Grundlagen des maschinelles Lernens - Möglichkeiten zur Anwendung und Vertiefung des Mathematik-Lernstoffes in der Sekundarstufe II

E. v. Berg

Entwurf - Stand Sommer 2024

1 Einleitung

Kenntnisse über die Grundlagen des maschinellen Lernens sind wegen der zunehmenden Durchdringung des beruflichen, schulischen und privaten Lebens mit KI-Methoden von Nutzen.

Bei einem näheren Blick auf die mathematischen Grundlagen der beim maschinellen Lernen verwendeten Verfahren zeigt sich, dass sich ein Teil davon bereits mit den Mathematik-Schulkenntnissen der Sekundarstufe II zumindest im Prinzip und für einfache Anwendungsfälle verstehen und auch mit einer geeigneten Programmiersprache algorithmisch umsetzen und testen lässt.

Daher ist dieses Thema eventuell auch für eine Mathematik/Informatik-Projektwoche oder die letzte Schulwoche vor den Ferien geeignet. Es kann aber auch gut verteilt im Verlauf des Schuljahres zur Vertiefung von Stoffgebieten wie lineare Gleichungen, Exponential- und Logarithmusfunktionen sowie Differenzialrechnung oder Wahrscheinlichkeitsrechnung im Rahmen des regulären Unterrichts verwendet werden.

Die Beschäftigung mit den Grundlagen des maschinellen Lernens kann u.a. auch bei der Beantwortung der Frage vieler Schüler hilfreich sein, wozu man denn manche, dem Alltagsleben so fern erscheinende Mathematikthemen wie z.B Logarithmusfunktionen oder die Kettenregel der Differenzialrechung überhaupt im Mathematikunterricht behandelt. Zumindest für einige Schülerinnen und Schüler mag das die Motivation und das Interesse am Mathematikunterricht erhöhen, weil damit der Zusammenhang zwischen Mathematik, Informationstechnik und Alltagsleben exemplarisch sichtbar wird.

Durch die einfache programmtechnische Umsetzung kann man das Thema dann in gewissem Sinne auch selbst in die Hand nehmen, indem man eigene numerische Experimente zur Rolle der Modellparameter und der für die Lernalgorithmen verwendeten Traingsdaten durchführt. Das kann dazu beitragen, dass Möglichkeiten und Grenzen der KI-Tools besser eingeschätzt und für eigene Fragestellungen zweckdienlich und verantwortlich eingesetzt werden - ganz im Sinne der bekannten englischen Redewendung: „Hear and forget, see and remember, do and understand. “

2 Kurzer Überblick über Aufgaben und Begriffe im Bereich des maschinellen Lernens

Beim maschinellen Lernen unterscheidet man zwei wichtige Aufgabenstellungen. Das sind einerseits die Klassifizierungsaufgaben, wobei Daten in Gruppen eingeteilt und Entscheidungen über die Zugehörigkeit zu den einzelnen Gruppen bzw. Klassen zu treffen sind. Es kommt dabei eine Ja/Nein Entscheidung oder auch eine Wahrscheinlichkeitsaussage über die jeweiligen Klassenzugehörigkeiten heraus. Ein Beispiel dafür sind Spamfilter bei E-Mails, die erkennen müssen, welche Mails Spam und welche Mails unproblematisch sind. Ein weiteres Beispiel ist die Erkennung handgeschriebener Zahlen oder Buchstaben, z.B. auf Banküberweisungsformularen,

die maschinell gelesen werden. Dabei sind nicht mehr nur zwei Klassen wie beim Spam-Beispiel zu unterscheiden, sondern 10 Möglichkeiten bei den Ziffern und mehr als 26 bei den Buchstaben.

Der andere wichtige Aufgabentyp beim maschinellen Lernen umfasst die Regressionsaufgaben. Dabei wird anhand von Eigenschaften eines Datensatzes durch Einordnung in eine Menge von Trainingsdatensätzen eine zahlenmäßige Abschätzung vorgenommen, die den neu vorgegebenen Datensatz quantitativ beschreibt. Ein Beispiel dafür ist etwa die Voraussage von Hauspreisen aufgrund von charakteristischen Eigenschaften von Immobilien, wie z.B. Wohnfläche, Grundstücksgröße, Zimmerzahl, Alter und Wohngegend. Es kommt dabei als Ergebnis ein Zahlenwert heraus. Ein anderes Beispiel ist die Wettervorhersage, wo Temperaturen und Regenwahrscheinlichkeiten mit Hilfe von Wettermodellen und Datenbanken über ähnliche Wetterlagen aus der Vergangenheit vorhergesagt bzw. abgeschätzt werden.

Eine weitere wichtige Unterscheidung beim maschinellen Lernen betrifft die Lernarten, wobei man zwischen überwachtem Lernen, unüberwachtem Lernen, teilüberwachtem Lernen und verstärkendem Lernen unterscheidet. Das überwachte Lernen ist dadurch charakterisiert, dass der jeweilige Lernalgorithmus zuächst mit einer großen Zahl von Datensätzen, deren Klassifikation bereits bekannt ist, trainiert wird. Durch die Verarbeitung dieser Daten lernt der Algorithmus darin enthaltene Muster und erstellt ein Modell - meist in Form eines mehr oder weniger komplizierten mathematischen Ausdrucks - mit dem danach ungekennzeichnete Datensätze möglichst zutreffend klassifiziert werden können. Mit überwachtem Lernen können sowohl Klassifikations- als auch Regressionsaufgaben gelöst werden. Beispiele von überwachten Lernverfahren, die unten näher betrachtet werden, sind das K-Nearst-Neighbour-Verfahren, die künstlichen neuronalen Netze, die logistische Regression, das Verfahren der Support-Vector-Machine sowie Entscheidungsbäume.

Beim unüberwachten Lernen werden keine gekennzeichneten Trainingsbeispiele vorgegeben, sondern es geht darum, dass der Algorithmus eigenständig Muster in den Daten aufspürt und eine Klasseneinteilung der Daten in sogenannte Cluster vornimmt. Dazu müssen die Unterschiede zwischen den Daten durch eine Maßzahl, z.B. in der Art eines verallgemeinerten Abstandes zwischen den Datenpunkten ermittelt und kategorisiert werden. Ein Verfahren dieses Typs ist der K-Means-Algorithmus, der in einem folgenden Kapitel genauer betrachtet wird. Die abstandsbasierten Verfahren funktionieren allerdings nur dann gut, wenn die Daten in konvexen Gruppierungen vorliegen, die sich möglichst wenig durchdringen sollten. Für die Erkennung von Daten-Clustern beliebiger Form eignen sich die sogenannten dichtebasierten Verfahren besser, die zunächst Punkte mit einem Abstandskriterium sammeln und dann noch die Ränder der gefundenen Gebiete abfahren und mit dem Abstandskriterium weitere Datenpunkte hinzufügen. Dadurch können auch nicht-konvexe Anordnungen aufgefunden werden (vgl. z.B. [1]).

Andere wichtige Lernarten sind das teilüberwachte Lernen, bei dem der Lernalgorithmus zur Clusterfindung mit einer kleinen Menge von gekennzeichneten Daten vortrainiert wird, und das sogenannte verstärkende Lernen (Reinforcement learning), bei dem einem Apparat oder einem Roboter durch Kontakte mit der seiner Umgebung, die teils belohnt und teils bestraft werden, ein optimales Verhaltensmuster im Sinne maximaler Belohnung antrainiert wird. Diese beiden Lernverfahren werden im Folgenden nicht weiter behandelt.

Weitere Informationen und genauere Erklärungen zu wichtigen Begriffen des maschinellen Lernens findet man z.B. in den Büchern von Burkov [2], Grunert [3], Fischer [1], Otte [5], Cleve [6], Frochte [7] oder Ertel [8]. Der Schwerpunkt dieses Aufsatzes liegt aber auf der vereinfachten, aber doch das jeweilige Grundprinzip vermittelnden Darstellung und programmtechnischen Umsetzung einiger wichtiger Lernverfahren, soweit das im Rahmen des Oberstufenunterrichts möglich ist. Dazu werden auch aus der verwendeten Programmiersprache Python nur sehr einfache und elementare Konstrukte verwendet, die rasch im schulischen Rahmen vermittelt

werden können [1]. Die verwendeten Programme sind als Zip-Datei zum Download als Referenz auf derselben Webseite wie der vorliegende Aufsatz verfügbar. Bei der Umsetzung ist es aber zweckmäßig eigene Programme zu erstellen.

3 Einige wichtige Lernalgorithmen

Die für das maschinelle Lernen aktuell verwendeten Methoden und Algorithmen sind oftmals sehr hochentwickelt, anspruchsvoll und komplex. Wie einige der im vorigen Kapitel genannten Lernmethoden zumindest prinzipiell funktionieren, kann man aber doch bereits mit der Schulmathematik erklären. Das wird im Folgenden am Beispiel von einfachen Klassifizierungsproblemen, wie z.B. der Unterscheidung von Punkten, die ober- und unterhalb einer Grenzkurve in der Ebene liegen, oder der Clusterbildung bei konkaven Punktwolken in der Ebene durchgeführt.

3.1 K-Nearest-Neighbour-Lernmethode

Das K-Nearest-Neighbour-Verfahren funktioniert so, dass alle Trainingsdaten und ihre Kennzeichnung gespeichert werden und für jeden neuen zu klassifizierenden Testpunkt die k nächsten Trainingsdatenpunkte gesucht werden. Dazu braucht man eine Vorschrift zur Berechnung des Abstands zwischen zwei Datensätzen. Eine einfache Abstandsformel ist der sogenannte Euklidische Abstand, den man für die Länge von Vektoren in der Ebene oder im Raum verwendet und der als Wurzel aus der Quadratsumme der Koordinaten des Vektors definiert ist. Wenn man diesen Abstand auf mehr als 2 oder 3 Merkmale erweitert, bekommt man

$$a = \sqrt{\sum_{i=1}^{n} (x_{ti} - x_i)^2} \qquad \text{mit n gleich Anzahl der Merkmale.} \tag{1}$$

Es werden dann die Abstände zwischen dem Testpunkt und allen Trainingsdaten berechnet. Anschließend werden die Abstandsdaten der Größe nach sortiert und die k Datenpunkte bestimmt, die den kleinsten Abstand zum Testpunkt haben. Danach werden die Merkmalswerte dieser k nächsten Trainingsdaten gemittelt und dem Testpunkt als Merkmal zugewiesen. Bei binären Ja/Nein-Merkmalen, die mit 1 und 0 codiert sind, wird der Mittelwert gerundet. Das führt bei ungeradem k immer auf eine eindeutige Ja/Nein bzw. 1/0-Zuordnung. Vorteile des KNN-Verfahrens sind nach [3], dass es intuitiv und einfach zu verstehen ist und schnell an Trainingsdaten angepasst werden kann. Weiter ist es auch für Probleme mit mehreren Klassen als Zielgröße geeignet und kann sowohl für Klassifizierungs- als auch für Regressionsaufgaben verwendet werden. Nachteile sind u.a., dass das Verfahren bei großen Datenmengen mit mehreren Merkmalen langsam wird, weil immer mit allen Trainingsdaten verglichen werden muss und es keine einfache Modellgleichung für die Vorhersage gibt. Als weiterer Nachteil wird genannt, dass es schwierig sein kann, die optimale Zahl von Nachbarn zu wählen.

Zum Testen des Verfahrens wurde ein kleines Python-Programm für die Zuordnung von Punkten der Ebene im Intervall [-1,1]x[-1,1] unterhalb und oberhalb der Geraden $x_2 = 2 \cdot x_1 - 0.5$ mit den folgenden Berechnungsschritten erstellt:

1. Trainingsdaten erzeugen

[1] Die Einfachheit liegt allerdings auch an den sehr bescheidenen Python-Kenntnissen des Verfassers, der Verbesserungsvorschläge daher gerne entgegennimmt.

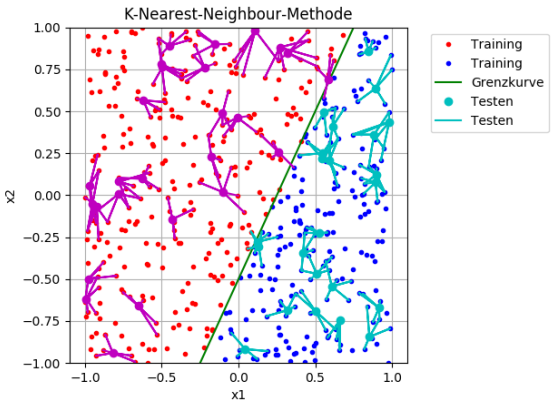

Abb. 1: Klassifizierung von Punkten in der Ebene durch die KNN-Methode mit Gerade als Grenzkurve

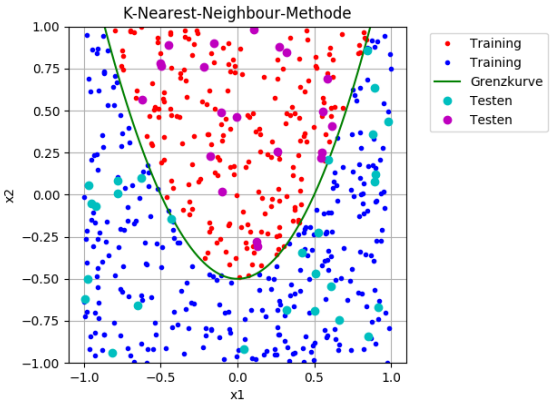

Abb. 2: Klassifizierung von Punkten in der Ebene durch die KNN-Methode mit Parabel als Grenzkurve

2. Testpunkt erzeugen
3. Berechnung der Abstände von allen Trainingsdaten zum Testpunkt
4. Sortieren der Abstände nach der Größe
5. Addieren der Kennzeichnungen der k nächsten Nachbarn und Mittelwertbildung
6. Runden des Mittelwerts, um die Kennzeichnung des Testpunkts zu erhalten

Abbildung 1 zeigt das Ergebnis des KNN-Verfahrens für das Testproblem. Es wurden 500 zufällig verteilte Trainingsdaten verwendet, die mit kleinen rot bzw. blau markierten Punkten gekennzeichnet sind. Die 50 Testdaten sind mit magenta- bzw. cyanfarbigen Punkten für oberhalb bzw. unterhalb markiert, die alle richtig zugeordnet werden. Auch für eine parabelförmige Grenzkurve $x_2 = 2 \cdot x_1^2 - 0.5$ gelingt das ohne Fehler, wie man in Abbildung 2 sieht. Im gezeigten Beispiel wurden 5 nächste Nachbarn für die Bestimmung der passenden Zuordnung gewählt. Genauere Erklärungen und mehr Beispiele auch für Klassifizierung mit mehreren Merkmalen findet man in [10] und [11].

3.2 Neuronale Netze

Weil künstliche neuronale Netze ein zentraler Baustein von KI-Methoden sind, die auch dem Deep-Learning und dem Chat-GPT-Sprachmodell zugrunde liegen, soll im Folgenden ein einfaches kleines neuronales Netz, das aus nur 3 Neuronen besteht, genauer untersucht werden. Zuerst wird der Aufbau des Netzes erklärt und danach die Signalweiterleitung durch das Netz in Vorwärtsrichtung beschrieben und der Vorhersagefehler ermittelt. Anschließend wird die Korrektur der Netzgewichte mit dem Verfahren des Gradientenabstiegs und der Fehler-Backpropagation-Lernregel durchgeführt. Diese Modellierungsschritte werden in einem Python-Programm umgesetzt und es wird damit das Netz für zwei einfache Klassifizierungsaufgaben trainiert und danach getestet.

Künstliche neuronale Netze sind nach dem biolgischen Vorbild der Nervenzellen im menschlichen Gehirn aufgebaut (s. Abbildung 3), wobei aber starke Vereinfachungen gemacht werden. Die künstlichen neuronalen Netze bestehen aus verschiedenen Schichten, und zwar aus der Eingabeschicht, der Ausgabeschicht und einer oder mehreren dazwischen liegenden Schichten.

Die dazwischen liegenden Schichten werden als verborgene Schichten bezeichnet. Der prinzipielle Aufbau ist in Bild 4 gezeigt. Netze, die die Eingabedaten nur in einer Richtung vom Eingang zum Ausgang durch das Netz schicken, heißen Feed-Forward-Netze. Die Neuronen einer Schicht sind dabei nur mit den Neuronen der vorhergehenden und der nachfolgenden Schicht verbunden. Die Stärke der Verbindung kann mit den in Bild 4 mit dem Buchstaben w bezeichneten Parametern, den sogenannten Gewichten, eingestellt werden.

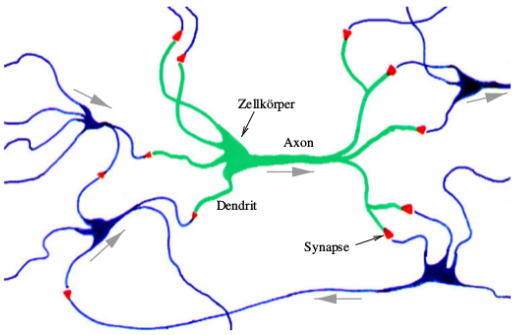

Abb. 3: Nervenzelle in der Biologie und Vernetzung mit weiteren Zellen nach [13]

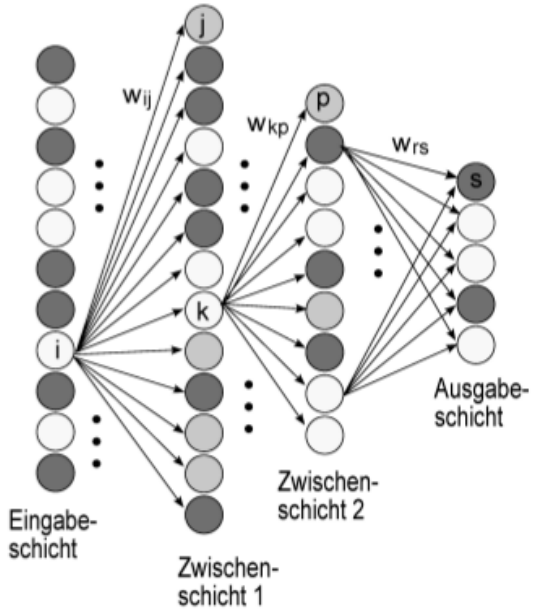

Abb. 4: Künstliches neuronales Netz mit Eingabeschicht, Ausgabeschicht und mehreren verborgenen Zwischenschichten nach [14]

Neuronale Netze eignen sich gut für Klassifizierungsaufgaben. Dazu muss das Ausgabeneuron oder die Ausgabeneuronen jeweils den Wert 0 oder 1 annehmen, um eine Auswahl zwischen den Ergebnisklassen vorzunehmen, die den Ausgabeneuronen zugewiesen sind. Für das unten behandelte Beispielnetz ist das nur eine einzige Entscheidung, die festlegt, ob ein Punkt in der Ebene oberhalb oder unterhalb einer bestimmten Grenzkurve liegt. Damit die Klassifizierungsaufgabe möglichst gut gelingt, muss das Netz zuerst mit einer ausreichenden Zahl von Trainingsdaten mit bekanntem Ergebnis trainiert werden. Im Verlauf dieses Trainings werden die Modellparameter mit einer geeigneten Lernregel so angepasst, dass die Vorhersagefehler möglichst klein werden.

