Maschinelles Lernen 04

Prof. Dr. Sarah Brockhaus

Hochschule München

13. April 2023

Perzeptron

Perzeptron

Perzeptron

Biologischer Hintergrund

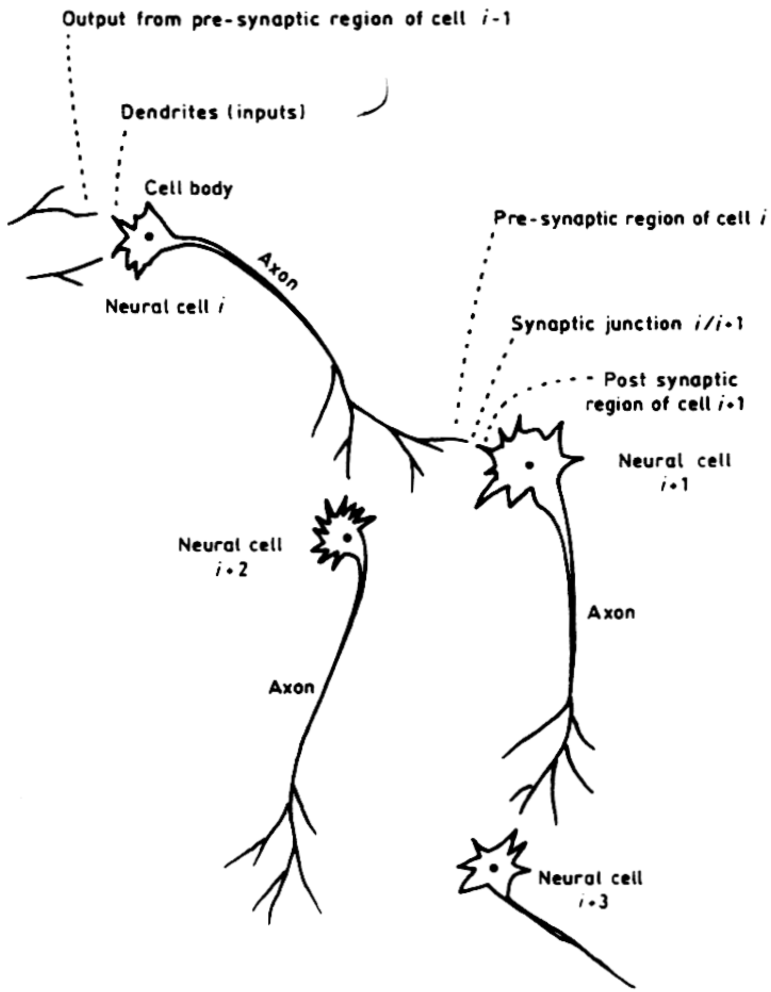


Abbildung 1: Biologisches neuronales Netz [Principles of Artificial Neural Networks, D. Graupe, 1997]

Perzeptron

Biologischer Hintergrund

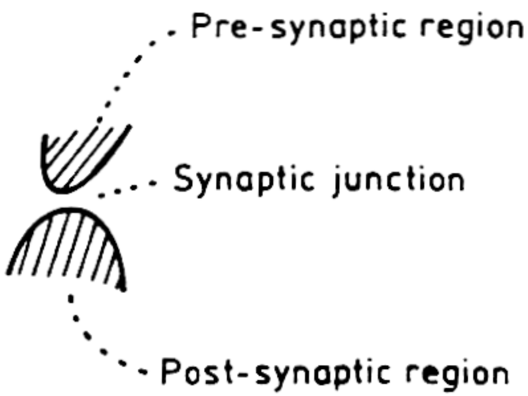


Abbildung 2: Synaptischer Spalt [Principles of Artificial Neural Networks, D. Graupe, 1997]

- In einem biologischen neuronalen Netz findet Berechnung statt, indem elektrische Ladungen zwischen Nervenzellen ausgetauscht wird.
- Die elektrische Ladung wandert das Axon entlang, bis sie durch Diffusion den synaptischen Spalt überwindet und von den Dendriten anderen Neuronen aufgegriffen wird.

Perzeptron

Biologischer Hintergrund

- Ein Neuron kann viele Synapsen haben und somit mit hunderten weiteren Neuronen verknüpft sein.
- Ein Neuron kann auch viele Dendriten besitzen und somit Input von vielen Neuronen besitzen.
- Verbindungen können verstärkend or hemmend wirken, je nach der Chemie innerhalb des synapischen Spalts.

Perzeptron

Biologischer Hintergrund

	Computer	Biologische neuronale Netze
Einheiten	Prozessoren	Neuronen
Geschwindigkeit	GHz	100 Hz
Signal/Rauschen	$\gg 1$	~ 1
Signalgeschw.	$\sim 10^8 m/s$	$\sim 1 m/s$
Berechnung	sequenziell	parallel
Konfiguration	Programm und Daten	Verbindungen und Chemie (Synapsen)
Programmierung	statisch	adaptiv
Robustheit	gering	hoch
Anwendbarkeit	nur bekannte Daten	chaotische, unvorhergesehene, inkonsistente Daten

Tabelle 1: Vergleich der Berechnungsmodelle adaptiert von [Theory of Neural Information Processing Systems, A. Coolean et al., 2005]

