OPEN ACCESS
OPUS AUGSBURG

Universität Augsburg
Universitätsbibliothek

Mathematische Grundlagen als Schlüssel zu einem allgemeinbildenden Verständnis von KI: theoretische Perspektiven und praktische Unterrichtsideen

Sarah Schönbrodt, Reinhard Oldenburg

Angaben zur Veröffentlichung / Publication details:

Schönbrodt, Sarah, and Reinhard Oldenburg. 2024. “Mathematische Grundlagen als Schlüssel zu einem allgemeinbildenden Verständnis von KI: theoretische Perspektiven und praktische Unterrichtsideen.” *Mathematik im Unterricht* 15: 17–36. https://doi.org/10.25598/miu/2024-15-2.

Nutzungsbedingungen / Terms of use: **CC BY 3.0**

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
CC-BY 3.0: Creative Commons - Namensnennung
Weitere Informationen finden Sie unter: / For more information see:
https://creativecommons.org/licenses/by/3.0/de/deed.de

Mathematische Grundlagen als Schlüssel zu einem allgemeinbildenden Verständnis von KI: Theoretische Perspektiven und praktische Unterrichtsideen

Sarah Schönbrodt, Reinhard Oldenburg

Abstract. Im aktuellen politischen Diskurs nimmt Künstliche Intelligenz (KI) eine herausragende Stellung ein. Die Erwartung, dass durch KI-Systeme die Lebens- und Arbeitsverhältnisse grundlegend transformiert werden, weist der Schule die Aufgabe zu, Lernende zu einem reflektierten, informierten Umgang mit ebendiesen Technologien und zugrundeliegenden Methoden zu befähigen. Alle verbreiteten KI-Anwendungen basieren wesentlich auf mathematischen Grundlagen und es stellt sich die Frage, inwieweit eine Beschäftigung mit diesen Grundlagen möglich und sinnvoll ist. Dies wird im Beitrag sowohl theoretisch als auch auf Basis von konkreten Ideen für den Unterricht diskutiert.

Einleitung

Es scheint breiter politischer und gesellschaftlicher Konsens zu sein, dass KI-Technologien unsere Gesellschaft in nahezu allen Lebensbereichen transformieren und Kompetenzen im Umgang mit diesen sowohl im Alltag als auch in zahlreichen Berufsfeldern immer wichtiger werden (vgl. etwa www.bundesregierung.de/breg-de/themen/digitalisierung/kuenstliche-intelligenz). Die Europäische Kommission hebt in ihrem Bericht “DigComp 2.2” hervor, dass hierfür durchaus auch ein grundlegendes technisches Verständnis von KI-Technologien erforderlich ist (European Commission, 2022, S. 77). Zudem betont die UNESCO (2024) im “AI Competency Framework for Students”, dass es heutzutage zur Aufgabe von (allgemeinbildenden) Schulen gehört, Lernende in die Lage zu versetzen, KI-Entwicklungen zu bewerten sowie deren Potenziale und Gefahren auf Basis von Wissen und eigener Kompetenz zu beurteilen. Sie sollen darauf vorbereitet werden, als verantwortungsvolle Nutzer*innen und Mitgestalter*innen von KI zu agieren (UNESCO, 2024, S. 3). Einen Beitrag dazu können vor allem die Schulfächer Mathematik und Informatik leisten, wobei in der bisherigen Diskussion vor allem das Fach Informatik als relevant erachtet wird: So ist das Thema KI u. a. in Bayern (Staatsinstitut für Schulqualität und Bildungsforschung, 2022), Nordrhein-Westfalen (Ministerium für Schule und Weiterbildung des Landes Nordrhein-Westfalen, 2021, S. 18) sowie in Österreich im Schulfach Informatik verankert. In Österreich wird KI zudem im Lehrplan der Digitalen Grundbildung benannt (Bundesministerium für Bildung, Wissenschaft und Forschung, 2024).

Der Relevanz des Informatikunterrichts für KI-Bildung soll hier zwar nicht widersprochen werden, dennoch soll in diesem Beitrag herausgearbeitet werden, welche Rolle die Mathematik für Verfahren des maschinellen Lernens spielt und welche grundlegenden Eigenschaften dieser Verfahren durch mathematische Überlegungen verstanden werden können. Die pädagogische Intention dabei ist die des “Empowerment” (Ernest, 2002), also die Förderung von Handlungskompetenz in bestimmten Tätigkeitsbereichen mit dem Ziel, Jugendliche zum aktiven Gestalten zu befähigen.[1] Dies führt dazu, dass unsere Perspektive eine stark mathematisch-technologische Dimension hat. Dies soll keine Reduktion des Bildungsanspruchs auf eine technokratische Perspektive implizieren. Vielmehr sind wir der Meinung, dass auch der gesellschaftliche Diskurs über die Auswirkungen des maschinellen Lernens von einer Demystifizierung der Technik profitiert und insbesondere den in diversen Gesellschaften vorhandenen Ängsten vor KI (Sindermann et al., 2022) durch Aufklärung entgegenwirken kann. Gleichzeitig kann ein solcher Unterricht die fundamentale Rolle der Mathematik[2] für moderne, relevante

[1] Ernest (2002) klassifiziert drei Arten des Empowerments: “mathematical empowerment” (Mathematik und ihre Sprache nutzen und anwenden können), "social empowerment" (die Nutzung mathematischer Kompetenzen zur Verbesserung der sozialen Situation) und "epistemological empowerment" (Korrektheit und Gültigkeit von Wissen einschätzen können). Alle drei Arten können durch die Beschäftigung mit den von uns behandelten Themen gefördert werden.

[2] Diese Rolle wird nunmehr auch von den mathematischen Fachgesellschaften in ihrer Stellungnahme “Warum ist Mathematik für Künstliche Intelligenz unentbehrlich?” betont, www.mathematik.de/dmv-blog/5090-warum-ist-mathematik-f%C3%BCr-k%C3%BCnstliche-intelligenz-unentbehrlich. Zugegriffen: 14.08.2024.

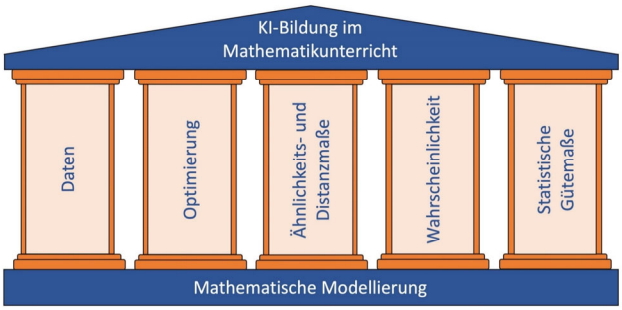
Technologien aus dem Bereich KI ersichtlich machen. Dass es möglich ist, die komplexen mathematischen Hintergründe von KI-Systemen auf Schulniveau zugänglich zu machen, wird in verschiedenen Veröffentlichungen an exemplarischen maschinellen Lernmethoden herausgearbeitet (vgl. z. B. Biehler & Fleischer, 2021; Hazzan & Mike, 2021; Schönbrodt et al., 2022; Kindler et al., 2023; Schönbrodt et al., 2023).

Der Begriff *Künstliche Intelligenz* wird in der Literatur sehr breit gefasst – eine einheitliche Definition sucht man vergeblich. Im Wesentlichen bezeichnen wir KI hier als Oberbegriff für alle Technologien, die Computersysteme in die Lage versetzen, eine Aufgabe bzw. ein Problem zu lösen, dessen Lösung gemeinhin mit Fähigkeiten von „intelligenten“ Menschen assoziiert wird und deren Beherrschung durch Maschinen überraschend ist. Früher wurden auch klassische Schachcomputer oder Computeralgebrasysteme der KI zugerechnet – in einer modernen Sichtweise ist das nicht mehr so, da diese Probleme rein algorithmisch gelöst werden können und die Algorithmen explizit programmiert werden.

Etwas greifbarer wird es, wenn man unter die Haube von modernen KI-Systemen schaut und die zugrunde liegenden (mathematischen) Methoden aus dem Bereich des maschinellen Lernens (ML), zu denen u. a. künstliche neuronale Netze gehören, in den Blick nimmt. Genau darauf legen wir in diesem Beitrag den Fokus. Das ML beruht ganz wesentlich auf Mathematik und Daten. Es umfasst eine Vielzahl an Verfahren, darunter solche aus dem Bereich des *überwachten Lernens* (supervised learning), bei dem auf einen Fundus richtiger Beispiele zurückgegriffen wird, und aus dem *unüberwachten Lernen* (unsupervised learning), bei dem Daten bspw. durch Clustering organisiert werden. Der Schwerpunkt dieses Beitrags liegt auf Methoden des überwachten Lernens.

Wenn Lernenden ein komplexes Thema wie die Funktionsweise von ML-Methoden nahegebracht werden soll, stellt sich die Frage des richtigen Erklärungsmaßstabs. Eine Analogie soll das erläutern: Wer das Leben mit biologischen Begriffen verstehen will, kann bei der Zellchemie anfangen, das Zusammenwirken der Organe in einem Organismus studieren, oder die evolutionären Mechanismen in ganzen Populationen. Das Phänomen des Lebens kann also auf unterschiedlichen Maßstäben verstanden werden und alle haben offensichtlich ihre Berechtigung. Übertragen auf das Gebiet der KI stellt sich die Frage, wie wichtig ein Verständnis von KI-Systemen und den zugrundeliegenden ML-Methoden auf unterschiedlichen Maßstäben ist. Rahwan et al. (2019) argumentieren, dass eine solche mehrstufige Herangehensweise auch einen geeigneten Ansatz für das Verständnis von Maschinen liefert. In diesem Beitrag werden zentrale mathematische Säulen von ML-Methoden (Daten, Optimierung, Messung von Ähnlichkeit und Distanzen, Wahrscheinlichkeit und statistische Gütemaße, s. Abb. 1) erläutert und drei verschiedenen Erklärungsmaßstäben bzw. Ebenen zugeordnet: der *Mikroebene* einzelner elementarer mathematischer Objekte und Operationen, der *Mesoebene*, in der es um das Zusammenspiel der elementaren mathematischen Komponenten geht und der *Makroebene*, auf der das gesamte System und dessen Einbettung in die Gesellschaft betrachtet wird. Die drei Ebenen werden im Abschnitt “Didaktische Einordnung” detaillierter aufgeschlüsselt und es wird diskutiert, welchen spezifischen Bildungsbeitrag unterrichtliche Zugänge auf den drei Ebenen leisten können.

Abb. 1: Mathematische Säulen zahlreicher maschineller Lernmethoden



Die fünf (sicherlich nicht abschließenden) mathematischen Säulen (siehe Abb. 1) werden im Folgenden in unterschiedlicher Tiefe beschrieben. Als mathematische Säulen wurden Konzepte bzw. Themengebiete ausgewählt, die bei der Entwicklung von KI-Systemen basierend auf verschiedenen ML-Methoden eine zentrale Rolle spielen. Besonders detailliert gehen wir auf die Säule “Optimierung” ein. Diese hat in der didaktischen Literatur rund um die Vermittlung von ML-Methoden bisher weniger Beachtung gefunden, erscheint uns jedoch als sehr relevant und durchaus zugänglich. Die mathematische Modellierung bildet in Abbildung 1 das Fundament. Sie spielt eine tragende, säulenübergreifende Rolle. Letztlich kann der Entwicklungsprozess von KI-

Systemen, die auf ML-Methoden basieren, als (datenbasierte) mathematische Modellierung verstanden werden.

Um die mathematischen Grundlagen von ML-Methoden im Unterricht nicht nur theoretisch zu thematisieren, sondern computergestützt auch umzusetzen, finden sich unter https://github.com/Schoenbrodt/KI-Bildung-im-MU diverse Umsetzungsbeispiele in Form von Jupyter Notebooks[3] basierend auf der Programmiersprache Python. Dies ermöglicht es Lernenden, ihr Verständnis durch computergestützte Anwendung und Variation der im Folgenden vorgestellten mathematischen Methoden zu überprüfen und den vorhandenen Code als Basis für eigene Anwendungen zu nutzen (und somit den Zielen des Empowerments gerecht zu werden). Eine solche aktive Auseinandersetzung scheint gerade bei einem so tiefgründigen und mathematisch facettenreichen Thema wichtig.

Säule 1: Daten

In der Literatur wird das maschinelle Lernen häufig als Entwicklung von Computerprogrammen oder Algorithmen beschrieben, die „aus Erfahrung" (Mitchell, 1995, S. 2) bzw. „aus Daten" lernen. Diese Beschreibung ist aus unserer Sicht nicht ideal (da mystisch und mathematisch unpräzise), macht aber zumindest deutlich, dass die Basis vieler ML-Methoden (zahlreiche!) Daten sind. Der Umgang mit Daten und die Analyse und Vorverarbeitung der Daten mit mathematischen (insbesondere statistischen) Methoden spielt im Kontext des MLs in vielerlei Hinsicht eine wichtige Rolle. Dazu ließe sich einerseits ein ganzer Beitrag füllen, andererseits gibt es aus der Forschung zu Data Science Education und Data Literacy bereits zahlreiche Publikationen und Vorschläge für die unterrichtspraktische Umsetzung (vgl. z. B. Gould et al., 2016; Gould, 2021; Engel, 2017; Dvir et al., 2022). Wir umreißen daher lediglich verschiedene Teilfragen und gehen auf ausgewählte mathematische Aspekte detaillierter ein, die eng mit den weiteren mathematischen Säulen verzahnt sind und interessante Anregungen für die unterrichtliche Diskussion der mathematischen Aspekte von ML-Methoden liefern.