3.2.1 Aufbau und Funktionsweise

Das hier untersuchte Beipielnetz hat zwei Eingabewerte x_1 und x_2, die jeweils zu zwei Neuronen in der verborgenen Schicht führen. Weiter sind die beiden Neuronen der verborgenen Schicht noch mit einem Eingang mit festem Eingabewert 1 verbunden. Die beiden Neuronen der verborgenen Schicht leiten ihre Ausgangssignale an das Ausgabeneuron in der Ausgabeschicht weiter, das ebenfalls noch einen zusätzlichen Eingang mit Wert 1 besitzt. Das Ausgangneuron liefert das Ausgangssignal y des Netzes. Abbildung 5 zeigt die im Folgenden untersuchte Netzstruktur. Für jedes Neuron werden die Eingangssignale zuächst aufsummiert. Allerdings geschieht das nicht in der Art der üblichen Addition, sondern die Signale werden noch mit den Gewichtsfaktoren w_{ji} multipliziert. Das ist an einem Beispiel im Bild 6 veranschaulicht. Dabei bezeichnet j die Nummer

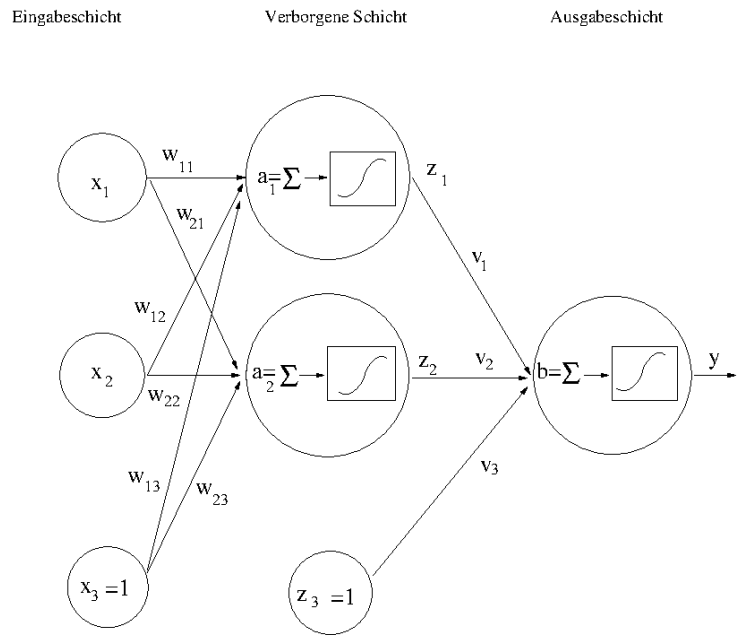

Abb. 5: Struktur des als Beispiel untersuchten kleinen künstlichen neuronalen Netzes mit drei Neuronen und zwei Schichten

des Neurons in der verborgenen Schicht und i den Index des Eingabesignals. Der fix mit 1 belegte Eingang wird dabei als drittes Eingangssignal gezählt. Die mit den Zahlen w_{ji} gewichteten Eingangssignale, werden dann in eine sogenannte Aktivierungsfunktion hineingegeben, die das weiterzuleitende Signal auf das Intervall [0,1] abbildet und über die Stärke der Weiterleitung entscheidet. Bei biologischen Neuronen erfolgt die Weiterleitung nur dann, wenn ein bestimmter Schwellwert überschritten ist. Das kann bei künstlichen neuronalen Netzen näherungsweise z.B. durch die Funktion

$$\sigma(x) = \frac{1}{1 + exp(-x)} \quad , \tag{2}$$

die in Abb. 7 dargestellt ist, erfolgen. Für große negative Eingabewerte ist sie ungefähr gleich 0, für große positive Eingabewerte ist sie ungefähr gleich 1, dazwischen liegt eine Übergangszone um den Punkt (0|0.5) herum. Beim Eingabewert 0 hat die Funktion einen Wendepunkt mit y-Wert 0.5. Wenn man alle Ausgaben $y > 0.5$ auf 1 aufrundet und alle Ausgabewerte $y < 0.5$ auf 0 abrundet, kann man damit auch eine scharfe Umschaltung zwischen den Ausgabewerten 0 und 1 abbilden. Wegen des s-förmigen Verlaufs heißt diese Funktion auch sigmoide Aktivierungsfunktion (σ : griechischer Buchstabe s).

Die Ausgangswerte der Neuronen der verborgenen Schicht werden schließlich ebenfalls mit Gewichten multipliziert und als Eingangswert für das Ausgangsneuron aufaddiert. Das Ausgangssignal y entsteht schlussendlich, indem man nochmals die Aktivierungsfunktion auf die gewichtete Summe der Eingänge des Ausgabeneurons anwendet.

3.2.2 Signalverarbeitung in Vorwärtsrichtung

Die Signalweiterleitung zwischen Eingang und verborgener Schicht beziehungsweise zwischen verborgener Schicht und Ausgangsschicht wird durch lineare Funktionen modelliert. Wenn man die Eingangssummen der beiden Neuronen in der verborgenen Schicht mit a_1 und a_2 bezeichnet, erhält man dafür die in Tabelle 1 zusammengestellten Berechnungsformeln. Die Gewichte w_{13} und w_{23} dienen dazu, den von x_1 und x_2 abhängigen Summenwert noch um den

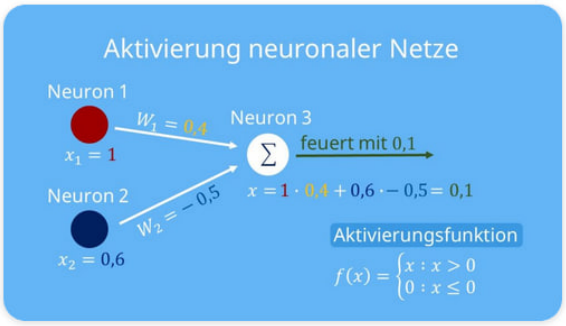

Abb. 6: Beispiel für gewichtete Signalweiterleitung nach [12]

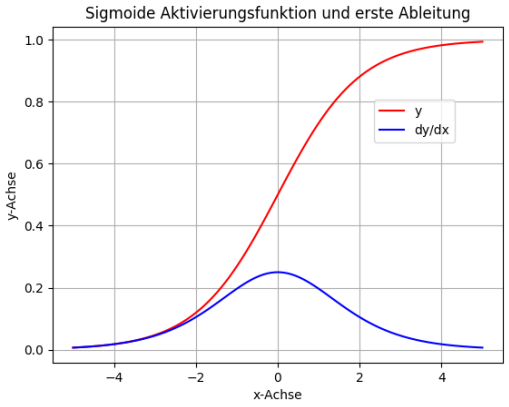

Abb. 7: Sigmoide Aktivierungsfunktion $y = 1./(1. + exp(-x))$ und erste Ableitung

Tabelle 1: Vorwärtspropagation der Eingabedaten durch das neuronales Netz

	Summierung	Aktivierung
Verborgene Schicht	$a_1 = w_{12}x_1 + w_{12}x_2 + w_{13}$ $a_2 = w_{22}x_1 + w_{22}x_2 + w_{23}$	$z_1 = \sigma(a_1) = \frac{1}{1+exp(-a_1)}$ $z_2 = \sigma(a_2) = \frac{1}{1+exp(-a_2)}$
Ausgabeschicht	$b = v_1z_1 + v_2z_2 + v_3$	$y = \sigma(b) = \frac{1}{1+exp(-b)}$
Fehler $\delta = y - t$ Kostenfunktion $E = \frac{1}{2}(y-t)^2$		

Wert von w_{13} bzw. w_{23} zu verschieben, was für die Aktivierung des Ausgangs des Neurons von Bedeutung ist. Diese Werte werden in der Literatur auch als Bias-Werte bezeichnet (von engl. bias = Verschiebung) [15]. Die Summenwerte a_1 und a_2 ergeben dann nach Durchgang durch die sigmoide Aktivierungsfunktion die Ausgangswerte z_1 und z_2 der Neuronen der verdeckten Schicht.

Diese Ausgangswerte bilden zusammen mit dem ebenfalls wie x_3 mit 1 belegten Bias-Eingang z_3 die Inputsignale für das Ausgangsneuron. Auch sie werden im Ausgangsneuron wieder mit Gewichten multipliziert und zum Wert b aufsummiert. Da es im Anwendungsbeispiel nur ein Ausgangsneuron gibt, kann der zweite Index bei den Gewichten entfallen. Sie werden hier mit v_1, v_2 und v_3 bezeichnet. Damit erhält man auch für den Summenwert b des Ausgangsneurons eine lineare Berechnungsformel, die ebenfalls in Tabelle 1 angegeben ist. Der Ausgangswert y des Netzes ergibt sich schließlich, nachdem der Summenwert b wiederum durch die Aktivierungsfunktion hindurchgeschickt wurde.

3.2.3 Fehlerkorrektur

Da die korrekten Gewichte zunächst unbekannt sind, werden sie zu Beginn des Trainings mit kleinen zufälligen Startwerten initialisiert. Damit wird man beim späteren Training des Netzes zuerst noch keine richtigen Vorhersagen erhalten, sondern es wird sich ein recht großer Fehler δ zwischen den Vorhersagewerten y und den korrekten Ergebniswerten $t = 1$ bzw. $t = 0$ für die jeweiligen Eingabewerte (x_1, x_2) der Trainingsdaten ergeben.

$$\delta = y - t \tag{3}$$

Die Aufgabe besteht nun darin, im Verlauf des Trainings mit den vorgegebenen Daten, deren Klassifizierung bekannt ist, die Gewichte so zu modifizieren, dass der Fehler möglichst klein wird. Da Abweichungen vom Sollwert t nach oben und unten möglich sind, wird dafür der quadratische Fehler, der immer positiv ist, in der Form

$$E = \frac{1}{2}(y - t)^2 \tag{4}$$

verwendet, um Fallunterscheidungen in der Fehlerkorrektur zu vermeiden. Die Größe E wird auch als Kostenfunktion bezeichnet [16], weil sie möglichst klein werden soll. Zur Fehlerkorrektur nutzt man die Idee, dass die Gewichte proportional zur negativen Änderungsrate des Fehlers nach den jeweiligen Gewichten korrigiert werden. Dieses Verfahren wird als Gradientenabstieg bezeichnet [17]. Wenn die erste Ableitung des Fehlers nach einem bestimmten Gewicht positiv ist, so nimmt der Fehler bei positiver Veränderung dieses Gewichts zu. Man muss deshalb in die Richtung des negativen Gradienten gehen. Ist die Ableitung nach diesem Gewicht jedoch negativ, so wird der Fehler, wenn man das betreffende Gewicht weiter in die negative Richtung verschiebt, größer und man muss auch in diesem Fall das Vorzeichen umdrehen. Da bei sehr großen Gradienten die Gefahr besteht, über das angestrebte Minimum hinauszuschießen, wird der negative Gradient noch mit einem zwischen 0 und 1 liegenden Faktor η multipliziert. Dieser Faktor η wird als Lernrate bezeichnet, weil er die Geschwindigkeit der Gewichtsänderung bestimmt. Die Lernrate ist ein weiterer Modellparameter, der für das Training passend ausgewählt werden muss.

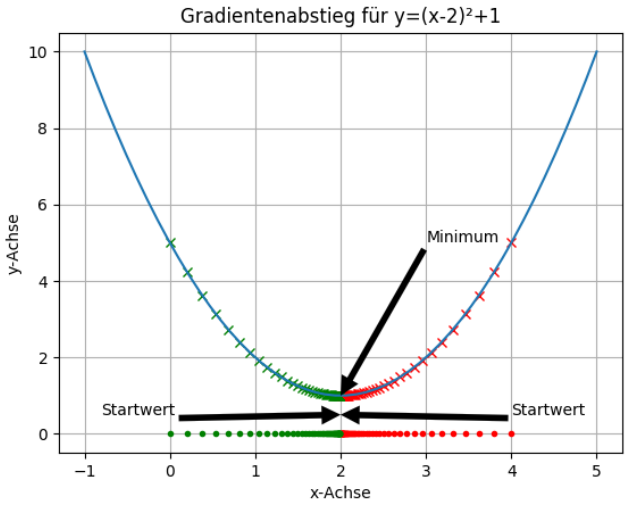

Abb. 8: Gradientenabstiegsverfahren am Beispiel der Minimumsuche für eine Parabel

In Abbildung 8 wird das Verfahren des Gradientenabstiegs am Beispiel der Minimumsuche einer parabolischen Funktion mit den Startwerte x=0 und x=4 veranschaulicht. Links vom Minimum werden die verbesserten Werte wegen der negativen Steigung in positiver x-Richtung korrigiert,

Tabelle 2: Fehler-Rückwärtspropagation für die Gewichtskorrekturen in der der Ausgabeschicht und der verborgenen Schicht

Schicht	Kettenregel	Teilableitungen	Gesamtableitung
Ausgabe-schicht	$\frac{dE}{dv_j} = \frac{dE}{dy}\frac{dy}{db}\frac{db}{dv_j}$	$\frac{dE}{dy} = y - t = \delta$ $\frac{dy}{db} = \sigma'(b) = \sigma(b)(1-\sigma(b))$ $\frac{db}{dv_j} = z_j$	$\frac{dE}{dv_j} = \delta\sigma'(b)z_j$ mit j=1,2,3 und $z_3 = 1$
Gewichtskorrektur nach Gl. 5: $v_{j,neu} = v_{j,alt} - \eta\delta\sigma'(b)z_j$			
Ver-borgene Schicht	$\frac{dE}{dv_j} = \frac{dE}{dy}\frac{dy}{db}\frac{db}{dz_j}\frac{dz_j}{da_j}\frac{da_j}{dw_{ji}}$	$\frac{db}{dz_j} = v_j$ $\frac{dz_j}{da_j} = \sigma'(a_j) = \sigma(a_j)(1-\sigma(a_j))$ $\frac{da_j}{dw_{ji}} = x_i$	$\frac{dE}{dw_{ji}} = \delta\sigma'(b)v_j\sigma'(a_j)x_i$ mit j=1,2 und i=1,2,3 und $x_3 = 1$
Gewichtskorrektur nach Gl. 6: $w_{ji,neu} = w_{ji,alt} - \eta\delta\sigma'(b)v_j\sigma'(a_j)x_i$			

rechts vom Minimum wegen der positiven Steigung in negativer x-Richtung, was jeweils zum Minimum führt.

Damit ergeben sich die folgenden Korrekturformeln für die Gewichte v_j zwischen verborgener Schicht und Ausgabeschicht

$$v_{j,neu} = v_{j,alt} - \eta\frac{dE}{dv_j} \qquad \text{mit} \qquad j = 1,2,3 \tag{5}$$

und

$$w_{ji,neu} = w_{ji,alt} - \eta\frac{dE}{dw_{ji}} \qquad \text{mit} \qquad j = 1,2,3 \qquad \text{und} \qquad i = 1,2 \tag{6}$$

für die Gewichte w_{ji} zwischen der Eingabeschicht und der verborgenen Schicht.

Nun müssen als nächster Schritt die Ableitungen der Fehlerfunktion E nach den Gewichten v_j der verborgenen Schicht bestimmt werden. Weil der Fehler E(y)=E(y(b))=E(y(b(v_1, v_2, v_3))) eine Funktion von mehreren anderen Funktionen ist, die ineinander verschachtelt sind, bekommt man diese Ableitungen mit der Kettenregel. Dadurch wird die gesamte Ableitung, wie in Tabelle 2 angegeben ist, in Faktoren aufgespalten.

Die einzelnen Faktoren in der Kettenregel kann man aus den bisher gefundenen Gleichungen für E(y), y(b) und b(v_1, v_2, v_3) ausrechnen. Die dabei auch gebrauchte Ableitung der Aktivierungsfunktion $\frac{dy}{db} = \sigma'(b)$ muss nicht mit Hilfe von Ableitungsregeln berechnet werden, weil man sie nach [17] auch direkt durch die Aktivierungsfunktion als $\sigma'(b) = \sigma(b)(1-\sigma(b))$ ausdrücken kann. Die Kettenregel für die Ableitung der Kostenfunktion E nach den Gewichten v_j und die Formeln für die einzelnen Faktoren sind ebenfalls in der Tabelle 2 zusammengestellt.

Tabelle 3: Formeln für die Gewichtskorrekturen bei zwei Neuronen in der verborgenen Schicht und einem Ausgabeneuron

Ausgabeschicht	verborgene Schicht
$v_{1,neu} = v_{1,alt} - \eta\delta\sigma'(b)z_1$ $v_{2,neu} = v_{2,alt} - \eta\delta\sigma'(b)z_2$ $v_{3,neu} = v_{3,alt} - \eta\delta\sigma'(b)$	$w_{11,neu} = w_{11,alt} - \eta\delta\sigma'(b)v_1\sigma'(a_1)x_1$ $w_{12,neu} = w_{12,alt} - \eta\delta\sigma'(b)v_1\sigma'(a_1)x_2$ $w_{13,neu} = w_{13,alt} - \eta\delta\sigma'(b)v_1\sigma'(a_1)$ $w_{21,neu} = w_{21,alt} - \eta\delta\sigma'(b)v_2\sigma'(a_2)x_1$ $w_{22,neu} = w_{22,alt} - \eta\delta\sigma'(b)v_2\sigma'(a_2)x_2$ $w_{23,neu} = w_{23,alt} - \eta\delta\sigma'(b)v_2\sigma'(a_2)$

Damit bekommt man schließlich eine recht einfache Formel für die Änderungsrate der Fehlerfunktion E mit den Gewichten v_j, die oben in der rechten Spalte von Tabelle 2 eingetragen ist. Damit kann die Korrektur der Gewichte der Ausgangschicht, wie in Tabelle 2 in der dritten Tabellenzeile angegeben ist, berechnet werden.

Für die Korrektur der Gewichte w_{ji}, die in die Eingabe der Neuronen der verdeckten Schicht eingehen, muss die Fehlerfunktion zur Anwendung des Gradientenabstiegs auch noch nach diesen Gewichten abgeleitet werden. Das geht ebenfalls mit der Kettenregel, wobei b nun als Funktion von z_1 und z_2 zu verstehen ist. Die Werte z_1 und z_2 sind wiederum Funktionen von a_1 und a_2. Und die Werte von a_1 und a_2 sind schließlich Funktionen der Gewichte w_{ji} mit j=1, 2, 3 sowie i=1, 2. Demnach berechnet sich die Ableitung der Kostenfunktion E nun mit Hilfe der Kettenregel aus 5 Faktoren. Dabei sind aber die ersten beiden Terme $\frac{dE}{dy}$ und $\frac{dy}{db}$ schon aus der Korrektur der Gewichte der Ausgabeschicht bekannt. Die Formeln für Kettenregel und ihre Faktoren sind ebenfalls in Tabelle 2 zusammengestellt und können wieder aus den Gleichungen der Vorwärtspropagation nach Tabelle 1 leicht ausgerechnet werden.

Wenn man alle benötigten Teilableitungen miteinander multipliziert, erhält man die Gleichung für die Änderungsrate der Fehlerfunktion E nach w_{ji}, die ebenfalls in der rechten Spalte von Tabelle 2 eingetragen ist. Damit bekommt man ebenfalls eine Berechnungsformel für die Gewichtskorrekturen der Neuronen der verdeckten Schicht, die in der letzten Zeile von Tabelle 2 angegeben ist. Im vorliegenden konkreten Anwendungsbeispiel mit zwei Neuronen in der verborgenen Schicht und einem Ausgabeneuron ergeben sich daraus insgesamt die in Tabelle 3 zusammengestellten Gewichtskorrekturen.

Mit diesen Gleichungen können die Gewichte nach Vorwärtspropagation der Eingabewerte eines Trainingsbeispiels durch das Netz korrigiert werden, so dass der Fehler vermindert wird. Danach wird das Netz mit dem nächsten Trainingsdatensatz (x_1, x_2, t) gefüttert und man kann mit dem Ergebnis die nächste Fehlerkorrektur durchführen. Das wird solange wiederholt, bis der Fehler ausreichend klein geworden ist bzw. bis Tests mit weiteren Datensätzen eine ausreichende Klassifikationsleistung des Netzes zeigen.

