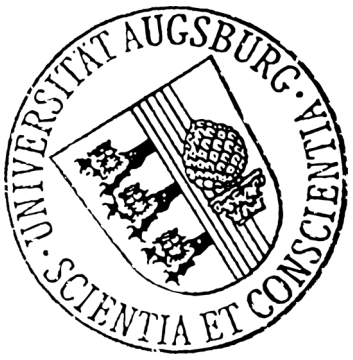
Bachelorarbeit

Decision Forests for Regression

Moritz Einfalt

30.09.2014



Universität Augsburg
Fakultät für Angewandte Informatik
Multimedia Computing and
Computer Vision Lab

Reviewer
Prof. Dr. Rainer Lienhart

Second Reviewer
Prof. Dr. Elisabeth André

Supervisor
Dipl. Inf. Christoph Lassner

Abstract

While decision trees have been a well known approach to classification related tasks for a long time, their popularity further increased with the emerging decision forests – ensembles of decision trees. In 2013, Criminisi et al. developed a unifying forest model with the intent to apply decision forest induction not only to classification but to a variety of different machine learning disciplines, including regression.

This work has the goal to analyse regression forests based on the unified model and to discuss the most important aspects of an implementation that were not described in the original publication from Criminisi et al. The implementation provided alongside this work is part of the fertilized project, a library for decision forests which uses the unified model as its theoretical foundation. Before limiting the scope to regression, decision trees and forests are introduced in general by means of the unified model. This generic model then is instantiated for regression tasks using the theory of linear regression. Further remarks are made on the implementation focusing on error handling and efficiency. The conducted experiments involve two of the best known regression datasets that are widely used for benchmarking purposes. It is shown that the incorporation of linear regression into the forests can have a positive effect on the learning performance.

Kurzbeschreibung

Während Decision Trees schon seit langem als Ansatz für klassifikationsbasierte Aufgaben bekannt sind, gewannen sie noch weiter an Popularität mit dem Aufkommen von Decision Forests – der Kombination von Decision Trees. Criminisi et al. entwickelten im Jahr 2013 ein generisches Forest Modell mit der Absicht, Decision Forests nicht nur für Klassifikation sondern für eine Vielzahl von verschiedenen Disziplinen im Bereich des maschinellen Lernens einzusetzen, Regression miteingeschlossen.

Das Ziel dieser Arbeit ist es, Decision Forests für Regression, welche auf dem generischen Modell basieren zu analysieren und die wichtigsten Aspekte einer Implementierung zu diskutieren, die in der ursprünglichen Veröffentlichung von Criminisi et al. nicht enthalten sind. Die Implementierung, die begleitend zu dieser Arbeit zur Verfügung gestellt wird ist Teil des fertilized-Projekts, einer Bibliothek für Decision Forests welche das generische Forest Modell als theoretische Grundlage verwendet. Bevor der Fokus auf Regression gerichtet wird, werden Decision Trees und Forests im Allgemeinen anhand des generischen Modells vorgestellt. Dieses Modell wird dann für Regressionsaufgaben instanziiert, indem es mit der Theorie der linearen Regression kombiniert wird. Weiterhin werden Details der Implementierung bezüglich der Fehlerbehandlung und der Effizienz beschrieben. Die durchgeführten Experimente basieren auf zwei der bekanntesten Datensätze für Regression, deren Verwendung weit verbreitet ist um vergleichbare Regressionsergebnisse zu erhalten. Es wird gezeigt, dass die Verwendung von linearerer Regression in den Forests einen positiven Effekt auf die Lernergebnisse haben kann.

Contents

1 Introduction

Decision trees in the scope of machine learning are usually linked to classification tasks due to the popularity they have gained for classification-related applications. Nevertheless, the concept of decision trees is not limited to a specific form of application. In contrast to classification, regression is one of the fewer encountered problems in machine learning. Among a variety of machine learning methods for solving regression problems, the usage of decision trees is one possibility.

Since ensemble-based methods have constantly gained popularity in machine learning, decision trees are no exception. The combination of distinct decision trees into a single "forest" led to the concept of *Decision Forests*. This work has the objective to describe and evaluate decision forests for regression problems based on the unified forest model proposed by Criminisi et al. [CSK12]. In contrast to other forest models, this form of regression forests is based on linear regression. Of special interest is how this incorporation of linear regression models into decision forests affects their overall learning performance. Since a functional implementation of regression forests is provided alongside this work, some of the most important implementation details are discussed as well.

1.1 Evolution of Decision Trees for Machine Learning

In machine learning, different models of decision trees are usually denominated by the algorithm that is used for their computer-aided construction. Probably the best known work on decision trees used in machine learning is from Breiman et al. from 1984 [BFSO84]. The incorporated CART-algorithm is used to automatically generate binary decision trees for both classification and regression problems based on available training data. This work acted as the foundation for further research in decision trees for machine learning. The best known results incorporate the ID3-algorithm and its successor C4.5 from Quinlan [Qui93]. The decision trees constructed by these algorithms were no longer limited to a binary form. In order to keep the resulting decision trees comprehensible, the trees were no longer seen as one entity but rather as a set of rules which could be assessed one by one.

While decision trees for machine learning tasks have been known for a long time, their usage changed with the upcoming popularity of ensemble methods. Beginning with the work from Schapire [Sch90], the idea to combine several weak learners of low quality to obtain a single strong learner seemed to be a fruitful approach.

Before, decision trees have been seen as strong learners for themselves and quite a lot of optimization was done to increase their performance. Using the ensemble technique, decision forests as collections of trees established themselves. While it was not the first work mentioning forest-based learning, in [Bre01] tree ensembles were studied and compared to single decision trees which further increased their popularity. Instead of heavily optimizing each decision tree, rather simple trees were used as weak learners. By constructing each tree on a different, randomized subset of data, the trees differed in their output and its quality. Nevertheless, their combination led to an overall better performance compared to a single-tree approach, especially in terms of lesser over-fitting to the training data. Due to the randomization, these forests were also referred to as *Random Forests*. For consistency with [CSK12], the term *Decision Forests* will be used throughout this work. In the following, the structure of this work will be described.

1.2 Outline

Chapter 2 begins with a short introduction into the structure and function of decision trees in general. Afterwards, the unified model for decision forests from [CSK12] is presented. It uses the combination of multiple decision trees created through a randomized training process. Due to the generality of the model it is not limited to classification but can be used for a variety of machine learning tasks. In order to use this model for a specific application only small changes have to be applied to it.

Chapter 3 is dedicated to linear regression. After an introduction of regression in general and different regression models to solve such problems, linear regression is discussed in detail. Starting with the formulation of linear regression models and the incorporated assumptions, a way to find an optimal solution for such models based on available data is presented. Additionally, the quality of a linear regression model is derived using its statistical properties. This leads to the formulation of a probabilistic linear model.

In Chapter 4, regression forests are introduced. First, the need for more complex regression models compared to linear regression is motivated. Using the derivations on linear regression from the previous chapter, the unified forest model is instantiated for regression by refining the relevant parts of its definition. This includes a continuous form of entropy as well as probabilistic regression predictions.

Chapter 5 includes additional remarks related to the provided implementation of regression forests. It is described how the forest construction process can be kept functional in the presence of disadvantageously shaped data. Furthermore, an alternative forest construction is presented based on constant regression models. That way, the construction process can be kept feasible in the presence of large sets of training data but equally good regression results can be achieved at the same time.

In all the previous chapters, toy examples are used to demonstrate the discussed topics in a descriptive way. In Chapter 6, two real world datasets are introduced and the performance of the discussed regression forests on these datasets is presented. In order to obtain comparable results, a different form of regression forests, which is implemented in the *scikit-learn* project, is shorty introduced and used on the datasets as well. The results are discussed afterwards with possible explanations for the observable differences in the results.

Chapter 7 concludes this work. The main results are summarized and further research on regression forests is motivated.

2 The Unified Model for Decision Forests

While there is plenty of literature related to decision trees and forests, most of the work typically focused on classification tasks. One of the few exceptions can be seen in [BFSO84], where decision trees were used for classification as well as regression. Therefore, the decision forest model proposed in [CSK12] was a novelty. In their work, Criminisi et al. presented techniques to use forest-based learning not only for classification or regression but also for other machine learning tasks like density estimation or manifold learning. And yet, they did not use a unique forest structure and construction strategy for each of the different tasks. They introduced a unified approach to handle all these tasks with the same forest model with only slight modifications.

In this chapter, this unified model for decision forests is presented and discussed in detail. The following sections will describe the basics of decision trees, how they are constructed and in what way they are combined to form a single forest.

2.1 Decision Trees

Before looking at the exact tree definition used in [CSK12], the basic structure of decision trees is explained using a simple classification example. Afterwards, a short insight into the automated construction of decision trees is given.

2.1.1 Methodology of Decision Trees

In general, decision trees are used to obtain information about an object of interest that is unknown so far. To achieve this, a set of rules is assessed. Each of these rules consist of criterions for the object and its attributes and implicate a result. Based on whether the object complies with the criterions or not, one of the rules is applied to the object. The result of this rule is then presented as the desired information. A simple example would be as follows:

In a medical setting, information about the age of a male and whether he is overweight or not is available. Based on this information it has to be determined if

he has a high risk to suffer a heart attack or not. Basically, this is a classification problem with classes

$$c_1 := (\text{"High heart attack risk"}),$$
$$c_2 := (\text{"No high heart attack risk"}).$$

A set of rules to determine the class for a given male might be of the following form:

1. If his age is greater or equal than 75, he has a high heart attack risk (class c_1).
2. If his age is less than 75 and he is overweight, he has a high heart attack risk (class c_1).
3. If his age is less than 75 and he is not overweight, he has no high heart attack risk (class c_2).

These rules at first don't have an apparent resemblance to a tree. But the criterions themselves do have a hierarchical structure. Figure 2.1 shows the same criterions, arranged as a binary tree. The inner nodes of the tree (all nodes with outgoing directed edges) contain one of the criterions. The leaf nodes contain a solution to the actual problem, in this case one of the two classes. For a male that has to be classified, its information about age and overweight is applied to the tree, beginning at the root node. At each inner node, the corresponding criterion is assessed. If the criterion is fulfilled the left edge is traversed, otherwise the right one. When arriving at a leaf node, the information stored at the leaf is used as the classification result. While the criterions are represented by the inner nodes of the tree, the rules from above are the possible paths from the root node to a leaf node.

In this example, the decision tree is of binary form. But this is no general restriction for decision trees. Another attribute of interest in a medical-related problem might be the blood group of a person. Since more than two blood groups exist, a simple "yes or no"-criterion as in the above example is inappropriate. Instead, a criterion with more than two possibilities is needed here. This can be modeled with a tree node with four outgoing edges representing the four basic blood groups.

2.1.2 Decision Trees as a Machine Learning Technique

So far, the principle of decision trees has been described. While their usage is quite simple, the construction is the more complex part. It consists of two steps:

1. Deciding what criterion is applied at which inner node.
2. Determining what results the different leaf nodes represent.

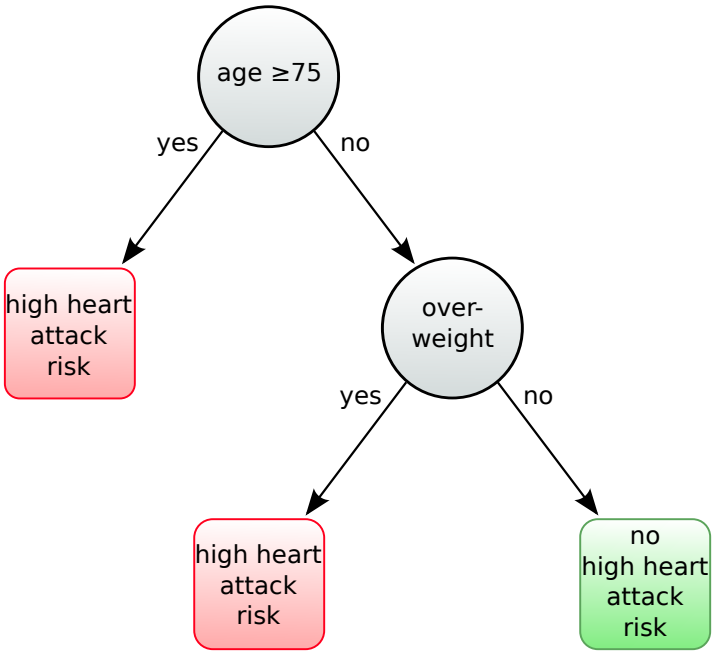


Figure 2.1: A simple decision tree to determine if a male has a high risk to suffer from a heart attack. The result is obtained by following the path from the root node to a leaf, based on the criterions at each inner node.

One possibility is the tree creation by hand, probably by experts on the field where the tree is intended to be applied. In the example above, medical scientists could have used their expertise to create a more sophisticated set of criterions. This way, the tree would probably produce more accurate results. For simple problems where the result is only related to a small number of properties, such an approach might be feasible. In more complex scenarios, the input data for such a tree might be of much higher dimensionality compared to the two dimensions (properties) used in the heart attack example in Section 2.1.1. Considering the pixel-wise information from images it is hard to tell, which pixels contribute in what way to a possible classification result for the content of the image.

This leads to the approach to use computer-based learning techniques in order to generate decision trees. While these trees are not constructed by hand, some form of experience and knowledge is still needed. This time in the form of training data. In general, the input for a decision tree can be seen as a set of p properties, combined into a data point in vector form $\mathbf{x} := (x_1, x_2, \ldots, x_p)^T$. Each of the properties x_k is part of an input space $\mathbb{I}$ and therefore $\mathbf{x} \in \mathbb{I}^p$. In the heart attack example, a data point representing a male could have the form $\mathbf{x} = (55, 1)^T$, with the age as the first entry and the second entry being used as a boolean value denoting "overweight". For the tree construction, a set of such data points $\mathcal{S} := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is used,

denoted as the training data. Additionally, for each data point $\mathbf{x}_i$ the true value $\mathbf{y}_i$ is needed that the tree will later be used to determine. This is typically referred to as the annotation or label assigned to $\mathbf{x}_i$. In a classification scenario it is the class a data point belongs to or a continuous value in a regression setting. If this label information is available, the training process is called supervised, otherwise unsupervised [CSK12, p. 92]. The training (or learning) process itself incorporates a refinement from the root node to the leaves similar to the usage of a tree described in Section 2.1.1. It is shown for the construction of binary trees but can be adjusted for non-binary ones as well.

Figure 2.2 shows a graphical representation of the process. In the beginning, the tree contains only the root node. Using a continuous numbering of the tree nodes, the root node is referred to as node 0. The whole set $\mathcal{S}$ of training data starts at the root node and is therefore named $\mathcal{S}_0$. For this node, a criterion is chosen that splits $\mathcal{S}_0$ into a partition with subsets $\mathcal{S}_0^L$ and $\mathcal{S}_0^R$ (using the notation from [CSK12]). The chosen criterion should lead to an optimal split with respect to a measure of quality for the subsets. The type of measure depends on the tree model that is used and/or the task the tree will be used for. In the case of the unified model presented in this work, this measure will be discussed in Section 2.2.2. The criterion itself is often called a *split function* [CSK12, p. 91]. With the two subsets obtained, two child nodes are created. Subset $\mathcal{S}_1 \coloneqq \mathcal{S}_0^L$ is then applied to the left child (node 1), repeating the process. The same is done for $\mathcal{S}_2 \coloneqq \mathcal{S}_0^R$ and the right child (node 2) respectively. This way, the tree grows deeper while the subsets become smaller. When a subset $\mathcal{S}_j$ with $|\mathcal{S}_j| = 1$ arrives at a node, it can not be partitioned any more and the node becomes a leaf node. That way, the tree becomes as deep as possible. Other rules for turning a node into a leaf node are discussed in Section 2.2.4. From the arriving set $\mathcal{S}_j$, the result that this particular leaf node will represent is derived. In what way depends again on the purpose the tree will be used for. The method used in the unified model will be presented in Section 2.2.5. As soon as all data points have arrived at a leaf node, the tree construction is finished.

2.2 Randomized Decision Forests

In the previous section, a generic description for decision tree construction was presented. Many details have been left out intentionally. This includes the exact form of the split function, the applied criterion to specify optimal splits and the determination of the result a specific leaf node represents. How these parts of the tree training process are defined in the unified model will be subject in the following sections. Additionally, with the tree training specified, the model of decision forests as a combination of individual decision trees is described.

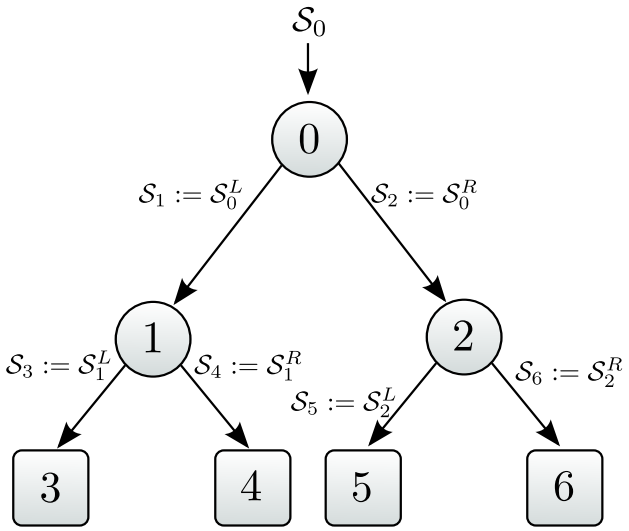


Figure 2.2: The scheme for decision tree creation. Inner nodes are represented by circles, leaf nodes by squares. The training data is partitioned into increasingly smaller subsets by optimizing and applying split functions. Each inner tree node represents a split function. The nodes are numbered based on their depth in the tree. The tree growth stops for unsplitable subsets of size 1.

2.2.1 The Split Function

So far, the split function applied at an inner tree node represented a criterion related to the properties of an instance that the decision tree is used on. Since the properties were written as a vector $\mathbf{x} := (x_1, x_2, \ldots, x_p)^T$, a split function $h(\cdot)$ is of the form

$$h(\mathbf{x}) : \mathbb{I}^p \rightarrow \mathbb{N}, \tag{2.1}$$

and its result denotes the edge of the node to proceed on. In the unified model, only binary trees are used. Additionally, another parameter θ is added to the function definition. Therefore, the split function is refined as

$$h(\mathbf{x}, \theta) : \mathbb{I}^p \times \mathcal{T} \rightarrow \{0, 1\}, \tag{2.2}$$

where the binary result denotes the left or right arc to proceed on. $\mathcal{T}$ is specified as a space for parameter triples $\theta := (\phi, \boldsymbol{\psi}, \boldsymbol{\tau})$. In short, ϕ is a function to select a subset of the properties from $\mathbf{x}$, $\boldsymbol{\psi}$ contains weights for the selected properties and $\boldsymbol{\tau} := (\tau_1, \tau_2)$ is a tuple of thresholds used for the binary decision in h. A detailed discussion of these three parameters is needed to understand their usage inside the split function.

