

Perceptron

Vorlesung 186.844

20.11.2014



TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology



Grundgedanke

... dieser Vorlesungseinheit

Bisher:

nicht-parametrische Verfahren	parametrische Verfahren
<ul style="list-style-type: none">• Anzahl der Parameter nicht fix• keine Annahme über die Verteilung	<ul style="list-style-type: none">• Anzahl der Parameter ist fix• man macht am Beginn eine Annahme über die Verteilung (z.B. Normalverteilung)

Jetzt: Annahme über die Entscheidungsgrenze (lineare Diskriminantenfunktion) → parametrisches Verfahren

Überblick

- I. Wiederholung (WH): Diskriminantenfunktionen
- II. Perceptron
- III. Perceptron-Algorithmus
- IV. Perceptron als Neuron
- V. Generalisierungsfähigkeit des Perceptron

Wiederholung: Diskriminanten- funktionen

WH: Diskriminatenfunktion

Ein Klassifikator kann als eine **Gruppe von Diskriminantenfunktionen** $g_i(\mathbf{x}), i = 1, \dots, c$ dargestellt werden. Der Klassifikator weist den Merkmalsvektor \mathbf{x} der Klasse w_i zu, wenn gilt: $g_i(\mathbf{x}) > g_j(\mathbf{x})$ für alle $j \neq i$.

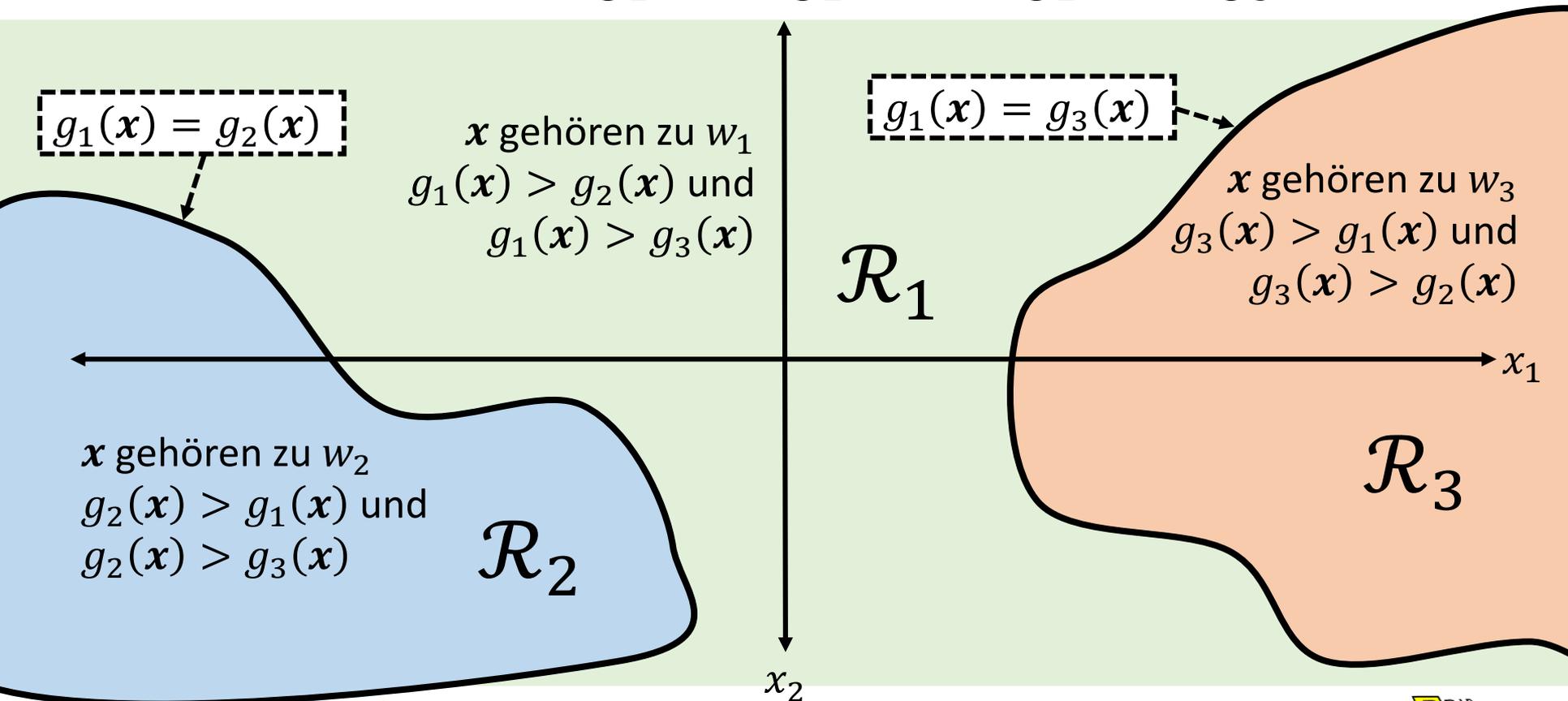


Im Fall von 2 Klassen ist es üblich **nur eine Diskriminantenfunktion** zu definieren $g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x})$. Diese Funktion entscheidet für Klasse w_1 wenn $g(\mathbf{x}) > 0$; ansonsten für Klasse w_2 .



WH: Beispiel

- Entscheidungsregionen: \mathcal{R}_1 , \mathcal{R}_2 und \mathcal{R}_3
- Entscheidungsgrenzen: $g_1(\mathbf{x}) = g_2(\mathbf{x})$ und $g_1(\mathbf{x}) = g_3(\mathbf{x})$



WH: Diskriminantenfunktionen

... für multivariate Normalverteilungen unter Verwendung eines **Bayes-Klassifikators**

- durch Logarithmieren der Posteriors

$$g_j(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{p(\mathbf{x})}$$

- erhält man die folgenden (optimalen) Diskriminantenfunktionen

$$g_j(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x} - \boldsymbol{\mu}_j) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_j| + \ln P(\omega_j) - \frac{p}{2} \ln 2\pi - \ln p(\mathbf{x})$$

WH: Vereinfachung

$$g_j(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x} - \boldsymbol{\mu}_j) - \frac{1}{2} \ln|\boldsymbol{\Sigma}_j| + \ln P(\omega_j) - \cancel{\frac{p}{2} \ln 2\pi - \ln p(\mathbf{x})}$$

- die letzten beiden Terme können weggelassen werden, weil sie unabhängig von den Klassen ω_j sind
- Diskriminantenfunktionen vereinfachen sich zu

$$g_j(\mathbf{x}) = -\frac{1}{2} d_j^2(\mathbf{x}) + \left[-\frac{1}{2} \ln|\boldsymbol{\Sigma}_j| + \ln P(\omega_j) \right]$$

Mahalanobisdistanz

Determinante von $\boldsymbol{\Sigma}$

Prior

WH: Spezialfall 1

$$\Sigma_i = \sigma^2 I$$

- Merkmale sind unabhängig $\sigma_{ij} = 0$ für alle $i \neq j$
- alle Merkmale haben dieselbe Varianz $\sigma_{ii} = \sigma^2$

$$g_j(\mathbf{x}) = -\frac{1}{2} d_j^2(\mathbf{x}) + \left[-\frac{1}{2} \ln |\Sigma_j| + \ln P(\omega_j) \right]$$



$$g_j(\mathbf{x}) = \left(-\frac{1}{2} \right) \left(\frac{1}{\sigma^2} \right) (\mathbf{x} - \boldsymbol{\mu}_j)^T (\mathbf{x} - \boldsymbol{\mu}_j) - \frac{1}{2} \ln |\Sigma_j| + \ln P(\omega_j)$$



$$= \frac{1}{2\sigma^2} (\cancel{\mathbf{x}^T \mathbf{x}} - 2\boldsymbol{\mu}_j^T \mathbf{x} + \boldsymbol{\mu}_j^T \boldsymbol{\mu}_j) - \frac{1}{2} \ln |\Sigma_j| + \ln P(\omega_j)$$

... für alle Klassen gleich

WH: Spezialfall 1

... man erhält die äquivalente lineare Diskriminantenfunktion

$$g_j(\mathbf{x}) = \frac{1}{2\sigma^2} (-2\boldsymbol{\mu}_j^T \mathbf{x} + \boldsymbol{\mu}_j^T \boldsymbol{\mu}_j) + \ln P(\omega_j)$$