Zum Training und Testen des Netzes wird das folgende Berechnungsschema verwendet und in einem Python-Programm umgesetzt:

1. Trainingsbeispiel (x_1, x_2, t) als Eingabe ins Netz hineinschicken
2. Ausgabewert y mit den Gleichungen aus Tabelle 1 berechnen
3. Fehler $\delta = y - t$ und quadratische Kostenfunktion E ausrechnen
4. Ableitungen von E nach den Gewichten ausrechnen

5. Alle Gewichte mit den Gleichungen nach Tabelle 3 korrigieren
6. Nächstes Trainingsbeispiel ins Netz hineingeben, solange bis alle Trainingsbeispiele verbraucht sind
7. Testen des Netzes mit den korrigierten Gewichten mit Testbeispielen x_1, x_2 mit unbekannten Ergebniswerten t
8. Graphische Darstellung der vom trainierten Netz ermittelten Klassifizierungsergebnisse und der Entwicklung von mittlerem Fehler und Gewichten im Verlauf des Trainings

Für die Fehlerkorrektur gibt es nach [18] die Möglichkeit, zunächst alle Daten zu verarbeiten und die Korrektur erst danach auf die Summe aller Fehler anzuwenden. Anschließend werden wieder alle Beispiele mit den neuen Gewichten durchgerechnet und erst dann werden die Gewichte erneut aktualisiert. Dieses Verfahren wird als Batch-Learning (von engl. batch=Stapel) bezeichnet. Alternativ dazu können auch kleinere Datenstapel aus Teilmengen der Trainingsdaten gewählt werden, was als Mini-Batch-Learning bezeichnet wird. Man spricht dann auch von der Methode des stochastischen Gradientenabstiegs. Aus Gründen der Einfachheit werden im vorliegenden Beispielfall die Gewichte bereits nach jedem Trainigsbeispiel aktualisiert, was einem Mini-Batch-Learning mit Stapelgröße 1 entspricht.

3.2.4 Ergebnisse

Das mit den obigen Gleichungen optimierte Netz wurde für zwei Klassifizierungsaufgaben trainiert und getestet. Die gestellten Aufgaben waren die Einteilung der $x_1 - x_2$-Ebene im Intervall [-1,1] x [-1,1] in Punkte, die oberhalb oder unterhalb einer Grenzkurve liegen. Die Grenzkurven haben die Form

$$x_2 = kx_1 + d \qquad \text{oder} \qquad x_2 = kx_1^2 + d \qquad . \tag{7}$$

Das sind Geraden mit Steigung k und Achsenabschnitt d oder Parabeln, die um den Wert d entlang der x_2-Achse verschoben sind und mit dem Faktor k gestreckt oder gestaucht sind.

Für die Testbeispiele wurde jeweils d=-0.7 und k=2 gewählt. Die Anzahl der Trainingsdatensätze betrug 100000, die Anzahl der Testdatensätze 500 und für die Lernrate η wurde ein Wert von 0.1 festgelegt.

Die Abbildung 9 zeigt die Klassifizierungsergebnisse für den Fall einer Geraden als Grenzkurve. Im Teilbild links oben sieht man, dass die Testpunkte alle richtig eingeteilt werden. Das danebenstehende Bild zeigt aber, dass nicht für alle Testpunkte Werte sehr nahe bei 1 für oberhalb und Werte sehr nahe bei 0 für unterhalb der Grenzkurve gelegene Datenpunkte erreicht werden. Da als Entscheidungskriterium im Programm festgelegt wurde, dass Ergebniswerten ab $y > 0.5$, die somit näher bei 1 als bei 0 liegen, auch zur Zuordnung oberhalb führen, werden auch diese nicht so genau erkannten Fälle ebenfalls richtig zugeordnet. Analog wird es dann für Fälle mit $y < 0.5$ gemacht, die alle den Punkten unterhalb der Grenzkurve zugerechnet werden.

Das untere linke Teilbild zeigt, dass der mittlere Fehler der Trainingsdaten langsam abnimmt. Für den mittleren Fehler wurden immer 100 Werte gemittelt, da sonst die selten doch noch auftretenden Einzelfälle mit hohem Fehler den Eindruck erwecken, dass die Fehlergröße nicht abnimmt. Die Mittelung mittelt diese Ausnahmen dann heraus.

Das rechte untere Bild zeigt schließlich, wie sich die Gewichte im Verlauf des Trainings entwickeln. Es sieht so aus, als ob sie sich einem Grenzwert annäherten, doch müssten, um das abzusichern, noch mehr Trainingsdaten verwendet werden.

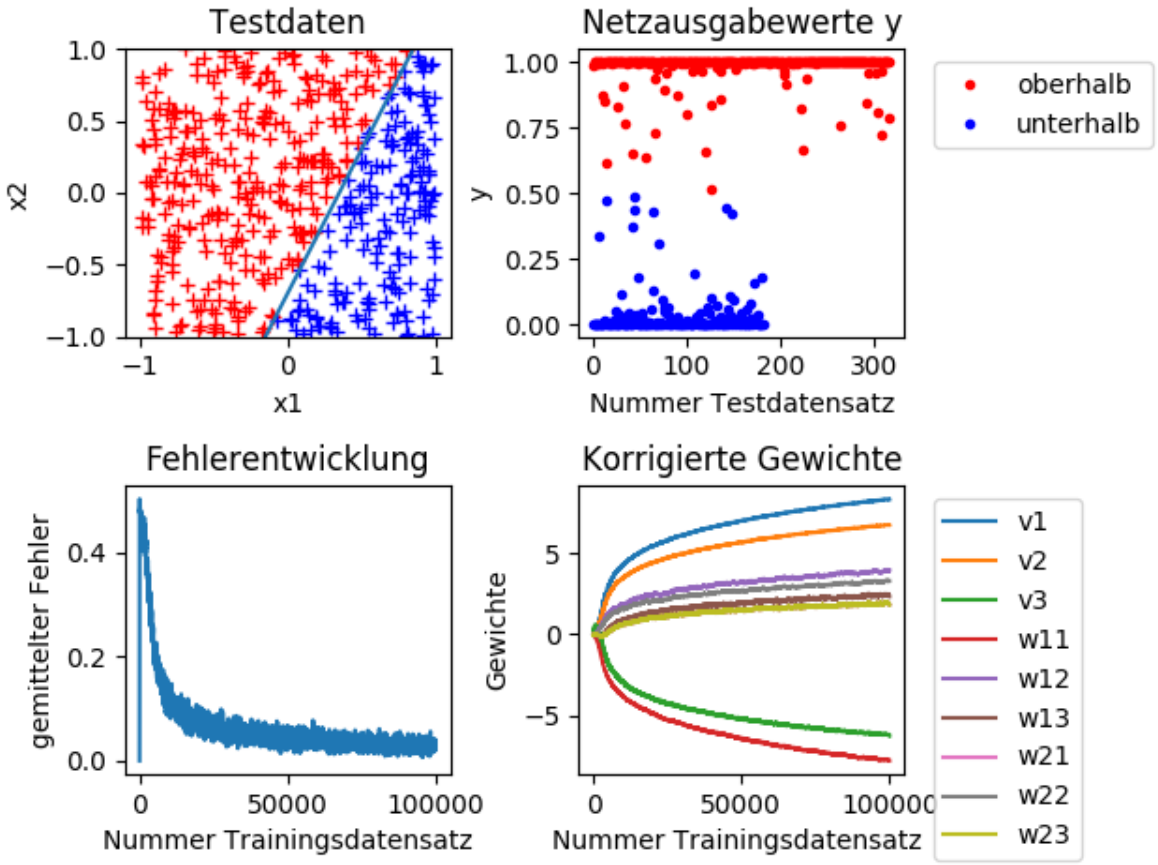

Abb. 9: Anwendungsbeispiel für Klassifizierung von Punkten oberhalb und unterhalb einer Geraden

Die Abbildung 10 zeigt die Ergebnisse für den Fall einer Parabel als Grenzkurve. Auch hier gelingt die Einteilung der Testpunkte ziemlich gut. Es gibt aber einige wenige Ausnahmen, wo eine leichte Fehlzuordnung auftritt. Das zeigt an, dass die Unterteilung durch eine Parabel schwieriger zu lernen ist, was man auch im rechten oberen Teilbild sieht, wo mehr Punkte in größerer Entfernung von 1 oder 0 liegen als im Fall der Geraden. Dementsprechend sinkt hier auch der Fehler nicht so stark ab. Durch längeres Training oder höhere Lernraten können die Ergebnisse aber weiter verbessert werden.

3.2.5 Vor- und Nachteile neuronaler Netze

Dass neuronale Netze wegen ihrer großen Flexibilität im Aufbau für sehr komplexe Aufgaben geeignet sind und z.B. die Grundlage für die sehr leistungsfähigen Sprachmodelle wie ChatGPT bilden, kann als genereller Vorteil angesehen werden. Weiter können neuronale Netze rasch durch weiteres Training verbessert werden und haben sehr kurze Antwortzeiten. Als Nachteile kann ihre schwer nachvollziehbare Entscheidungsfindung, die nicht auf einfache Weise auf der Ebene von Teilschritten überprüft werden kann, angesehen werden, was als Blackbox-Problem bezeichnet wird. Als weitere Nachteile werden der Wahrscheinlichkeitscharakter der Antworten, der immer eine gewisse Unsicherheit enthält, und die für komplexe Aufgaben oft sehr großen für das Training benötigten Datenmengen genannt (s. [19]).

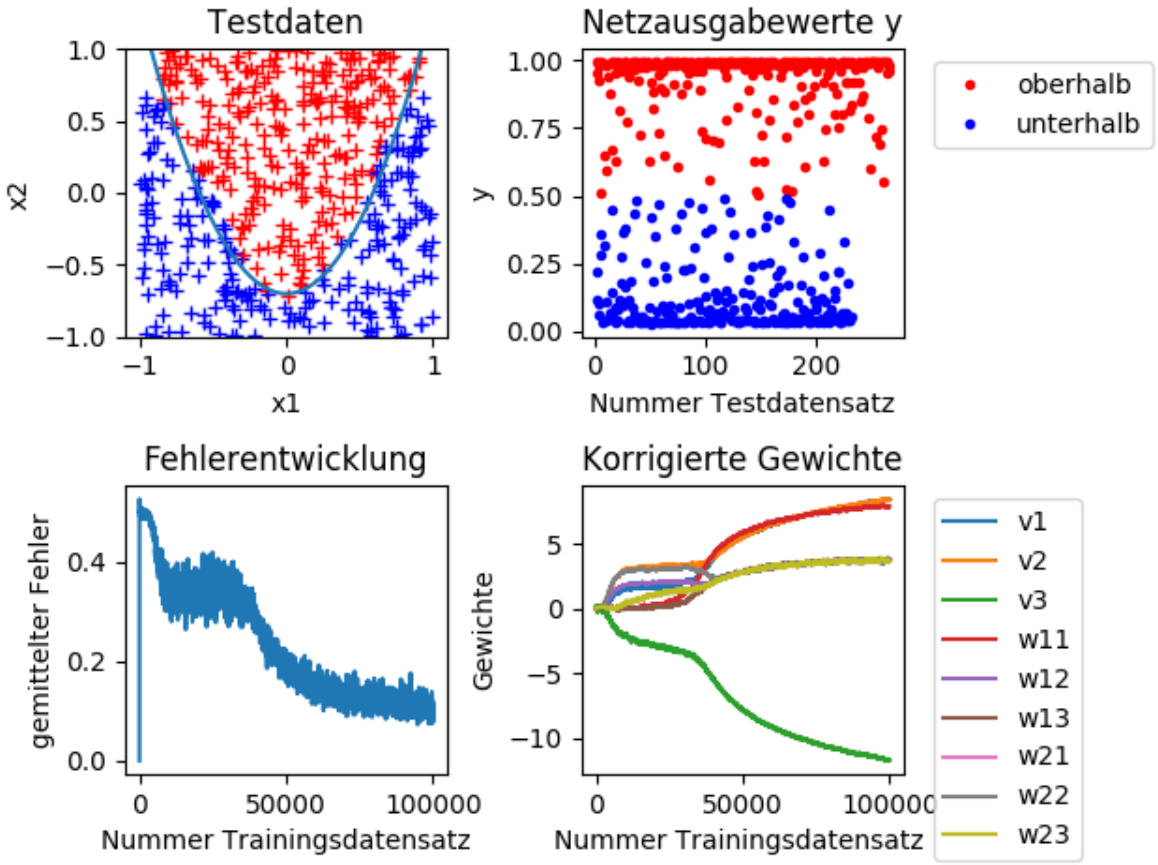

Abb. 10: Anwendungsbeispiel für Klassifizierung von Punkten oberhalb und unterhalb einer Parabel

3.3 Logistische Regression

Die Methode der logistischen Regression ist eine weitere Klassifikationsmethode zur Einteilung von Daten entsprechend einer binären 0/1-Kennzeichnung. Die logistische Regression wird hier wieder am Beispiel der Einteilung der $x_1 - x_2$-Ebene in Punkte, die über- oder unterhalb einer Geraden liegen, in Anlehnung an die Darstellung in [20] erklärt.

Dieses Einteilungsproblem der Ebene in zwei Punkteklassen kann auch dadurch gelöst werden, dass man eine Funktion findet, die gegebene Punktkoordinaten direkt in den Wert 1 für oberhalb und in den Wert 0 für unterhalb der Geraden gelegene Punkte abbildet. Weil die möglichen Koordinaten im Prinzip Werte zwischen $-\infty$ und $+\infty$ annehmen können, sind Funktionen, die zwischen zwei Schranken mit einem steilen Übergang dazwischen liegen, dafür gut geeignet. Eine solche Funktion ist z.B. die im vorigen Kapitel schon verwendete Sigmoidfunktion.

$$y = \sigma(\vec{x}) = \frac{1}{1 + exp(-(\vec{w} \cdot \vec{x} + b))} \qquad \text{mit} \quad \vec{x} = (x_1, x_2) \quad \text{und} \quad \vec{w} = (w_1, w_2) \quad . \tag{8}$$

Dabei kann das Argument der Exponentialfunktion bei normiertem Vektor $\vec{w}$ als Abstandsfunktion von Punkten der Ebene von einer durch $\vec{w} \cdot \vec{x} + b = 0$ definierten Geraden verstanden werden. Wenn man die Trenngerade des später durchgerechneten Beispiels $x_2 = -2 \cdot x_1 + 0.7$ in die Normalform bringt, so ergibt sich damit der in Abbildung 11 gezeigte Verlauf, an dem man das gewünschte Übergangsverhalten entlang einer Grenzlinie gut erkennen kann. Mit Hilfe dieser Funktion kann man die gesuchte Zuordung von Punkten oberhalb der ebenfalls eingezeichneten Trenngeraden zu 1 und unterhalb zu 0 schließlich korrekt durchführen,

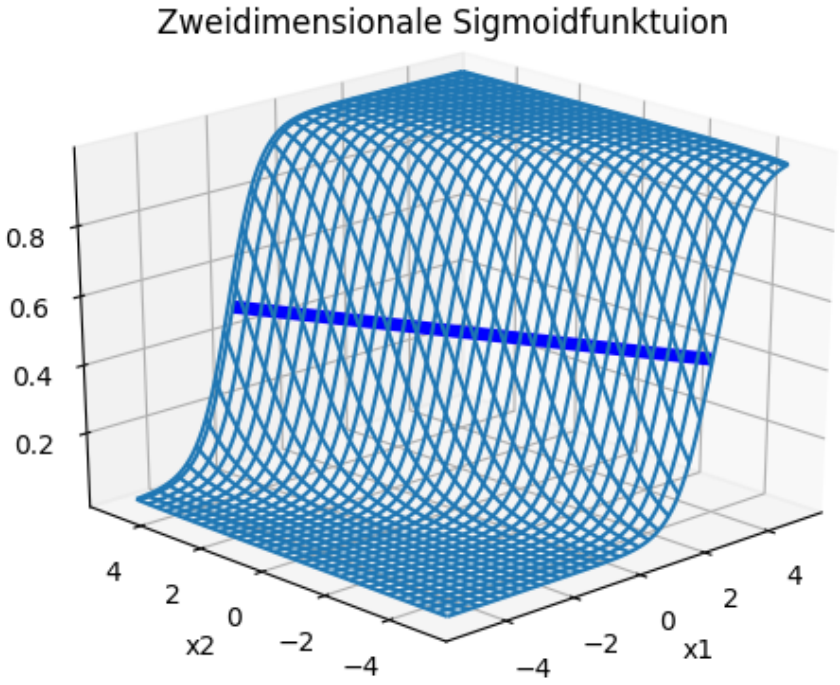

Abb. 11: Zweidimensionale Sigmoidfunktion $y = 1./(1. + exp(-(2 \cdot x_1 + x_2 - 0.7)))$ und Trennlinie zwischen den Klassifizierungsgebieten

wenn man alle Werte < 0.5 anschließend abrundet und alle Werte ≥ 0.5 aufrundet.

Allerdings sind die passenden Parameterwerte für w_1, w_2 und b zunächst nicht bekannt. Um sie zu bestimmen, deutet man nach [20] die Ergebniswerte der Sigmoidfunktion für die gegebenen Trainingsdaten als Wahrscheinlichkeiten p_i dafür, dass ein untersuchter Datenpunkt (x_{1i}, x_{2i}) oberhalb der Grenzkurve liegt. Deshalb soll er die Kennzeichnung $y_i = 1$ erhalten, wenn $\sigma(\vec{x_i}, \vec{w}, b) \geq 0.5$ gilt. Dementsprechend ist $1 - p_i$ die Wahrscheinlichkeit dafür, dass der Datenpunkt die Zuordnung $y_i = 0$ bekommt, dann wenn $\sigma(\vec{x_i}, \vec{w}, b) < 0.5$ ist. Die bedingten Wahrscheinlichkeiten $p_i = P(y_i = 1|\sigma(\vec{x_i}, \vec{w}, b)) \geq 0.5$ und $1 - p_i = P(y_i = 0|\sigma(\vec{x_i}, \vec{w}, b)) < 0.5)$, deren Werte auch noch von den Modellparametern w_1, w_2 und b abhängen, werden als Likelihood bezeichnet. Die Likelikood kann man als eine Wahrscheinlichkeit verstehen, die von zusätzlichen Parametern abhängt, die noch zu bestimmen sind. Wie bei den gewöhnlichen Wahrscheinlichkeiten ist auch die Likelihood von zusammengesetzten und voneinander unabhängigen Ereignissen gleich dem Produkt der Likelihoodwerte der Einzelereignisse. Demnach kann man die Gesamtlikelihood $L_{w,b}$, die dann ebenfalls von w_1, w_2 und b abhängt, nach [20] mit der Formel

$$L_{w,b} = \prod_{i=1}^{n} p_i^{y_i} \cdot (1 - p_i)^{(1-y_i)} \qquad \text{mit} \tag{9}$$

$$p_i = \sigma(\vec{x_i}, \vec{w}, b)) = 1./(1. + exp(-(w_1 x_{1i} + w_2 x_{2i} + b))) \tag{10}$$

ausrechnen. Die obige Schreibweise der Produktfaktoren sieht kompliziert aus, bewirkt aber nur, dass der Faktor unter dem Produktzeichen genau dann gleich p_i wird, wenn $y_i = 1$ ist, und

umgekehrt genau dann gleich $(1-p_i)$ wird, wenn $y_i = 0$ ist. Die Gesamtlikelihood wird demnach bei richtiger Klassifizierung aller Trainingsdaten maximal, weil dann die p_i und $(1-p_i)$ die größtmöglichen Werte annehmen. Deshalb kann man die optimalen $\vec{w}$- und b-Parameter finden, indem man das Maximum der Likelihoodfunktion bezüglich der unbekannten Variablen $\vec{w}$ und b bestimmt.

