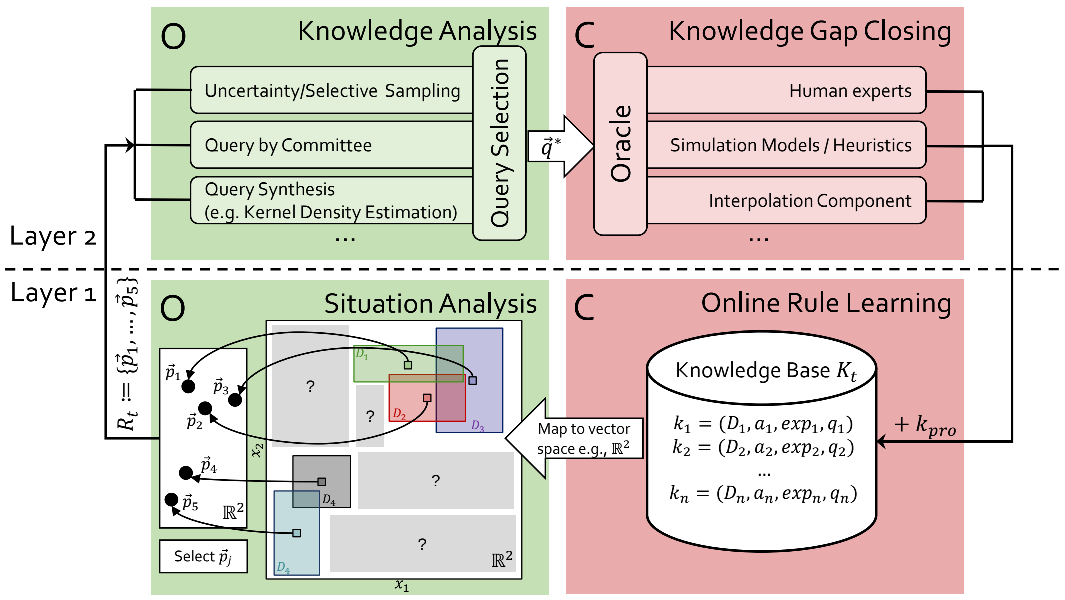

O
Knowledge Analysis
Uncertainty/Selective Sampling
Query by Committee
Query Synthesis (e.g. Kernel Density Estimation)
...
Query Selection
$\vec{q}^*$
C
Knowledge Gap Closing
Oracle
Human experts
Simulation Models / Heuristics
Interpolation Component
...
Layer 2
Layer 1
$R_t := \{\vec{p}_1, \dots, \vec{p}_5\}$
O
Situation Analysis
Select $\vec{p}_j$
Map to vector space e.g., $\mathbb{R}^2$
C
Online Rule Learning
Knowledge Base K_t
$k_1 = (D_1, a_1, exp_1, q_1)$
$k_2 = (D_2, a_2, exp_2, q_2)$
$k_n = (D_n, a_n, exp_n, q_n)$
$+ k_{pro}$

FIGURE 10.7.: Schematic showing a possible integration of the Proactive Knowledge Construction mechanisms into layers 1 and 2 of the MLOC architecture

The second layer is realized as another O/C pair devoted to the task of knowledge analysis and closing of identified KGs. In order to provide the system with the capability of proactive knowledge construction, in a first step the current knowledge base K_t is translated into a vector space. This enables the observer of layer 2 to apply several knowledge analysis mechanisms. The output of this process is a selected query $\vec{q}^*$, which is deemed to obtain high benefit when being handled by at least one of the available oracle models in the corresponding controller. At this place, a new knowledge element k_{pro} is constructed and finally fed back to the knowledge base at layer 1, what brings the just delineated proactive knowledge construction cycle full-circle.

These steps are deemed necessary in order to equip SLAS with self-reflection and proactive knowledge construction capabilities. Initial thoughts of how they can be realized are briefly described in the next paragraphs.

Mapping K into a vector space R

In a first step, the system itself needs to be enabled to perform computations on the basis of its current knowledge base K_t. As Figure 10.7 illustrates, the required conversion of K_t into a vector space is situated at the first layer's observer component of the MLOC-based system. The involved *Online Rule Learning* algorithm (as exemplarily outlined in Algorithm 1) passes the current K_t to the neighboring observer. As defined in Section 3.3, each $k_i \in K_t$ is assigned a subspace $D_i \subset X$

of the input space as well as an action element a_i (or another target variable). In combination, $D_i \times \{a_i\}$ comprises a certain subspace of the problem space PS. Since PS is usually very huge in realistic applications, it might be the case that not every possible input-output pair $(\vec{x}, a) \in D_i \times \{a_i\}$ of all knowledge elements $k_i \in K_t$ can be incorporated into the analysis process for the sake of ensuring efficient computation. Thus, a mechanism to select *representative knowledge vectors* $\vec{p}_j \in D_i$ from all $k_i \in K_t$ is needed. On the basis of the selected representatives $R_t \coloneqq \{\vec{p}_j\}$, inferences regarding the coverage of the theoretical knowledge space KS in terms of its underlying input space X can be drawn. Naively, a vector near the center of the geometric shape determined by D_i might be selected as one representative, as depicted in Figure 10.7. Intuitively this would, however, neglect the entire surrounding area/volume of the knowledge element k_i as determined by D_i and, thus, only yields a very rough approximation of the actual coverage of X. Nonetheless, for the sake of simplicity and the scope of this chapter on future research, this mechanism is further assumed in the following. With that, a first possibility to seek for type-1 knowledge gaps is provided. Please note that it is assumed that the selected representatives $\vec{p}_j$ still carry references to their corresponding knowledge elements k_i. This permits access to the remaining knowledge attributes a_i, exp_i, q_i and, thus, facilitates the search for knowledge gaps of type 2. As described in Section 3.2, the input spaces X are assumed to be vector spaces, typically Euclidean. Accordingly, all subspaces of X are vector spaces and, thus, the selected representative elements $\vec{p}_j \in R$ can be regarded as vectors as well. Further steps are based on this assumption.

Knowledge Gap Identification

With the set of vectors R_t representing the system's current knowledge at time t, a SLAS is now enabled to apply different computations for identifying underrepresented niches in the current knowledge base K_t. As Figure 10.7 suggests, the idea of using AL techniques is carried on. In order to facilitate a more purposeful KG search, beyond simply creating knowledge elements by chance as done with QSR before, in the following another approach based on *Kernel Density Estimation* (KDE) [SS05] is outlined.

KDE is a means to estimate the *probability density function* (pdf) of a random variable based on a sample of concrete realizations. For the purpose of utilizing this technique in the context of proactive knowledge construction, the random variable is assumed to be multivariate and thus can be mapped to the input space X. Accordingly, the pdf of X yields a measure of how densely the input space is populated by knowledge elements $p_j \in R_t \subset X$ at time t. The general formula of a kernel density estimator is given by:

$$\hat{f}_X(x) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{K}_h(x - x_i) \tag{10.8}$$

Here, $\mathcal{K}_h$ denotes the *kernel* parameterized with its *bandwidth* h, which is a smoothing parameter for scaling purposes. $\mathcal{K}_h$ can be a hypercube kernel, the *Parzen window*, or the Gaussian kernel, for instance. Beside the univariate definition (Eq. 10.8), there also exist variants for multivariate estimators which are needed in the present case. Let X be a d-dimensional continuous random variable. Assume further n samples $\vec{x} \in \mathbb{R}^d$. Then, the multivariate KDE can be estimated using the product kernel [SS05]:

$$\hat{f}_X(\vec{x}) = \frac{1}{n} \sum_{i=1}^{n} \prod_{k=1}^{d} \frac{1}{h_k^d} \mathcal{K}_h \left(\frac{\vec{x}^{(k)} - \vec{x}_i^{(k)}}{h_k} \right) \tag{10.9}$$

Again, $\mathcal{K}_h$ is a univariate kernel parameterized by the smoothing bandwidth h, which in this case can differ for each dimension $k = 1 \ldots d$ (denoted by h_k). More sophisticated multivariate KDEs exist, e.g., based on multivariate Gaussian kernels. For the sake of simplicity, here only a methodology which is more straightforward to implement is outlined.

With this methodology to estimate the density at arbitrary points in a vector space at hand, the next step is to adopt it to the set of knowledge representing vectors R_t. All $\vec{p}_j \in R_t$ are used as samples $\vec{x}_i$ for the KDE. Next, let $Q := \{\vec{q}_l \mid \vec{q}_l \in X\}, l = 1 \ldots m$ be a set of m query locations where the '*knowledge density*' $kd(\vec{q}_l)$ is to be estimated. This can be done by means of substituting $\vec{x}$ with $\vec{q}_l$ in the multivariate density estimation given by Equation 10.9. Thus, n in Equation 10.9 becomes $|R_t|$ and $\vec{x}_i$ will be $\vec{p}_j \in R_t$. The query locations $\vec{q}_l$ can be selected completely at random, or grid-like via partitioning the input space in predefined bins. Bearing the *curse of dimensionality* in mind, the number m has to be selected properly with regard to the dimensionality of X. Finally, the query location $\vec{q}^*$ is selected via

$$\vec{q}^* := \underset{\vec{q}_l \in Q}{\operatorname{argmin}} \; kd(\vec{q}_l). \tag{10.10}$$

This sort of greedy approach results in a strong exploratory behavior. This means that each time the outlined proactive learning cycle is executed, it is very likely that a knowledge element is constructed in an entirely uncovered region. This seems a legitimate way to eradicate type-1 knowledge gaps. Apparently, the query selection strategy is one of the most crucial decisions of the proactive knowledge construction process. This is due to the fact that it initiates the construction of knowledge in regions that are assumed to become relevant in the near future (see also the discussion at the end in Sect. 10.2.2). An alternative way to decide on knowledge gaps to be closed would be to select queries $\vec{q}^*$ in regions of *moderate density.* In such regions, knowledge has already been acquired in the course of learning (most likely reactively), which suggests relevance of this niche. Finding more sophisticated ways to determine the most valuable gap to be closed constitutes a major aspect of future research.

Knowledge Gap Closing

The logical next step after knowledge gaps have been identified by the *knowledge analysis* mechanism in the observer at layer 2 of an MLOC-based system is to find appropriate ways to 'bridge' those gaps. In Figure 10.7 it is suggested to extend the controller situated at layer 2 toward a more general *knowledge gap closer*. The envisioned main task of this controller instance is to serve as the (artificial) *oracle* which can be realized by various means, as discussed before.

Putting Humans into the Learning Loop The first obvious choice for an oracle is a that of a *human expert*. One requirement to realize an adequate involvement of humans in the learning loop is to define an intuitive *human-machine-interface* which can be accessed by a domain expert. These experts can be physicians in a medical context or traffic engineers in a traffic control scenario, for instance. On the one hand, these experts can be asked to provide entire, hand-crafted knowledge elements k_{pro} in form of rules comprising their expertise with the task at hand. On the other hand, classification labels or other target variables can be requested from time to time on demand. Challenging aspects that need thorough attention are: (1) Latency in response when the human oracle is not immediately available or answering the query takes time (e.g., experiments need to be conducted). (2) Idiosyncratic human factors such as reluctance, uncertainty and presumed benevolence [DC08].

Model- and Simulation-based Approaches The incorporation of computational models and optimization (meta-)heuristics into the MLOC architecture has already been demonstrated to be beneficial in various domains such as network [TH11] and traffic control [Pro+09]. In such convenient situations, where computational simulations or mathematical models describing the complex problem at hand exist, the resulting simulation-optimization tandem can be adopted for the purpose of knowledge gap closing. Strictly mathematical models provided as formulas are referred to as *heuristics* to such problems. Such heuristics allow for solving the underlying problems more efficiently in contrast to computationally expensive simulation tools. In terms of closing an identified knowledge gap, the selected query $\vec{q}^*$ is taken as argument for a heuristic or as starting point for a simulation. In either case, the system utility u_{t+1} serves as objective/fitness function the applied optimizer attempts to maximize by seeking the most appropriate action or target value (solution candidate). In other domains such as machine learning, such models are also known as *surrogate models*. Such models can themselves be subject to learning, i.e., approximations of the input-output relationships can be constructed and continually improved on the fly in the course of the system's runtime. One methodlogy that resembles this approach has been presented in Chapter 5 of this thesis. One advantage of simulation models and heuristics is their lesser time effort compared to waiting for respective observations in the real world. Another advantage is the omnipresence and non-reluctance of such artificial oracles in contrast to human oracles, which might be confused and negatively affected by untypical queries. However, the advantages often come at the expense of limited expressiveness and accuracy what needs to be taken into account when designing proactively learning systems.

Transductive Inference based on Interpolation A third method to close identified knowledge gaps has been subject of thorough investigation in the course of this thesis – the direct, i.e., transductive inference of knowledge elements from existing ones in their neighborhood. Consider an *inner* knowledge gap kg_{in} of type 2 as defined in Section 3.3. Such a kg_{in} per definition lies within the convex hull $conv(K_t \setminus KG_{2,t})$ and is thus surrounded by knowledge elements k_i with a sufficient degree of quality $q_i > \theta_q$ and experience $exp_i > \theta_{exp}$. These knowledge elements k_i can therefore be used to construct sampling points based on which an interpolation can be calculated. More concretely, it is proposed to construct sampling points $sp_k := (\vec{x}, a)$ with properly selected sampling point coordinate vectors $\vec{x} \in D_i \subset X$, and their actions $a_i \in A$ serving as function values to be interpolated such as introduced in Chapter 5 for the ASI technique. This permits the transduction of the actions a^* for the identified query q^*, with which in turn a novel knowledge element k_{pro} can be constructed. The same methodology can also be applied to interpolate the quality estimate $k_{pro}.q$ for instance, which can be understood as a form of knowledge *bootstrapping*. This concept can be straightforwardly extended in order to allow for *extrapolation*. Therefore, only the condition that the queried gaps have to lie within the convex hull of K needs to be relaxed. However, this modification necessitates powerful techniques with reasonable properties regarding their extrapolation characteristics to prevent misleading or even disruptive transductions. The transductive approach is deemed advantageous since no additional effort is required for requesting human experts or for the development or appropriate configuration of analytical models and computational simulations, respectively, for the problem at hand. Instead, it is possible to simply rely on existing knowledge elements within the system's current knowledge base K_t.

Thorough investigations that compare the outlined oracle methodologies on various scenarios constitute a promising direction of future research.

10.2.2. Discussion

An inevitable question that should be discussed is:

When, how and where can proactively constructed knowledge be considered sensible?

Regardless of whether the general rationale for incorporating the just introduced novel concept of proactive knowledge construction proofs plausible or not, the elaborated ideas rely on a few assumptions that are subject of the following discussion.

Problem Space and Knowledge Relevance A major assumption implicitly made is that each existing niche in the problem space PS is *equally relevant* and, thus, needs to be fully covered in order to obtain the desired system robustness and performance. Carrying on this assumption, the PS should not be deemed to only contain relevant and sensible combinations of situations and actions. It strongly depends on the actual input space X, which in case of many SLAS is determined by the ranges of e.g., the sensors and inner configurations, whether obscuring feature combinations are possible and thus can be presented to the oracles. It is reasonable to not assume that any niche within the problem space has to be covered to obtain a desired utility level. Consider e.g., certain combinations of sensor readings that are theoretically possible but practically (or physically) irrelevant. An automatic way to figure out the important and valuable regions in PS and to rank the identified knowledge gaps in terms of their relevance is an important aspect of future research. A first solution to that problem might be to interpret the convex hull of K_t as the area of relevance within PS at time t. On that basis it would be possible to restrict the potential queries to knowledge gaps which are inner knowledge gaps, i.e., $kg \in KG_{in}$, and to those which are in the direct proximity to the boundaries of $conv(K_t \setminus KG_{2,t})$. A similar behavior can be realized by the proposed KDE approach, when the selection criterion is set to the medium (average) instead of the lowest density, for instance.

