Das stromlose Perzeptron

Stift-und-Papier-Einführung eines einfachen Algorithmus des maschinellen Lernens

Thomas Kraska

Zur Einführung eines Lernalgorithmus in der Sekundarstufe I wird ein haptischer Algorithmus für eine lineare Klassifizierung vorgestellt, der in wenigen Schritten zu einem Programmcode führt. Hierbei wird ausgenutzt, dass die Lernparameter die Elemente des Normalenvektors der Trenngerade zwischen zwei Arten von Objekten darstellen. So kann ausgehend von den Parametern mit einem Geodreieck die Trenngerade konstruiert werden und der Fehler der Klassifizierung abgelesen werden. Die Methode wurde in einem MINT Kurs der Klasse 10 durchgeführt.

1 Einführung

Das maschinelle Lernen als aktuell bedeutender Teil der künstlichen Intelligenz hat in den letzten Jahren durch eine Reihe von Alltagsanwendungen große Aufmerksamkeit insbesondere auch bei Schüler/inne/n erhalten. Auch schon vor der Einführung von ChatGPT haben Schüler/innen im Internet verfügbare Übersetzungstools oder Rechtschreibekorrekturen verwendet. Aber auch individuelle Vorschläge zum Beispiel auf Streamingplattformen basieren auf dem maschinellen Lernen. Bei der Nutzung bleibt den Schüler/inne/n die grundlegende Funktionsweise jedoch verborgen. Dessen Verständnis ist jedoch Voraussetzung für die Einschätzung der Fähigkeiten des maschinellen Lernens. Ist ein solches Programm tatsächlich intelligent im menschlichen Sinne? Das Prinzip des maschinellen Lernens kann man mit grundlegenden historischen Modellen einführen. Dabei lassen sich Fragen klären, wie das Lernen abläuft, an welcher Stelle das Gelernte abgespeichert wird und wovon die Qualität des Gelernten abhängt. Für ein solch einfaches Modell wird hier ein haptischer Algorithmus zur Klassifizierung von zwei Arten von Objekten vorgeschlagen, der von den Schüler/inne/n in einen Code umgesetzt wird. In Abbildung 1 ist ein schematisches Beispiel für ein solches System dargestellt. Die Objekte haben zwei Eigenschaften x und y. Beim Lernvorgang wird eine Trenngerade gesucht, die die zwei Arten von Objekten in dem Koordinatensystem der Eigenschaften voneinander trennt. Hier beschränken wir uns zunächst auf Trenngeraden, die durch den Ursprung verlaufen. Der Lernvorgang optimiert in diesem Fall die Steigung der Trenngeraden.

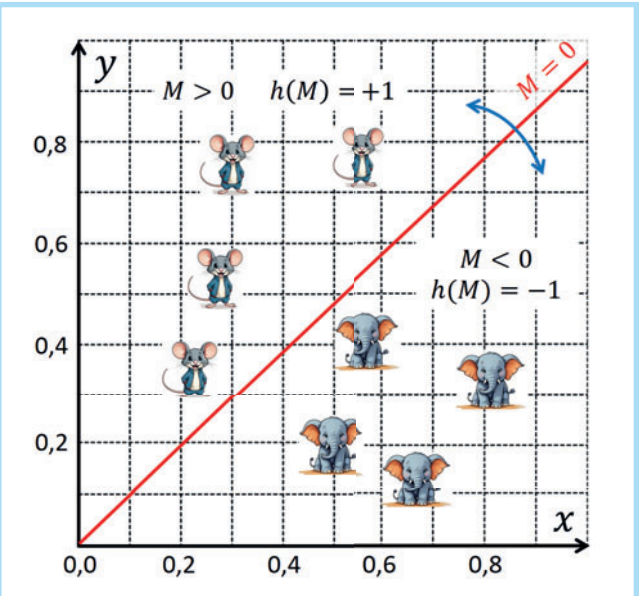


Abb. 1. Eine einfache Klassifizierung von zwei Arten von Objekten anhand von zwei Prädiktormerkmalen x und y, d.h. Objekteigenschaften.

Es existiert eine Reihe von Arbeiten, die die Vermittlung des maschinellen Lernens in der Schule behandeln. Mühling und Grosse-Bölting (2023) zeigen, dass die Schülervorstellung von maschinellem Lernen durch unplugged Aktivitäten verbessert werden kann. Sie folgern, dass die Schüler/inne/n keine signifikanten Fehlvorstellungen zu der Frage haben, ob ein Computer ein intelligentes Wesen sei und folgern daraus, dass man die Black Box nicht zwingend öffnen müsse. Während dies hinsichtlich solcher Fehlvorstellungen und auch der ethischen und sozialen Implikationen ausreichen mag, bleibt unklar wie ein Computer tatsächlich lernt. Dies wird erst deutlich, wenn neben einem anwendungsorientierten Ansatz den Schüler/inne/n die Möglichkeit gegeben wird, selbstständig einen entsprechenden Algorithmus zu erarbeiten. Dementsprechend plädieren andere Autoren (Strecker & Modrow 2019) für die Entmystifizierung der Künstlichen Intelligenz (KI) durch Programmierung mit Scratch. In Ausgabe 02/2024 in diesem Journal finden sich eine Reihe von Arbeiten zu anwendungsorientierten Aspekten des Einsatzes von KI und eine mit der Vermittlung des grundlegenden Verständnisses der KI (Modrow 2024).