Perzeptron

Einführung

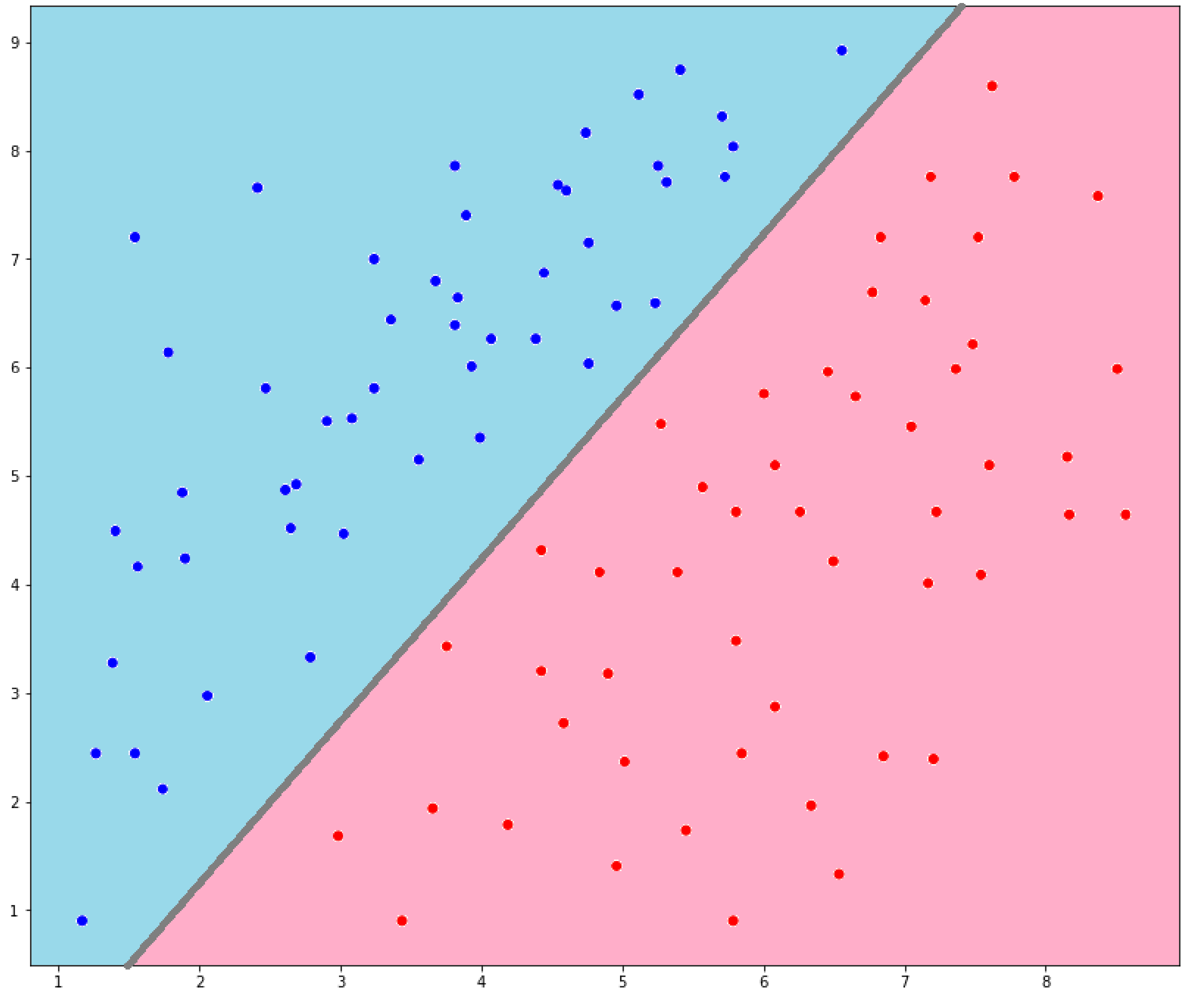


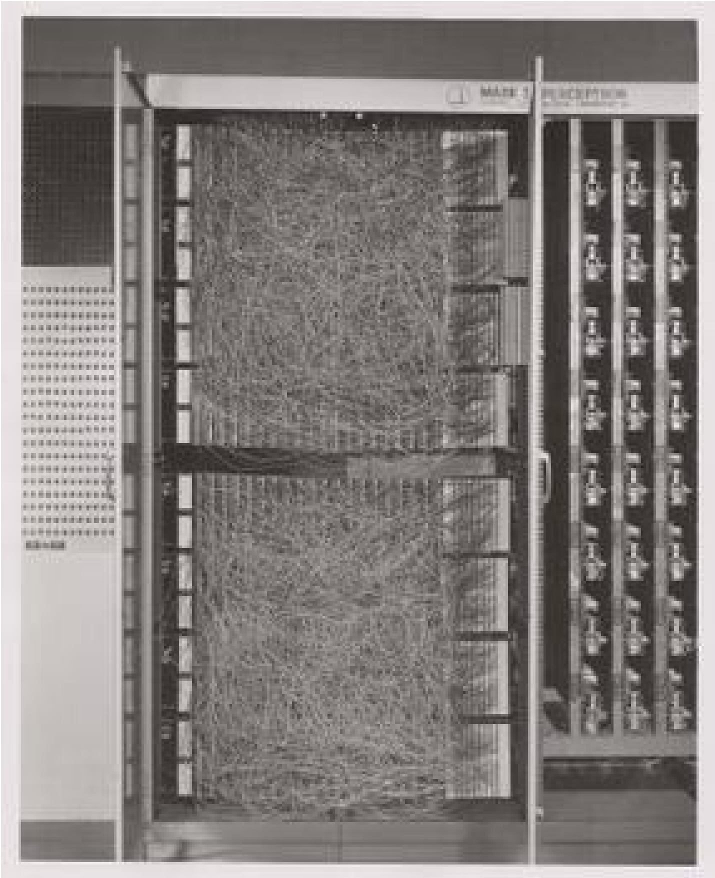
Abbildung 3: Binärer Klassifikator $f : \mathbb{R}^d \rightarrow \{0, 1\}$, welcher jedem Punkt $\mathbf{x} \in \mathbb{R}^d$ eine Klasse 0 oder 1 zuweist. In diesem Beispiel ist $d = 2$.

Perzeptron

Einführung

Geschichte des Perzeptrons [https://en.wikipedia.org/wiki/Perceptron]:

- Erfunden 1957 von Frank Rosenblatt als physische Maschine, dem Mark 1 perceptron
- Ursprüngliches Design als Software auf einem IBM 704
- 20×20 Fotozellen (pixels) für Bilderkennung
- Lernen der Parameter durch die Anpassung von Potentiometern mit Hilfe elektrischer Motoren