Property Selection The parameter ϕ is a function of the form

$$\phi(\mathbf{x}) : \mathbb{I}^p \rightarrow \mathbb{I}^{p'}, \tag{2.3}$$

with $p' \leq p$. It acts as a selection of p' dimensions from the original p contained in one data point $\mathbf{x}$. Therefore, it determines which properties from $\mathbf{x}$ are of interest at this specific tree node and its split function. In [CSK12, p. 91] it is stated that typically $p' \ll p$, with p' usually being as small as 1 or 2. This coincides with the basic tree creation in [BFSO84, p. 29], where only one property is used in each split function. In both [BFSO84, CSK12] the possibility to add additional features to the initial property set of $\mathbf{x}$ is mentioned. Considering $\mathbf{x}$ again as representing the pixel-wise information of an image, the responses of filters applied at different image regions could be added to $\mathbf{x}$. That way, the initial p can grow even larger or become unbounded. Selecting a small number of properties enables to generate these features only when they are needed.

Feature Calculation After the properties of interest have been selected, a single feature is calculated from their values. Using this feature and the threshold discussed next, it is determined if an instance $\mathbf{x}$ traverses the left or right edge of the node during tree application and at the same time how the set of data points is split during tree training. These features are obtained using geometric surfaces. The following examples are based on [CSK12, p. 95f.]. Considering the case where $\mathbf{x} \in \mathbb{I}^p$ and the selection $\phi(\mathbf{x}) := (x_1, x_2, 1)$ is used, $\phi(\mathbf{x})$ is a point in $\mathbb{I}^2$ in homogenous coordinates. Defining $\boldsymbol{\psi} := (\psi_1, \psi_2, \psi_3)$ as a vector in $\mathbb{I}^3$, it represents at the same time a single line in $\mathbb{I}^2$ in homogenous form. In this form, $\phi(\mathbf{x})\boldsymbol{\psi}^T \in \mathbb{I}$ can be used as a feature representing the distance between $\phi(\mathbf{x})$ and the line $\boldsymbol{\psi}$. In Figure 2.3 (b) an example is given, where a 2D-line is used for feature calculation. It splits the projected input space into two parts. A specialization of this case are axis-aligned splits. This is accomplished by using for example $\boldsymbol{\psi} := (1, 0, 0)$. That way, only one of the selected dimensions (x_1) is used for the feature calculation. In Figure 2.3 (a) the geometric representation of this case can be seen. Since the line is parallel to the x_1-axis, only x_1 contributes to the distance between a data point and the line. In general, if $\phi(\mathbf{x})$ is a homogenous point in $\mathbb{I}^{p'}$ with $p' > 2$, the same feature calculation can be accomplished using $\boldsymbol{\psi} \in \mathbb{I}^{p'+1}$ as a hyperplane in $\mathbb{I}^{p'}$ in homogenous coordinates. In all these cases the feature is calculated as a linear combination of the selected dimensions in $\phi(\mathbf{x})$, weighted by $\boldsymbol{\psi}$. Another possibility is to use more complex surfaces for a non-linear feature calculation or in fact any function of arbitrary complexity with a scalar output. $\phi(\mathbf{x}) := (x_1, x_2, 1)$ is again considered as a point in $\mathbb{I}^2$ in homogenous form. A quadratic feature based on the selected dimensions could be of the form

$$\psi_1 x_1^2 + \psi_2 x_2^2 + \psi_3 + \psi_4 x_1 x_2 + \psi_5 x_2 + \psi_6 x_1 \quad = \quad \phi(\mathbf{x})\,\boldsymbol{\psi}\phi(\mathbf{x})^T, \tag{2.4}$$

with the symmetric matrix

$$\boldsymbol{\psi} \quad := \quad \begin{pmatrix} \psi_1 & \psi_4 & \psi_6 \\ \psi_4 & \psi_2 & \psi_5 \\ \psi_6 & \psi_5 & \psi_3 \end{pmatrix}. \tag{2.5}$$

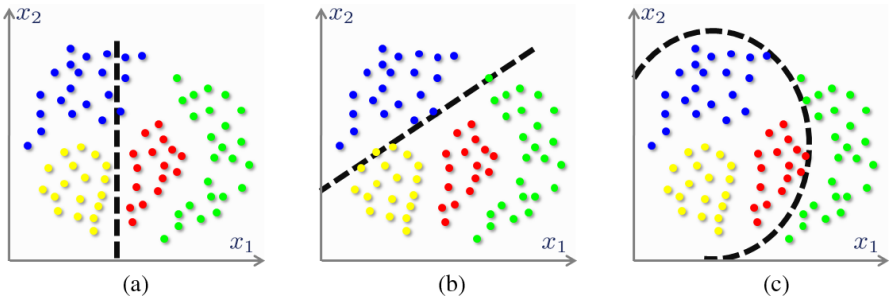


Figure 2.3: Three examples for geometric primitives used in the split function: (a) Axis-aligned line (hyperplane in higher dimensions). (b) General line (hyperplane in higher dimensions). (c) Conic section, defined by a quadratic surface in $\mathbb{I}^3$ [CSK12, p. 95].

The geometric representation of $\boldsymbol{\psi}$ is a quadratic surface in $\mathbb{I}^3$ [Pre07, p. 84]. This corresponds to the homogenous coordinates of a conic section in $\mathbb{I}^2$ [CSK12, p. 96]. Figure 2.3 (c) shows an example for such a conic section. The calculated feature represents the distance between a data point an the curve defined by the conic section.

Thresholds With the properties (dimensions) selected and a single feature calculated, the last parameter applied is $\boldsymbol{\tau} = (\tau_1, \tau_2)$, a tuple of threshold for a binary split decision. Let the feature calculated on a data point $\mathbf{x}$ be $g(\phi(\mathbf{x}), \boldsymbol{\psi})$. If $g(\phi(\mathbf{x}), \boldsymbol{\psi})$ lies within the interval (τ_2, τ_1), it is send down the left outgoing edge of the node, otherwise the right edge. Often, either $\tau_1 = \infty$ or $\tau_2 = -\infty$ is used to obtain a one-sided split.

Since the parameter set $\theta := (\phi, \boldsymbol{\psi}, \boldsymbol{\tau})$ has been discussed, the split function now remains defined as

$$h(\mathbf{x}, \theta) \quad := \quad I(\tau_1 > \phi(\mathbf{x})\,\boldsymbol{\psi} > \tau_2), \tag{2.6}$$

if a linear feature calculation is used. Here, $I(\cdot)$ is the indicator function that returns 1 or 0 depending on whether the contained predicate is true or false. In case of the quadratic surface used for feature calculation, the split function slightly changes to

$$h(\mathbf{x}, \theta) \quad := \quad I\left(\tau_1 > \phi(\mathbf{x})\,\boldsymbol{\psi}\phi(\mathbf{x})^T > \tau_2\right). \tag{2.7}$$

For each inner node j the split function h_j induces a split on the arriving set of sample $\mathcal{S}_j$ during the training process. Based on the parameter θ, this partition is defined as

$$\begin{aligned} \mathcal{S}_j^L &:= \{\mathbf{x}_i \in \mathcal{S}_j \mid h_j(\mathbf{x}_i, \theta) = 1\}, \\ \mathcal{S}_j^R &:= \{\mathbf{x}_i \in \mathcal{S}_j \mid h_j(\mathbf{x}_i, \theta) = 0\}. \end{aligned}$$

Using the definition of the split function, the optimization criterion used in the unified model is presented next.

2.2.2 Optimization Criterion

During tree training, split functions are applied to the data points of the training set to create increasingly smaller subsets. That way, the input space is partitioned and the initial problem as a whole reduced to simpler problems. While the form of the split functions is defined, their parameters need to be determined in the training process. In order to find an optimal parameter set for a split function and the corresponding set of data points, some form of measure of quality for the subsets produced by the split function is needed. This section presents a measure of quality based on information theory and formally describes the optimization process.

2.2.2.1 Entropy and Information Gain

In the unified model for decision trees and forests, the quality of a split function at node j is assessed by comparing the initial set of data points S_j and the split subsets S_j^L and S_j^R with respect to their contained information. The difference in the amount of contained information is called *Information Gain* [CSK12, p. 96]. An informal interpretation of this might be the amount of information obtained when it becomes known for a data point $\mathbf{x}_k \in S_j$, to which of the subsets S_j^L and S_j^R it belongs to. But before specifying the information gain formally, a measure for information itself is needed.

The measure used is called *Entropy*. The exact definition of the entropy depends on the type of annotation. At first, the case is considered where the annotation $\mathbf{y}_i$ assigned to each $\mathbf{x}_i \in \mathcal{S}_j$ is from a finite set $\mathcal{Y}$. In order to calculate the entropy for $\mathcal{S}_j$, it is necessary to derive a discrete probability distribution $p(\mathbf{y})$ from it. This probability distribution describes the likelihood of all possible annotations in $\mathcal{Y}$, based on the labeled data points in $\mathcal{S}_j$. A possibility is to state $p(\mathbf{y})$ as the proportion of data points $\mathbf{x}_i \in \mathcal{S}_j$ with annotation $\mathbf{y}_i = \mathbf{y}$. In this discrete case, the entropy $H(\mathcal{S}_j)$ is defined as

$$H(\mathcal{S}_j) \quad := \quad -\sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}) \log\left(p(\mathbf{y})\right). \tag{2.8}$$

This form of entropy is called the *Shannon Entropy* [CSK12, p. 98]. Typically, it is used with a base two logarithm [Mit97, p. 57]. In this case, the Shannon entropy is not only a measure of information but rather a measure of the minimal number of bits that are needed to encode the information. Considering a binary classification scenario, a class label c_i is assigned to each data point $\mathbf{x}_i \in \mathcal{S}_j$. If the information about the class label of a randomly drawn $\mathbf{x}_k \in \mathcal{S}_j$ has to be sent

in an bitwise encoded message, the Shannon entropy of $\mathcal{S}_j$ is the minimal number of bits needed for the encoding. For example, if it is known (to the receiver of the message) that all $\mathbf{x}_i \in \mathcal{S}_j$ belong to class c_1 then it is clear what class $\mathbf{x}_k$ will belong to $(p(c_1) = 1, p(c_2) = 0)$. Therefore, no message has to be sent at all and the number of bits needed is 0. In the case that half of the data points in $\mathcal{S}_j$ belong to class c_1 and the other half to c_2, no assumption about the class membership of $\mathbf{x}_k$ can be made $(p(c_1) = 0.5, p(c_2) = 0.5)$. Thus, a single bit is needed to encode it [Mit97, p. 57].

If the annotation is of continuous nature (e.g. $\mathbf{y} \in \mathbb{R}$) then a continuous probability distribution $p(\mathbf{y})$ must be derived from set $\mathcal{S}_j$ to use the differential form of the Shannon entropy [CSK12, p. 99]. It is defined as

$$H(\mathcal{S}_j) := -\int_{\mathbf{y}\in\mathbb{R}} p(\mathbf{y}) \log(p(\mathbf{y}))\, d\mathbf{y}. \tag{2.9}$$

In both discrete and continuous cases, the method to obtain the probability distribution depends on the specific task the decision tree is used for. In the context of regression problems, this will be discussed in Section 4.2. Applying a split function $h_j(\mathbf{x}, \theta)$ onto all $\mathbf{x}_i \in \mathcal{S}_j$ results into two subsets S_j^L and S_j^R. For this split, the information gain G is then defined as

$$G(\mathcal{S}_j, \theta) = H(\mathcal{S}_j) - \left(\frac{|\mathcal{S}_j^L|}{|\mathcal{S}_j|} H\left(\mathcal{S}_j^L\right) + \frac{|\mathcal{S}_j^R|}{|\mathcal{S}_j|} H\left(\mathcal{S}_j^R\right) \right), \tag{2.10}$$

independent of the form of entropy used [CSK12, p. 99].

2.2.2.2 Optimization of the Split Function Parameters

With the information gain as the optimization criterion, the optimization process at each inner tree can now be summarized: For the set $\mathcal{S}_j$ of data points arriving at node j, an optimal parameter triple θ_j^* is needed such that

$$\theta_j^* = \arg\max_{\theta\in\mathcal{T}} G(\mathcal{S}_j, \theta). \tag{2.11}$$

It is important to note that $\mathcal{T}$, the set of all possible parameter combinations for the split function, can be either extremely large or even infinite [CSK12, p. 101]. How this optimization can kept feasible is described in the next section.

2.2.3 Randomized Tree Creation

Until now, the tree training process is still deterministic. Therefore, two trees trained on the same data would be identical. Since the overall purpose is to combine multiple trees to a decision forest, the trees need to differ from each other.

Combining identical trees to a forest would not provide better results than using only one of them. In order to obtain different trees from the same data, some form of randomization has to be added to the training process. Mainly, there are two different methods to do so [CSK12, p. 100]:

1. Randomly select a subset of the total training data for each tree trained.
2. Optimize each split function using only a subset of all possible parameters.

The first method, usually referred to as *Bagging*, is used for the decision forests proposed in [Bre01]. Here, for each tree a subset of the initial training data is drawn with replacement and used for the training process. In the unified model, only the second method is used [CSK12, p. 100]. For each inner node j of a tree, a subset of all possible split parameters $\mathcal{T}_j \subset \mathcal{T}$ is drawn randomly. Then, the optimal parameter triple θ_j^* complying with

$$\theta_j^* = \arg\max_{\theta \in \mathcal{T}_j} G(\mathcal{S}_j, \theta) \tag{2.12}$$

is searched for. The only difference to Equation 2.11 is the smaller parameter space. By choosing $\mathcal{T}_j$ small enough, it is possible to obtain the optimal parameter set through an exhaustive search over $\mathcal{T}_j$. That way, randomized trees can be created while keeping the optimization process efficient.

2.2.4 Stopping Criteria for the Tree Growth

Using the randomized splitting process described above, the initial data is split into more and more subsets while additional nodes are added to the tree. As described in Section 2.1.2, this can be continued until each data point from the initial set is a subset on its own. Tree nodes where such one-element subsets arrive are then turned into leaf nodes. How a leaf node can derive its prediction from the arriving subset will be subject of the next section. Nevertheless, this maximum tree growth strategy can already be criticized at this point. Supposing one of the data points in the training set is wrongly annotated, e.g. a wrong class label is assigned to it. During the tree training, a leaf node will be generated with this sample as the arriving subset. Thus, the prediction of the leaf is solely based on this single data point. Because of the wrong annotation, the prediction will be wrong, too.

This simple example motivates other strategies, where the tree is not grown to its maximum size. In [CSK12, p. 94], several stopping criteria for the tree growth are presented:

1. Defining a minimal information gain.
2. Defining a maximal tree depth.
3. Defining a minimal number of data points arriving at a leaf node.

Every time a split function at a node would lead to a minor information gain, a deeper tree or a child node with less data points than the defined limit, the node is turned into a leaf node. Because these parameters are not part of the optimization process, they have to be selected by hand. Finding optimal values for these limits depends on the application and the size and quality of the training data.

A popular approach for tree size optimization during training is *Tree Pruning*. It was presented in [BFSO84], before the use of tree ensembles emerged. Prior to actually optimizing the tree size, the tree is grown to maximum size with the same one-data-point-per-leaf stopping criterium presented in Section 2.1.2. The pruning process starts afterwards. From the leaves upward, sub-trees are evaluated on a set of data distinct from the initial training data. If the splitting inside the sub-tree does not lead to an increase in prediction performance compared to a single leaf node created with the same training data, the sub-tree is "pruned" and replaced by a leaf node. That way, the overfitting of a full sized tree to the training data is avoided. Due to the change from single decision trees to tree ensembles, both in the presented unified model and the decision forests proposed by Breiman, tree pruning is neglected in favor of a simpler training process for the individual trees. [Bre01, CSK12].

2.2.5 Leaves as Predictors

The last step in the decision tree training is the construction of the leaf nodes. The leaves are responsible for the actual prediction performed by the tree. Using the arriving subset of training data $\mathcal{S}_j$, statistics have to be derived from it to make predictions on future data points $\mathbf{x}$ that will arrive at this leaf during tree application. In the unified forest model, this prediction takes the form of a probability distribution $p(\mathbf{y})$. It states the likelihood of all the possible predictions $\mathbf{y} \in \mathcal{Y}$. This is similar to the entropy calculation presented in Section 2.2.2.1, where such distributions have to be derived, too. The difference here is that the specific data point $\mathbf{x}$ can also be incorporated into this probability distribution. Instead of presenting a general distribution $p(\mathbf{y})$ solely based on the training data $\mathcal{S}_j$, a distribution $p(\mathbf{y} \mid \mathbf{x})$ additionally dependent on the arriving data point could be returned. That way, different data points arriving at the same leaf can still lead to different predictions. In both cases, the nature of the probability distribution (discrete or continuous) depends on whether the annotation space $\mathcal{Y}$ is finite or not. How such a distribution can be derived is again dependent on the specific task the tree is used for. An example for $p(\mathbf{y})$ in the finite case has been presented in Section 2.2.2.1. Since the overall interest in this work is in trees and forests for regression, a method to derive a continuous probability distribution in that case will be presented in Section 4.2.

Instead of returning a whole distribution, it is also possible to derive a single prediction from it. The maximum likelihood prediction $\mathbf{y}^*$ given $\mathbf{x}$, in both the discrete

and continuous case is

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}).$$

Nevertheless, the additional information from the probability distribution can be used as a confidence measure for predictions obtained from this leaf [CSK12, p. 100]. An example will be shown in Section 4.2.2.

2.2.6 From Trees to Forests

So far, the general training process for decision trees has been explained. A single tree can already be used for a prediction based on a new sample $\mathbf{x}$. Depending on the depth of the tree, these predictions can suffer from two problems: in the case of shallow trees, the few leaf nodes base their predictions on rather large and impure subsets of the training data. This can lead to predictions not discriminative enough for different input data the tree is applied to. Very deep trees, on the other hand, tend to overfit the initial training data since each leaf node only has a very small subset of data to derive its prediction from. That way, input data dissimilar to the data used for training can lead to very imprecise predictions. To counter these problems, ensembles of randomized trees are used. Denoted as random decision forests or simply random forests, they are collections of trees produced by the tree training process described above. Therefore, training a forest of size m with a set of training data incorporates the training of m individual trees t_l on this data. Due to the randomization, these trees differ from each other. The smaller the subsets of splitting parameters $\mathcal{T}_j$ assessed at each tree node j, the more different the individual trees are.

Applying a forest to a new sample $\mathbf{x}$ involves the application of each tree t_l to $\mathbf{x}$. The result from each tree has the form $p_{t_l}(\mathbf{y} \mid \mathbf{x})$. The result of the forest can simply be defined as the mean of all tree results [CSK12, p. 102]:

$$p_{forest}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{m} \sum_{l=1}^{m} p_{t_l}(\mathbf{y} \mid \mathbf{x}).$$

This description of how decision forests are trained and used completes the discussion of the unified forest model.