Dazu werden die Ableitungen der Likelihoodfunktion nach den Variablen w_1, w_2 und b benötigt. Da Summen von Funktionen einfacher abzuleiten sind als Produkte, logarithmiert man nun noch die Gleichung 9, was am Maximum nichts ändert, da der Logarithmus eine streng monotone Funktion ist. Man bekommt damit die sogenannte Log-Likelihood-Funktion

$$L^*_{w,b} = ln(L_{w,b}) = \sum_{i=1}^{n} y_i ln(p_i) + (1-y_i) ln(1-p_i) \qquad \text{mit} \qquad p_i = \sigma(w_1 x_{1i} + w_2 x_{2i} + b) \quad . \tag{11}$$

Das Maximum dieser Funktion lässt sich wieder mit dem numerischen Verfahren des Gradientenabstiegs finden, das auch zur Maximumsuche verwendet werden kann, wenn man bei der schrittweisen Lösung in Richtung des zunehmenden statt des abnehmenden Gradienten geht. Weil $L^*_{w,b} = L^*_{w,b}(p_i(w_1, w_2, b))$ und $p_i = \sigma(w_1, w_2, b)$ ist, erhält man die Ableitungen wiederum mit der Kettenregel. Für die Ableitung nach w_1 gilt, wenn a_i das Argument der σ-Funktion bezeichnet

$$\frac{dL^*_{w,b}}{dw_1} = \frac{dL^*_{w,b}}{d\sigma} \frac{d\sigma}{da_i} \frac{da_i}{dw_1} \quad . \tag{12}$$

Man muss deshalb als nächsten Schritt die einzelnen Faktoren dieses Produkts aus drei Ableitungen bestimmen. Für $\frac{dL^*_{w,b}}{d\sigma}$ bekommt man

$$\frac{dL^*_{w,b}}{d\sigma} = \sum_{i=1}^{n} \frac{y_i - \sigma}{\sigma(1-\sigma)} \quad . \tag{13}$$

Für $\frac{d\sigma}{da_i}$ erhält man

$$\frac{d\sigma}{da_i} = \sigma(a_i) \cdot (1 - \sigma(a_i)) \tag{14}$$

und für $\frac{da_i}{dw_1}$ ergibt sich

$$\frac{da_i}{dw_1} = x_{1i} \qquad . \tag{15}$$

Wenn man die letzten drei Gleichungen in die Ableitung der Log-Likelihood-Funktion nach w_1 in Gleichung 12 einsetzt, so hebt sich die Ableitung der Sigmoidfunktion $\sigma(1-\sigma)$ heraus und man bekommt dafür den recht einfachen Ausdruck

$$\frac{dL^*_{w,b}}{dw_1} = \sum_{i=1}^{n} (y_i - \sigma) x_{1i} \quad . \tag{16}$$

Auf die gleiche Weise erhält man die beiden anderen Ableitungen zu

$$\frac{dL^*_{w,b}}{dw_2} = \sum_{i=1}^{n} (y_i - \sigma) x_{2i} \quad . \tag{17}$$

und

$$\frac{dL^*_{w,b}}{db} = \sum_{i=1}^{n} (y_i - \sigma) \quad . \tag{18}$$

Damit kann man nun die optimalen Werte für die Modellparameter nach dem Gradientenverfahren ermitteln, wobei man wieder mit geschätzten Startwerten beginnt.

$$w_{1,neu} = w_{1,alt} + \eta \frac{dL^*_{w,b}}{dw_1} = w_{1,alt} + \eta \sum_{i=1}^{n} (y_i - \sigma) x_{1i} \tag{19}$$

$$w_{2,neu} = w_{2,alt} + \eta \frac{dL^*_{w,b}}{dw_2} = w_{2,alt} + \eta \sum_{i=1}^{n} (y_i - \sigma) x_{2i} \tag{20}$$

$$b_{neu} = b_{alt} + \eta \frac{dL^*_{w,b}}{db} = b_{alt} + \eta \sum_{i=1}^{n} (y_i - \sigma) \quad . \tag{21}$$

Das wird solange wiederholt bis sich die Modellparameter nicht mehr ändern.

Dieser Lernalgorithmus wurde ebenfalls in Python programmiert und auf das Klassifizierungsbeispiel der Zuordnung von Punkten der Ebene ober- und unterhalb einer Geraden angewendet. Abbildung 12 zeigt die Entwicklung der Werte für Steigung k und Achsenabschnitt d der Trenngeraden. Man erkennt, dass sich nach etwa 200 Iterationen weitgehend stabile Werte eingestellt haben. Abbildung 13 zeigt die Zuordnungsergebnisse von 50 Testdaten nach Ermittlung der optimalen Modellparameter, was fehlerfrei gelingt.

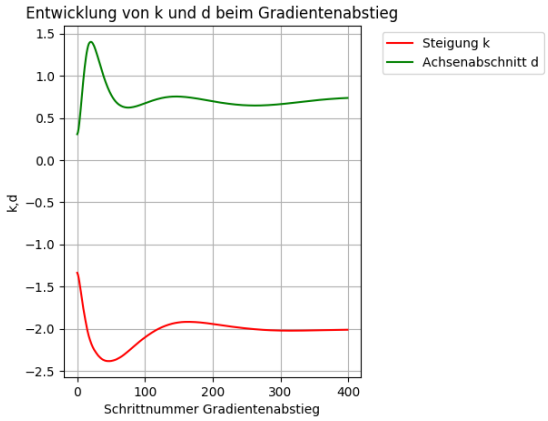

Abb. 12: Entwicklung von Steigung k und Achsenabschnitt d der Trenngeraden im Verlauf des Gradientenabstiegs.

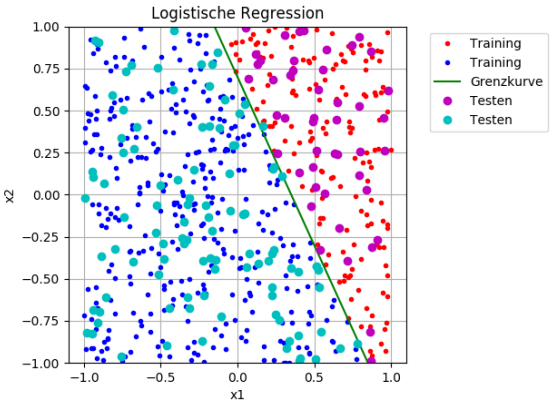

Abb. 13: Klassifizierungsergebnisse mit dem Verfahren der logistischen Regression für das Beispiel der Einteilung von Punkten der Ebene ober- und unterhalb einer Trenngeraden.

3.4 Support Vector Machine

Ein weiteres viel verwendetes und leistungsstarkes Verfahren des maschinellen Lernens ist der Support Vector Machine-Algorithmus [21], das zur Gruppe der überwachten Lernverfahren gehört und sowohl bei Klassifizierungs- als auch Regressionsaufgaben verwendet werden kann. Bei den Klassifizierungsaufgaben wird auch bei dieser Lernmethode zunächst eine Trennkurve zwischen den Trainingsdaten gesucht, die nun aber noch die Nebenbedingung erfüllen muss, dass sie einen möglichst großen Abstand zu den verschiedenen Datenklassen haben soll. Dadurch entsteht im zweidimensionalen Fall ein Band mit einer bestimmten Breite, in dem sich keine Trainingsdaten befinden. Die Trennkurve verläuft in der Mitte dieses Bandes. Dadurch wird die Wahrscheinlichkeit erhöht, dass die Testdaten richtig zugeordnet werden, weil die Trennkurve zu beiden Klassen hin einen möglichst großen Randabstand einhält, wie in Abbildung 14 veranschaulicht ist.

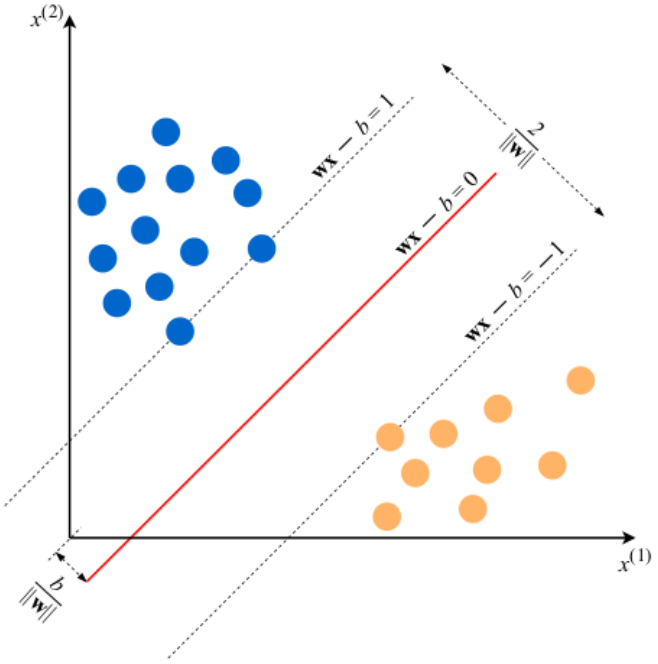

Abb. 14: Skizze zum SVM-Algorithmus in zwei Dimensionen nach [21]

Im zweidimensionalen Fall lässt sich demnach die Trennkurve durch eine Gerade in der Form

$$\vec{w} \cdot \vec{x} - b = 0 \tag{22}$$

darstellen, wobei $\vec{w} = (w_1, w_2)$ ein Normalenvektor und b eine reelle Zahl ist. Gelingt es, die für den jeweiligen Trainingsdatensatz passenden $\vec{w}$- und b-Werte zu finden, so kann man mit

$$f(x) = sign(\vec{w} \cdot \vec{x} - b) \tag{23}$$

die Lage neuer ungekennzeichneter Testpunkte vorhersagen. Die Signum-Funktion gibt dabei das Vorzeichen ihres Arguments $\vec{w} \cdot \vec{x} - b$ an. Punkte, die auf der Seite liegen, zu der der Normalenvektor zeigt, erhalten damit die Kennzeichnung 1, Punkte, die auf der anderen Seite liegen, erhalten die Kennzeichnung -1. Das ergibt sich daraus, dass Gleichung 22 nach Division durch $|\vec{w}|$ die normierte Normalvektordarstellung der Geraden ergibt, aus der man bekanntlich durch Einsetzen eines Punktes den Abstand dieses Punktes zur Geraden erhält.

Die Datenpunkte, die der Trenngeraden am nächsten liegen, bestimmen die Breite des Randes zwischen den beiden Datenklassen, in dem keine Trainingsdaten liegen. Diese Punkte heißen Support-Vektoren. Durch sie verlaufen zwei zur Trennkurve parallele Geraden, die ebenfalls in Abbildung 14 eingezeichnet sind und die die Breite des Randes definieren. Wenn $\pm d$ den Abstand von der Trenngeraden zu den Stützvektoren bezeichnet, so ergeben sich die Gleichungen der Randgeraden durch Addition oder Subtraktion von $d\frac{\vec{w}}{|\vec{w}|}$ zu $\vec{x}$ als

$$\vec{w} \cdot \vec{x} - b = \pm d|\vec{w}| \quad . \tag{24}$$

Wenn man diese Gleichung durch $d|\vec{w}|$ dividiert, d.h. den Normalenvektor $\vec{w}$ und die Konstante b damit skaliert, vereinfacht sich sich Gleichung 24 zu

$$\vec{w'} \cdot \vec{x} - b' = \pm 1 \quad . \tag{25}$$

Das entspricht, wenn man die Apostrophzeichen wegläßt, den in Abbildung 14 angegebenen Gleichungen. In dieser Darstellung beträgt der Abstand von der Trenngeraden zu den Punkten

der Randgeraden $1/|\vec{w}|$. Dieser Abstand wird maximal, wenn $|\vec{w}|$ möglichst klein wird. Man erhält die optimale Trenngerade also, wenn man $|\vec{w}|$ mit den Nebenbedingungen

$$\begin{aligned} \vec{w} \cdot \vec{x_i} - b \geq +1 \qquad & \text{falls} \qquad y_i = +1 \\ \vec{w} \cdot \vec{x_i} - b \geq -1 \qquad & \text{falls} \qquad y_i = -1 \end{aligned} \tag{26}$$

für alle Trainingswerte $\vec{x_i}$ minimiert. Die Lösung dieser Optimierungsaufgabe kann mit numerischen Methoden aus dem Fachgebiet der konvexen quadratischen Optimierung gelöst werden, was aber über die Schulmathematik hinausführt.

Der Support Vector-Machine-Algorithmus hat nach [22] die Vorteile, dass er für hochdimensionale Daten, d.h. Datensätze mit sehr vielen Merkmalen sehr effektiv ist und wenig Speicherplatz erfordert, weil er nur die Support-Vector-Datenpunkte für die Klassifizierung benötigt. Nachteilig ist, dass er nur konkrete Klassenzuordnungen und keine Wahrscheinlichkeiten wie z.B. die logistische Regression liefert.

In der Programmbibliothek Scikit-learn von Python ist der Support-Vector-Machine-Algorithmus mit der Bestimmung optimaler Trennkurven als Klassifizierungswerkzeug eingebaut. Dort können auch nicht linear trennbare Datensätze, die durch kompliziertere Grenzkurven getrennt sind, mittels sogenannter Kernel-Methoden (siehe z.B. [23],[24]) klassifiziert werden. Dazu wird die Grenzkurve durch eine geeignete Abbildung in ein anderes Koordinatensystem transformiert, so dass dort eine lineare Trennung möglich ist. Ein Beispiel dafür ist die Abbildung von Daten, die innerhalb und ausserhalb eines Kreises mit Mittelpunkt (0|0) und Radius R in der Ebene angeordnet sind, in das Polarkoordinatensystem. Dadurch lassen sich die Datenpunkte allein über den Abstand zum Koordinatenursprung, der größer oder kleiner als R ist, bereits leicht linear trennen.

Die SVM-Implementierung in Python wurde ebenfalls auf das bisher verwendete Beispiel der Klassifizierung von Punkten der Ebene in über oder unterhalb einer Geraden liegende Punkte angewendet. Für das Training und für das Testen wurden jeweils 50 Datensätze benutzt. Die Ergebnisse in Abbildung 15 zeigen, dass die Klassifizierung für alle Testdatenpunkte korrekt durchgeführt wird. In die Abbildung sind auch die Grenzkurve für die Datengenerierung, die vom Algorithmus gefundene Trenngerade und die Randgeraden mit den Support-Vektoren eingezeichnet. Man erkennt auch, dass der SVM-Algorithmus eine etwas andere Trennlinie als die für die Datengenerierung verwendete Trennkurve findet, die anscheinend die Trainingsdaten besser separiert.

3.5 Entscheidungsbaum-Lernen

Das Lernen mit Hilfe von Entscheidungsbäumen ist ein weiterer wichtiger Algorithmus des maschinellen Lernens. Entscheidungsbäume gehören ebenfalls zum Typ der überwachten Verfahren. Nach [26] bestehen sie aus einer Abfolge von hierarchisch, d.h. von oben nach unten durchlaufenen Entscheidungsregeln. Ihre Struktur besteht aus Elementen, die Knoten, Äste und Blätter genannt werden. In den Knoten befinden sich Entscheidungsfragen zu den verschiedenen Merkmalen der Datensätze. In den Blättern befinden sich die getroffenen Entscheidungen (z.B. Zuordnung zu einer Klasse) und die Äste verbinden die Elemente miteinander. Der Baum startet mit dem sogenannten Wurzelknoten und verzweigt sich nach unten immer weiter, bis alle Datensätze einem Endblattknoten zugeordnet sind.

Für die Erstellung der Bäume zu einem bestimmten Datensatz gibt es viele Möglichkeiten und sie können in einfachen und übersichtlichen Fällen auch von Menschen erzeugt werden. Bei der maschinellen Erstellung ist das Ziel, dass die Bäume möglichtst schnell und effizient

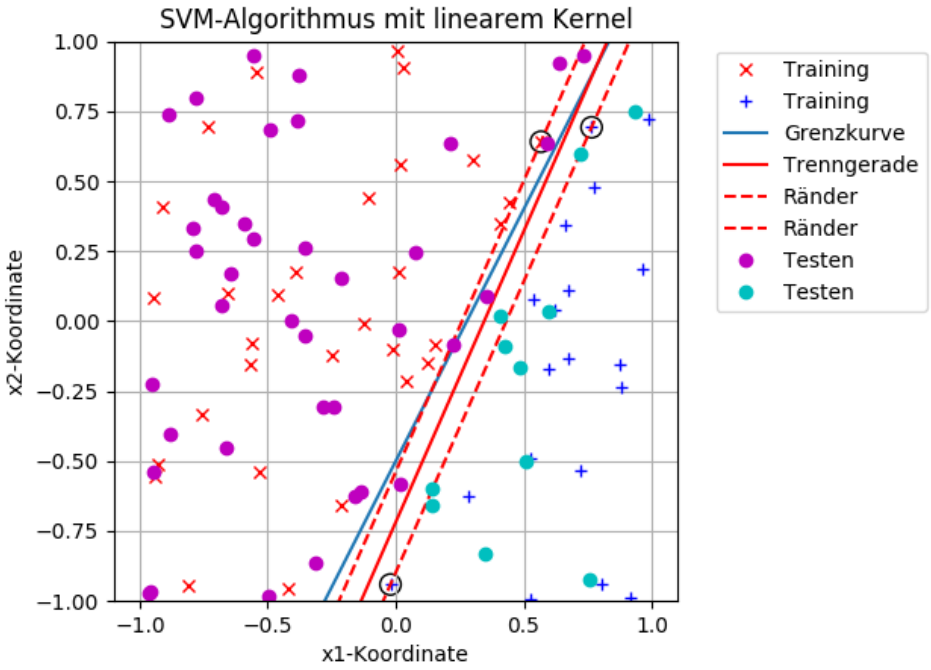

Abb. 15: Anwendung des linearen Support-Vector-Machine-Algorithmus auf die Klassifizierung von Punkten, die ober- oder unterhalb der Geraden $x_2 = 1.8x_1 - 0.7$ liegen

die gewünschte Klassifizierungsentscheidung finden. Dazu wird mit Hilfe der Trainingsdaten ermittelt, welche Verzweigungen den größten Informationsgewinn oder die geringste Rate an Fehlklassifikation liefern. Den Informationsgewinn kann man mit dem Entropiemaß der Information und die Fehlklassifikationen mit der sogenannten Gini-Unreinheit [25] bestimmen. Beide Maße hängen von den Auftretenswahrscheinlichkeiten für richtige und falsche Klassifikationen aufgrund der gegebenen Merkmalsausprägungen ab und sie werden zunächst für alle möglichen Auswahlentscheidungen auf einer Ebene des Baumes ermittelt. Diejenige mit dem größten Informationsgewinn (kleinster Entropie) bzw. kleinster Gini-Unreinheit wird dann als bester Knoten gewählt. Danach wird für die Datensätze, die diese Entscheidung trennt, wiederum eines der obigen Auswahlkriterien angewendet und der nächste Knoten bestimmt. Das geschieht solange, bis alle Trainingsdatensätze aufgebraucht, d.h. zugeordnet sind.