Perzeptron

Einführung

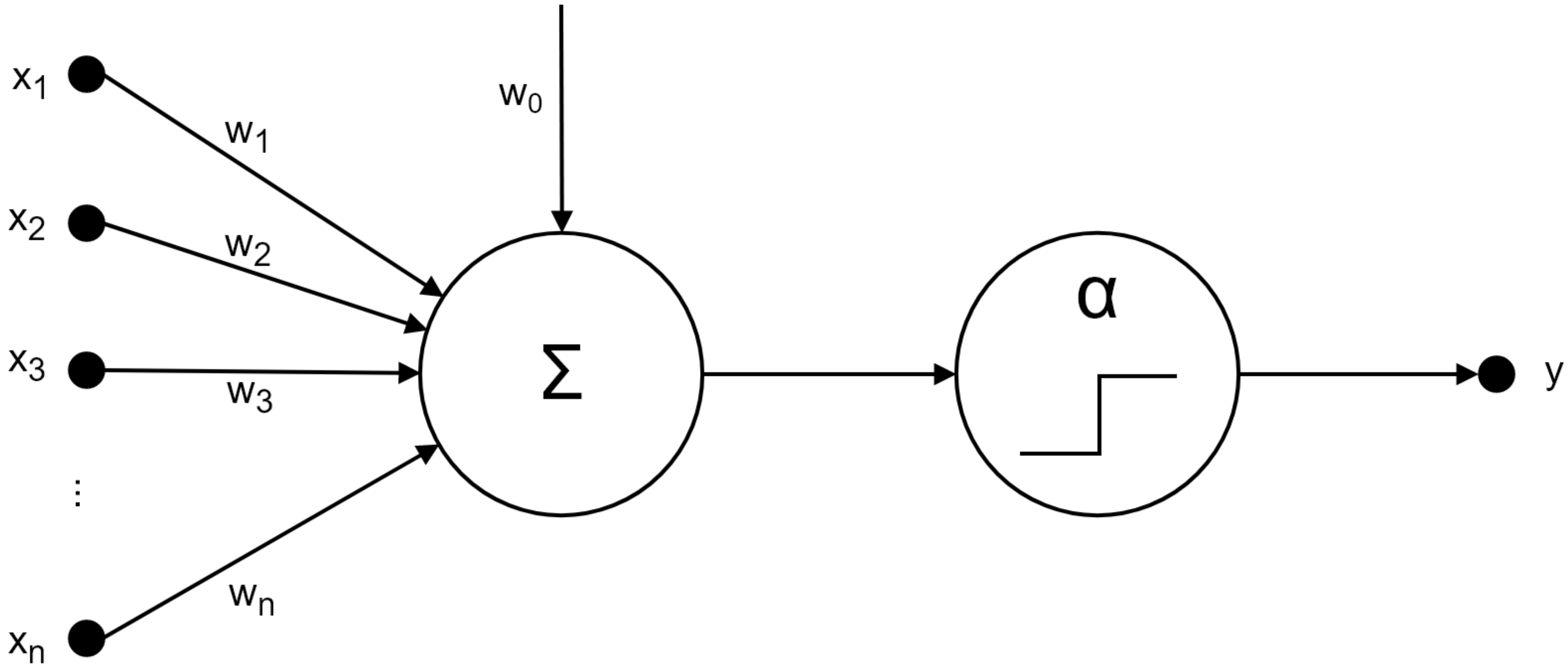


Abbildung 4: Grafische Darstellung eines Perzeptrons.

Perzeptron
Einführung

Heaviside Funktion

Die Heaviside Aktivierungsfunktion ist definiert durch

$$\alpha(x) = \begin{cases} 1 & \text{falls } x > 0 \text{ und} \\ 0 & \text{anderfalls.} \end{cases}$$

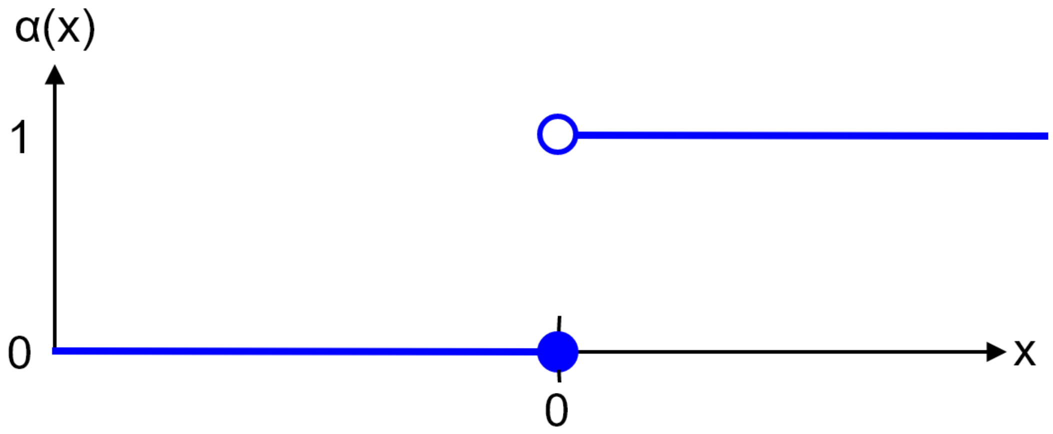


Abbildung 5: Die Heaviside Aktivierungsfunktion, wie sie im Perzeptron verwendet wird.

Perzeptron
Einführung

Perzeptron

Ein Perzeptron ist ein binärer Klassifikator $f : \mathbb{R}^d \to \{0, 1\}$ definiert durch

$$f(\mathbf{x}) = \alpha(\mathbf{w} \circ \mathbf{x} + \mathbf{w}_0)$$

wobei die Aktivierungsfunktion α die Heavyside Funktion ist.