3 Linear Regression

Regression problems are not restricted to decision trees or forests in any way. In fact, methods for their solution are a well known discipline in statistics on their own. This chapter introduces the concept of regression and discusses some of the related methods and techniques without the decision tree context. Nevertheless, the presented results will be used in Chapter 4 to specialize the unified decision forest model for regression tasks.

In general, regression analysis has the objective to find a relationship between a set of p independent variables $(x_1, x_2, \ldots, x_p) =: \mathbf{x}^T$ (also known as regressors) and a single dependent (or response) variable y. The analysis is based on data that exemplary represents this relationship. The data consists of n observations in form of tuples $(x_{i1}, x_{i2}, \ldots, x_{ip}, y_i)$. The goal is to obtain a single model $f(\mathbf{x})$ that describes the relationship for all observations in the form

$$y_i \quad = \quad f(\mathbf{x}_i) + \varepsilon_i, \forall i = 1 \ldots n, \tag{3.1}$$

up to an error term ε . Finding such a model has two major benefits:

1. It reveals how the response variable is related to the regressor.
2. If for a set of regressors the response variable is unknown, a prediction for the response can be obtained from the model.

The quality of the model depends not only on the quality of the gathered data, from which it gets derived. It also depends on how well the form and complexity of the model can describe the true underlying process that has generated the data. In the following sections, different types of regression models will be introduced. And with the focus on linear models, methods of fitting such models onto the data and assessing their quality will be discussed.

3.1 Regression Types

Before beginning an in-depth discussion of linear regression, different types of regression models are presented. Additionally, some of the most frequent terms related to these models are introduced.

3.1.1 Linear Regression Models

Linear regression seeks to model the response variable as a linear combination of the regressors. In its simplest form, the regression is performed on one regressor only ($\mathbf{x} \coloneqq x_1$), the so called *Simple Linear Regression*. In this case the model has the following form:

$$y = \beta_0 + \beta_1 x_1 + \varepsilon. \tag{3.2}$$

Here, β_0 and β_1 are the parameters of the model and ε represents a random error. With β_0 seen as the intersect and β_1 as the slope, this model resembles a simple line equation. This explains why linear regression is often referred to as *Line Fitting*. The term for the random error ε is needed, because data rarely fits perfectly onto a linear model, especially if the data is gathered by error-prone measurements. It will be further discussed in Section 3.4.

When there is more than one regressor, *Multiple Linear Regression* is applied with its general form

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \varepsilon = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \varepsilon. \tag{3.3}$$

Again, $\beta_0, \beta_1, \ldots, \beta_p$ are the model parameters and ε the random error. It is important to note that the linear model allows a regressor to be a higher order term in other regressors [YS09, p. 48]. For example, the following model is still defined as linear:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \varepsilon. \tag{3.4}$$

From now on, the term *Linear Regression* is used instead of *Multiple Linear Regression*, as it includes *Simple Linear Regression* as a specialization.

3.1.2 Non-linear Regression Models

In contrast to the linear case, non-linear regression models the response variable with more than just linear combinations of regressor variables. An example is the use of a polynomial function, with its degree being parameterized:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \cdots + \beta_\gamma x_1^\gamma + \varepsilon. \tag{3.5}$$

While such models have more possibilities to explain the relationship between the variables, they are also far more complicated to handle. As in the current topic of regression forests only linear models will be used, the non-linear approach is not discussed any further.

3.2 Matrix Notation

Because the following discussion of linear regression will contain many mathematical formulations, it is convenient to introduce a vector and matrix notation for the related variables. This way the presented equations can be kept as simple as possible. As it has been introduced in Section 3.1.1, the linear regression model contains a constant factor β_0, the so called bias [HTF03, p. 11]. Therefore, it is advantageous to add an additional constant regressor to the existing ones. By defining $\mathbf{x}_i := (1, x_{i1}, x_{i2}, \ldots, x_{ip})^T$, the regression model for each of the n observations can be written as

$$y_i = \mathbf{x}_i^T \boldsymbol{\beta} + \varepsilon_i, \forall i = 1 \ldots n \tag{3.6}$$

with

$$\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)^T . \tag{3.7}$$

Using matrix notation, all n observations can be combined in the single equation

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{3.8}$$

with

$$\mathbf{X} := \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}, \mathbf{y} := \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \boldsymbol{\varepsilon} := \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} . \tag{3.9}$$

The definitions in Equation 3.7 and Equation 3.9 will be used frequently in the following sections as well as in the Appendix.

3.3 Least Squares Estimation

The goal is now to find an estimation of the model parameters $(\beta_0, \beta_1, \ldots, \beta_p)$, that describes the relationship between y_i and $\mathbf{x}_i$ for all n observations as good as possible. What "good" actually means, depends on the criterion that will be optimized. Probably the best known approach is the *Least Squares Estimation* which goes back to Legendre and Gauss at the beginning of the 19th century [YS09, p. 2]. Here, the optimization criterion is the so called residual sum of squares (RSS). For each observation, the residual is defined as the difference $e_i := y_i - \hat{y}_i$ between

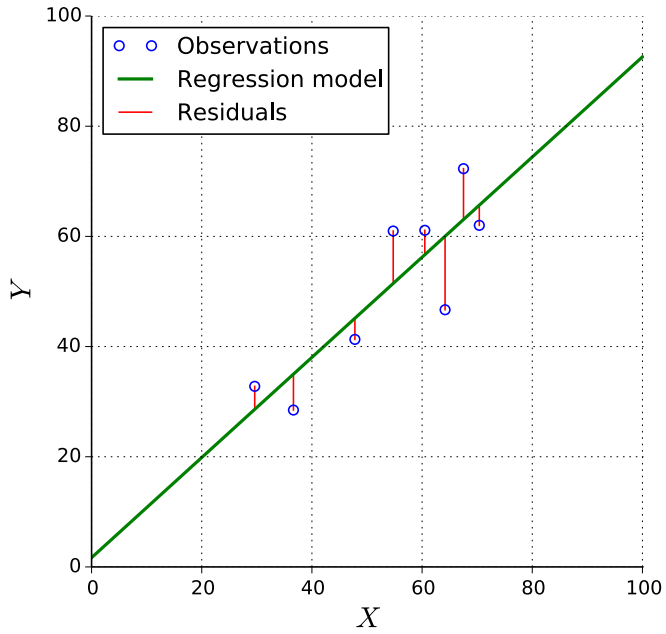


Figure 3.1: A set of observations in form of 2D points. In this case, the regression model resembles a 2D line with intersect and slope. The distances in y-direction from each point to the line are the residuals of the model.

the actual observed response value y_i and the prediction of the regression model $\hat{y}_i = \mathbf{x}_i^T \boldsymbol{\beta}$. The residual sum of squares then has the following form:

$$\begin{aligned} RSS(\boldsymbol{\beta}) &:= \sum_{i=1}^{n} e_i^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^T \boldsymbol{\beta}\right)^2 . \end{aligned} \tag{3.10}$$

Figure 3.1 shows an example with several 2D points acting as observations and a 2D line as a linear regression model. For each point, the residual is displayed which resembles the distance between the point and the line in Y-direction. The residual sum of squares is a measurement of how far the observations and the predictions differ from each other and acts as an assessment of how good the model fits to the data. Therefore, the goal is to find a a set of model parameters that produces the smallest residual sum of squares possible. The linear model shown in the figure fulfills this condition.

More formally, parameters $\mathbf{b} := (b_0, b_1, \ldots, b_p)^T$ are looked for with

$$\mathbf{b} = \arg\min_{\boldsymbol{\beta}} RSS(\boldsymbol{\beta}) . \tag{3.11}$$

In order to minimize Equation 3.10, it is differentiated with respect to $\boldsymbol{\beta}$ (see Section A.1):

$$\frac{\delta RSS\left(\boldsymbol{\beta}\right)}{\delta \boldsymbol{\beta}} = -2\mathbf{X}^T\mathbf{y} + 2\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta} \tag{3.12}$$

By setting Equation 3.12 equal to zero, it follows that

$$\begin{aligned} 0 &= -2\mathbf{X}^T\mathbf{y} + 2\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta}, \\ \left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta} &= \mathbf{X}^T\mathbf{y}. \end{aligned} \tag{3.13}$$

If the matrix $\left(\mathbf{X}^T\mathbf{X}\right)$ is non-singular (and therefore an inverse exists), the solution for the model parameters $\mathbf{b}$ is given by

$$\mathbf{b} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{3.14}$$

In the case of $\left(\mathbf{X}^T\mathbf{X}\right)$ being singular (due to the fact, that $\mathbf{X}$ is not of full rank), it is still possible to get a solution using the generalized inverse. But the solution obtained from this approach is neither unique nor guaranteed to be optimal. Details can be found in [SS90, p. 30]. The meaning of $\mathbf{X}$ being not of full rank is that at least one of the regressors in the model is a linear combination of some of the other regressors. The independence of the regressors is violated. In this case, the data is usually called ill-conditioned [YS09, p. 82]. In an exemplary regression model with three regressors (x_1, x_2, x_3), a dependence of the form

$$x_3 = 2x_1 - x_2 \tag{3.15}$$

in all observations would lead to such an ill-conditioned case. Therefore, the selection of regressors used in the regression model is an important step prior to estimating the model parameters. How such cases can be handled in regression trees and forests will be discussed in Section 5.1.

3.4 Regression Model Diagnosis

So far, a set of model parameters was obtained that optimally fits to the initial data. While the residual sum of squares has already been used to evaluate the quality of the regression model, it only contains information about the model as a whole. By deriving some statistical properties of the obtained model parameters, it allows in the end not only to evaluate the regression model itself, but also to provide an individual confidence for every prediction based on the model. Since the next part heavily relies on the concepts of error and residual, they will be discussed first.

3.4.1 Errors and Residuals

At the beginning, the assumption was made that the response variable is generated by an underlying model of the form

$$y = f(\mathbf{x}) + \varepsilon, \tag{3.16}$$

while f is a linear function in $\mathbf{x}$ with $f(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta}$. The additive random error ε is needed, since it is likely that the relationship $y = f(\mathbf{x})$ will not exactly reproduce all observations $(\mathbf{x}, y)$. One reason for this might be that y depends on other entities besides $\mathbf{x}$ that are not considered in the model. Measurement errors included in the process of obtaining observations have a similar effect [HTF03, p. 28]. So far, only f has been discussed and its parameters have been estimated. Now, an estimation of ε follows.

The error is the difference between the actual observed value y and the expected value $f(\mathbf{x})$. f itself is dependent on all possible observations $(\mathbf{x}, y)$, in statistical terms the whole "population". Since these are typically not available, f is unknown. This leads to the fact that, while y is observable, ε is not. Instead, n observations were drawn randomly and the parameter set $\mathbf{b}$ as an estimation for $\boldsymbol{\beta}$ was derived from them. This led to the definition of the residual e with

$$y = \mathbf{x}^T\mathbf{b} + e. \tag{3.17}$$

Thus, the residual is the difference between the observed y and the output of the estimated model, and therefore is observable. This enables the estimation of the error based on the residuals [YS09, p. 12].

3.4.2 Gauss-Markov Conditions

Until now, no further assumptions on the properties of the error ε were made, except that it is of random nature. But to obtain the desired statistical properties from the parameter estimation $\mathbf{b}$, the assumption is necessary that ε fulfills three conditions, the so called Gauss-Markov conditions [SS90, p. 35]. For each of the observations $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, the conditions for the corresponding error ε_i are

$$E(\varepsilon_i) = 0, \tag{3.18}$$

$$E\left(\varepsilon_i^2\right) = \sigma^2, \tag{3.19}$$

and

$$E(\varepsilon_i\varepsilon_j) = 0, \text{ if } i \neq j, \tag{3.20}$$

where $E(v)$ denotes the expectation of a random variable v. It is important to note that, while the expectation for all ε_i^2 is the same, σ^2 is unknown and will be subject in the following section.

3.4.3 Error Variance Estimation

In order to evaluate the predictions of the regression model, the relationship between the error and the predictions will be used in Section 3.4.4. But before this can be done, an estimation of the error itself is needed. While the assumption of the Gauss-Marokov conditions enables the derivation of some further error properties (see Section A.2), the error variance $\text{var}(\varepsilon_i)=\sigma^2$ is still unknown. Because this variance is, like the error itself, unobservable, an estimation is necessary. If it is possible to find a s^2 with $\exp(s^2) = \sigma^2$, then s^2 is an unbiased estimator of σ^2 [SS90, p. 284]. In Section 3.4.1 an error estimation using the residuals of the regression model has already been motivated. More specifically, the residual sum of squares and its expected value will again be used.

In Section A.3 it is shown that the expectation for the residual sum of squares is

$$E(RSS(\mathbf{b})) = (n-p-1)\,\sigma^2. \tag{3.21}$$

By defining

$$s^2 := \frac{RSS(\mathbf{b})}{n-p-1}, \tag{3.22}$$

it follows that

$$\begin{aligned} E\left(s^2\right) &= E\left(\frac{RSS(\mathbf{b})}{n-p-1}\right) \\ &= \frac{E(RSS(\mathbf{b}))}{n-p-1} \\ &= \sigma^2. \end{aligned} \tag{3.23}$$

Hence, under the Gauss-Markov conditions, s^2 is an unbiased estimator for the error variance σ^2 that can be used from now on. This enables the discussion on how the error influences the estimated model parameters and the predictions.

3.4.4 Confidence Estimation

The last step of the regression diagnosis is to provide a confidence measure for the predictions the estimated regression model generates. In particular, not only single predictions but complete probability distributions will be provided to determine how likely the prediction of the response value is, given a set of input values. To achieve this, the variation of the model parameter estimation and the predictions will be derived, leading to the desired probability distributions. It is important to note that the estimated model parameters are influenced by the error, and thus the

predictions are too. While this influence is not apparent in the following equations, it can be seen in the proofs of some of them in the Appendix. Therefore, the estimation of the error and its variance from the previous section play an important role here.

At first, similar to the relationship between s^2 and σ^2, the expectation for the estimated model parameters is

$$\begin{aligned} E(\mathbf{b}) &= \exp\left(\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}\right) \\ &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\exp(\mathbf{y}) \\ &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\ &= \boldsymbol{\beta}, \end{aligned} \tag{3.24}$$

because $E(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta}$ (see Equation A.14). Thus, $\mathbf{b}$ is a true unbiased estimation of the underlying model parameters $\boldsymbol{\beta}$. This holds true, even if the latter two Gauss-Markov conditions are violated [SS90, p. 35]. In Section A.4 it is shown that

$$\operatorname{cov}(\mathbf{b}) = \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}. \tag{3.25}$$

Furthermore, given an arbitrary set of input variables $(1, x_{k1}, x_{k2}, \ldots, x_{kp}) =: \mathbf{x}_k$ and its corresponding prediction $\hat{y}_k = \mathbf{x}_k^T\mathbf{b}$, the prediction expectation and variance are shown to be

$$\exp(\hat{y}_k) = \mathbf{x}_k^T\boldsymbol{\beta}, \tag{3.26}$$

and

$$\operatorname{var}(\hat{y}_k) = \mathbf{x}_k\operatorname{cov}(\mathbf{b})\,\mathbf{x}_k^T = \sigma^2\mathbf{x}_k^T\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k. \tag{3.27}$$

The probability of a prediction value at a specific set of input values is modeled as a normal distribution in the context of the discussed regression forests [CSK12, p. 136] . Thus, the last assumption necessary is that all observation errors $\varepsilon_1, \ldots, \varepsilon_n$ and accordingly all response values $y_1, \ldots, y_n$ follow a normal distribution ($\varepsilon_i \sim N(0, \sigma^2), \forall i = 1, \ldots, n$). Therefore, any prediction $\hat{y}_k$ is normally distributed with

$$\hat{y}_k \sim N\left(\mathbf{x}_k^T\boldsymbol{\beta},\ \sigma^2\mathbf{x}_k^T\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k\right). \tag{3.28}$$

Using the estimations in Equation 3.23 and Equation 3.24, it follows that

$$\hat{y}_k \sim N\left(\mathbf{x}_k^T\mathbf{b},\ s^2\mathbf{x}_k^T\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k\right), \tag{3.29}$$

by replacing $\boldsymbol{\beta}$ and σ^2 with $\mathbf{b}$ and s^2 [SS90, p. 71f.]. Equivalently, the probability distribution for $\hat{y}_k$ given $\mathbf{x}_k$ is

$$p\left(\hat{y}_k \mid \mathbf{x}_k\right) = N\left(\mathbf{x}_k^T\mathbf{b},\ s^2\mathbf{x}_k^T\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k\right). \tag{3.30}$$

Figure 3.2 shows the same situation of 2D observations as in Figure 3.1. Apart from deriving the regression model itself, regression diagnosis enables to provide a probability distribution to each x-value that resembles how likely different prediction values are. For each pair of (x, y)-values, the figure shows the likelihood of prediction y given x in form of red saturation, using the probability density function for Equation 3.30. The darker the red color, the more likely the prediction is. Furthermore, the variance of the normal probability distribution for each x-value can directly be seen as a confidence for predictions based on x. The first thing to note is that the regression model shows uncertainty at all, since for all x-values there is a whole range of y-predictions with likelihood greater than zero. Interestingly, this uncertainty is not constant but varies for different x-values. The further away from the center of the observations, the larger the uncertainty becomes. This is due to the variability of the model parameters stated in Equation 3.25. Since the predictions on different x-values are influenced differently by varying model parameters, the uncertainty differs, too. This makes sense, since the further away an x-value is from the mean of the observations, the more likely it is that this x-value behaves differently than the ones in the region of the observations.