Das Entscheidungsbaumverfahren kann aber nicht nur binäre Ja/Nein-Datensätze bewerten , sondern es kann auch kontinuierliche Zahlenwertdaten verarbeiten. Dazu werden nach [23] die Zahlenreihen der Merkmale der Datensätze der Größe nach sortiert und Mittelwerte gebildet, die zusätzliche Klassengrenzen für die Unterteilung der Daten mittels Ja/Nein-Entscheidungen bilden. Man unterteilt also die Zahlenwerte in Bereiche, in die man einträgt, ob der entsprechende Wert enthalten ist (ja) oder nicht enthalten ist (nein). Dadurch ergeben sich zusätzliche Entscheidungsknoten, die mit binären Entscheidungen abgefragt werden können.

Vorteile des Entscheidungsbaumlernens sind nach [27] seine Übersichtlichkeit und gute anschauliche Verständlichkeit, wodurch man die getroffenen Entscheidungen leichter nachvollziehen kann. Weiter ist das Verfahren nicht sensibel gegen offensichtlich fehlerhafte Daten, sogenannte Ausreißer. Ein Nachteil ist, dass sich der Ereignisbaum sehr stark an die Testdaten anpassen kann, die er dann vollständig korrekt erkennt, aber dadurch zu streng klassifiziert, so dass leicht abweichende Testdaten nicht mehr richtig verarbeitet werden können. Das ist insbesondere der Fall, wenn der Baum mit Blättern endet, die jeweils nur noch zwischen zwei Datensätzen entscheiden. Dem kann etwas entgegengewirkt werden, wenn man vorschreibt,

dass Blätter mit einer Mindestzahl von Daten enden müssen, die > 1 ist.

Eine weitere Verbesserungsmöglichkeit ist das sogenannte Random-Forest-Verfahren. Dabei werden verschiedene Bäume durch Auswahl von Teilmengen der Trainingsdaten erstellt, deren Ergebnisse dann gemittelt werden. Dadurch wird das Modell robuster gegen Veränderungen in den Daten und auch die Überanpassung an den Trainingsdatensatz wird reduziert (vgl. [26]).

Da die Auswahlmethodik für die Knoten des Baumes mittels Entropie- oder Gini-Kriterium insbesondere für kontinuierliche Daten rasch sehr aufwändig werden kann, wurde auch hier zur Veranschaulichung ein Beispiel mit Hilfe des in Python bereitgestellten Hilfsprogramms DecisionTreeClassifier aus dem Programmpaket Scikit-Learn berechnet, das auch in [23] angegeben ist. Dabei handelt es sich um die Entscheidung, ob abhängig von Temperatur, Luftfeuchtigkeit, Windgeschwindigkeit und Regenwetter Fussball gespielt werden soll oder nicht. Dafür sind 8 Datensätze mit bekannten Entscheidungen gegeben und es soll mit Hilfe der oben beschriebenen Auswahlkriterien ein geeigneter Entscheidungsbaum erstellt werden. Der zugehörige Programmcode mit den Eingabedaten ist in Abbildung 16 gezeigt. Der damit automatisch von Python erstellte Entscheidungsbaum ist in Abbildung 17 dargestellt. Man erkennt, dass die Knoten nach abnehmender Gini-Impurity angeordnet sind und das Kriterium einer Luftfeuchtigkeit von mehr als 75 % bereits für 3 Datensätze das Spielen auszuschließt. Die restlichen Datensätze können dann ebenfalls nach dem Kriterium von mehr oder weniger als 55 % Luftfeuchtigkeit zur Entscheidung Spielen oder nicht Spielen klassifiziert werden. Es reicht demnach bei diesem Datensatz bereits das Merkmal der Luftfeuchtigkeit für die Entscheidung aus.

```python
# Importiere die notwendigen Bibliotheken
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Erstelle ein Beispiel-Array (Tabelle) für Training
# [Temperatur,Luftfeuchtigkeit,Windgeschwindigkeit,Regen,Kennzeichnung]
# [Grad Celsius, %, m/s, 0/1, 0/1], wobei 0 = nein und 1 = ja bedeutet
data = [
    [20, 65, 10, 0, 1],  # Schönes Wetter, wir spielen Fußball
    [25, 80, 5, 1, 0],   # Es regnet, wir spielen nicht
    [18, 70, 15, 0, 1],  # Noch gutes Wetter, wir spielen
    [10, 90, 20, 1, 0],  # Kalt und regnerisch, wir spielen nicht
    [15, 85, 25, 0, 0],  # Zu windig, wir spielen nicht
    [22, 60, 5, 0, 1],   # Schönes Wetter, wir spielen
    [30, 50, 10, 0, 0],  # Zu heiß, wir spielen nicht
    [20, 70, 8, 0, 1]    # Schönes Wetter, wir spielen
]

# Trenne die Merkmale (Features) und das Ziel (Target)
X = [row[:-1] for row in data]
y = [row[-1] for row in data]

# Erstelle und trainiere den Entscheidungsbaum
tree_classifier = DecisionTreeClassifier()
tree_classifier.fit(X, y)

# Visualisiere den Baum
plt.figure(figsize=(10, 7))
plot_tree(tree_classifier,
          filled=True,
          feature_names=['Temperatur', 'Luftfeuchtigkeit', 'Windgeschwindigkeit', 'Regen'],
          class_names=['Nicht spielen', 'Spielen'],
          fontsize=10)
plt.show()
```

Abb. 16: Programmcode für Entscheidungsbaum-Lernen am Beispiel Fussballspielen abhängig von Wetterbedingungen nach [23].

Fügt man jedoch zwei weitere Trainingsdatensätze hinzu, die das Spielen bei Kälte ohne Regen und ohne starken Wind ([5, 30, 1, 0, 1]) und das nicht Spielen bei starkem Wind trotz günstiger andere Bedingungen ([20, 70, 30, 0, 0]) einbringen, so ergibt sich ein deutlich abweichendes Baumdiagramm, das nun auch noch die Temperatur- und Windkriterien berücksichtigt, was realistischer ist. Das Ergebnisdiagramm ist in Abbildung 18 dargestellt. Es muss also bei den

Trainingsdaten darauf geachtet werden, dass alle typischen Situation, die bei der Entscheidung einer Rolle spielen, ausreichend abgedeckt sind.

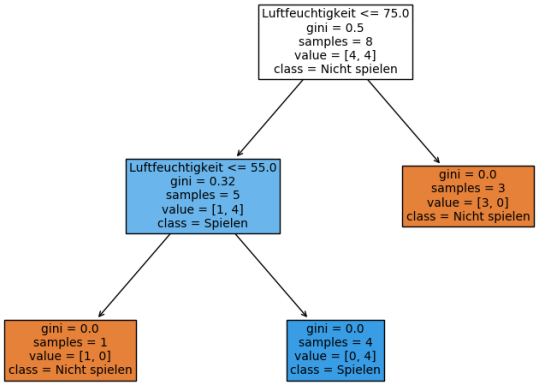

Abb. 17: Entscheidungsbaum für wetterabhängiges Fussballspielen mit Scikit-Learn für die Trainingsdaten nach Abbildung 16.

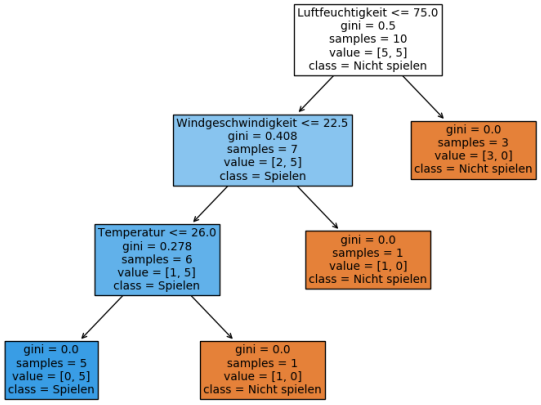

Abb. 18: Entscheidungsbaum für wetterabhängiges Fussballspielen mit Scikit-Learn für erweiterte Trainingsdaten.

Um die oben angegebenen Kriterien für die optimale Auswahl der Merkmalsabfragen im Entscheidungsbaum etwas besser zu verstehen, wurde für die Fussball-Aufgabenstellung mit dem ursprünglichen Datensatz von 8 Merkmalskombinationen, wie sie in Abbildung 16 angegeben sind, ebenfalls ein kleines Python-Programm erstellt, das die Merkmalswahl für die Entscheidungsebenen direkt aus der Minimierung des Gini-Kriteriums bzw. der Entropie trifft. Da bei diesem Beispiel die Wettermerkmale nicht binär gegeben sind, muss zuerst eine Umwandlung in binäre Daten erfolgen, was durch die oben genannte Einteilung der Zahlenwertbereiche in Intervalle geschieht, in denen ein Datensatz dann enthalten ist oder nicht enthalten ist. Durch die Mittelwertbildung ergeben sich zusätzlich zur bereits binär vorliegenden Regen-Klasse somit folgende Klassengrenzen

- Temperatur T: $< 12.5, < 16.5, < 19, < 21, < 23.5, < 27.5$ Grad Celsius
- Luftfeuchtigkeit H: $< 55, < 62.5, < 67.5, < 75, < 82.5, < 87.5$ %
- Windgeschwindigkeit V: $< 6.5, < 9, < 12.5, < 17.5, < 22.5$ m/s ,

was auf insgesamt 18 binär codierte Merkmalsklassen führt. Die Zielsetzung besteht nun darin, den Baum möglichst kompakt und übersichtlich zu gestalten. Dazu sollte der Informationsgewinn in jeder Entscheidungsebene möglichst groß sein. Um den Informationsgewinn zu bestimmen, wird entweder die Shannon'sche Informationsentropie oder der Gini-Koeffizient verwendet [31]. Beide Größen messen die Ungleichheit oder Ungeordnetheit der Ausprägungen der Attribute. Große Einheitlichkeit bedeutet große Übereinstimmung von bestimmten Merkmalsausprägungen und Ergebnisausprägungen, was zum raschen Sortieren der Beispieldatensätze im Sinne der Auswahl von möglichst aussagekräftigen Attributen für die schnelle Entscheidungsfindung führt. Die Auswahl der Baumknoten soll also möglichst rasch die Anzahl der verbleibenden Auswahlmöglichkeiten verringeren.

Man prüft darum zunächst alle Merkmale bzw. Attribute auf den mit ihrer Anwendung als Knoten verbundenen Informationsgewinn. Das Merkmal mit dem höchsten Informationsgewinn wird dann für den sogenannten Wurzelknoten ausgewählt. Dadurch zerfallen die Trainingsdatensätze in zwei Gruppen, wobei eine Gruppe auch bereits eine Entscheidung ermöglichen kann, wie es in den vorigen Beispielen der Fall war. Im ersteren Fall wird das

Tabelle 4: Formeln für die Berechnung des optimalen Informationsgewinns beim Entscheidungsbaumlernen nach [31] mit
E = verfügbare Beispielmenge
E_j = Teilmenge der Beispielmenge mit Ausprägung j (hier 0 oder 1 mit n=2)
B = Attribut bzw. Merkmal aus den Trainingsdatensätzen
b_i = Ausprägung der Zielwerte (hier 0 oder 1 mit k=2)
$p(b_i)$ = Wahrscheinlichkeit für das Eintreffen von b_i im Zielattribut der Trainingsmenge

Auswahlverfahren	Gini-Methode	Entropie-Methode
Wahrscheinlichkeit $p(b_i)$	$p_j(b_i) = \frac{\text{Anzahl von korrekten 0 oder 1 in B bezogen auf } E_j}{\text{Anzahl 0 oder 1 im betrachteten Attribut B}}$	
Ungleichheitsmaß	$\text{gini}_B(E_j) = 1. - \sum_{i=1}^{k} p_j(b_i)^2$	$I_B(E_j) = \sum_{i=1}^{k} -p_j(b_i) log_2 p_j(b_i)$
Gain	GINI(B)= $\sum_{j=1}^{n} \frac{\lvert E_j \rvert}{E} \text{gini}_B(E_j)$	$G(B) = \sum_{j=1}^{n} \frac{\lvert E_j \rvert}{E} I(E_j)$
Gewinn	Gewinn(B)=gini(E)-GINI(B)	Gewinn(B)=I(E)-G(B)

obige Verfahren analog in jeder der beiden Gruppen wiederum durchgeführt, bis die gesuchte Entscheidung getroffen werden kann. Im letzteren Fall reduziert sich die Anzahl der weiter zu behandelnden Trainingsbeispiele entsprechend und es muss nur noch ein Ast des Baumes weiter untersucht werden.

Die für die Berechnung des größten Informationsgewinns verwendeten Formeln sind in Tabelle 4 zusammengestellt. Zuerst muss für jedes Attribut festgestellt werden, wie oft es in den Trainingsdatensätzen als gegeben (1) bzw. nicht gegeben (0) vorkommt. Danach wird ermittelt, wie oft die gegebenen Fälle des Merkmalsattributs mit der zugehörigen Untermenge der Ergebnisse, also dem Zielattribut (hier Spielen oder nicht Spielen) übereinstimmen. Damit ergeben sich nach Tabelle 4 bei k=n=2, also bei je 2 Möglichkeiten 0 oder 1 für die Attribut- und Ergebnisdatenausprägungen 4 Wahrscheinlichkeiten für $p_j(b_i)$.

In Abbildung 19 ist die Matrix der Trainingsbeispiele nach Umwandlung in binär codierte Merkmale angegeben. Berechnet man bei Wahl von Spalte 0 (entsprechend $T < 12.5$ Grad Celsius) den damit verbundenen Informationsgewinn, so findet man 7 mal die 1 und 1 mal die 0 in dieser Spalte, die mit den korrespondierenden Werten in der Ergebnisspalte 18 zu verknüpfen sind. Von den 7 Einsen in Spalte 0 sagen 6 das Ergebnis E richtig voraus, eine Eins führt jedoch auf eine 0 in der Ergebnisspalte. Damit ergibt sich $p_1(b_1 = 1) = 4/7$ und $p_1(b_2 = 0) = 3/7$. Entsprechend sagt die verbleibende Null in Spalte 0 das korrespondierende Ergebnis 0 in der Ergebnisspalte richtig voraus und es ergibt sich $p_2(b_2 = 0) = 1/1$ und daraus folgend $p_2(b_1 = 1) = 0$, weil diese Kombination gar nicht auftritt. Damit kann nun der Gini-Index bzw. die Entropie für j=1 und j=2 für diese Spalte ausgerechnet werden, was jeweils die Reinheit oder Gleichmäßigkeit bzw. die Informationsentropie der Einsen bzw. Nullen dieser Attributspalte ergibt. Der gesamte Gini-Index bzw. die Gesamtentropie wird schließlich als mit den Häufigkeiten der positiven oder negativen Zuordnungen in B gewichtetes Mittel berechnet. Diese in Formelzeile 3 von Tabelle 4 angegebene Größe wird auch als Gain bezeichnet [31]. Aus ihr erhält man schließlich den Gini-Gewinn oder Informationsgewinn als Differenz von Gini-Index bzw. Informationsentropie der Ergebnisspalte und Gain, der in der vierten Zeile von Tabelle 4

angegeben ist. Für alle anderen Spalten geht man dann genau so vor und ermittelt danach die Tabellenspalte mit dem größten Gewinnwert. Das zu dieser Spalte gehörenden Attribut wählt man schließlich als Wurzelknoten. Im Fall des vorliegenden Beispiels ergibt sich daraus die Spalte 9 entsprechend einer Luftfeuchtigkeit von < 75 %, wie aus Bild 19 hervorgeht.

Weil die 3 Fälle mit Luftfeuchtigkeit $> 75\%$ bereits zur Entscheidung führen, dass nicht gespielt wird, muss der Nein-Zweig nicht weiter bearbeitet werden. Im Ja-Zweig kann man nun den Datensatz durch Weglassen der Fälle 2, 4 und 5, in denen wegen zu hoher Luftfeuchtigkeit nicht gespielt wird, verkleinern und es entfallen auch alle Spalten mit größeren Luftfeuchtigkeiten als 75 oder mehr Prozent. Auf diesen reduzierten Datensatz wendet man nun die gleiche Vorgehensweise wie oben geschildert an und erhält so den nächsten Baumknoten für die zweite Ebene des Entscheidungsbaumes. Die Resultate sind in Abbildung 20 dargestellt und führen auf 3 Möglichkeiten mit maximalem Wert der Gewinnfunktion, der jeweils in den Spalten 4, 5 und 6 angenommen wird. Bei Wahl von Spalte 5, was einer Temperatur von kleiner oder gleich 27,5 Grad Celsius entspricht, ergibt sich der in Abbildung 21 dargestellte Ereignisbaum. Man ersieht aus diesem Sachverhalt, dass das gewählte Optimierungsverfahren, nicht zwingend eine eindeutige Lösung ergeben muss. Hier sind drei gleichwertige Bäume möglich, wovon einer auch dem mit Scikit-Learn zuvor ermittelten Baum entspricht.

```
Gini-Index des gesamten Datensatzes = 0.5
Entropie des gesamten Datensatzes = 1.0


| |  0|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12| 13| 14| 15| 16| 17|| 18|
|-------------------------------------------------------------------------------|
| |<T1|<T2|<T3|<T4|<T5|<T6|<H1|<H2|<H3|<H4|<H5|<H6|<V1|<V2|<V3|<V4|<V5|  R||  E|
|-------------------------------------------------------------------------------|
|1|  1|  1|  1|  0|  0|  0|  1|  1|  0|  0|  0|  0|  1|  1|  0|  0|  0|  0||  1|
|2|  1|  1|  1|  1|  1|  0|  1|  1|  1|  1|  0|  0|  0|  0|  0|  0|  0|  1||  0|
|3|  1|  1|  0|  0|  0|  0|  1|  1|  1|  0|  0|  0|  1|  1|  1|  0|  0|  0||  1|
|4|  0|  0|  0|  0|  0|  0|  1|  1|  1|  1|  1|  1|  1|  1|  1|  1|  0|  1||  0|
|5|  1|  0|  0|  0|  0|  0|  1|  1|  1|  1|  1|  0|  1|  1|  1|  1|  1|  0||  0|
|6|  1|  1|  1|  1|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0||  1|
|7|  1|  1|  1|  1|  1|  1|  0|  0|  0|  0|  0|  0|  1|  1|  0|  0|  0|  0||  0|
|8|  1|  1|  1|  0|  0|  0|  1|  1|  1|  0|  0|  0|  1|  0|  0|  0|  0|  0||  1|

|Spalte| I(E)-G(B) | gini(E)-GINI(B) |
|-----------------------------------|
|     0|      0.596|            0.071|
|     1|      0.708|            0.167|
|     2|      0.577|            0.033|
|     3|      0.471|            0.033|
|     4|      0.604|            0.167|
|     5|      0.542|            0.071|
|     6|      0.596|            0.071|
|     7|        0.5|              0.0|
|     8|      0.471|            0.033|
|     9|       0.71|              0.3|
|    10|      0.604|            0.167|
|    11|      0.542|            0.071|
|    12|        0.5|              0.0|
|    13|      0.471|            0.033|
|    14|      0.471|            0.033|
|    15|      0.604|            0.167|
|    16|      0.542|            0.071|
|    17|      0.604|            0.167|

1. Entscheidungsebene:
Maximaler Informationsgewinn in Spalte 9 = 0.71
Maximaler Ginigewinn in Spalte 9 = 0.3
```

Abb. 19: Auswahl des Wurzelknotens für die 1. Ebene des Entscheidungsbaumes für das Fussball-Beispiel.

```
Gini-Index des reduzierten Datensatzes im Ja-Zweig = 0.32
Entropie des reduzierten Datensatzes im Ja-Zweig = 0.722

| |  0|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12| 13| 14|| 15|
|-------------------------------------------------------------------|
| |<T1|<T2|<T3|<T4|<T5|<T6|<H1|<H2|<H3|<V1|<V2|<V3|<V4|<V5|  R||  E|
|-------------------------------------------------------------------|
|1|  1|  1|  1|  0|  0|  0|  1|  1|  0|  1|  1|  0|  0|  0|  0||  1|
|2|  1|  1|  0|  0|  0|  0|  1|  1|  1|  1|  1|  1|  0|  0|  0||  1|
|3|  1|  1|  1|  1|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0||  1|
|4|  1|  1|  1|  1|  1|  1|  0|  0|  0|  1|  1|  0|  0|  0|  0||  0|
|5|  1|  1|  1|  0|  0|  0|  1|  1|  1|  1|  0|  0|  0|  0|  0||  1|

|Spalte| I(E)-G(B) | gini(E)-GINI(B) |
|-------------------------------------|
|     0|      0.464|             -0.0|
|     1|      0.464|             -0.0|
|     2|      0.473|             0.02|
|     3|      0.522|             0.12|
|     4|      0.722|             0.32|
|     5|      0.722|             0.32|
|     6|      0.722|             0.32|
|     7|      0.522|             0.12|
|     8|      0.405|            0.053|
|     9|      0.473|             0.02|
|    10|      0.488|            0.053|
|    11|      0.322|             0.02|
|    12|      0.258|              0.0|
|    13|      0.258|              0.0|
|    14|      0.258|              0.0|

2. Entscheidungsebene:
Maximaler Informationsgewinn in Spalte 5 = 0.7219
Maximaler Ginigewinn in Spalte 5 = 0.32
Spalten 4 und 6 sind ebenfalls möglich
```

Abb. 20: Auswahl des Knotens für die 2. Ebene des Entscheidungsbaumes für das Fussball-Beispiel.