Ausgangspunkt für die Entwicklung von KI-Systemen sind in der Regel reale Problemstellungen, zum Beispiel:

- Wie können Bilder von Gesichtern korrekt den jeweiligen Personen zugeordnet werden?
- Wie können Fitness-Tracker menschliche Aktivitäten möglichst genau erkennen?
- Wie kann das Risiko, an einer Herzkrankheit zu erkranken, möglichst präzise vorhergesagt werden?

Zur Beantwortung solcher Fragen mit ML-Methoden werden vergangene Daten verwendet. Beim überwachten Lernen bestehen diese aus Inputdaten (auch Eingabedaten oder Werte von Prädiktorvariablen)[4] $\boldsymbol{x}_i \in \mathbb{R}^n$ und zugehörigen Outputdaten (auch Ausgabedaten oder Werte der Zielvariablen) $\boldsymbol{y}_i \in \mathbb{R}^m$ für $i = 1, \ldots, N$. Am Beispiel der menschlichen Aktivitätserkennung können die Inputdaten bspw. aus der *Zeit*, der *Herzfrequenz* und der *Beschleunigung* (d. h. $\boldsymbol{x}_i \in \mathbb{R}^3$) bestehen. Die zugehörigen Outputdaten können die Klassen 1 (= Laufen), 2 (= Gehen), 3 (= Treppensteigen) und 4 (= Sitzen) sein, d. h. $m = 1, y_i \in \{1,2,3,4\}$.

Basierend auf den bekannten Daten wird ein mathematisches Modell (meist eine Funktion $f: \mathbb{R}^n \to \mathbb{R}^m$) entwickelt, das Zusammenhänge zwischen den bekannten Input- und Outputdaten möglichst präzise beschreiben soll. Ziel ist es, das entwickelte mathematische Modell zu nutzen, um auch für neue, bisher ungesehenen Inputdaten $\boldsymbol{x}$ eine möglichst gute Vorhersage für den zugehörigen Output $\mathbf{y}$ berechnen zu können. Um validieren zu können, wie gut die Vorhersagen des Modells sind, werden die bekannten Daten in *Trainings-* und *Testdaten* unterteilt. Die Trainingsdaten werden zur Entwicklung des mathematischen Modells verwendet – in der KI-Sprechweise zum "Lernen bzw. Trainieren des Modells". Die Testdaten dienen zur Validierung. Dazu werden mithilfe des mathematischen Modells die Outputs für alle Inputdaten des Testdatensatzes vorhergesagt. Die Vorhersagen können dann mit den tatsächlichen Outputs verglichen werden. Dazu kommen statistische Gütemaße zum Einsatz (vgl. Abschn. Statistische Gütemaße).

Um reale Fragestellungen mit datenbasierten ML-Methoden zu lösen, sind schon vor der eigentlichen Entwicklung der Methode diverse Fragen im Hinblick auf die Verwendung und Verarbeitung der Daten relevant.

[3] Im folgenden Text wird an verschiedenen Stellen auf die Jupyter Notebooks hingewiesen, indem ihr Dateiname angegeben wird (mit der Endung *.ipynb*). Alle Notebooks sind unter dem oben angegebenen Link verfügbar.
[4] In diesem Beitrag bezeichnet $\boldsymbol{x}_i \in \mathbb{R}^n$ (fett gedruckt) den i-ten Datenvektor. Der j-te Eintrag des Vektors wird mit $x_j \in \mathbb{R}$ bezeichnet.

Unter anderem: Welche Informationen bzw. Daten benötige ich? Wie kann ich diese beschaffen? Muss ich die entsprechenden Daten zunächst selbst erheben oder aufnehmen oder gibt es bereits frei verfügbare Datensätze? Wie stelle ich die Daten in geeigneter Weise dar? Welche (mathematischen) Repräsentationen sind dafür geeignet? Welche Erkenntnisse über die Verteilung und Qualität der Daten liefert eine erste Erkundung, u. a. durch Visualisierung der Daten? Sind die Daten fehlerbehaftet? Gibt es Ausreißer? Wie gehe ich mit fehlenden Daten um? Sind meine Daten divers und repräsentativ genug, um das gegebene Problem möglichst robust zu lösen? Beinhalten die vergangenen Daten womöglich statistische Verzerrungen (Bias)? Kann es passieren, dass statistische Verzerrungen in den vergangenen Daten durch Anwendung prädiktiver Modelle in die Zukunft fortgeschrieben werden? Kann es damit womöglich zu Diskriminierungen kommen? Welche ethischen und gesetzlichen Regelungen, bspw. zu Datenschutz, Privatsphäre oder geschützten Merkmalen von Personen (u. a. das Geschlecht), müssen im Hinblick auf die Verwendung der Daten beachtet werden?

Die Diskussion von Daten als Ausgangspunkt für die Entwicklung von KI-Systemen ist im Unterricht auf unterschiedlichen Ebenen möglich. Auf der *Mikroebene* können bspw. detaillierte Analysen der statistischen Eigenschaften des Datensatzes durchgeführt oder statistischen Methoden zur Ausreißeridentifikation diskutiert und angewendet werden. Auch die Kodierung von Daten liegt auf dieser Ebene. Auf der *Mesoebene* können Daten aggregiert und etwa mit Streu- oder Lagemaßen charakterisiert werden. Auf der *Makroebene* können der Einfluss verschiedener Trainingsdatensätze auf die Ergebnisse eines ML-Modells erkundet und ethisch-gesellschaftliche sowie gesetzliche Fragen im Umgang mit Daten diskutiert werden.

Säule 2: Optimierung

Intelligenz wird oft verstanden als Anpassungsleistung – so etwa William Stern (1911), der Intelligenz als Fähigkeit zur Anpassung an unbekannte Situationen definierte. In gewissem – wenn auch anderem – Sinne gilt das auch für Systeme, die als *künstliche Intelligenz* bezeichnet werden. Diese Anpassung kann als Optimierung verstanden werden – die Diskrepanz zwischen einem berechneten und einem gewünschten Ergebnis (bekannte Outputdaten) wird verringert. Diese Sichtweise wird hier als Leitlinie verwendet, um ausgehend von einfachen, schulüblichen Extremwertaufgaben bis zu künstlichen neuronalen Netzen voranzuschreiten.

Minimierung von Funktionswerten

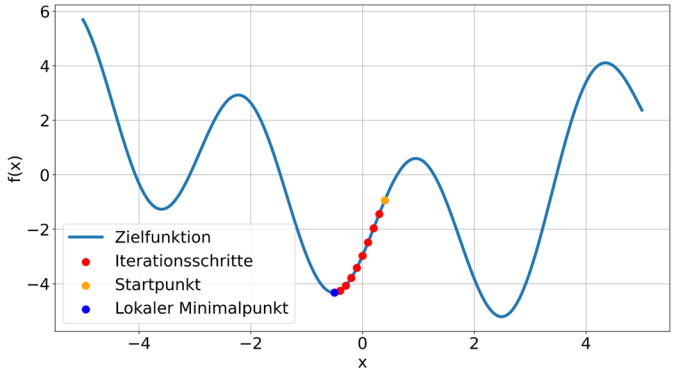
Die in der Schule dominierende Methode zur Bestimmung von Extremstellen von Funktionen $\mathbb{R} \to \mathbb{R}$ ist das Ableitungskalkül. Numerische Verfahren kommen aber – in den einfachsten Versionen – mit viel weniger Theorie aus. Ein Minimum einer Funktion $f: \mathbb{R} \to \mathbb{R}$ zu finden, ist dann ein iterativer Suchprozess: Ausgehend von einem Startwert x_0 wird der Funktionswert $f(x_0)$ mit dem Wert $f(x_0 + \delta)$ an der Stelle $x_0 + \delta$ verglichen. Falls $f(x_0 + \delta) < f(x_0)$ wird im nachfolgenden Schritt die Umgebung der Stelle $x_1 := x_0 + \delta$ betrachtet. Andernfalls wird untersucht, ob die Funktionswerte nach links kleiner werden. Gilt also $f(x_0 - \delta) < f(x_0)$ wird die Stelle zu $x_1 := x_0 - \delta$ geändert. Dies wird so lange fortgeführt, bis ein gewähltes Abbruchkriterium erfüllt ist. Mögliche Abbruchkriterien sind, dass eine maximale Anzahl an Iterationsschritten erreicht wurde oder die Änderung der Funktionswerte unter einen gewissen Schwellenwert ϵ fällt. Die konkrete Umsetzung in Python erfordert nur wenige Programmzeilen (vgl. *numerische_Optimierungsverfahren.ipynb).*

Dieser simple Algorithmus, der für Maximierungsprobleme auch als *Bergsteigeralgorithmus* bezeichnet wird, ist sehr ineffizient – es gibt viel schnellere Verfahren – aber er zeigt doch vier grundlegende Prinzipien, die er mit den meisten effizienteren Algorithmen gemein hat:

- Die Suche beginnt bei einem Startwert und es wird mit einer gewissen, ggf. dynamischen Schrittweite iterativ vorangeschritten.
- Das Verfahren endet, wenn ein vorgegebenes Abbruchkriterium erreicht wurde.
- Das Ergebnis kann sowohl vom Startwert als auch von der Schrittweite (im Bereich des MLs oft als Lernrate bezeichnet) abhängen. Da in der Regel nur ein lokales Minimum gefunden wird, haben der Startwert und die Schrittweite entscheidenden Einfluss auf das gefundene Minimum, insbesondere wenn mehrere lokale Minima existieren (Abb. 2).
- Die Ergebnisse sind in der Regel nicht exakt, sondern lediglich numerische Approximationen.

Numerische Algorithmen zur Minimierung sind ein klassisches Thema der mathematischen Forschung und die Zahl der Methoden ist unüberschaubar groß. Es gibt deutlich bessere Algorithmen, aber als mentales Modell von deren typischer Arbeitsweise reicht das beschriebene Verfahren völlig aus.

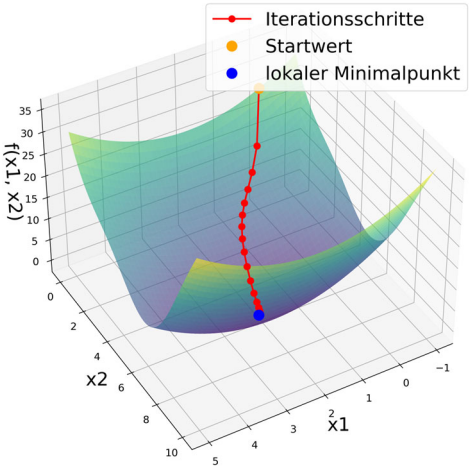
Abb. 2: Simples numerisches Optimierungsverfahren mit Startwert $x_0 = 0{,}4$, Schrittweite $\delta = 0{,}1$ und Zielfunktion f mit $f(x) = cos(x) + 3 \cdot sin(0{,}6x - 2) + 3 \cdot cos(2x - 2)$



Im Falle von Funktionen einer reellen Variablen können in der Diskussion mit Schüler*innen die Ergebnisse eines solchen Algorithmus den Berechnungen im Ableitungskalkül gegenübergestellt werden (sofern die Funktion differenzierbar ist, was der Algorithmus nicht voraussetzt).

Mehrdimensionale Funktionen: Beim ML spielen in der Regel Funktionen eine Rolle, die von mehr als einer Variable abhängen. Mit Schüler*innen kann sukzessive vom ein-, zum zwei-, zum N-dimensionalen Fall vorangeschritten werden – gestützt durch Visualisierungen in den ersten beiden Fällen. Die Bedeutung eines Minimums einer Funktion in zwei reellen Variablen lässt sich am Funktionsgraph im Dreidimensionalen deutlich machen (Abb. 3). Auch die rechnerische Vorstellung, dass eine lokale Minimalstelle $\boldsymbol{x}^* \in \mathbb{R}^2$ eine Stelle ist, an der der Funktionswert $f(\boldsymbol{x}^*)$ so klein ist, dass es – zumindest in einer Umgebung – keinen kleineren gibt, lässt sich damit aufbauen.

Abb. 3: Visualisierung des Gradientenabstiegsverfahrens an einer Zielfunktion in zwei Variablen



Eine simple Erweiterung unseres Minimierungsalgorithmus für Funktionen f auf $\mathbb{R}^2$ ist die Folgende: Man minimiert $f(x_1, x_2)$ zunächst mit dem eindimensionalen Algorithmus bzgl. x_1 (x_2 ist fest), dann wechselt man zu x_2 (bei festem x_1). Einfache Beispiele zeigen, dass es damit oft noch nicht getan ist: Man muss beide Optimierungsschritte vielfach wiederholen, bis sich das Ergebnis stabilisiert. Das Verfahren ist rechenaufwendig, aber es ist plausibel, dass Computer das für wenige Variablen leisten können. Wird dieses Verfahren jedoch zum Lösen von Minimierungsproblemen mit Millionen von Variablen eingesetzt (wie es bei künstlichen neuronalen Netzen oft der Fall ist), ist das Vorgehen nicht effizient genug. Es braucht also raffiniertere Verfahren, um möglichst schnell zu möglichst exakten Ergebnissen zu kommen. Wir diskutieren hier einen zweiten elementaren Algorithmus, um den Schüler*innen die Erfahrung zu ermöglichen, dass es sich lohnt, weiter an besseren Algorithmen zu forschen.