Perzeptron
Einführung

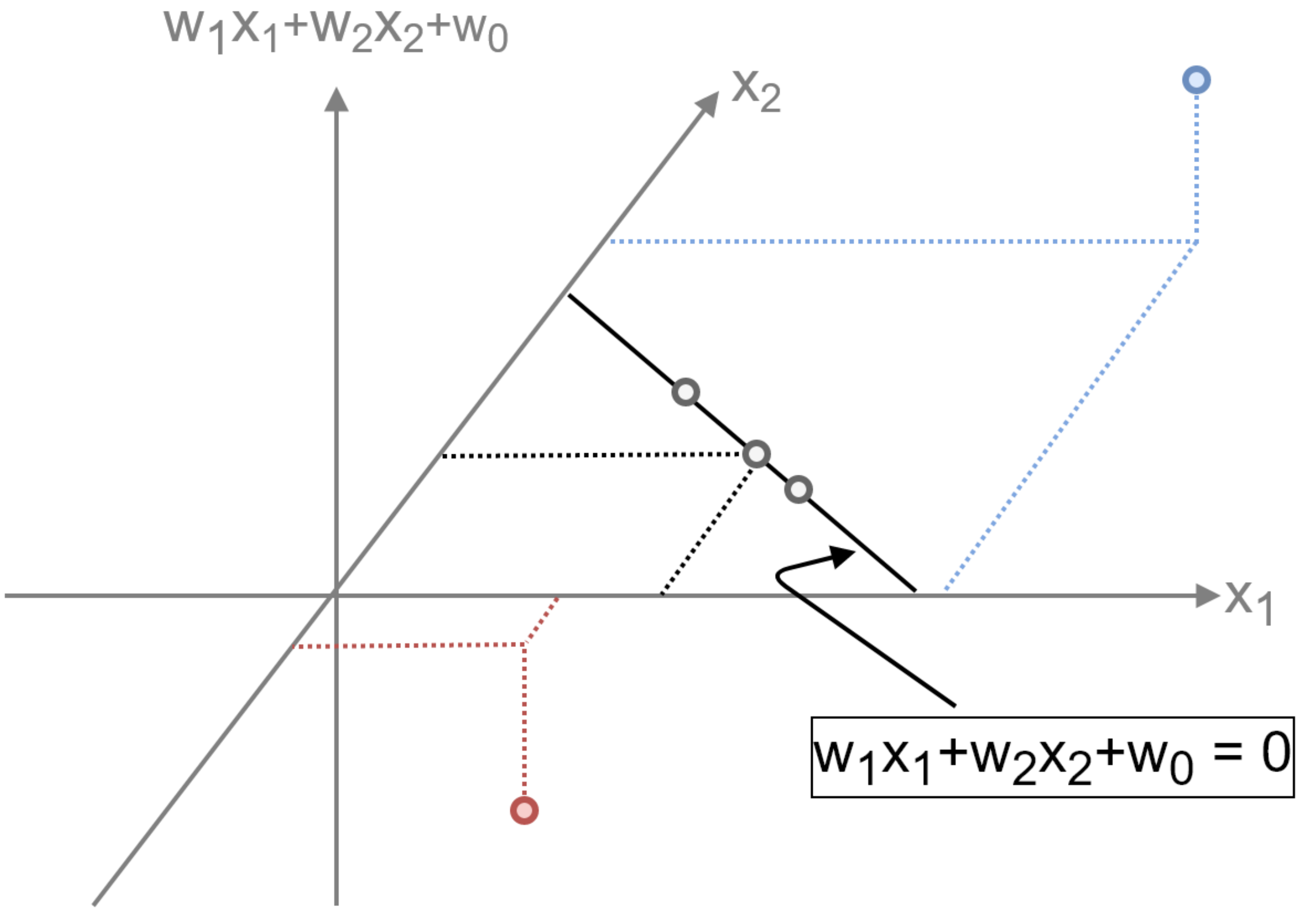
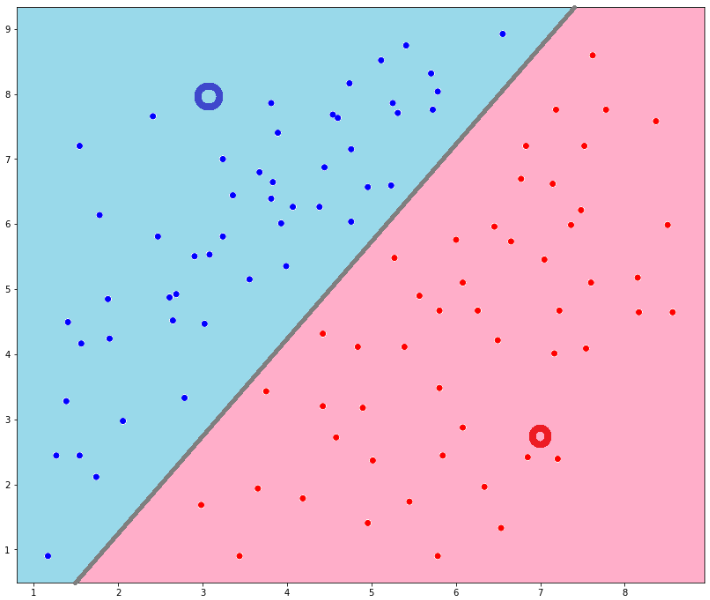


Abbildung 6: Grafische Darstellung der Hyperebene $\mathbf{w} \circ \mathbf{x} + \mathbf{w}_0 = 0$.

Perzeptron

Einführung



Beispiel

$\mathbf{w}_0 = 17, \mathbf{w}_1 = -37, \mathbf{w}_2 = 30$

- $\mathbf{x} = [7\ 3]^T$:
  $-37 \cdot 7 + 30 \cdot 3 + 17 = -152 \leq 0 \Rightarrow$
  class 0
- $\mathbf{x} = [3\ 8]^T$:
  $-37 \cdot 3 + 30 \cdot 8 + 17 = 146 > 0 \Rightarrow$
  class 1

Perzeptron
Einführung

Alternative Repräsentation

In der Literatur wird das Perzeptron auch oft definiert durch

$$f(\mathbf{x}) = \alpha(\mathbf{w} \circ \mathbf{x}).$$

Hier ist $\mathbf{x} = (1, x_1, \dots, x_n)^T$ und $\mathbf{w} = (w_0, w_1, \dots, w_n)^T$, d.h., $x_0 = 1$ und w_0 sind in $\mathbf{x}$ und $\mathbf{w}$ enthalten.

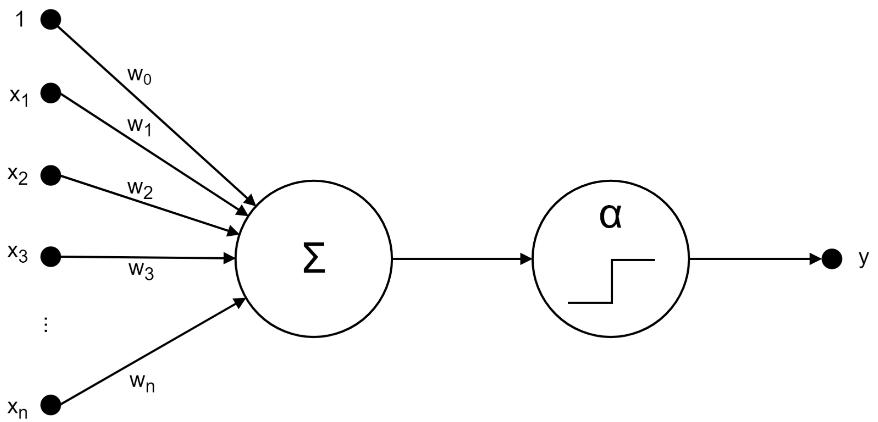


Abbildung 7: Alternative grafische Darstellung eines Perzeptron.

Perzeptron

Lernalgorithmus

Perzeptron Parameter

Für ein Perzeptron

$$f(\mathbf{x}) = \alpha(\mathbf{w} \circ \mathbf{x}),$$

müssen die Parameters $\mathbf{w} = (\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_n)^T$ mit Hilfe einer Lernregel bestimmt werden.

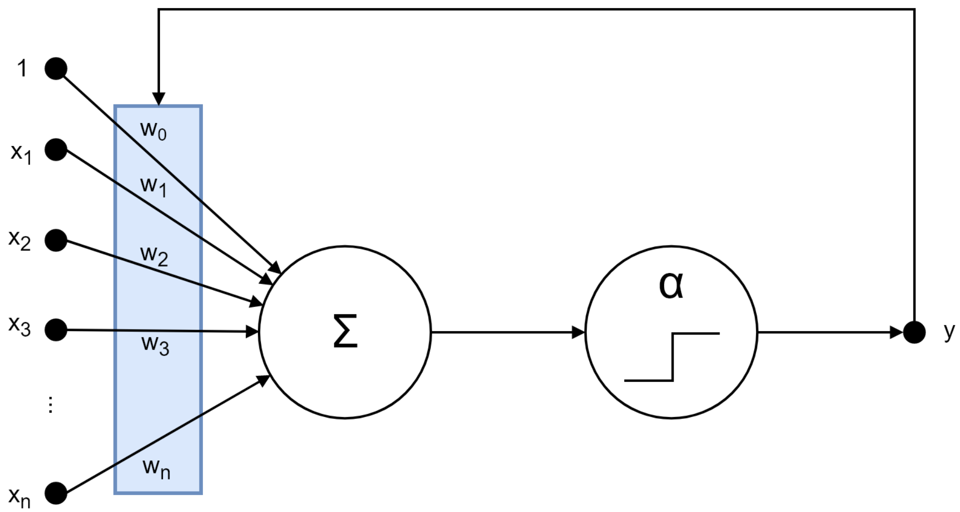


Abbildung 8: Schema des Perzeptron Lernalgorithmus.