Oracle Reliability, Availability and Benevolence A further assumption on which the concept relies is the omnipresence and omniscience of the knowledge gap closing mechanisms at layer 2, at least to a certain degree. Most notably, a central assumption is that each time a type-2 knowledge gap is identified, the newly created k_{pro} at least provides a marginally higher quality. As already recognized by Baum and Lang in [BL92], synthesized queries can obscure human oracles what leads to a rejection of the queries or, even worse, to answers with insufficient confidence and quality. Furthermore, applied simulations can be imprecise as can be heuristics for certain inputs. The corresponding optimization heuristics can get stuck in local optima when the underlying fitness landscape is complex, what also might affect the reliability of artificial oracles. In future work, model parameters for e.g., oracle uncertainty and absence, as proposed by Donmez et al. in [DC08], will be considered in order to account for this aspect.

Knowledge Handling of Base Algorithms $L(K)$ So far, the proposed concept is mainly thought of as extension to existing algorithms of a specific kind –

rule-based online learning algorithms that incrementally build up their knowledge bases K. In the context of online learning, however, specific challenges such as the *plasticity-stability dilemma* [DP13] have been recognized for a long time. The dilemma essentially poses the question of when and how much of *old* knowledge (which might have become obsolete in the course of learning due to appeared concept drifts) can be sensibly forgotten in favor of novel, kind of fresh knowledge that potentially better approximates the underlying problem function F_t at the current time t. The means of how the considered base algorithms handle this dilemma clearly affects the outcomes of proactive knowledge construction. In order to thoroughly analyze the impacts of different kinds of *concept drifts* [Web+16], it is planned to design particular benchmark problems. In that way, revealing insights regarding the development of knowledge bases K and the set sizes of KG_1 and KG_2 can be obtained, e.g., by means of visualizing the convex hull $conv(K)$ over time. As a further milestone, the adoption of the proposed concept in several online machine learning algorithms such as XCS, online learning bagging and boosting algorithms, or Gaussian mixture models, appears to be a logical next step. In addition to that, the development of entirely new *knowledge gap-centric* learning algorithms, with specifically designed mechanisms and policies that account for the abovementioned aspects, seems to be an interesting branch of further research.

10.3. Related Work

10.3.1. Active and Human-Assisted Learning in LCS

In an early attempt reported in a 1999 paper of Xi et al. [XLY99], an approach to active learning for classifier systems is proposed. Even if the title suggests a strong relation to the work presented in this chapter, by having a closer look it turns out that their approach differs fundamentally. The authors redefine the term ‘active learning’ for their purposes, although the concept of AL already appeared prior to that work in the literature. In essence, what is proposed in this work is that an artificial agent modeled by a ZCS classifier system is extended to generate, recognize and process *signals* (might be interpreted as pheromones) that are spread over the environment and lose *intensity* over time. These signals can then be perceived and used to manipulate the learning process. Via so-called *tuning* and *damped strength* operations the signals influence the agent’s perception and credit assignment. This in turn affects its moving decisions in a way that cells with low or no signals are visited more likely. The result is an increased efficiency when solving animat problems such as Woods1 and Woods7.

Injecting knowledge from external sources is not a novel means to enhance the performance of LCS. As already mentioned in the section on related work in Chapter 6, Urbanowicz et al. propose to use expert knowledge to guide the discovery processes of LCS in [UGM12a; UM15]. This approach differs from the concept of ALXCS in that probabilities to influence the specificity of classifiers newly generated by covering or the GA are derived in a preprocessing step based on the a priori available training data. For the outlined work on combining XCS with AL techniques, the learning process of XCS is guided online, i.e., during the actual learning phase.

In [NSC15b; NSC15a], Najar et al. present the *Socially Guided XCS* and the *Social-Value XCS*. The authors introduce a model-based learning approach, where the task model is supported by a social model as well as by a *contingency model* which serves as a bridge between the former two. Their system is applied to a multi-step RL task where a robot is asked to decide which button to press according to a visual stimulus. A teacher can guide the robot's decision by means of *teaching signals*, e.g., pointing to the correct button. The authors found that the learning success can be improved in terms of the total amount of received reward, the number of steps until the robot finds the correct button, as well as the compactness of the evolved rule-base. In contrast to the method proposed in this thesis, the Socially Guided XCS is a model-based reinforcement learner that may, or may not be taught by a human teacher. Here, XCS is enabled to be curious about environmental niches where it could not gain a sufficient level of experience so far. Furthermore, the external source of knowledge is not necessarily assumed to be a human teacher, but can also be realized by artificial oracles.

10.3.2. Anticipatory Classifier System

Anticipatory Classifier Systems are due to Stolzmann [Sto99] (ACS) and Butz [But02] (ACS2). These systems are regarded *anticipatory* because they build up a model of the environments in which they are asked to act, more precisely of the underlying state transition function, what in turn facilitates planning. Due to their model-learning and subsequent planning nature, they are reminiscent of Sutton's Dyna approach [Sut91]. ACS replace the *condition-action* rule representation by a *condition-action-effect* representation [SW07]. Thus it facilitates a one step look-ahead for all classifiers that match the current situation what in turn allows to bias the action-selection mechanism. ACS constitute an interesting and psychologically inspired descendant of LCS. However, the relation to the outlined research direction of (pro-)active LCS is somewhat limited so far. Nevertheless, the model which is learned during the online interaction with an environment can be used to simulate future state transitions. This can potentially be utilized to obtain hints in which niches of the problem space a proactive construction of knowledge can be useful or not.

10.3.3. Detrimental Forgetting

Butz and Sigaud in [BS12] investigate the effect of global deletion on the occurrence of *detrimental forgetting* and propose a novel local deletion scheme for being used in XCSF. Especially in problems where a non-uniformly distributed sampling of the underlying function is assumed, important but rarely sampled niches can be 'forgotten' due to a higher deletion probability. This phenomenon clearly facilitates KGs of the first type. The *local deletion* mechanism attenuates the resulting negative effects – at least for functions with specific characteristics. However, results also indicate that XCSF already seems to be inherently robust against non-uniform sampling – at least for rather low complex two-dimensional functions as investigated in their study. Since their technique is intended to lessen the effect of uncovered niches within the knowledge base it can regarded as related in some sense. However, local deletion is neither an active nor a proactive means of dealing with KGs. Nevertheless, it still constitutes a valuable and complementary way to mitigate the resulting negative side-effects of KGs.

10.3.4. State-Action Map Covering in LCS

In [Nak+15], Nakata et al. elaborate on the important question of how XCS should cover a state-action space in noisy learning environments. By endowing XCS with the capability to build up knowledge before it is actually requested, the input space coverage strategy is directly affected. Nakata et al. introduce a couple of ways to combine the advantages of accuracy-based LCS and strength-based LCS, a question also investigated by Kovacs in [Kov00a]. A learning strategy to create a so-called *weighted complete action map* is introduced which enables XCS to evolve a population that is complete in the sense of learning the entire $X \times A \rightarrow P$ mapping, but also assigns more classifiers to the highest-payoff niches in the environment. The authors thus trade-off acquiring complete knowledge about the environment and purely retaining knowledge that yields reward, thereby omitting valuable knowledge of what is definitely wrong in which situation. In view of the concepts as introduced in this chapter, their approaches can neither be deemed active nor proactive in constructing novel knowledge. Still, adopting their *weighted complete action map* might be beneficial since the transductive interpolation-based knowledge inference approaches of Chapters 6 and 8 could obtain better sampling points which are constructed out of the classifiers.

10.3.5. Further Work on Active Learning

Naturally, there exists a strong relation between the proposed concepts and the field of AL. Beside the aspects of AL research that have already been mentioned above, the following paragraphs briefly appreciate a few further relevant works.

Another similar concept constitutes the *selective sampling* approach of Cohn et al. [CAL94], which defines a *region of uncertainty* over the class of concepts and strictly samples from within that region to improve generalization. With respect to our notions introduced in this chapter, their concept can be incorporated in order to reduce the amount of type-2 KGs within the inner regions (e.g., modeled by the convex hull) of the current knowledge base.

As an extension to the general AL framework, in [DC08] Donmez and Carbonell introduced *Proactive Learning* (PAL). With PAL the authors attempt to tackle certain limitations regarding oracle assumptions often made in the original AL setting. Among these limitations, the restriction to only one oracle, its omniscience, absence of oracle reluctance, as well as the assumption of equal costs for all queries are explicitly considered. Even if the name suggests the goal of achieving 'proactive' learning behavior, Donmez's and Carbonell's view of proactive learning is rather different from the perspective which is intended to be conveyed in this thesis. Their focus is set on challenges such as latencies occurring when the human oracle is not immediately available, and other factors such as reluctance, uncertainty and benevolence. In future work, estimates reflecting uncertainty, absence and other idiosyncratic characteristics of human oracles are integrated into the concepts outlined in this chapter.

Calma et al. discuss different closing strategies in terms of oracle types and study the question of how they can be used in a complementary way [Cal+17]. Their proposed concept fits well with the proactive knowledge construction cycle introduced here, especially when it comes to the eradication of gaps identified at layer 2. The same authors have also proposed two extensions of AL toward *dedicated* [Cal+16] and *opportunistic* [Bah+16] *collaborative interactive learning*, denoted D-CIL and O-CIL, respectively. Collaborative interactive learning [Sic+18] is a newly initiated research branch that attempts to holistically approach the challenges of today's collective systems by combining techniques form the domains of ML, AL, OC and AC. The overall objective is to facilitate *life-long learning* of future complex and intelligent systems that are deployed in time-variant productive environments where also humans are involved. In order to achieve the system wide goals, all participating entities (intelligent subsystems, human experts, workers, etc.) are assumed to interact and need to *collaborate*. These systems are referred to as *interactive*, since the exchange of knowledge – which is *learned* throughout the system's lifetime – is assumed to work bidirectionally among the artificial and naturally intelligent systems, i.e., agents and humans. In D-CIL similar shortcomings to those revealed in [DC08] regarding oracle assumptions are considered. The expertise of human oracles which are experts in a *clearly defined* application domain (hence the term 'dedicated') is subject to continual assessment by the D-CIL systems. This allows the exploitation of the individual strengths and attenuation of the weaknesses identified for those dedicated knowledge sources queried by means of AL techniques. O-CIL in contrast focuses on the collaboration aspect and the heterogeneity of the participating entities. Humans are not necessarily deemed as experts and other knowledge sources such as the vast amount of unstructured information available through the internet or crowd-sourcing services are considered. The collective systems are open and subject to churn which impacts the availability of knowledge sources.

10.3.6. Further Related Concepts from the AI Domain

Throughout the following paragraphs, further initiatives found in the broader field of AI which represent promising candidates for further investigation and combination with the conceptualized curious and proactive knowledge acquiring LCS-based agents are briefly mentioned.

Another related branch of research can be found in the domain of evolutionary algorithms (EAs). Lehman and Stanley introduced the *Novelty Search* algorithm in [LS08; LS11]. Novelty search replaces the objective or else fitness function of EAs with a novelty metric that forces the optimization process to seek *behavioral novelty*, i.e., underexplored areas in the space of unique behaviors – the so-called *novelty space*. A more thorough investigation of their definition of novelty might be also promising for the work on ALXCS as introduced here. It appears to relate more to the adoption of AL concepts as outlined in Section 10.1. Since in ALXCS the system is enabled to query oracles for gaining information about problem space regions where no experience could be made thus far. This in turn results in modified behaviors. However, a distinguishing aspect of the work of Lehman and Stanley is that the objective of proactive knowledge construction is to seek KGs in the evolved knowledge bases of learning algorithms instead of explicitly seeking novel behaviors of the learning agents in order to speed up learning and facilitate open-ended evolution.

Attenberg's and Provost's research on *Guided Learning* [AP10] is concerned with the question of how human resources can be used to construct accurate classification models in highly imbalanced scenarios beyond the means of classical AL. They propose to involve human experts to explicitly seek for rare objects in the data and pass the identified rarities to the learning algorithm for targeted training. Bringing their approach in line with the introduced notion of proactive knowledge construction, it becomes apparent that a strong potential to involve the idea of guided learning as a human-assisted step during the knowledge gap identification phase exists.

A psychologically inspired RL framework which is intended to enable learning agents to autonomously build up so-called *skill-hierarchies* that divide the overall task into smaller sub-learning tasks is called *Intrinsically Motivated Reinforcement Learning* [BSC04; CBS05]. It is based on the insight that the knowledge which is acquired for the own sake (i.e., due to intrinsic motivation) is crucial for solving complex tasks autonomously later on. Such skills are acquired during the exploration of the environment which happens reactively due to the perception of novel salient events.

These basic skills are then stored in a skill knowledge base where each entry constitutes an individual model producing intrinsic rewards that guide the agent's behavior. In contrast to proactively seeking KGs in an agent's knowledge base, in their work the agent acquires a skill knowledge base in a reactive manner where models of smaller sub-problems (skills) are created and stored as a response to salient events. Nevertheless, the integration of both ideas seems to be a promising and viable aspect of future research.

Another aspiration to develop cognitive agents that integrates the psychological model of *curiosity* with computer science has appeared under the name *Computational Curiosity.* See [WM13] for a recent survey. Already in the early 90s, Schmidhuber investigates the possibility of equipping learning agents with curiosity and boredom [Sch90]. In [Sch91] he proposed to extend (neural) control systems with a so-called *confidence module*, which aims at learning an estimate of the main controller's prediction reliability. Schmidhuber defines curiosity as "(...)the desire to improve a predictor of the reactions of an environment (a 'world model')" [Sch91]. Therefore, he proposes to spend reward when there exists a gap between reality and expectations. Thus, an agent becomes *curious* with regard to situations where the prediction accuracy is low. However, since the agent should improve with increasing experience, the agent becomes also *bored* about states about which it has been curious before as soon as the accuracy increases. A more general approach which keeps the agent away from states which are either too unpredictable or too predictable is presented by Oudeyer and Kaplan in [Oud04; OKH07], who introduced *Intelligent Adaptive Curiosity* (IAC).

Linden and Weber [LW93] proposed *competence modeling.* They used this approach in order to steer the exploration steps of reinforcement learning agents in the direction where their so-called world model is least confident in regard of the expected error. Therefore, a second neural network is trained to approximate the error from past experiences which is then used to guide the agent to states where the error is still high. Since the authors train neural networks which obtain a global approximation of the entire problem space, the approach fundamentally differs from the knowledge base-centric view postulated in this thesis. In the latter, the identification of uncertain and entirely uncovered regions beyond simply relying on error-based measures of competence is set in the spotlight.

Macedo and Cardoso extend the concept of curiosity modeling in learning agents toward also considering other *motivations* such as *suprise* and *hunger* in their *motivational agents* approach [MC04]. They argue that not only curiosity affects the exploratory behavior of natural beings. Based on single- and multi-agent settings their exploration by motivation idea has been shown to work as intended. Unknown environments have been efficiently explored with motivational agents selecting their actions by minimizing negative feelings and maximizing positive ones.

In summary, there exist different endeavors to endow learning systems with capabilities to act curiously in order to guide their exploratory behavior – a number of which have been briefly outlined above. Certainly, there are further concepts that have not been considered yet. However, since this chapter is intended to provide an outlook to potential future research directions a comprehensive literature review is beyond the scope here. A thorough investigation of the aforementioned and further methodologies to implement purposeful exploration of the underlying problem spaces and thereby facilitate proactive knowledge gap eradication constitutes a promising next step.

