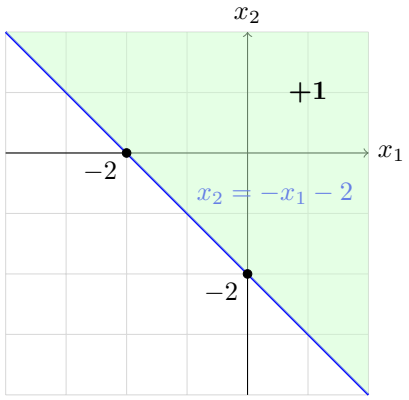
NN.Perzeptron.01: Entscheidungsgrenze

1. **(1P) Trennebene und Klassifikationsbereich**
   Die Entscheidungsgrenze ist definiert durch $w^T x = 0$:

   $$2 \cdot 1 + 1 \cdot x_1 + 1 \cdot x_2 = 0 \quad \Longleftrightarrow \quad x_2 = -x_1 - 2$$

   Das ist eine Gerade mit den Achsenabschnitten $(-2, 0)$ und $(0, -2)$. Da $w^T(0,0)^T = 2 > 0$ ist, liegt der Ursprung im positiven Bereich. Der Bereich $+1$ befindet sich somit **oberhalb** der Geraden.

2. **(1P) Vergleich der Gewichtsvektoren**
   Ein Vektor $\mathbf{w}'$ definiert dieselbe **Trennebene**, wenn er kollinear zu $\mathbf{w}$ ist ($\mathbf{w}' = k \cdot \mathbf{w}, k \neq 0$). Er bewirkt dieselbe **Klassifikation**, wenn zudem die Richtung gleich bleibt ($k > 0$).

   - **Gleiche Trennebene:**
     - $(1, 0.5, 0.5)^T \quad (k = 0.5)$
     - $(200, 100, 100)^T \quad (k = 100)$
     - $(-2, -1, -1)^T \quad (k = -1)$
   - **Exakt gleiche Klassifikation:**
     - $(1, 0.5, 0.5)^T$
     - $(200, 100, 100)^T$

   Der Vektor $(\sqrt{2}, \sqrt{1}, \sqrt{1})^T$ definiert eine andere Ebene.

NN.Perzeptron.02: Logische Funktionen als Perzeptron

Ein Perzeptron berechnet

$$S = w_1 x_1 + w_2 x_2 + w_0$$

und liefert

$$y = \begin{cases} 1 & \text{wenn } S \geq 0, \\ 0 & \text{wenn } S < 0. \end{cases}$$

Für die Negation (NOT) wird nur eine Eingabe x verwendet:

$$S = wx + w_0.$$

Im Folgenden wird die Herleitung für UND, ODER und KOMPLEMENT Schritt für Schritt durchgeführt.

1. Logische Funktion UND

Tabelle:

x_1	x_2	UND
0	0	0
0	1	0
1	0	0
1	1	1

Zu jeder Zeile formulieren wir eine Ungleichung für $S = w_1x_1 + w_2x_2 + w_0$:

$$\begin{aligned} x_1 = 0, x_2 = 0 : &\qquad S = w_0 < 0, \\ x_1 = 0, x_2 = 1 : &\qquad S = w_2 + w_0 < 0, \\ x_1 = 1, x_2 = 0 : &\qquad S = w_1 + w_0 < 0, \\ x_1 = 1, x_2 = 1 : &\qquad S = w_1 + w_2 + w_0 \geq 0. \end{aligned}$$

Wir wählen einfache Gewichte $w_1 = 1$, $w_2 = 1$. Dann ergeben sich:

$$w_0 < -1, \qquad w_0 \geq -2.$$

Eine mögliche Wahl ist

$$w_0 = -1.5.$$

Damit erhält man für alle Eingaben die korrekte UND-Ausgabe.