Die Auswirkungen von maschinellem Lernen auf das klassische informatische Denken (computational thinking) wird von Shamir und Levin (2022) diskutiert. Die Autoren betonen, dass maschinelles Lernen nicht den üblichen Strategien zur Problemlösung folgt, sondern dass die Algorithmen von Daten lernen. Dies führt dazu, dass man den Datenflüssen beim maschinellen und tiefen Lernen nicht transparent folgen kann so wie es bei regelbasierten Algorithmen der Fall ist. Regelbasierte Algorithmen

setzen eine Fragestellung oder ein Modell in einen deterministischen Code um, d.h. die Datenflüsse sind in dem Code nachvollziehbar. Die Autoren folgern, dass das Konzept des informatischen Denkens um die Behandlung und die Bewertung von Daten erweitert werden muss.

Ebenso plädiert Micheuz (2020) dafür, dass KI und in diesem Zusammenhang auch Data Science Bestandteil der modernen Schulbildung sein sollten, da KI häufig von irreführenden Aussagen begleitet wird, die von „utopischer Begeisterung bis hin zu dystopischen Ängsten reichen". Um dem entgegenzuwirken, fordert Micheuz eine fundierte Erklärung wie KI funktioniert, was einen programmiersprachlichen Hintergrund einschließt.

Zu einem ähnlichen Ergebnis kommen Yim und Su (2024) in einem Review über 46 Studien zu der Frage wie KI in der Sekundarstufe unterrichtet werden sollte. Sie folgern, dass neben der Beschäftigung mit Anwendungsprogrammen das Programmieren mit Scratch oder Python eingesetzt werden kann, um KI Prinzipien zu entmystifizieren.

Strecker (2020) hat bereits einen Ansatz zur Umsetzung von kleinen neuronalen Netzwerken in der Schule vorgeschlagen, der auf die Programmierung eingeht. Eines der Anliegen von Strecker ist es, keine Black Box zu verwenden, sondern Schüler/innen selber programmieren zu lassen. Dabei geht es unter anderem darum zu zeigen, dass es sich um Algorithmen handelt, die kein Bewusstsein entwickeln.

Die Arbeit hier beschränkt sich auf einen Algorithmus für ein einzelnes Neuron. Es wird eine Methode vorgestellt, mit der Schüler/innen spielerisch im Rahmen des informatischen Denkens einen Lernalgorithmus erarbeiten können und für diesen einen kompakten Programmcode erstellen können. Damit wird auf sehr grundlegender Ebene deutlich, wie maschinelles Lernen funktioniert, wo die Informationen gespeichert werden und inwiefern dies mit Intelligenz im menschlichen Sinne zu tun hat. Ein Ziel ist es, auch Schüler/innen in anderen Fächern für deren fachliche Kontexte einen Lernalgorithmus entwickeln und anwenden zu lassen und so einen fundamentalen Eindruck vom maschinellen Lernen im Rahmen dieser Anwendungen zu vermitteln. Aus diesem Grunde wird hier auch auf die Einbettung des Kerns des Algorithmus in übergeordnete Strukturen wie die Objektorientierung verzichtet, da diese für Anwendungsfächer einen vermeidbaren Overhead darstellt.

2 Das Perzeptron

Das einfachste System für maschinelles Lernen ist das Perzeptron (Rosenblatt 1958), das ein einzelnes Neuron darstellt. In Abbildung 2 ist die Datenverarbeitung für ein Perzeptron dargestellt. Die Vorwärtsrechnung dient der Bestimmung der Art eines Objektes. Dazu benötigt man die Eigenschaften der Objekte x und y sowie Parameter a und b, die die Gewichtung dieser Eigenschaften für die Vorhersage der Art des Objektes ermöglichen. Zusätzlich wird noch ein sogenannter Bias oder Schwellwert benötigt, dessen Eingabewert immer 1 ist, der aber ebenfalls ein Gewicht c besitzt. Die Variablen der Eigenschaften werden dann mit ihren Gewichten multipliziert und zusammen mit dem Bias aufaddiert. Dies wird hier in Analogie zu biologischen Neuronen als Membranpotential M bezeichnet.

$$M(x_i, y_i) = a \cdot x_i + b \cdot y_i + c \cdot 1$$

Das Ergebnis für M ist eine kontinuierliche Zahl, die beim Perzeptron mit einer Aktivierungsfunktion $h(M)$ in ein diskretes Ergebnis umgewandelt wird:

$$r = h(M) = \begin{cases} -1, & M<0 \\ +1, & M\geq 0 \end{cases}$$

Für $r = -1$ liegt die eine Art von Objekt (Elefant in Abbildung 1) vor, für $r = +1$ die andere Art (Maus in Abbildung 1). Für die Trennlinie gilt folglich $M = 0$. Die Parameter müssen nun so gewählt werden, dass man bei bekannten Eigenschaften x, y die Art des Objektes korrekt vorhersagen kann. Hierzu ist es erforderlich, dass man einen Datensatz zum Training des Systems zur Verfügung hat. Man spricht hierbei von supervised learning, bei dem das System anhand von bekannten Klassifikationen lernt. Ein Lernalgorithmus basiert auf einer Loss-Funktion, die die Abweichung zwischen den berechneten und den tatsächlichen Klassifizierungen des Trainingsdatensatzes wiedergibt. Eine häufig verwendete Loss-Funktion ist die quadratische Abweichung, die sich jedoch strenggenommen nur auf normalverteilte Daten anwenden lässt. Sofern die Aktivierungsfunktion eine Steigung besitzt und differenzierbar ist, kann man die Korrekturen der Parameter über das Gradientenverfahren erhalten. Da diese Bedingungen beim Perzeptron nicht erfüllt sind, führt man direkt die sogenannten Delta-Regeln zur Korrektur der Lernparameter ein:

$$a \leftarrow a - L\, e_i\, x_i \qquad b \leftarrow b - L\, e_i\, y_i \qquad c \leftarrow c - L\, e_i$$