10.4. Chapter Summary

The purpose of this chapter was to provide first insights on the potentials of designing *Proactive LCS* that are endowed with the capability of actively seeking KGs and creating novel knowledge elements in advance in order to close them. Therefore, in the first part methods from the domain of AL have been reviewed and adapted in order to be incorporated into an *Active Learning XCS* (ALXCS). Preliminary results on a newly designed toy problem, called the *Mario problem*, and on a realistic data set for breast cancer classification have been briefly summarized. These revealed that the use of AL bears strong potentials for reducing the system's prediction errors over the entire learning task. Positive benefits on the classifier evolution in terms of smaller population sizes could also be observed. In the second part of this chapter, this approach was taken a step further toward the notion of *Proactive LCS*. It was abstracted from concrete LCS systems to a generic incremental knowledge building learning algorithm which was then extended to facilitate *Proactive Knowledge Construction*. It has been shown formally that the fundamental idea behind proactive knowledge constructing algorithms leads to effective KG reduction and quality improvements over the conducted learning period. A concrete proposal how the MLOC reference architecture from the domain of OC can be enhanced to facilitate proactive knowledge construction was discussed. Furthermore, a first methodology based on KDE to proactively identify particular regions in the knowledge base where the coverage is poor has been delineated. Finally, directly related concepts and further aspirations that strive to make ML algorithms anticipatory, curious and intrinsically motivated have been reviewed and brought in line with the KG-centric approach proposed.

Chapter 11.

Conclusion

11.1. Summary & Results

Learning is an inevitable capability to deal with the ever increasing complexity of today's and future *Self-Adaptive Systems* (SAS). Numerous *Machine Learning* (ML) concepts and techniques exist but not all of them satisfy the requirements of *Self-Learning Adaptive Systems* (SLAS) that are asked to act online in real world environments. Endowing SAS with learning algorithms appears to be a double-edged sword. On the one hand, these SLAS are enabled to adapt their control behaviors to changing conditions. On the other hand, however, requiring these systems to further improve (i.e., learn) at runtime implies evolving knowledge bases that determine how appropriately these systems react to unforeseen situations. Both the exposure of unforeseen situations to the systems and the need for continuing adaptation of their knowledge bases elicit the occurrence of a challenging issue in SLAS – *Knowledge Gaps* (KGs).

This thesis has introduced first strategies to counter such KGs immediately at the productive runtime of XCS-based SLAS. In contrast to existing approaches that employ offline simulations coupled with optimization heuristics in order to reactively create new knowledge elements on demand, this thesis proposed a variety of ways to endow a specific ML technique called XCS with internal mechanisms in this respect. Therefore, the posed KG challenge was approached from two directions in this thesis:

1. Bridging gaps in the knowledge base via *knowledge transduction* by means of using *scattered data interpolation.*
2. Preventing the occurrence of gaps by endowing the learning mechanisms with *Active Learning* (AL) techniques and the ability of *Proactive Knowledge Construction.*

The first methodology formed the core topic of this thesis. For the second way of countering KGs, initial concepts have been proposed and partly elaborated within a specifically dedicated chapter on future research directions at the end of this work.

After providing necessary background information and introducing theoretical prerequisites in Chapter 2, a system model for SLAS realized by OC concepts has been derived in Chapter 3 which states the research problem. In the same breath, the notion of KGs was formally introduced and the suitability of XCS, which served as reference algorithm under investigation throughout this thesis, was justified.

The main part of this doctoral thesis was then devoted to the development of first interpolation-based approaches for dealing with KGs. Architectural extensions to LCS have been proposed in Chapter 4. A novel *Interpolation Component* (IC) was introduced and thoroughly discussed. This component has laid the foundation for the four novel strategies that were developed to integrate *interpolation* into the algorithmic structure of XCS.

The first approach was presented in Chapter 5. It is intended to steer the *action-selection* behavior of XCS-based learning agents by means of *transductive knowledge inference* from collected experiences already made with the learning environment. This integration strategy called *Action Selection Integration* (ASI) was introduced in two different forms: (1) XCS-IC: A loosely coupled IC instance serves as memory for a limited number of made experiences, which was then used for an interpolation-based ad-hoc creation of an action-selection *surrogate* without the need for inductive approximation of an entire surrogate model. (2) XCS-CIC: A tightly integrated variant where the experiences are extracted directly from the already existing knowledge elements within XCS's knowledge base that match similar situations. This action-selection surrogate yields interpolated values that allow for a purposeful guidance of the actual action-selection regimes of XCS. This first ASI strategy was demonstrated to yield significant reductions between 30% and 40% in terms of the overall system prediction error on the challenging checkerboard problem. This was accompanied by slight increases of the fraction of correct decisions up to 2.98% for the first XCS-IC approach. Slight performance degradations regarding the same figure of merit have been observed for the tightly integrated XCS-CIC approach. A reduction of up to 1.08% regarding the fraction of correct decisions compared to standard XCS was observed – a result that was thoroughly discussed and deserves more research attention in the future.

In the Chapters 6 and 8, two strategies that enhance the rule discovery mechanisms of XCS (i.e., *covering* and the *genetic algorithm*) have been introduced. Both the *Covering Intialization Integration* (CII) as well as the *Offspring Initialization Integration* (OII) technique change the means of how newly created knowledge elements (i.e., classifiers) are initialized before added to XCS's knowledge (or rule) base. The former CII approach counteracts so-called type-1 KGs by means of interpolating the initial values for a subset of the classifiers' learning parameters on the basis of adjacent knowledge elements matching similar situations. Again the desired system error reductions could be observed when XCS-CIC was applied to several function approximation tasks. The improvements are ranging from 19% for the most simplistic benchmark function to less than 6% for the more complicated ones over the entire learning period of 200k steps. However, since covering only occurs at early stages of learning, it turned out that the highest and most substantial benefits naturally occur during the first 10k steps. A result that exactly supports the hypothesis initially conjectured in this thesis. The decreased approximation errors were found to come at neglectable costs of a marginal increase in the population sizes (number of evolved rules), which constantly stays below 1%.

In order to gain insights on the applicability of the so far developed interpolation strategies on a realistic application scenario in the context of self-adaptive urban traffic light control, an additional case study has been presented in Chapter 7. The so-called *Organic Traffic Control* (OTC) system was the subject of investigation. It utilizes a substantially adapted version of XCS (called XCS-O/C) in order to implement the self-learning property. Due to the modifications of XCS-O/C, only an adoption of ASI and CII appeared to be plausible. A case study focusing on an existing intersection situated at Hamburg, Germany simulated by a microscopic traffic simulator was reported. The XCS-based OTC system's task is to observe the intersection and self-adapt the signal plans of the installed traffic lights in order to reduce delay times of the traffic participants. Previous studies already confirmed the superiority of XCS-O/C in comparison to human-engineered signal plans. The obtained results that have been discussed in this chapter revealed that the interpolation-based XCS-O/C variant bears to potential for futher reductions of the average waiting times up to $2.17\frac{sec}{km}$ compared to the existing system. Especially in high-demanding periods, i.e., the morning rush-hours from 6 to 8 am, the interpolation-based XCS-O/C showed slightly depressed delay increases as well as a faster recovery from these peak conditions. Since the systems were configured to start with empty knowledge bases except for a human-designed default rule, these observed results clearly indicate improved KG resistance in a realistic scenario with near-to-reality conditions.

The second approach for interpolation-based classifier initialization called OII works similarly to CII but takes existing rules in the same problem space niche into account when creating novel classifiers. It was devised to eradicate the shortcoming of the former CII strategy which acts mainly at early learning stages. In contrast, the involved genetic algorithm is periodically invoked and has continual impact on the evolution of the knowledge base of XCS. The conducted empirical studies revealed that the OII strategy mostly impacts the number of classifiers stored in the population. The average population sizes have found to decrease between 9% and 18% on average, whereas the system errors partially increased up to 3.46% and partially decreased by up to 2.35% depending on the complexity of the considered function to be approximated. Complex functions were found to be approximated more accurately using OII. In combination with CII, however, the negative impacts on the system error can be entirely eradicated, while the positive effects on the population size can be retained. In numbers, the combination yields system error reductions between approximately 11% and 19% among all examined functions. Accordingly, as of yet it is suggested to always use OII and CII in combination.

As a last approach to integrate interpolation with XCS, the *Interpolated Prediction Integration* (IPI) strategy was introduced in Chapter 9. Each classifier is modified to maintain a small set of past experiences, i.e., data instances with the corresponding target values. This modification is intended to allow each classifier an interpolation-based computation of the individual reward prediction. Former approaches have mainly focused on iteratively fitting parametric approximation models which demand for a certain number of update steps before a reasonable predictive accuracy can be assumed. In addition to that, a novel classifier mixing strategy has been introduced that uses the union of all collected experiences among matching classifiers to calculate a global system prediction. With the IPI strategy applied, a substantially higher level of approximation accuracy could be observed. This results in possible system error reductions up to 92% for the complex Eggholder function. The least improvement regarding the approximation error that have been observed among all tested functions was a system error reduction by 19% for the six-dimensional Styblinski-Tang function which was already found to be hard to approximate earlier. Furthermore, the experimental results revealed strong impacts of the IPI strategy on the average generality of the evolved rules. Depending on the complexity of the underlying functions either strong increases (up to 159%) or significant reductions (up to 49%) regarding the average generality appeared. In the latter case, however, XCSF-IC is still able to significantly outperform the contending variants using polynomial approximation in terms of the system's approximation error. This has lead to the conclusion that the IPI strategy allows for more generalized classifiers when appropriate but forces conditions to specialize when necessary with respect to the underlying function complexity. The sustainability of the beneficial effects has also been confirmed on noisy functions adding random deviations up to 20% on the target value. Even under these challenging circumstances, XCSF-IC showed robust system error reductions of approximately 20% compared to the best alternative which employed cubic approximation.

In summary, the entirety of results obtained throughout all the conducted empirical investigations confirm the formulated hypothesis that transductive knowledge inference within XCS based on scattered data interpolation leads to increased learning efficiency. This insight is important in view of further corroborating the usefulness of XCS-based learning algorithms within SLAS which are deployed in dynamic environments that facilitate the occurrence of KGs. With these initial strategies for countering KGs in an ad-hoc fashion during the system's runtime as presented in this thesis, a next step toward robust SLAS has been taken.

To sum up, the results of this doctoral thesis comprise several scientific contributions which can be condensed as follows:

1. A proposal of a unified view on and a formal description of the task of learning in OC systems as well as the identification of resulting core challenges

2. A derivation of a novel notion of *knowledge gaps* complemented with a formal definition
3. An extension of the most prominent representative of the class of ERBML algorithms, the XCS classifier system, in terms of:
   a) Integration of several interpolation techniques to increase the learning efficiency and thus the robustness against KGs in SLAS
   b) A combination of AL concepts with XCS to endow LCS-based SLAS with a knowledge self-reflection capability
4. A first-ever confirmation of the initially posed hypothesis regarding the beneficial effects of incorporating interpolation in XCS's algorithmic structure
5. A demonstration regarding the real world applicability of an interpolation-assisted special-purpose XCS within in the OTC system
6. A first preliminary confirmation of the usefulness of incorporating AL concepts in XCS on a newly proposed synthetic and a medical classification task
7. A proposal for extension of OC's MLOC architecture in order to explicitly identify and close KGs

11.2. Outlook

An elaborate discussion about two concrete future research directions has already been given in Chapter 10 – the design of *Active Learning Classifier Systems* and the subsequent step toward the ultimate goal of *Proactive Learning Classifier Systems* endowed with the capability of *Proactive Knowledge Construction*.

As a more visionary outlook, the next paragraphs are supposed to shed light on related research initiatives to which the insights and techniques that resulted from this thesis could potentially contribute:

Collaborative Interactive Learning (CIL) This recent initiative is aimed at facing the challenges of interconnected intelligent systems (artificial or human) that collaborate and interactively exchange knowledge in order to facilitate lifelong learning of technical systems. Parts of these systems might solve particular subproblems collectively what implies related problem spaces. The exchange of knowledge constitutes a central aspect. These systems could also collaborate in identifying and closing KGs and proactively share the created knowledge elements among those systems that share a common problem space. The methodologies for a purposeful identification of gaps in knowledge bases as outlined in Chapter 10 integrate well with the envisaged topics of CIL. It further postulates to make strong use of AL techniques to exploit available knowledge sources (oracles) and continually reflects about the gained knowledge of the participating systems – a goal for which proactive knowledge construction constitutes a possible solution.

Autonomous Learning The developed techniques for interpolating new knowledge from existing knowledge are intended to increase the learning efficiency and the robustness of SLAS against KGs. This in turn allows for higher degrees of autonomy transferable to the systems, since not any situation that might occur during the runtime needs to be fully specified and foreseen a priori anymore. Actively created queries that are targeted to eradicate knowledge gaps and which are posed by the self-learning systems themselves clearly constitute a promising way to further increase system autonomy. Thereby, initial efforts of the system designers can be reduced during the design time. Combining the introduced KG-counter-strategies with online novelty and other concept drift detection techniques seems to be a viable next step. This would enable a proactive prediction of those regions within the underlying problem space that will become relevant in the near future. Providing a mechanism to predict knowledge relevance is deemed a crucial aspect to more purposefully guide the proactive knowledge construction processes of future autonomously learning systems.

Explainable AI and Self-aware Computing The demand for self-explainability and transparency of AI technology is indisputably present these days. Coming up with sophisticated techniques that facilitate interpretable knowledge is an inevitable stepping stone in that direction. Rule-based learning algorithms as considered in this thesis exhibit implicit interpretability by design. In view of system verification, rule-based systems have strong advantages in contrast to subsymbolic AI techniques such as neural networks. The novel notion of KGs facilitates further analysis of rule-bases. In a next step, this might allow for automatic test-case generation which explicitly seek and exploit gaps in order to finally reveal where the SLAS has its current deficiencies. The autonomous reasoning about the own knowledge base is a key aspect of self-aware computing systems. Proactive KG identification provides exactly such a mechanism and thus might also fertilize this research area.

A Final Thought As a last envisioned bottom line, this thesis is assumed to possess the potential for the initiation of a novel branch of ML techniques – *Knowledge Gap-centric algorithms.* Envisaged research aspirations comprise the foundation of this novel family of ML algorithms. The idea is to complement the ordinary objective of maximizing predictive accuracy with an intrinsic pressure to minimize the amount of remaining gaps in their knowledge base. A first step toward theoretical work has already been taken in Section 10.2, where the general rationale for proactive knowledge construction could be shown formally based on the definitions of KGs introduced in this thesis. The overall goal is to design novel online learning techniques based on the proposed KG framework in order to: (1) Improve the learning efficiency in contrast to conventional techniques. (2) Increase the system robustness against unforeseen situations, especially when regarded as to be deployed in highly-dynamic environments which demand for autonomous and explainable runtime learning – An ambitious goal for which this thesis developed first stepping stones.

Appendices

Appendix A.

Notations

This section provides an overview of the formal notations that are going to be introduced and used in this thesis. The following table also indicates abbreviations and terms that are used interchangeably, even if this was tried to be mostly prevented.