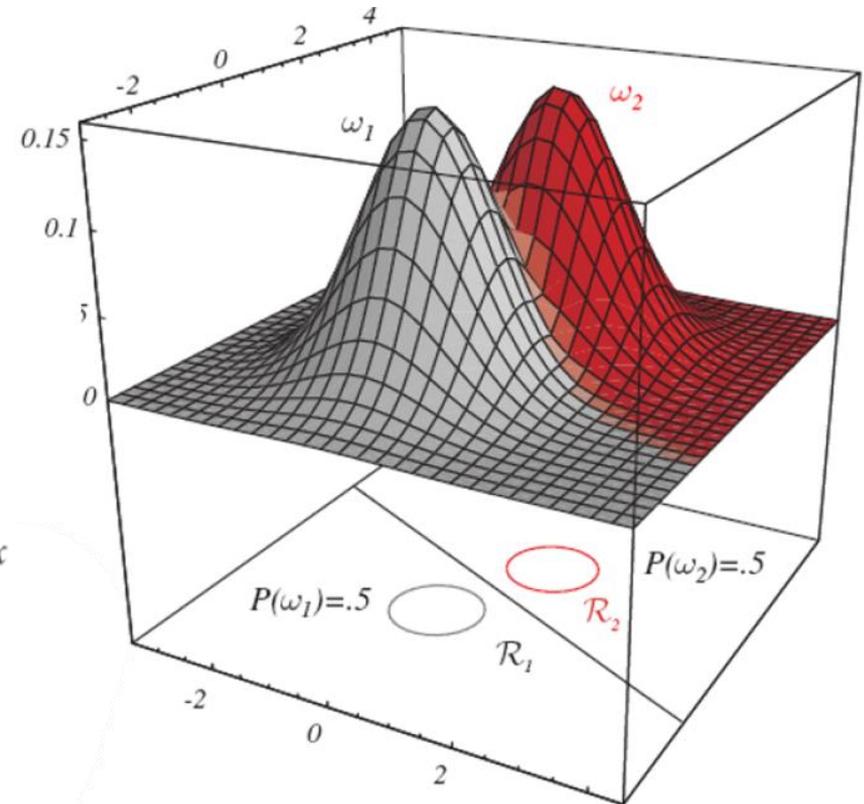
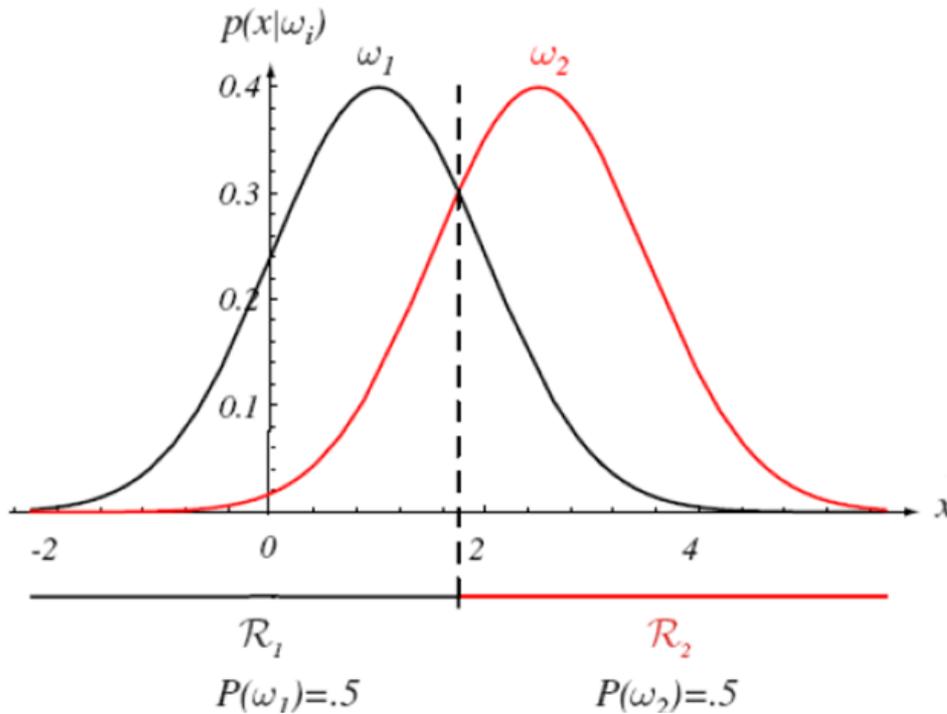


$$g_j(\mathbf{x}) = \mathbf{w}_j^T \mathbf{x} + w_{j0}$$

$$\mathbf{w}_j = \frac{1}{\sigma^2} \boldsymbol{\mu}_j$$

$$w_{j0} = -\frac{1}{2\sigma^2} \boldsymbol{\mu}_j^T \boldsymbol{\mu}_j + \ln P(\omega_j)$$

WH: Spezialfall 1



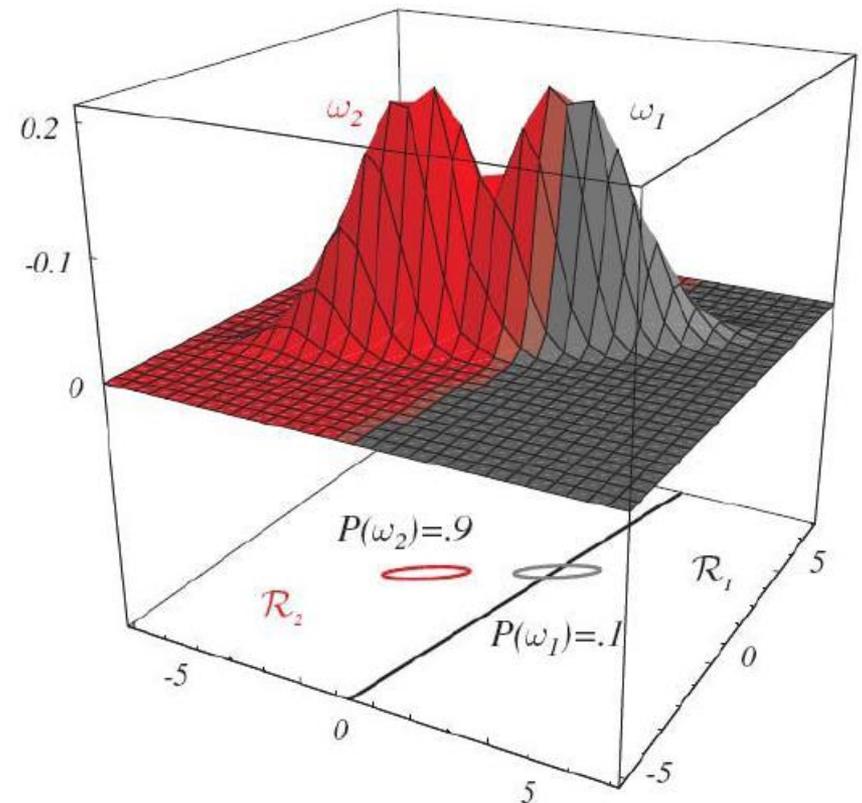
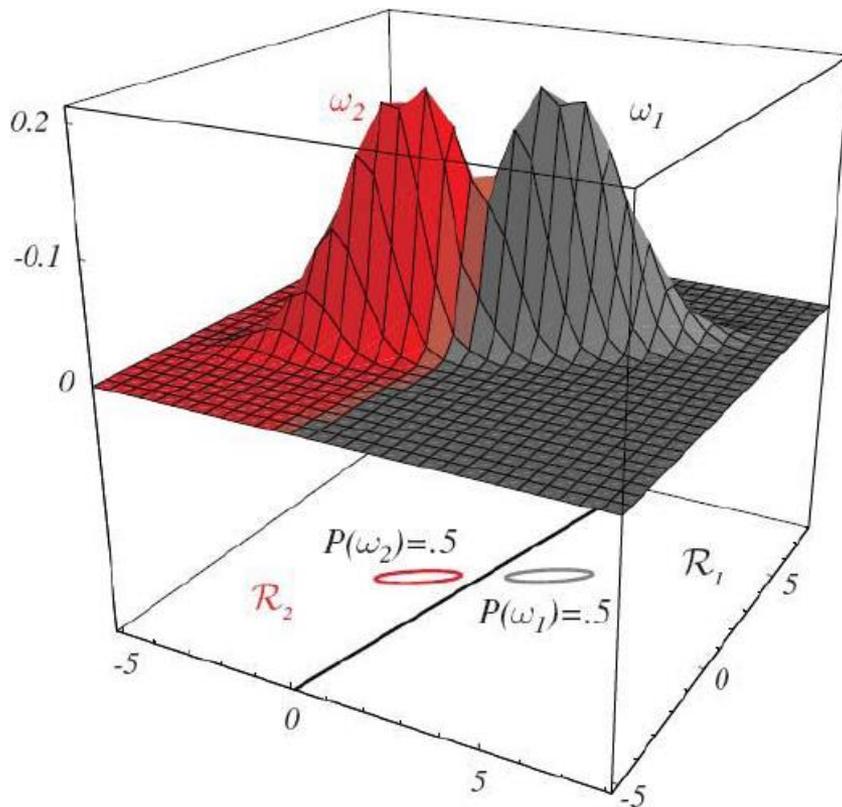
- univariate (links) bzw. bivariate (rechts) Normalverteilungen mit $\Sigma_1 = \Sigma_2 = \sigma^2$
- Entscheidungsgrenzen sind **linear** und **normal zur Verbindungsstrecke zwischen den beiden Klassenmitteln**
- bei gleichen Priors verläuft Entscheidungsgrenze durch $(\mu_1 + \mu_2)/2$
- ansonsten wird sie von Prior der wahrscheinlicheren Klasse wegverschoben.