3.5 Multiple Outputs

So far, multiple linear regression was discussed which denotes that the regression model has several input variables but only one response variable. In general, cases with multiple response values are of interest, too. More formally, the n observations have the form

$$\left(\mathbf{x}_i^T, \mathbf{y}_i^T\right) = (x_{i1}, x_{i2}, \ldots, x_{ip}, y_{i1}, y_{i2}, \ldots, y_{iq})\ \forall i = 1, \ldots, n, \tag{3.31}$$

in the case of p (independent) input variables and q (dependent) response variables. The simplest approach is to tackle each response variable on its own and therefore convert the problem into q distinct ones [HTF03, p. 56]. Thus, for each dimension $d \in \{1, \ldots, q\}$ of the response vectors $\mathbf{y}_i$, a linear model is needed with the form

$$\begin{aligned} y_{id} &= \beta_{d0} + \beta_{d1}x_{i1} + \beta_{d2}x_{i2} + \cdots + \beta_{dp}x_{ip} + \varepsilon_{id},\ \forall i = 1, \ldots, n \\ y_{id} &= \mathbf{x}_i^T\boldsymbol{\beta}_d + \varepsilon_{id},\ \forall i = 1, \ldots, n. \end{aligned} \tag{3.32}$$

Again, matrix notation can be used to combine all q linear models into the equation

$$\mathbf{Y} = \mathbf{XB} + \mathbf{E}, \tag{3.33}$$

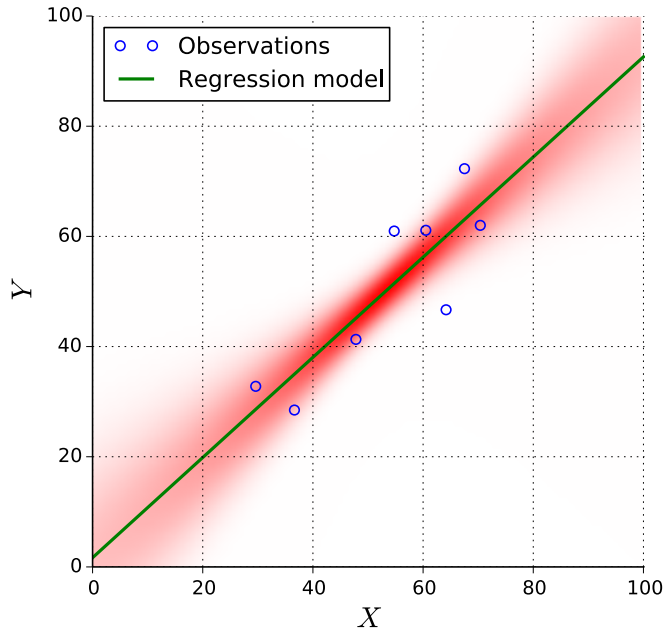


Figure 3.2: A set of observations in form of 2D points and the derived regression model. For each pair of (x, y)-values, the red saturation resembles the likelihood of a prediction y given x.

with

$$\mathbf{Y} \coloneqq \begin{pmatrix} y_{11} & \dots & y_{1d} \\ \vdots & \ddots & \vdots \\ y_{n1} & \dots & y_{nd} \end{pmatrix}, \mathbf{B} \coloneqq \begin{pmatrix} \beta_{10} & \dots & \beta_{d0} \\ \vdots & \ddots & \vdots \\ \beta_{1p} & \dots & \beta_{dp} \end{pmatrix}, \mathbf{E} \coloneqq \begin{pmatrix} \varepsilon_{11} & \dots & \varepsilon_{1d} \\ \vdots & \ddots & \vdots \\ \varepsilon_{n1} & \dots & \varepsilon_{nd} \end{pmatrix}, \quad (3.34)$$

and $\mathbf{X}$ as usual. To obtain each linear model separately, least squares estimation can be used equal to Section 3.3. Therefore, the solution for each linear model is given by

$$\mathbf{b}_d = \underbrace{\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T}_{\text{independent from } d} \bar{\mathbf{y}}_d, \ \forall d = 1, \dots, q, \quad (3.35)$$

with

$$\bar{\mathbf{y}}_d \coloneqq \begin{pmatrix} y_{1d} \\ y_{2d} \\ \vdots \\ y_{nd} \end{pmatrix}, \mathbf{b}_d \coloneqq \begin{pmatrix} b_{d0} \\ b_{d1} \\ \vdots \\ b_{dp} \end{pmatrix} \quad (3.36)$$

Thus, the solution for all linear models is just a generalization over the columns of $\mathbf{Y}$ with

$$\hat{\mathbf{B}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}, \quad (3.37)$$

and

$$\hat{\mathbf{B}} \coloneqq \begin{pmatrix} b_{10} & \dots & b_{d0} \\ \vdots & \ddots & \vdots \\ b_{1p} & \dots & b_{dp} \end{pmatrix} \tag{3.38}$$

Even though the complete solution can be presented in one equation, multiple linear regression is still performed q times to obtain q distinct regression models. Because of that the regression diagnosis presented in Section 3.4 can be applied on each of the separate models. The final result, the probability distribution in Equation 3.30, can then be summarized into one multivariate probability distribution of the form

$$p\left(\hat{\mathbf{y}}_k \mid \mathbf{x}_k\right) = N\left(\mathbf{x}_k^T \hat{\mathbf{B}},\ \Lambda_{\hat{\mathbf{y}}_k}\right), \tag{3.39}$$

with covariance matrix

$$\Lambda_{\hat{\mathbf{y}}_k} \coloneqq \begin{pmatrix} \operatorname{var}\left(\hat{y}_{k1}\right) & 0 & \dots & 0 \\ 0 & \operatorname{var}\left(\hat{y}_{k2}\right) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \operatorname{var}\left(\hat{y}_{kq}\right) \end{pmatrix}. \tag{3.40}$$

Again, $\mathbf{x}_k \coloneqq (1, x_{k1}, x_{k2}, \dots, x_{kp})^T$ is an arbitrary set of input variables and $\hat{\mathbf{y}}_k \coloneqq (y_{k1}, \dots, y_{kq}) = \mathbf{x}_k^T \hat{\mathbf{B}}$ the multivariate prediction as a combination of the q regression models.

This generalization of multiple linear regression to multiple response variables closes the discussion of linear regression.

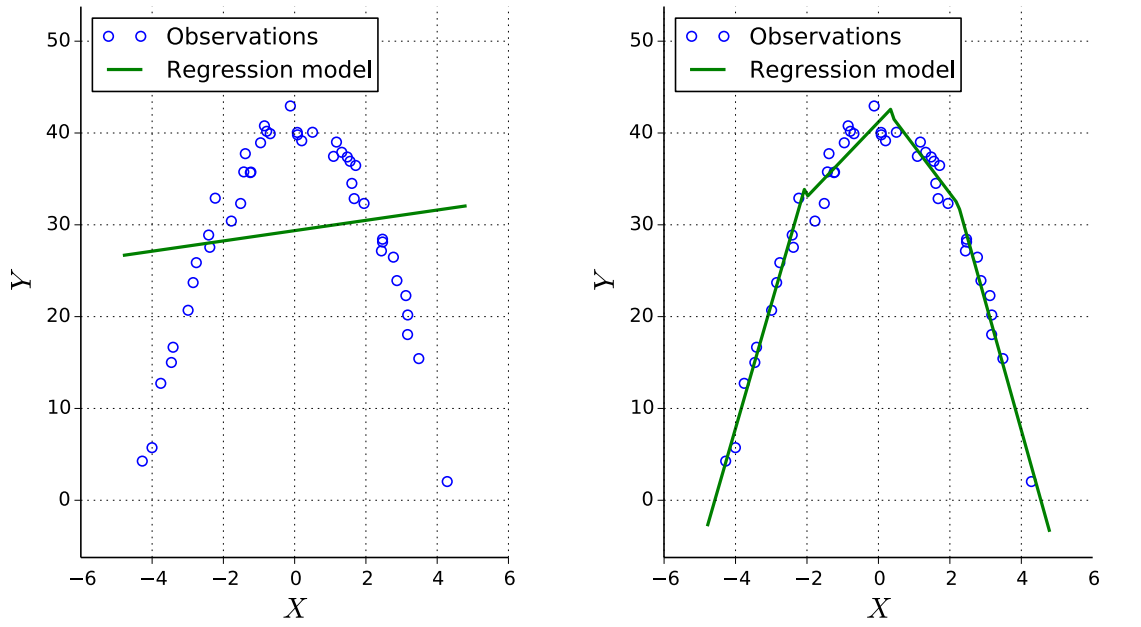
4 Regression Forests

Decision trees and forests are not limited to a specific application, but can be applied to a variety of tasks, including regression. A mathematical solution for such problems has already been presented in the form of linear regression. Nevertheless, this solution has its limitations. In Section 4.1 an example for these limitations will be presented and the usage of tree structures for regression problems will be motivated. While there is the possibility to define a customized form of decision trees and forests only intended for regression, this approach is not pursued. Instead, the unified forest model will be used to obtain decision forests for regression by refining its definition where necessary. These refinements will be discussed in Section 4.2.

4.1 Motivation

In general, linear regression as presented in Chapter 3 can be applied to many regression problems. Such a linear regression model is derived from a set of data. The main assumption of linear regression is that the true underlying process that has generated the data in the first place is of linear form, too. Only due to the perturbation of the data by random noise, no model can be obtained that perfectly reconstructs it. Therefore, as soon as this assumption is wrong, the theory of linear regression does no longer guarantee an optimal model. Despite this fact, it is possible to derive a linear regression model from data of nearly arbitrary origin and quality. But a priori, no statement about the quality of this model can be made.

Figure 4.1a shows a set of 2D data points that were generated by a polynomial function. For these data points, a linear regression model is calculated. Due to the higher complexity of a polynomial model, the linear model can not represent the true structure of the data. To resolve this, a non-linear regression model is needed. Figure 4.1b shows a second approach. Instead of using a regression model of higher complexity, multiple linear models are combined to approximate the parabolic shape of the data. This is achieved by separating the input space into small intervals and using an individual linear model on each interval. This resembles the approximation of a function of higher complexity using Taylor polynoms at different expansion points in its simplest form. In fact, the combination of all linear models leads to a non-linear behaviour. Since the initial problem was too complex to be solved directly, it was divided into smaller ones that can be handled more easily. This



(a) Single linear regression model. **(b)** Combination of linear regression models.

Figure 4.1: A set of 2D points created by a polynomial function and additional random noise. A single linear model can not describe the complexity of the data. Using multiple linear models, the actual shape of the data can be approximated.

approach closely resembles the strategy of decision trees. Thus, using decision trees and forests for regression seems to be a fruitful approach.

4.2 Specializing the Unified Forest Model for Regression

The unified forest model presented in Chapter 2 was created with the intent to use decision forests for many tasks without the necessity to define the structure and the construction process of the forests for each task anew. Thus, some of the key elements of decision forests were defined generically. The actual instantiation of a forest for a specific application now only requires to specify these elements, while the rest of the forest structure can be used unchanged. This does not only lead to a theoretical unification. The actual implementation of decision forests is simplified so that the forest structure and its construction must be implemented once and can then be reused for many applications with only slight modifications [CSK12, p. 87]. In the context of regression, these modifications concern the entropy used as the optimization criterion and the predictors stored at the leaf nodes. These

modifications will be described in the following sections. It is important to note that the generalization from decision trees to forests is not affected by these modifications in any way.

In the regression forests presented, linear regression models are used in the individual trees similar to the example in Figure 4.1b. This is no general restriction. Less complex regression models as well as models of higher complexity could be incorporated into regression forests as well. While these possibilities are not further discussed here, the usage of simpler constant regression during tree training will be subject in Section 5.2.

4.2.1 Differential Entropy

The main part of the tree training process involves the search for optimal parameters for the split function at each node. Entropy and the derived information gain are used as the measure of quality for a split. While the split function itself is independent from the actual task a decision tree is used for, the definition and calculation of the entropy is not. Two different forms of entropy have already been presented: The Shannon entropy for the discrete case and the differential Shannon entropy as its continuous counterpart. In the subject of regression, each regressor x and response variable y is a continuous value, for example $x, y \in \mathbb{R}$. In the case of p regressors and q response variables both $\mathbf{x} \in \mathbb{R}^p$ and $\mathbf{y} \in \mathbb{R}^q$ are vectors of continuous values respectively. Thus, the differential entropy has to be used here. It is defined as

$$H(\mathcal{S}) \;:=\; -\int_{\mathbf{y}\in\mathbb{R}^q} p(\mathbf{y}) \log p(\mathbf{y})\, d\mathbf{y}, \tag{4.1}$$

with the probability distribution $p(\mathbf{y})$ stating the likelihood of the response vector $\mathbf{y}$ given a set of data points $\mathcal{S}$. Since the presented regression trees and forests are based on linear regression, it can now be used to obtain such a probability distribution. By calculating a linear regression model on $\mathcal{S}$, it was shown in Section 3.4.4 how its statistical properties lead to the desired distribution, albeit in the form $p(\hat{\mathbf{y}}_i \mid \mathbf{x}_i)$ where $\hat{\mathbf{y}}_i$ is the prediction of the regression model given the input $\mathbf{x}_i$. Therefore, the entropy of $\mathcal{S}$ is based on $p(\hat{\mathbf{y}}_i \mid \mathbf{x}_i)$ for each $\mathbf{x}_i \in \mathcal{S}$. More formally,

$$H(\mathcal{S}) = \sum_{\mathbf{x}_i\in\mathcal{S}} -\frac{1}{|\mathcal{S}|} \int_{\hat{\mathbf{y}}_i\in\mathbb{R}^q} p(\hat{\mathbf{y}}_i \mid \mathbf{x}) \log p(\hat{\mathbf{y}}_i \mid \mathbf{x})\, d\hat{\mathbf{y}}_i \tag{4.2}$$

describes the mean entropy of $n = |\mathcal{S}|$ random variables $\hat{\mathbf{y}}_i$, each of them dependent on a single data point $\mathbf{x}_i$ [CSK12, p. 219].

While it was not further motivated in Section 3.4.4, modeling $\hat{\mathbf{y}}_i$ as a normally distributed random variable now enables a simple entropy calculation, despite the

infinite integral in the above equation. Considering the case where $\hat{\mathbf{y}}_i = \hat{y}_i \in \mathbb{R}$, Section A.5 shows that the entropy can be calculated as

$$H(\mathcal{S}) = \sum_{\mathbf{x}_i \in \mathcal{S}} \frac{1}{2\,|\mathcal{S}|} \log\left(2\pi e \sigma^2_{\hat{y}_i}\right), \tag{4.3}$$

where $\sigma^2_{\hat{y}_i}$ is the variance of $\hat{y}_i$ given $\mathbf{x}_i$. In the multivariate case with q response variables, the multivariate probability distribution $p(\hat{\mathbf{y}}_i \mid \mathbf{x}_i)$ can be used to obtain a similar entropy formula of the form

$$H(\mathcal{S}) = \sum_{\mathbf{x}_i \in \mathcal{S}} \frac{1}{2\,|\mathcal{S}|} \left(q \log(2\pi e) + \log\left(\det\left(\Lambda_{\hat{\mathbf{y}}_i}\right)\right)\right), \tag{4.4}$$

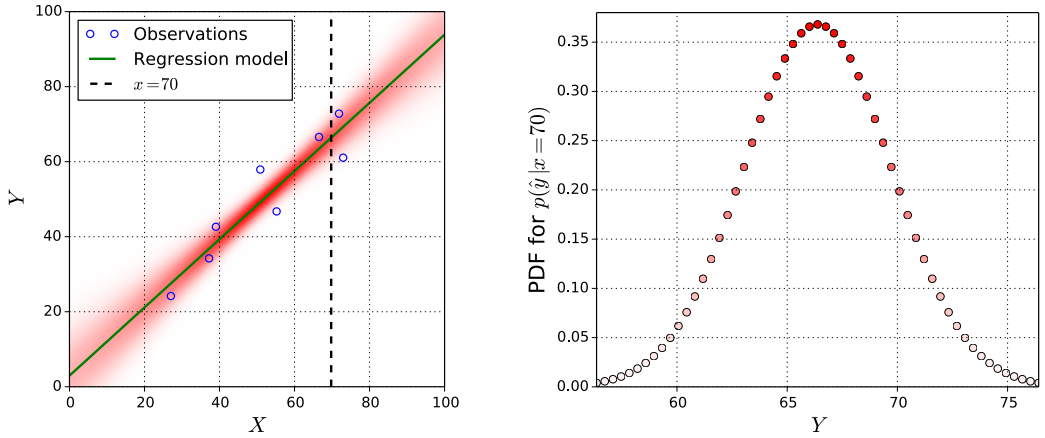
where $\Lambda_{\hat{\mathbf{y}}_i}$ denotes the covariance of $\hat{\mathbf{y}}_i$ [AG89, p. 688]. Due to the form of $\Lambda_{\hat{\mathbf{y}}_i}$ with only its diagonal entries being unequal to zero (see Equation 3.40), its determinant $\det(\Lambda_{\hat{\mathbf{y}}_i})$ is simply the product of the diagonal entries. That way, the differential entropy can in fact be calculated in both the univariate and the multivariate case.

During tree training, the parameters of a split function are evaluated by deriving linear regression models for the complete set of data arriving at an inner tree node as well as the two split subsets. The information gain as the difference in entropy then reflects the uncertainty of the complete model compared to the uncertainty of the two sub-models. Even though these models are not used for later predictions inside the decision tree, they play a decisive role in the optimization process.

4.2.2 Probabilistic Leaf Predictions

In the presented form of regression forests, the predictions of the individual trees are based on linear regression. But instead of computing a single regression model for the complete training data, the data is split up during tree training, until small subsets of the data reside at the leaves of the tree. For each leaf node j and its corresponding set of data $\mathcal{S}_j$, a linear regression model is calculated and stored at the leaf. One of the characteristics of the unified model is the fact that once a tree is trained and it is applied to a new sample $\mathbf{x}$, the tree does not return a single prediction $\hat{\mathbf{y}}$, but instead a complete probability distribution $p(\hat{\mathbf{y}})$. Therefore, additionally to the regression model itself its statistical properties are stored, too. Namely, this includes the estimated error variance σ^2 and the covariance of the model parameters $\mathrm{cov}(\mathbf{b})$ (see Section 3.4.3 and Section 3.4.4). With this supplemental information, a distribution $p(\hat{\mathbf{y}} \mid \mathbf{x})$ can be derived for every new instance $\mathbf{x}$ that the tree is applied to (see Equation 3.30).

Assuming a regression tree in a simple 2D regression setting is applied to a new instance $\mathbf{x} = (70)$, the tree is traversed until a leaf node is reached. The path from the root node to a leaf is determined by the split functions in the tree nodes. Figure 4.2a shows the linear regression model stored at that leaf. By applying $\mathbf{x}$ to



(a) The linear regression model stored at a leaf node.

(b) The probability density function for $\mathbf{x} = (70)$.

Figure 4.2: A new instance $\mathbf{x} = (70)$ arrives at a leaf node. The linear regression model stored at the leaf is applied to it. Using the statistical properties of the model, the mean and variance of the prediction can be derived. This defines the returned probability distribution.

the model, the mean and variance of the prediction $\hat{\mathbf{y}}$ can be derived. Figure 4.2b shows the probability density function of the probability distribution returned by the leaf.

4.2.3 Regressor Selection

One important aspect in linear regression is the selection of regressors. Before actually deriving a regression model, the variables that act as regressors have to be chosen. This is a non-trivial task and it is often necessary to evaluate several regression models obtained through different regressor selections in order to find the best one. The usage of more regressors compared to less does not necessarily lead to a better model and may even have a negative effect [YS09, p. 64]. Additionally, deriving a regression model with a large number of regressors can become a computationally expensive task.