TABLE A.1.: Notations used throughout this thesis

Notation	**Short Description**	**Synonyms**
$\vec{x}$	Feature vector	Situation, System state, Observation, Data instance
$x_i^{(j)}$	j-th component of the i-th feature vector $\vec{x}$	Independent variable, Covariate, Feature
s_i	The i-th sampling point	Experience, Sample
SP	The sampling point set comprising s_i	Experience memory, Sampling point storage
θ or $\vec{w}$	Model parameters	Coefficients, Weights
K	Knowledge base	Rule set, Population, Model
KS	Knowledge space	Knowledge universe
k	A single knowledge element	Knowledge, Knowledge artifact
KG	Entirety of knowledge gaps	-
KG_1	All knowledge gaps of type 1	-
KG_2	All knowledge gaps of type 2	-
KG_{in}	All inner knowledge gaps	-
KG_{out}	All outer knowledge gaps	-
kg	A single knowledge gap	-
PS	Problem space $S \times A$	Problem
S or X	Situation space	State space, Feature space
A	Action space	-
$h(\vec{x})$	Hypothesis with regard to a vector $\vec{x}$	Prediction model
f	Approximation or interpolation for function f	Interpolant
$\{x_i\}_{i=1\ldots m}$	A set with m elements x_i	$\{x_i \mid i = 1 \ldots m\}$
$\mathcal{P}(A)$	Power set of a set A	-
$U[x, y]$	A uniformly distributed random number between x and y	-
$\mathcal{N}(\mu, \sigma)$	A normal distribution with mean μ and standard deviation σ	A Gaussian
$P(X)$	Marginal distribution of a random variable X	Prior, Covariate distribution
XCS Specifics	**Short Description**	**Synonyms**
cl	An individual classifier	IF-THEN / production rule, see also “knowledge element”
σ_t	Derived situation description at time t	see “feature vector”
$[P]$	Population of classifiers or rule base	see “knowledge base”
$[M]$	Match set	-
$[A]$	Action set	Approximation of an environmental niche
a_{exec}	Selected action to be executed	Control action, Reaction, Target, Class, Label
cl_{off}	A GA generated offspring classifier	Offspring, Child
cl_{par}	A parental classifier for the GA	Parent
cl_{cov}	A new classifier generated by covering	-
$cl.*$	Reference to a particular classifier's parameter $*$, e.g., $cl.F$	Classifier parameter
$cl.p(\vec{x})$	A classifier's prediction model with regard to an input vector $\vec{x}$	Computed prediction
r_{imm}	The immediate reward signal	Feedback signal, Payoff, Reinforcement, Utility
$PA(a)$	The entry of the prediction array for action a	System prediction

Appendix B.

XCS Hyperparameter Overview

TABLE B.1.: Overview of hyperparameters for XCS, XCSR and XCSF

Symbol	**Designation**	**Short Description**
N	Population size limit	Determines the max. number of microclassifiers allowed in $[P]$
β	Learning rate	Determines the step-width of Widrow-Hoff classifier parameter adjustments.
γ	Discount rate	Determines the influence of future rewards in relation to immediate rewards during classifier prediction updates
δ	Fraction of mean fitness	Defines the fraction of the populations mean fitness a classifier has to exceed to prevent a deletion vote increase
α, ν	Accuracy parameters	Control the steepness of the exponential drop in the absolute accuracy function κ.
ϵ_0	Prediction error tolerance / Target error	Can be interpreted as an offset in the accuracy function. When a classifier's prediction error is less than ϵ_0, it is considered to be fully accurate.
θ_{GA}	GA threshold	The minimum number of the action set's average time steps passed since the last GA invocation before the GA can be performed again

θ_{del}	Deletion consideration threshold	The minimum classifier experience required before it is deletion vote is further increased.
θ_{mna}	Minimum number of actions threshold	The minimum number of actions (mna) required to be present in $[M]$. This controls the number of covering operations in one step.
θ_{sub}	Subsumption consideration threshold	The minimum number of classifier experience required before a classifier is allowed to subsume another one.
$P_{\#}$	Wildcard probability / Initial generality	The probability that a single symbol in a classifier's condition is replaced by a wildcard symbol $\#$.
μ	Mutation probability	The probability for each offspring's symbol/gene of being mutated during the GA
χ	Crossover probability	The probability that crossover is applied to offspring classifiers
p_{ini}	Initial prediction value	Initialization value for the reward prediction parameter $cl.p$ of a newly covered classifier
ϵ_{ini}	Initial prediction error value	Initialization value for the absolute reward prediction error parameter $cl.\epsilon$ of a newly covered classifier
F_{ini}	Initial fitness value	Initialization value for the fitness parameter $cl.F$ of a newly covered classifier
$F_{reduction}$	Fitness reduction factor	A reduction factor applied to the averaged fitness values of the offspring classifiers
$\epsilon_{reduction}$	Prediction error reduction factor	A factor applied to the averaged error values of the offspring classifiers
r_0 (XCSR)	Default spread / Initial generality	The maximum deviation of the lower and upper bounds in each dimension from the current situation vector σ_t of a newly covered classifier

m_0 (XCSR)	Mutation step size	Same meaning as r_0 but for the case of mutation within the GA.
λ (XCSF)	RLS forgetting rate	Sort of discount that controls the impact of previous updates on the next matrix of the RLS update.
δ_{RLS} (XCSF)	RLS matrix initialization	The entries set during the initialization of the gain matrix of the RLS update.

Appendix C.

Supplemental Results

C.1. Additional Results for XCS-CIC on the Checkerboard Problem

The following tables provide additional results obtained by applying the interpolation strategies ASI, CII, OII as introduced in this thesis. The tables summarize each of the individual strategies (ASI, CII, OII) not described in the respective sections yet, as well as their combinations on the CBP instances CBP(3,3) and CBP(3,6). As in the corresponding sections in this thesis, three different interpolation techniques have been investigated, namely *Nearest Neighbor* (NeNe), *Inverse Distance Weighting* (IDW) and *Modified Shepard's Method* (MSM). The results have also been published and are discussed in more detail in [Ste+17a]. The experimental setup is the same as described in Chapter 5. The tables show the averaged values over the entire learning task (100k and 400k exploit trials for CBP(3,3) and CBP(3,6), respectively) with the corresponding standard deviations from the conducted 30 i.i.d. repetitions. * (**) indicates statistically (highly) significant improvements regarding the reported figure of merits compared with standard XCSR. This means that for the p-values of paired one-sided t-tests the assertion $p < \alpha = 0.05$ (0.01) is true. Bold values highlight statistically confirmed improvements and the arrows indicate whether the values have increased (↑) or decreased (↓) in comparison to standard XCS.

Table C.1.: Summary of additional results on CBP(3,3) and CBP(3,6) using *Nearest Neighbor* (NeNe) interpolation.

CBP(3,3)	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
XCS-CIC w/ **CII**	$949.33^{\uparrow} \pm 3.67$	$105.37^{\downarrow} \pm 5.30$	$678.65^{\uparrow **} \pm 12.98$
XCS-CIC w/ **OII**	$948.46^{\uparrow} \pm 3.07$	$107.45^{\uparrow} \pm 4.31$	$\mathbf{612.93}^{\downarrow **} \pm 11.30$
XCS-CIC w/ **CII+OII**	$949.01^{\uparrow} \pm 3.49$	$107.76^{\uparrow} \pm 5.35$	$\mathbf{623.62}^{\downarrow **} \pm 14.25$
XCS-CIC w/ **ASI+OII**	$944.30^{\downarrow **} \pm 3.24$	$\mathbf{65.73}^{\downarrow **} \pm 3.55$	$\mathbf{612.93}^{\downarrow **} \pm 11.30$
XCS-CIC w/ **ASI+CII**	$946.07^{\downarrow *} \pm 2.30$	$\mathbf{64.71}^{\downarrow **} \pm 3.31$	$683.09^{\uparrow **} \pm 10.10$
XCS-CIC w/ **ASI+OII+CII**	$943.89^{\downarrow **} \pm 4.02$	$\mathbf{66.44}^{\downarrow **} \pm 4.62$	$\mathbf{621.14}^{\downarrow **} \pm 13.39$
Standard XCS	947.97 ± 3.89	105.56 ± 5.30	669.51 ± 11.99
CBP(3,6)	*Fraction Correct*	*System Error*	*Macro-Classifiers*
XCS-CIC w/ **CII**	$878.41^{\uparrow} \pm 4.55$	$218.66^{\downarrow} \pm 6.69$	$8824.11^{\uparrow **} \pm 75.11$
XCS-CIC w/ **OII**	$872.84^{\downarrow **} \pm 5.39$	$227.14^{\uparrow **} \pm 6.64$	$\mathbf{7900.81}^{\downarrow **} \pm 75.71$
XCS-CIC w/ **CII+OII**	$875.15^{\downarrow} \pm 5.52$	$225.68^{\uparrow **} \pm 6.89$	$\mathbf{7960.78}^{\downarrow **} \pm 84.69$
XCS-CIC w/ **ASI+OII**	$858.31^{\downarrow **} \pm 6.27$	$\mathbf{163.59}^{\downarrow **} \pm 6.83$	$\mathbf{7900.81}^{\downarrow **} \pm 75.71$
XCS-CIC w/ **ASI+CII**	$868.28^{\downarrow **} \pm 6.48$	$\mathbf{154.14}^{\downarrow **} \pm 7.54$	$8824.73^{\uparrow **} \pm 70.77$
XCS-CIC w/ **ASI+OII+CII**	$861.44^{\downarrow **} \pm 5.92$	$\mathbf{160.97}^{\downarrow **} \pm 6.54$	$\mathbf{7933.82}^{\downarrow **} \pm 71.97$
Standard XCS	876.62 ± 4.67	220.97 ± 8.30	8761.07 ± 76.05
Nearest Neighbor			

Table C.2.: Summary of additional results on CBP(3,3) and CBP(3,6) using *Inverse Distance Weighting* (IDW) interpolation.

CBP(3,3)	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
XCS-CIC w/ **CII**	$\mathbf{950.18}^{\uparrow **} \pm 3.12$	$106.60^{\uparrow} \pm 5.48$	$698.28^{\uparrow **} \pm 11.75$
XCS-CIC w/ **OII**	$949.31^{\uparrow} \pm 3.92$	$105.38^{\downarrow} \pm 5.53$	$\mathbf{623.18}^{\downarrow **} \pm 13.77$
XCS-CIC w/ **CII+OII**	$\mathbf{951.27}^{\uparrow **} \pm 3.58$	$107.37^{\uparrow} \pm 5.63$	$\mathbf{653.07}^{\downarrow **} \pm 14.22$
XCS-CIC w/ **ASI+OII**	$948.60^{\uparrow} \pm 3.92$	$\mathbf{60.88}^{\downarrow **} \pm 4.28$	$\mathbf{623.18}^{\downarrow **} \pm 13.77$
XCS-CIC w/ **ASI+CII**	$949.70^{\uparrow} \pm 4.31$	$\mathbf{60.38}^{\downarrow **} \pm 5.22$	$699.93^{\uparrow **} \pm 14.07$
XCS-CIC w/ **ASI+OII+CII**	$949.49^{\uparrow} \pm 3.83$	$\mathbf{60.79}^{\downarrow **} \pm 4.55$	$\mathbf{655.68}^{\downarrow **} \pm 10.45$
Standard XCS	947.97 ± 3.89	105.56 ± 5.30	669.51 ± 11.99
CBP(3,6)	*Fraction Correct*	*System Error*	*Macro-Classifiers*
XCS-CIC w/ **CII**	$\mathbf{879.36}^{\uparrow **} \pm 4.98$	$218.84^{\downarrow} \pm 6.10$	$8901.60^{\uparrow **} \pm 73.29$
XCS-CIC w/ **OII**	$\mathbf{882.63}^{\uparrow **} \pm 3.70$	$\mathbf{214.98}^{\downarrow **} \pm 4.79$	$\mathbf{8128.04}^{\downarrow **} \pm 92.10$
XCS-CIC w/ **CII+OII**	$\mathbf{887.08}^{\uparrow **} \pm 2.92$	$\mathbf{211.00}^{\downarrow **} \pm 3.71$	$\mathbf{8263.77}^{\downarrow **} \pm 73.31$
XCS-CIC w/ **ASI+OII**	$\mathbf{879.10}^{\uparrow *} \pm 4.35$	$\mathbf{141.72}^{\downarrow **} \pm 4.35$	$\mathbf{8128.04}^{\downarrow **} \pm 92.10$
XCS-CIC w/ **ASI+CII**	$\mathbf{879.55}^{\uparrow *} \pm 4.45$	$\mathbf{142.17}^{\downarrow **} \pm 5.32$	$8869.92^{\uparrow **} \pm 90.17$
XCS-CIC w/ **ASI+OII+CII**	$\mathbf{884.00}^{\uparrow **} \pm 3.83$	$\mathbf{136.92}^{\downarrow **} \pm 4.49$	$\mathbf{8268.54}^{\downarrow **} \pm 76.86$
Standard XCS	876.62 ± 4.67	220.97 ± 8.30	8761.07 ± 76.05
Inverse Distance Weighting			

Table C.3.: Summary of additional results on CBP(3,3) and CBP(3,6) using *Modified Shepard's Method* (MSM) interpolation.

CBP(3,3)	*Fraction Correct* mean $\pm$1SD	*System Error* mean $\pm$1SD	*Macro-Classifiers* mean $\pm$1SD
XCS-CIC w/ **CII**	**949.57**$^{\uparrow *}$ $\pm$ 4.16	106.57$^{\uparrow}$ $\pm$ 5.54	695.69$^{\uparrow **}$ $\pm$ 13.02
XCS-CIC w/ **OII**	948.68$^{\uparrow}$ $\pm$ 3.49	106.22$^{\uparrow}$ $\pm$ 5.63	**623.49**$^{\downarrow **}$ $\pm$ 14.21
XCS-CIC w/ **CII+OII**	**951.18**$^{\uparrow **}$ $\pm$ 3.61	106.42$^{\uparrow}$ $\pm$ 5.00	**649.26**$^{\downarrow **}$ $\pm$ 13.70
XCS-CIC w/ **ASI+OII**	947.94$^{\downarrow}$ $\pm$ 3.79	**61.42**$^{\downarrow **}$ $\pm$ 4.41	**623.49**$^{\downarrow **}$ $\pm$ 14.21
XCS-CIC w/ **ASI+CII**	948.99$^{\uparrow}$ $\pm$ 3.64	**61.46**$^{\downarrow **}$ $\pm$ 4.09	693.92$^{\uparrow **}$ $\pm$ 14.82
XCS-CIC w/ **ASI+OII+CII**	949.76$^{\uparrow}$ $\pm$ 3.87	**60.35**$^{\downarrow **}$ $\pm$ 4.44	**654.21**$^{\downarrow **}$ $\pm$ 16.09
Standard XCS	947.97 $\pm$ 3.89	105.56 $\pm$ 5.30	669.51 $\pm$ 11.99
CBP(3,6)	*Fraction Correct*	*System Error*	*Macro-Classifiers*
XCS-CIC w/ **CII**	**879.57**$^{\uparrow *}$ $\pm$ 5.15	**217.80**$^{\downarrow *}$ $\pm$ 5.67	8882.25$^{\uparrow **}$ $\pm$ 76.77
XCS-CIC w/ **OII**	**883.10**$^{\uparrow **}$ $\pm$ 4.06	**213.68**$^{\downarrow **}$ $\pm$ 4.91	**8093.53**$^{\downarrow **}$ $\pm$ 96.44
XCS-CIC w/ **CII+OII**	**887.68**$^{\uparrow **}$ $\pm$ 3.08	**210.45**$^{\downarrow **}$ $\pm$ 3.97	**8241.39**$^{\downarrow **}$ $\pm$ 62.41
XCS-CIC w/ **ASI+OII**	**879.33**$^{\uparrow **}$ $\pm$ 4.44	**140.97**$^{\downarrow **}$ $\pm$ 4.82	**8093.53**$^{\downarrow **}$ $\pm$ 96.44
XCS-CIC w/ **ASI+CII**	**879.22**$^{\uparrow *}$ $\pm$ 4.28	**141.51**$^{\downarrow **}$ $\pm$ 4.71	8867.25$^{\uparrow **}$ $\pm$ 76.65
XCS-CIC w/ **ASI+OII+CII**	**883.86**$^{\uparrow **}$ $\pm$ 3.55	**136.59**$^{\downarrow **}$ $\pm$ 3.85	**8220.57**$^{\downarrow **}$ $\pm$ 77.49
Standard XCS	876.62 $\pm$ 4.67	220.97 $\pm$ 8.30	8761.07 $\pm$ 76.05
Modified Shepard's Method			

C.2. Additional Results for ALXCS on the Mario and the WBC problem

The following tables complement the discussions of Section 10.1. A more detailed discussion on the results can be found in [SMH17].