Hierbei ist e_i der Fehler eines Datenpunktes $e_i = r_i - s_i$ mit dem berechneten $r_i = h(M(x_i, y_i)) = \pm 1$ und dem tatsächlichen Wert $s_i = \pm 1$ für die Art des Objekts. Zusätzlich wird eine Lernrate L eingeführt. Der Wert für L liegt zwischen null und eins und soll verhindern, dass die Parameter zu große Sprünge machen und ggfs. über eine Lösung hinwegspringen.

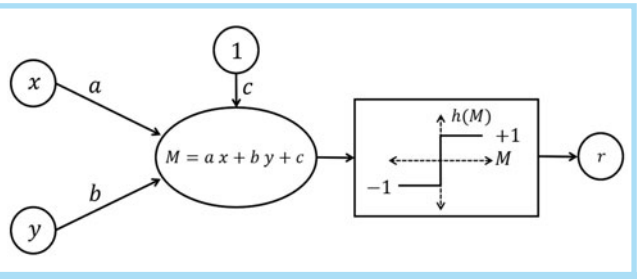


Abb. 2. Schematische Darstellung eines Perzeptrons

3 Methode

Die detaillierte Umsetzung im Unterricht ist in den Online-Ergänzungen erläutert. Die Schüler/innen erstellen zunächst

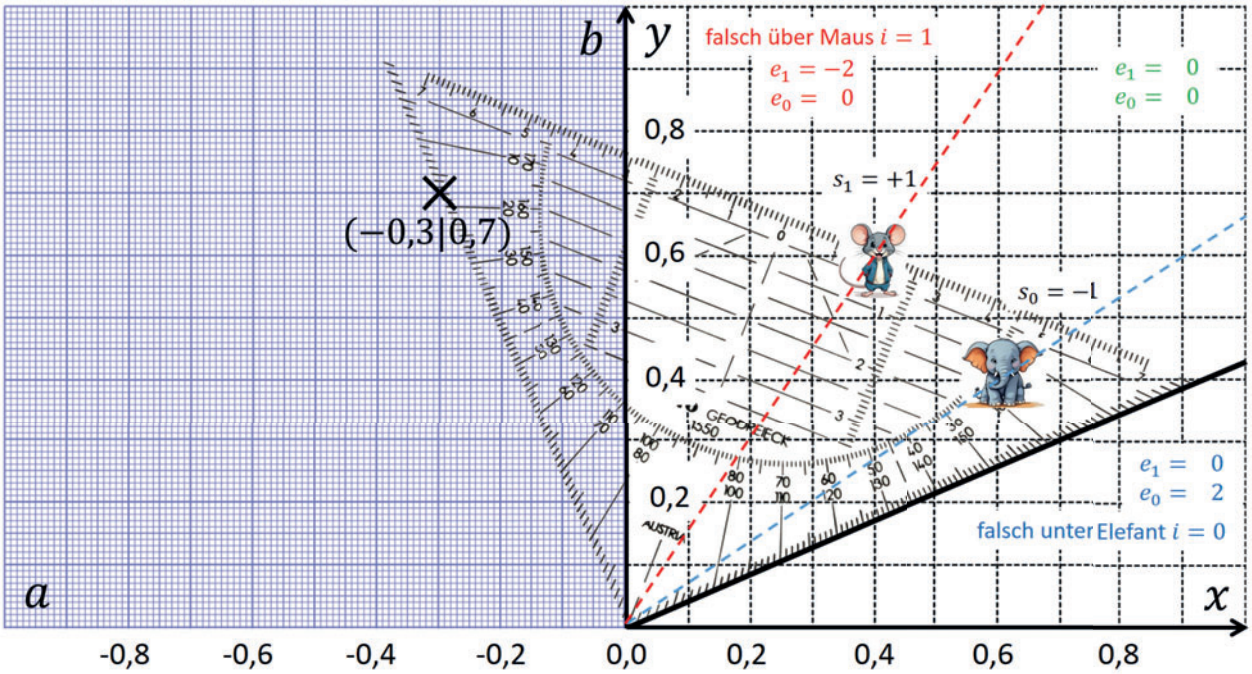


Abb. 3. Ausschnitt aus dem Arbeitsblatt mit der Trenngeraden (fett) rechts und den Lernparametern links (Kreuz), hier bei $a = -0{,}3$ und $b = 0{,}7$. Das Geodreick stellt den Zusammenhang her. Wenn die Trenngerade in dem Bereich oberhalb der roten Gerade liegt, dann gelten die in rot angegebenen Fehler. Liegt die Trenngerade unterhalb der blauen Gerade, dann gelten die in blau angegebenen Fehler.