In the introduction of the unified model the possibility of high dimensional data as well as unlimited dimensionality through further feature generation on the initial data was mentioned. Nevertheless, inside the decision trees only small subsets of the available data dimensions are used at the individual nodes, dependent on the selections inside the split functions. These selections are chosen as the best from a random set of possibilities during the node optimization process. Therefore, the same can be done to select the regressors used for the linear regression models needed in the tree training. Since the optimization process for a node j incorporates

entropy calculation on a set of data points and possible subsets, a regressor selection is necessary here. The simplest form of obtaining such a selector function is to reuse the split function parameter $\theta = (\phi, \boldsymbol{\psi}, \boldsymbol{\tau})$ and more specifically the selector $\phi(\cdot)$ within, which is already part of the optimization process. It is originally intended for the dimension selection for feature calculation inside the split function. By reusing it, the same data dimensions are responsible for both the definition of the split as well as its quality assessment. Thus, the entropy of a set of data points $\mathcal{S}_j$ is then obtained from a linear regression model on $\mathcal{S}_j^{'}$ with

$$\mathcal{S}_j^{'} \quad := \quad \{\phi(\mathbf{x}_i) \mid \mathbf{x}_i \in \mathcal{S}_j\}, \tag{4.5}$$

which describes the set reduced to the dimensions used as regressors. This is done for the left and right subsets accordingly.

While this regressor selection is used for the temporary regression models inside the inner tree nodes, the permanently stored linear models at the leaf nodes require such a selection, too. At this point, the original tree training has to be changed slightly. The creation of the leaf nodes now also includes an optimization process, although of simpler form. With the arriving subset of data $\mathcal{S}_k$ at leaf k, different selectors $\phi_l(\cdot)$ from randomly drawn possibilities are applied. This leads to different sets $\mathcal{S}_{k,l}$ containing the data points with reduced dimensionality as in Equation 4.5. Again, an entropy function is used to evaluate different regressor selections. The best selection $\phi^*(\cdot)$ stays defined as

$$\phi^* \quad = \quad \arg\max_{\phi_l} H(\mathcal{S}_k). \tag{4.6}$$

Like above, the entropy $H(\mathcal{S}_k)$ is calculated through a linear regression model on $\mathcal{S}_{k,l}$. The linear model obtained through the best selection is then stored at the leaf node.

5 Implementation Details

So far, the structure and function of regression forests was presented. This theoretical discussion is the basis for an implementation of regression forests within the scope of the *fertilized* library[1] for decision trees and forests of various kinds. For the actual implementation some further problems and pitfalls have to be taken into consideration to obtain a functional and robust framework. In Section 5.1 further remarks on tree training are made focussing on disadvantageously shaped data and ways to handle these cases. While this preserves the equivalence of the implementation to the theoretical model, Section 5.2 presents a way to improve the training efficiency by simplifying the optimization process involved and at the same time maintain equally good regression results.

5.1 Handling ill-conditioned Data

In order to derive a linear regression model from a set of data, some conditions need to be fulfilled. Recalling the least squares technique, the optimal linear model was defined as

$$\mathbf{b} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{5.1}$$

where $\mathbf{y}$ contains the response values and $\mathbf{X}$ is the design matrix containing the data samples as its rows (see Equation 3.9). This solution is only available if $\mathbf{X}^T\mathbf{X}$ is invertible and thus is non-singular. In order to be non-singular, the matrix $\mathbf{X}$ has to be of full rank. Since the columns of $\mathbf{X}$ represent the regressors of the linear model, linear dependency between any of the regressors leads to reduced rank and thus precludes the solution above.

In the scope of regression tree training, the initial set of data samples is intended to be as large as possible. If linear dependency occurs in this initial set, the linear dependent dimensions have to be removed before the tree training is started. While the absence of linear dependency can be guaranteed for the initial data samples, smaller subsets are generated during the training due to the splitting behaviour at the inner tree nodes. Because the subsets are generated by splitting the data according to a feature based on the input data, correlation between the samples and the dimensions is increasingly likely. Thus, at some point, linear dependency

[1] www.fertilized-forests.org

i	x_1	x_2	y
⋮	⋮	⋮	⋮
4	1.2	2.4	2.64
5	1.3	2.6	2.99
6	2.4	4.8	8.16
7	4.4	1.2	19.96
8	4.4	2.2	20.46
9	4.4	9.1	23.91
⋮	⋮	⋮	⋮

Table 5.1: A set of data samples used in a regression tree that can lead to subsets with linear dependency inside the data.

between several input dimensions might occur during training, even though not present in the initial set of samples.

Table 5.1 shows a part of some exemplary data used in a regression tree, containing two input dimensions and a single output value. From the six samples shown, it is obvious that there is no linear dependency between x_1 and x_2 for the complete set of data. Since linear regression models are derived from increasingly small subsets of the data the deeper the tree becomes, some possible subsets are taken into consideration. If at some point during tree training a subset containing the samples with indices $i \in \{4, 5, 6\}$ needs to be assessed, a linear regression model has to be derived from these samples. In this case, a dependency of the form $x_2 = 2x_1$ occurs and the discussed method for linear regression fails. A less obvious problem arises from a potential subset with samples $\{7, 8, 9\}$. No linear dependency between x_1 and x_2 occurs. But the design matrix $\mathbf{X}$ for linear regression takes the form

$$\mathbf{X} = \begin{pmatrix} 4.4 & 1.2 & 1 \\ 4.4 & 2.2 & 1 \\ 4.4 & 9.1 & 1 \end{pmatrix}. \tag{5.2}$$

Because the linear model includes a constant factor (see β_0 in Equation 3.3), a constant regressor x_3 with value 1 is added to the samples. This again leads to linear dependency, this time of the form $x_1 = 4.4x_3$. Therefore not only correlation between the input dimensions can lead to linear dependency at some point, but also equalities within different samples.

In Section 3.3, a possibility to obtain a solution for a linear regression model even from data suffering of linear dependency was mentioned. This can be done by using the generalized inverse instead of the non-existent inverse of $\mathbf{X}^T\mathbf{X}$. While this is a valid possibility, the generalized inverse is not unique in the case of a $\mathbf{X}^T\mathbf{X}$ being singular and thus the obtained solution is neither [SS90, p. 30]. Therefore, there

is no guarantee for the optimality of the solution. In order not to go beyond the scope of this work, the details of this approach are omitted. In general, solving linear regression problems in the presence of linear dependency is an often encountered problem and a variety of solutions exists (see e.g. [Bjö96]). While these methods focus on the linear model itself, they do not enable the statistical derivation and quality estimation of the model as it was presented in Section 3.4. Since this is a decisive part of the regression forests in this work, the approach using the generalized inverse is the only valid solution. Because of the mentioned disadvantages of this approach, it is also neglected in favor of a more elegant way to handle this problem. The actual strategy used is to avoid such ill-conditioned cases entirely. This is possible due to the high dimensionality of the data that regression forests are intended for. As described in Section 4.2.3, the linear regression models during tree training are derived from a rather small number of available data dimensions. This alone makes a linear dependency between any of the regressors less probable due to the fact that there usually are only few. Nonetheless, such an ill-conditioned case can still appear.

During node optimization (inner nodes as well as leaf nodes), a set of randomly drawn dimension selections used as regressors is assessed. The number of different selections depends on the amount of randomness that is intended to be injected in the tree training. If one of these selections leads to regression data with no solution obtainable from, it is simply rejected in favor of other possible selections. As described, this scenario is more likely the smaller the sample subsets become. Therefore, specifying a minimum number of samples for leaf nodes or a maximum tree depth can greatly alleviate the need to reject regressors. This can especially be advantageous if otherwise regressors are rejected that would actually perform well on a larger number of samples but can not be used at the leaf nodes with too small data subsets due to arising linear dependency in the data. These constraints are of course specific to the actual application and have to be evaluated with respect to the structure and quality of the available data. In Section 6.2 the effect of possible linear dependency in the input data will be discussed.

5.2 Efficiency Improvements through simpler Optimization

The vital part of the regression tree training is the optimization of the split function at each inner tree node as it was described in Section 2.2.1. For this optimization, a randomly drawn set of split function parameters is assessed, including a selector ϕ, a geometric surface $\boldsymbol{\psi}$ and a pair of thresholds $\boldsymbol{\tau}$. Each parameter combination induces a split on the set of data samples $\mathcal{S}$ at the corresponding tree node, which leads to the left and right subsets $\mathcal{S}_L$ and $\mathcal{S}_R$. These two subsets are then evaluated using linear regression. Thus, for each parameter combination two linear regression models have to be computed.

While this approach works and is included in the provided implementation, its practical application is limited. This is due to the computational overhead arising from the calculation of many linear regression models during each split optimization. In contrast to the models stored at the leaf nodes, these models are only needed during optimization and are discarded afterwards. Thus, depending on the number of data samples, the amount of randomness and the number of regression trees, this optimization can become impractical. This motivates a simpler optimization strategy that greatly reduces the time consumed by tree training but at the same time results in similar regression performance. In fact, the same conclusion was drawn in the exemplary implementation of regression forests included in the *Sherwood* project[2] which has been provided alongside [CS13]. In this implementation, no regression model is computed at all to evaluate a split. Furthermore, the annotation of the data samples is ignored completely. Instead, a split is just rated by how well the samples are separated with respect to the regressors.

In the context of this work, a different approach compared to the Sherwood implementation is presented. For the optimization of the split functions inside inner tree nodes, no linear regression is used but much simpler constant regression instead. Section 5.2.1 gives a short overview of constant regression and its usage for split evaluation. In Section 5.2.2, the computational efficiency gained when using constant regression for split function optimization is discussed. Finally, Section 5.2.3 compares the two different optimization strategies by means of a simple example.

5.2.1 Constant Regression

This section introduces constant regression as a simplified alternative to linear regression. Due to its similarity to linear regression, it is only discussed as detailed as needed for the usage in regression tree training. A constant regression model is of the form

$$y = \beta + \varepsilon, \tag{5.3}$$

with β being a constant value and ε being again a normally distributed random error with expectation 0 and variance σ^2. The main difference to the linear case is that there are no mutable regressors included in the model. Deriving a constant regression model consists of calculating the constant b as an estimation for β. Given an input $\mathbf{x}$, the prediction based on a constant regression model is simply

$$\hat{y} = b. \tag{5.4}$$

Thus, the prediction is independent from the actual input. Nonetheless, the derivations regarding linear regression presented in Chapter 3 can also be applied to a

[2] research.microsoft.com/en-us/projects/decisionforests/

constant model, since it is only a specialization of the linear form. But, because of the mentioned independence, the equations are greatly simplified. Again, the model is derived from a set of n data samples $\mathbf{x}_i$ and their corresponding response variable y_i. Using the residual sum of squares, the estimated model parameter b is given by

$$b = \frac{\sum_{i=1}^{n} y_i}{n} = \overline{y}, \tag{5.5}$$

which is simply the mean response variable of the data samples. This result can be directly derived from Equation 3.14. While in the presented simplified split optimization, linear regression is replaced in favor of constant regression, the differential Shannon entropy is still used to evaluate a given regression model. Therefore, the same statistical derivations for linear regression beginning with Section 3.4.3 are applied to the constant regression model. The two main results obtained are the estimated error variance and the prediction variance. The estimation s^2 for the error variance σ^2 is

$$\begin{aligned} s^2 &= \frac{RSS(b)}{n-1} \\ &= \frac{\sum_{i=1}^{n} (y_i - b)^2}{n-1}, \end{aligned} \tag{5.6}$$

which directly coincides with the linear case (see Equation 3.22). The prediction variance can be expressed as

$$\operatorname{var}(\hat{y}) = \frac{\sigma^2}{n}, \tag{5.7}$$

using Equation 3.27. Combining both equations, the probability distribution expressing the likelihood of prediction $\hat{y}$ given an input $\mathbf{x}$ can be estimated as

$$p(\hat{y} \mid \mathbf{x}) = p(\hat{y}) = N\left(b, \frac{s^2}{n}\right). \tag{5.8}$$

Using this information, the differential entropy can be applied to a constant regression model to assess its quality in the same way as described in Section 4.2.1.

5.2.2 Incremental Solutions

Constant regression was introduced with the intent to obtain a simpler and faster optimization process for the split functions incorporated in regression tree training. While the above equations related to constant regression are in fact simpler than in the linear case, the computational efficiency gained has yet to be evaluated. In order to optimize a split function at an inner tree node, the arriving set of data

samples $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is assessed. Considering a fixed selector ϕ and surface $\boldsymbol{\psi}$ from the randomly drawn subset of the parameter space, each sample is described by a feature v_i, for example in the case of a linear surface

$$v_i \quad := \quad \phi\left(\mathbf{x}_i\right) \boldsymbol{\psi}^T, \; \forall i = 1, \ldots, n.$$

This feature now defines a sorting permutation $\mathcal{P} : \mathbb{N} \to \mathbb{N}$ with

$$v_{\mathcal{P}(1)} \leq v_{\mathcal{P}(2)} \leq \cdots \leq v_{\mathcal{P}(n)}.$$

In the case of a one-sided threshold $\boldsymbol{\tau}$, for example $\boldsymbol{\tau} = (\tau_1, -\infty)$, there are only $n-1$ different splits that can be induced by τ_1 based on the given order no matter how many different thresholds actually have to be assessed. In the original optimization process, each split has to be evaluated through newly computed linear regression models. The main advantage of a constant regression model is that its solution can be iteratively computed for all $n-1$ possible splits.

Beginning with the first possible split $\mathcal{S}_{L1} = \left\{\mathbf{x}_{\mathcal{P}(1)}\right\}$ and $\mathcal{S}_{R1} = \mathcal{S} \setminus \mathcal{S}_{L1}$, initial constant regression models are computed for both subsets. These models are described by parameter b_{L1} and s^2_{L1} for the left model and parameter b_{R1} and s^2_{R1} for the right one. To obtain the other possible splits, samples are iteratively shifted from $\mathcal{S}_{R1}$ to $\mathcal{S}_{L1}$. Thus, the second possible split is defined by $\mathcal{S}_{L2} = \left\{\mathbf{x}_{\mathcal{P}(1)}, \mathbf{x}_{\mathcal{P}(2)}\right\}$ and $\mathcal{S}_{R2} = \mathcal{S} \setminus \mathcal{S}_{L2}$. In Section A.6 it is shown that the initial left and right regression models just have to be changed slightly in order to reflect this shift:

$$b_{L2} = \frac{|\mathcal{S}_{L1}|}{|\mathcal{S}_{L2}|} b_{L1} + \frac{1}{|\mathcal{S}_{L2}|} y_{\mathcal{P}(2)}, \tag{5.9}$$

$$s^2_{L2} = \frac{|\mathcal{S}_{L1}| - 1}{|\mathcal{S}_{L1}|} s^2_{L1} + \frac{1}{|\mathcal{S}_{L2}|} \left(y_{\mathcal{P}(2)} - b_{L1}\right)^2 \tag{5.10}$$

$$b_{R2} = \frac{|\mathcal{S}_{R1}|}{|\mathcal{S}_{R2}|} b_{R1} - \frac{1}{|\mathcal{S}_{R2}|} y_{\mathcal{P}(2)} \tag{5.11}$$

$$s^2_{R2} = \frac{|\mathcal{S}_{R2}|}{|\mathcal{S}_{R2}| - 1} s^2_{R1} - \frac{|\mathcal{S}_{R1}|}{\left(|\mathcal{S}_{R2}|\right)\left(|\mathcal{S}_{R2}| - 1\right)} \left(y_{\mathcal{P}(2)} - b_{R1}\right)^2. \tag{5.12}$$

Such an iterative adjustment can not only be used for the first sample shift from right to left but to calculate the constant regression models for all $n-1$ different splits. According to Equation 5.5 and Equation 5.6, the initial regression models can be computed in linear time. Each shift of a sample from right to left requires only a small constant number of elementary calculations to be reflected by the regression models. Thus, when using a one-sided threshold the constant regression models for all possible splits defined by a fixed ϕ and $\boldsymbol{\psi}$ can be computed very efficiently, namely in $\mathcal{O}\left(n\right)$ steps in addition to $\mathcal{O}\left(n \log n\right)$ steps for the initial sorting.

The impact of this change on the overall efficiency of the tree training process will be presented in Chapter 6.

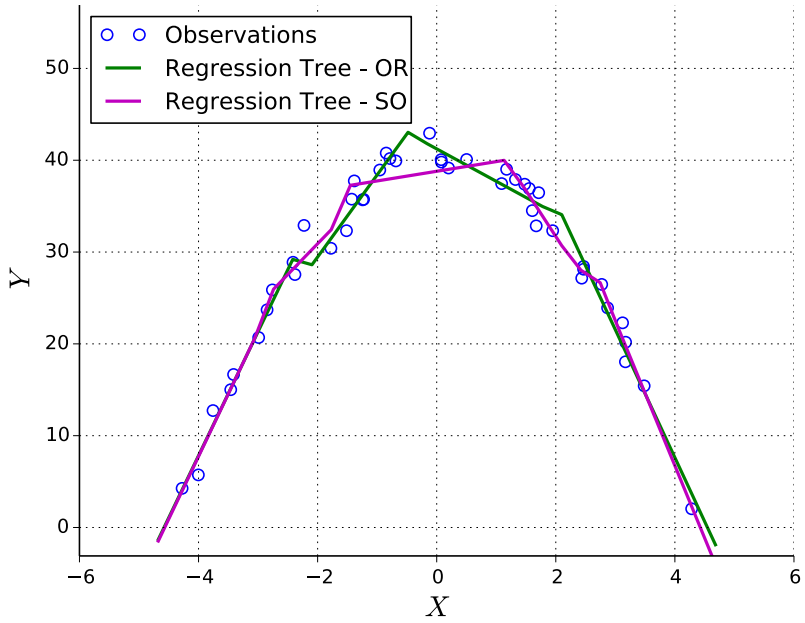


Figure 5.1: Performance of single regression trees on perturbed parabola shaped 2D data points. The OR-tree was trained using the original optimization strategy for its inner tree nodes. The SO-tree used the simplified optimization using constant regression. Both trees use linear regression models at their leaf nodes. Despite its larger depth the SO-tree is not capable of reflecting the parabola shape as well as the OR-tree.

5.2.3 Comparison

While a quantitative comparison of the original optimization process and the simplified one described above will follow in the next chapter, a short example will be discussed here. For a descriptive comparison, a 2D regression setting is used with data samples obtained from a polynomial function perturbed with noise. The performance of single regression trees as well as small regression forests will be presented with both the original optimization process (referred to as OR) and the simplified optimization (SO) using constant regression. It is important to note that these two forms of regression trees and forests differ in the used regression models during split optimization, but both use linear regression models at their leaf nodes.