Table C.4 contains a summary of the obtained results for ALXCS on the WBC dataset. Average values and standard deviations over two phases of the entire learning task (until $20k$ trials on the left and $100k$ exploit trials on the right) from 30 i.i.d. runs are presented. * (**) indicates statistically (highly) significant deviations of the reported metrics compared to standard XCS. This means that for the p-values of paired one-sided t-tests the assertion $p < \alpha = 0.05$ (0.01) is true. $\uparrow$ and $\downarrow$ indicate whether the values have increased or decreased in comparison to standard XCS.

Table C.5 contains a summary of the obtained results of ALXCS on the novel Mario classification environment. The presented values and standard deviations are to be interpreted analogously as for the previous Table C.4. The same applies for the symbols * (**), $\uparrow$ and $\downarrow$.

TABLE C.4.: Results for ALXCS on the WBC classification data set

WBC Data Set **Artificial Oracle**	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
Exploit Trials	First 20.000			Entire 100.000		
ALXCS w/ **QBF**	987.00$^{\downarrow **}$ ± 2.03	20.01$^{\downarrow **}$ ± 2.58	2285.78$^{\downarrow **}$ ± 32.48	995.32$^{\downarrow **}$ ± 1.28	9.79$^{\downarrow **}$ ± 1.76	2345.03$^{\downarrow **}$ ± 28.81
ALXCS w/ **QBC,** $\theta_{sim} = 0.01$	988.38$^{\downarrow}$ ± 1.72	27.37$^{\downarrow **}$ ± 2.26	2906.76$^{\downarrow **}$ ± 33.90	996.61$^{\downarrow **}$ ± 0.85	8.68$^{\downarrow **}$ ± 1.41	2964.57$^{\downarrow **}$ ± 32.59
ALXCS w/ **QBC,** $\theta_{sim} = 0.20$	987.16$^{\downarrow **}$ ± 1.98	20.58$^{\downarrow **}$ ± 2.43	2396.39$^{\downarrow **}$ ± 28.30	995.72$^{\downarrow **}$ ± 1.01	9.28$^{\downarrow **}$ ± 1.55	2417.17$^{\downarrow **}$ ± 31.07
ALXCS w/ **QSR**	985.22$^{\downarrow **}$ ± 1.60	87.36$^{\uparrow **}$ ± 3.44	5392.15$^{\uparrow **}$ ± 14.39	988.70$^{\downarrow **}$ ± 0.76	71.38$^{\uparrow **}$ ± 3.15	5902.69$^{\uparrow **}$ ± 9.14
Standard XCS	988.78 ± 1.48	74.30 ± 2.80	3287.55 ± 38.99	997.20 ± 0.55	17.70 ± 0.94	3749.73 ± 32.48

TABLE C.5.: Results of ALXCS on the novel Mario classification problem

Mario Pixel Art **Artificial Oracle**	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD	*Fraction Correct* mean ±1SD	*System Error* mean ±1SD	*Macro-Classifiers* mean ±1SD
Exploit Trials	First 20.000			Entire 100.000		
ALXCS w/ **QBF**	872.78$^{\uparrow **}$ ± 4.43	139.57$^{\downarrow **}$ ± 3.71	2480.61$^{\downarrow **}$ ± 39.92	889.40$^{\uparrow **}$ ± 2.92	140.50$^{\downarrow **}$ ± 2.98	2307.29$^{\downarrow **}$ ± 33.03
ALXCS w/ **QBC,** $\theta_{sim} = 0.01$	819.32$^{\uparrow **}$ ± 6.38	238.41$^{\downarrow **}$ ± 6.09	2788.34$^{\downarrow}$ ± 36.72	870.25$^{\uparrow **}$ ± 4.71	188.29$^{\downarrow **}$ ± 4.27	2296.43$^{\downarrow **}$ ± 26.02
ALXCS w/ **QBC,** $\theta_{sim} = 0.20$	873.07$^{\uparrow **}$ ± 4.59	140.37$^{\downarrow **}$ ± 4.31	2493.43$^{\downarrow **}$ ± 26.81	898.99$^{\uparrow **}$ ± 3.12	140.59$^{\downarrow **}$ ± 3.13	2313.82$^{\downarrow **}$ ± 27.44
ALXCS w/ **QSR**	834.02$^{\uparrow **}$ ± 4.89	195.86$^{\downarrow **}$ ± 4.79	3094.81$^{\uparrow **}$ ± 37.62	869.96$^{\uparrow **}$ ± 3.82	163.09$^{\downarrow **}$ ± 3.24	2940.72$^{\uparrow **}$ ± 17.65
Standard XCS	801.00 ± 6.87	276.23 ± 6.46	2784.98 ± 41.92	866.23 ± 5.22	201.14 ± 5.84	2385.95 ± 20.60

Bibliography

[AP94] A. Aamodt and E. Plaza. "Case-based reasoning: Foundational issues, methodological variations, and system approaches". In: *AI communications* 7.1 (1994), pp. 39–59.

[Alp10] E. Alpaydin. *Introduction to Machine Learning.* 2nd. The MIT Press, 2010.

[AI18] A. Andoni and P. Indyk. "Handbook of Discrete and Computational Geometry". In: ed. by C. D. Toth, J. O'Rourke, and J. E. Goodman. 3rd. Chapman and Hall/CRC, 2018. Chap. Nearest neighbors in high-dimensional spaces, pp. 1135–1156.

[Ang88] D. Angluin. "Queries and Concept Learning". In: *Machine Learning* 2.4 (1988), pp. 319–342.

[AMS97] C. G. Atkeson, A. W. Moore, and S. Schaal. "Locally weighted learning for control". In: *Lazy learning.* Springer, 1997, pp. 75–113.

[AP10] J. Attenberg and F. Provost. "Why Label when You Can Search?: Alternatives to Active Learning for Applying Human Resources to Build Classification Models Under Extreme Class Imbalance". In: *Proc. of 16th ACM SIGKDD KDD.* Washington, DC, USA: ACM, 2010, pp. 423–432.

[ALM08] A. E. S. Auf, M. Litza, and E. Maehle. "Distributed Fault-Tolerant Robot Control Architecture Based on Organic Computing Principles". In: *Proc. of BICC'08.* Springer, 2008, pp. 115–124.

[BS07] O. Babaoglu and H. E. Shrobe. "Foreword from the General Co-chairs". In: *2007 First International Conference on Self-Adaptive and Self-Organizing Systems.* Los Alamitos, CA, USA: IEEE Computer Society, July 2007, p. ix.

[Bah+16] G. Bahle, A. Calma, J. M. Leimeister, P. Lukowicz, S. Oeste-Reiß, T. Reitmaier, A. Schmidt, B. Sick, G. Stumme, and K. A. Zweig. "Life-long Learning and Collaboration of Smart Technical Systems in Open-Ended Environments – Opportunistic Collaborative Interactive Learning". In: *Proc. of 2016 IEEE International Conference on Autonomic Computing (ICAC).* July 2016, pp. 315–324.

[BK93] L. C. Baird III and A. H. Klopf. *Reinforcement learning with high-dimensional, continuous actions.* Tech. rep. WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1993.

[Ban+98] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic programming: an introduction.* Vol. 1. Morgan Kaufmann San Francisco, 1998.

[BKG04] P. Baranyi, L. T. Koczy, and T. D. Gedeon. “A generalized concept for fuzzy rule interpolation”. In: *IEEE Transactions on Fuzzy Systems* 12.6 (Dec. 2004), pp. 820–837.

[Bar+05] J. Barceló, E. Codina, J. Casas, J. Ferrer, and D. García. “Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems”. In: *Journal of Intelligent and Robotic Systems* 41.2–3 (2005), pp. 173–203.

[BSC04] A. G. Barto, S. Singh, and N. Chentanez. “Intrinsically motivated learning of hierarchical collections of skills”. In: *Proceedings of the 3rd International Conference on Development and Learning.* 2004, pp. 112–19.

[BL92] E. B. Baum and K. Lang. “Query learning can work poorly when a human oracle is used”. In: *International joint conference on neural networks.* Vol. 8. 1992.

[Beh+12] M. Behdad, T. French, L. Barone, and M. Bennamoun. “On principal component analysis for high-dimensional XCSR”. In: *Evolutionary Intelligence* 5.2 (June 2012), pp. 129–138.

[BTW14] K. L. Bellman, S. Tomforde, and R. P. Würtz. “Interwoven Systems: Self-Improving Systems Integration”. In: *Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2014, London, United Kingdom, September 8-12, 2014.* 2014, pp. 123–127.

[Ber08] M. de Berg. “Orthogonal Range Searching”. In: *Computational Geometry: Algorithms and Applications.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 95–120.

[Ber+08] M. de Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications.* 3rd. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008.

[Ber18] M. Berghofer. “Extension of a Function Generation Tool by Drifting Data Samplers and Standard Benchmark Problems”. Advisor: Anthony Stein. Bachelor’s Thesis. University of Augsburg, Department of Computer Science, 2018.

[BG03] E. Bernadó-Mansilla and J. M. Garrell-Guiu. “Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks”. In: *Evol. Comp.* 11.3 (2003), pp. 209–238.

[Ber+11] A. Bernauer, J. Zeppenfeld, O. Bringmann, A. Herkersdorf, and W. Rosenstiel. “Combining Software and Hardware LCS for Lightweight On-chip Learning”. In: *Organic Computing.* Basel, CH: Birkhäuser Verlag, 2011, pp. 253–265.

[BM99] M. W. Berry and K. S. Minser. “Algorithm 798: High-dimensional Interpolation Using the Modified Shepard Method”. In: *ACM Trans. Math. Softw.* 25.3 (Sept. 1999), pp. 353–366.

[BS02] H.-G. Beyer and H.-P. Schwefel. “Evolution strategies – A comprehensive introduction”. In: *Natural Computing* 1.1 (2002), pp. 3–52.

[BRC04] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa. “Heuristically Accelerated Q–Learning: A New Approach to Speed Up Reinforcement Learning”. In: *Advances in Artificial Intelligence – SBIA 2004.* Ed. by A. L. C. Bazzan and S. Labidi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 245–254.

[BRC08] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa. “Accelerating autonomous learning by using heuristic selection of actions”. In: *Journal of Heuristics* 14.2 (Apr. 2008), pp. 135–168.

[BU08] T. Bobach and G. Umlauf. “Natural neighbor concepts in scattered data interpolation and discrete function approx- imation”. In: *Visualization of large and unstructured data sets.* Ed. by H. Hagen, M. Hering-Bertram, and C. Garth. Bonn: Gesellschaft für Informatik e.V., 2008, pp. 23–35.

[Bob+09] T. Bobach, G. Farin, D. Hansford, and G. Umlauf. “Natural neighbor extrapolation using ghost points”. In: *Computer-Aided Design* 41.5 (2009). Voronoi Diagrams and their Applications, pp. 350–365.

[Buh03] M. D. Buhmann. *Radial basis functions: theory and implementations.* Vol. 12. Cambridge university press, 2003.

[Bul15] L. Bull. “A brief history of learning classifier systems: from CS-1 to XCS and its variants”. In: *Evolutionary Intelligence* 8.2 (Sept. 2015), pp. 55–70.

[BO02] L. Bull and T. O’Hara. “Accuracy-based neuro and neuro-fuzzy classifier systems”. In: *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation.* Morgan Kaufmann Publishers Inc. 2002, pp. 905–911.

[But+04a] M. Butz, T. Kovacs, P. Lanzi, and S. Wilson. “Toward a Theory of Generalization and Learning in XCS”. In: *Evolutionary Computation, IEEE Transactions on* 8.1 (Feb. 2004), pp. 28–46.

[BLW08] M. Butz, P. Lanzi, and S. Wilson. “Function Approximation With XCS: Hyperellipsoidal Conditions, Recursive Least Squares, and Compaction”. In: *IEEE Transactions on Evolutionary Computation* 12.3 (June 2008), pp. 355–376.

[But02] M. V. Butz. *Anticipatory learning classifier systems.* Vol. 4. Springer Science & Business Media, 2002.

[But05a] M. V. Butz. *Rule-based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design.* Vol. 191. Springer, 2005.

[BGL07] M. V. Butz, D. E. Goldberg, and P. L. Lanzi. “Effect of Pure Error-Based Fitness in XCS”. In: *Learning Classifier Systems.* Springer Berlin Heidelberg, 2007, pp. 104–114.

[But+04b] M. V. Butz, P. L. Lanzi, X. Llorà, and D. E. Goldberg. “Knowledge Extraction and Problem Structure Identification in XCS”. In: *Parallel Problem Solving from Nature - PPSN VIII.* Ed. by X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1051–1060.

[BLW06] M. V. Butz, P. L. Lanzi, and S. W. Wilson. “Hyper-ellipsoidal Conditions in XCS: Rotation, Linear Approximation, and Solution Structure”. In: *Proc. of GECCO ’06.* 2006, pp. 1457–1464.

[BSG03] M. V. Butz, K. Sastry, and D. E. Goldberg. “Tournament Selection: Stable Fitness Pressure in XCS”. In: *Genetic and Evolutionary Computation – GECCO 2003.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 1857–1869.

[BS12] M. V. Butz and O. Sigaud. “XCSF with Local Deletion: Preventing Detrimental Forgetting”. In: *Evolutionary Intelligence* 5.2 (2012), pp. 117–127.

[But05b] M. V. Butz. “Kernel-based, Ellipsoidal Conditions in the Real-valued XCS Classifier System”. In: *Proc. of GECCO ’05.* New York, NY: ACM, 2005, pp. 1835–1842.

[BW02] M. Butz and S. W. Wilson. “An Algorithmic Description of XCS”. In: *Soft Computing* 6.3-4 (2002), pp. 144–153.

[Cal+16] A. Calma, J. M. Leimeister, P. Lukowicz, S. Oeste-Reiss, T. Reitmaier, A. Schmidt, B. Sick, G. Stumme, and K. A. Zweig. “From Active Learning to Dedicated Collaborative Interactive Learning”. In: *ARCS 2016; 29th International Conference on Architecture of Computing Systems.* Apr. 2016, pp. 1–8.

[Cal+17] A. Calma, D. Kottke, B. Sick, and S. Tomforde. “Learning to Learn: Dynamic Runtime Exploitation of Various Knowledge Sources and Machine Learning Paradigms”. In: *2nd IEEE International Workshops on Foundations and Applications of Self* Systems, FAS*W@SASO/ICCAC 2017, Tucson, AZ, USA*. 2017, pp. 109–116.

[Cam+03] S. Camazine, J.-L. Deneubourg, N. R. Franks, G. T. James Sneyd and, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton Univers. Press, 2003.

[CC06] V. R. Carvalho and W. W. Cohen. “Single-pass Online Learning: Performance, Voting Schemes and Online Feature Selection”. In: *Proc. of the 12th ACM SIGKDD*. Philadelphia, PA, USA: ACM, 2006, pp. 548–553.

[CSZ06] O. Chapelle, B. Scholkopf, and A. Zien. “Semi-Supervised Learning”. In: The MIT Press, 2006. Chap. Discussion of Semi-Supervised Learning and Transduction.