Lernprotokoll 1

				Parameter $a - e \to a$ $b - e \to b$ $m = -\frac{a}{b}$
Epochen und Datenpunkte			**Prüfung**, ob die Trenngerade korrekt liegt	Startwerte $\boldsymbol{a = 0 \quad b = 1}$ $m = 0$
0		$i = 0$ $s_0 = -1$ $x_0 = 0{,}6$ $y_0 = 0{,}4$	$e_0 = 2$ Gerade falsch unter Elefant	$0 - (2) \to \boldsymbol{a = -2}$ $1 - (2) \to \boldsymbol{b = -1}$ $m = -2$
		$i = 1$ $s_1 = +1$ $x_1 = 0{,}4$ $y_1 = 0{,}6$	$e_1 = -2$ Gerade falsch über Maus	$-2 - (-2) \to a = 0$ $-1 - (-2) \to b = 1$ $m = 0$

Abb. 4. Ausgefülltes Lernprotokoll für einen Lernalgorithmus zunächst ohne Gewichtung mit den Eingangsdaten x_i, y_i. In den Online-Ergänzungen steht ein DIN A3 Diagramm und ein unausgefülltes Lernprotokoll zur Verfügung. In diesem Beispiel erhält man nach einem Durchlauf die Startwerte zurück. Eine Epoche ist das einmalige Durchlaufen eines Datensatzes, hier also einmal Elefant und einmal Maus.

Lernprotokoll 2

Epochen und Datenpunkte			**Prüfung**, ob die Trenngerade korrekt liegt	**Parameter** $a = 0 \quad b = 1$ $m = 0$ $a - e \cdot x_i \to a$ $b - e \cdot y_i \to b$ $m = -a/b$
0		$i = 0$ $s_0 = -1$ $x_0 = 0{,}6$ $y_0 = 0{,}4$	$e_0 = 2$ Gerade falsch unter Elefant	$0 - (2) \cdot 0{,}6 \to \boldsymbol{a = -1{,}2}$ $1 - (2) \cdot 0{,}4 \to \boldsymbol{b = 0{,}2}$ $m = 6$
		$i = 1$ $s_1 = +1$ $x_1 = 0{,}4$ $y_1 = 0{,}6$	$e_1 = -2$ Gerade falsch über Maus	$-1{,}2 - (-2) \cdot 0{,}4 \to \boldsymbol{a = -0{,}4}$ $0{,}2 - (-2) \cdot 0{,}6 \to \boldsymbol{b = 1{,}4}$ $m = 0{,}286$
1		$i = 0$ $s_0 = -1$ $x_0 = 0{,}6$ $y_0 = 0{,}4$	$e_0 = 2$ Gerade falsch unter Elefant	$-0{,}4 - 2 \cdot 0{,}6 \to \boldsymbol{a = -1{,}6}$ $1{,}4 - 2 \cdot 0{,}4 \to \boldsymbol{b = 0{,}6}$ $m = 2{,}667$
		$i = 1$ $s_1 = +1$ $x_1 = 0{,}4$ $y_1 = 0{,}6$	$e_1 = -2$ Gerade falsch über Maus	$-1{,}6 - (-2) \cdot 0{,}4 \to \boldsymbol{a = -0{,}8}$ $0{,}6 - (-2) \cdot 0{,}6 \to \boldsymbol{b = 1{,}8}$ $m = 0{,}444$
	...	...	...	...

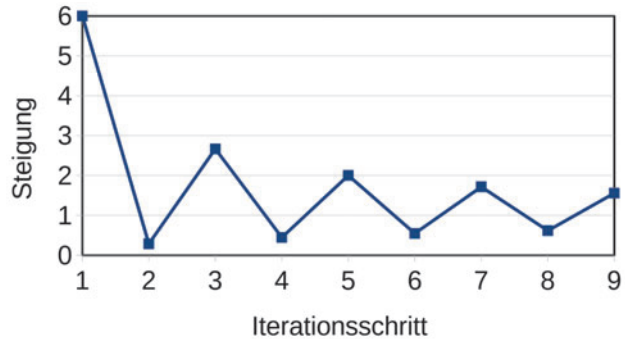
Abb. 5. Ausgefülltes Lernprotokoll mit Gewichtung mit den Eingangsdaten für die ersten beiden Epochen. In den Online-Ergänzungen steht eine vollständig ausgefüllte sowie eine unausgefüllte Version zur Verfügung.

einen eigenen Datensatz von zwei Arten von Objekten mit zwei Eigenschaften, die über Zahlen angegeben werden können. Die Eigenschaften werden durch ihre Maximalwerte geteilt, um Zahlen für beide Eigenschaften zwischen 0 und 1 zu erhalten. Diese werden mit GeoGebra (GeoGebra classic 2024) aufgetragen, um ein Diagramm ähnlich zu Abbildung 1 zu erhalten. Der Lernalgorithmus wird dann mit einem haptischen Verfahren mit einem Blatt Papier, einem Stift und einem Geodreieck eingeführt. Das Perzeptron wird vereinfacht, indem der Bias weggelassen wird.