Ein effizienterer Algorithmus, der auch in der Praxis Einsatz findet, ist das *Verfahren des steilsten Abstiegs* (Gradientenverfahren, vgl. z. B. Deisenroth et al., 2020, Kap. 7). Anstatt immer nur in eine Koordinatenrichtung zu gehen, wird der Gradient bestimmt. Dieser gibt die Richtung des steilsten Anstiegs und entgegengesetzt die Richtung des steilsten Abstiegs an. Wenn man in diese Richtung geht, kommt man schneller zu einem lokalen Minimum als entlang der Koordinatenrichtung (Abb. 3). Für eine differenzierbare Funktion $f: \mathbb{R}^2 \to \mathbb{R}$ lautet die Iterationsvorschrift des Verfahrens $\boldsymbol{x}_{k+1} \coloneqq \boldsymbol{x}_k - \delta \cdot \nabla f(\boldsymbol{x}_k)$, wobei $\nabla f(\boldsymbol{x}_k)$ der Gradient von f an der Stelle $\boldsymbol{x}_k$ des k-ten Iterationsschritts und δ die Schrittweite ist. An diesem Verfahren können die Lernenden erkennen, dass die Differentialrechnung, die in der Schule stark mit Optimierungsproblemen in einer Variablen verbunden wird, eine große Bedeutung behält, auch wenn Computer zur approximativen Lösung benutzt werden. Wie erwähnt gibt es effizientere Verfahren, die aber das Gleiche leisten. Viele davon können in Python mit dem Paket *scipy* genutzt werden.

Es bietet sich an, Lernende computergestützt mit unterschiedlichen Zielfunktionen, Startwerten und Schrittweiten experimentieren und die oben genannten vier Prinzipien numerischer Optimierungsverfahren erkunden zu lassen. Dazu kann das Notebook *numerische_Optimierungsverfahren.ipynb* eingesetzt werden. Auch der Übergang zu Funktionen auf $\mathbb{R}^n$ kann computergestützt realisiert werden.

Allgemein betrachtet bestimmen die beschriebenen Algorithmen eine Lösung des allgemeinen Optimierungsproblems $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ mit stetiger Zielfunktion $f: \mathbb{R}^n \to \mathbb{R}$, indem sie ausgehend von einem Startwert $\boldsymbol{x}_0 \in \mathbb{R}^n$ approximativ ein lokales Minimum mit zugehöriger Minimalstelle $\boldsymbol{x}^* \in \mathbb{R}^n$ suchen – sofern eine solche existiert. Für eine lokale Minimalstelle muss $f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$ für alle $\boldsymbol{x} \in U(\boldsymbol{x}^*)$ gelten, wobei $U(\boldsymbol{x}^*)$ eine Umgebung von $\boldsymbol{x}^*$ bezeichnet. Zahlreiche ML-Methoden lösen Spezialfälle des allgemeinen Optimierungsproblems (künstliche neuronale Netze, lineare Regressionsanalyse, Support Vector Machine).

Lineare Regression

Es gibt unzählige Anwendungen von Optimierungsverfahren, hier werden nur solche aus dem Bereich des MLs behandelt. Als eines der einfachsten ML-Verfahren gilt die lineare Regressionsanalyse, deren Kern das Lösen eines Optimierungsproblems ist.

Wir betrachten zunächst ein Beispiel mit zweidimensionalen Datenpunkten $(x_i, y_i) \in \mathbb{R}^2, i = 1, \ldots, N$ (die Trainingsdaten), aus denen ein Modell entwickelt („gelernt“) werden soll, welches für weitere x-Werte einen passenden y-Wert vorhersagen kann. Wir unterstellen hier, dass es einen linearen Zusammenhang $y = mx + b$ gibt, und versuchen für die Parameter m und b die besten Werte zu schätzen. Dazu muss definiert werden, was mit “den besten” Werten gemeint ist. Dafür gibt es mehrere Strategien, die auch mit Schüler*innen erarbeitet werden können (Schönbrodt & Frank, 2024). Die verbreitetste ist die Minimierung der Summe der Fehlerquadrate (Methode der kleinsten Quadrate): Für x_i prognostiziert das Modell den Wert $\hat{y}_i = mx_i + b$. Für die Trainingsdaten ist der Wert der Zielvariablen y_i bekannt, sodass der Fehler $y_i - \hat{y}_i$ berechnet werden kann. Ziel ist es dann, die Quadratsumme der Fehler zu minimieren, also das Minimum der Zielfunktion F mit $F(m, b) \coloneqq \sum_{i=1}^{N} \big(y_i - (mx_i + b)\big)^2$ zu bestimmen. Eine computergestützte Umsetzung, die als Ausgangspunkt für die unterrichtliche Erarbeitung dienen kann, findet sich im Notebook *lineare_Regression.ipynb*.

Da das ML-Verfahren „Lineare Regressionsanalyse” nicht nur für die Trainingsdaten, sondern insbesondere für neue Daten gute Vorhersagen liefern soll, wird das Modell auf Testdaten validiert. Die Vorhersagegenauigkeit auf den Testdaten dient als Abschätzung der Vorhersagegüte für gänzlich unbekannte Daten (vgl. Abschn. Statistische Gütemaße).

Allgemeiner lassen sich Regressionsprobleme folgendermaßen beschreiben: Gegeben sind N Paare von Input- und Outputdaten $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ mit $\boldsymbol{x}_i \in \mathbb{R}^n, \boldsymbol{y}_i \in \mathbb{R}^k$ für $i = 1, \ldots, N$ (die Trainingsdaten) und eine Modellfunktion $g: \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$. In $g(\boldsymbol{p}, \boldsymbol{x})$ ist $\boldsymbol{p} \in \mathbb{R}^p$ ein Vektor von Parametern, im obigen linearen Fall also $\boldsymbol{p} = (m, b)$ und $g((m, b), x) \coloneqq mx + b$. Der Wert dieser Funktion ist die Vorhersage des Modells für die Ausgabe zur Eingabe x. Der Parametervektor $\boldsymbol{p}$ soll so gewählt werden, dass das Modell die Daten möglichst gut widerspiegelt. Dazu wird eine Zielfunktion, z. B. $F: \mathbb{R}^p \to \mathbb{R}, F(\boldsymbol{p}) \coloneqq \sum_{i=1}^{N} ||\boldsymbol{y}_i - g(\boldsymbol{p}, \boldsymbol{x}_i)||^2$ formuliert. Das “Lernen” der optimalen Parameter beruht somit auf dem Lösen eines Minimierungsproblems und kann mit den oben beschriebenen Optimierungsverfahren gelöst werden (auch wenn es spezialisierte Verfahren gibt, die effizienter sind). Diese allgemeine Beschreibung eines Regressionsproblems macht deutlich, dass es neben dem linearen Fall auch die Möglichkeit gibt, die Prognosefunktion g beliebig, also z. B. nichtlinear zu wählen. Die Nichtlinearität

kann die Prädiktorvariablen oder die Parameter $\boldsymbol{p}$ betreffen, im zweiten Fall spricht man von nichtlinearer Regression.

Es lohnt sich zu reflektieren, wie viel *klassische* Mathematik in dieser *modernen* Anwendung steckt: Die Modellierung des Zusammenhangs zwischen Input- und Outputdaten mit Funktionen g, die Minimierung mit Mitteln der Analysis und nicht zuletzt das statistische Quantifizieren der Ergebnisse auf den Testdaten. Für diesen letzten Schritt kann die euklidische Norm oder eine beliebige andere Metrik verwendet werden – womit ein weiterer klassischer Gegenstand der Mathematik relevant ist (vgl. Abschn. Ähnlichkeits- und Distanzmaße).

Zwischenfazit: Zentrale Schritte bei der Entwicklung von ML-Methoden

Die lineare Regression macht einige zentrale Elemente zahlreicher ML-Methoden ersichtlich, so auch von den im Folgenden beschriebenen künstlichen neuronalen Netzen (vgl. dazu auch Biehler et al., 2024):

1. **Trainings- und Testdaten:** Ausgangspunkt sind bekannte Input- und zugehörige Outputdaten. Diese werden in der Regel geeignet vorverarbeitet und dann in Trainings- und Testdaten unterteilt.
2. **Festlegung auf ein mathematisches Modell:** Es wird ein mathematisches Modell festgelegt, vielfach eine gewisse Funktionsklasse, aus der die Funktion gewählt werden soll, die die gegebenen Trainingsdaten „bestmöglich" beschreibt. Mit anderen Worten sollen gewisse Parameter der Funktion / des Modells bestmöglich geschätzt werden.
3. **Formulierung eines Optimierungsproblems:** Es ist zu klären, was mit „bestmöglich" gemeint ist. Dazu wird eine Metrik gewählt und darüber eine Fehlerfunktion definiert, die die Abweichung zwischen den tatsächlichen Outputdaten $\boldsymbol{y}_i$ und den Prognosen $\hat{\boldsymbol{y}}_i$ beschreibt.
4. **Lösen des Optimierungsproblems:** Ziel ist es, die Parameter des Modells so zu wählen, dass der Fehler bzgl. der Trainingsdaten möglichst klein wird. Dazu kommen meist[5] numerische Verfahren zum Einsatz. Dieser Schritt wird auch als *Trainingsphase* bezeichnet.
5. **Testen und Validieren des Modells:** Bevor das Modell in die Anwendung geht, wird es auf Daten validiert, die nicht in der Phase der Modellentwicklung eingesetzt wurden. Dazu kommen die Testdaten zum Einsatz. Dieser Schritt wird auch als *Testphase* bezeichnet.

Insbesondere in Schritt 2 und 3 wird die Rolle des oder der Modellierer*in ersichtlich: gewisse Entscheidungen werden eben auch beim ML noch vom Menschen getroffen, z. B. die Wahl der Funktionsklasse und der Fehlerfunktion, aber auch Parameter (sog. Hyperparameter) im Optimierungsalgorithmus (bspw. der Startwert).

Die Regressionsanalyse ist ein Beispiel für ein Verfahren des MLs, das sehr transparent ist: Es ist leicht verständlich, wie das „Training" wirkt, wie auf der Basis der gelernten Parameter Vorhersagen berechnet werden und was die Parameter darin bedeuten (s. folgendes Beispiel). Dies liegt an der simplen Modellfunktion. Wird diese komplexer, wie das bei künstlichen neuronalen Netzen der Fall ist, geht die Interpretierbarkeit der einzelnen Parameter oft verloren.

Beispiel: Regression zur Vorhersage von Ferienhauspreisen

Als Anwendungsbeispiel für Regressionsprobleme betrachten wir folgende Frage: Wie lässt sich aus verschiedenen Angaben über ein Ferienhaus dessen Mietpreis vorhersagen? Als Trainingsdatensatz wurden aus einem Online-Portal folgende Daten von 83 Ferienhäusern auf Bornholm herausgesucht: Zahl der erlaubten Personen, Zahl der Zimmer, Wohnfläche in qm, Erlaubnis von Hunden (kodiert als nein = 0 bzw. ja = 1), Zahl der Sterne bzgl. der Qualität der Ausstattung (3, 4 oder 5), Meerblick (0-1-kodiert), die Entfernung zum Meer und der Mietpreis für eine Woche im Spätsommer. Bei fast allen Variablen spricht ein hoher Wert für einen hohen Preis – außer bei der Entfernung zum Meer. Die Prädiktorvariablen für jedes Haus sind $\boldsymbol{x} = (Pers, Zimmer, Wfl, Hund, Sterne, Meerblick, Meerentf)$, die Zielvariable ist der Mietpreis. Welche Prognosefunktion man für die Vorhersage nimmt, ist eine Frage, bei der man viel diskutieren und ebenso viel ausprobieren kann. Wir haben uns für die Funktion g mit

$$g(\boldsymbol{p}, \boldsymbol{x}) = cPers \cdot Pers + cZimmer \cdot Zimmer + cWfl \cdot Wfl + cHund \cdot Hund + \cdots$$

[5] Im Falle der linearen Regression ließe sich das Problem noch analytisch lösen, dies ist bei den meisten anderen Optimierungsproblemen im Kontext von ML-Methoden nicht möglich.

$$cSterne \cdot (Sterne - 3) + cMeerblick \cdot Meerblick + \frac{cMeerentf}{\sqrt{Meerentf}} + c_0$$

und Parametervektor $\boldsymbol{p} = (cPers, cZimmer, cWfl, cHund, cSterne, cMeerblick, cMeerentf, c_0) \in \mathbb{R}^8$ entschieden. Der Parametervektor wird durch Lösen eines Optimierungsproblems bestmöglich bestimmt. Der Ansatz mit der Quadratwurzel der Entfernung zum Meer folgt der Intuition, dass ein Haus nahe am Meer teurer sein sollte als ein weiter entferntes, und dass die Nähe zum Meer den Preis treibt. Eine Realisierung findet man im Notebook *Ferienhauspreise_linReg.ipynb.* Die aus der Optimierung gewonnenen optimalen Parameterwerte können zum einen genutzt werden, um bei weiteren Häusern vorherzusagen, wie teuer sie vermutlich sind, zum anderen können die Werte auch interpretiert werden: So ergibt sich etwa $cPers = 10{,}20\,€$ und $cZimmer = 108\,€$, d. h. vor allem bestimmt die Zahl der Zimmer den Preis. Der Blick aufs Wasser ist – bei sonst gleichen Hausparametern – gar nicht so teuer: $cMeerblick = 39\,€$.