Figure 5.1 shows the performance of single regression trees with no randomization. The OR result is based on a regression tree constructed with the original optimization definition while the SO result reflects the performance of a regression tree with the simplified optimization. While both trees are capable of reflecting the parabolic shape of the polynomial data, the OR-variant performs slightly better. Especially at the peak of the parabola, the OR-variant fits closer to the data

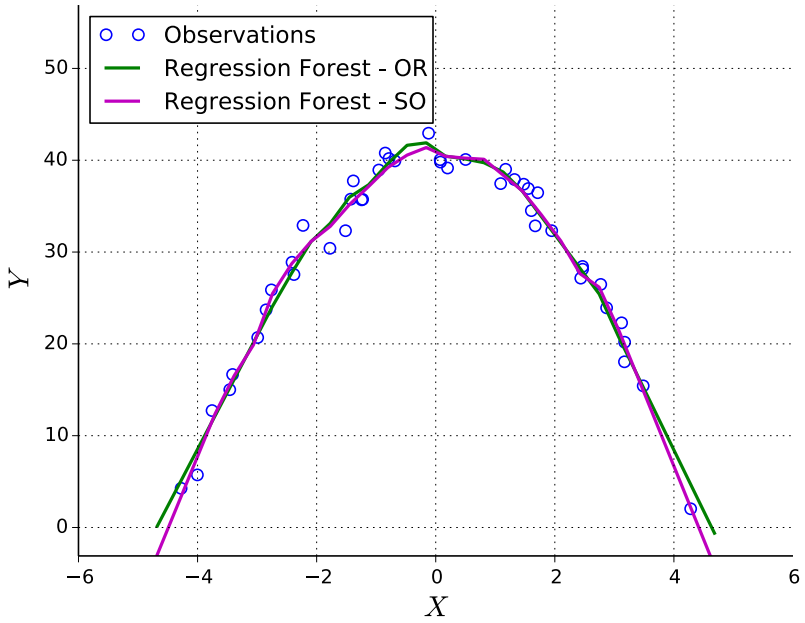


Figure 5.2: Performance of regression forests built from five randomized regression trees each, again on the parabola shaped 2D data points. The OR-forest was constructed using the original optimization strategy for the inner tree nodes of its trees. The SO-forest used the simplified optimization using constant regression. Both approaches lead to nearly identical results.

than the SO-variant. Both trees were trained with the same parameters using axis-aligned surfaces and a one-sided threshold. The only exception to this is the maximum tree depth. The OR-tree has a tree depth of two. Using a SO-tree of equal depth leads to way inferior results such that a direct comparison does not provide much insight. In order to obtain similar results with the SO-variant, the tree has to be grown twice as deep to a maximum tree depth of four.

Figure 5.2 shows the results of the same experiment, this time with small regression forests containing five trees. Again, the different optimization strategies are compared. The training of both forests used the same parameters and the same amount of randomness. Only the tree depth again differed with a depth of two for the OR-variant and a depth of four for the SO-variant due to the same reason as before. This time, the results of both regression forests are nearly identical. Despite the simpler and faster tree training process, the SO-variant performs equally well compared to the OR-variant. The effect of using an ensemble of randomly trained regression trees is able to compensate the drawbacks of the simpler tree training. This motivates the further evaluation of regression forests containing trees that use constant regression for the optimization process at their inner nodes but keep the more complex linear models at the leaves where the actual prediction is performed.

While the results of this approach were shown for a simple 2D example, the same comparison will be done in the next section using larger, multivariate datasets.

6 Results on Regression Datasets

Until now the performance of regression forests was evaluated using descriptive low dimensional examples which made it easy to indicate some of the key properties of the discussed forests. Nevertheless, they are intended to be used on large-scale multivariate data. This chapter will present the results on two well known datasets used for regression performance evaluation. In order to have meaningful results they are compared to the results of another form of regression forests. These forests will be briefly introduced in Section 6.1 and the main differences will be highlighted. Section 6.2 covers the Boston Housing dataset. Using it as a regression problem, the main goal is to predict average house prices based on statistical data regarding population, infrastructure and the environment. The results on this dataset will be discussed including the influence of the quality and form of the data. In Section 6.3, the Abalone dataset is introduced. The related regression problem consists of predicting the age of sea snails based on biological statistics. Again, the results of the different forms of regression forests will be compared and discussed.

6.1 Regression Forests from the Scikit-learn Library

In order to rate the achieved results on the datasets in the next sections, results from another machine learning implementation for regression are needed for comparison. In this case, a different form of random regression forests is used. These forests are part of the *scikit-learn* library[1], a collection of various machine learning techniques implemented in Python. Albeit the interface of the library is provided in Python, much of the internal implementation is done with Cython in order to combine C with Python and therefore reach efficiency equal to compiled languages [PVG+11, p. 2827]. Before using the scikit-learn decision forests (in the following referred to as SK-forests) on the mentioned datasets, a short overview over their function and structure is given. The main differences to the regression forest implementation provided alongside this work are shown using descriptive examples.

[1] scikit-learn.org, version 0.15.2, 09.04.2014

6.1.1 Forests of randomized CART-Trees

The SK-forests are based on the random forests proposed by Breiman [Bre01] although with some simplifications. The individual regression trees are created by a simplified and randomized version of the CART-algorithm (**C**lassification **A**nd **R**egression **T**rees) from [BFSO84].

In the terms of the unified model from Chapter 2, the split functions at the inner tree nodes are limited to the selection of single data dimensions, axis-aligned surfaces and one-sided thresholds. The split optimization is based on constant regression similar to the method presented in Section 5.2, although no entropy measure is used for the information gain as the optimization criterion. Instead, a minimization of the mean squared error is performed. Furthermore, the regression prediction at the leaf nodes also uses a constant regression model in contrast to the probabilistic linear model presented in this work. The tree training has two sources of randomness: On the one hand, a randomly drawn subset of data dimensions is evaluated for split optimization at each tree node. On the other hand, each tree of the regression forest is trained on a different subset of data samples using bootstrapping. The subset for each tree is drawn randomly from the complete set with replacement [Sl14]. This can be seen as a random weighting of the individual data samples. Thus, a combination of both randomization techniques presented in Section 2.2.3 is used. No randomization concerning the threshold selection is involved. Instead, the optimal threshold for each split function is obtained by an exhaustive search. This is possible because of the finite set of thresholds leading to unique splits (see Section 5.2.2).

Figure 6.1 shows the same regression setting as in the examples from the previous chapter. The result of a regression tree using linear regression from the fertilized library (below referred to as fertilized-tree) is compared to a single regression tree from the scikit-learn implementation (SK-tree). Due to the higher complexity of linear models, the fertilized-tree only needs a depth of two to achieve good results. The SK-tree has to be grown to a depth of six to obtain similar results in reflecting the parabolic shape of the data. Most obvious are the stair-like artifacts which result from the usage of constant regression models at the tree leaves. Furthermore, the fertilized-tree is more capable of extrapolating regions of the parabola where data samples are missing.

6.1.2 Comparison of the Forest Models

While the previous example gave insight into the performance of individual regression trees, the overall intention is to use regression forests to obtain more generalization (less overfitting). The performance of individual regression trees does not necessarily hint at the performance of a randomized regression forest. Therefore, another experiment is presented in order to compare the properties of fertilized-

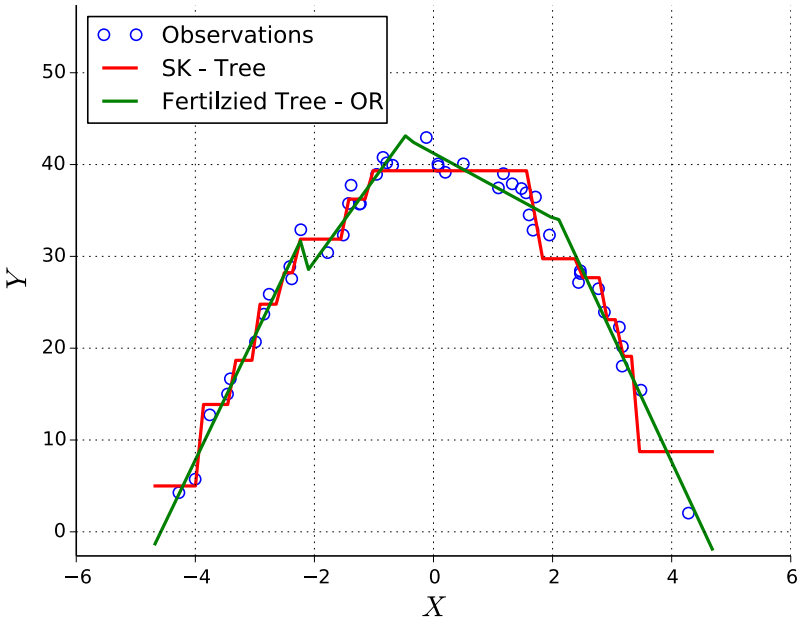


Figure 6.1: Comparison of a single regression tree from the fertilized library and a regression tree from the scikit-learn project. Again, the polynomial 2D data is used. Due to the usage of constant regression models in the SK-tree, a deeper tree is needed to achieve similar results. Using linear models instead of constant ones creates a smoother result without the stair-like artifacts.

forests and SK-forests. Again a 2D-regression setting is used to gain descriptive results. The set of data samples is taken from [CSK12].

Figure 6.2 shows the results on the 2D data. Both forests were constructed from 25 regression trees. In the case of the SK-forest, the individual trees again had to be grown deeper compared to the fertilized-forest. In both cases the results on the parabolic shaped part of the data are nearly identical. The results on the data samples arranged as a line on the right are also similar, albeit the SK-forest tends to overfit slightly more to the data due to its larger depth. The biggest difference is notable between the two clusters of data samples. While the SK-forest covers this area of the input space with a single step, the fertilized-forest produces a smoother result when transitioning from the left sample cluster to the right. Nonetheless, both approaches lead to overall good results. The stair-like artifacts in the SK-tree result from the experiment before are largely remediated by the usage of tree ensembles.

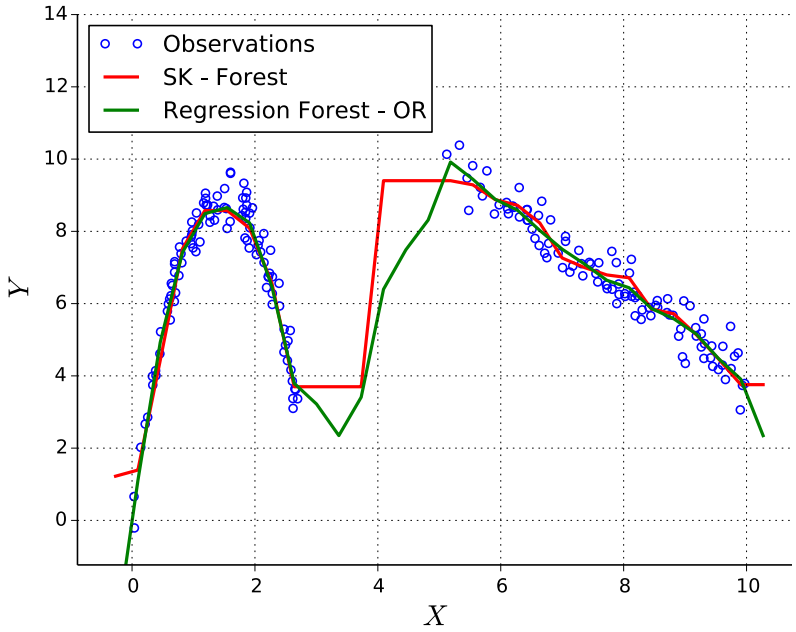


Figure 6.2: Comparison of a fertilized-forest and a regression forest from the scikit-learn project. Both forests lead to similar results, with the biggest difference in the area with missing data samples in the middle. The fertilized-forest tends to extrapolate these areas more smoothly.

6.2 Results on the Boston Housing Dataset

The first two experiments of larger scale to evaluate the performance of the different regression forests use the *Boston Housing* dataset. The dataset is available at the UCI repository[2]. It contains 506 samples representing different neighbourhoods in the area of Boston, Massachusetts. Each sample consists of 14 attributes representing statistics on infrastructure, population and the environment in the relevant neighbourhood. The dataset was first introduced in [HJR78] with the intent to explain the median value of all owner-occupied dwellings in the respective neighbourhood in terms of the other 13 attributes. Of special interest was the influence of air pollution, in this case the concentration of nitrogen oxide, on the median value. Nevertheless, the dataset established itself as a benchmark for regression models beginning with [BKW80]. Of the 14 attributes, 12 are of numeric type while the other two are categorial values.

Similar to the original intention, the regression task is to predict the median dwelling value based on the other attributes. The fertilized-forests were used in their simplest form with axis-aligned surfaces and a one-sided threshold. The incorporated linear regression models used a single regressor variable. Both the OR-

[2] archive.ics.uci.edu/ml/datasets/Housing

and the SO-variants were used. Additionally, a third variant was used which is solely based on constant regression. This applies to the split function optimization as well as the leaf nodes. Denoted as the CR (constant regression) variant, it is very similar to the SK-forests in its function with the only differences regarding the induced form of randomization and the optimization criterion. The results of all variants were then compared to the performance of SK-forests. In the first experiment, the same set of fixed parameters was used for all forest models as far as possible. The maximum depth of each tree is limited to 12, while the minimum number of samples at each leaf is set to four. At each inner node, four randomly drawn input dimensions are used for the split optimization. For the fertilized forests, seven different threshold values are are evaluated for each split optimization. The SK-forests in general use lesser randomization regarding the split parameters since the threshold is always chosen perfectly. Instead, bootstrapping is used to create different training sets for the individual trees. All forests were constructed from 100 trees. As regression performance measure, the mean squared error (MSE) of the prediction was used. In the dataset, the median dwelling value ranges from $5.0 - 50.0$ in \$1000's. Since the dataset is not separated into training and testing data, it was randomly partitioned using 90% of the samples for training and the remaining 10% for testing. This was repeated 100 times and the results were averaged. In addition to the MSE, the time needed for the forest training was measured.

Table 6.1 shows the results of the first experiment on the Boston Housing dataset. It contains the average mean squared errors the different regression forests achieved, as well as the average time in seconds needed for training. The fertilized-forests using the original optimization strategy (OR) did not perform best. The time used for training is roughly 10 to 15 times higher compared to the other forest results. This is expected due to the large number of linear regression models that have to be computed during the split optimization at each tree node. Surprisingly, the regression performance is worse compared to the SO-variant. Despite being the most complex forest in terms of split optimization and leaf predictors, it did not achieve better or equal results. Thus, it seems that using more complex regression models within the trees does not necessarily lead to better results. The regression forests from the scikit-learn library achieved the worst MSE. Albeit the similarity to the CR-variant, the usage of bootstrapping is inferior to a higher randomization of the tree parameters in this experiment. Due to the relative simplicity of the SK-forest,

	Fert.-forest OR	Fert.-forest SO	Fert.-forest CR	SK-forest
MSE	11.01	**9.43**	11.35	13.04
Time / sec	1.43	0.18	**0.09**	0.22

Table 6.1: Results of the first experiment on the Boston Housing dataset using fixed forest parameters.

the training is much more time efficient. Nevertheless, the implementation proves to be slower than both the SO- and the CR-variant, at least in this experiment. The CR-variant led to the fastest tree training, but also to the worst result of the three fertilized-forests. Finally, the SO-variant achieved by far the best result, even though the forest training took only twice as long as the fastest type of forest.

For the second experiment, a fixed partition of the dataset was used. The first 90% of the samples were used for training and the remaining 10% for testing. For all forest models, their parameters were optimized using cross-validation on the training samples. A detailed listing of the optimal parameter values for all forests is omitted. But is was observable that the forests using linear regression performed best with more samples at the leaf nodes compared to the CR- and SK-forests. The optimal amount of randomization was similar for all forest models. Table 6.2 shows the results of the experiment. This time the OR-variant performed worst with respect to both the MSE and the training time. The SO-variant again achieved the best regression result with a nearly equal training time. The CR-variant achieved a similar result compared to the previous experiment, but the optimized parameters led to a doubled time for forest training. Nevertheless, it was still the best training time achieved. The biggest improvement was observed for the SK-forest, which achieved the second best MSE paired with a training time similar to the first experiment. It performed noticeably better compared to the similar CR-variant. In contrast to the first experiment, bootstrapping proved to be superior to a higher tree parameter randomization in this setting.

	Fert.-forest OR	Fert.-forest SO	Fert.-forest CR	SK-forest
MSE	11.52	**9.91**	11.41	10.48
Time / sec	1.84	0.19	**0.17**	0.20

Table 6.2: Results of the second experiment on the Boston Housing dataset using optimized forest parameters.

Since the SO-variant achieved the best regression results in both experiments, the usage of linear regression models as leaves predictors seems to have a positive effect when applied on this dataset. For an explanation of the weaker performance of the fertilized-forests solely based on linear regression (OR) compared to the SO-variant, Table 6.3 shows a selection of four samples from the Boston Housing dataset, reduced to eight of the 13 attributes. A description of the exact meaning of every attribute is omitted here but can be found in [HJR78, p. 96f.]. All four samples are equal with respect to eight of its attributes. This is not surprising for the two categorial attributes. The boolean RIV-attribute describes whether the neighbourhood is adjacent to Charles river or not. The RAD-attribute is an index rating the accessibility to radial highways. Even though it is of integer type, it only takes nine different values throughout the dataset. While the other attributes are of numeric type, the majority of them also take discrete values that repeatedly occur

ZN	INDUS	CHAS	NOX	RAD	TAX	PTRATIO	B
0	9.69	0	0.585	6	391	19.2	396.9
0	9.69	0	0.585	6	391	19.2	396.9
0	9.69	0	0.585	6	391	19.2	396.9
0	9.69	0	0.585	6	391	19.2	396.9

Table 6.3: Selection of four samples from the Boston Housing dataset. The table shows eight of the 13 attributes. With respect to these, all samples are equal.

in many samples. Since samples with similar attributes are likely to be grouped together during tree training, subsets with data equally shaped like in Table 6.3 will repeatedly occur.

As it was described in Section 5.1, attributes that are equal in all samples can not be used for linear regression. Therefore the number of possible regressor selections is largely reduced, especially when more than one regressor is used in the linear regression model. This might be the reason why the OR-variant of the fertilized-forests does not perform as well as the SO-variant. Since its tree training process is completely based on linear regression, the number of attributes that can be used will decrease the deeper a tree gets. Therefore, some of the attributes will hardly influence the tree training and the resulting regression predictions. Because of that, the OR-variant can not use all the information the dataset provides.

The SO-variant does also use linear regression at the leaf nodes of the individual trees. Hence, it suffers from the same problems there. Since the rest of the tree training does not rely on linear regression, all attributes can be used when optimizing the split functions and thus can still be widely incorporated into the overall regression prediction. This seems to have a major effect on the regression performance.