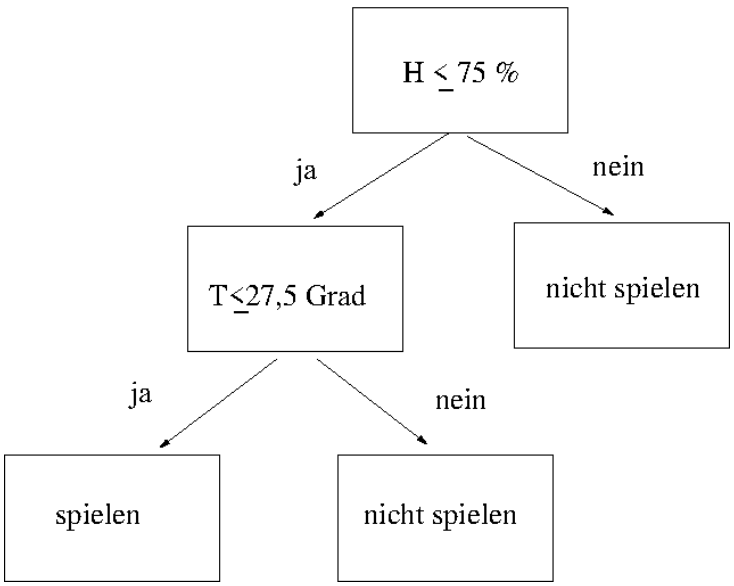

Abb. 21: Alternatives Entscheidungsbaumdiagramm für das Fussball-Beispiel.

3.6 Naive Bayes-Klassifikation

Mit der naiven Bayes-Klassifikation wird ein Datensatz derjenigen Zielklasse zugeordnet, für die aufgrund des Vorwissens über bekannte Zuordnungen die größte Wahrscheinlichkeit besteht. Dazu wird die bedingte Wahrscheinlichkeit $P(Klassenzugehörigkeit|Datensatzmerkmale)$ für eine bestimmte Klassenzugehörigkeit bei einem vorgegebenen Datensatz ermittelt. Das geschieht mit Hilfe des Bayes'schen Satzes, [32], der diese Wahrscheinlichkeit aus der umgekehrten bedingten Wahrscheinlichkeit für das Auftreten der jeweiligen Merkmale bei gegebener Klassenzugehörigkeit zu berechnen erlaubt. Das ist mit dem Wissen über vorherige Zuordnungen möglich. Weiter benötigt der Bayes'sche Satz noch die ebenfalls vorab bekannte Wahrscheinlichkeit der Klassenzuordnung der Trainingsdaten ohne Berücksichtigung der Einflüsse der Merkmalszusammensetzungen. Da die einzelnen Merkmale eines Datensatzes jeweils vorhanden oder nicht vorhanden sein können, gibt es bei n Merkmalen 2^n mögliche Ausprägungen bzw. Merkmalskombinationenn, die im Folgenden als K_i mit i=1, ... 2^n bezeichnet werden. Bezeichnet man weiter im Fall von nur einer vorhandenen Klasse die Klassenzugehörigkeit mit y, so lautet der Bayes'sche Satz

$$P(y|K_i) = \frac{P(K_i|y) \cdot P(y)}{P(K_i)} \quad . \tag{27}$$

$P(y)$ bezeichnet dabei die Vorwahrscheinlichkeit der Klassenzugehörigkeit der Merkmalsausprägung eines Datensatzes und $P(y')$ entsprechend die Vorwahrscheinlichkeit dafür, dass die Merkmalsausprägung des Datensatzes nicht zur Klasse gehört, was auch als Priori-Wahrscheinlichkeit bezeichnet wird. Demnach gilt für die Priori-Wahrscheinlichkeiten, dass $P(y) + P(y') = 1$ ist. Das Ergebnis $P(y|K_i)$ wird als die Posteriori-Wahrscheinlichkeit für y bezeichnet, weil diese zusätzlich die bedingten Wahrscheinlichkeiten der Merkmalsausprägungen bei bekanntem Ausgang $P(K_i|y)$ einbezieht. Die bedingten Wahrscheinlichkeiten $P(K_i|y)$ werden auch Likelihood-Werte genannt, während der Nenner $P(K_i)$ als Normierungs- oder Bezugsgröße verstanden werden kann. Weil die Ereignisse y und y' alle möglichen Ergebnisse abdecken, bilden sie nach [36] ein sogenanntes vollständiges Ereignissystem Ω mit $\Omega = y + y'$. Damit kann man den Nenner von Gleichung 27 umformen in

$$\begin{aligned} P(K_i) =& P(K_i \cap \Omega) \\ =& P(K_i \cap (y \cup y')) \\ =& P(K_i \cap y) + P(K_i \cap y') \\ =& P(K_i|y) \cdot P(y) + P(K_i|y') \cdot P(y') \quad , \end{aligned} \tag{28}$$

wobei in der letzten Zeile von Gleichung 28 wieder der Bayes'sche Satz angewendet wurde. Gleichung 28 heißt auch der Satz über die vollständige Wahrscheinlichkeit und seine Herleitung findet man z.B. im Buch von Kolmogoroff [37], wo die Wahrscheinlichkeitsrechnung auf der Grundlage der Mengenlehre beschrieben wird. In dieser Betrachtungsweise stellen die in Gleichung 28 erscheinenden Mengen $K_i \cap y$ und $K_i \cap y'$ sogenannte Elementarereignisse dar, aus denen schließlich die hier gesuchten bedingten Wahrscheinlichkeiten berechnet werden können. Die Summe der Wahrscheinlichkeiten der Elementarereignisse muss dabei 1 gegeben, da eines von ihnen auf jeden Fall eintritt und ihre Vereinigung ebenfalls die Menge Ω des bereits genannten vollständigen Ereignissystems für die Ergebnisklasse y bildet. Die entsprechenden Ereignismengen sind in Abbildung 22 skizziert, die auch die Herleitung des Satzes von der totalen Wahrscheinlichkeit nochmals veranschaulicht.[2] Die

[2]Betrachtet man nun weiter eine Zusammenstellung von Teilmengen von Ω, worin Ω selbst, die leere Menge sowie alle Vereinigungs-, Schnitt- und Differenzmengen der Elemente wieder enthalten sind, und verlangt, dass für zwei disjunkte Elemente A und B daraus gilt, dass $P(A + B) = P(A) + P(B)$, so lassen sich damit alle Wahrscheinlichkeiten für diese Teilmengen sowie daraus abgeleitete Größen ermitteln [38].

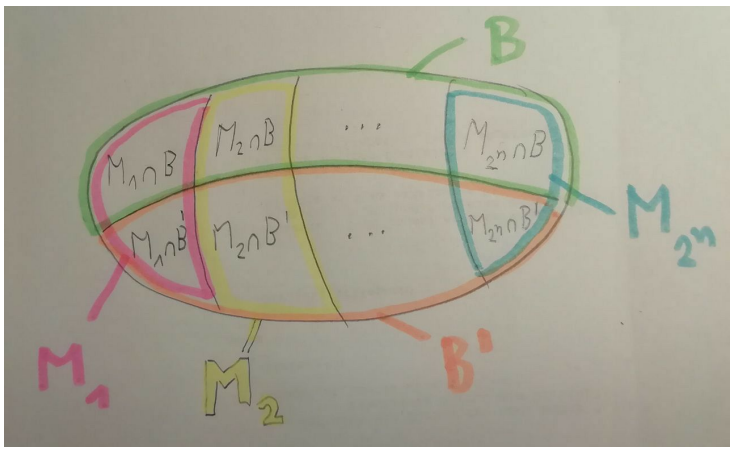

Abb. 22: Veranschaulichung der Mengen der Elementarereignisse und der Zerlegung der Grundmenge Ω mit y=B und y'=B' und $K_i = M_i$.

Vorgangsweise nach Gleichung 28 wird nach [35] als Marginalisierung bezeichnet. Damit wird die Bezugsgröße der gesuchten Wahrscheinlichkeiten auf die Summe der mit den Vorwahrscheinlichkeiten gewichteten Likelihood-Werte der einzelnen Merkmalsausprägungen zurückgeführt. Diese Bezugswahrscheinlichkeiten werden auch Randwahrscheinlichkeiten genannt, wie aus der unten gezeigten Tabelle 6 ersichtlich ist. Nimmt man an, dass die in der Merkmalskombination K_i enthaltenen Merkmale voneinander unabhängig sind, so lassen sich die bedingten Wahrscheinlichkeiten der Likelihood-Werte analog zur Wahrscheinlichkeit gewöhnlicher unabhängiger Wahrscheinlichkeitskombinationen auch als Produkt der Einzelwahrscheinlichkeiten angeben. Dann gilt nach [34] für die Likelihood-Wahrscheinlichkeiten

$$P(K_i|y) = \prod_{k=1}^{n} P(m_{ik}|y) \quad \text{mit} \quad K_i = (m_{i1}, ..., m_{in}) \quad \text{und} \quad i = 1, ..., 2^n \quad , \tag{29}$$

wobei m_{ik} die einzelnen Merkmale, also die Komponenten der Merkmalskombination K_i, die jeweils gegeben oder nicht gegeben sein können, bezeichnen. Wegen der vereinfachenden und nicht immer zutreffenden Annahme, dass alle Merkmale voneinander unabhängig sind, wird das Verfahren als naiver Bayes-Klassifikator bezeichnet. Setzt man die Gleichungen 28 und 29 in Gleichung 27 ein, so erhält man schließlich für die gesuchte Wahrscheinlichkeit der Zugehörigkeit eines neuen Datensatzes zu Klasse y den Ausdruck

$$P(y|K_i) = \frac{\prod_{k=1}^{n} P(m_{ik}|y) \cdot P(y)}{P(y) \prod_{k=1}^{n} P(m_{ik}|y) + P(y') \prod_{k=1}^{n} P(m_{ik}|y')} \quad . \tag{30}$$

Damit ist die Klassifizierung eines neuen Datensatzes auf Größen zurückgeführt, die sich aus den gegebenen Vorinformationen unmittelbar berechnen lassen. Diese Vorinformationen nehmen also eine den Trainingsdaten bei den anderen Verfahren entsprechende Rolle ein. Weil sich die Trainingsdaten hier in Kombinationen von binären Merkmalsausprägungen unterteilen lassen, kann deren Anzahl bzw. Häufigkeit für die Abschätzung der in Gleichung 30 benötigten Wahrscheinlichkeiten verwendet werden. Demnach ergeben sich bei der Klassifikation mit dem naiven Bayes-Verfahren keine klaren Ja/Nein-Entscheidungen. Stattdessen lassen sich für jede Merkmalskombination lediglich Wahrscheinlichkeiten für die Zugehörigkeit oder Nicht-Zugehörigkeit zur Ergebnisklasse angeben. Das erlaubt allerdings auch eine detaillierte Bewertung und Reihung der möglichen Merkmalskombinationen, was nun am Beispiel der Chancen für das Bestehen der Mathematik-Matura abhängig vom Vorbereitungsstil der Kandidatinnen und Kandidaten illustriert wird.

Tabelle 5: Merkmalsausprägungen und bedingte Wahrscheinlichkeiten der Einzelmerkmale bei gegebenem positivem oder negativem Ausgang der Prüfung

Merkmale (M)		Positiv (B) 22	$P(M\|B)$	Negativ (B') 7	$P(M\|B')$
Regelmäßig im Unterricht	U	21	21/22	4	4/7
Nicht regelmäßig im Unterricht	U'	1	1/22	3	3/7
Stoff geübt	S	16	16/22	2	2/7
Stoff nicht geübt	S'	6	6/22	5	5/7
Daheim geblieben	D	14	14/22	1	1/7
Veranstaltungen besucht	D'	8	8/22	6	6/7

Das hier durchgerechnete Beispiel ist eine Erweiterung eines Anwendungsfalles mit zwei Merkmalen aus [34] auf drei Merkmale. Dabei sind als Vorinformation Daten von 29 Schülerinnen und Schülern gegeben, von denen 22 die Mathematik-Reifeprüfung bestanden und 7 nicht bestanden haben. Die Zugehörigkeit zur Klasse y wird im Folgenden daher mit B, die zu y' mit B' bezeichnet. Über die Vorbereitungsmerkmale sind die in Tabelle 5 angegebenen Informationen bekannt. Daraus soll mit Hilfe des naiven Bayes-Klassifikators für künftige Kandidaten die Wahrscheinlichkeit dafür, die Prüfung zu bestehen, in Abhängigkeit von deren Vorbereitungsmerkmalen ermittelt werden. An diesem Beispiel kann man auch bereits die generelle Problematik der naiven Bayes-Klassifikation betreffend die Unabhängigkeit und Vollständigkeit der Merkmale erkennen. Die betrachteten Merkmale sind

- regelmäßige Teilnahme am Mathematikunterricht (gekennzeichnet mit U bzw. U' für Negation)
- intensive Durcharbeitung des Prüfungsstoffes (gekennzeichnet mit S bzw. S' für Negation)
- daheim bleiben und keine Veranstaltungsbesuche in der Vorbereitungszeit (gekennzeichnet durch D bzw. D' für Negation)

Dabei kann man nun die Frage stellen, ob Kandidaten, die öfters im Unterricht fehlen, nicht auch wegen geringerem Interesse am Fach Mathematik den Stoff weniger intensiv durcharbeiten und vielleicht auch trotz Prüfungsdruck eher eine Vergnügungsveranstaltung in der Zeit vor der Prüfung besuchen werden. In diesem Fall wären die Merkmale dann nicht mehr unabhängig voneinander. Weiter ist offensichtlich, dass es noch weit mehr Einflüsse auf den Prüfungserfolg als die drei genannten Beispiele geben wird. Bei realen Anwendungen sollten die Datensätze also möglichst vollständig und unabhängig gewählt werden.

Der Gang der Berechnung für das gewählte einfache Beispiel lässt sich in Anlehnung an [33] übersichtlich tabellarisch darstellen, wie in Tabelle 6 gezeigt ist. Dort finden sich in der ersten Spalte die möglichen Merkmalskombinationen, von denen es hier $2^3 = 8$ gibt. In der zweiten Spalte sind die Produkte aus Likelihood und Priori-Wahrscheinlichkeiten für den Fall bestandener Prüfung angegeben, was nach dem Bayes'schen Satz mit der Wahrscheinlichkeit $P(K_i \cap B)$ identisch ist, und in der dritten Spalte die analogen Wahrscheinlichkeiten $P(K_i \cap B')$ für den Fall des Merkmals K_i und nicht bestandener Prüfungen, was den Zählerausdrücken in Gleichung 30 entspricht. Spalte 4 enthält dann die Summe von Spalte 2 und 3, was die Randwahrscheinlichkeiten bzw. den Nenner von Gleichung 30 darstellt. Damit lassen sich schließlich für jede Merkmalskombination die gesuchten Posteriori-Wahrscheinlichkeiten für das Bestehen oder Nicht-Bestehen der Prüfung durch Division des jeweiligen Spaltenelements durch die Randwahrscheinlichkeit der zugehörigen Zeile ermitteln. Diese Ergebniswahrscheinlichkeiten sind in den Spalten 5 und 6 angegeben. Die mit diesem Berechnungsschema für das Prüfungs-Anwendungsbeispiel erhaltenen Resultate sind in Tabelle 7 zusammengestellt.

Tabelle 6: Tabellarische Darstellung zur Berechnung der Klassifikationswahrscheinlichkeiten für die verschiedenen Merkmalskombinationen (Posteriori-Wahrscheinlichkeiten der für den Prüfungsausgang bei gegebenen Merkmalskombinationen) [%]

K_i	B	B'	Randwahrscheinlichkeiten	$P(B'\|K_i)$	$P(B\|K_i)$
K_1	$P(K_1 \cap B)$	$P(K_1 \cap B')$	$P(K_1) = P(K_1 \cap B) + P(K_1 \cap B')$	$\frac{P(K_1 \cap B')}{P(K_1)}$	$\frac{P(K_1 \cap B)}{P(K_1)}$
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
K_8	$P(K_8 \cap B)$	$P(K_8 \cap B')$	$P(K_8) = P(K_8 \cap B) + P(K_8 \cap B')$	$\frac{P(K_8 \cap B')}{P(K_8)}$	$\frac{P(K_8 \cap B)}{P(K_8)}$
	$P(B)$	$P(B')$	$P(B) + P(B') = \sum_{i=1}^{8} P(K_i) = 1$		

Tabelle 7: Ergebnisse des Anwendungsbeispiels entsprechend Tabelle 6 für den Prüfungsausgang bei gegebenen Merkmalskombinationen K_i [%]

i	K_i	$P(B)P(K_i\|B)$	$P(B')P(K_i\|B')$	Summe	$P(B'\|K_i)$	$P(B\|K_i)$
1	D'U'S'	6.33	0.34	6.68	94.88	5.12
2	D'U'S	2.53	0.91	3.45	73.53	26.47
3	D'US	8.44	7.18	15.63	54.04	45.96
4	D'US	3.38	19.15	22.53	14.99	85.01
5	DU'S'	1.06	0.6	1.65	63.82	36.18
7	DUS'	0.42	1.6	2.02	20.92	79.08
8	DU'S'	1.41	12.57	13.98	10.07	89.93
9	DUS	0.56	33.51	34.08	1.65	98.35

Die Klassifikationsergebnisse der einzelnen Merkmalskombinationen sind auch noch in Abbildung 23 als Balkendiagramme dargestellt. Wie zu erwarten hat man die größten Chancen von etwa 98,4 % für das Bestehen der Prüfung, wenn man immer am Unterricht teilnimmt, den Prüfungsstoff gut durcharbeitet und in der Vorbereitungszeit keine Vergnügungsveranstaltungen besucht. Wenn man all das nicht macht, hat man erwartungsgemäß die schlechtesten Chancen von nur etwa 5 % für einen positiven Prüfungsausgang. Ein weiteres interessantes und eventuell auch etwas beruhigendes Ergebnis ist, dass der Besuch von Veranstaltungen während der Prüfungsvorbereitungszeit keinen sehr starken Einfluss auf den zu erwartenden Prüfungserfolg hat, sofern man regelmäßig am Unterricht teilnimmt und den Stoff fleißig wiederholt, wie die Merkmalskombination 3 (D'US) mit einer zu erwartenden Erfolgsquote von 85 % zeigt.