Die Qualität eines Regressionsmodells lässt sich z. B. mit dem mittleren quadratischen Fehler der Prognosen auf den Trainings- und Testdaten bewerten. Damit lässt sich Modellvariation systematisch betreiben. Eine Option wäre im obigen Modell zur Zielfunktion einen nichtlinearen Term $cWfl \cdot Wfl \cdot cSterne \cdot (Sterne - 3)$ hinzuzufügen. Die Idee dabei ist, dass hochwertige Wohnfläche besonders teuer ist. Modelldiskussion und Modellkritik ergeben sich aus einer solchen Fragestellung fast automatisch.

Künstliche neuronale Netze

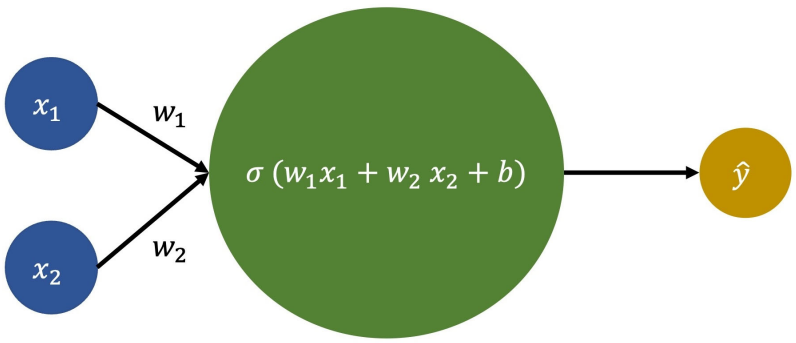
Mit einem künstlichen neuronalen Netz (KNN) lassen sich unterschiedliche Aufgaben lösen – insbesondere *Regressionsprobleme* ($\boldsymbol{y}_i$ numerisch) und *Klassifikationsprobleme* ($\boldsymbol{y}_i$ kategorial), aber auch das Auffinden von Clustern in Datensätzen ist damit möglich (unüberwachtes Lernen).

Die Antworten eines KNNs stammen im Gegensatz zur recht transparenten linearen Regression meist aus für uns „unergründlichen Tiefen“ seiner inneren Vernetzung (also einer komplexen Prognosefunktion g). Gemeinsam ist beiden Verfahren, dass sie auf ein Optimierungsproblem hinauslaufen und gewisse Parameter einer von uns definierten Modellklasse (Funktionsklasse) optimal gewählt werden sollen.

Das Neuron

Die Idee von KNNs stammt aus der Biologie: Gewisse Strukturen des Nervensystems werden mit Mitteln der Mathematik und der Informatik modelliert. Nervenzellen (Neuronen) sammeln über mehrere Dendriten Signale und aggregieren diese. Zusammen bestimmen sie das Aktionspotential, das über das Axon und Synapsen an weitere Neuronen weitergegeben wird, sofern ein gewisser Schwellenwert erreicht wurde. Man spricht auch davon, dass das Neuron “feuert”.

Abb. 4: Visualisierung eines künstlichen Neurons



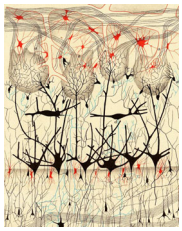
In KNNs wird ein Neuron mathematisch durch eine Funktion $g\colon \mathbb{R}^k \to \mathbb{R}$ mit $g(\boldsymbol{x}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$ modelliert. Hierbei sind $\boldsymbol{w} \in \mathbb{R}^k$ Parameter (Gewichte), die im Skalarprodukt $\boldsymbol{w} \cdot \boldsymbol{x}$ die Sensitivität des Neurons auf Einträge des Inputvektors $\boldsymbol{x} \in \mathbb{R}^k$ gewichten. Der Parameter $b \in \mathbb{R}$ ist eine Gesamtverschiebung (Bias). Schematisch wird dies oft wie in Abbildung 4 dargestellt. Das “Feuern” eines biologischen Neurons wird modelliert durch eine nichtlineare Funktion, die als Aktivierungsfunktion bezeichnet wird. Oft verwendete Aktivierungsfunktionen sind die Sigmoid-Funktion $\sigma\colon \mathbb{R} \to \mathbb{R}, \sigma(z) = \frac{1}{1+e^{-z}}$ (ein netter Gegenstand für eine schnelle Kurvendiskussion) oder die ReLU-Funktion $\sigma\colon \mathbb{R} \to \mathbb{R}, \sigma(z) = \max(0, z)$.

Das Training bzw. der "Lernprozess" eines künstlichen (mathematischen) Neurons beruht auf der Bestimmung der optimalen Gewichte $\boldsymbol{w}$ und des Bias-Parameters b, sodass ein unbekannter Zusammenhang zwischen gegebenen Input- und Outputdaten bestmöglich beschrieben wird. Um dies mit Schüler*innen zu erarbeiten, bietet es sich an mit niedrigdimensionalen Daten einzusteigen: Es sind Inputdaten $\boldsymbol{x}_i \in \mathbb{R}^2$ und zugehörige Outputdaten $y_i \in \mathbb{R}, i = 1, \ldots, N$ gegeben. Basierend auf diesen Trainingsdaten sollen die Einträge des Gewichtsvektors $\boldsymbol{w} \in \mathbb{R}^2$ und der Wert des Bias-Parameters $b \in \mathbb{R}$ so gewählt werden, dass die Summe der Fehlerquadrate zwischen den berechneten Outputs $\hat{y}_i$ und den tatsächlichen Outputdaten y_i minimal wird:

$$\min_{\boldsymbol{w},b} \sum_{i=1}^{N} (\sigma(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - y_i)^2 = \min_{\boldsymbol{w},b} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 .$$

Das Training eines einfachen künstlichen Neurons beruht damit wiederum auf dem Lösen eines Optimierungsproblems und funktioniert nach dem gleichen Schema wie die numerische Bestimmung der optimalen Parameter eines linearen Regressionsmodells. Durch die nichtlineare Aktivierungsfunktion können nun auch nichtlineare Zusammenhänge modelliert werden. Das Training von KNNs ist demnach ein Spezialfall der nichtlinearen Regression. Allerdings ist diese im Allgemeinen sehr rechenaufwendig. Die spezielle, im Folgenden beschriebene Struktur von KNNs erlaubt hingegen die Anwendung sehr effizienter Optimierungsalgorithmen. Damit können sehr große Netze mit Millionen Parametern trainiert werden.

Abb. 5: Eine Ebene von Neuronen aus den Geruchsnerven eines Hundes (Quelle: https://garystockbridge617.getarchive.net/amp/media/camillo-golgis-image-of-a-dogs-olfactory-bulb-detail-2-957500)



Vom einzelnen Neuron zum neuronalen Netz

Ein einzelnes Neuron macht noch kein Netz. Reale Neuronen sind im Nervensystem und auch im Gehirn vielfältig verschachtelt, wie etwa Abbildung 5 aus den Geruchsnerven eines Hundes zeigt. Solche Verschachtelungen werden durch Schichten von Neuronen idealisiert, was mathematisch auf eine Hintereinanderausführung von Funktionen hinausläuft. An der Eingabeschicht legt man ein Signal (einen Inputdatenpunkt) $\boldsymbol{x} \in \mathbb{R}^n$ an, der in der ersten „versteckten" Schicht von k_1 Neuronen in Vektoren aus $\mathbb{R}^{k_1}$ transformiert wird. Diese Vektoren dienen als Eingabe für die nächste Schicht usw. bis an der Ausgabeschicht (L-te Schicht) ein Vektor aus $\mathbb{R}^{k_L}$ als Ausgabe berechnet wird. Zur Vereinfachung der Notation erweitert man die Sigmoid-Funktion $\sigma: \mathbb{R} \to \mathbb{R}$ durch komponentenweises Anwenden auf $\sigma: \mathbb{R}^n \to \mathbb{R}^n$. Damit lässt sich die Ausgabe $\boldsymbol{a}^{(l)}$ einer beliebigen Schicht $l \in \{1, \ldots, L\}$ schreiben als:

$$\boldsymbol{a}^{(l)} = g^{(l)}(\boldsymbol{a}^{(l-1)}) = \sigma(\boldsymbol{W}^{(l)} \cdot \boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}) \in \mathbb{R}^{k_l}$$

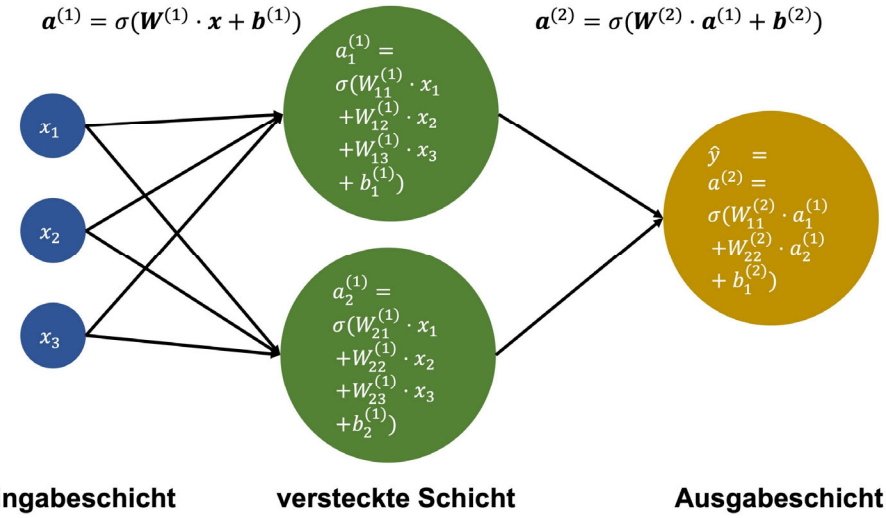
mit einer Gewichtsmatrix $\boldsymbol{W}^{(l)} \in R^{k_l \times k_{l-1}}$ und einem Bias-Vektor $\boldsymbol{b}^{(l)} \in \mathbb{R}^{k_l}$. Dabei entspricht $\boldsymbol{a}^{(0)}$ dem Inputvektor. Die Gewichtsmatrix beinhaltet zeilenweise die Gewichtsvektoren der einzelnen Neuronen und der Bias-Vektor beinhaltet die Bias-Parameter der einzelnen Neuronen einer Schicht. Das komplette Netz besteht aus der Hintereinanderausführung mehrerer solcher Funktionen $g(\boldsymbol{x}) = g^{(L)} \circ \ldots \circ g^{(1)}(\boldsymbol{x})$. Dass dabei tiefe Verschachtelungen möglich sind, gab dem "deep learning" seinen Namen. Ein KNN ist im Kern somit eine vielfach verkettete, nichtlineare Funktion. In Abbildung 6 werden an einem kleinen KNN relevante Rechnungen veranschaulicht.

Auch bei KNNs aus vielen Neuronen und zahlreichen Schichten besteht das Training im Lösen eines Optimierungsproblems. Formal soll für gegebene Trainingsdaten $(\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{x}_i \in \mathbb{R}^n, \boldsymbol{y}_i \in \mathbb{R}^m$ die Zielfunktion $F(\boldsymbol{p}) := \sum_{i=1}^{N} ||\boldsymbol{y}_i - g(\boldsymbol{p}, \boldsymbol{x}_i)||^2$ minimiert werden, wobei $\boldsymbol{p} = (\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \ldots, \boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)})$ alle Gewichtsmatrizen und alle Bias-Vektoren umfasst. Je nach Anwendungsproblem sind andere Definitionen der Zielfunktion sinnvoll. Die Summe der Fehlerquadrate erleichtert jedoch die Anwendung von Methoden der Analysis.

Das Training eines KNNs ist abstrakt betrachtet die optimale Wahl der Parameter $\boldsymbol{p}$ einer Funktion $g: \mathbb{R}^n \to \mathbb{R}^m$, so dass die Funktion die dem Trainingsdatensatz zugrundeliegende, unbekannte Zuordnung

$\boldsymbol{x}_i \to \boldsymbol{y}_i$ möglichst gut widerspiegelt. Nimmt man an, dass diese Zuordnung durch eine Funktion G beschrieben wird (also $\boldsymbol{y}_i = G(\boldsymbol{x}_i)$, evtl. mit zufälligem Fehler), beschreibt das Optimierungsproblem ein Approximationsproblem.

Abb. 6: Darstellung eines künstlichen neuronalen Netzes mit einer versteckten Schicht aus zwei Neuronen



Die Wahl der Anzahl an Schichten und der Neuronen pro Schicht sowie der Aktivierungsfunktion σ und damit die Struktur der Funktionen $g^{(l)}$ beruht meist auf der Erfahrung der Person, die das KNN für einen bestimmten Zweck entwirft. Klar ist aber, dass für σ keine linearen Funktionen verwendet werden dürfen, weil die Verkettung linearer Funktionen wieder eine lineare Funktion ergibt, und damit keine nichtlinearen Zusammenhänge beschrieben werden könnten. Wird eine nichtlineare Aktivierungsfunktion verwendet, kann bewiesen werden, dass jede zugrundeliegende stetige Funktion G auf einer kompakten Definitionsmenge gleichmäßig durch ein hinreichend großes Netz approximiert werden kann (allgemeiner Approximationssatz von Cybenko (1989) und Hornik et al. (1989), s. https://en.wikipedia.org/wiki/Universal_approximation_theorem für eine kompakte Erklärung). Durchaus gibt es viele andere Funktionsklassen, die eine solche gleichmäßige Approximation leisten (etwa Polynome), aber es zeigt sich, dass neuronale Netze dies mit verhältnismäßig wenig Schichten erreichen (weiterführende Literatur findet man z. B. in Kutyniok, 2024).