Perzeptron

Lernalgorithmus

Idee eines iterativen Lernalgorithmus

- Beginne mit einer zufälligen oder festen Wahl für $\mathbf{w}$ (z.B. $\mathbf{w} = \mathbf{0}$)
- Bestimme die falsch klassifizierten Datenpunkte
- Versuche iterativ die einzelnen Parameter so zu verändern, dass die Anzahl der falsch klassifizierten Datenpunkte sinkt
- Höre auf sobald keine Verbesserung mehr eintritt

Perzeptron

Lernalgorithmus

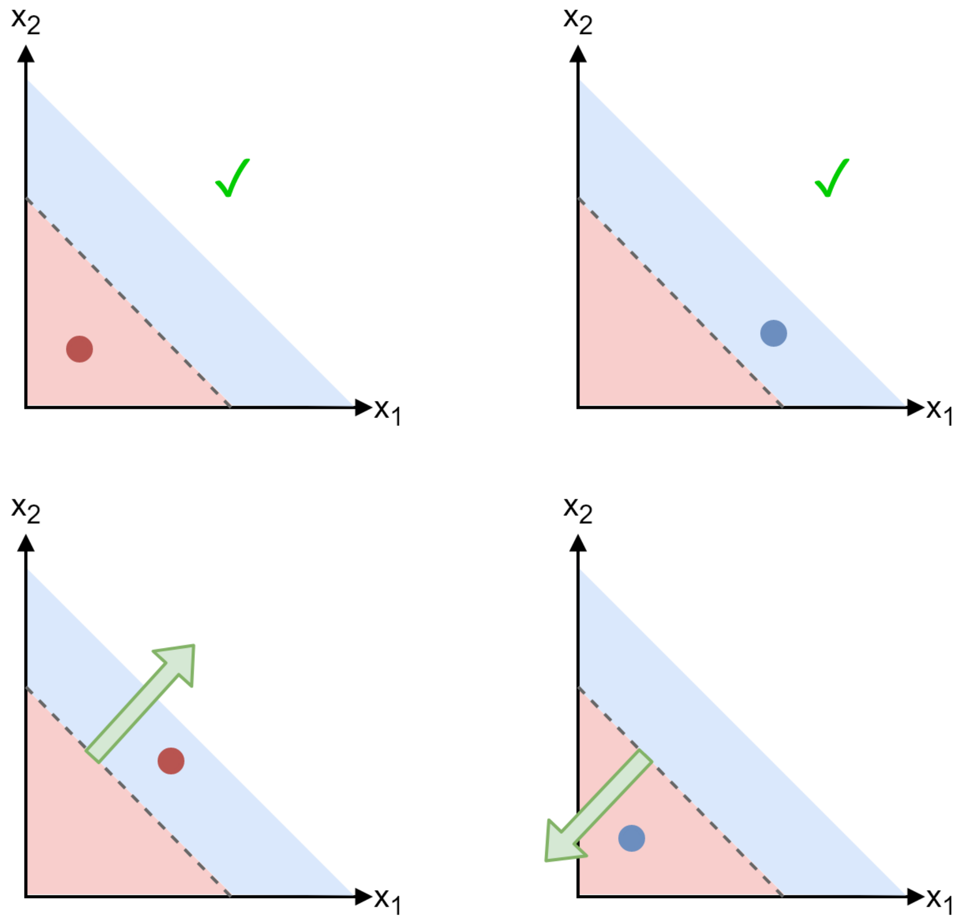


Abbildung 9: Die vier möglichen Fälle, die bei binärer Klassifikation auftreten können.

Perzeptron

Lernalgorithmus

Fall 1

$\mathbf{x}$ wurde als Klasse 1 eingestuft sollte jedoch Klasse 0 sein:

- Falschklassifikation: $f_{\mathbf{w}}(\mathbf{x}) = \alpha(\mathbf{w} \circ \mathbf{x}) = 1 \Rightarrow \mathbf{w} \circ \mathbf{x} > 0$
- Update: $\mathbf{w}' = \mathbf{w} - \mathbf{x}$
- Auswirkung: $\mathbf{w}' \circ \mathbf{x} = (\mathbf{w} - \mathbf{x}) \circ \mathbf{x} = \mathbf{w} \circ \mathbf{x} - \underbrace{\mathbf{x} \circ \mathbf{x}}_{\geq 0}$
- Daher wahrscheinlicher $\mathbf{w} \circ \mathbf{x} - \mathbf{x} \circ \mathbf{x} \leq 0$ und schließlich $f_{\mathbf{w}'}(\mathbf{x}) = \alpha(\mathbf{w} \circ \mathbf{x} - \mathbf{x} \circ \mathbf{x}) = 0$.

Perzeptron

Lernalgorithmus

Fall 2

$\mathbf{x}$ wurde als Klasse 0 eingestuft sollte jedoch Klasse 1 ein:

- Falschklassifikation: $f_{\mathbf{w}}(\mathbf{x}) = \alpha(\mathbf{w} \circ \mathbf{x}) = 0 \Rightarrow \mathbf{w} \circ \mathbf{x} \leq 0$
- Update: $\mathbf{w}' = \mathbf{w} + \mathbf{x}$
- Auswirkung: $\mathbf{w}' \circ \mathbf{x} = (\mathbf{w} + \mathbf{x}) \circ \mathbf{x} = \mathbf{w} \circ \mathbf{x} + \underbrace{\mathbf{x} \circ \mathbf{x}}_{\geq 0}$
- Daher wahrscheinlicher $\mathbf{w} \circ \mathbf{x} + \mathbf{x} \circ \mathbf{x} > 0$ und schließlich $f_{\mathbf{w}'}(\mathbf{x}) = \alpha(\mathbf{w} \circ \mathbf{x} + \mathbf{x} \circ \mathbf{x}) = 1$.