Dazu wird in Abbildung 2 das c bei der Berechnung von M und der obere Eingang gestrichen. Es können also nur Datensätze klassifiziert werden, deren Trenngerade durch den Ursprung verläuft. Man kann die Gleichung für das Membranpotential als Koordinatenform einer Geraden in einem zweidimensionalen Koordinatensystem auffassen, die sich als Skalarprodukt schreiben lässt:

$$M = a \cdot x + b \cdot y = \begin{pmatrix} a \\ b \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \vec{n} \cdot \vec{u}$$

Da für die Trenngerade für $M = 0$ gilt, ist $\vec{n} = \begin{pmatrix} a \\ b \end{pmatrix}$ ein Normalenvektor zu $\vec{u} = \begin{pmatrix} x \\ y \end{pmatrix}$. Da der Bias vernachlässigt wird, ist der Ortsvektor der Trenngeraden der Nullvektor und die Trenngerade liegt auf dem Richtungsvektor $\vec{u}$. Der Normalenvektor $\vec{n}$ steht senkrecht auf $\vec{u}$. Man setzt ihn im Ursprung an. Man kann die Trenngerade $M = 0$ aber auch in eine Geradengleichung mit der Steigung $m = -a/b$ umschreiben. Die Bedeutung der Steigung ist den Schüler/inne/n bekannt und sie können damit direkt auf den Verlauf der Trenngeraden schließen. Zur haptischen Durchführung des Lernalgorithmus benötigt man ein zweigeteiltes Koordinatensystem wie in Abbildung 3 im Ausschnitt dargestellt. In dem rechten Teildiagramm werden die zwei Objekte mit ihren Koordinaten eingetragen. Im linken Teildiagramm sind die Werte der Lernparameter aufgetragen.



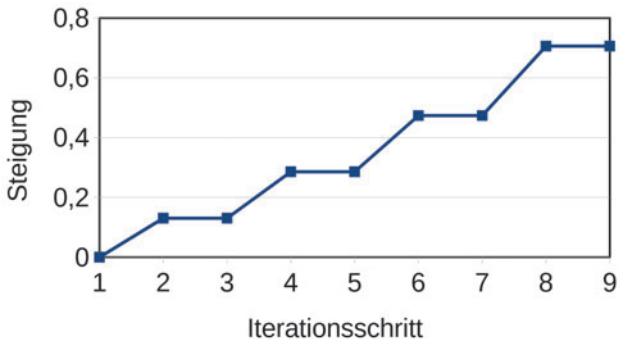


Abb. 6. Verlauf der Steigung der Trenngeraden entsprechend Lernprotokoll 2 mit a) $L = 1$ und b) $L = 0{,}1$

Der Zusammenhang zwischen den beiden Teildiagrammen ergibt sich durch die oben erwähnte Orthogonalität und kann praktisch mit einem Geodreieck hergestellt werden. Die rechtwinklige Ecke des Geodreiecks wird in den Ursprung gelegt. Legt man die linke Seite des Geodreiecks an dem Punkt für den Parametersatz an, dann kann man die zugehörige Trenngerade auf der anderen Seite des Geodreiecks einzeichnen und dort den Fehler direkt ablesen. Während diese Hintergründe im Mathematikunterricht in der Sekundarstufe II als Anwendungsbeispiel besprochen werden können, wurde bei einem MINT-Kurs der Klasse 10 darauf verzichtet.

4 Durchführung

Um die Delta-Regel empirisch einzuführen, zieht man zunächst lediglich die Fehler von den Parametern ab, ohne die Gewichtung durch die Eingangsdaten x_i und y_i und die Lernrate L zu thematisieren. Der Ablauf des haptischen Lernalgorithmus ist dann wie folgt:

A) Die Startwerte $a = 0$, $b = 1$ werden als Koordinatenpunkt in das linke Diagramm eingetragen. Die Spitze des rechten Winkels des Geodreiecks (Abb. 3) wird auf den Ursprung gelegt. Es folgt, dass die Trenngerade genau auf der x-Achse im positiven Bereich liegt.

B) Im rechten Diagramm wird die relative Lage zwischen Objekt (Elefant) und Trenngerade geprüft und der Fehler für den Bereich, in dem die Gerade liegt, abgelesen. Da sie falsch unter dem Elefanten liegt, ist der Fehler $e_0 = 2$.

C) In das Lernprotokoll 1 (Abb. 4) wird in Feld *Prüfung* „$e_0 = 2$“ sowie „Gerade falsch unter Elefant“ eingetragen.

D) In der rechten Spalte *Parameter* werden die neuen Werte für a und b sowie die Steigung $m = -a/b$ berechnet. Der Fehler e_i wird in die Klammern in der rechten Spalte übertragen und die neuen Werte für a und b aus ihren aktuellen Werten berechnet.

E) Die gleiche Prozedur wird mit den neuen Parametern a und b durchgeführt. Es folgt also Schritt A) jetzt allerdings für die Maus. Im Folgenden werden die Berechnung abwechselnd für die Objekte durchgeführt (Abb. 4).