6.3 Results on the Abalone Dataset

For the next experiments, the *Abalone* dataset is used which also is available at the UCI repository[3]. It is larger than the Boston Housing dataset consisting of 4177 samples. Each sample represents an abalone exemplar, a specific form of sea snail [enc07, p. 7f.]. The samples consist of eight attributes. The task assigned with the dataset is to predict the age of abalone exemplars by means of the other seven attributes. Six of them are numeric values describing various size and weight measurement results. The seventh attribute is of categorial type and determines the gender of the exemplar. The dataset does not directly provide the age for each exemplar. “The age of abalone is determined by cutting the shell through the

[3] archive.ics.uci.edu/ml/datasets/Abalone

cone, staining it, and counting the number of rings through a microscope" [UR95]. Therefore, the number of rings is annotated to each sample, which is in the range of $1 - 29$.

Since there is no predetermined separation of training and testing samples in the dataset, such a partition has to be generated again. Due to the larger size of the dataset, 75% of the samples were used for training and the remaining 25% for testing purpose. The size of the partition coincides with the experiments on the Abalone dataset in [Bre01]. For the first experiment, the same setting as in the first experiment on the Boston Housing dataset was used. The same fixed set of parameters was applied to each forest model and the results were averaged for 100 randomly generated dataset partitions.

	Fert.-forest OR	Fert.-forest SO	Fert.-forest CR	SK-forest
MSE	4.78	4.73	4.60	**4.56**
Time / sec	12.86	1.10	0.61	**0.28**

Table 6.4: Results of the first experiment on the Abalone dataset using fixed forest parameters.

Table 6.4 shows the result of the first experiment on the Abalone dataset. This time, the results on regression performance are less distinctive. Yet it appears that incorporating linear regression into the individual regression trees has a negative effect on the performance related to this dataset. The OR-variant of the fertilized forests performed worst. One reason for this might be the categorial attribute incorporated in the dataset which can hardly be used for linear regression. The SO-variant performed only slightly better. Because of the size of the dataset the differences in time consumption for the forest training are larger compared to the Boston Housing results. The OR-variant was by far the slowest due to the overhead of linear regression involved in the split optimization. Even though the SO-variant uses linear regression only at the leaf nodes of each tree, the overhead still is noticeable with a roughly four times longer training time compared to the SK-forests. The CR-variant and the SK-forests achieved similar results. Surprisingly, the training time of the SK-forests is noticeably shorter which contradicts the runtime results on the Boston Housing Dataset. At this point, no assured reason that explains this difference in training time can be given.

For the second experiment, a fixed partition of the dataset is used with the first 75% of the samples as the training set and the remaining samples for testing. Again, the parameters for all forest models were optimized by cross-validation on the training set. The values for the optimal parameters showed the same tendencies as observed in the second Boston Housing experiment. Table 6.5 shows the result of the second experiment. In this setting, all forest models performed equally well. While the OR-variant again achieved the worst regression results, the optimized parameters led to shallower trees and thus a reduced training time. A similar runtime improvement

	Fert.-forest OR	Fert.-forest SO	Fert.-forest CR	SK-forest
MSE	4.40	**4.30**	4.32	4.32
Time / sec	8.75	0.64	0.55	**0.24**

Table 6.5: Results of the second experiment on the Abalone dataset using optimized forest parameters.

was observed for the SO-variant. This time, it performed best among all forests and showed the biggest improvement in regression performance compared to the fixed parameter setting of the first experiment. The CR-variant and the SK-forests achieved an equal MSE, but the same difference in training time as in the previous experiment was observable.

One possible explanation for the similarity in regression performance of all forests is the attribute of the abalone samples describing the number of rings in their shell and therefore their respective age. Since this attribute is of integer type, it only takes 29 different values. Thus in terms of regression, the response variable which has to be predicted is not of continuous type but limited to a small set of discrete values. Figure 6.3 shows the distribution of the samples regarding the number rings. It reveals that the samples are by far not equally distributed among the possible ring numbers. More than 90% of the samples are in the range between five

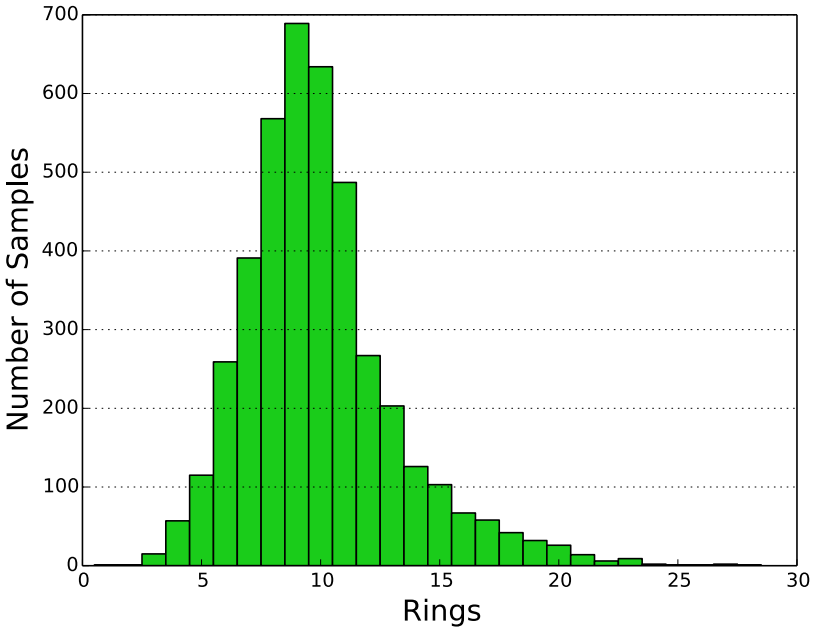


Figure 6.3: Distribution of the abalone samples regarding their annotated number of rings.

and 15 rings. This is the reason why the dataset is also used as a classification task [UR95]. Therefore, the regression task that this dataset offers is rather limited and less complex regression forest are suitable to solve it. The ability to extrapolate, linear regression models incorporate, is not needed here. In fact, it has even a negative influence on the results since the predictions of the individual tree leaves seem to overfit the training data. This negative effect can be alleviated by using more restrictive tree growth rules, for example increase the minimum number of samples at each leaf node as it was the case in the second experiment.

7 Conclusion

The usage of ensembles of randomized decision trees for regression has the ability to improve performance especially in terms of generalization in much the same way as forest-based classification does. The idea to use individual decision trees as weak learners is a fruitful approach with various applications and regression being no exception.

The usage of linear regression within regression trees as motivated in [CSK12] does have a solid theoretical foundation. By avoiding the common pitfalls related to linear regression, a robust implementation has been provided alongside this work. But the conducted experiments indicated a limited practical application. Despite the larger complexity compared to other regression forest models, the learning performance in the presented experiments was never the best. While this is true for the regression forests that are solely based on linear regression, the alternative approach combining constant and linear regression performed favorably on both the Boston Housing and the Abalone dataset. Paired with a larger randomization of the tree parameters, ensembles of these trees were able to outperform the regression forest implementation of the scikit-learn library in three of the four conducted experiments. At the same time, the overhead for forest training was kept feasible.

Nevertheless, the experiments have shown that the results of different regression forest models are highly dependent on the data they are used on. On the one hand, data that is disadvantageously shaped e.g. through the presence of categorial variables can negate the positive effect of linear regression incorporated into regression forests. On the other hand, data that presents only a limited regression task, as in the case of the Abalone dataset, can lead to equally good results albeit the different forest models and their varying complexity. In this case, linear regression paired with too deep trees can negatively influence the results.

In the experiments, the presented regression forests and the incorporated trees were used in their simplest form: The split functions were limited to axis-aligned surfaces and one-sided thresholds. When linear regression was applied, the model was built upon a single regressor only. Effects of the usage of more complex decision surfaces and multivariate linear regression models would be an interesting subject for further examination. Additionally, decoupling the selection of input data for the feature calculation in split functions and the selection of regressors to evaluate the respective split using linear regression is a possible choice. While this would further increase either the complexity of forest training or the involved randomization, it could possibly alleviate to some extent the dependence of the results of regression

forests using linear regression and the quality and shape of the data it is applied to.

Concluding, randomized decision forests are a developing field in machine learning. The unified forest model proposed by Criminisi et al. enabled decision forests to be applied to a variety of machine learning tasks. While decision forests have already proven their quality in the case of classification, the application to regression tasks has yet to gain equal popularity. By arguing on the results of this work, decision forests for regression appear to be a fruitful topic for further research.

A Appendix

A.1 Differentiation of the Residual Sum of Squares

The following derivation is based on [SS90, p. 46].

In order to differentiate Equation 3.10 with respect to $\boldsymbol{\beta}$, the equation is further transformed:

$$\begin{aligned} RSS\left(\boldsymbol{\beta}\right) &= \sum_{i=1}^{n}\left(y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right)^2 \\ &= \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^T\left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) \\ &= \mathbf{y}^T\mathbf{y} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\ &= \mathbf{y}^T\mathbf{y} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} \\ &= \mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{y}\right) + \boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta} \end{aligned} \tag{A.1}$$

Now, the differentiation is given by

$$\begin{aligned} \frac{\delta RSS\left(\boldsymbol{\beta}\right)}{\delta\boldsymbol{\beta}} &= \frac{\delta}{\delta\boldsymbol{\beta}}\left(\mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{y}\right) + \boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta}\right) \\ &= \frac{\delta}{\delta\boldsymbol{\beta}}\left(\mathbf{y}^T\mathbf{y}\right) - 2\frac{\delta}{\delta\boldsymbol{\beta}}\left(\boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{y}\right)\right) + \frac{\delta}{\delta\boldsymbol{\beta}}\left(\boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta}\right). \end{aligned} \tag{A.2}$$

The differentiation of the three terms in Equation A.2 will be derived separately. First, it can be noticed that

$$\frac{\delta}{\delta\boldsymbol{\beta}}\left(\mathbf{y}^T\mathbf{y}\right) = \mathbf{0} \tag{A.3}$$

By defining $\mathbf{v} \coloneqq \mathbf{X}^T\mathbf{y}$ with $\mathbf{v} \in \mathbb{R}^{p+1}$, the second term in Equation A.2 is simplified to

$$\frac{\delta}{\delta\boldsymbol{\beta}}\left(\boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{y}\right)\right) = \frac{\delta}{\delta\boldsymbol{\beta}}\left(\boldsymbol{\beta}^T\mathbf{v}\right). \tag{A.4}$$

Its differentiation with respect to a single β_i is

$$\frac{\delta}{\delta\beta_i}\left(\boldsymbol{\beta}^T\mathbf{v}\right) = v_i, \forall i = 0, 1, \ldots, p. \tag{A.5}$$

This leads to the differentiation with respect to $\boldsymbol{\beta}$ as

$$\frac{\delta}{\delta\boldsymbol{\beta}}\left(\boldsymbol{\beta}^T\mathbf{v}\right) = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_p \end{pmatrix} = \mathbf{v} = \mathbf{X}^T\mathbf{y}. \tag{A.6}$$

It is helpful to define the symmetric matrix $\mathbf{A} \coloneqq \mathbf{X}^T\mathbf{X} \in \mathbb{R}^{p+1\times p+1}$, with $\mathbf{A} = (\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_p)$, and $\mathbf{a}_i = (a_{i0}, a_{i1}, \ldots, a_{ip})^T$ being the i-th row of $\mathbf{A}$. Using this for the third term in Equation A.2, it follows that

$$\begin{aligned} \boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta} &= \boldsymbol{\beta}^T\mathbf{A}\boldsymbol{\beta} = \sum_{i=0}^{p}\beta_i\left(\sum_{j=0}^{p}a_{ji}\beta_j\right) \\ &= \sum_{i=0}^{p}\sum_{j=0}^{p}a_{ji}\beta_j\beta_i. \end{aligned} \tag{A.7}$$

For $m \in \{0, 1, \ldots, p\}$, the differentiation of Equation A.7 with respect to β_m is

$$\begin{aligned} \frac{\delta}{\delta\beta_m}\left(\sum_{i=0}^{p}\sum_{j=0}^{p}a_{ji}\beta_j\beta_i\right) &= \frac{\delta}{\delta\beta_m}\left(\sum_{\substack{i=0\\ i\neq m}}^{p}\sum_{j=0}^{p}a_{ji}\beta_j\beta_i\right) + \frac{\delta}{\delta\beta_m}\left(\sum_{\substack{j=0\\ j\neq m}}^{p}a_{jm}\beta_j\beta_m\right) + \frac{\delta}{\delta\beta_m}\left(a_{mm}\beta_m^2\right) \\ &= \left(\sum_{\substack{i=0\\ i\neq m}}^{p}a_{mi}\beta_i\right) + \left(\sum_{\substack{j=0\\ j\neq m}}^{p}a_{jm}\beta_j\right) + 2a_{mm}\beta_m \\ &= \left(\sum_{i=0}^{p}a_{mi}\beta_i\right) + \left(\sum_{j=0}^{p}a_{jm}\beta_j\right) \\ &= \left(\sum_{i=0}^{p}a_{mi}\beta_i\right) + \left(\sum_{j=0}^{p}a_{mj}\beta_j\right) \\ &= 2\sum_{i=0}^{p}a_{mi}\beta_i = 2\mathbf{a}_m^T\boldsymbol{\beta}. \end{aligned} \tag{A.8}$$

Therefore, Equation A.8 leads to

$$\frac{\delta}{\delta\boldsymbol{\beta}}\left(\boldsymbol{\beta}^T\mathbf{A}\boldsymbol{\beta}\right) = 2\begin{pmatrix} \mathbf{a}_0^T\boldsymbol{\beta} \\ \mathbf{a}_1^T\boldsymbol{\beta} \\ \vdots \\ \mathbf{a}_p^T\boldsymbol{\beta} \end{pmatrix} = 2\begin{pmatrix} \mathbf{a}_0^T \\ \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_p^T \end{pmatrix}\boldsymbol{\beta} = 2\mathbf{A}\boldsymbol{\beta} = 2\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta}. \tag{A.9}$$

Finally, the differentiation of the complete residual sum of squares is obtained by resubstituting Equation A.3, Equation A.6 and Equation A.9 into Equation A.2:

$$\frac{\delta RSS\left(\boldsymbol{\beta}\right)}{\delta\boldsymbol{\beta}} = -2\mathbf{X}^T\mathbf{y} + 2\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\beta}. \tag{A.10}$$

A.2 Derivations from the Gauss-Markov Conditions

Using the assumptions of the Gauss-Markov conditions in Section 3.4.2, further useful relations can be derived, that will be used later. The derivations are partly based on [SS90, p. 35].

As the variance of a random variable v is defined as $\operatorname{var}(v) := E\left[(v - E(v))^2\right]$, the error variance is

$$\begin{aligned} \operatorname{var}(\varepsilon_i) &= E\left[(\varepsilon_i - E(\varepsilon_i))^2\right] \\ &= E\left[\varepsilon_i^2\right] \\ &= \sigma^2. \end{aligned} \tag{A.11}$$

Furthermore, given a random vector $\mathbf{v} \in \mathbb{R}^k$, its expectation is defined as $E(\mathbf{v}) := (E(v_1), \ldots, E(v_k))^T$, and its covariance as $\operatorname{cov}(\mathbf{v}) := E\left[(\mathbf{v} - E(\mathbf{v}))(\mathbf{v} - E(\mathbf{v}))^T\right]$. This leads to

$$E(\boldsymbol{\varepsilon}) = \begin{pmatrix} E(\varepsilon_1) \\ \vdots \\ E(\varepsilon_n) \end{pmatrix} = \mathbf{0}, \tag{A.12}$$

as well as

$$\begin{aligned} \operatorname{cov}(\boldsymbol{\varepsilon}) &= E\left[(\boldsymbol{\varepsilon} - E(\boldsymbol{\varepsilon}))(\boldsymbol{\varepsilon} - E(\boldsymbol{\varepsilon}))^T\right] \\ &= E\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T\right] \\ &= \mathbf{I}\sigma^2. \end{aligned} \tag{A.13}$$

For each observation, y_i is dependent on ε_i it therefore is itself a random variable and $\mathbf{y}$ a random vector respectively [SS90, p. 6]. Hence, it follows that

$$\begin{aligned} E(\mathbf{y}) &= E(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}) \\ &= \mathbf{X}\boldsymbol{\beta} \end{aligned} \tag{A.14}$$

and

$$\begin{aligned} \operatorname{cov}(\mathbf{y}) &= E\left[(\mathbf{y} - E(\mathbf{y}))(\mathbf{y} - E(\mathbf{y}))^T\right] \\ &= E\left[(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right] \\ &= E\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T\right] \\ &= \mathbf{I}\sigma^2. \end{aligned} \tag{A.15}$$

A.3 Variance Estimation from the Residual Sum of Squares

In oder to obtain the relation between the error variance σ^2 and the residual sum of squares, it is necessary to express the residuals in terms of the error. Again, [SS90, pp. 31–37] has given the key ideas to this derivation.