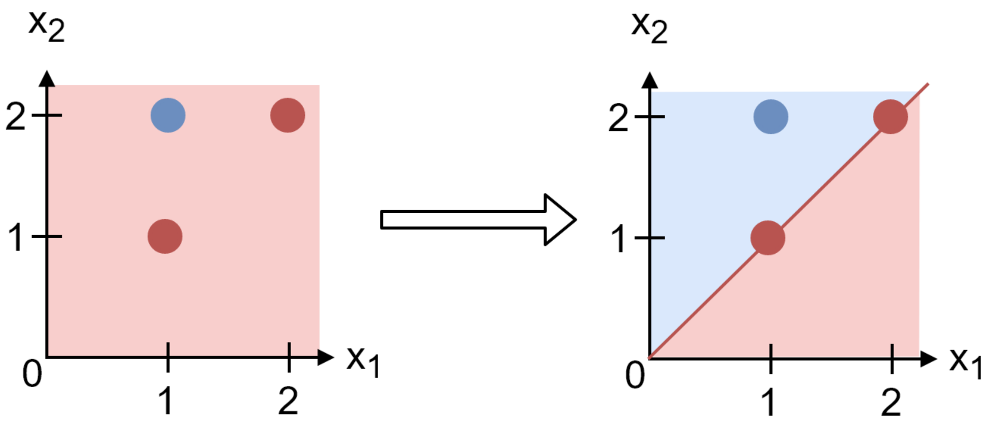
Perzeptron

Lernalgorithmus

Algorithm 1 perceptron_learn($\{(\mathbf{x}^{(i)}, y^{(i)})\} \subset (\mathbb{R}^d \times \{0,1\})^n, \gamma$)

$\mathbf{w} = \mathbf{0}$
while $\frac{1}{n}\sum_{i=1}^{n} |y^{(i)} - \alpha(\mathbf{w} \circ \mathbf{x}^{(i)})| > \gamma$ **do**
    $\mathbf{w}' = \mathbf{w}$
    for $i = 1, \ldots, n$ **do**
        $o^{(i)} = \alpha(\mathbf{w} \circ \mathbf{x}^{(i)})$
        $\mathbf{w}' = \mathbf{w}' + (y^{(i)} - o^{(i)})\mathbf{x}^{(i)}$
    end for
    $\mathbf{w} = \mathbf{w}'$
end while

Perzeptron
Lernalgorithmus



$$(\mathbf{x}^{(1)}, y^{(1)}) = ([1,1]^T, 0)$$
$$(\mathbf{x}^{(2)}, y^{(2)}) = ([1,2]^T, 1)$$
$$(\mathbf{x}^{(3)}, y^{(3)}) = ([2,2]^T, 0)$$

Iterationen:

1. $\mathbf{w} = [0,0,0]^T$, $o^{(1)} = 0$ ✓, $o^{(3)} = 0$ ✓, $o^{(2)} = 0$ –
   $\Rightarrow \mathbf{w} = [0,0,0]^T + [1,1,2]^T = [1,1,2]^T$
2. $\mathbf{w} = [1,1,2]^T$, $o^{(2)} = 1$ ✓, $o^{(1)} = 1$ –, $o^{(3)} = 1$ –
   $\Rightarrow \mathbf{w} = [1,1,2]^T - [1,1,1]^T - [1,2,2]^T = [-1,-2,-1]^T$
3. $\mathbf{w} = [-1,-2,-1]^T$, $o^{(1)} = 0$ ✓, $o^{(3)} = 0$ ✓, $o^{(2)} = 0$ –
   $\Rightarrow \mathbf{w} = [-1,-2,-1]^T + [1,1,2]^T = [0,-1,1]^T$
4. $\mathbf{w} = [0,-1,1]^T$, $o^{(1)} = 0$ ✓, $o^{(2)} = 1$ ✓, $o^{(3)} = 0$ ✓

Perzeptron

Grenzen des Perzeptrons

Lineare Trennbarkeit

Probleme wie das Exklusiv-Oder (XOR), welche nicht linear trennbar sind, können von einem Perzeptron nicht gelernt werden.

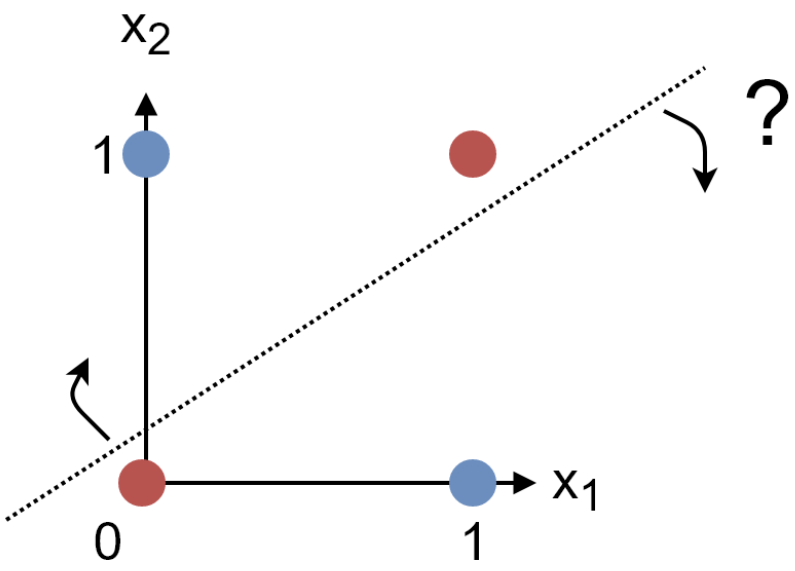


Abbildung 10: Ein Perzeptron kann das Exclusiv-Oder nicht lernen, da es keine Gerade gibt, die die beiden Klassen trennt.

Perzeptron

Grenzen des Perzeptrons

Uneindeutigkeit

Auch wenn ein Problem linear trennbar ist, erhält man mit dem Perzeptron Lernalgorithmus kein eindeutiges Modell.

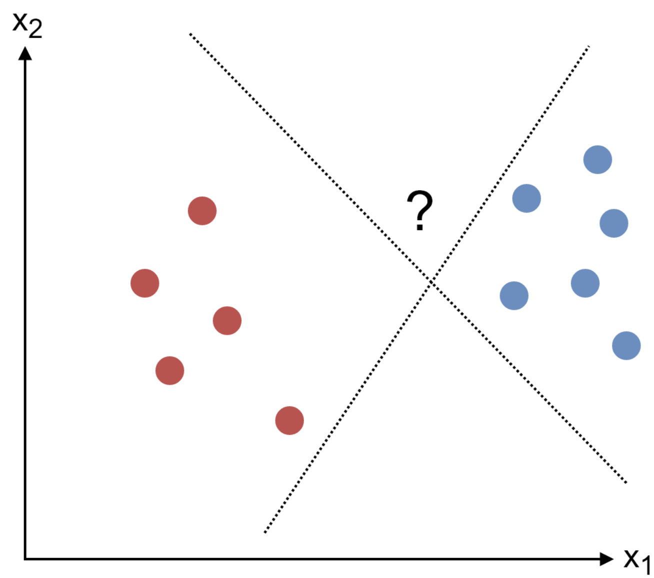


Abbildung 11: In diesem Beispiel gibt es unendlich viele Geraden, welche die Klassen trennen und der Perzeptron Lernalgorithmus gibt nur eine der Lösungen zurück.