WH: Spezialfall 2

$$\Sigma_i = \Sigma$$

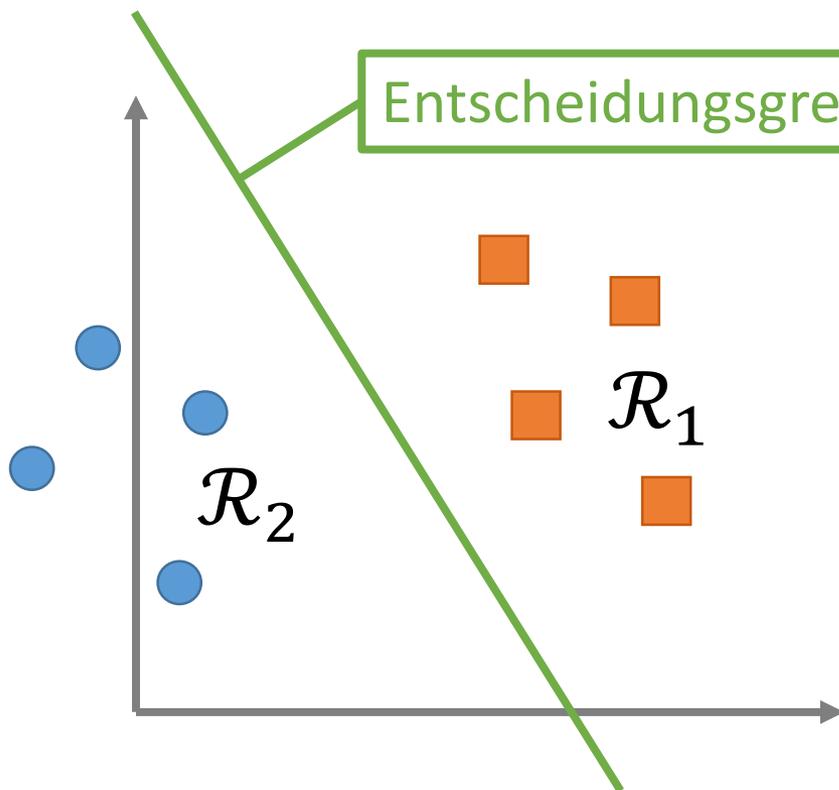
- alle Klassen haben dieselbe Kovarianzmatrix
- Form der Verteilungen ist durch Hyperellipsoide in \mathcal{R}^p gegeben
- Entscheidungsgrenzen sind wieder **linear**
- jedoch **nicht normal** zur Verbindungsstrecke zwischen den beiden Klassenmitteln
- Details: siehe Kapitel 2, Duda et al., 2001

WH: Spezialfall 2



Eigenschaften von

... linearen Diskriminantenfunktionen



- können optimal sein, wenn die Datenverteilungen geeignet sind
- schnell und einfach zu trainieren
- geeignet für einen ersten „Probe-Klassifikator“

Perceptron

Perceptron

Das **Perceptron** ist ein Trainingsalgorithmus, um für binäre Klassifikationsprobleme (zwei Klassen) eine lineare Entscheidungsgrenze zu finden.



- Annahmen: d -dimensionaler Merkmalsvektor $\mathbf{x} \in \mathbb{R}^d$ und zwei Klassen w_1 und $w_2 \rightarrow$ binäre Klassifikationsproblem
- Ziel: Finde eine Diskriminantenfunktion $g: \mathbb{R}^d \rightarrow \mathbb{R}$ welche die Klassenzugehörigkeit wie folgt kodiert:

$$\begin{aligned} g(\mathbf{x}) &> 0 \text{ falls } \mathbf{x} \in \omega_1 \\ g(\mathbf{x}) &< 0 \text{ falls } \mathbf{x} \in \omega_2 \end{aligned}$$

Perceptron

Der **Absolutbetrag der Diskriminantenfunktion** $|g|$ ist ein Maß für das Vertrauen in die geschätzte Klassenzugehörigkeit eines Merkmalsvektors \mathbf{x} .



- eine lineare Diskriminantenfunktion hat folgende Form

$$g(\mathbf{x}) = \sum_{i=1}^d w_i x_i - \theta = \mathbf{w}^T \mathbf{x} - \theta$$

$\mathbf{x} \in \mathbb{R}^d$... Merkmalsvektor

$\mathbf{w} \in \mathbb{R}^d$... Gewichtsvektor

$\theta \in \mathbb{R}$... Bias / Schwellwert (Threshold)

Geschichte

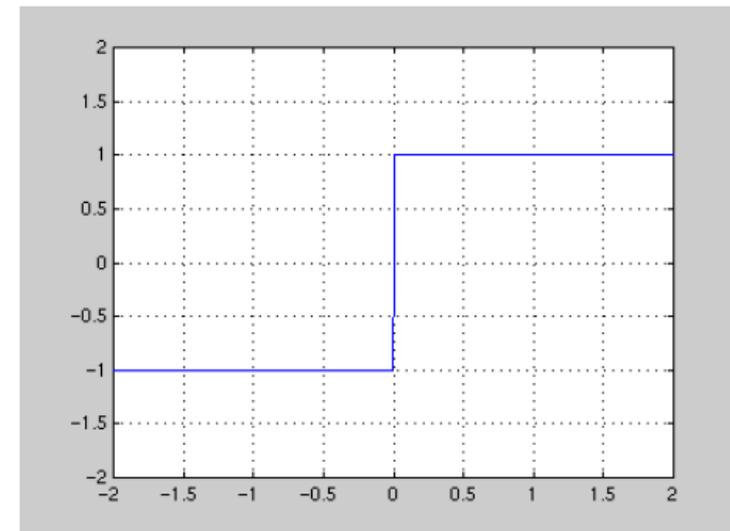
- Perceptron-Algorithmus wurde 1957 von Frank Rosenblatt am Cornell Aeronautical Laboratorium erfunden
 - Rosenblatt, Frank (1957), The Perceptron – a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.



Die **Architektur** eines Perceptrons entspricht einer linearen Diskriminantenfunktion mit nachgeschalteter Signumfunktion.



$$\text{sgn}(x) = \begin{cases} 1 & \text{falls } x \geq 0 \\ -1 & \text{falls } x < 0 \end{cases}$$



Signumfunktion

- man kann eine lineare Diskriminantenfunktion auch als Signumfunktion anschreiben:

$$o = o(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x} - \theta) = \begin{cases} 1 & \text{falls } \mathbf{w}^T \mathbf{x} \geq \theta \\ -1 & \text{falls } \mathbf{w}^T \mathbf{x} < \theta \end{cases}$$

$o(\mathbf{x})$... ist die Ausgabe des Perceptrons

Ziel des Perceptron-Algorithmus: Bestimme einen Gewichtsvektor \mathbf{w} und einen Bias θ , sodass

$$\begin{aligned} o(\mathbf{x}) &= 1 && (\Leftrightarrow \mathbf{w}^T \mathbf{x} \geq \theta) && \text{falls } \mathbf{x} \in \omega_1 \\ o(\mathbf{x}) &= -1 && (\Leftrightarrow \mathbf{w}^T \mathbf{x} < \theta) && \text{falls } \mathbf{x} \in \omega_2 \end{aligned}$$