[Cha05] N. V. Chawla. “Data Mining for Imbalanced Datasets: An Overview”. In: *Data Mining and Knowledge Discovery Handbook*. Ed. by O. Maimon and L. Rokach. Boston, MA: Springer US, 2005, pp. 853–867.

[Cha+02] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. “SMOTE: Synthetic Minority Over-sampling Technique”. In: *J. Artif. Int. Res.* 16.1 (June 2002), pp. 321–357.

[CHS05] C. Chen, Y. Hon, and R. Schaback. “Scientific computing with radial basis functions”. In: *Department of Mathematics, University of Southern Mississippi, Hattiesburg, MS* 39406 (2005).

[CBS05] N. Chentanez, A. G. Barto, and S. P. Singh. “Intrinsically Motivated Reinforcement Learning”. In: *Advances in Neural Information Processing Systems 17*. MIT Press, 2005, pp. 1281–1288.

[CR94] D. Cliff and S. Ross. “Adding Temporary Memory to ZCS”. In: *Adaptive Behavior* 3.2 (1994), pp. 101–150.

[CAL94] D. Cohn, L. Atlas, and R. Ladner. “Improving generalization with active learning”. In: *Machine Learning* 15.2 (1994), pp. 201–221.

[DLA07] H. H. Dam, C. Lokan, and H. A. Abbass. “Evolutionary Online Data Mining: An Investigation in a Dynamic Environment”. In: *Evolutionary Computation in Dynamic and Uncertain Environments*. 2007, pp. 153–178.

[Dav97] S. Davies. “Multidimensional triangulation and interpolation for reinforcement learning”. In: *Advances in neural information processing systems*. 1997, pp. 1005–1011.

[DS17] E. Debie and K. Shafi. “Implications of the curse of dimensionality for supervised learning classifier systems: theoretical and empirical analyses”. In: *Pattern Analysis and Applications* (Aug. 2017).

[DD16] F. Dell’Accio and F. Di Tommaso. “Scattered data interpolation by Shepard’s like methods: classical results and recent advances”. In: *Special Issue: Dolomites Research Notes on Approximation* 9 (2016).

[Dia+16] A. Diaconescu, S. Frey, C. Müller-Schloer, J. Pitt, and S. Tomforde. “Goal-Oriented Holonics for Complex System (Self-)Integration: Concepts and Case Studies”. In: *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. Sept. 2016, pp. 100–109.

[DP13] G. Ditzler and R. Polikar. “Incremental Learning of Concept Drift from Streaming Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.10 (Oct. 2013), pp. 2283–2301.

[Dit+15] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. “Learning in Nonstationary Environments: A Survey”. In: *IEEE Computational Intelligence Magazine* 10.4 (Nov. 2015), pp. 12–25.

[DC08] P. Donmez and J. G. Carbonell. “Proactive Learning: Cost-sensitive Active Learning with Multiple Imperfect Oracles”. In: *Proc. of 17th ACM Conference on Information and Knowledge Management*. Napa Valley, California, USA: ACM, 2008, pp. 619–628.

[DB94] M. Dorigo and H. Bersini. “A comparison of Q-learning and Classifier Systems”. In: *In Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior*. MIT Press, 1994, pp. 248–255.

[Dru08] J. Drugowitsch. *Design and Analysis of Learning Classifier Systems - A Probabilistic Approach*. Vol. 139. Studies in Computational Intelligence. Springer, 2008.

[DB08] J. Drugowitsch and A. Barry. “A formal framework and extensions for function approximation in learning classifier systems”. In: *Machine Learning* 70.1 (2008), pp. 45–88.

[DB07] J. Drugowitsch and A. M. Barry. “Mixing independent classifiers”. In: *Proc. of GECCO 2007*. ACM. 2007, pp. 1596–1603.

[Du 08] W. Du Toit. “Radial basis function interpolation”. PhD thesis. Stellenbosch: Stellenbosch University, 2008.

[Emi+16] M. S. Emigh, E. G. Kriminger, A. J. Brockmeier, J. C. Príncipe, and P. M. Pardalos. “Reinforcement Learning in Video Games Using Nearest Neighbor Interpolation and Metric Learning”. In: *IEEE Transactions on Computational Intelligence and AI in Games* 8.1 (Mar. 2016), pp. 56–66.

[Fas07] G. F. Fasshauer. *Meshfree Approximation Methods with MATLAB.* River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2007.

[For86] S. Fortune. “A Sweepline Algorithm for Voronoi Diagrams”. In: *Proceedings of the Second Annual Symposium on Computational Geometry.* SCG ’86. Yorktown Heights, New York, USA: ACM, 1986, pp. 313–322.

[Fra79] R. Franke. *A Critical Comparison of some Methods for Interpolation of Scattered Data.* Tech. rep. DTIC Document, 1979.

[FN80] R. Franke and G. Nielson. “Smooth Interpolation of Large Sets of Scattered Data”. In: *International journal for numerical methods in engineering* 15.11 (1980), pp. 1691–1704.

[FKS12] N. Fredivianus, K. Kara, and H. Schmeck. “Stay Real!: XCS with Rule Combining for Real Values”. In: *Proc. of GECCO 2012.* Philadelphia, Pennsylvania, USA: ACM, 2012, pp. 1493–1494.

[FPS10] N. Fredivianus, H. Prothmann, and H. Schmeck. “XCS Revisited: A Novel Discovery Component for the eXtended Classi-fier System”. In: *Proceedings of 8th International Conference of Simulated Evolution and Learning.* 2010, pp. 289–298.

[GR05] T. Gabel and M. Riedmiller. “CBR for state value function approximation in reinforcement learning”. In: *International Conference on Case-Based Reasoning.* Springer. 2005, pp. 206–221.

[Gam+14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. “A Survey on Concept Drift Adaptation”. In: *ACM Comput. Surv.* 46.4 (Mar. 2014), 44:1–44:37.

[GVV98] A. Gammerman, V. Vovk, and V. Vapnik. “Learning by Transduction”. In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence.* UAI’98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., 1998, pp. 148–155.

[GSC17] R. Glatt, F. L. da Silva, and A. H. R. Costa. “Case-based policy inference for transfer in reinforcement learning”. In: *Workshop on Scaling-Up Reinforcement Learning at ECML.* 2017, pp. 1–8.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

[GW78] W. J. Gordon and J. A. Wixom. “Shepard’s Method of "Metric Interpolation" to Bivariate and Multivariate Interpolation”. In: *Mathematics of Computation* 32.141 (1978), pp. 253–264.

[GR92] J. J. Grefenstette and C. L. Ramsey. “An Approach to Anytime Learning”. In: *Proceedings of the 9th International Conference on Machine Learning.* 1992, pp. 189–195.

[GS16] C. Gruhl and B. Sick. “Detecting Novelty with CANDIES – Improved Awareness Techniques Based on Probabilstic Knowledge Models”. In: *International Journal of Machine Learning and Cybernetics* 7.33 (Nov. 2016), pp. 1–19.

[Gu+17] S. Gu, T. P. Lillicrap, Z. Ghahramani, R. E. Turner, B. Schölkopf, and S. Levine. “Interpolated Policy Gradient: Merging On-Policy and Off-Policy Gradient Estimation for Deep Reinforcement Learning”. In: *CoRR* abs/1706.00387 (2017). arXiv: `1706.00387`.

[GA19] D. Gunning and D. Aha. “DARPA’s Explainable Artificial Intelligence (XAI) Program”. In: *AI Magazine* 40.2 (June 2019), pp. 44–58.

[HS13] M. F. A. Hady and F. Schwenker. “Semi-supervised Learning”. In: *Handbook on Neural Information Processing.* Ed. by M. Bianchini, M. Maggini, and L. C. Jain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 215–239.

[Han+18] S. Hansen, A. Pritzel, P. Sprechmann, A. Barreto, and C. Blundell. “Fast deep reinforcement learning using online adjustments from the past”. In: *Advances in Neural Information Processing Systems 31.* Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 10567–10577.

[HG09] H. He and E. A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (Sept. 2009), pp. 1263–1284.

[Her08] F. Herrera. “Genetic Fuzzy Systems: Taxonomy, Current Research Trends and Prospects”. In: *Evolutionary Intelligence* 1.1 (2008), pp. 27–46.

[HR78] J. H. Holland and J. S. Reitman. “Cognitive Systems Based on Adaptive Algorithms”. In: *Pattern directed inference systems.* Ed. by D. A. Waterman and F. Hayes-Roth. NY: Academic Press, 1978, pp. 313–329.

[Hol62] J. H. Holland. “Outline for a Logical Theory of Adaptive Systems”. In: *J. ACM* 9.3 (July 1962), pp. 297–314.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems.* 2nd (1992). Ann Arbor, MI: University of Michigan Press, 1975.

[Hol76] J. H. Holland. “Adaptation”. In: *Progress in Theoretical Biology.* Ed. by R. Rosen and F. Snell. Vol. 4. New York: Academic Press, 1976, pp. 263–293.

[Hol85] J. H. Holland. "Properties of the Bucket Brigade". In: *Proceedings of the 1st International Conference on Genetic Algorithms.* Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 1–7.

[Hol92] J. H. Holland. "Complex Adaptive Systems". In: *Daedalus* 121.1 (1992), pp. 17–30.

[HNT+17] H. H. Hoos, F. Neumann, H. Trautmann, et al. "Automated Algorithm Selection and Configuration (Dagstuhl Seminar 16412)". In: *Dagstuhl Reports* 6 (2017), p. 42.

[IBZ12] M. Iqbal, W. N. Browne, and M. Zhang. "XCSR with Computed Continuous Action". In: *AI 2012: Advances in Artificial Intelligence: 25th Australasian Joint Conference, Sydney, Australia, December 4-7, 2012. Proceedings.* Ed. by M. Thielscher and D. Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 350–361.

[JY13] M. Jamil and X. Yang. "A Literature Survey of Benchmark Functions for Global Optimisation Problems". In: *IJMNO* 4.2 (2013).

[KC03] J. O. Kephart and D. M. Chess. "The Vision of Autonomic Computing". In: *IEEE Computer* 36.1 (2003), pp. 41–50.

[KOB08] F. Kharbat, M. Odeh, and L. Bull. "Knowledge Discovery from Medical Data: An Empirical Study with XCS". English. In: *Learning Classifier Systems in Data Mining.* Vol. 125. Studies in Computational Intelligence. Springer, 2008, pp. 93–121.

[KH93] L. T. Kóczy and K. Hirota. "Interpolative reasoning with insufficient evidence in sparse fuzzy rule bases". In: *Information Sciences* 71.1 (1993), pp. 169–201.

[Kov98] T. Kovacs. "XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions". In: *Soft Computing in Engineering Design and Manufacturing.* Springer London, 1998, pp. 59–68.

[Kov00a] T. Kovacs. "Strength or Accuracy? Fitness Calculation in Learning Classifier Systems". In: *Learning Classifier Systems: From Foundations to Applications.* Ed. by P. L. Lanzi, W. Stolzmann, and S. W. Wilson. Springer Berlin Heidelberg, 2000, pp. 143–160.

[Kov00b] T. Kovacs. "Towards a Theory of Strong Overgeneral Classifiers". In: *Proceedings of the Sixth Workshop on Foundations of Genetic Algorithms.* Charlottesville, VA, USA, July 2000, pp. 165–184.

[Kov12] T. Kovacs. "Genetics-Based Machine Learning". In: *Handbook of Natural Computing.* Springer Berlin Heidelberg, 2012, pp. 937–986.

[KB07] T. Kovacs and L. Bull. “Toward a better understanding of rule initialisation and deletion”. In: *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO) Companion.* ACM. 2007, pp. 2777–2780.

[Kra16] B. Krawczyk. “Learning from imbalanced data: open challenges and future directions”. In: *Progress in Artificial Intelligence* 5.4 (Nov. 2016), pp. 221–232.

[Kru+15] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker. “A survey on engineering approaches for self-adaptive systems”. In: *Pervasive and Mobile Computing* 17, Part B (2015). 10 years of Pervasive Computing’ In Honor of Chatschik Bisdikian, pp. 184–206.

[Lan98a] P. L. Lanzi. “Adding memory to XCS”. In: *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence.* May 1998, pp. 609–614.

[Lan02] P. L. Lanzi. “Learning classifier systems from a reinforcement learning perspective”. In: *Soft Computing* 6.3 (June 2002), pp. 162–170.

[Lan+06] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. “Prediction update algorithms for XCSF: RLS, kalman filter, and gain adaptation”. In: *Proc. of GECCO 2006.* ACM. 2006.

[Lan98b] P. L. Lanzi. “An analysis of the memory mechanism of XCSM”. In: *Genetic Programming* 98 (1998), pp. 643–651.

[LL06] P. L. Lanzi and D. Loiacono. “XCSF with neural prediction”. In: *Proc. of Congress on Evolutionary Computation (CEC) 2006.* IEEE. 2006.

[Lan+05a] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. *Generalization in the XCSF Classifier System: Analysis, Improvement, and Extension.* IlliGAL Rep. 2005012. Illinois Genetic Algorithms Lab, University of Illinois Urbana-Champaign, 2005.

[Lan+05b] P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. “Extending XCSF beyond linear approximation”. In: *Proc. of GECCO 2005.* ACM. 2005, pp. 1827–1834.

[LR00] P. L. Lanzi and R. L. Riolo. “A Roadmap to the Last Decade of Learning Classifier System Research”. In: *Learning Classifier Systems, From Foundations to Applications.* Berlin, Heidelberg: Springer-Verlag, 2000, pp. 33–62.

[LSW00] P. L. Lanzi, W. Stolzmann, and S. W. Wilson, eds. *Learning Classifier Systems, From Foundations to Applications.* Vol. 1813. Lecture Notes in Computer Science. Springer, 2000.

[LW00] P. L. Lanzi and S. W. Wilson. “Toward Optimal Classifier System Performance in Non-Markov Environments”. In: *Evol. Comput.* 8.4 (Dec. 2000), pp. 393–418.

[LW06] P. L. Lanzi and S. W. Wilson. “Using Convex Hulls to Represent Classifier Conditions”. In: *Proc. of GECCO ’06.* 2006, pp. 1481–1488.

[LG04] H. Ledoux and C. Gold. “An Efficient Natural Neighbour Interpolation Algorithm for Geoscientific Modelling”. In: *Proc. 11th International Symosium on Spatial Data Handling.* 2004, pp. 23–25.

[LS08] J. Lehman and K. Stanley. “Exploiting open-endedness to solve problems through the search for novelty”. In: *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems.* Cambridge, MA: MIT Press, Cambridge, MA, 2008, pp. 329–336.

[LS11] J. Lehman and K. O. Stanley. “Abandoning Objectives: Evolution Through the Search for Novelty Alone”. In: *Evolutionary Computation* 19.2 (2011), pp. 189–223.

[LG94] D. Lewis and W. Gale. “A sequential algorithm for training text classifiers”. In: *Proc. of Conf. on Res and Devel. in Inf. Retrieval.* ACM, 1994, pp. 3–12.

[LH10] L. Liang and D. Hale. *A Stable and Fast Implementation of Natural Neighbor Interpolation.* Tech. rep. Colorado School of Mines,Golden, CO 80401, USA: Center for Wave Phenomena, 2010.

[Lic13] M. Lichman. *UCI Machine Learning Repository.* Online: `http://archive.ics.uci.edu/ml` (last accessed 14.08.2019). 2013.

[LW93] A. Linden and F. Weber. “Implementing Inner Drive Through Competence Reflection”. In: *Proc. of 2nd International Conference on Simulation of Adaptive Behavior: From Animals to Animats 2.* Honolulu, Hawai, USA: MIT Press, 1993, pp. 321–326.