Es kann die Frage aufkommen, warum man den Fehler e_i bei der Korrektur der Gewichte abzieht und nicht addiert. Wenn die Trenngerade (Abb. 3) falsch unter dem Elefanten liegt, dann muss man ihre Steigung $m = -a/b$ erhöhen, d.h. das negative a muss stärker negativ werden, und/oder das positive b kleiner werden. Wenn die Trenngerade unter dem Elefanten liegt, dann ist $e_0 = +2$ und eine Verkleinerung von $b\,(b > 0)$ ist nur möglich, wenn e_0 abgezogen wird.

$$b \leftarrow b - e_0$$

Da a negativ ist, führt das Abziehen von e_0 zu einem noch stärker negativen Wert.

$$a \leftarrow a - e_0$$

In beiden Fällen führt das Abziehen von e_0 zu einer Erhöhung der Steigung der Trenngerade, was erforderlich ist, um einer korrekten Klassifizierung näher zu kommen. Die analoge Betrachtung für eine Trenngerade, die oberhalb der Maus liegt, führt ebenfalls dazu, dass man den Fehler $e_1 = -2$ abziehen muss, um verbesserte Parameter zu erhalten.

Der erste Durchlauf in Abbildung 4 führt jedoch nicht zum Ziel, weil die Trenngerade zwischen zwei Positionen alterniert, die keine Lösung darstellen. Auf der Grundlage dieser Erkenntnis kann man die Gewichtung des Fehlers e_i mit den Eingangsdaten x_i und y_i erarbeiten. Wenn z.B. der Eingabewert x_i klein ist, dann muss auch der Korrekturterm entsprechend klein sein. Statt $a \leftarrow a - e_i$ benutzen wir also $a \leftarrow a - e_i \cdot x_i$ und $b \leftarrow b - e_i \cdot y_i$. Der Ablauf ist der gleiche wie oben beschrieben, es müssen nur noch die Fehler e_i mit x_i bzw. y_i multipliziert werden. Die Werte werden ins Lernprotokoll 2 (Abb. 5) eingetragen. Bei der dritten Durchführung des Lernalgorithmus kann man die Lernrate L hinzufügen. In Lernprotokoll 3 (Online-Ergänzungen) wird $L = 0{,}1$ verwendet. Die mehrfache Durchführung dieses Papier-Algorithmus dient einerseits der Wiederholung. Die erste Version ist zudem stark vereinfacht, so dass die Schüler/innen das Prinzip leichter erkennen können, als wenn sie mit der letzten Version beginnen würden.