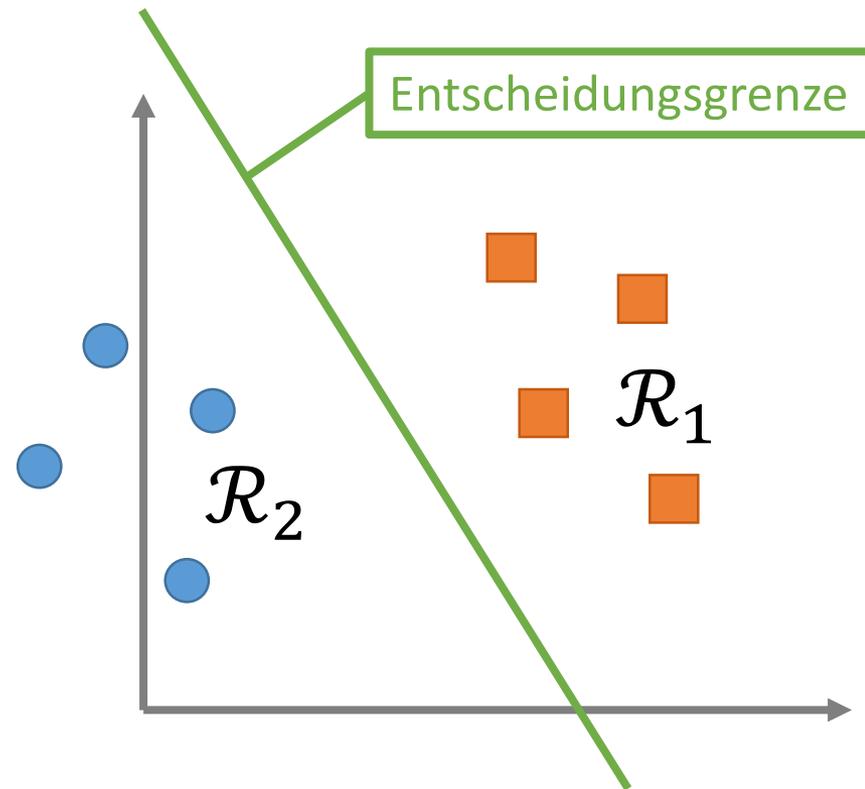
Geometrische Interpretation

- für $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$, legt

$$\sum_{i=1}^d w_i x_i = \mathbf{w}^T \mathbf{x} = \theta$$

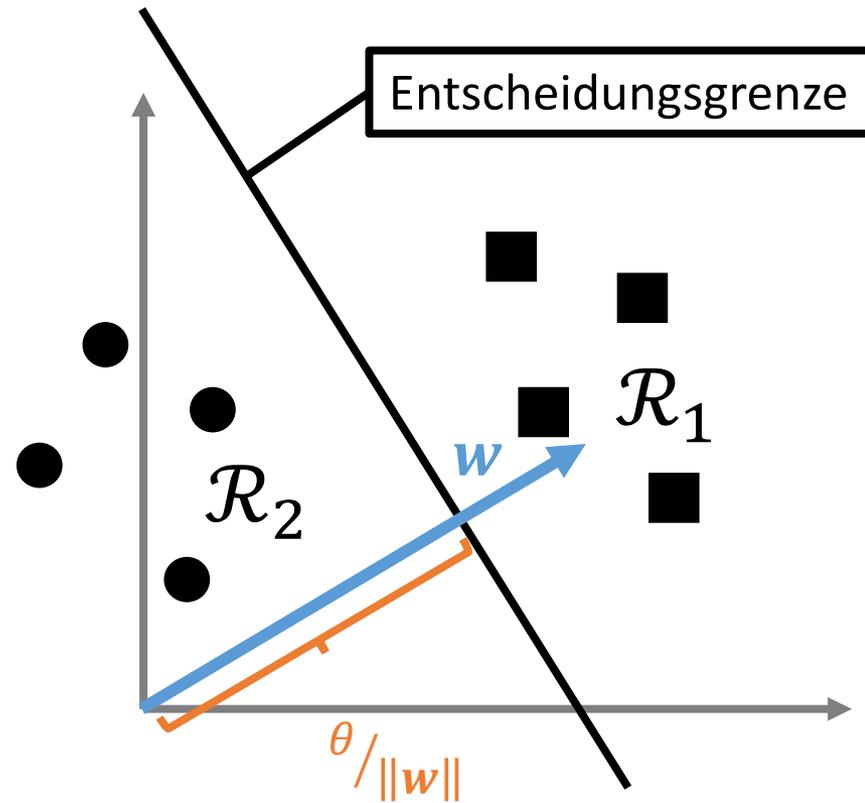
- eine in \mathbb{R}^d eingebettet ($d - 1$)-dimensionale Hyperebene fest

- für \mathbb{R}^2 eine Gerade (1D)
- für \mathbb{R}^3 eine Ebene (2D)
- ...



Geometrische Interpretation

- falls $\theta = 0$, dann geht die Hyperebene durch den Ursprung
- ansonsten, ist sie entlang \mathbf{w} um den Betrag $\theta / \|\mathbf{w}\|$ vom Ursprung verschoben

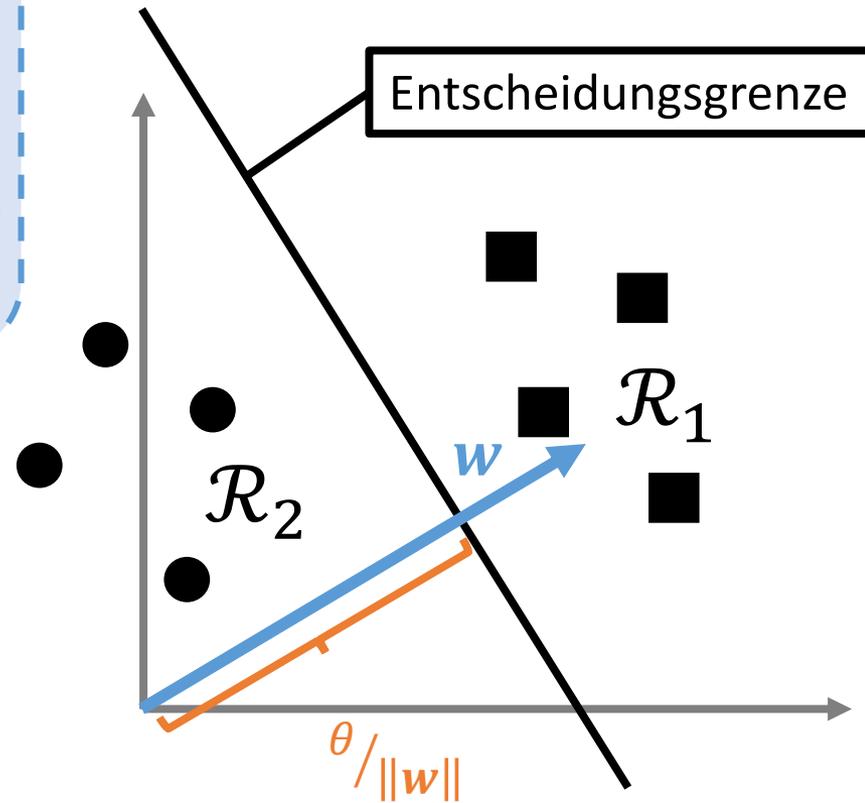


Geometrische Interpretation

Die **Entscheidungsgrenze** (Hyper-ebene) $\mathbf{w}^T \mathbf{x} = \theta$ kann durch ihren **Normalvektor** \mathbf{w} und ihre **Distanz von Ursprung** $\theta / \|\mathbf{w}\|$, gemessen entlang \mathbf{w} , beschrieben werden. 

im Beispiel links:

- falls $\|\mathbf{w}\| = 1$, dann ist $\theta / \|\mathbf{w}\| = \theta$
- für alle quadratischen Datenpunkte $\in \omega_1$ gilt $\mathbf{w}^T \mathbf{x} \geq \theta$ und
- für alle runden Datenpunkte $\in \omega_2$ gilt $\mathbf{w}^T \mathbf{x} < \theta$



Lineare Separierbarkeit

- im Englischen „linear separability“
- $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \in \mathbb{R}^{d \times N}$... N Merkmalsvektoren der Länge d
- $\mathbf{y} = (y_1, y_2, \dots, y_N), y_i \in \{1, -1\}$... N Klassenlabels

\mathbf{X} ist genau dann **linear separierbar** (bezüglich \mathbf{y}), falls es einen Gewichtsvektor \mathbf{w} und einen Bias θ gibt, sodass

$$o(\mathbf{x}_i) = \text{sgn}(\mathbf{w}^T \mathbf{x}_i - \theta) = y_i, 1 \leq i \leq N$$



Homogene Koordinaten

... für eine kompaktere Schreibweise

- Bias kann durch einen kleinen Kunstgriff **in den Gewichtsvektor „hineingezogen“** werden
- Dazu erweitert man den Merkmalsvektor \mathbf{x} und den Gewichtsvektor \mathbf{w} um eine Dimension:
 - $x_0 = 1$ und $w_0 = -\theta$

$$\check{\mathbf{x}} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad \check{\mathbf{w}} = \begin{bmatrix} -\theta \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$$

Homogene Koordinaten

Dadurch erhält man:

$$g(\mathbf{x}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}} = \sum_{i=1}^d w_i x_i - \theta = \mathbf{w}^T \mathbf{x} - \theta$$

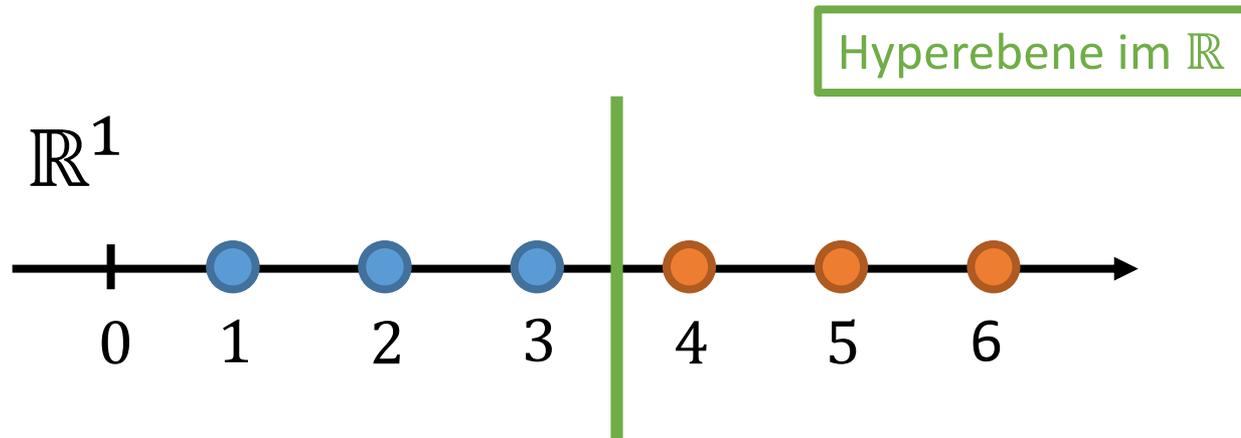
- Transformation in homogene Koordinaten vereinfacht Problem indem **Dimensionalität um 1** (von d auf $d + 1$) **erhöht** wird
- obige Gleichung definiert eine **d -dimensionale Hyperebene** im \mathbb{R}^{d+1} , **welche durch den Ursprung geht**
- im folgenden werden stets homogene Koordinaten verwendet \rightarrow daher schreibt man einfach **$\mathbf{w}^T \mathbf{x}$ für $\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}$**