[LXB17] Y. Liu, B. Xue, and W. N. Browne. “Visualisation and Optimisation of Learning Classifier Systems for Multiple Domain Learning”. In: *Simulated Evolution and Learning.* Ed. by Y. Shi, K. C. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, and Y. Jin. Cham: Springer International Publishing, 2017, pp. 448–461.

[LML07] D. Loiacono, A. Marelli, and P. L. Lanzi. “Support vector regression for classifier prediction”. In: *Proc. of GECCO 2007.* ACM. 2007, pp. 1806–1813.

[Lóp+13] V. López, A. Fernández, S. García, V. Palade, and F. Herrera. “An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics”. In: *Information Sciences* 250 (2013), pp. 113–141.

[MC04] L. Macedo and A. Cardoso. “Exploration of Unknown Environments with Motivational Agents”. In: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS ’04. New York, New York: IEEE Computer Society, 2004, pp. 328–335.

[MT03] G. G. Maisuradze and D. L. Thompson. “Interpolating Moving Least-Squares Methods for Fitting Potential Energy Surfaces: illustrative approaches and applications”. In: *The Journal of Physical Chemistry A* 107.37 (2003), pp. 7118–7124.

[MSW95] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. “Breast Cancer Diagnosis and Prognosis Via Linear Programming”. In: *Operations Research* 43.4 (1995), pp. 570–577.

[Mar09] S. Marsland. *Machine Learning: An Algorithmic Perspective*. 1st. Chapman & Hall/CRC, 2009.

[Mat+18] K. Matsumoto, R. Takano, T. Tatsumi, H. Sato, T. Kovacs, and K. Takadama. “XCSR based on compressed input by deep neural network for high dimensional data”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO ’18. Kyoto, Japan: ACM, July 2018, pp. 1418–1425.

[Mei17] D. Meier. “Ein Werkzeug zur Erstellung mathematischer Funktionen unterschiedlicher Charakteristika zur Modellierung von maschinellen Lernproblemen”. Advisor: Anthony Stein. Master’s Thesis. University of Augsburg, Department of Computer Science, 2017.

[Men17] S. Menssen. “Der Einsatz von Interpolation als Methode für die stückweise lokale Approximation im XCSF”. Advisor: Anthony Stein. Bachelor’s Thesis. University of Augsburg, Department of Computer Science, 2017.

[Mit97] T. M. Mitchell. *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[Mni+15] V. Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533.

[MRT12] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. Adaptive computation and machine learning. MIT Press, 2012.

[Mol19] C. Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. Online: `https://christophm.github.io/interpretable-ml-book/` (last accessed 14.08.2019). 2019.

[MSU11] C. Müller-Schloer, H. Schmeck, and T. Ungerer, eds. *Organic Computing – A Paradigm Shift for Complex Systems*. Birkhäuser Verlag, 2011.

[MT17a] C. Müller-Schloer and S. Tomforde. *Organic Computing – Technical Systems for Survival in the Real World.* Birkhäuser Verlag, 2017.

[MT17b] C. Müller-Schloer and S. Tomforde. “The Major Context”. In: *Organic Computing – Technical Systems for Survival in the Real World.* Birkhäuser Verlag, 2017, pp. 549–572.

[NSC15a] A. Najar, O. Sigaud, and M. Chetouani. “Social-Task Learning for HRI”. In: *Social Robotics: 7th International Conference, ICSR 2015, Paris, France, October 26-30, 2015.* Cham: Springer International Publishing, 2015, pp. 472–481.

[NSC15b] A. Najar, O. Sigaud, and M. Chetouani. “Socially Guided XCS: Using Teaching Signals to Boost Learning”. In: *Proceedings of the Companion Publication of GECCO'15.* GECCO Companion '15. Madrid, Spain: ACM, 2015, pp. 1021–1028.

[Nak+15] M. Nakata, P. L. Lanzi, T. Kovacs, W. N. Browne, and K. Takadama. “How should learning classifier systems cover a state-action space?” In: *2015 IEEE Congress on Evolutionary Computation (CEC).* May 2015, pp. 3012–3019.

[NBH18] M. Nakata, W. N. Browne, and T. Hamagami. “Theoretical adaptation of multiple rule-generation in XCS”. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* GECCO '18. Kyoto, Japan: ACM, July 2018, pp. 482–489.

[Nak+17] M. Nakata, W. N. Browne, T. Hamagami, and K. Takadama. “Theoretical XCS parameter settings of learning accurate classifiers”. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* GECCO '17. Berlin, Germany: ACM, July 2017, pp. 473–480.

[NCK11] H. M. Nguyen, E. W. Cooper, and K. Kamei. “Online Learning from Imbalanced Data Streams”. In: *Soft Computing and Pattern Recognition (SoCPaR), 2011 International Conference of.* Oct. 2011, pp. 347–352.

[Nin85] Nintendo. *Super Mario Bros.* Video Game for the Nintendo Entertainment System, Japan. 1985.

[Orr+09] A. Orriols-Puig, E. Bernado-Mansilla, D. E. Goldberg, K. Sastry, and P. L. Lanzi. “Facetwise Analysis of XCS for Problems With Class Imbalances”. In: *IEEE Transactions on Evolutionary Computation* 13.5 (Oct. 2009), pp. 1093–1119.

[OB06] A. Orriols-Puig and E. Bernadó-Mansilla. “Bounding XCS's Parameters for Unbalanced Datasets”. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation.* GECCO '06. Seattle, Washington, USA: ACM, 2006, pp. 1561–1568.

[OB08] A. Orriols-Puig and E. Bernadó-Mansilla. “Evolutionary rule-based systems for imbalanced data sets”. In: *Soft Computing* 13.3 (2008), p. 213.

[OCB08] A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. “Genetic-based Machine Learning Systems are Competitive for Pattern Recognition”. In: *Evolutionary Intelligence* 1.3 (2008), pp. 209–232.

[Orr+07] A. Orriols-Puig, D. E. Goldberg, K. Sastry, and E. Bernadó-Mansilla. “Modeling XCS in Class Imbalances: Population Size and Parameter Settings”. In: *Proc. of GECCO ’07*. London, England: ACM, 2007, pp. 1838–1845.

[OB05] A. Orriols and E. Bernado-Mansilla. “Class Imbalance Problem in UCS Classifier System: Fitness Adaptation”. In: *2005 IEEE Congress on Evolutionary Computation*. Vol. 1. Sept. 2005, pp. 604–611.

[OKH07] P. Y. Oudeyer, F. Kaplan, and V. V. Hafner. “Intrinsic Motivation Systems for Autonomous Mental Development”. In: *IEEE Transactions on Evolutionary Computation* 11.2 (Apr. 2007), pp. 265–286.

[Oud04] P.-Y. Oudeyer. “Intelligent Adaptive Curiosity: a source of Self-Development”. In: *Proceedings of the Fourth International Workshop on Epigenetic Robotics*. Ed. by L. Berthouze, H. Kozima, C. G. Prince, G. Sandini, G. Stojanov, G. Metta, and C. Balkenius. Vol. 117. Lund University Cognitive Studies, 2004, pp. 127–130.

[OR01] N. C. Oza and S. Russell. “Experimental Comparisons of Online and Batch Versions of Bagging and Boosting”. In: *Proc. of the 7th ACM SIGKDD*. San Francisco, California: ACM, 2001, pp. 359–364.

[PSH19] D. Pätzel, A. Stein, and J. Hähner. “A Survey on Formal Theoretical Advances Regarding XCS”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO ’19. accepted, to appear. Prague, Czech Republic: ACM, 2019.

[Pen95] J. Peng. “Efficient Memory-based Dynamic Programming”. In: *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*. ICML’95. Tahoe City, California, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 438–446.

[PK15] T. Pickering and T. Kovacs. “TP-XCS: An XCS classifier system with fixed-length memory for reinforcement learning”. In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. May 2015, pp. 3020–3025.

[Pol+01] R. Polikar, L. Upda, S. S. Upda, and V. Honavar. “Learn++: an incremental learning algorithm for supervised neural networks”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 31.4 (Nov. 2001), pp. 497–508.

[Pro+09] H. Prothmann, J. Branke, H. Schmeck, S. Tomforde, F. Rochner, J. Hähner, and C. Müller-Schloer. "Organic Traffic Light Control for Urban Road Networks". In: *International Journal of Autonomous and Adaptive Communications Systems* 2.3 (2009), pp. 203–225.

[Pro+08] H. Prothmann, F. Rochner, S. Tomforde, J. Branke, C. Müller-Schloer, and H. Schmeck. "Organic Control of Traffic Lights". In: *Proceedings of the 5th International Conference on Autonomic and Trusted Computing (ATC-08).* Ed. by C. Rong et al. Vol. 5060. LNCS. Oslo, Norway: Springer Verlag, June 2008, pp. 219–233.

[Pro11] H. Prothmann. *Organic Traffic Control.* KIT Scientific Publishing, Karlsruhe, 2011, pp. 1–279.

[Pro+11] H. Prothmann, S. Tomforde, J. Branke, J. Hähner, C. Müller-Schloer, and H. Schmeck. "Organic Traffic Control". In: *Organic Computing – A Paradigm Shift for Complex Systems.* Ed. by C. Müller-Schloer, H. Schmeck, and T. Ungerer. Basel, CH: Birkhäuser Verlag, 2011, pp. 431–446.

[PF13] F. Provost and T. Fawcett. *Data Science for Business: What You Need to Know About Data Mining and Data-analytic Thinking.* 1st. O'Reilly Media, Inc., 2013.

[Rau16] D. Rauh. "Der Einsatz von RBF-basierter Interpolation im Status Quo des XCSF". Advisor: Anthony Stein. Master's Thesis. University of Augsburg, Department of Computer Science, 2016.

[Rei+16] W. Reif, G. Anders, H. Seebach, J.-P. Steghöfer, E. André, J. Hähner, C. Müller-Schloer, and T. Ungerer. *Trustworthy Open Self-Organising Systems.* 1st. Basel, CH: Birkhäuser Verlag, 2016.

[RCS15] T. Reitmaier, A. Calma, and B. Sick. "Transductive active learning – A new semi-supervised learning approach based on iteratively refined generative models to capture structure in data". In: *Inf. Sci.* 293 (2015), pp. 275–298.

[Ren88] R. J. Renka. "Multivariate Interpolation of Large Sets of Scattered Data". In: *ACM Trans. Math. Softw.* 14.2 (1988), pp. 139–148.

[Ric+06] U. Richter, M. Mnif, J. Branke, C. Müller-Schloer, and H. Schmeck. "Towards a Generic Observer/Controller Architecture for Organic Computing". In: *GI Jahrestagung (1).* 2006, pp. 112–119.

[Row07] J. Rowley. "The wisdom hierarchy: representations of the DIKW hierarchy". In: *Journal of information science* 33.2 (2007), pp. 163–180.

[Rud+15] S. Rudolph, S. Tomforde, B. Sick, and J. Hähner. “A Mutual Influence Detection Algorithm for Systems with Local Performance Measurement”. In: *Proceedings of the 9th IEEE International Conference on Self-adapting and Self-organising Systems.* SASO ’15. Bosten, USA, Sept. 2015, pp. 144–150.

[RN95] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[SBM95] M. Sambridge, J. Braun, and H. McQueen. “Geophysical Parameterization and Interpolation of Irregular Data using Natural Neighbours”. In: *Geophysical Journal International* 122.3 (1995), pp. 837–857.

[Sca+19] S. Scardapane, M. Scarpiniti, D. Comminiello, and A. Uncini. “Learning Activation Functions from Data Using Cubic Spline Interpolation”. In: *Neural Advances in Processing Nonlinear Dynamic Signals.* Ed. by A. Esposito, M. Faundez-Zanuy, F. C. Morabito, and E. Pasero. Cham: Springer International Publishing, 2019, pp. 73–83.

[SA94] S. Schaal and C. G. Atkeson. “Robot juggling: implementation of memory-based learning”. In: *IEEE Control Systems Magazine* 14.1 (Feb. 1994), pp. 57–71.

[Sch+15] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. “Prioritized experience replay”. In: *arXiv preprint arXiv:1511.05952* (2015).

[Sch91] J. Schmidhuber. “Curious model-building control systems”. In: *Proc. of IEEE International Joint Conference on Neural Networks.* 1991, 1458–1463 vol.2.

[Sch90] J. Schmidhuber. “A Possibility for Implementing Curiosity and Boredom in Model-building Neural Controllers”. In: *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats.* Paris, France: MIT Press, 1990, pp. 222–227.

[SK11] H. R. Schwarz and N. Köckler. “Interpolation und Approximation”. German. In: *Numerische Mathematik.* Vieweg+Teubner Verlag, 2011, pp. 91–182.

[SS05] D. W. Scott and S. R. Sain. “Multidimensional density estimation”. In: *Handbook of statistics* 24 (2005), pp. 229–261.

[Set09] B. Settles. *Active Learning Literature Survey.* Computer Sciences Technical Report 1648. University of Wisconsin, Department of Computer Science, 2009.

[SOS92] H. S. Seung, M. Opper, and H. Sompolinsky. “Query by Committee”. In: *Proc. of the 5th Annual Workshop on Computational Learning Theory.* COLT ’92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 287–294.

[SB14] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* New York, NY, USA: Cambridge University Press, 2014.

[Sha48] C. E. Shannon. “A Mathematical Theory of Communication”. In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[She68] D. Shepard. “A Two-dimensional Interpolation Function for Irregularly-spaced Data”. In: *Proceedings of the 1968 23rd ACM National Conference.* ACM ’68. New York, NY, USA: ACM, 1968, pp. 517–524.

[Sib81] R. Sibson. “A Brief Description of Natural Neighbour Interpolation”. In: *Interpreting Multivariate Data.* Vol. 21. John Wiley & Sons, 1981, pp. 21–36.

[Sic+18] B. Sick, S. Oeste-Reiß, A. Schmidt, S. Tomforde, and A. K. Zweig. “Collaborative Interactive Learning”. In: *Informatik-Spektrum* 41.1 (Feb. 2018), pp. 52–55.

[SW07] O. Sigaud and S. W. Wilson. “Learning Classifier Systems: A Survey.” In: *Soft Comput.* 11.11 (2007), pp. 1065–1078.

[Ska13] V. Skala. “Fast interpolation and approximation of scattered multidimensional and dynamic data using radial basis functions”. In: *WSEAS Transactions on Mathematics* 12.5 (2013), pp. 501–511.

[Smi80] S. F. Smith. “A Learning System Based on Genetic Adaptive Algorithms”. PhD thesis. Pittsburgh, PA, USA: University of Pittsburgh, 1980.

[SSH16a] M. Sommer, A. Stein, and J. Hähner. “Ensemble Time Series Forecasting with XCSF”. In: *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems.* SASO ’16. Sept. 2016, pp. 150–151.

[SSH16b] M. Sommer, A. Stein, and J. Hähner. “Local ensemble weighting in the context of time series forecasting using XCSF”. In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI).* Dec. 2016, pp. 1–8.

[STH16] M. Sommer, S. Tomforde, and J. Hähner. “An Organic Computing Approach to Resilient Traffic Management”. In: *Autonomic Road Transport Support Systems.* Springer International, 2016, pp. 113–130.

[Sta14] P. Stalph. *Analysis and design of machine learning techniques: evolutionary solutions for regression, prediction, and control problems.* Springer Science & Business Media, 2014.

[SB10a] P. O. Stalph and M. V. Butz. “Current XCSF Capabilities and Challenges”. In: *Learning Classifier Systems.* Vol. 6471. LNCS. Springer Berlin Heidelberg, 2010, pp. 57–69.