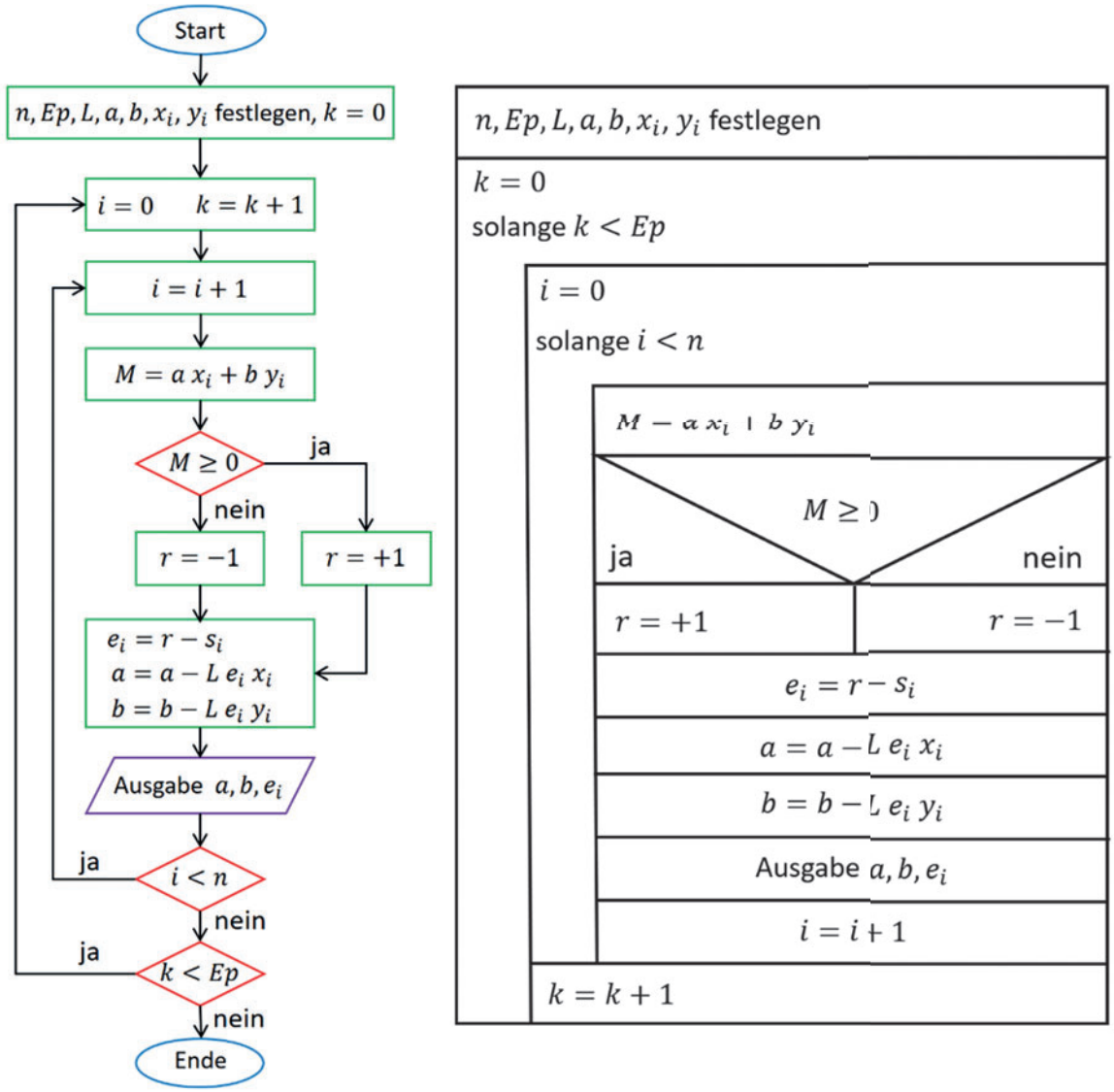


Abb. 7. a) Datenflussdiagramm und b) Nassi-Shneidermann-Diagramm des Lernalgorithmus. Ep: Anzahl der Epochen, n: Anzahl an Objekten im Trainingsdatensatz. Leere Diagramme stehen in den Online-Ergänzungen zur Verfügung.

Die Entwicklung der Steigung der Trenngarden ist in Abbildung 6 für $L = 1$ und $L = 0{,}1$ dargestellt. Für $L = 1$ sind die Sprünge in den Parametern so groß, dass die Gerade sich alternierend der Lösung annähert. Bei $L = 0{,}1$ nähert sich die Gerade von einer Seite in kleinen Schritten der Lösung an. In Abbildung 6b sieht man auch, dass jeder zweite Schritt zu keiner Veränderung führt, da die Maus immer auf der richtigen Seite der Gerade liegt.

Nach dem mehrfachen Durchlaufen des Algorithmus auf dem Papier können die Schüler/innen einen Pseudocode erstellen, also einen Ablaufplan in einer allgemeinverständlichen Sprache. Er kann strukturiert oder als Text formuliert werden. Als Alternative bzw. auch als zusätzlichen Schritt kann man Struktogramme verwenden. Während bei dem Datenflussdiagramm in Abbildung 7a die Schleifen über bedingte Anweisungen umgesetzt werden, ermöglicht das Nassi-Shneidermann-Diagramm in Abbildung 7b eine etwas übersichtlichere Darstellung der Schleifen. Der letzte Schritt ist die Umsetzung in einen konkreten Programmcode. Abbildung 8 zeigt ein Beispiel für den schulisch geeigneten TigerJython-Interpreter (Arnold et al. 2024). In den Online-Ergänzungen stehen dieses Programm und zusätzlich zwei Varianten mit grafischer Ausgabe zur Verfügung.

5 Der Adaline-Algorithmus

Das hier beschriebene Verfahren lässt sich auch auf den Adaline-Algorithmus (Widrow & Hoff 1959) anwenden. Adaline leitet sich von „adaptive linear neuron“ ab und verwendet eine lineare Aktivierungsfunktion $h(M) = M$, was praktisch bedeutet, dass keine Aktivierungsfunktion vorliegt. Der Unterschied zum Perzeptron besteht darin, dass kontinuierliche Zahlen ausgegeben werden und nicht zwei diskrete Werte, was sich auf den Lernvorgang auswirkt. Am Ende werden die Arten der Objekte jedoch wieder anhand des Vorzeichens der Ausgabe

```
1  L       = 1
2  Epochen = 20
3
4  # Datensatz
5  x = [ 0.6, 0.4]
6  y = [ 0.4, 0.6]
7  s = [  -1, 1  ]
8  ndat = len(x)
9
10 # Startwerte der Parameter
11 a =  0.0
12 b =  1.0
13
14 #Lernalgorithmus
15 for k in range(Epochen):
16     for i in range(ndat):
17         M = a * x[i] + b * y[i]
18         if M>0: r= 1
19         else:   r=-1
20         e = r - s[i]
21         a = a - L * e * x[i]
22         b = b - L * e * y[i]
23         Stg = -a/b
24         print i,k,round(a,4),round(b,4),Stg,M
```

Abb. 8. Programmcode für den Lernalgorithmus

bestimmt. Im Gegensatz zum Perzeptron endet die Iteration nicht, sobald alle Objekte korrekt klassifiziert sind, sondern geht weiter bis ein Minimum im mittleren quadratischen Abstand zwischen Ausgabe von M und den korrekten Werten s_i erreicht ist. Mit einer kleinen Änderung kann man den Adaline Algorithmus implementieren. Dazu löscht man in dem Code in Abbildung 8 die Zeilen 18 und 19 und fügt stattdessen die Zeile $r = M$ ein.

6 Beobachtungen

Bei dem Einstieg über die Erzeugung eigener Datensätze haben die Schüler/innen engagiert ihre eigenen Interessengebiete eingebracht. Dabei hat sich die Strukturierung ihrer Daten durch die Vorgabe einer Tabelle (Online-Ergänzungen) als hilfreich erwiesen. Dies bezieht sich insbesondere auf die Skalierung der Daten mittels Division durch den entsprechenden Maximalwert. Die Erstellung des Diagramms wie in Abbildung 1 auf der GeoGebra Web-Seite (GeoGebra classic 2024) gelang ohne Probleme.

In der Diskussion vorab sind einzelne Schüler/innen selbständig auf den Gedanken gekommen, dass man die Parameter mit dem Fehler korrigieren muss. Zur Erläuterung des Ablaufes des Algorithmus erwies sich die Bereitstellung von Abbildung 3 als erforderlich. Zusätzlich wurde ein Diagramm, in dem die drei Bereiche für die Fehler e_i in Abbildung 3 farblich unterlegt sind (Online-Ergänzungen) zur Verfügung gestellt. Damit war es den Schüler/innen möglich, ausgehend von den Parametern die Trenngerade einzuzeichnen, die Fehler abzulesen und die neuen Werte für die Parameter in dem Lernprotokoll zu berechnen.