Für die Behandlung im Unterricht kann man das Thema auch vielleicht etwas anschaulicher über Baumdiagramme angehen. Wenn man sich wieder wie in [34] auf zwei Merkmale, nämlich die regelmäßige Teilnahme am Unterricht (U) und das Üben des Lernstoffs (S) beschränkt, ergeben sich nur die 4 Merkmalskombinationen $K_1 = U'S'$, $K_2 = U'S$, $K_3 = US'$ und $K_4 = US$. Man kann den Prüfungserfolg der Vorbereitungsstrategien dann als diskrete Zufallsvariable X ansehen, deren Auftretenswahrscheinlichkeiten sich durch Kombination zweier Ereignisse ergeben. Das kann mit Hilfe von Baumdiagrammen veranschaulicht werden, die standardmäßig für alle weiterbildenden Schulen in Österreich zum Maturastoff gehören. Die beiden Ereignisse sind dabei der Prüfungserfolg sowie die Wahl der Vorbereitungsstrategie. Die Zufallsvariable X kann dann die 8 Ausgänge $X = \{B \cap K_1, B \cap K_2, B \cap K_3, B \cap K4, B' \cap K_1, B' \cap K_2, B' \cap K_3, B' \cap K4\}$ annehmen und die zugehörigen Wahrscheinlichkeiten können mit den aus dem Oberstufenunterricht bekannten Pfadregeln für Baumdiagramme berechnet werden, sofern die Wahrscheinlichkeiten der verschiedenen Verzweigungsmöglichkeiten bekannt sind.

Man kann das Baumdiagramm entweder rückblickend von den bekannten Prüfungserfolgen als

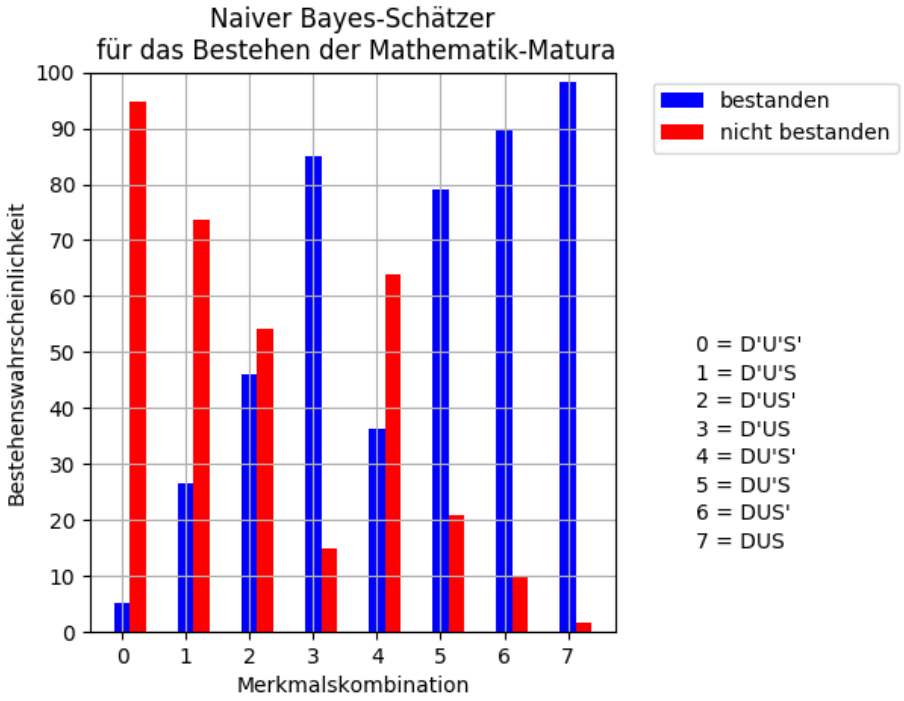

Abb. 23: Wahrscheinlichkeiten für das Bestehen der Mathematik-Reifeprüfung in Abhängigkeit von der Art der Vorbereitung (Merkmalskombinationen sind hier im Unterschied zu Tabelle 4 beginnend mit 0 indiziert)

erste Entscheidungsebene und der bekannten Verteilung der Vorbereitungsstrategien als zweite Entscheidungsebene her betrachten, wie es in der linken Hälfte von Abbildung 24 dargestellt ist. Aber auch die umgekehrte Reihenfolge mit Auswahl der Vorbereitungsstrategie in der ersten Ebene und darauffolgendem Prüfungsausgang in der nächsten Ebene ist möglich und führt auf denselben Satz von Ausprägungen der Zufallsvariable X des Prüfungserfolgs der Vorbereitungsstrategien. Das ist in der rechten Hälfte von Abbildung 24 gezeigt.[3]

Auf der linken Seite des Diagramms finden sich die gegebenen Größen, auf der rechten Seite findet man die gesuchten Schätzgrößen $P(B|K_i)$ bzw. $P(B'|K_i$. Aus den jeweiligen linken und rechten Pfaden, die auf dasselbe Ereignis $B \cap K_i$ bzw. $B' \cap K_i$ mit i=1,2,3,4 führen, ergibt sich direkt der Bayes'sche Satz, der somit zur Verfügung steht, falls er nicht im Unterricht behandelt worden ist. So ergibt sich beispielsweise aus dem obersten Pfad im Baumdiagramm

$$P(B) \cdot P(K_1|B) = P(B \cap K_1) = P(B|K_1) \cdot P(K_1). \tag{31}$$

Durch Umstellen nach $P(B|K_1)$ ergibt sich dann die gesuchte Schätzgröße

$$P(B|K_1) = \frac{P(K_1|B) \cdot P(B)}{P(K_1)} \tag{32}$$

sowie auf analogem Weg auch die weiteren gesuchten Schätzwerte. Für die unbekannte Größe $P(K_1)$ im Nenner von Gleichung 32 lässt sich aus der graphischen Veranschaulichung der möglichen Prüfungserfolge der Vorbereitungsstrategien in Abbildung 25 unmittelbar

[3]Durch die beidseitige Erreichbarkeit der Wahrscheinlichkeitsverteilung der Kombinationen von Vorbereitungsstil und Prüfungserfolg wird illustriert, dass für den Begriff der bedingten Wahrscheinlichkeit die zeitliche Reihenfolge der Ereignisse nicht von Belang ist, wie es die umgangssprachliche Bedeutung des Wortes „bedingt"nahelegt. Darauf wird bereits von Hausdorff in [39] hingewiesen, der stattdessen den Begriff der relativen Wahrscheinlichkeit verwendet.

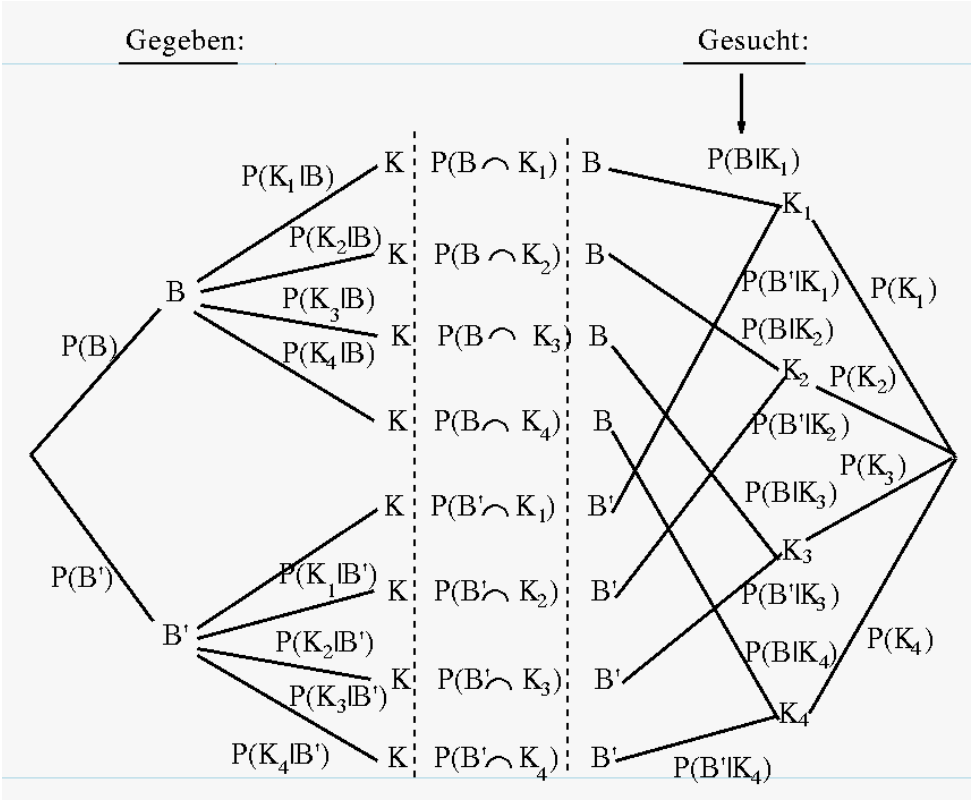

Abb. 24: Baumdiagramm für das Beispiel zum Bestehen der Mathematik-Reifeprüfung in Abhängigkeit von der Art der Vorbereitung bei 4 Kombinationen K_1 bis K_4 von je zwei Merkmalsausprägungen

die Zusammensetzung der Wahrscheinlichkeitswerte $P(K_i)$ mit i=1 bzw. 2,3,4 aus je zwei Summanden ablesen und unter Verwendung der Definitionsgleichung der bedingten Wahrscheinlichkeit sieht man, dass die Beziehung

$$P(K_i) = P(B \cap K_i) + P(B' \cap K_i) = P(B)P(K_i|B) + P(B')P(K_i|B') \tag{33}$$

gilt. Aus Abbildung 25 ist somit auch der hier benötigte Satz von der totalen Wahrscheinlichkeit unmittelbar ersichtlich. Die Unabhängigkeitsannahme für je zwei Merkmale aus $\{U, U', S, S'\}$ muss jedoch auch hier getroffen werden. Damit ergibt sich schließlich analog zur oben geschilderten Vorgangsweise

$$P(B|K_1) = \frac{P(U'|B) \cdot P(S'|B) \cdot P(B)}{P(U'|B) \cdot P(S'|B) \cdot P(B) + P(U'|B') \cdot P(S'|B') \cdot P(B')} \tag{34}$$

$$P(B|K_2) = \frac{P(U'|B) \cdot P(S|B) \cdot P(B)}{P(U'|B) \cdot P(S|B) \cdot P(B) + P(U'|B') \cdot P(S|B') \cdot P(B')} \tag{35}$$

$$P(B|K_3) = \frac{P(U|B) \cdot P(S'|B) \cdot P(B)}{P(U|B) \cdot P(S'|B) \cdot P(B) + P(U|B') \cdot P(S'|B') \cdot P(B')} \tag{36}$$

$$P(B|K_4) = \frac{P(U|B) \cdot P(S|B) \cdot P(B)}{P(U|B) \cdot P(S|B) \cdot P(B) + P(U|B') \cdot P(S|B') \cdot P(B')} \quad , \tag{37}$$

was mit Gleichung 30 für n=2 und i=1 identisch ist. Die gesuchten Wahrscheinlichkeiten für das Nicht-Bestehen der Prüfung erhält man analog, indem man B' jeweils durch B ersetzt. Sie ergeben sich aber auch unmittelbar als Gegenwahrscheinlichkeiten zu den oben berechneten Größen.

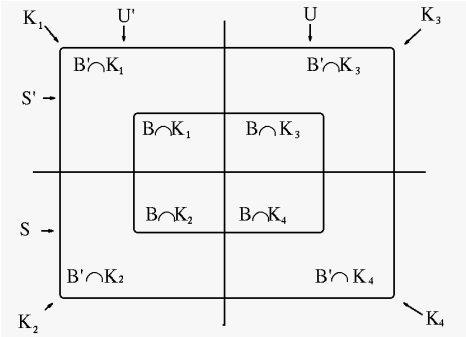

Abb. 25: Aufteilung der Merkmale und Merkmalskombinationen betreffend den Prüfungserfolg (B: bestanden, B': nicht bestanden) bei 4 Merkmalskombinationen K_1 bis K_4 aus je zwei Merkmalsausprägungen

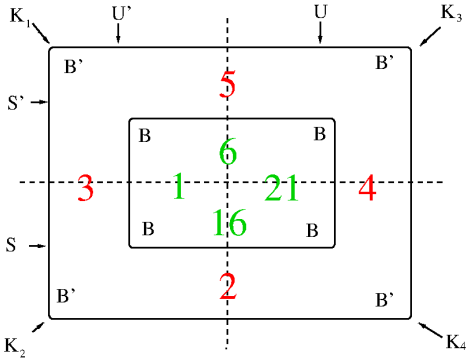

Abb. 26: Vorgegebene Informationen über die Summen von Merkmalskombinationen und Prüfungserfolgen bei 4 Merkmalskombinationen K_1 bis K_4 aus je zwei Merkmalsausprägungen

Anhand dieses gut überschaubaren Beispiels lässt sich noch die für das naive Bayes-Verfahren zu treffende Annahme der Unabhängigkeit der Merkmale U', U, S' und S überprüfen, weil man hier die Zusammensetzung der Merkmalskombinationen und ihre Zerlegung in bestandene und nicht bestandene Prüfungen wegen der geringen Anzahl von Testpersonen explizit ausrechnen kann. Damit können auch die gesuchten Wahrscheinlichkeiten für das Bestehen der Mathematik-Reifeprüfung abhängig vom Vorbereitungsstil direkt aus der Definitionsgleichung der bedingten Wahrscheinlichkeit

$$P(B|K_i) = \frac{P(B \cap K_i)}{P(K_i)} \tag{38}$$

berechnet werden. Die korrespondierenden Wahrscheinlichkeiten für das Nicht-Bestehen erhält man wiederum analog durch Ersetzen von B durch B'.

Dazu sind die gegebenen Daten in Tabelle 8 nochmals zusammengestellt. Daraus ergeben sich die in Abbildung 26 gezeigten Summen für die verschiedenen Merkmalskombinationen bei Erfolg oder Misserfolg in der Prüfung. Man kann demnach 8 Gleichungen für die 8 möglichen Fälle B'U'S', B'U'S, B'US', B'US, BU'S', BU'S, BUS'und BUS aufstellen, die in Tabelle 9 in Matrixform dargestellt sind. Die Gleichungen für bestandene und nicht bestandene Prüfungen sind voneinander enkoppelt und auch nicht linear unabhängig, wie man beim Umformen des Gleichungssystems auf Dreiecksform rasch sieht (vgl. Tabelle 10). Da die Lösungen ganzzahhlig sein müssen, ergeben sich aus den farbig markierten Gleichungen mit den niedrigsten Absolutgliedern auf der rechten Seite sechs mögliche ganzzahligge Lösungen. Und zwar können in der rot markierten Gleichung die Variablen nur die Werte 0 und 1 oder umgekehrt annehmen,

Tabelle 8: Auswirkungen der Prüfungsvorbereitungsmaßnahmen auf den positiven oder negativen Ausgang der Prüfung

Merkmale (M)	Positiv (B)	Negativ (B')	Summe
U mit S, S' beliebig	21	4	25
U' mit S, S' beliebig	1	3	4
Summe	22	7	29
S mit U, U' beliebig	16	2	18
S' mit U, U' beliebig	6	5	11
Summe	22	7	29

Tabelle 9: Gleichungssystem zur Bestimmung der Anzahl der Schüler mit den verschiedenen Vorbereitungsstilen und Prüfungserfolgen

B' U' S'	B' U' S	B' U S'	B' U S	B U' S'	B U' S	B U S'	B U S	
1	1							3
1		1						5
	1		1					2
		1	1					4
				1	1			1
				1		1		6
					1		1	16
						1	1	21

Tabelle 10: Gleichungssystem in Dreiecksform

B' U' S'	B' U' S	B' U S'	B' U S	B U' S'	B U' S	B U S'	B U S	
1	1							3
	1	-1						-2
		1	1					4
		1	1					4
				1	1			1
					1	-1		-5
						1	1	21
						1	1	21

Tabelle 11: Mögliche Lösungen für die Schülerzahlen der Merkmalskombinationen und Prüfungsausgänge sowie Gesamtschülerzahlen mit den Merkmalskombinationen K_i, i=1,2,3,4

	B' U' S'	B' U' S	B' U S'	B' U S	B U' S'	B U' S	B U S'	B U S	K_1	K_2	K_3	K_4
a)	3	0	2	2	1	0	5	16	4	0	7	18
b)	2	1	3	1	1	0	5	16	3	1	8	17
c)	1	2	4	0	1	0	5	16	2	2	9	16
d)	3	0	2	2	0	1	6	15	3	1	8	17
e)	2	1	3	1	0	1	6	15	2	2	9	16
f)	1	2	4	0	0	1	6	15	1	3	10	15

während sich aus der blau markierten Gleichung nur die Werte 0, 1 2 und korrespondierend 2,1,0 für die beiden beteiligten Variablen ergeben können. Damit erhält man die in Tabelle 11 dargestellten 6 möglichen Lösungen. Die verschiedenen Lösungen sind in Abbildung 27 graphisch dargestellt, wobei der innere Kreis jeweils die Anzahl der bestandenen Prüfungen und der äußere Ring die nicht bestandenen Prüfungen enthält. Für die Lösungsmöglichkeiten a) bis f) können nun nach Gleichung 38 die gesuchten bedingten Wahrscheinlichkeiten $P(B|K_i)$ bzw. $P(B'|K_i)$ direkt durch Quotientenbildung gemäß $P(K_i \cap B)/P(K_i)$ bzw. $P(K_i \cap B')/P(K_i)$ ausgerechnet und mit den Ergebnissen der naiven Bayes-Klassifizierung verglichen werden. Die Resultate sind in Tabelle 12 zusammengestellt und zeigen deutliche Abweichungen, was daran liegt, dass die Merkmale U, U', S und S' tatsächlich nicht unabhängig sind.

