

Nr. 1

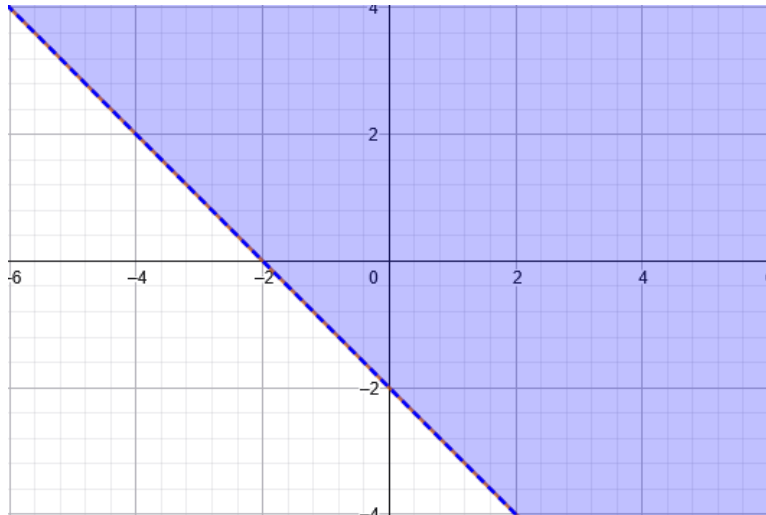
Betrachten Sie das durch den Gewichtsvektor $(w_0, w_1, w_2)^T = (2, 1, 1)^T$ gegebene Perzeptron. Zeichnen Sie die Trennebene und markieren Sie den Bereich, der mit +1 klassifiziert wird.

$$2 + x_1 + x_2 > -2 \text{ (+1 Klasse)}$$

$$2 + x_1 + x_2 < -2 \text{ (-1 Klasse)}$$

Geogebra Eingabe: $x + y = -2$

Markierung: $x + y > -2$



blau markiert: +1 Klasse

Welche der folgenden Perzeptrons haben die selbe Trennebene? Welche weisen exakt die gleiche Klassifikation auf?

- $(w_0, w_1, w_2)^T = (1, 0.5, 0.5)^T$
- $(w_0, w_1, w_2)^T = (200, 100, 100)^T$
- $(w_0, w_1, w_2)^T = (2, 1, 1)^T$
- $(w_0, w_1, w_2)^T = (-2, -1, -1)^T$

Alle Perzeptrons haben die selbe Trennebene, da sie nur skaliert worden sind.

Perzeptron	gleiche Klassifikation	Skalarfaktor zu 2,1,1
1, 0.5, 0.5	ja	0.5
200, 100, 100	ja	100
$\sqrt{2}, \sqrt{1}, \sqrt{1}$	ja	$\sqrt{2}$
-2, -1, -1	nein, Klassifikation invertiert	-1

Bei einem positiven Skalarfaktor bleibt die + oder - Klassifikation gleich, bei einem Negativen werden aber die -1 und die +1 Klassen invertiert. Deswegen ist die Klassifikation unterschiedlich.

Nr. 2

Das Perzeptron kann zur Ausführung zahlreicher logischer Funktionen verwendet werden. Implementieren Sie die binären Logikfunktionen UND, ODER und KOMPLEMENT

UND Funktion: $-1,9 + 1x + 1y$

- nur wenn x und y beide 1 sind, ist das Ergebnis größer als 0. Dann ist die Klassifizierung +1, sonst ist sie -1.

ODER Funktion: $-0,1 + 1x + 1y$

- x oder y muss 1 sein, damit das Ergebnis größer ist als 0.

NOT Funktion: $0,1 - 1x - 1y$

- wenn x = 1 oder y = 1, dann ist das Ergebnis kleiner als 0 und wird negativ gewertet.

Eine grundlegende Einschränkung des Perzeptrons besteht darin, dass es die EXKLUSIV-ODER-Funktion nicht implementieren kann. Erklären Sie den Grund für diese Einschränkung.

Das Perzeptron hat eine lineare Funktion, daher kann man das XOR nicht richtig trennen. Egal, wie man eine Linie ziehen würde, man würde bestimmte Punkte falsch Klassifizieren.

Nr. 3

Führen Sie nun den Perzeptron-Lernalgorithmus 1000 mal hintereinander aus. Initialisieren Sie jedes Mal die Gewichte mit 0. Wählen Sie in jedem Lernschritt einen Punkt $x(i)$ zufällig aus der Menge der falsch klassifizierten Punkte und aktualisieren Sie die Gewichte entsprechend der folgenden Formel:

$$w := w + \alpha (y(i) - h(x(i))) x(i)$$

Nehmen Sie $\alpha = 1$ als Lernrate. Halten Sie für jeden Durchlauf fest, wie viele Schritte der Algorithmus benötigt, um zu der endgültigen Hypothese $h^(x)$ zu konvergieren. Berechnen Sie am Ende die durchschnittliche Anzahl von benötigten Schritten. In welcher Größenordnung liegt sie?*

Die durchschnittliche Schrittzahl für $m=10$ ist ca. 4.4 Schritte, kleine Datensätze werden anscheinend sehr schnell gelernt.

Wiederholen Sie das obige Experiment mit $m = 100$ und $m = 1000$ Datenpunkten, jeweils ein Mal mit den Lernraten $\alpha = 1$ und $\alpha = 0.1$. In welcher Größenordnung liegt die durchschnittliche Anzahl von benötigten Schritten in diesen Fällen?

Um eine zuverlässigere Schätzung zu erhalten, können Sie dasselbe Experiment mehrfach mit anderen zufällig generierten Datensätzen derselben Größe m wiederholen und danach den Durchschnitt über alle Wiederholungen betrachten.

Die Schrittzahl wächst bei einzelnen Experimenten eher proportional zur Anzahl der Punkte, ist aber bei unterschiedlichen seeds ganz anders. Die Lernrate ($\alpha=1$ und $\alpha=0.1$) hat keinen Einfluss auf die Anzahl der Schritte. Sie ändert nur die Größe der Gewichtsänderung.

Experimente mit verschiedenen seeds (für Punkte und boundary):

$m=10$: ca. 3-40 Schritte

$m=100$: ca. 50-100 Schritte

$m=1000$: ca. 100-500 Schritte

Je öfter ich unterschiedliche seeds verwendet habe, desto unterschiedlicher waren die Werte für die benötigten Schritte.