Beispiel

- zur Motivation der homogenen Koordinaten

Trainingsdaten Klasse w_1 : {1, 2, 3}

Trainingsdaten Klasse w_2 : {4, 5, 6}



Beispiel

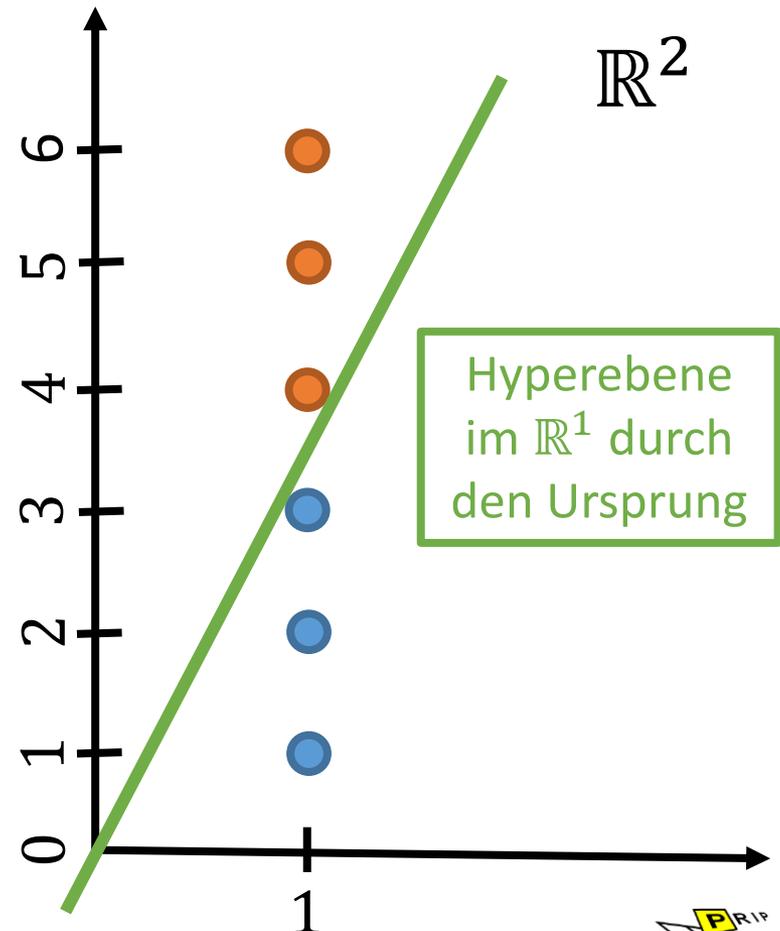
- Transformation in homogene Koordinaten

Trainingsdaten Klasse w_1 :

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \end{pmatrix} \right\}$$

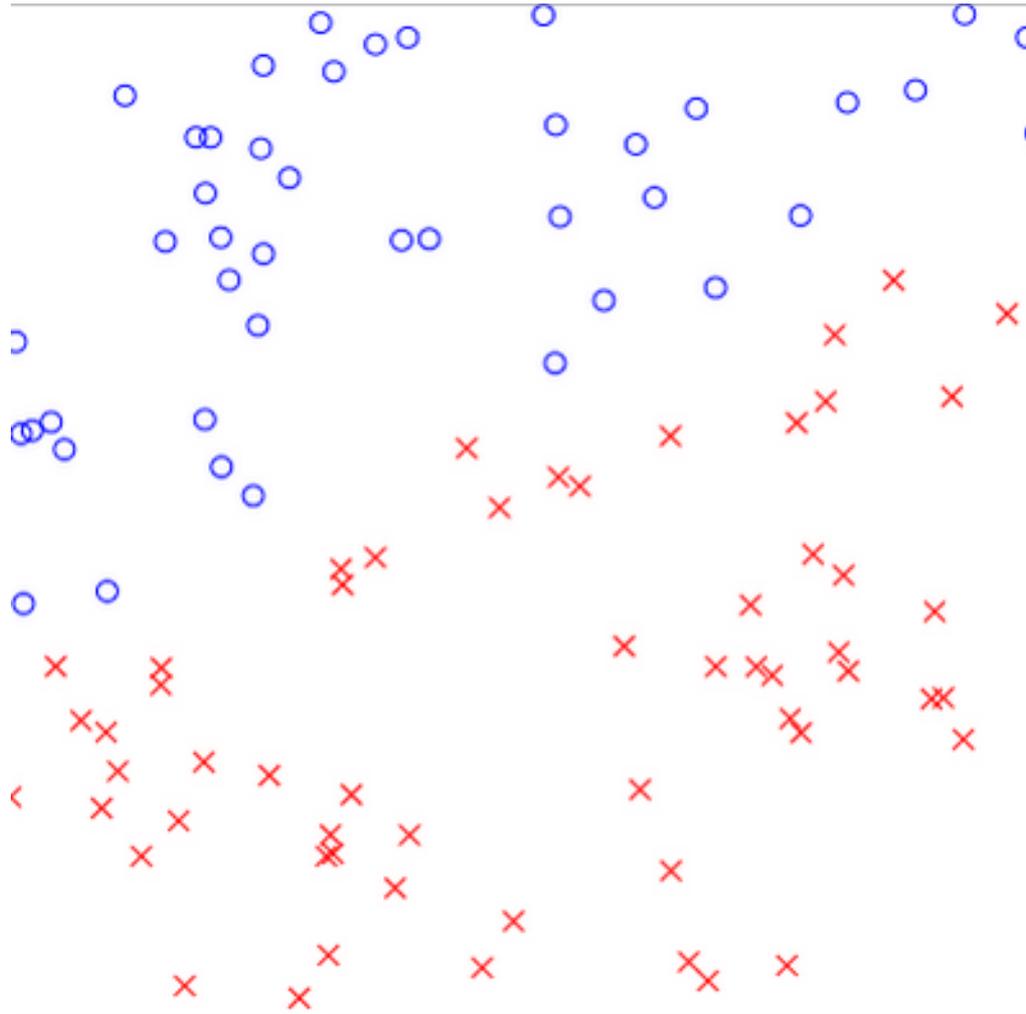
Trainingsdaten Klasse w_2 :

$$\left\{ \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \begin{pmatrix} 1 \\ 5 \end{pmatrix}, \begin{pmatrix} 1 \\ 6 \end{pmatrix} \right\}$$



Perceptron- Algorithmus

DEMO



Trainingsdatensatz

Trainingsdatensatz:

$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \in \mathbb{R}^{(d+1) \times N}$... N homogene Merkmalsvektoren

$\mathbf{y} = (y_1, y_2, \dots, y_N), y_i \in \{1, -1\}$... N Klassenlabels

Beispiel:

Wenn man das binäre AND-Problem mit Hilfe eines Perceptrons lösen wollte, hätte der Trainingsdatensatz folgende Form:

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \mathbf{y} = (-1, -1, -1, 1)$$

Grundidee

Ziel: Finde einen homogenen Gewichtsvektor \mathbf{w} , sodass

$$o(\mathbf{x}_i) = \text{sgn}(\mathbf{w}^T \mathbf{x}_i) = y_i, 1 \leq i \leq N$$

Idee:

- falls ein „positiver“ Trainingsvektor \mathbf{x}_i (mit $y_i = 1$) falsch klassifiziert wird, d.h.: $\mathbf{w}^T \mathbf{x}_j < 0$
- dann addiere ein Vielfaches von \mathbf{x}_i zu \mathbf{w}
- dadurch wird die Hyperebene auf den falsch klassifizierten Vektor hinbewegt.

Lernrate

$$(\mathbf{w} + \gamma \mathbf{x}_i)^T \mathbf{x}_i > \mathbf{w}^T \mathbf{x}_i, \gamma > 0$$

- γ ... positiver Faktor \rightarrow wird auch Lernrate genannt
- wird verwendet um „Vielfaches“ von \mathbf{x}_i zu \mathbf{w} zu addieren

Analog zum positiven Fall (siehe oben):

- im Fall eines falsch klassifizierten „negativen“ Trainingsvektors \mathbf{x}_j (mit $y_j = -1$), d.h.: $\mathbf{w}^T \mathbf{x}_j > 0$
- wird ein Vielfaches von \mathbf{x}_j von \mathbf{w} subtrahiert
- dadurch wird die Hyperebene vom falsch klassifizierten Vektor wegbewegt