Beispiel 1: Klassifikation mit künstlichen neuronalen Netzen

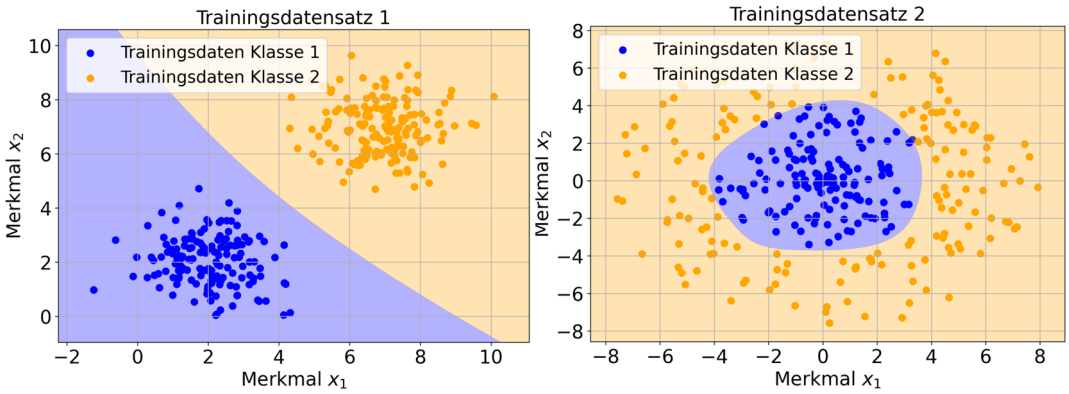
Klassifizierungsprobleme treten in diversen Anwendungen auf: bei der Erkennung von Pflanzenarten, in Spamfiltern, bei der Klassifikation von Gesichtern auf Bildern oder im Bereich der medizinischen Diagnose. In diesem Beispiel wird mit synthetischen Daten und unterschiedlichen Architekturen eines KNNs zur Klassifikation experimentiert und zentrale Bausteine von KNNs auf der *Mesoebene* betrachtet. Im Unterricht kann man dazu wiederum mit einem simplen Klassifizierungsproblem mit zweidimensionalen Inputdaten $\boldsymbol{x}_i \in \mathbb{R}^2$ und lediglich zwei Klassen einsteigen (Abb. 7). Für Inputdaten $\boldsymbol{x}_i$ der Klasse 1 wählen wir als zugehörigen Output $\boldsymbol{y}_i = 0$ und für Daten der Klasse 2 $\boldsymbol{y}_i = 1$. Andere Kodierungen der Klassenzuordnungen (Labels) sind durchaus denkbar.[6]

Wie geht man nun an die Entwicklung eines KNNs zum Lösen des Klassifizierungsproblems heran? Die Inputdaten sind zweidimensional, entsprechend sollte die Eingabeschicht aus zwei Neuronen bestehen. Die Outputdaten haben wir eindimensional gewählt, d. h. es wird ein Ausgabeneuron benötigt. Die Vorhersagen sollen in [0,1] liegen. Damit sollte in der letzten Schicht eine Aktivierungsfunktion eingesetzt werden, die in dieses Intervall abbildet – bspw. die Sigmoid-Funktion. Die Schichten dazwischen können relativ beliebig gewählt werden. Wir starten mit zwei versteckten Schichten aus je 10 Neuronen und verwenden in jeder Schicht

[6] Oft werden Outputvektoren verwendet, deren Länge der Anzahl der gegebenen Klassen entspricht. Die Einträge eines vorhergesagten Outputvektors können dann als Wahrscheinlichkeiten interpretiert werden, mit denen ein Datenpunkt den jeweiligen Klassen angehört.

als Aktivierungsfunktion die Sigmoid-Funktion. Den Fehler (und damit die Zielfunktion des Optimierungsproblems) definieren wir über die Summe der Fehlerquadrate[7]. Im Notebook *Klassifikation_KNN.ipynb* kann (auf der Mesoebene) erkundet werden, wie sich die Änderungen der Netzstruktur (Hinzunahme von Schichten oder Neuronen; Änderung der Aktivierungsfunktion) auf die Genauigkeit der Klassifikation auswirkt.

Abb. 7: Zwei Trainingsdatensätze und Entscheidungsgrenzen des jeweiligen KNNs



Wir trainieren für zwei verschiedene Trainingsdatensätzen je ein KNN (Abb. 7). Datensatz 1 ist simpel, da die Datenpunkte der beiden Klassen linear separierbar sind. Bei Datensatz 2 sieht das anders aus. Hier brauchen wir ein komplexeres, nichtlineares Modell.

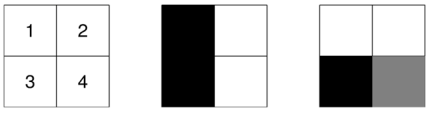
Mit der oben beschriebenen Struktur der KNNs ergeben sich die in Abbildung 7 dargestellten Entscheidungsgrenzen. Diese Grenzen veranschaulichen, welche Klassenzuordnung das jeweilige Modell für Inputdaten aus verschiedenen Bereichen des Merkmalsraums (hier: $\mathbb{R}^2$) vorhersagt. Die Entscheidungsgrenze kann bei KNNs oft nicht explizit durch eine geschlossene mathematische Formel dargestellt werden. Um die Grenze zu bestimmen, wird der Merkmalsraum in ein feines Gitter unterteilt (also diskrete Punkte ausgewählt). Für jeden Gitterpunkt wird die Klassenzuordnung berechnet. So ergibt sich die näherungsweise Unterteilung in Bereiche und die Entscheidungsgrenze. Zur spielerischen Erkundung des Einflusses verschiedener Netzstrukturen eines KNNs für Klassifizierungs- und Regressionsprobleme kann der Tensorflow Playground (https://playground.tensorflow.org) oder eine deutschsprachige Variante davon auf https://kiwi.schule eingesetzt werden.

Beispiel 2: Neuronales Netzwerk als Kantendetektor

Das folgende Beispiel soll zeigen, wie das Training eines KNNs zum Erkennen von Strukturen in Pixelbildern funktioniert. Das Netz soll erkennen, ob in einem Bild aus 2x2 Pixeln eher vertikale oder horizontale Strukturen vorliegen. Dies ist eine extrem vereinfachte Situation, die aber doch wesentliche Prinzipien der Bilderkennung mit KNNs zeigt.

Um die Eingabe des Netzes in einen Vektor zu überführen, strukturieren wir die Pixel in einer linearen Anordnung. Dazu nummerieren wir diese von 1 bis 4 (Abb. 8) und kodieren ihre Helligkeitswerte als Zahlen zwischen 0 (weiß) und 1 (schwarz). Die Eingabevektoren $\boldsymbol{x}_i, i = 1, \ldots, N$ sind in diesem Fall also aus $\mathbb{R}^4$.

Abb. 8: Links Anordnung der Pixel, mittig ein Bild, das als $(1,0,1,0)$ kodiert wird, rechts ein Bild, das $(0,0,1,0.5)$ entspricht



Als Ausgabevektoren $\boldsymbol{y}_i$ wählen wir Vektoren im $\mathbb{R}^2$ mit Einträgen zwischen 0 und 1. Ein hoher Wert des ersten Eintrags signalisiert das Vorliegen von vertikalen Strukturen. Analog zeigt der zweite Eintrag des Ausgabevektors eher horizontale Strukturen an. Als Netzstruktur wählen wir (da gibt es erneut viel willkürliche Freiheit)

[7] Anstelle der Summe der Fehlerquadrate werden bei Klassifizierungsproblemen oft komplexere Fehlerfunktionen verwendet, bspw. die Kreuzentropie.

eine versteckte Schicht mit drei Neuronen: Zunächst wird der $\mathbb{R}^4$ also auf den $\mathbb{R}^3$ und schließlich auf $\mathbb{R}^2$ abgebildet. Als Fehlerfunktion betrachten wir die Summe der Fehlerquadrate zwischen den gewünschten Ausgaben und den vorhergesagten Ausgaben (vgl. *Kantendetektor_KNN.ipynb*).

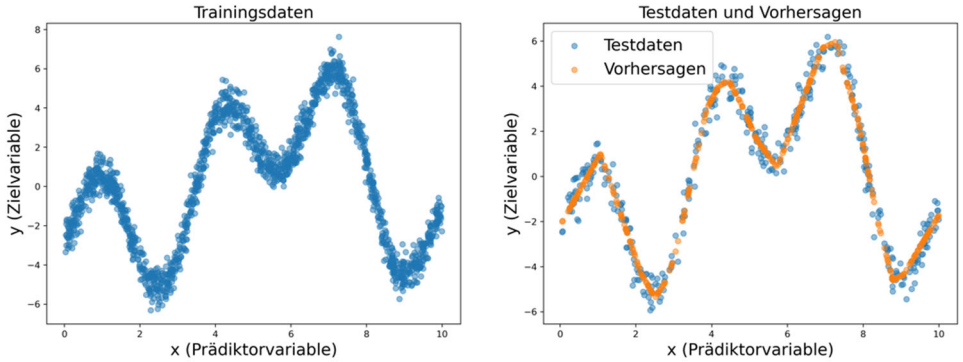
Um von diesem Beispiel zu echten Anwendungen zu kommen, etwa dem Erkennen einer Person auf einem Bild, muss man vor allem groß denken: Statt 2x2 Bildern als Eingabe werden Bilder in authentischer, deutlich höherer Auflösung verwendet. Damit sind die Eingabedaten hochdimensional. Statt einer versteckten Schicht gibt es viele Schichten und auch die Anzahl Neuronen pro Schicht wird oft sehr groß gewählt. Dies gilt auch für die Ausgabeschicht. Wenn es etwa um die Gesichtserkennung geht, hat man für jede Klasse, d. h. jeden zu erkennenden Menschen, ein Ausgabeneuron.

Die Beschreibung macht klar, dass in authentischen Anwendungen Millionen, wenn nicht Milliarden von Parametern zu bestimmen sind. Die Anpassung der Gewichte mit einem simplen Minimierungsalgorithmus, wie dem beschriebenen Verfahren des steilsten Abstiegs, würde dann zu viel Rechenzeit beanspruchen. Dies liegt unter anderem daran, dass die direkte Berechnung des Gradienten sehr zeitaufwendig ist. Verschiedene Strategien helfen, dieses Problem zu lösen. Zum einen wird die Zielfunktion nicht auf einmal gebildet; stattdessen werden die Trainingsdaten schrittweise eingegeben und die Gewichte nach und nach angepasst. Ein Algorithmus, der das leistet, ist das Backpropagation (z. B. Deisenroth et al., 2020). Zum anderen können häufig viele Gewichte in solchen Netzen fest auf Null gesetzt werden. Dies hat sich bei der Bilderkennung als effektiv erwiesen, da benachbarte Pixel gemeinsam verarbeitet werden, während weit entfernte Pixel erst in späteren Schichten miteinander in Kontakt kommen. Dieses Vorgehen ist die Idee hinter sogenannten Faltungsnetzen (engl. Convolutional Neural Networks).

Beispiel 3: Regression mit künstlichen neuronalen Netzen

KNNs werden zum Lösen von (komplexen, nichtlinearen) Regressionsproblemen eingesetzt. Auch das Ferienhausbeispiel kann mit einem neuronalen Netz modelliert werden (siehe *Ferienhauspreise_KNN.ipynb*). Da KNNs durch Hinzunahme weiterer Schichten und zusätzlicher Neuronen pro Schicht systematisch vergrößert werden können, lässt sich eine hohe Flexibilität in der Anpassung erreichen, was oft zu besseren Vorhersagen auf den Trainingsdaten führt, als wenn der Modellierende eine feste Modellfunktion vorgibt. Nachteil ist aber, dass die Parameter schlechter zu interpretieren sind. Außerdem steigt mit der Zahl der Neuronen und damit der Parameter auch der Rechenaufwand und der Bedarf an Trainingsdaten, um die Parameter hinreichend gut festzulegen.

Abb. 9: Trainingsdatensatz, Testdatensatz und Vorhersagen eines KNNs



Im Unterricht können KNNs im Kontext der Regression am Beispiel eines Datensatzes mit eindimensionalen Inputdaten $x_i \in \mathbb{R}$ und Outputdaten $y_i \in \mathbb{R}$ erarbeitet und der Einfluss der Struktur eines KNNs erkundet werden. Im Notebook *Regression_KNN.ipynb* wird dazu der Datensatz aus Abbildung 9 (links) bereitgestellt und ein KNN mit zwei versteckten Schichten mit je 10 Neuronen trainiert. Die Vorhersage berechnet sich gemäß

$$\hat{y}_i = g(\boldsymbol{p}, x_i) = \boldsymbol{W}^{(3)}(\sigma(\boldsymbol{W}^{(2)}(\sigma(\boldsymbol{W}^{(1)}x_i + \boldsymbol{b}^{(1)})) + \boldsymbol{b}^{(2)})) + b^{(3)},$$

wobei $\boldsymbol{p} = (\boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}, \boldsymbol{W}^{(3)}, b^{(3)})$ und $\boldsymbol{W}^{(1)} \in \mathbb{R}^{10x1}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^{10}$, $\boldsymbol{W}^{(2)} \in \mathbb{R}^{10x10}$, $\boldsymbol{b}^{(2)} \in \mathbb{R}^{10}$, $\boldsymbol{W}^{(3)} \in \mathbb{R}^{1x10}$, $b^{(3)} \in \mathbb{R}$, sowie σ der ReLU-Funktion entspricht. Als Fehlerfunktion wurde die mittlere Summe der Fehlerquadrate gewählt. Bei Regressionsproblemen wird in der Ausgabeschicht oft keine Aktivierungsfunktion eingesetzt. Grund ist, dass die Outputs je nach Anwendungsfall nicht auf den Wertebereich der Akti-

vierungsfunktion beschränkt werden sollen. Das Training des KNNs besteht nun im Lösen des Optimierungsproblems $\min_{\boldsymbol{p}} \frac{1}{N} \sum_{i=1}^{N} (y_i - g(\boldsymbol{p}, x_i))^2$. Mit dem trainierten KNN aus nur wenigen Schichten und Neuronen können die Testdaten bereits einigermaßen genau vorhergesagt werden (Abb. 9 rechts).