[SB10b] P. O. Stalph and M. V. Butz. “How Fitness Estimates Interact with Reproduction Rates: Towards Variable Offspring Set Sizes in XCSF”. In: *Learning Classifier Systems.* Berlin, Heidelberg: Springer, 2010, pp. 47–56.

[SB12] P. O. Stalph and M. V. Butz. “Guided Evolution in XCSF”. In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation.* Proc. of GECCO ’12. Philadelphia, Pennsylvania, USA: ACM, 2012, pp. 911–918.

[Sta+12a] P. O. Stalph, X. Llora, D. E. Goldberg, and M. V. Butz. “Resource management and scalability of the XCSF learning classifier system”. In: *Theoretical Computer Science* 425 (2012), pp. 126–141.

[Sta+12b] P. O. Stalph, J. Rubinsztajn, O. Sigaud, and M. V. Butz. “Function approximation with LWPR and XCSF: a comparative study”. In: *Evolutionary Intelligence* 5.2 (June 2012), pp. 103–116.

[Sta+19] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen. “Designing neural networks through neuroevolution”. In: *Nature Machine Intelligence* 1.1 (2019), pp. 24–35.

[Ste+16a] A. Stein, S. Tomforde, D. Rauh, and J. Hähner. “Dealing with Unforeseen Situations in the Context of Self-Adaptive Urban Traffic Control: How to Bridge the Gap?” In: *Proc. of 2016 IEEE International Conference on Autonomic Computing (ICAC).* July 2016, pp. 167–172.

[Ste17] A. Stein. “Reaction Learning”. In: *Organic Computing – Technical Systems for Survival in the Real World.* Ed. by C. Müller-Schloer and S. Tomforde. Birkhäuser Verlag, 2017. Chap. Basic Methods, pp. 287–328.

[SMH17] A. Stein, R. Maier, and J. Hähner. “Toward Curious Learning Classifier Systems: Combining XCS with Active Learning Concepts”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion.* GECCO ’17. Berlin, Germany: ACM, 2017, pp. 1349–1356.

[SMH18] A. Stein, S. Menssen, and J. Hähner. “What About Interpolation? A Radial Basis Function Approach to Classifier Prediction Modeling in XCSF”. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* GECCO ’18. Kyoto, Japan: ACM, 2018, pp. 537–544.

[Ste+16b] A. Stein, D. Rauh, S. Tomforde, and J. Hähner. “Augmenting the Algorithmic Structure of XCS by Means of Interpolation”. In: *Proc. of 29th International Conference on Architecture of Computing Systems.* Nuremberg, Germany, Apr. 2016, pp. 348–360.

[Ste+17a] A. Stein, D. Rauh, S. Tomforde, and J. Hähner. “Interpolation in the eXtended Classifier System: An architectural perspective”. In: *Journal of Systems Architecture* 75 (2017), pp. 79–94.

[Ste+17b] A. Stein, S. Rudolph, S. Tomforde, and J. Hähner. "Self-Learning Smart Cameras – Harnessing the Generalization Capability of XCS". In: *Proceedings of International Joint Conference on Computational Intelligence IJCCI 2017*. Funchal, Madeira - Portugal, Nov. 2017, pp. 129–140.

[Ste+18] A. Stein, S. Tomforde, A. Diaconescu, J. Hähner, and C. Müller-Schloer. "A Concept for Proactive Knowledge Construction in Self-Learning Autonomous Systems". In: *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. 2018.

[Sto99] W. Stolzmann. "An Introduction to Anticipatory Classifier Systems". In: *Learning Classifier Systems, From Foundations to Applications.* 1999, pp. 175–194.

[SB03] C. Stone and L. Bull. "For Real! XCS with Continuous-Valued Inputs." In: *Evolutionary Computation* 11.3 (2003), pp. 298–336.

[ST90] M. A. Styblinski and T. S. Tang. "Experiments in Nonconvex Optimization: Stochastic Approximation with Function Smoothing and Simulated Annealing". In: *Neural Netw.* 3.4 (July 1990), pp. 467–483.

[SB98] R. Sutton and A. Barto. *Reinforcement learning: An introduction.* Vol. 116. Cambridge University Press, 1998.

[Sut91] R. S. Sutton. "Dyna, an integrated architecture for learning, planning, and reacting". In: *ACM SIGART Bulletin* 2.4 (1991), pp. 160–163.

[SC06] J. Sylvester and N. V. Chawla. "Evolutionary ensemble creation and thinning". In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings.* IEEE. 2006, pp. 5148–5155.

[SS04] C. Szepesvári and W. D. Smart. "Interpolation-based Q-learning". In: *Proceedings of the Twenty-first International Conference on Machine Learning.* ICML '04. Banff, Alberta, Canada: ACM, 2004.

[TMU13] J. Tan, J. Moore, and R. Urbanowicz. "Rapid Rule Compaction Strategies for Global Knowledge Discovery in a Supervised Learning Classifier System". In: *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)* (2013), pp. 110–117.

[Ten00] D. Tennenhouse. "Proactive Computing". In: *Communications of the ACM* 43.5 (2000), pp. 43–50.

[Tha+09] W. I. Thacker, J. Zhang, L. T. Watson, J. B. Birch, M. A. Iyer, and M. W. Berry. *Algorithm XXX: SHEPPACK: Modified Shepard Algorithm for Interpolation of Scattered Multivariate Data.* Departmental Technical Report TR 09-13. Department of Computer Science, Virginia Polytechnic Institute & State University, 2009.

[Tho+13] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. “Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms”. In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD ’13. Chicago, Illinois, USA: ACM, 2013, pp. 847–855.

[Tom11] S. Tomforde. “An architectural framework for self-configuration and self-improvement at runtime”. PhD thesis. Leibniz Universität Hannover, Institute for Systems Engineering, System and Computer Architecture Group, 2011.

[Tom12] S. Tomforde. *Runtime Adaptation of Technical Systems: An Architectural Framework for Self-configuration and Self-improvement at Runtime.* Südwestdeutscher Verlag für Hochschulschriften AG Company KG, 2012.

[Tom+11a] S. Tomforde, A. Brameshuber, J. Hähner, and C. Müller-Schloer. “Restricted On-line Learning in Real-world Systems”. In: *Proc. of the IEEE Congress on Evolutionary Computation (CEC ’11).* New Orleans, USA: IEEE, June 2011, pp. 1628–1635.

[TCH09] S. Tomforde, E. Cakar, and J. Hähner. “Dynamic Control of Network Protocols - A new vision for future self-organised networks”. In: *Proceedings of the 6th International Conference on Informatics in Control, Automation, and Robotics (ICINCO’09).* Ed. by J. Filipe, J. A. Cetto, and J.-L. Ferrier. Milan, Italy: INSTICC, July 2009, pp. 285–290.

[TH11] S. Tomforde and J. Hähner. “Organic Network Control – Turning Standard Protocols Into Evolving Systems”. In: *Biologically Inspired Networking and Sensing: Algorithms and Architectures.* IGI, 2011, pp. 11–35.

[Tom+14] S. Tomforde, J. Hähner, H. Seebach, W. Reif, B. Sick, A. Wacker, and I. Scholtes. “Engineering and Mastering Interwoven Systems”. In: *ARCS 2014 - 27th International Conference on Architecture of Computing Systems, Workshop Proceedings.* Luebeck, Germany, Feb. 2014, pp. 1–8.

[Tom+11b] S. Tomforde, H. Prothmann, J. Branke, J. Hähner, M. Mnif, C. Müller-Schloer, U. Richter, and H. Schmeck. “Observation and Control of Organic Systems”. In: *Organic Computing – A Paradigm Shift for Complex Systems.* Ed. by C. Müller-Schloer, H. Schmeck, and T. Ungerer. Autonomic Systems. Birkhäuser Verlag, 2011, pp. 325–338.

[THH11] S. Tomforde, B. Hurling, and J. Hähner. “Distributed Network Protocol Parameter Adaptation in Mobile Ad-Hoc Networks”. In: *Informatics in Control, Automation and Robotics.* Vol. 89. LNEE. Springer, 2011, pp. 91–104.

[TSD08] T. H. Tran, C. Sanza, and Y. Duthen. "Evolving prediction weights using evolution strategy". In: *Proc. of GECCO 2008*. ACM. 2008, pp. 2009–2016.

[Tra00] Transportation Research Board. *Highway Capacity Manual*. Tech. rep. Washington D.C., US: National Research Council, 2000.

[UBM14] R. J. Urbanowicz, G. Bertasius, and J. H. Moore. "An Extended Michigan-Style Learning Classifier System for Flexible Supervised Learning, Classification, and Data Mining". In: *Parallel Problem Solving from Nature – PPSN XIII*. Ed. by T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith. Cham: Springer International Publishing, 2014, pp. 211–221.

[UB17] R. J. Urbanowicz and W. N. Browne. *Introduction to Learning Classifier Systems*. 1st. Springer Publishing Company, Incorporated, 2017.

[UGM12a] R. J. Urbanowicz, D. Granizo-Mackenzie, and J. H. Moore. "Using Expert Knowledge to Guide Covering and Mutation in a Michigan Style Learning Classifier System to Detect Epistasis and Heterogeneity". In: *Parallel Problem Solving from Nature - PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I*. Ed. by C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone. Springer Berlin Heidelberg, 2012, pp. 266–275.

[Urb+18] R. J. Urbanowicz, C. Lo, J. H. Holmes, and J. H. Moore. "Attribute Tracking: Strategies Towards Improved Detection and Characterization of Complex Associations". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '18. Kyoto, Japan: ACM, 2018, pp. 553–560.

[UM10] R. J. Urbanowicz and J. H. Moore. "The Application of Michigan-style Learning Classifiersystems to Address Genetic Heterogeneity and Epistasisin Association Studies". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. GECCO '10. Portland, Oregon, USA: ACM, 2010, pp. 195–202.

[UM15] R. J. Urbanowicz and J. H. Moore. "ExSTraCS 2.0: description and evaluation of a scalable learning classifier system". In: *Evolutionary Intelligence* 8.2 (2015), pp. 89–116.

[UM09] R. J. Urbanowicz and J. H. Moore. "Learning Classifier Systems: A Complete Introduction, Review, and Roadmap". In: *Journal of Artificial Evolution and Applications* 2009 (2009), p. 1.

[UGM12b] R. Urbanowicz, A. Granizo-Mackenzie, and J. Moore. "Instance-linked Attribute Tracking and Feedback for Michigan-style Supervised Learning Classifier Systems". In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. GECCO '12. Philadelphia, Pennsylvania, USA: ACM, 2012, pp. 927–934.

[Vap98] V. N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

[Ven94] G. Venturini. “Apprentissage adaptatif et apprentissage supervise par algorithme genetique”. PhD thesis. Paris 11, 1994.

[Vin17] D. Vincze. “Fuzzy rule interpolation and reinforcement learning”. In: *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI).* Jan. 2017, pp. 173–178.

[Wag17] A. Wagner. “Der Einsatz von konvexen Hüllen für die gesicherte Interpolation in Learning Classifier Systemen”. Advisor: Anthony Stein. Bachelor’s Thesis. University of Augsburg, Department of Computer Science, 2017.

[Wan+18] B. Wang, X. Luo, Z. Li, W. Zhu, Z. Shi, and S. Osher. “Deep Neural Nets with Interpolating Function as Output Activation”. In: *Advances in Neural Information Processing Systems 31.* Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 743–753.

[WMY13] S. Wang, L. L. Minku, and X. Yao. “A Learning Framework for Online Class Imbalance Learning”. In: *Computational Intelligence and Ensemble Learning (CIEL), 2013 IEEE Symposium on.* Apr. 2013, pp. 36–45.

[WMY15] S. Wang, L. L. Minku, and X. Yao. “Resampling-Based Ensemble Methods for Online Class Imbalance Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 27.5 (May 2015), pp. 1356–1368.

[WD92] C. J. Watkins and P. Dayan. “Q-learning”. In: *Machine learning* 8.3 (1992), pp. 279–292.

[Web+16] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean. “Characterizing concept drift”. In: *Data Mining and Knowledge Discovery* (2016), pp. 1–31.

[Web58] F. V. Webster. *Traffic Signal Settings.* Road Research Technical Paper No. 39, Road Research Laboratory, England, published by HMSO, 1958.

[Wei04] G. M. Weiss. “Mining with Rarity: A Unifying Framework”. In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 7–19.

[WH88] B. Widrow and M. E. Hoff. “Adaptive switching circuits”. In: *Neurocomputing: Foundations of Research.* Ed. by J. A. Anderson and E. Rosenfeld. Cambridge, MA, USA: MIT Press, 1988. Chap. Adaptive Switching Circuits, pp. 123–134.

[Wik] Wikipedia. *Styblinski-Tang Function.* Licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license. Last accessed: July 29, 2019.

[Wil16] P. Williams. “SINN: Shepard Interpolation Neural Networks”. In: *Advances in Visual Computing.* Ed. by G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger, and T. Isenberg. Cham: Springer International Publishing, 2016, pp. 349–358.

[Wil94] S. W. Wilson. “ZCS: A Zeroth Level Classifier System.” In: *Evolutionary Computation* 2.1 (1994), pp. 1–18.

[Wil95] S. W. Wilson. “Classifier Fitness Based on Accuracy”. In: *Evolutionary Computation* 3.2 (1995), pp. 149–175.

[Wil98] S. W. Wilson. “Generalization in the XCS Classifier System”. In: *Genetic Programming 1998: Proc. of 3rd Annual Conf.* 1998, pp. 665–674.

[Wil01] S. W. Wilson. “Mining Oblique Data with XCS”. In: *Advances in Learning Classifier Systems.* Vol. 1996. LNCS. Springer, 2001, pp. 158–174.

[Wil02] S. W. Wilson. “Classifiers that Approximate Functions”. In: *Natural Computing* 1.2-3 (2002), pp. 211–234.

[Wil07] S. W. Wilson. “Three Architectures for Continuous Action”. In: *Learning Classifier Systems.* Vol. 4399. LNCS. Springer, 2007, pp. 239–257.

[WG89] S. W. Wilson and D. E. Goldberg. “A Critical Review of Classifier Systems”. In: *Proceedings of the 3rd International Conference on Genetic Algorithms.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 244–255.

[Wil00] S. Wilson. “Get Real! XCS with Continuous-Valued Inputs”. In: *Learning Classifier Systems.* Ed. by P. Lanzi, W. Stolzmann, and S. Wilson. Vol. 1813. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, pp. 209–219.

[WN99] S. Wright and J. Nocedal. “Numerical optimization”. In: *Springer Science* 35 (1999), pp. 67–68.

[WM13] Q. Wu and C. Miao. “Curiosity: From Psychology to Computation”. In: *ACM Comput. Surv.* 46.2 (Dec. 2013), 18:1–18:26.

[XLY99] H. Xi, Y. Luo, and S. Yang. “An approach to active learning for classifier systems”. In: *Journal of Computer Science and Technology* 14.4 (July 1999), pp. 372–378.

[Yao99] X. Yao. “Evolving artificial neural networks”. In: *Proceedings of the IEEE* 87.9 (Sept. 1999), pp. 1423–1447.

[Zep+11] J. Zeppenfeld, A. Bouajila, W. Stechele, A. Bernauer, O. Bringmann, W. Rosenstiel, and A. Herkersdorf. “Applying ASoC to Multi-core Applications for Workload Management”. In: *Organic Computing – A Paradigm Shift for Complex Systems*. Basel, CH: Birkhäuser Verlag, 2011, pp. 461–472.

[ZH11] J. Zeppenfeld and A. Herkersdorf. “Applying Autonomic Principles for Workload Management in Multi-core Systems on Chip”. In: *Proc. of 2011 IEEE International Conference on Autonomic Computing (ICAC)*. 2011, pp. 3–10.