Adaline

Adaline

Adaline

Einführung

- Das Adaline (Adaptive Linear Neuron) wurde 1960 von B. Widow in 1960 eingeführt.
- Es ähnelt im Aufbau dem Perzeptron besitzt jedoch eine andere Aktivierungsfunktion und einen unterschiedlichen Lernalgorithmus genannt Deltaregel.

Adaline

Einführung

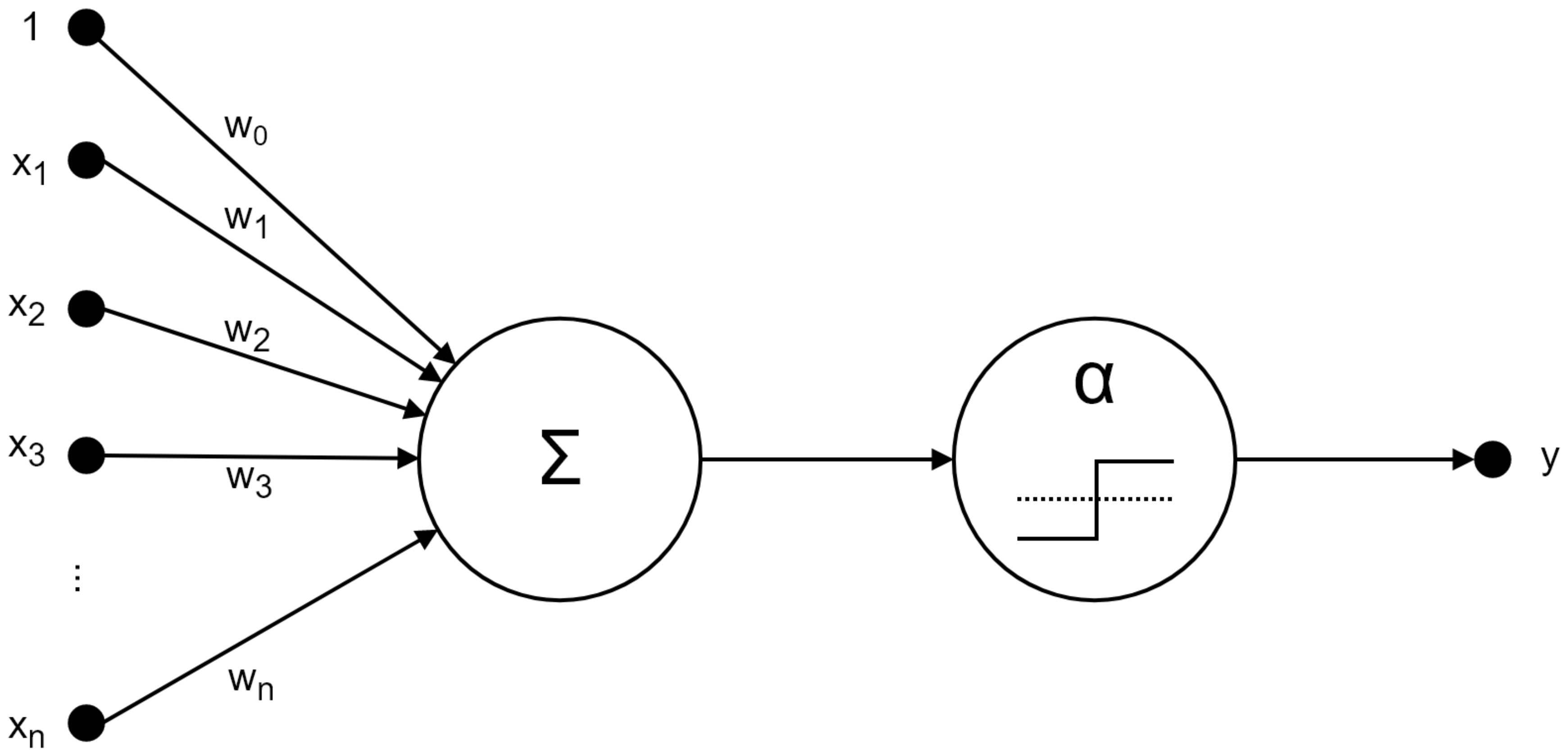


Abbildung 12: Grafische Darstellung eines Adalines.

Adaline

Einführung

Signum Aktivierungsfunktion

Die Signum Aktivierungsfunktion ist definiert als

$$\alpha(x) = \begin{cases} 1 & \text{falls } x > 0, \\ 0 & \text{falls } x = 0, \text{und} \\ -1 & \text{andernfalls.} \end{cases}$$

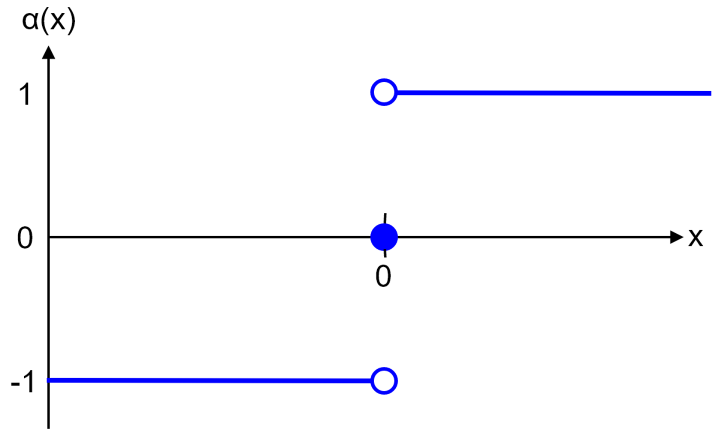


Abbildung 13: Plot der Signum Aktivierungsfunktion.

Adaline

Einführung

Adaline

Das Adaline ist ein Binärklassifikator $f : \mathbb{R}^d \to \{-1, 0, 1\}$ definiert als

$$f(\mathbf{x}) = \alpha(\mathbf{w} \circ \mathbf{x} + \mathbf{w}_0)$$

wobei α die Signum Aktvierungsfunktion ist.

Notation

Auch hier nehmen wir implizit an, dass $\mathbf{x}_0 = 1$ und $\mathbf{w}$ den Biasparameter $\mathbf{w}_0$ beinhaltet.