Zur Überprüfung der Unabhängigkeit der Merkmale wurde einerseits $P(K_i|B)$ bzw. $P(K_i|B')$ als Produkt der bedingten Wahrscheinlichkeit der Merkmale bei gegebenem B oder B' entsprechend Tabelle 6 berechnet, andrerseits wurden diese Werte für die Fälle a) bis f) aus Tabelle 11 durch Quotientenbildung $P(K_i \cap B)/P(B)$ bzw. $P(K_i \cap B')/P(B')$ ermittelt. Die dazu in Tabelle 13 angegebenen Resultate zeigen ebenfalls erhebliche Unterschiede, was die bereits vermutete wechselseitige Abhängigkeit der Merkmale, aus denen sich die Vorbereitungsstile zusammensetzen, bestätigt. Die Resultate sind in Abbildung 28 und 29 auch noch in Diagrammform dargestellt. Abbildung 28 zeigt einen Vergleich der bedingten Wahrscheinlichkeiten für den Prüfungserfolg bzw. Nicht-Erfolg für die 4 Vorbereitungsstrategien berechnet nach dem naiven Bayes-Verfahren (NB) mit der Berechnung auf direktem Wege für die

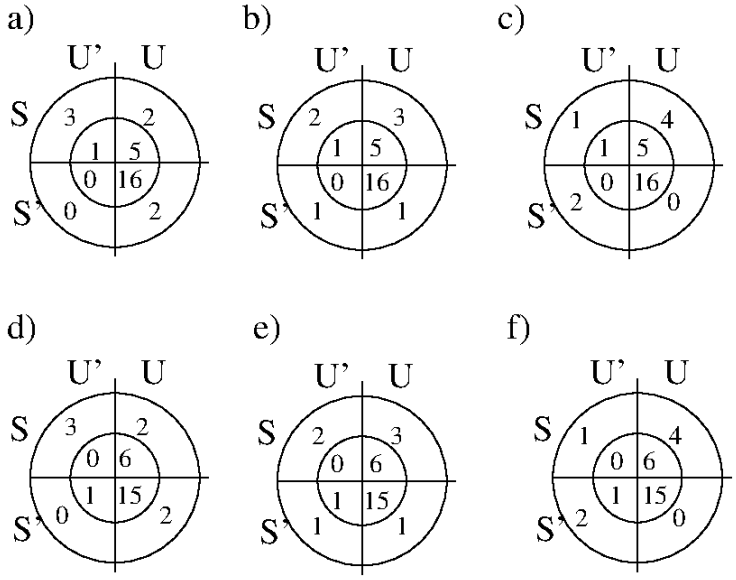

Abb. 27: Lösungen für die Verteilung der Schüler auf die Kombinationen von Prüfungserfolg und Vorbereitungsstil (innerer Kreis: bestanden, äußerer Ring: nicht bestanden)

Tabelle 12: Vergleich der bedingten Wahrscheinlichkeiten für das Bestehen der Mathematik-Reifeprüfung mit dem naiven Bayes-Verfahren und den direkt berechneten Wahrscheinlichkeiten in Abhängigkeit vom Vorbereitungsstil der 6 möglichen Schüleraufteilungen, die mit den Vorinformationen verträglich sind

	Naiver Bayes	a)	b)	c)	d)	e)	f)
$P(B'\|K_1)$	0.887	0.75	0.666	0.5	1	1	1
$P(B'\|K_2)$	0.541	-	1	1	0	0.5	0.666
$P(B'\|K_3)$	0.333	0.286	0.375	0.444	0.25	0.333	0.4
$P(B'\|K_4)$	0.070	0.111	0.059	0	0.118	0.067	0
$P(B\|K_1)$	0.113	0.25	0.333	0.5	0	0	0
$P(B\|K_2)$	0.459	-	0	0	1	0.5	0.333
$P(B\|K_3)$	0.666	0.714	0.625	0.555	0.75	0.666	0.6
$P(B\|K_5)$	0.930	0.889	0.941	1	0.882	0.936	1

oben ermittelten 6 Aufteilungsmöglichkeiten der Schüler (Fälle a) bis f) sowie Mittelwert MW). Man erkennt aus Abbildung 28, dass die direkt berechneten Wahrscheinlichkeitsverteilungen prinzipiell den gleichen Verlauf zeigen wie das naive Bayes-Verfahren, jedoch gibt es teilweise

Tabelle 13: Überprüfung des Kriteriums der paarweisen Unabhängigkeit der in den Merkmalskombinationen K_i auftretenden Merkmale

	Naiver Bayes	a)	b)	c)	d)	e)	f)
$P(K_1\|B')$	0.306	0.428	0.286	0.143	0.428	0.285	0.143
$P(K_2\|B')$	0.122	0	0.143	0.286	0	0.143	0.286
$P(K_3\|B')$	0.408	0.286	0.428	0.571	0.286	0.428	0.571
$P(K_4\|B')$	0.163	0.286	0.143	0	0.286	0.143	0
$P(K_1\|B)$	0.012	0.045	0.045	0.045	0	0	0
$P(K_2\|B)$	0.033	0	0	0	0.045	0.045	0.045
$P(K_3\|B)$	0.260	0.227	0.227	0.227	0.273	0.273	0.273
$P(K_4\|B)$	0.694	0.727	0.727	0.727	0.682	0.682	0.682

beträchtliche quantitative Abweichungen. Besonders gut passen die Fälle b), e) und f) sowie die mit MW bezeichnete ebenfalls eingezeichnete Mittelwertkurve. Letztere wurde mit aus Tabelle 11 über die Fälle a) bis f) gemittelten Schülerzahlen berechnet. Das korrespondiert mit den in Abbildung 29 gezeigten bedingten Wahrscheinlichkeiten $P(K_i|B')$ bzw. $P(K_i|B')$, die für den NB-Fall mit der Unabhängigkeitsannahme und für die Fälle a) bis f) direkt aus den Schülerzahlen berechnet wurden. Auch hier liegen die Ergebnisse der Fälle a), e) und f) am nächsten bei der NB-Kurve, was bedeutet, dass für diese drei Fälle die für das NB-Verfahren geforderte Unabhängigkeit am ehesten gegeben ist.

Trotz der hier beispielhaft aufgezeigten Unsicherheiten betreffend Unabhängigkeit der betrachteten Merkmale kann der naive Bayes-Klassifikator aber erfolgreich für die Auswahl der besten Möglichkeit eingesetzt werden, weil hierfür bereits der richtige Trend bzw. die Auswahl der Maximumstelle ausreicht, ohne dass der quantitative Maximalwert richtig bestimmt sein muss.

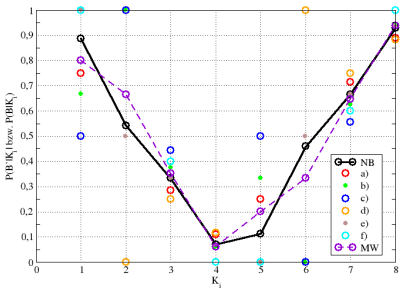

Abb. 28: Wahrscheinlichkeitsverteilung für den Prüfungserfolg abhängig von den Vorbereitungsstilen nach naivem Bayesverfahren (NB) und mit direkter Ausrechnung a) bis f) sowie den Mittelwerten aus a) bis f) (MW). Die ersten 4 Datenpunkte stehen für nicht bestanden bei K_i, die folgenden 4 Datenpunkte für bestanden bei K_i.

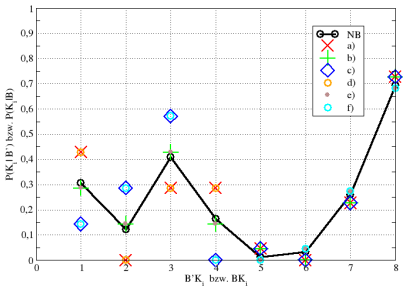

Abb. 29: Wahrscheinlichkeitsverteilung für die Priori-Wahrscheinlichkeiten berechnet mit der Unabhängigkeitsannahme bei NB sowie direkt berechnet in den Fällen a) bis f). Die ersten 4 Datenpunkte stehen für nicht bestanden bei K_i, die folgenden 4 Datenpunkte für bestanden bei K_i.

3.7 K-Means-Clustering

Als Beispiel für das unüberwachte Lernen wird nun noch die Lösung einer Clustering-Aufgabe mit dem Verfahren des K-Means-Clustering beschrieben. Beim Clustering müssen für eine nicht gekennzeichnete Datenmenge Gruppen mit ähnlichen Eigenschaften erkannt und gekennzeichnet werden. Das ist eine schwierigere Aufgabe als die Klassifizierung, da das Lernverfahren auch noch die charakteristischen Unterschiede zwischen den Datenpunkten selbst herausfinden muss. Um das zu etwas erleichtern, werden bei den verschiedenen Clustering-Verfahren noch zusätzliche Parameter angegeben, wie zum Beispiel die Anzahl der Cluster, ein minimaler Abstand zwischen Datenpunkten oder die minimale Anzahl von Mitgliedern in einem Cluster. Beim K-Means-Verfahren muss die Anzahl k der Cluster als Eingabeparameter festgelegt werden.

Das Verfahren läuft in den folgenden Schritten ab (s. [29]):

1. Wähle Anzahl k der zu findenen Cluster aus
2. Wähle k zusätzliche Datenpunkte als Clustermittelpunkte, die sogenannten Zentroiden
3. Berechne den Abstand aller Beispiele zu den Zentroiden mit Hilfe einer passenden Abstandsformel
4. Ordne den Zentroiden alle Punkte zu, die von ihnen den kleinsten Abstand im Vergleich zu den Abständen zu den anderen Zentroiden haben
5. Berechne für jeden Zentroiden den Schwerpunkt der ihm zugeordneten Datenpunkte und ersetze den Zentroiden durch diesen verbesserten Mittelpunkt seiner zugeordneten Daten
6. Berechne dann wieder die Abstände aller Datenpunkte zu den neuen Zentroiden, finde die Zuordnungen zum nun nächstgelegenen Zentroiden und bilde neue Schwerpunkte der neu zugeordneten Daten, die wieder zu verbesserten Zentroiden gemacht werden.
7. Wiederhole diese Prozedur solange, bis sich nichts mehr an der Punkteeinteilung ändert

Dieses Verfahren liefert immer eine bestimmte Unterteilung der Datenmenge in k Gebiete. Es ist aber etwas von den Startwerten abhängig, so dass bei veränderten Startwerten nicht immer exakt die gleichen Cluster gefunden werden. Weiter liefert das Verfahren immer Cluster, in denen die Punkte um ein Zentrum herum angeordnet sind. Bei komplizierteren Anordnungsmustern funktioniert das nicht mehr so gut. Dafür eigenen sich dann andere, sogenannte dichtebasierte Verfahren besser. Diese Verfahren sammeln Punkte mit ähnlichen Eigenschaften, die sie durch die gesamte Datenmenge hindurch suchen und dann dem Cluster hinzufügen. (s. [14]).

Für das k-Means-Verfahren wurde ebenfalls ein kleines Python-Programm geschrieben und mit der Aufgabe getestet, drei Gruppen mit ähnlichen Eigenschaften, worunter hier enge Nachbarschaft zu verstehen ist, zu finden und farblich zu kennzeichnen. Abbildung 30 zeigt die Ausgangsdaten, die um die Punkte $(0.8|0.1), (-0.3|0.6)$ und $(-0.4|-0.4)$ herum mit zufälligen Schwankungen gruppiert sind. Die Abbildung 31 zeigt das Ergebnis des K-Means-Verfahrens für die Sortierung sowie auch die Wegkurven der Zentroiden. Die Sortierung gelingt gut und die Zentroidenpunkte bewegen sich im Verlauf der Cluster-Einteilung immer genauer in Richtung der passenden Cluster-Schwerpunkte.

4 Zusammenfassung

Es wurden für einige wichtige Klassifzierungsverfahren die Funktionsweise und, soweit das auf der Basis von Schulkenntnissen möglich ist, auch die zugrunde liegende Mathematik dargestellt.

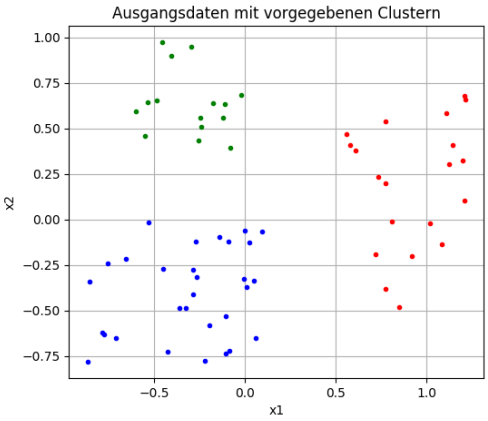

Abb. 30: Ausgangsdaten für K-Means-Clustering mit 3 farblich gekennzeichneten Punktegruppen.

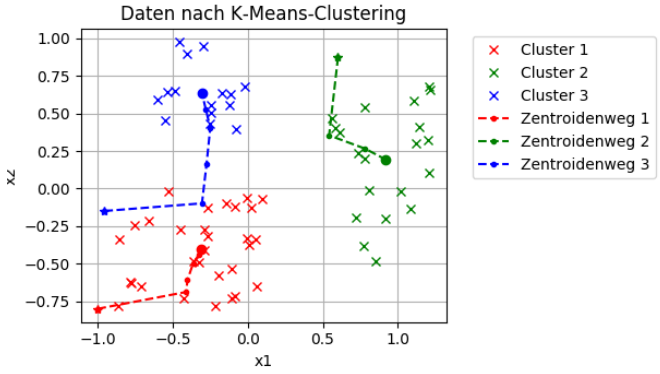

Abb. 31: Klassifizierungsergebnisse K-Means-Verfahren und Wegkurven der Zentroiden während der Einteilungsprozedur (Sternmarkierungen bezeichnen die Startpunkte).

Aus dem Bereich der überwachten Lernverfahren wurden das K-Nearest-Neighbour-Verfahren, die Methode der künstlichen neuronalen Netze, die Methode der logistischen Regression, das Support-Vector-Machine-Verfahren und das Entscheidungsbaumlernen sowie das naive Bayes-Klassifikationsverfahren behandelt. Weil künstliche neuronale Netze für die fortschrittlichen KI-Methoden wie Deep Learning und große Sprachmodelle wie Chat-GPT die Basis sind, wurde dieses Thema ausführlicher behandelt. Dafür wurde die Grundstruktur eines einfachen dreischichtigen neuronalen Netzes mit 2 Eingängen und einem Ausgang sowie 2 Neuronen in der verdeckten Schicht und einem Neuron in der Ausgangsschicht beschrieben und es wurden die Gleichungen für die Signaldurchleitung durch das Netz aufgestellt. Danach wurden die Netzgewichte mit dem Verfahren der Fehler-Backpropagation korrigiert, das hier nach jedem Trainingsbeispiel nach der Methode des stochastischen Gradientenabstiegs verwendet wurde. Aus dem Bereich des unüberwachten Lernens wurde die Methode des K-Means-Clustering als Beispiel beschrieben.

Für alle Methoden wurden kleine Python-Programme erstellt und auf einfache Beispiele zur Klassifizierung von Punkten in der Ebene oder einfache Entscheidungsprobleme erfolgreich angewendet. Durch Beschränkung auf die grundlegenden Merkmale der betrachteten Lernverfahren und auf sehr einfache Anwendungsbeispiele, hat sich gezeigt, dass es schon mit Schulmathematik-Kenntnissen möglich ist zu verstehen, wie wichtige KI-Verfahren im Prinzip funktionieren. Für die sechs Verfahren k-Nearest-Neighbours, neuronale Netze, logistische Regression, Entscheidungsbaumlernen und naive Bayes-Klassifikation sowie das k-Means-Clustering wurden einfache Umsetzungen der Modelle in Python direkt auf der Basis der Modellgleichungen erstellt. Für die Verfahren der Support Vector Machine, wo das nicht möglich war, und auch für das Entscheidungsbaum-Lernen wurden einfache Beispiele mittels der im Scikit-Learn-Modul dafür bereitgestellten Hilfsprogramme in Python getestet. Damit kann durch eigene Variationen der Modellparameter die Anwendung der Verfahren geübt und ein gewisses Gefühl für sensible Einflußgrößen wie Datensatzgröße und Modellparameter gewonnen werden, was zu einem besseren Verständnis von Stärken, Schwächen und Grenzen der KI-Methoden beitragen kann.

Literatur

[1] Jörg Fischer, Maschinelles Lernen für Dummies, 1. Auflage 2024, Wiley-VCH-GmbH, Weinheim, 2024, S. 119

[2] Andriy Burkov, Machine Learning kompakt, mitp-Verlags GmbH, Frechen, 2019

[3] Philipp Grunert, Machine Learning und Neuronale Netze, BMU-Verlag, 2021

[4] Jörg Fischer, Maschinelles Lernen für Dummies, a.a.O.

[5] Ralf Otte, Künstliche Intelligenz für Dummies, 2. Auflage, Wiley-VCH, Weinheim, 2023

[6] Jürgen Cleve, Uwe Lämmel, Data Mining, De Gruyter-Velag, Berlin, 2020

[7] Jörg Frochte, Maschinelles Lernen - Grundlagen und Algorithmen in Python, 3. Auflage, Hanser-Verlag, München, 2021

[8] Wofgang Ertel, Grundkurs Künstliche Intelligenz, 4. Auflage, Springer Vieweg, Wiesbaden 2016, S. 267

[9] Philipp Grunert, Machine Learning und Neuronale Netze, a.a.O., S. 257

[10] Andriy Burkov, Machine Learning kompakt, a.a.O., S. 55

[11] Jürgen Cleve, Uwe Lämmel, Data Mining, a.a.O., S 129

[12] https://studyflix.de/informatik/was-ist-kuenstliche-intelligenz-6919?topic_id=590 [abgerufen am 1.8.2024]

[13] Wofgang Ertel, Grundkurs Künstliche Intelligenz, a.a.O., S. 26

[14] Jürgen Cleve, Uwe Lämmel, a.a.O., S 87 ff

[15] Jörg Fischer, Maschinelles Lernen für Dummies, a.a.O., S.160

[16] Andriy Burkov, Machine Learning kompakt, a.a.O., S. 41

[17] Jörg Frochte, Maschinelles Lernen - Grundlagen und Algorithmen in Python, a.a.O., S.184

[18] Jörg Fischer, Maschinelles Lernen für Dummies, a.a.O., S. 182

[19] https://merehead.com/de/blog/vor-nachteile-neuronalen-netzwerkarchitektur/ [abgerufen am 30.8.2024]

[20] Andriy Burkov, Machine Learning kompakt, a.a.O., S. 44 ff

[21] Andriy Burkov, Machine Learning kompakt, a.a.O., S. 19 ff

[22] Philipp Grunert, Machine Learning und Neuronale Netze, a.a.O., S. 240

[23] Jörg Fischer, Maschinelles Lernen für Dummies, a.a.O., S. 132

[24] John Shawe-Taylor and Nello Christianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004, p. 213

[25] https://en.wikipedia.org/wiki/Decision_tree_learning [abgerufen am 30.8.2024]

[26] Jörg Fischer, Maschinelles Lernen für Dummies, a.a.O., S. 134 ff

[27] Philipp Grunert, Machine Learning und Neuronale Netze, a.a.O., S. 224

[28] Jörg Fischer, Maschinelles Lernen für Dummies, a.a.O., S. 140

[29] Andriy Burkov, Machine Learning kompakt, a.a.O., S. 154 ff

[30] Jürgen Cleve, Uwe Lämmel, a.a.O., S 150

[31] Jürgen Cleve, Uwe Lämmel, a.a.O., S 96 ff

[32] Kurt Stange, Bayes-Verfahren, Springer Verlag, 1977, S. 14

[33] Kurt Stange, Bayes-Verfahren, a.a.O., S.6

[34] Thomas A. Runkler, Data Mining, 2. Auflage, Springer-Vieweg, 2015, S.94

[35] Wofgang Ertel, Grundkurs Künstliche Intelligenz, a.a.O., S. 145

[36] Bernd Hofmann, Mathematik IV für Informatiker, TU Chemnitz, 2018, https://www.tu-chemnitz.de/mathematik/inverse_probleme/hofmann/teaching/Vorlesungsskripten/Stochastik_Baustein_1.pdf [abgerufen am 30.8. 2024]

[37] A. Kolmogoroff, Grundbegriffe der Wahrscheinlichkeitsrechnung, Springer Verlag 1933, S.7

[38] A. Kolmogoroff, Grundbegriffe der Wahrscheinlichkeitsrechnung, a.a.O., S.2

[39] Felix Hausdorff, Beiträge zur Wahrscheinlichkeitsrechnung, Berichte der mathematisch-physischen Classe der Königl. Sächs. Gesellschaft der Wissenschaften zu Leipzig, Sitzung vom 6. Mai 1901, S. 152 ff