Lernende können das Notebook als Ausgangspunkt nutzen, um verschiedene Aktivierungsfunktionen zu testen (Mikroebene), oder die Anzahl der versteckten Schichten und die Anzahl der Neuronen pro Schicht zu variieren und den Einfluss auf den Fehler bzgl. der Trainings- und Testdaten zu erkunden (Mesoebene).

Zwischenfazit: Optimierung

Die ML-Säule der Optimierung besteht darin, Parameter einer Modellfunktion durch Minimierung einer Fehlerfunktion (Zielfunktion) zu bestimmen. Diese Fehlerfunktion basiert in der Regel auf dem Abstand zwischen den gewünschten und den tatsächlichen Ausgaben. Neben der Minimierung einer Funktion spielt daher auch die Modellierung von Abständen eine zentrale Rolle. Dafür gibt es viele mathematische Ansätze, die auch von der Schulmathematik bereitgestellt werden (vgl. folgender Abschnitt).

Optimierungsprobleme sind zentraler Bestandteil zahlreicher weiterer ML-Methoden wie beispielsweise der Support Vector Machine, mit der u. a. Klassifizierungsprobleme gelöst werden können. Bei dieser wird eine Gerade oder Ebene oder Hyperebene so gewählt, dass sie die Datenpunkte zweier Klassen *bestmöglich* voneinander trennt. Auch diese Methode lässt sich auf schulmathematische Inhalte reduzieren und bietet neben der Optimierung eine schöne Anwendung von Inhalten der analytischen Geometrie (Schönbrodt et al., 2022).

Säule 3: Ähnlichkeits- und Distanzmaße

Das Messen von Ähnlichkeiten und Abständen zwischen Datenpunkten spielt eine wichtige Rolle im Bereich des MLs. Ansätze der Ähnlichkeits- und Abstandsmessung sind durchaus eng verknüpft. Der Blickwinkel ist jedoch ein leicht anderer: Ähnlichkeitsmaße geben typischerweise an, wie ähnlich sich zwei Datenpunkte sind (hoher Wert = große Ähnlichkeit), wohingegen Metriken zur Bestimmung von Abständen angeben, wie unähnlich zwei Datenpunkte sind (hoher Wert = großer Unterschied). Ähnlichkeitsmaße lassen sich oft aus Metriken gewinnen, indem bspw. die Inverse betrachtet wird.

Oben haben wir bereits gesehen, dass die Modellierung von Abständen zwischen Datenpunkten bei der Beschreibung der Zielfunktion eines KNNs eine wichtige Rolle spielt. Auch bei der statistischen Bewertung eines ML-Modells anhand von Testdaten (vgl. Abschn. Statistische Gütemaße) sind Metriken wesentlich. Einige ML-Verfahren beruhen darüber hinaus auf der direkten Messung von Ähnlichkeiten bzw. Abständen zwischen Datenpunkten – bspw. die k-nächste-Nachbarn-Methode, die zum Lösen von Klassifizierungsproblemen eingesetzt werden kann. Um einen neuen Datenpunkt einer Klasse zuzuordnen, wird die “Nähe” des neuen Datenpunktes zu allen Trainingsdatenpunkten quantifiziert. Der Datenpunkt wird dann per Mehrheitsentscheid der Klasse zugewiesen, von denen unter den k „nächsten“ Datenpunkten die meisten vorliegen. Dazu muss also die Ähnlichkeit oder Distanz zwischen zwei Vektoren $\boldsymbol{u} = (u_1, \ldots, u_n)$ und $\boldsymbol{v} = (v_1, \ldots, v_n)$ mathematisch beschrieben und quantifiziert werden. Wesentliche auch für Schüler*innen verständliche Ansätze sind (Oldenburg, 2021; eine elementare Realisierung in Python findet sich in Oldenburg, 2011):

- **Die euklidische Norm:** Zwei Vektoren $\boldsymbol{u}$ und $\boldsymbol{v}$ sind umso ähnlicher, je kleiner die euklidische Norm ihrer Differenz ist: $||\boldsymbol{u} - \boldsymbol{v}||_2$.
- **Die Kosinus-Ähnlichkeit:** Zwei Vektoren $\boldsymbol{u}$ und $\boldsymbol{v}$ sind umso ähnlicher, je größer der Kosinus des Winkels α, d. h. je kleiner der Winkel zwischen den beiden Vektoren ist: $cos(\alpha) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{||\boldsymbol{u}|| \cdot ||\boldsymbol{v}||}$.
- **Skalarproduktähnlichkeit:** Zwei Vektoren $\boldsymbol{u}$ und $\boldsymbol{v}$ sind umso ähnlicher, je größer ihr Skalarprodukt ist. Dies entspricht der Kosinus-Ähnlichkeit, wenn mit normierten Vektoren gearbeitet wird.

Vor- und Nachteile dieser Ansätze liegen auf der Hand: Kommt es in einer konkreten Anwendung nur auf die Richtung oder auch auf die Länge der Vektoren an? Davon abgesehen ist die Wahl nicht immer kritisch, wie die folgende Übungsaufgabe zeigt: Wenn alle Vektoren auf Länge 1 normiert sind, und $\boldsymbol{u}$ ähnlicher zu $\boldsymbol{v}$ ist als zu $\boldsymbol{w}$ mit einem dieser Ähnlichkeits- bzw. Distanzmaße, dann gilt das Gleiche mit den anderen Maßen.

Darüber hinaus gibt es viele weitere Ansätze (Levy et al., 2024). Erfahrungsgemäß finden Lernende die Summe der absoluten Abweichungen oft intuitiv einleuchtend, also den Ausdruck $\sum_{i=1}^{n} |u_i - v_i|$. Die Verwendung dieser Metrik zur Definition einer Zielfunktion hat jedoch numerische Nachteile, da sie nicht differenzierbar ist.

Auch beim Clustern (unüberwachtes Lernen) ist die Messung von Ähnlichkeiten bzw. Distanzen relevant. Hier wird versucht, ähnliche Datenpunkte zu Clustern zu gruppieren. Ein Beispiel für eine solche Methode ist der k-means-Algorithmus.

Ansätze der Ähnlichkeitsmessung verbergen sich zudem hinter großen Sprachmodellen. Diese übersetzen Texte zunächst in eine Folge von Tokens. Das sind natürliche Zahlen, die bei vielen modernen Sprachmodellen in der Größenordnung von 10^5 liegen, und die für Wörter, Wortteile oder Satzzeichen stehen. Die Tokens werden dann in einen hochdimensionalen reellen Vektorraum abgebildet. Bei dieser Vorverarbeitung spielt Ähnlichkeit eine wichtige Rolle: inhaltlich ähnliche Tokens werden i.d.R. Vektoren zugeordnet, die ähnlich sind. Im Notebook *wordEmbeddings.ipynb* kann ausprobiert werden, wie solche Embedding-Vektoren bestimmt und wie mit ihnen gerechnet werden kann.

Säule 4: Wahrscheinlichkeiten

Eine weitere mathematische Säule sind Wahrscheinlichkeiten. Diese haben Auswirkungen auf die Entwicklung und Evaluierung von ML-Methoden. Bei der Entwicklung ist zu bedenken, dass die Trainingsdaten i.d.R eine Zufallsstichprobe darstellen. Bei der Evaluierung kann nicht geprüft werden, ob alle Gewichte “richtig” gewählt sind – es können nur Statistiken über richtige Vorhersagen angelegt werden.

Wahrscheinlichkeiten spielen auch bei der Entwicklung von Sprachmodellen eine zentrale Rolle. Dies können Nutzer*innen von Smartphones nachvollziehen, wenn beim Tippen ein Fehler passiert und eine plausible Korrektur vorgeschlagen wird. Angenommen, es wurde getippt „Ein schmackhafter Affel“. Das letzte Wort ist nicht im Wörterbuch enthalten. Das System sollte also einen Korrekturvorschlag machen. Zunächst wird wieder das Prinzip der größtmöglichen Ähnlichkeit bemüht: viele Fehler passieren vermutlich so, dass das gemeinte Wort ähnlich dem fehlerhaft getippten Wort sein dürfte. Im Wörterbuch finden sich beispielsweise „Apfel“ und „Affe“. Beide Wörter liegen nahe an dem eingetippten Wort in dem Sinne, dass nur ein falscher/überschüssiger Tastendruck ausreicht, um zu erklären, wie aus dem gemeinten Wort „Affel“ wurde.

Wie spielen nun Wahrscheinlichkeiten in Systeme der Wortkorrektur hinein? Ausgangspunkt ist ein umfangreicher Textkorpus der deutschen Sprache, gegebenenfalls ergänzt um Texte, die der Benutzer schon selbst verfasst hat. Für diesen Text lässt sich auszählen, wie oft auf das Wort “schmackhafter“ das Wort „Apfel“ bzw. wie oft das Wort „Affe“ folgt. Dividiert man die jeweiligen absoluten Häufigkeiten der Wortübergänge durch die Gesamtzahl des Auftretens des vorangegangenen Wortes (hier: Auftretenshäufigkeit des Wortes “schmackhaft”), so liefert dies relative Häufigkeiten für die entsprechenden Wortübergänge. Aus diesen lassen sich die bedingten Wahrscheinlichkeiten (Wahrscheinlichkeit, dass Wort B folgt, unter der Bedingung das Wort A getippt wurde) schätzen und damit auswählen, welches Wort vermutlich gemeint war. Noch besser wird die Prognose, wenn nicht nur das direkt vorhergehende Wort, sondern $2, 3, 4, \ldots$ vorhergehende Worte bei der Schätzung der bedingten Wahrscheinlichkeiten berücksichtigt werden. Dieses Modell wird im Bereich der natürlichen Sprachverarbeitung auch als N-Gramm-Modell bezeichnet. Dabei steht N für die Anzahl der betrachteten Wörter, die für die Schätzung der Übergangswahrscheinlichkeit herangezogen werden (Oldenburg, 2008, für eine didaktische Elementarisierung).

Das N-Gramm-Modell kann nicht nur zur Wortkorrektur, sondern auch zum Erzeugen von Wortvorschlägen beim Tippen einer Nachricht am Smartphone (vgl. Hofmann & Frank, 2022) oder gar zum “Generieren” eines ganzen Textes eingesetzt werden: Gibt man einen beliebigen Textanfang vor, kann das System die Wahrscheinlichkeiten möglicher Folgewörter berechnen und so das wahrscheinlichste nächste Wort ermitteln. Durch Wiederholung entsteht ein maschinell erzeugter Text. Kurze Teile davon sind in der Regel grammatikalisch einigermaßen okay, beispielsweise passen die Geschlechter von Artikeln und Nomen zusammen. Das ändert sich, wenn die Grammatik verlangt, dass im Satz Wörter in Beziehung stehen, die weiter entfernt sind als bei der Bestimmung der bedingten Wahrscheinlichkeiten berücksichtigt. Der Text ist deswegen in der Regel sinnlos. Die explizite Berechnung der relativen Häufigkeiten wird erschwert, je länger die Textstücke sind, da die Auftretenshäufigkeiten insgesamt gering sind. Es wäre also weder praktikabel noch sinnvoll, Folgen von

zehn Wörtern in den bedingten Wahrscheinlichkeiten zu berücksichtigen. Daher braucht es komplexere Methoden, beispielsweise KNNs.

Auch wenn die direkte Berechnung der bedingten Wahrscheinlichkeiten sehr aufwendig ist, liefert dies doch ein brauchbares Modell für ein grobes Verständnis von generativen Sprachmodellen wie ChatGPT: Basierend auf einem riesigen Datensatz an Texten lernt ein KNN, bei der Eingabe einer Folge von Wörtern das wahrscheinlichste nächste Wort vorherzusagen. Dazu wird der Eingabetext ("Prompt") in Token kodiert. Die daraus berechneten Embeddingvektoren sind dann die Inputdaten des Netzes. Für jedes mögliche Folge-Token gibt es ein Ausgabeneuron. Das Token mit dem höchsten Outputwert gilt als das Wahrscheinlichste. Ist der Text um ein Token verlängert, dient er als neue Eingabe. So wird der Text Token um Token generiert.

Dies ist ein grobes Modell auf der Mesoebene. Es ist trotz seiner Vereinfachungen nützlich, weil es einerseits Lernenden, die sich für die Funktion im Detail interessieren, einen Weg aufzeigt, auf die Mikroebene hinabzusteigen, und andererseits eine Einschätzung wichtiger Fragen auf der Makroebene ermöglicht – bspw. das Phänomen des "Halluzinierens" von ChatGPT oder etwa die Abhängigkeit von den Trainingsdaten.

Säule 5: Statistische Gütemaße

Die Testdaten dienen dazu, die Generalisierbarkeit des entwickelten ML-Modells auf neue, zuvor ungesehene Daten zu bewerten. Bisher wurde nur am Rande diskutiert, wie die Ergebnisse, die das entwickelte Modell auf den Testdaten liefert, quantifiziert und statistisch bewertet werden können. Welche statistischen Gütemaße geben hier aufschlussreiche Einblicke und wie sind diese zu interpretieren? Dies lässt sich auch ohne tiefergehendes Verständnis der mathematischen Grundlagen der jeweiligen ML-Methode (bspw. eines KNNs) im Unterricht diskutieren – wir bewegen uns auf der Makroebene. Im Zentrum dieser Diskussion steht die Erkenntnis, dass die Ergebnisse von ML-Methoden nur statistisch bewertet werden können. Je nach Problemklasse (Regressions- oder Klassifizierungsproblem) werden unterschiedliche statistische Gütemaße eingesetzt.