Since a single residual is given by $e_i = y_i - \hat{y}_i$, the residual vector $\mathbf{e} \coloneqq (e_1, \ldots, e_n)$ is

$$\begin{aligned} \mathbf{e} &= \mathbf{y} - \hat{\mathbf{y}} \\ &= \mathbf{y} - \mathbf{X}\mathbf{b} \\ &= \mathbf{y} - \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{y} - \mathbf{H}\mathbf{y} \end{aligned} \tag{A.16}$$

with

$$\hat{\mathbf{y}} \coloneqq (\hat{y}_1, \ldots, \hat{y}_n)^T, \tag{A.17}$$

$$\mathbf{H} \coloneqq \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T. \tag{A.18}$$

By further defining

$$\mathbf{M} \coloneqq \mathbf{I} - \mathbf{H}, \tag{A.19}$$

some useful properties of the matrices $\mathbf{M}$ and $\mathbf{H}$ can be shown. First, it can be noticed that

$$\begin{aligned} \mathbf{M}\mathbf{X} &= \mathbf{I}\mathbf{X} - \mathbf{H}\mathbf{X} \\ &= \mathbf{I}\mathbf{X} - \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X} \\ &= \mathbf{I}\mathbf{X} - \mathbf{X}\mathbf{I} \\ &= \mathbf{0}. \end{aligned} \tag{A.20}$$

It is helpful to know that for a symmetric matrix $\mathbf{A}$ with an inverse $\mathbf{A}^{-1}$, the relations

$$\begin{aligned} \mathbf{A}\left(\mathbf{A}^{-1}\right)^T &= \left(\mathbf{A}^{-1}\mathbf{A}^T\right)^T \\ &= \left(\mathbf{A}^{-1}\mathbf{A}\right)^T \\ &= \mathbf{I}, \end{aligned} \tag{A.21}$$

and

$$\begin{aligned}
\mathbf{A}\left(\mathbf{A}^{-1}\right)^T &= \mathbf{I}, \\
\mathbf{A}^{-1}\mathbf{A}\left(\mathbf{A}^{-1}\right)^T &= \mathbf{A}^{-1}\mathbf{I}, \\
\mathbf{I}\left(\mathbf{A}^{-1}\right)^T &= \mathbf{A}^{-1}, \\
\left(\mathbf{A}^{-1}\right)^T &= \mathbf{A}^{-1},
\end{aligned} \tag{A.22}$$

can be used. Second, both $\mathbf{M}$ and $\mathbf{H}$ are idempotent matrices:

$$\begin{aligned}
\mathbf{H}\mathbf{H}^T &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\left[\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\right]^T \\
&= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X}\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right]^T\mathbf{X}^T \\
&= \mathbf{X}\mathbf{I}\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right]^T\mathbf{X}^T \\
&= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \\
&= \mathbf{H},
\end{aligned} \tag{A.23}$$

since $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ applies to Equation A.22, and

$$\begin{aligned}
\mathbf{M}\mathbf{M}^T &= (\mathbf{I}-\mathbf{H})(\mathbf{I}-\mathbf{H})^T \\
&= \mathbf{I}\mathbf{I}^T - \mathbf{H}\mathbf{I}^T - \mathbf{I}\mathbf{H}^T + \mathbf{H}\mathbf{H}^T \\
&= \mathbf{I} - \mathbf{H} - \mathbf{H}^T + \mathbf{H} \\
&= \mathbf{I} - \mathbf{H}^T, \\
\left(\mathbf{M}\mathbf{M}^T\right)^T &= \left(\mathbf{I}-\mathbf{H}^T\right)^T, \\
\mathbf{M}\mathbf{M}^T &= \mathbf{I}^T - \mathbf{H} \\
&= \mathbf{M}
\end{aligned} \tag{A.24}$$

The third property, concerning the trace of $\mathbf{M}$ is

$$\begin{aligned}
\text{trace}(\mathbf{M}) &= \text{trace}\left(\underbrace{\mathbf{I}-\mathbf{H}}_{\in\ n\times n}\right) \\
&= \text{trace}(\mathbf{I}_n)\text{-trace}(\mathbf{H}) \\
&= n-p-1,
\end{aligned} \tag{A.25}$$

because of

$$\begin{aligned}
\text{trace}(\mathbf{H}) &= \text{trace}\left(\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\right) \\
&= \text{trace}\left(\underbrace{\mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}}_{\in p+1\times p+1}\right) \\
&= \text{trace}\left(\mathbf{I}_{p+1}\right) \\
&= p+1. \qquad \text{(A.26)}
\end{aligned}$$

Using Equation A.20, the residual gets simplified to

$$\begin{aligned}
\mathbf{e} &= \mathbf{y} - \mathbf{H}\mathbf{y} \\
&= \mathbf{M}\mathbf{y} \\
&= \mathbf{M}\mathbf{X}\boldsymbol{\beta} + \mathbf{M}\boldsymbol{\varepsilon} \\
&= \mathbf{M}\boldsymbol{\varepsilon}. \qquad \text{(A.27)}
\end{aligned}$$

Now, looking at the residual sum of squares, Equation A.24 and Equation A.27 can be used to obtain:

$$\begin{aligned}
RSS\left(\mathbf{b}\right) &= \sum_{i=1}^{n} e_i^2 \\
&= \mathbf{e}^T\mathbf{e} \\
&= \boldsymbol{\varepsilon}^T\mathbf{M}^T\mathbf{M}\boldsymbol{\varepsilon} \\
&= \boldsymbol{\varepsilon}^T\mathbf{M}\boldsymbol{\varepsilon} \\
&= \left(\sum_{\substack{i=1\\ i\neq j}}^{n}\sum_{j=1}^{n} m_{ij}\varepsilon_i\varepsilon_j\right) + \left(\sum_{i=1}^{n} m_{ii}\varepsilon_i^2\right). \qquad \text{(A.28)}
\end{aligned}$$

As Equation A.28 depends on $\boldsymbol{\varepsilon}$, it is a random variable itself with expected value

$$\begin{aligned}
E\left(RSS\left(\mathbf{b}\right)\right) &= E\left(\sum_{i=1}^{n} e_i^2\right) \\
&= E\left(\sum_{\substack{i=1\\ i\neq j}}^{n}\sum_{j=1}^{n} m_{ij}\varepsilon_i\varepsilon_j\right) + E\left(\sum_{i=1}^{n} m_{ii}\varepsilon_i^2\right) \\
&= 0 + \sum_{i=1}^{n} m_{ii}E\left(\varepsilon_i^2\right) \\
&= \text{trace}\left(\mathbf{M}\right)\sigma^2 \\
&= \left(n-p-1\right)\sigma^2, \qquad \text{(A.29)}
\end{aligned}$$

using Equation A.25 and the Gauss-Markov conditions.

A.4 Variance Estimation for Model Parameters and Predictions

For the variance of the model parameter estimation $\mathbf{b}$, first $\mathbf{A} := \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ is defined. Using Equation A.15 under the Gauss-Markov conditions, this leads to

$$\begin{aligned}
\operatorname{cov}(\mathbf{b}) &= \operatorname{cov}(\mathbf{Ay}) \\
&= E\left[(\mathbf{Ay} - E(\mathbf{Ay}))(\mathbf{Ay} - E(\mathbf{Ay}))^T\right] \\
&= E\left[\mathbf{A}(\mathbf{y} - E(\mathbf{y}))(\mathbf{y} - E(\mathbf{y}))^T\mathbf{A}^T\right] \\
&= \mathbf{A}E\left[(\mathbf{y} - E(\mathbf{y}))(\mathbf{y} - E(\mathbf{y}))^T\right]\mathbf{A}^T \\
&= \mathbf{A}\operatorname{cov}(\mathbf{y})\mathbf{A}^T \\
&= \mathbf{AI}\sigma^2\mathbf{A}^T \\
&= \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\left(\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\right)^T \\
&= \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{X}\left(\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right)^T \\
&= \sigma^2\mathbf{I}\left(\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right)^T \\
&= \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}, \qquad \text{(A.30)}
\end{aligned}$$

because of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$again applying to Equation A.22 .

Regarding an arbitrary set of input variables $(1, x_{k1}, x_{k2}, \ldots, x_{kp}) =: \mathbf{x}_k$ and its corresponding prediction $\hat{y}_k = \mathbf{x}_k^T\mathbf{b}$, the prediction expectation and variance are

$$\begin{aligned}
E(\hat{y}_k) &= E\left(\mathbf{x}_k^T\mathbf{b}\right) \\
&= \mathbf{x}_k^T\boldsymbol{\beta}, \qquad \text{(A.31)}
\end{aligned}$$

and

$$\begin{aligned}
\operatorname{var}(\hat{y}_k) &= E\left[(\hat{y}_k - E(\hat{y}_k))^2\right] \\
&= E\left[\left(\mathbf{x}_k^T\mathbf{b} - \mathbf{x}_k^T\boldsymbol{\beta}\right)^2\right] \\
&= E\left[\left(\mathbf{x}_k^T(\mathbf{b} - \boldsymbol{\beta})\right)^2\right] \\
&= E\left[\left(\mathbf{x}_k^T(\mathbf{b} - \boldsymbol{\beta})\right)\left(\mathbf{x}_k^T(\mathbf{b} - \boldsymbol{\beta})\right)^T\right] \\
&= E\left[\mathbf{x}_k^T(\mathbf{b} - \boldsymbol{\beta})(\mathbf{b} - \boldsymbol{\beta})^T\mathbf{x}_k\right] \\
&= \mathbf{x}_k^T E\left((\mathbf{b} - \boldsymbol{\beta})(\mathbf{b} - \boldsymbol{\beta})^T\right)\mathbf{x}_k \\
&= \mathbf{x}_k^T\operatorname{cov}(\mathbf{b})\mathbf{x}_k, \qquad \text{(A.32)}
\end{aligned}$$

using Equation 3.24 again.

A.5 Entropy of an normally distributed univariate Random Variable

In order to calculate the differential Shannon entropy of a normally distributed univariate random variable y, the properties of the normal distribution can be used to simplify the related equations. The following derivations are based on [Sha48].

Due to $p(y)$ being a probability distribution, it is apparent that

$$1 = \int_{y\in\mathbb{R}} p(y)\, dy. \tag{A.33}$$

Because y is normally distributed, its variance σ_y^2 can be stated as

$$\sigma_y^2 = \int_{y\in\mathbb{R}} (y-\mu)^2\, p(y)\, dy, \tag{A.34}$$

where μ is the expectation of y [Sha48, p. 32]. Now, the probability density function for y is defined as

$$p(y) = \frac{1}{\sqrt{2\pi}\sigma_y} \exp\left(-\frac{(y-\mu)^2}{2\sigma_y^2}\right). \tag{A.35}$$

This leads to the entropy of S as

$$\begin{aligned}
H(\mathcal{S}) &= -\int_{y\in\mathbb{R}} p(y) \log p(y)\, dy \\
&= -\int_{y\in\mathbb{R}} p(y) \log\left(\frac{1}{\sqrt{2\pi}\sigma_y} \exp\left(-\frac{(y-\mu)^2}{2\sigma_y^2}\right)\right) dy \\
&= -\int_{y\in\mathbb{R}} p(y) \left(\log\frac{1}{\sqrt{2\pi}\sigma_y} - \frac{(y-\mu)^2}{2\sigma_y^2}\right) dy \\
&= -\int_{y\in\mathbb{R}} p(y) \log\left(\frac{1}{\sqrt{2\pi}\sigma_y}\right) dy - \int_{y\in\mathbb{R}} p(y)\, \frac{(y-\mu)^2}{-2\sigma_y^2} dy \\
&= -\log\left(\frac{1}{\sqrt{2\pi}\sigma_y}\right) \underbrace{\int_{y\in\mathbb{R}} p(y)\, dy}_{=1} + \frac{1}{2\sigma_y^2} \underbrace{\int_{y\in\mathbb{R}} (y-\mu)^2\, p(y)\, dy}_{=\sigma_y^2} \\
&= \log\left(\sqrt{2\pi}\sigma_y\right) + \frac{1}{2} \\
&= \log\left(\sqrt{2\pi}\sigma_y\right) + \log\left(e^{\frac{1}{2}}\right) \\
&= \log\left(\sqrt{2\pi e}\sigma_y\right) \\
&= \frac{1}{2}\log\left(2\pi e\sigma_y^2\right)
\end{aligned} \tag{A.36}$$

A.6 Incremental solutions for constant Regression

Let $\mathcal{S}_p$ be a set of p data samples x_i with annotated response value y_i and $p > 0$. Considering a constant regression model based on $\mathcal{S}_p$ is available and described by model parameter b_p and estimated error variance s_p^2. Let x_{p+1} be a data sample with $x_{p+1} \notin \mathcal{S}_p$. By defining $\mathcal{S}_{p+1} := \mathcal{S}_p \cup x_{p+1}$, a constant regression model based on $\mathcal{S}_{p+1}$ is defined by parameters b_{p+1} and s_{p+1}^2. With the initial regression model, these parameters can be calculated as follows:

Using b_p, the model parameter b_{p+1} can be expressed as

$$\begin{aligned} b_{p+1} &= \frac{1}{p+1}\sum_{i=1}^{p+1} y_i \\ &= \frac{1}{p+1}\sum_{i=1}^{p} y_i + \frac{y_{p+1}}{p+1} \\ &= \frac{p}{p\,(p+1)}\sum_{i=1}^{p} y_i + \frac{y_{p+1}}{p+1} \\ &= \frac{1}{p+1}\left(b_p p + y_{p+1}\right). \end{aligned} \tag{A.37}$$

In order to express s_{p+1}^2 in terms of s_p^2, some helpful transformations are presented first. The idea for these transformations is taken from [Spä83]. For any $x_k \in \mathcal{S}_{p+1}$,

$$\begin{aligned} y_k - b_{p+1} &= y_k - b_p + b_p - b_{p+1} \\ &= (y_k - b_p) + \frac{1}{p}\sum_{i=1}^{p} y_i - \frac{1}{p+1}\sum_{i=1}^{p+1} y_i \\ &= (y_k - b_p) + \left(\frac{b_p}{p} - \frac{b_p}{p+1} - \frac{y_{p+1}}{p+1}\right) \\ &= (y_k - b_p) + \frac{1}{p+1}\left(b_p - y_{p+1}\right). \end{aligned} \tag{A.38}$$

Therefore,

$$\begin{aligned} \left(y_k - b_{p+1}\right)^2 &= \left(y_k - b_p\right)^2 + \frac{2}{p+1}\left(y_k - b_p\right)\left(b_p - y_{p+1}\right) + \\ &+ \frac{1}{(p+1)^2}\left(b_p - y_{p+1}\right)^2. \end{aligned} \tag{A.39}$$

Because of

$$(p-1)\,s_p^2 = \sum_{i=1}^{p}\left(y_i - b_p\right)^2, \tag{A.40}$$

it finally follows that

$$\begin{aligned}
ps_{p+1}^2 &= \sum_{i=1}^{p+1} (y_i - b_{p+1})^2 \\
&= \sum_{i=1}^{p+1} (y_i - b_p)^2 + \frac{2}{p+1} \sum_{i=1}^{p+1} (y_i - b_p)(b_p - y_{p+1}) + \frac{1}{(p+1)^2} \sum_{i=1}^{p+1} (b_p - y_{p+1})^2 \\
&= (p-1)\, s_p^2 + (y_{p+1} - b_p)^2 + \frac{2}{p+1} (b_p - y_{p+1}) \underbrace{\sum_{i=1}^{p} (y_i - b_p)}_{=0} + \\
&\quad + \frac{2}{p+1} (b_p - y_{p+1})(y_{p+1} - b_p) + \frac{p+1}{(p+1)^2} (b_p - y_{p+1})^2 \\
&= (p-1)\, s_p^2 + (y_{p+1} - b_p)^2 - \frac{2}{p+1} (y_{p+1} - b_p)^2 + \frac{1}{p+1} (y_{p+1} - b_p)^2 \\
&= (p-1)\, s_p^2 + \frac{p}{p+1} (y_{p+1} - b_p)^2 . \qquad \text{(A.41)}
\end{aligned}$$

Hence, the estimated error variance of the new regression model is given by

$$s_{p+1}^2 = \frac{p-1}{p} s_p^2 + \frac{1}{p+1} (y_{p+1} - b_p)^2 . \qquad \text{(A.42)}$$

In the case of $p > 2$, the same can be done to calculate the parameters of a constant regression model based on $\mathcal{S}_{p-1} := \mathcal{S}_p \setminus x_p$. Due to the same derivations as above, the solution stays defined as

$$b_{p-1} = \frac{1}{p-1} (b_p p - y_p) \qquad \text{(A.43)}$$

and

$$s_{p-1}^2 = \frac{p-1}{p-2} s_p^2 - \frac{p}{(p-1)(p-2)} (y_p - b_p)^2 . \qquad \text{(A.44)}$$

Acknowledgements

First of all, I want to thank my reviewer Prof. Rainer Lienhart for the possibility to write my bachelor thesis at the Multimedia Computing and Computer Vision Lab.

Secondly, I am very grateful to my supervisor Christoph Lassner for his patience, his helpful remarks and the enriching discussions.

Many thanks to my brother and sister for their feedback and the improvements they pointed out and my family in total for their support. Special thanks go to Alexandra Feß for her encouragement during the last months.

Bibliography

[AG89] N. A. Ahmed and D. V. Gokhale. Entropy expressions and their estimators for multivariate distributions. *Information Theory, IEEE Transactions on*, 35(3): 688–692, May 1989. (Cited on page 32.)

[BFSO84] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984. (Cited on pages 1, 5, 10, 15 and 46.)

[Bjö96] A. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. (Cited on page 37.)

[BKW80] D. A. Belsley, E. Kuh, and R.E. Welsch. *Regression diagnostics: identifying influential data and sources of collinearity*. Wiley series in probability and mathematical statistics. Wiley, Belmont, CA, USA, 1980. (Cited on page 48.)

[Bre01] L. Breiman. Random forests. *Machine Learning*, 45(1): 5–32, 2001. URL: http://link.springer.com/article/10.1023%2FA%3A1010933404324. (Cited on pages 2, 14, 15, 46 and 52.)

[CS13] A. Criminisi and J. Shotton, editors. *Decision Forests for Computer Vision and Medical Image Analysis*. Advances in Computer Vision and Pattern Recognition. Springer, London, 2013. (Cited on page 38.)

[CSK12] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2-3): 81–227, 2012. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=158806. (Cited on pages 1, 2, 5, 8, 10, 11, 12, 13, 14, 15, 16, 24, 30, 31, 47, 55 and XV.)

[enc07] *The New Encyclopaedia Britannica*, volume 1. Encyclopaedia Britannica, Inc., Chicago, 15 edition, 2007. (Cited on page 51.)

[HJR78] D. Harrison Jr. and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1): 81–102, 1978. URL: http://www.law.berkeley.edu/faculty/rubinfeldd/Profile/publications/Hedonic.PDF. (Cited on pages 48 and 50.)

[HTF03] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, 2003. (Cited on pages 19, 22 and 25.)

[Mit97] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. (Cited on pages 12 and 13.)

[Pre07] A. Pressley. *Elementary differential geometry*. Springer undergraduate mathematics series. Springer, London, Dordrecht, Heidelberg, 9 edition, 2007. (Cited on page 11.)

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011. URL: http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf. (Cited on page 45.)

[Qui93] J.R. Quinlan. *C4.5: Programs for Machine Learning*. C4.5 - programs for machine learning / J. Ross Quinlan. Morgan Kaufmann Publishers, 1993. (Cited on page 1.)

[Sch90] R. E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2): 197–227, July 1990. URL: http://www.cs.princeton.edu/~schapire/papers/strengthofweak.pdf. (Cited on page 1.)

[Sha48] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, July 1948. (Cited on page VIII.)

[Sl14] Scikit-learn. Decision trees, URL: http://scikit-learn.org/stable/modules/tree.html. (Cited on page 46.)

[Spä83] H. Späth. *Cluster-Formation und -Analyse: Theorie, FORTRAN-Programme und Beispiele*. R. Oldenbourg Verlag, München, 1983. (Cited on page IX.)

[SS90] A. Sen and S. Srivastava. *Regression Analysis: Theory, Methods, and Applications*. Lecture Notes in Statistics. Springer, New York, NY, USA, 1990. (Cited on pages 21, 22, 23, 24, 25, 36, I, III and IV.)

[UR95] UCI-Repository. Abalone dataset, URL: http://archive.ics.uci.edu/ml/datasets/Abalone. (Cited on pages 52 and 54.)

[YS09] X. Yan and X. G. Su. *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2009. (Cited on pages 18, 19, 21, 22 and 33.)

List of Figures

Eidesstattliche Erklärung

Ich versichere, dass die Bachelorarbeit von mir selbständig verfasst wurde und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Zitate habe ich klar gekennzeichnet.

Augsburg, den 30. September 2014

Moritz Einfalt