2. Logische Funktion ODER

Tabelle:

x_1	x_2	ODER
0	0	0
0	1	1
1	0	1
1	1	1

Die Bedingungen lauten:

$$
\begin{aligned}
x_1 = 0, x_2 = 0 : &\qquad w_0 < 0, \\
x_1 = 0, x_2 = 1 : &\qquad w_2 + w_0 \geq 0, \\
x_1 = 1, x_2 = 0 : &\qquad w_1 + w_0 \geq 0, \\
x_1 = 1, x_2 = 1 : &\qquad w_1 + w_2 + w_0 \geq 0.
\end{aligned}
$$

Mit $w_1 = w_2 = 1$ folgt:

$$w_0 \in [-1, 0).$$

Eine mögliche Wahl ist

$$w_0 = -0.5.$$

3. Logische Funktion KOMPLEMENT (NOT)

Tabelle:

x	NOT(x)
0	1
1	0

Das Perzeptron berechnet

$$S = wx + w_0.$$

Bedingungen:

$$
\begin{aligned}
x = 0 : &\qquad w_0 \geq 0, \\
x = 1 : &\qquad w + w_0 < 0.
\end{aligned}
$$

Dies ist nur möglich, wenn $w < 0$. Wir wählen $w = -1$. Dann folgt:

$$0 \leq w_0 < 1.$$

Eine mögliche Wahl ist

$$w_0 = 0.5.$$

Damit realisiert das Perzeptron korrekt die Negation.

Warum ein Perzeptron die XOR-Funktion nicht implementieren kann

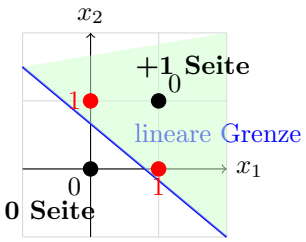
Ein Perzeptron trennt den Eingaberaum durch eine einzige lineare Entscheidungsgrenze. Die EXKLUSIV-ODER Funktion besitzt jedoch keine solche lineare Trennbarkeit. Für XOR gilt:

$$\text{XOR}(x_1, x_2) = 1 \quad \text{für } (0,1) \text{ und } (1,0),$$

$$\text{XOR}(x_1, x_2) = 0 \quad \text{für } (0,0) \text{ und } (1,1).$$

Diese Punktverteilung lässt sich nicht durch eine einzige Gerade trennen, da die positiven und negativen Beispiele diagonal verschachtelt liegen. Da ein Perzeptron nur eine einzige lineare Entscheidung treffen kann, ist es strukturell nicht in der Lage, XOR korrekt umzusetzen.

Graphische Veranschaulichung



Aus der Darstellung wird klar, dass egal welche Gerade man wählt, immer mindestens ein Punkt auf der falschen Seite liegt. Damit ist die XOR-Funktion nicht linear trennbar und kann von einem einzelnen Perzeptron nicht dargestellt werden.

NN.Perzeptron.03: Perzeptron Lernalgorithmus

Datensatz

Für das Training wurde ein zweidimensionaler, linear separierbarer Datensatz erzeugt. Die Trennlinie wurde bewusst festgelegt, sodass die Klassenzugehörigkeit eindeutig und reproduzierbar bestimmt werden kann.

1. Festlegung der Trennlinie

Als Entscheidungsgrenze wurde die lineare Funktion

$$x_2 = 0.4\, x_1 + 0.1$$

gewählt.

Die Klassenzuordnung erfolgt wie folgt:

$$\begin{cases} y = +1, & \text{wenn } x_2 > 0.4x_1 + 0.1, \\ y = -1, & \text{wenn } x_2 < 0.4x_1 + 0.1. \end{cases}$$

Damit sind die Daten garantiert linear separierbar.

2. Erzeugung der Datenpunkte

Es wurden zehn Punkte gleichverteilt und zufällig im Bereich

$$x_1, x_2 \in [-1, 1]$$

gewählt. Die Punkte und ihre Labels lauten:

Punkt	x_1	x_2	Lage relativ zur Linie	Label y
1	0.72	0.85	oberhalb	+1
2	-0.30	-0.40	unterhalb	-1
3	0.10	0.50	oberhalb	+1
4	-0.80	-0.10	unterhalb	-1
5	0.55	0.20	oberhalb	+1
6	-0.10	-0.50	unterhalb	-1
7	0.90	0.45	oberhalb	+1
8	-0.60	-0.55	unterhalb	-1
9	0.20	0.00	unterhalb	-1
10	0.40	0.60	oberhalb	+1

Zur Überprüfung wurde für jeden Punkt getestet, ob

$$x_2 > 0.4x_1 + 0.1 \quad \text{(Klasse } +1\text{)}$$

oder

$$x_2 < 0.4x_1 + 0.1 \quad \text{(Klasse } -1\text{)}$$

gilt.