Iterativer Prozess

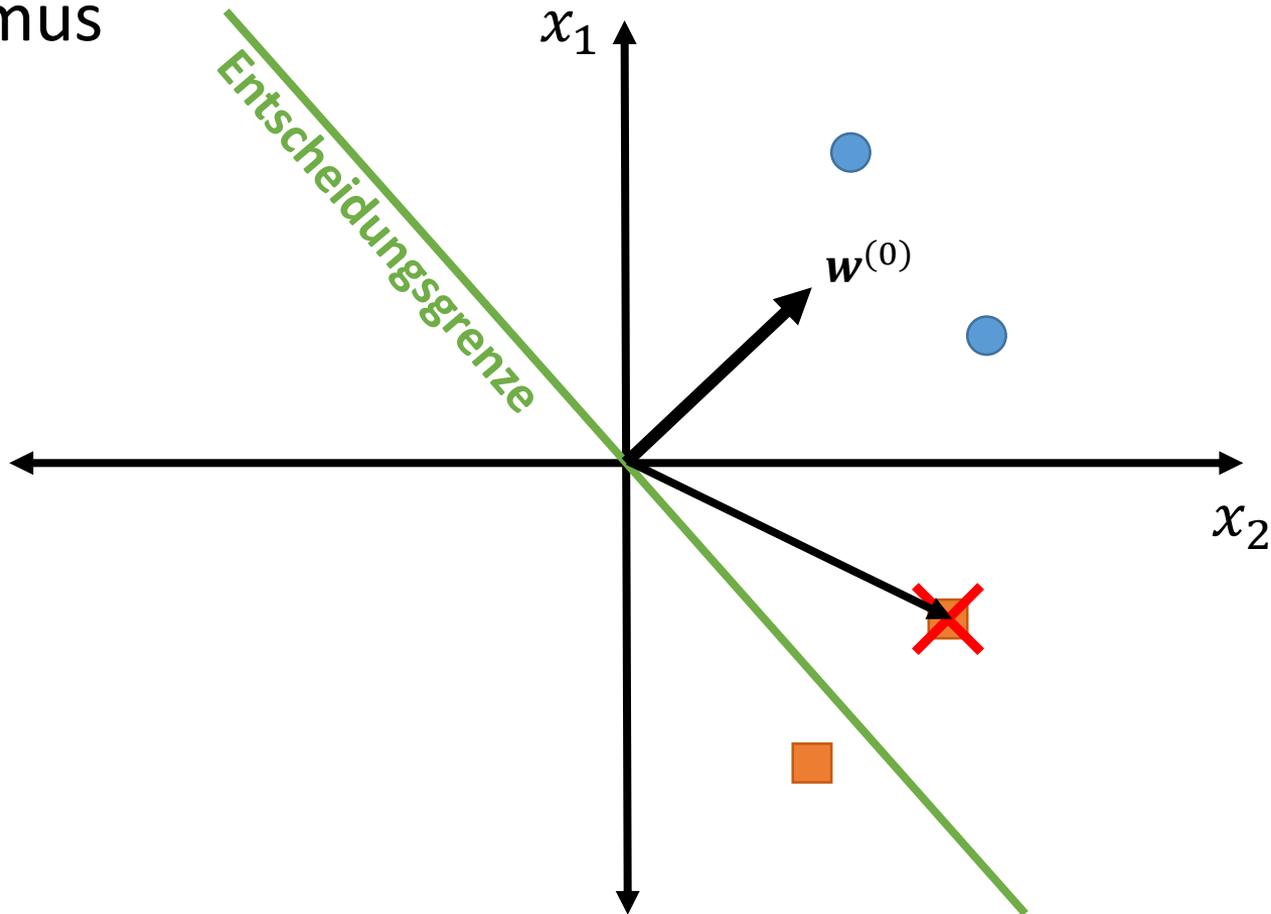
Der Perceptron-Algorithmus ist ein **iterativer Prozess** der die **Parameter** der linearen Diskriminantenfunktion, w und θ , aus einem Trainingsdatensatz bestimmt.



Während des iterativen Trainingsprozesses ist es möglich, dass zuvor richtig klassifizierte Vektoren durch die aktuelle Hyperebene falsch klassifiziert werden → siehe Beispiel auf den nächsten Folien