Klassifizierungsergebnisse werden oft in einer Wahrheitsmatrix (Konfusionsmatrix) zusammengefasst. Diese liefert eine kompakte Übersicht über die vorhergesagten und die tatsächlichen Klassenzuordnungen. Ein Beispiel mit fiktiven Ergebnissen eines Klassifizierungsmodells auf 300 Testdatenpunkten ist in Tabelle 1 dargestellt. Die fett gedruckten Werte auf der Diagonalen geben die Anzahl der korrekten Klassifikationen je Klasse an.

Tab. 1: Wahrheitsmatrix für die Ergebnisse eines Klassifizierungsmodells auf 300 Testdaten

	Vorhergesagt als Klasse A	Vorhergesagt als Klasse B	Vorhergesagt als Klasse C
Tatsächlich Klasse A	**7**	5	7
Tatsächlich Klasse B	22	**98**	2
Tatsächlich Klasse C	13	4	**142**

Typische statistische Gütemaße, die sich basierend auf der Wahrheitsmatrix leicht berechnen lassen, sind:

- **Genauigkeit**: Anteil der korrekten Klassifikationen an der Gesamtzahl aller Testdaten. Im Beispiel aus Tabelle 1 also $\frac{7+98+142}{300} = 0{,}82$.
- **Fehlerrate**: Anteil der Fehlklassifikationen an der Gesamtzahl aller Daten (d. h. 1 - Genauigkeit).
- **Präzision** (bzgl. einer Klasse): Anteil der korrekt als Klasse i klassifizierten Datenpunkte an der Anzahl aller als Klasse i klassifizierten Datenpunkte (Klasse A: $0{,}17$; Klasse B: $0{,}92$; Klasse C: $0{,}94$).
- **Recall** (auch Sensitivität): Anteil der korrekt als Klasse i klassifizierten Datenpunkte an der Anzahl aller zu Klasse i gehörenden Datenpunkte (Klasse A: $0{,}37$; Klasse B: $0{,}80$; Klasse C: $0{,}89$).

An der Wahrheitsmatrix aus Tabelle 1 wird schnell ersichtlich, warum es nicht ausreicht, ein Klassifizierungsmodell nur anhand der Gesamtgenauigkeit zu validieren. Dies ordnen wir am Beispiel der automatisierten Einstufung von Bewerber*innen für ein Studienstipendium ein. Die Bewerber*innen gehören einer der Klassen

“Ablehnen” (A), “Warteliste” (B) oder “Annehmen” (C) an.[8] Die Genauigkeit ist mit $82\,\%$ einigermaßen hoch (was als akzeptabel angesehen wird, hängt natürlich stark von der Anwendung ab); die Präzision für die Klasse “Ablehnen” ist jedoch gering. Personen werden somit abgelehnt und damit nicht für ein Stipendium berücksichtigt, obwohl sie dafür durchaus in Frage kämen. Auch der Recall ist für diese Klasse niedrig, d. h., es werden diverse Bewerber*innen ausgewählt, obwohl sie für das Stipendium eher nicht geeignet sind.

Ein Grund für derartige Ergebnisse könnte die Wahl des Trainingsdatensatzes sein. Wenn das Modell auf historischen Daten trainiert wurde, die Vorurteile enthalten oder aus Kontexten mit bestehenden oder früheren Ungleichverteilungen stammen, kann dies dazu führen, dass bestimmte Gruppen entweder über- oder unterrepräsentiert sind und Vorurteile bzw. Ungleichverteilungen reproduziert oder verstärkt werden. Beispielsweise könnten Absolvent*innen bestimmter Schulen oder Geschlechter systematisch benachteiligt werden, wenn deren Bewerbungen häufiger als „abgelehnt“ eingestuft wurden und weiterhin werden. Auch über derartige gesellschaftliche / ethische Implikationen sollte im Unterricht diskutiert werden (vgl. Orwat, 2019 für verschiedene Fallbeispiele). Im Notebook *Klassifikation_KNN.ipynb* können die Auswirkungen unausgeglichener Klassen auf die Klassifikationsergebnisse mithilfe der hier diskutierten statistischen Gütemaße untersucht werden.

Das gewählte Beispiel ist nicht nur mit Blick auf die Interpretation der statistischen Gütemaße, sondern auch darüber hinaus diskussionswürdig: Nach welchen Kriterien wurden die Klassenlabels festgelegt? Wurde allein der Erfolg im Studium als Kriterium verwendet, um die Trainingsdaten und damit die Studierenden zu “labeln”? Ist die Auswahl der Trainingsdaten und die Kodierung der Klassenlabels womöglich bereits fragwürdig? Zwar wurde in diesem Abschnitt nicht mit realen Daten argumentiert, verwandte Szenarien haben im Kontext der Bewerberauswahl oder der Zulassung von Studierenden an Universitäten jedoch in realen Anwendungen durchaus schon zu kritischen Diskussionen geführt (vgl. Orwat, 2019).

Die Bewertung von Klassifizierungsergebnissen bietet die Möglichkeit klassische schulmathematische Inhalte aus dem Bereich “Daten & Zufall” im Kontext von KI zu thematisieren und neu zu akzentuieren: Vierfeldertafeln und Fehler 1. und 2. Art. Die Präzision und der Recall stehen in direktem Zusammenhang mit diesen Fehlertypen, die typischerweise für zwei Klassen (positiv und negativ) betrachtet werden. Bei Problemen mit mehr als zwei Klassen kann dies wie folgt ausgeweitet werden: eine ausgewählte Klasse i wird als positiv und alle übrigen Klassen zusammengenommen als negativ betrachtet. Eine niedrige Präzision für Klasse i bedeutet, dass viele Daten falsch der Klasse i zugeordnet wurden (falsch Positive) und geht mit einem hohen Fehler 1. Art einher. Ein niedriger Recall bedeutet, dass viele Datenpunkte der Klasse i einer anderen Klasse zugeordnet wurden (falsch Negative) und geht mit einem hohen Fehler 2. Art einher.

Für die Bewertung von *Regressionsergebnissen* werden bspw. die mittlere quadratische Abweichung und die mittlere absolute Abweichung verwendet. Auch hierzu ließen sich interessante Diskussionen führen; bspw. über die höhere Sensitivität der mittleren quadratischen Abweichung für Ausreißer oder die leichtere Interpretierbarkeit der mittleren absoluten Abweichung, da diese die durchschnittliche Größe des Fehlers in der Einheit der Outputdaten angibt.

Didaktische Einordnung

Oben wurde auf elementarer mathematischer Ebene (*Mikroebene*) erläutert, wie künstliche neuronale Netze mathematisch modelliert werden können. Neben dieser Detailbetrachtung sind auch darüber liegende Betrachtungsebenen, die Meso- und Makroebene, möglich.

Auf der *Mesoebene* geht es um das Zusammenspiel der elementaren Objekte der Mikroebene. Die Beschreibung eines neuronalen Netzes als vielfach verkettete Funktion, die sich aus affin-linearen und nichtlinearen Funktionen zusammensetzt, oder die Beschreibung der Schritte des Word-Embeddings liegen auf dieser Ebene.

Auf der *Makroebene* werden Details der mathematischen Beschreibung und programmtechnischen Umsetzung der einzelnen Komponenten einer ML-Methode (bspw. Neuronen) und des Trainierens nicht betrachtet. Stattdessen befasst man sich ausführlicher mit dem System als Ganzes (bspw. ein KNN als Funktion, die

[8] Hier ließe sich auch der klassische Anwendungsfall der medizinischen Diagnose bemühen (Klasse A = Krankheit A, Klasse B = Krankheit B, Klasse C = gesund).

Inputdaten gewissen Outputdaten zuordnen), mit der Bewertung der Performanz basierend auf statistischen Gütemaßen und mit der Auswahl und dem Einfluss von Trainings- und Testdaten auf die Ergebnisse. Auch die Diskussion von Problemen des Systems und mögliche, resultierende Auswirkungen auf die Gesellschaft bzw. auf einzelne Personengruppen zählen wir zu dieser Ebene.

In Tabelle 2 werden am Beispiel von KNNs die drei Betrachtungsebenen, relevante mathematische Inhalte, der jeweilige Bildungswert und der Beitrag zum Empowerment aufgeschlüsselt. Diese Einordnung ließe sich in ähnlicher Weise für weitere ML-Methoden realisieren.

Tab. 2: Ebenen der Durchdringung von KI-Systemen am Beispiel von KNNs mit Einordnung des Bildungswertes

Ebene	Mathematik	Bildungswert	Empowerment
Mikroebene	Skalarprodukt, Matrix-Vektor-Multiplikation, euklidischer Abstand, Optimierungsproblem und numerische Optimierungsverfahren	• Innermathematisch • Anwendung und Vertiefung von Schulmathematik an relevanten Fragestellungen • Demystifizierung von KI	Teilkomponenten eines KNNs implementieren (d. h. Implementierung “from Scratch”)
Mesoebene	Künstliches Neuron als Funktion, die sich aus der Verkettung einer linearen mit einer nichtlinearen Funktion zusammensetzt Zusammenspiel von Schichten in einem KNN als Verkettung von Funktionen	• Innermathematisch • Anwendung und Vertiefung von Schulmathematik an relevanten Fragestellungen • Komplexitätsbeherrschung durch Modularisierung • Demystifizierung von KI	KNN auf Basis der Nutzung von Softwarepaketen (bspw. PyTorch, Tensorflow) implementieren
Makroebene Einzelnes KI-System Gesellschaft	KNN als Funktion, die einem Input einen Output zuordnet Einfluss von Trainings- und Testdaten; Validierung mit statistischen Gütemaßen	• Bias, Diskriminierung, Modellkritik • Ethische / gesellschaftliche Fragestellungen diskutieren	Analysieren und Bewerten der Leistung eines trainierten KNN KI-Anwendungen und ihre Rolle in und für unsere Gesellschaft kritisch reflektieren

Es stellt sich die didaktische Frage, auf welcher Ebene allgemeinbildender Unterricht zur KI ansetzen sollte. Die Makroebene zielt darauf ab, Schüler*innen zu befähigen, Outputs von KI-Systemen kritisch einzuordnen und deren gesellschaftliche Auswirkungen einzuschätzen. Damit ist offensichtlich, dass die Makroebene eine zentrale Rolle bei der Lebensvorbereitung in einem KI-getriebenen Alltag spielt und damit gemäß Heymann (1989) allgemeinbildenden Wert hat. Zu klären bleibt, ob diese Ebene unabhängig von den beiden darunter liegenden Ebenen in einer Art und Weise im Unterricht behandelt werden kann, die nachhaltige Bildung ermöglicht und damit junge Menschen in die Lage versetzt, KI-Systeme reflektiert und zielführend zu nutzen, zu bewerten und selbst zu gestalten.

Unsere Hypothese ist, dass Wissen und Kompetenzen der unteren Ebenen durchaus auf die Makroebene durchschlagen. Werden KNNs auf der Mikro- oder Mesoebene erarbeitet, so wird greifbar, dass bei der Entwicklung von KI-Systemen mit ML-Methoden oftmals verschiedene Modellentscheidungen denkbar sind und von Menschen getroffen werden, bspw. die Wahl der Metrik zur Definition der Fehlerfunktion eines KNNs. Es wird deutlich, dass die Entwicklung von KI-Systemen ganz wesentlich auf Daten und elementarer Mathematik und mathematischer Modellierung beruht (gepaart mit sehr effizienten Implementierungen). Die Mikro- und Mesoebene spielen damit eine besondere Rolle bei der Demystifizierung von KI.

Weitere Beispiele, wie Wissen aus der Mikro- und Mesoebene auf der Makroebene relevant wird: Angenommen, ein großes Sprachmodell gibt fälschlicherweise aus, dass eine Person im Jahr 1980 geboren sei. Der/Die Nutzer*in beschwert sich. Dann kann der Anbieter des Sprachmodells, anders als bei Datenbanken, nicht einfach eine Zahl ändern, weil die 1980 nicht an einer einzigen Stelle codiert ist. Wegen der Komplexität des Netzes sind die Gewichte nicht interpretierbar und das Zustandekommen einer spezifischen Ausgabe ist für

Menschen nicht nachvollziehbar. Diese Erkenntnis ist durch ein tieferes Verständnis der mathematischen Struktur eines KNNs auf der Mikro- oder Mesoebene möglich. Das Verständnis von KNNs (oder N-Gramms) auf der Mesoebene erlaubt es zudem zu verstehen, warum große Sprachmodelle Logikfehler machen, teilweise bei simplen Rechenaufgaben versagen oder “halluzinieren”: Diese Systeme liefern basierend auf gegebenen Trainingstexten mithilfe von mathematischen Modellen eben nur Näherungen für wahrscheinliche Tokens (und damit Wörter). Aufgrund ihrer nicht völligen Zuverlässigkeit sind große Sprachmodelle also in einem gewissen Sinne das Gegenstück zur Mathematik, die Sicherheit und maximale Transparenz der Begründung anstrebt.