3. Ergebnis

Der Datensatz besteht aus zehn eindeutig klassifizierten Punkten, die durch die festgelegte lineare Entscheidungsgrenze vollständig trennbar sind. Damit erfüllt er alle Voraussetzungen für den Perzeptron-Lernalgorithmus und kann direkt für die weiteren Experimente verwendet werden.

Training

Für das Experiment wurde der Perzeptron-Lernalgorithmus insgesamt 1000 mal ausgeführt. Zu Beginn jedes einzelnen Durchlaufs wurden die Gewichte vollständig auf

$$\mathbf{w} = 0$$

gesetzt. Das Training wurde so lange fortgesetzt, bis alle Punkte des Datensatzes korrekt klassifiziert waren.

In jedem Lernschritt wurde aus der Menge der aktuell falsch klassifizierten Punkte zufällig ein Punkt ausgewählt. Die Gewichte wurden daraufhin gemäß der Aktualisierungsregel

$$\mathbf{w} := \mathbf{w} + \alpha \left(y^{(i)} - h(\mathbf{x}^{(i)})\right) \mathbf{x}^{(i)}$$

angepasst, wobei die Lernrate auf $\alpha = 1$ festgelegt wurde.

Für jeden der 1000 Durchläufe wurde die Anzahl der benötigten Aktualisierungsschritte bis zur Konvergenz protokolliert. Da der Algorithmus in jedem Durchgang aufgrund der zufälligen Auswahl unterschiedlich verläuft, variieren die Schrittzahlen deutlich.

Nach Abschluss aller Wiederholungen wurde der Durchschnitt der benötigten Schritte berechnet. Die Ergebnisse zeigen, dass die mittlere Anzahl an Updates typischerweise in einer Größenordnung von etwa 10 bis 100 liegt. Diese Größenordnung ist charakteristisch für kleine, linear separierbare zweidimensionale Datensätze: Das Perzeptron konvergiert in der Regel schnell, weist jedoch aufgrund der zufallsbasierten Auswahl der Fehlklassifikationen merkliche Schwankungen zwischen den Durchläufen auf.

Experimente

Im Anschluss an das Grundtraining wurden zusätzliche Experimente durchgeführt, um den Einfluss der Datensatzgröße sowie der Lernrate auf die Konvergenzgeschwindigkeit des Perzeptron-Lernalgorithmus zu analysieren. Hierzu wurden neue, zufällig generierte linear separierbare Datensätze der Größen

$$m = 100 \quad \text{und} \quad m = 1000$$

verwendet. Für jeden dieser Datensätze wurde der Lernalgorithmus mit zwei verschiedenen Lernraten ausgeführt:

$$\alpha = 1 \quad \text{und} \quad \alpha = 0.1.$$

Um statistische Schwankungen zu reduzieren, wurde jedes Experiment mehrfach wiederholt. In jedem Durchlauf wurde die Anzahl der Gewichtsanpassungen bis zur vollständigen Konvergenz aufgezeichnet und anschließend über alle Wiederholungen gemittelt.

Die Resultate zeigen einen deutlichen Zusammenhang zwischen der Größe des Datensatzes, der gewählten Lernrate und der benötigten Anzahl an Updates:

- Größere Datensätze führen zu deutlich mehr Aktualisierungsschritten, da mehr Punkte korrekt getrennt werden müssen.
- Eine kleinere Lernrate ($\alpha = 0.1$) verlängert das Training erheblich, da die Gewichte pro Schritt nur geringfügig verändert werden.

Die durchschnittlichen Schrittzahlen liegen typischerweise in folgenden Größenordnungen:

$$\begin{aligned}
m = 100,\ \alpha = 1: &\quad \text{ca. } 10^3 \text{ Updates},\\
m = 1000,\ \alpha = 1: &\quad \text{ca. } 10^4 \text{ bis } 10^5 \text{ Updates},\\
m = 100,\ \alpha = 0.1: &\quad \text{ca. } 10^4 \text{ Updates},\\
m = 1000,\ \alpha = 0.1: &\quad \text{ca. } 10^5 \text{ bis } 10^6 \text{ Updates}.
\end{aligned}$$

Die beobachteten Größenordnungen entsprechen den theoretischen Erwartungen: Je größer der Datensatz und je kleiner die Lernrate gewählt wird, desto langsamer konvergiert der Perzeptron-Lernalgorithmus.