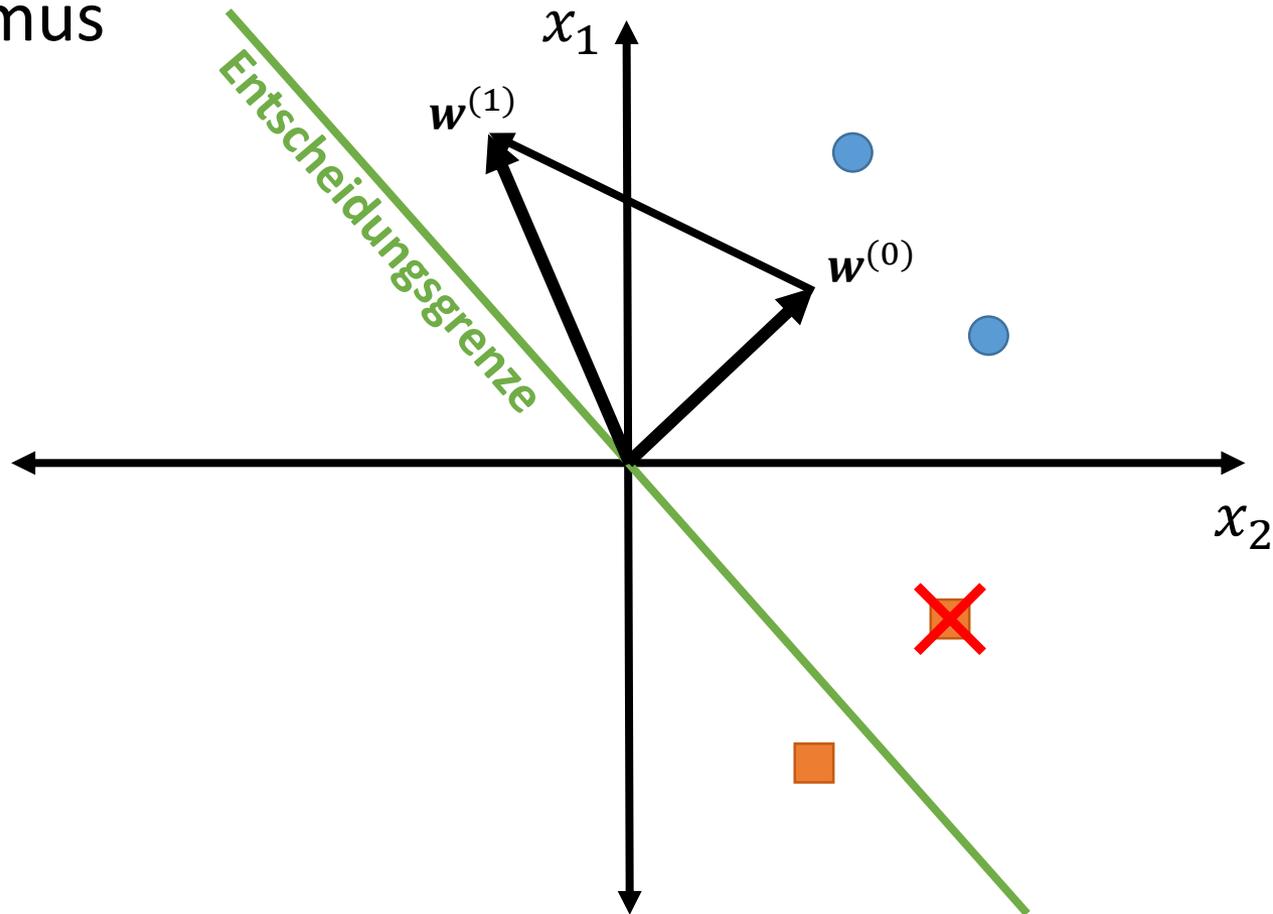
Beispiel

.. des iterativen Trainingsprozesses im Perceptron-Algorithmus



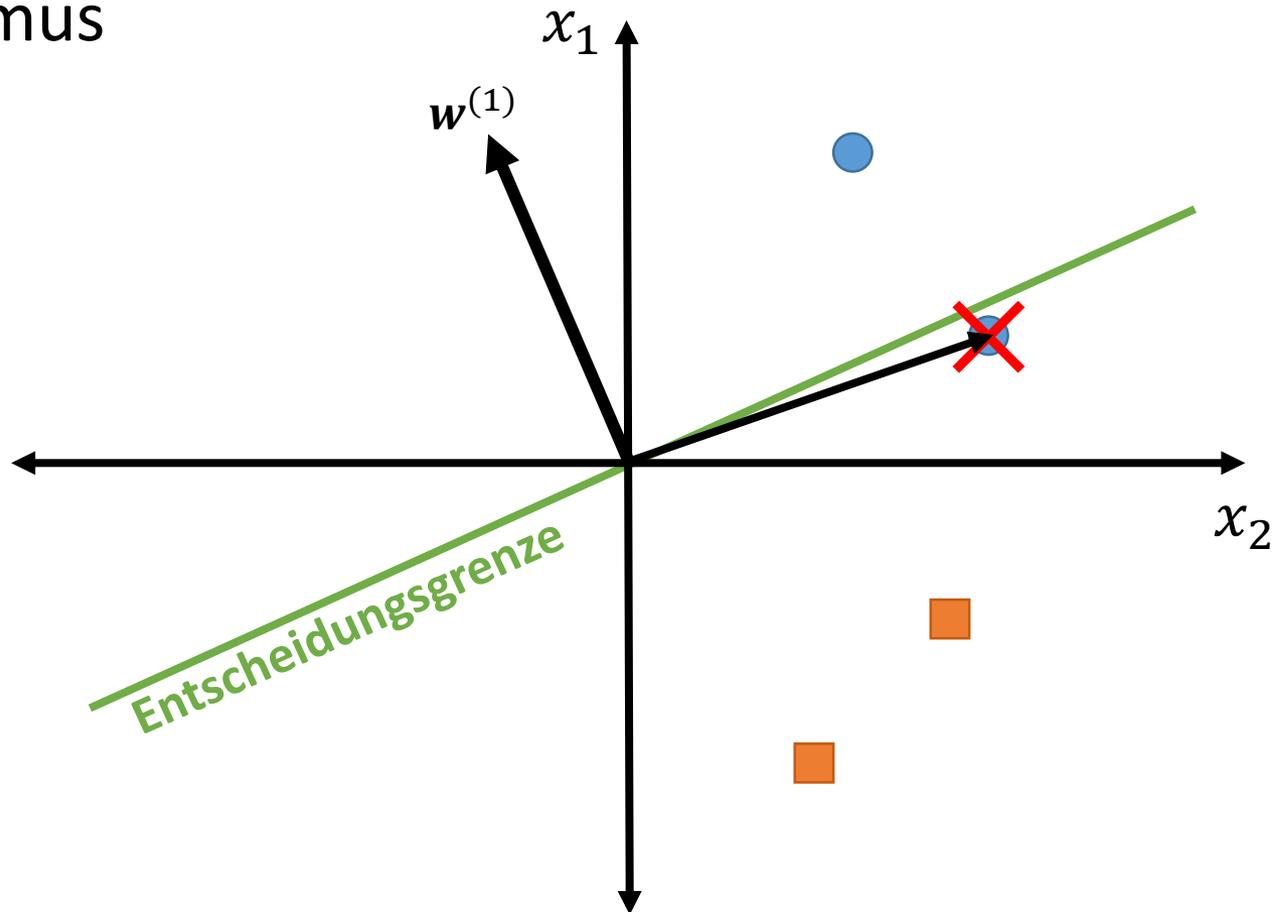
Beispiel

.. des iterativen Trainingsprozesses im Perceptron-Algorithmus



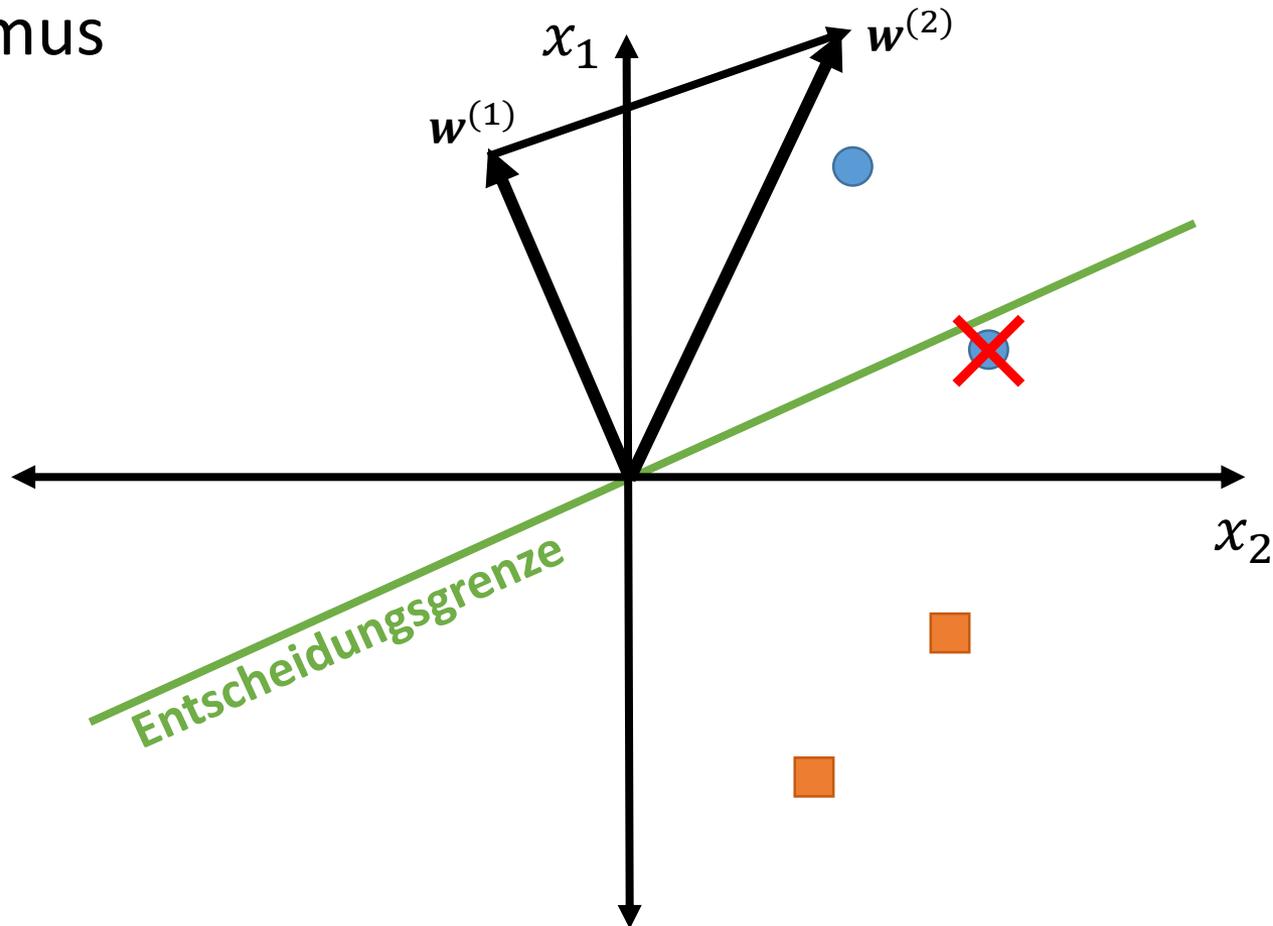
Beispiel

.. des iterativen Trainingsprozesses im Perceptron-Algorithmus



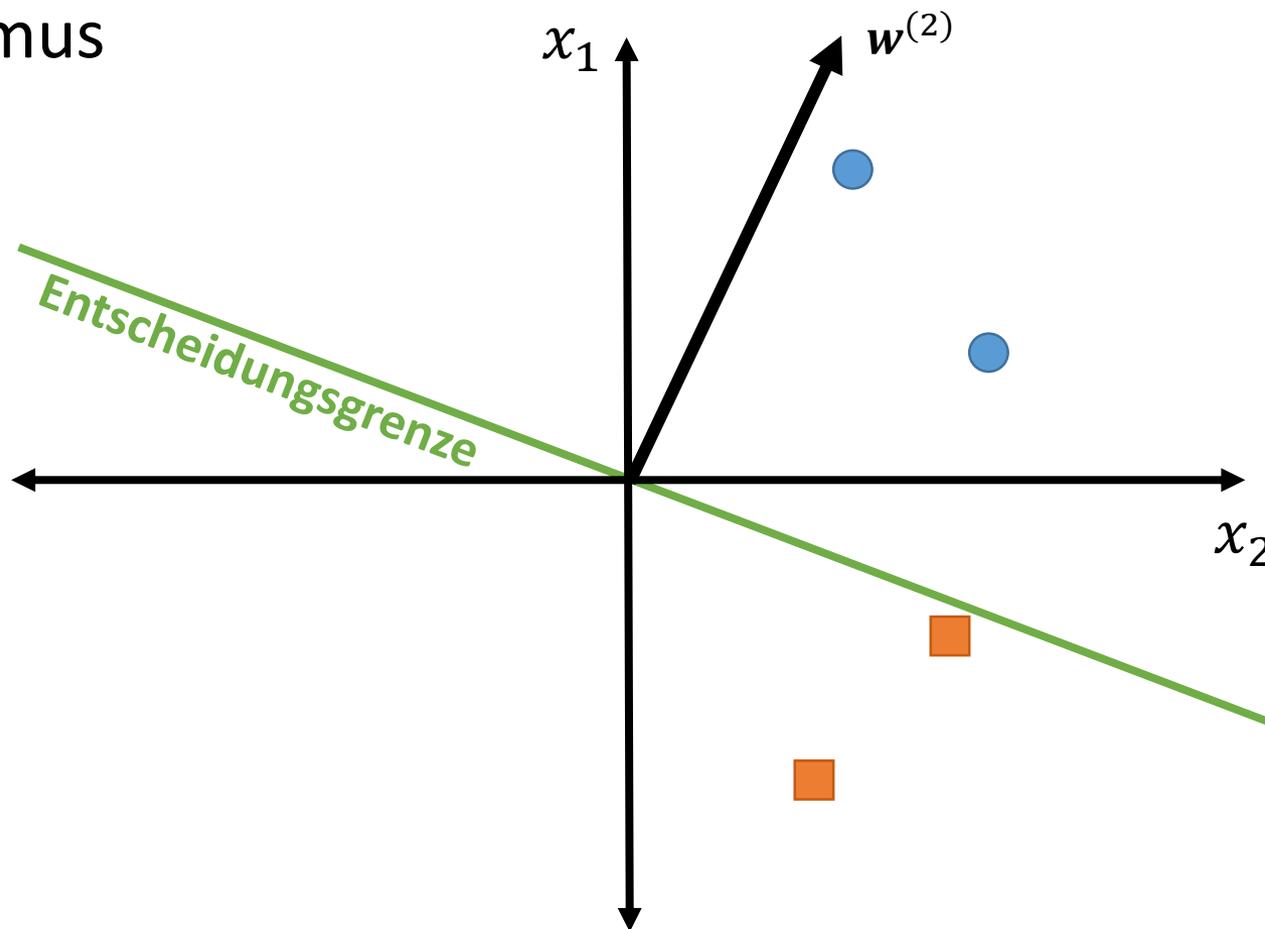
Beispiel

.. des iterativen Trainingsprozesses im Perceptron-Algorithmus



Beispiel

.. des iterativen Trainingsprozesses im Perceptron-Algorithmus



Vereinfachung

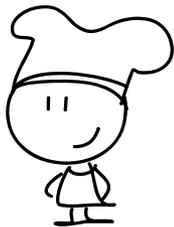
Falls man **unabhängig von der Klasse** („positive“ oder „negative“, 1 oder -1) immer die gleiche Operation durchführen will, dann kann man folgendes tun:

$$\begin{aligned} \text{sgn}(\mathbf{w}^T \mathbf{x}_i) = y_i &\Leftrightarrow \text{sgn}(\mathbf{w}^T \mathbf{x}_i) y_i = 1 \\ \Rightarrow (\mathbf{w}^T \mathbf{x}_i) y_i > 0 &\Leftrightarrow \mathbf{w}^T (\mathbf{x}_i y_i) > 0 \end{aligned}$$