Bei den Berechnungen in den Lernprotokollen unterliefen einigen Schüler/inne/n Flüchtigkeitsfehler bei der Subtraktion negativer Zahlen wie z.B. $-1{,}2-(-2)\cdot 0{,}4$.
Da sich bei einer Iteration solche Fehler fortpflanzen, ist es sinnvoll vorab auf solche Fehler hinzuweisen.

Nachdem die Schüler/innen im ersten Durchlauf das alternierende Verhalten der Parameter erkannt haben, haben sie bei der Diskussion zur Verbesserung vorgeschlagen, den Fehlerwert zu verringern z.B. durch Multiplikation mit einer kleinen Zahl. Damit haben sie den Lernparameter L vorweggenommen. Zunächst wurde jedoch die Multiplikation der Fehler mit den Eingangswerten berücksichtigt. Der Algorithmus war den Schüler/inne/n nach Ausfüllen von Protokoll 2 verständlich, so dass Lernprotokoll 3 nicht mehr erforderlich war. Schließlich haben sie den Algorithmus als Pseudocode verschriftlicht.
Da die Schüler/innen bereits Erfahrungen mit Struktogrammen hatten, war ihnen die Umsetzung des Pseudocodes in passende, aber unausgefüllte Diagramme möglich. Alternativ ist es auch möglich, solche Diagramme ausgefüllt zur Verfügung zu stellen und den Zusammenhang zu dem Stift-und-Papier-Algorithmus herstellen zu lassen. Im letzten Schritt folgte die Umsetzung in einen Code. Mit diesem konnte dann die händische Berechnung für Lernprotokoll 2 überprüft werden. Schließlich wurde noch die Lernregel für den Bias ergänzt, bei dem der Eingangswert immer 1 ist und folglich die Multiplikation mit dem Eingangswert praktisch entfällt. Für die Berechnung mit den eigenen Datensätzen wurde ein Programm mit graphischer Ausgabe zur Verfügung gestellt (Online-Ergänzungen). In dieses Programm haben die Schüler/innen ihre Daten in Listen eingegeben. Zusammenfassend kann man feststellen, dass die Durchführung des Papieralgorithmus für Schüler/innen in der Klasse 10 erfolgreich war. Die Schüler/innen konnten den Algorithmus gut nachvollziehen und erkennen, wie maschinelles Lernen für dieses Beispiel abläuft.

Danksagung

Der Autor dankt der Firma GEOTEC Schul- und Bürowaren GmbH *für die Bereitstellung* einer Bilddatei für das Geodreieck. Die Abbildungen für die Maus und den Elefanten wurden mit dem KI-Bildgenerator https://app.leonardo.ai/ erzeugt.

Literatur

Arnold, J., Kohn, T., Komm, D. & Roth, N. (2024). Programmierumgebung TigerJython, https://www.tigerjython.ch/de

GeoGebra classic (2024). Geometrie und Algebra Programm. https://www.geogebra.org/classic

Micheuz, P. (2020). Anmerkungen zur Künstlichen Intelligenz als Thema im Schulunterricht, Bildung und Digitalisierung, 1(4) 271–292.

Modrow, E. (2024). Datenkompression und Maschinelles Lernen mit SciSnap! *MNU-Journal*, 77, 111–118

Mühling, A. & Grosse-Bölting, G. (2023). Novices' conceptions of machine learning, *Computers and Education: Artificial Intelligence*, Volume 4, 100142

Rosenblatt, F. (1958). The Perceptron: a probabilistic model for information storage and organization in the brain. *Psych. Rev., 65*, 386–408.

Shamir, G. & Levin, I. (2022). Teaching machine learning in elementary school, *Int. J. Child-Comp. Interaction, 31*, 100415.

Strecker, K. (2020) Ein kleines Neuronales Netz selbst programmieren *MNU-Journal*, 73, 92–96

Strecker, K. & Modrow, E. (2019). Eine Unterrichtssequenz zum Einstieg in Konzepte des maschinellen Lernens, in Arno Pasternak (Hrsg.): Informatik für alle. *Lecture Notes in Informatics (LNI)*, Gesellschaft für Informatik, Bonn 2019, S. 335-344

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *IRE WESCON convention record, 4*, 96–104.

Yim, I.H.Y. & Su, J. (2024). Artificial intelligence (AI) learning tools in K-12 education: A scoping review. *J. Comput. Educ.* (https://doi.org/10.1007/s40692-023-00304-9)

Thomas Kraska, Institut für Physikalische Chemie, Naturwissenschaftliche Fakultät, Universität zu Köln, Greinstraße 4-6, 50939 Köln, https://van-der-waals.pc.uni-koeln.de/, t.kraska@uni-koeln.de, studierte Chemie und promovierte an der Ruhr-Universität Bochum. Nach Aufenthalten an der Cornell University und an der UC Berkeley habilitierte er sich 1999 an der Universität zu Köln. Seit 2010 unterrichtet er außerdem an einem Gymnasium die Fächer Chemie, Physik, Mathematik und MINT.