Diese Beispiele zeigen exemplarisch, dass technische Bildung auf Mikro- und Mesoebene zu einer kompetenten Einschätzung eines komplexen KI-Systems befähigt und auf die Makroebene durchschlägt.

Fazit

Die Gestaltung von Mathematikunterricht zu den in diesem Beitrag behandelten Themen eröffnet Chancen, steht aber auch vor Herausforderungen.

Einerseits kann das Thema die Bedeutung von (Schul-)Mathematik für die Entwicklung von KI-Anwendungen im Mathematikunterricht betonen. Die Forschung und Entwicklung im Bereich KI lässt sich eben nicht nur der Informatik zuordnen. Auch die Mathematik leistet einen wesentlichen Beitrag und es gibt durchaus zahlreiche offene Forschungsfragen im Kontext des maschinellen Lernens, an denen in der Mathematik derzeit aktiv geforscht wird (Kutyniok, 2024). Positiv ist auch, dass das Thema eine Reihe von mathematischen Teilgebieten verbindet, etwa Analysis, Vektorrechnung und Wahrscheinlichkeitsrechnung. Der Unterricht kann damit deutlich machen, dass Mathematik bei realen Anwendungen vielfältig vernetzt Einsatz findet. Als verbindendes Element ist zudem die mathematische Modellierung hervorzuheben. Bei der Diskussion von ML-Methoden auf der Mikro- oder Mesoebene wurde an diversen Stellen ersichtlich, dass oft verschiedene Modellentscheidungen möglich sind. Dies betrifft etwa die Kodierung der Klassenlabels bei Klassifizierungsproblemen, die Wahl des Ähnlichkeits- oder Distanzmaßes bei der k-nächste-Nachbarnmethode, die Festlegung der Zielfunktion bei der Optimierung von KNNs oder die Wahl der Gesamtstruktur eines KNNs (Anzahl Schichten etc). Dies kann zur Erkenntnis beitragen, dass zahlreiche Entscheidungen bei der Entwicklung von KI-Systemen eben doch von Menschen getroffen werden und diese Systeme sich nicht voll autonom einstellen und “selbst entwickeln”. Damit besteht die Chance zur Demystifizierung von KI beizutragen.

Eine Herausforderung ist die Behandlung von Funktionen in mehr als einer Variablen, die in den diskutierten Beispielen auftreten. Da diese im Mathematikunterricht aber ohnehin implizit vorkommen (zum Beispiel als Formel für das Pyramidenvolumen) und in Tabellenkalkulationen genutzt werden, erscheint eine unterrichtliche Behandlung konsequent (vgl. dazu auch Schweiger, 2023).

Die technologiegestützte Umsetzung von Unterrichtseinheiten zu den mathematischen Hintergründen von KI-Systemen auf der Mikro- oder Mesoebene erfordert ein gewisses Maß an informatischer Bildung, sobald mehr Eigenaktivität der Lernenden ermöglicht werden soll. Dies stellt aktuell noch eine Herausforderung für den Mathematikunterricht dar. Da Informatik als Schulfach jedoch mittlerweile in mehr und mehr Ländern verpflichtend eingeführt wird, sollte dieses Problem auf Seiten der Lernenden mit der Zeit kleiner werden. Zugleich bedarf es auch eines Umdenkens in der Aus- und Weiterbildung von Mathematiklehrkräften, die Gelegenheit bekommen müssen, selbst informatische Grundkenntnisse zu erwerben.

Aus der Perspektive der Allgemeinbildung ist an diesem Thema reizvoll (mit Blick auf das Lernen in abgegrenzten Schulfächern zugleich herausfordernd), dass neben Mathematik und Informatik auch weitere Bildungsbereiche involviert sind: Dies betrifft die Biologie, die das Vorbild für die mathematische Modellierung von neuronalen Netzen geliefert hat und die durch Forschung in der Bioinformatik auch wesentlich von KI-getriebenen Innovationen beeinflusst wird (bspw. AlphaFold). Es betrifft auch die Physik, denn zahlreiche KI-Systeme werden basierend auf Sensordaten entwickelt (bspw. Fitness-Tracker, aber auch KI-Systeme in autonom fahrenden Autos) – ein grundlegendes physikalisches Verständnis ist beim Umgang mit diesen Daten hilfreich. Offensichtlich ist auch die Relevanz gesellschaftswissenschaftlicher Fächer und der Ethik für KI-Bildung. Hier ergeben sich u. a. Fragen nach der Verantwortung, Datenschutz, Privatsphäre und Diskriminierung. Auch philosophische Fragen ergeben sich, bspw. ob der Erfolg von großen Sprachmodellen zeigt, dass man Lernen auf das Konsumieren von viel Text reduzieren kann.

Es lässt sich argumentieren, dass ein detailliertes technisches Verständnis über KI nicht notwendig sei, weil man auch nichts über Motoren wissen müsse, um Auto zu fahren. Aber selbst in dieser Metapher zeigt sich, dass die Mesoebene relevant ist: Aus Wissen über die Eigenschaften von Verbrennungs- und Elektromotoren kann man etwa ableiten, wie man energiesparend fährt, oder dass im Stadtverkehr ein Elektroauto Effizienzvorteile hat. Selbstverständlich muss man dieses Wissen nicht unbedingt selbst herleiten, sondern kann es von Expert*innen übernehmen. Es ist jedoch allemal effizienter, einige wenige Grundprinzipien zu erlernen und daraus Schlüsse zu ziehen, als viele Einzelfakten von Expert*innen zu übernehmen – denen man zudem vertrauen muss. Wir sind deswegen überzeugt, dass auch mathematische Grundlagen der KI in allgemeinbildende Schulen gehören, wenn diese der Forderung von Hartmut von Hentig (2002) genügen sollen, dass die Menschen der technischen Zivilisation gewachsen bleiben sollen.

Literatur

Biehler, R., Schönbrodt, S., & Frank, M. (2024). KI als Thema für den Mathematikunterricht. *Mathematik lehren*, 244, 2–7.

Biehler, R., & Fleischer, Y. (2021). Introduction students to machine learning with decision trees using CODAP and Jupyter Notebooks. *Teaching Statistics*, 43, 133–142. https://doi.org/10.1111/test.12279

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*(4). 303–314. https://doi.org/10.1007/BF02551274

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). Mathematics for machine learning. Cambridge University Press. https://mml-book.github.io/book/mml-book.pdf

Dvir, M., Podworny, S., Ben-Zvi, D., & Frischemeier, D. (2022). The multidimensional pedagogical potential of data modeling. In S. Podworny, D. Frischemeier, M. Dvir, D. Ben-Zvi (Hrsg.), *Reasoning with data models and modeling in the big data era* (S. 7–13). Minerva School 2022. https://doi.org/10.17619/UNIPB/1-1815

Engel, J. (2017). Statistical literacy for active citizenship: A call for data science education. *Statistics Education Research Journal*, *16*(1), 44–49. https://doi.org/10.52041/serj.v16i1.213

Ernest, P. (2002). Empowerment in Mathematics Education. *Philosophy of Mathematics Education Journal*, *15*.

European Commission, Joint Research Centre, Vuorikari, R., Kluzer, S., Punie, Y. (2022). DigComp 2.2, The Digital Competence framework for citizens – With new examples of knowledge, skills and attitudes. Publications Office of the European Union. https://data.europa.eu/doi/10.2760/115376

Gould, R. (2021). Towards data-scientific thinking. *Teaching Statistics*, 43, S. S11–S22. https://doi.org/10.1111/test.12267

Gould, R., Machado, S., Ong, C., Johnson, T., Molyneux, J., Nolen, S., Tangmunarunkit, H., Trusela, L., & Zanontian, L. (2016). Teaching data science to secondary students – the mobilize introduction to data science curriculum. In: J. Engel (Hrsg.), Promoting understanding of statistics about society. Proceedings of the IASE Roundtable Conference. http://dx.doi.org/10.52041/SRAP.16402

Hazzan, O., & Mike, K. (2022). Teaching core principles of machine learning with a simple machine learning algorithm: the case of the KNN algorithm in a high school introduction to data science course. *ACM Inroads*, *13*(1), 18–25. https://doi.org/10.1145/3514217

Hentig, H. v. (2002). Der technischen Zivilisation gewachsen bleiben. Nachdenken über die Neuen Medien. Beltz.

Heymann, H. W. (1989). Allgemeinbildender Mathematikunterricht – was könnte das sein? *Mathematik lehren*, 33, 4-9.

Hofmann, S., & Frank, M. (2022). Teaching data science in school: Digital learning material on predictive text systems. In J. Hodgen, E. Geraniou, G. Bolondi, & F. Ferretti (Hrsg.), *Proceedings of CERME12*. https://hal.science/hal-03751829

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Kindler, S., Schönbrodt, S., & Frank, M. (2023). From school mathematics to artificial neural networks: Developing a mathematical model to predict life expectancy. In P. Drijvers, H. Palmér, C. Csapodi, K. Gosztonyi, & E. Kónya (Hrsg.), *Proceedings of CERME13* (S. 956–963). https://hal.archives-ouvertes.fr/hal-04410971

Kutyniok, G. (2024, 5. Juli). The mathematics of reliable artificial intelligence. SIAM News. www.siam.org/publications/siam-news/articles/the-mathematics-of-reliable-artificial-intelligence/

Bundesministerium für Bildung, Wissenschaft und Forschung (2024, 18. Juli). Gesamte Rechtsvorschrift für Lehrpläne – allgemeinbildende höhere Schulen. www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10008568

Staatsinstitut für Schulqualität und Bildungsforschung München (2022). Lehrplan Informatik 13 Bayern. www.lehrplanplus.bayern.de/fachlehrplan/lernbereich/313771

Levy, A. Shalom, B. R., & Chalamish, M. (2024). A Guide to Similarity Measures. ArXiv https://doi.org/10.48550/arXiv.2408.07706

Ministerium für Schule und Weiterbildung des Landes Nordrhein-Westfalen (2021). Kernlehrplan für die Sekundarstufe I – Klasse 5 und 6 in Nordrhein-Westfalen: Informatik. www.schulentwicklung.nrw.de/lehrplaene/lehrplan/256/si_kl5u6_if_klp_2021_07_01.pdf

Oldenburg, R. (2008). Phrasendreschmaschine und Text-Evolution: Unterrichtsideen für Zeichenketten mit PYTHON. *Log In*, 154/155, 91–98.

Oldenburg, R. (2009). Was kosten Ferienhäuser? *MNU*, 62, 340–341.

Oldenburg, R. (2011). Mathematische Algorithmen im Unterricht. Teubner. https://doi.org/10.1007/978-3-8348-8336-0

Oldenburg, R. (2021). Big data – small school: oder: Likert-Skalen, Kaufempfehlungen, soziale Netzwerke, das Skalarprodukt und all das. In H. Humenberger & B. Schuppar (Hrsg.): Neue Materialien für einen realitätsbezogenen Mathematikunterricht 7, S. 137-142, Springer Spektrum. https://doi.org/10.1007/978-3-662-62975-8_12

Orwat, C. (2019). Risks of Discrimination through the Use of Algorithms. A study compiled with a grant from the Federal Anti-Discrimination Agency. www.antidiskriminierungsstelle.de/SharedDocs/downloads/EN/publikationen/Studie_en_Diskriminierungsrisiken_durch_Verwendung_von_Algorithmen.pdf?__blob=publicationFile&v=2, Zugegriffen: 19.08.2024

Rahwan, I., Cebrian, M., Obradovich, N. et al. (2019). Machine behaviour. *Nature*, 568, 477–486. https://doi.org/10.1038/s41586-019-1138-y

Schönbrodt, S., Camminady, T., & Frank, M. (2022). Mathematische Grundlagen der Künstlichen Intelligenz im Schulunterricht – Chancen für eine Bereicherung des Unterrichts in linearer Algebra. *Mathematische Semesterberichte*. 69, 73–101. https://doi.org/10.1007/s00591-021-00310-x

Schönbrodt, S., Hoeffer, K., & Frank, M. (2023). AI education as a starting point for interdisciplinary STEM projects. In P. Drijvers, H. Palmér, C. Csapodi, K. Gosztonyi, & E. Kónya (Hrsg.), *Proceedings of CERME13* (S. 4703–4710). https://hal.science/hal-04420534

Schönbrodt, S., & Frank, M. (2024). Wie viel Mathe steckt in mathematischer Modellierung? – Eine Antwort am Beispiel der Optimierung. *Der Mathematikunterricht*, *70*(1), 35–44.

Schweiger, F. (2023). Funktionen in mehreren Variablen – ein Plädoyer. *Mathematik im Unterricht*, 14, 112–115. https://doi.org/10.25598/miu/2023-14-9

Sindermann, C., Yang, H., Elhai, J. D. et al. (2022). Acceptance and Fear of Artificial Intelligence: associations with personality in a German and a Chinese sample. *Discov Psychol,* 2(8). https://doi.org/10.1007/s44202-022-00020-y

Stern, W. (1911). Die Differentielle Psychologie in ihren methodischen Grundlagen. Barth.

UNESCO (2024). AI competency framework for students. https://doi.org/10.54675/JKJB9835

Adressen der Autorin und des Autors:
Ass.-Prof. Dr. Sarah Schönbrodt
Fachbereich Mathematik, AG Didaktik der Mathematik
Paris Lodron Universität Salzburg
Hellbrunnerstr. 34
5020 Salzburg
sarah.schoenbrodt@plus.ac.at

Prof. Dr. Reinhard Oldenburg
Lehrstuhl für Didaktik der Mathematik
Universität Augsburg
Universitätsstr. 14
86159 Augsburg
reinhard.oldenburg@math.uni-augsburg.de