Adaline

Lernalgorithmus

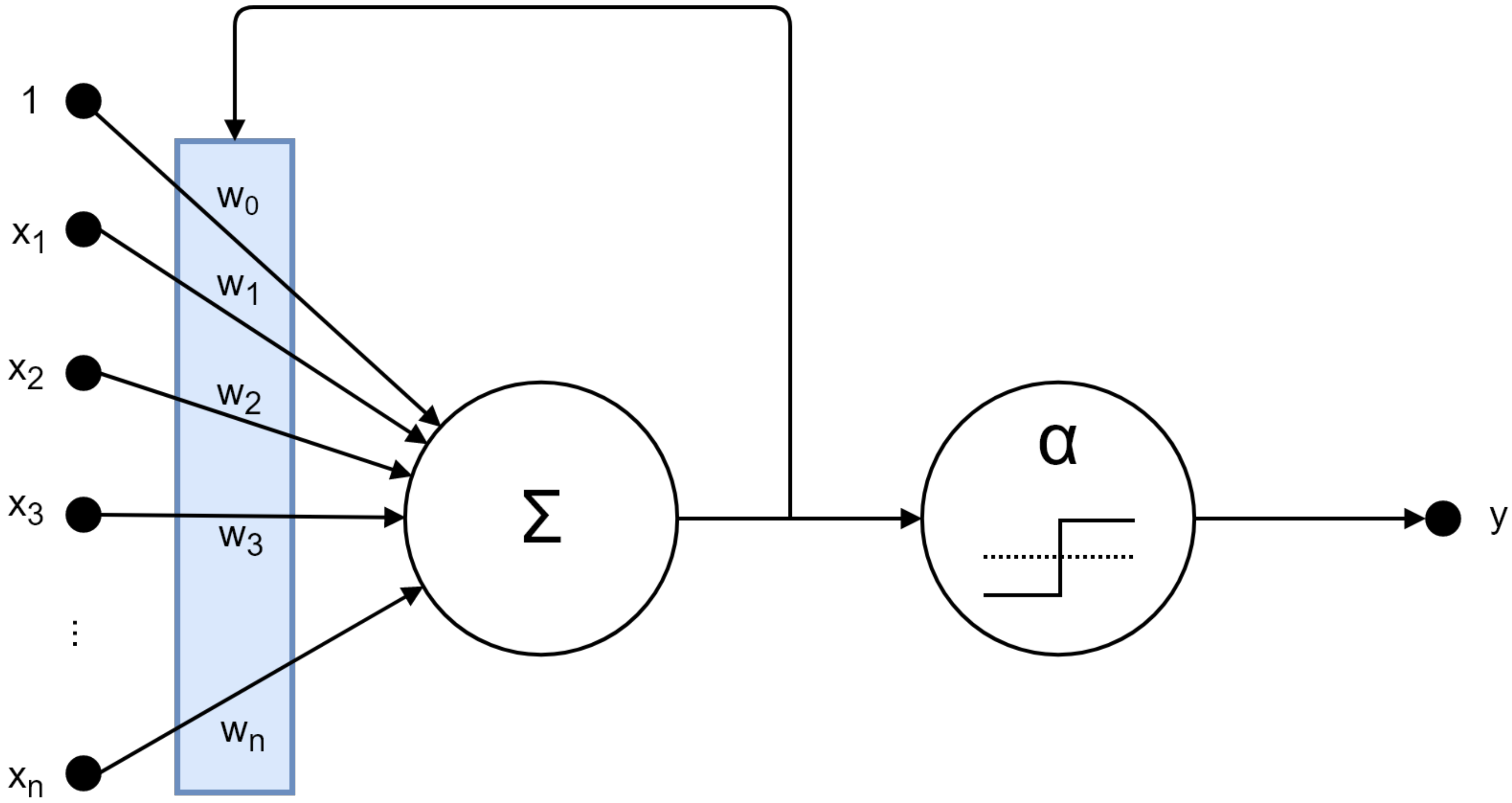


Abbildung 14: Schema des Adaline Lernalgorithmus.

Adaline

Lernalgorithmus

Wie bei der linearen regression, verwenden wir beim Adaline ein bekanntes Fehlermaß inspiriert durch die RSS/den MSE in Vebrindung mit dem Gradientenabstiegsverfahren, um dem negativen Gradient des Fehlermaßes zum Minimum zu folgen:

$$E(\mathbf{w})^{(i)} = \frac{1}{2}\left(y^{(i)} - f(\mathbf{x}^{(i)})\right)^2 = \frac{1}{2}\left(y^{(i)} - \mathbf{w} \circ \mathbf{x}^{(i)}\right)^2.$$

$$\frac{\partial E(\mathbf{w})^{(i)}}{\partial \mathbf{w}_j} = \left(y^{(i)} - \mathbf{w} \circ \mathbf{x}^{(i)}\right) \underbrace{\frac{\partial}{\partial \mathbf{w}_j}\left(y^{(i)} - \mathbf{w} \circ \mathbf{x}^{(i)}\right)}_{-\mathbf{x}_j^{(i)}} = -\left(y^{(i)} - \mathbf{w} \circ \mathbf{x}^{(i)}\right)\mathbf{x}_j^{(i)}$$

$$\nabla_{\mathbf{w}} E(\mathbf{w})^{(i)} = \left(\frac{\partial E(\mathbf{w})^{(i)}}{\partial \mathbf{w}_0}, \ldots, \frac{\partial E(\mathbf{w})^{(i)}}{\partial \mathbf{w}_n}\right)^T = -\left(y^{(i)} - \mathbf{w} \circ \mathbf{x}^{(i)}\right)\mathbf{x}^{(i)}$$

Adaline

Lernalgorithmus

Algorithm 2 adaline_learn($\{(\mathbf{x}^{(i)}, y^{(i)})\} \subset (\mathbb{R}^d \times \{-1, 1\})^n$, η, γ)

1: $\mathbf{w} = \mathbf{0}$
2: **while** $\frac{1}{n}\sum_{i=1}^{n} |y^{(i)} - \alpha(\mathbf{w} \circ \mathbf{x}^{(i)})| > \gamma$ **do**
3:     **for** $i = 1, \dots, n$ **do**
4:         $\mathbf{w} = \mathbf{w} + \eta(y^{(i)} - \mathbf{w} \circ \mathbf{x}^{(i)})\mathbf{x}^{(i)}$
5:     **end for**
6: **end while**

Das Update in Zeile 4 des Adaline Lernalgorithmus ist auch bekannt als Deltaregel.

Adaline

Lernalgorithmus

Eigenschaften des Adaline

- Aufgrund seiner linearen Natur kann auch das Adaline die XOR-Funktion nicht direkt lernen.
- Je nach Datensatz und Initialisierung ist der Klassifikator eindeutig.