- man sucht nun nach einem Gewichtsvektor \mathbf{w} der den modifizierten Trainingsdatensatz $\mathbf{x}_i y_i$, $1 \leq i \leq N$ (mit ausschließlich positiven Klassenlabels) in die positive Halbebene abbildet

Perceptron-Algorithmus

1. Initialisiere w und γ
2. **do**
3. **for** $i = 1$ to N
4. **if** $w^T(x_i y_i) \leq 0$ (falsche Klassifikation des Vektors x_i)
5. $w \leftarrow w + \gamma x_i y_i$
6. **end if**
7. **end for**
8. **until** \rightarrow alle Merkmalsvektoren x_i richtig klassifiziert werden



3. – 7. Schritt: **Epoche**
5. Schritt: **Gewichtsupdate**

Initialisierung: Gewichtsvektor

- mögliche Initialisierung des Gewichtsvektors: $\mathbf{w} = 0$
- dafür würde man mit dem Perceptron-Algorithmus folgenden Gewichtsvektor erhalten:

$$\mathbf{w}_p = \sum_{i=1}^N \mathbf{x}_i (y_i \gamma k_i) = \gamma \sum_{i=1}^N \mathbf{x}_i (y_i k_i), k_i \in \mathbb{N}_0$$

\mathbf{w}_p ... linear Kombination aller falsch klassifizierten Merkmalsvektoren \mathbf{x}_i

k_i ... gibt an wie oft \mathbf{x}_i falsch klassifiziert wurde

Initialisierung: Lernrate

$$\mathbf{w}^T \mathbf{x} = \theta \Leftrightarrow (\alpha \mathbf{w})^T \mathbf{x} = \alpha \theta \quad (\forall \mathbf{x} \in \mathbb{R}^d)$$

$$\mathbf{w}^T \mathbf{x} \geq \theta \Leftrightarrow (\alpha \mathbf{w})^T \mathbf{x} \geq \alpha \theta \quad (\forall \mathbf{x} \in \mathbb{R}^d)$$

Obige Gleichungen bedeuten: falls man \mathbf{w} und θ mit dem gleichen positiven Faktor $\alpha \in \mathbb{R}^+$ multipliziert, bleiben die Entscheidungsregionen unverändert.

Daraus folgt: γ ist nur ein Skalierungsfaktor \rightarrow hat keinen Einfluss auf die Entscheidungsgrenze

Einfachheitshalber setzt man $\gamma = 1$ für den Perceptron-Algorithmus

$\gamma = 1$ ist nicht für jedes Trainingsverfahren geeignet!



Konvergenztheorem

Das Konvergenztheorem besagt, dass der Perceptron-Algorithmus mit fixer Lernrate γ für jeden linear separierbaren Trainingsdatensatz mit der Lösung \mathbf{w}_p terminiert.



Details für Interessierte:

Pattern Recognition, Sergios Theodoridis, Seite 95 – 97

Pattern Classification, Richard Duda, Seite 229 – 232

[MITOpenCourseWare: Machine Learning von Prof. Tommi Jaakkola](#)

Der Perceptron-Algorithmus konvergiert nicht im Falle eines nicht linear separierbaren Trainingsdatensatzes.

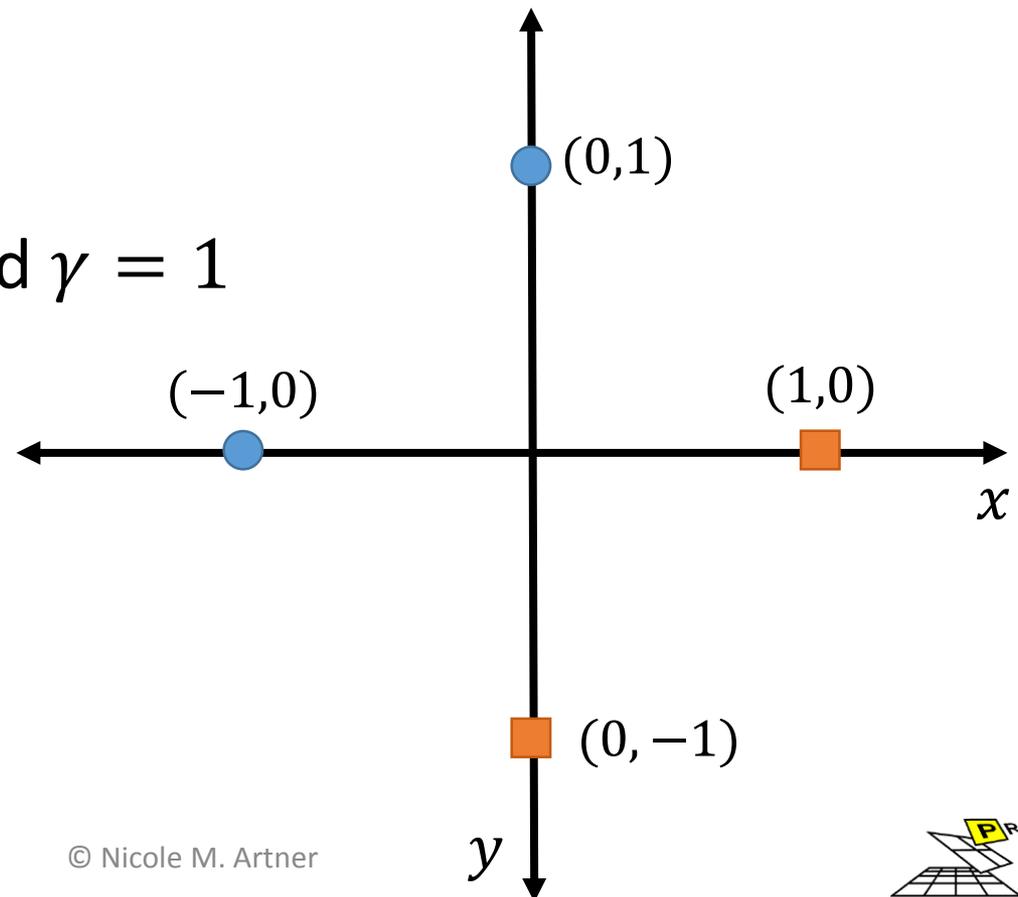


Beispiel

- Klasse $w_1 = \{(-1,0,1)^T, (0,1,1)^T\}$, $y_1 = 1$, $y_2 = 1$
- Klasse $w_2 = \{(0,-1,1)^T, (1,0,1)^T\}$, $y_3 = -1$, $y_4 = -1$

- $N = 4$

- $w = 0 = (0,0,0)^T$ und $\gamma = 1$



Beispiel

- für jeden Vektor \mathbf{x}_i überprüfe: $\mathbf{w}^T (\mathbf{x}_i y_i) \leq 0$
- und führe Update durch falls notwendig: $\mathbf{w} \leftarrow \mathbf{w} + \gamma \mathbf{x}_i y_i$

Beginn Epoche 1:

$$\mathbf{w}^{(0)T} (\mathbf{x}_1 y_1) = [0 \quad 0 \quad 0] \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \cdot 1 = 0$$

Update: $\mathbf{w}^{(1)} = \mathbf{w}^{(0)} + \gamma \mathbf{x}_1 y_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$

$$\mathbf{w}^{(1)T} (\mathbf{x}_2 y_2) = [-1 \quad 0 \quad 1] \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \cdot 1 = 1 > 0 \quad \checkmark$$

Beispiel

$$\mathbf{w}^{(1)T}(\mathbf{x}_3 y_3) = [-1 \quad 0 \quad 1] \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \cdot -1 = -1$$

Update: $\mathbf{w}^{(2)} = \mathbf{w}^{(1)} + \gamma \mathbf{x}_3 y_3 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \cdot -1 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$

$$\mathbf{w}^{(2)T}(\mathbf{x}_4 y_4) = [-1 \quad 1 \quad 0] \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \cdot -1 = 1 > 0$$



Ende Epoche 1

Beispiel

Beginn Epoche 2:

$$\mathbf{w}^{(2)T}(\mathbf{x}_1 y_1) = [-1 \quad 1 \quad 0] \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \cdot 1 = 1 > 0 \quad \checkmark$$

$$\mathbf{w}^{(2)T}(\mathbf{x}_2 y_2) = [-1 \quad 1 \quad 0] \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \cdot 1 = 1 > 0 \quad \checkmark$$

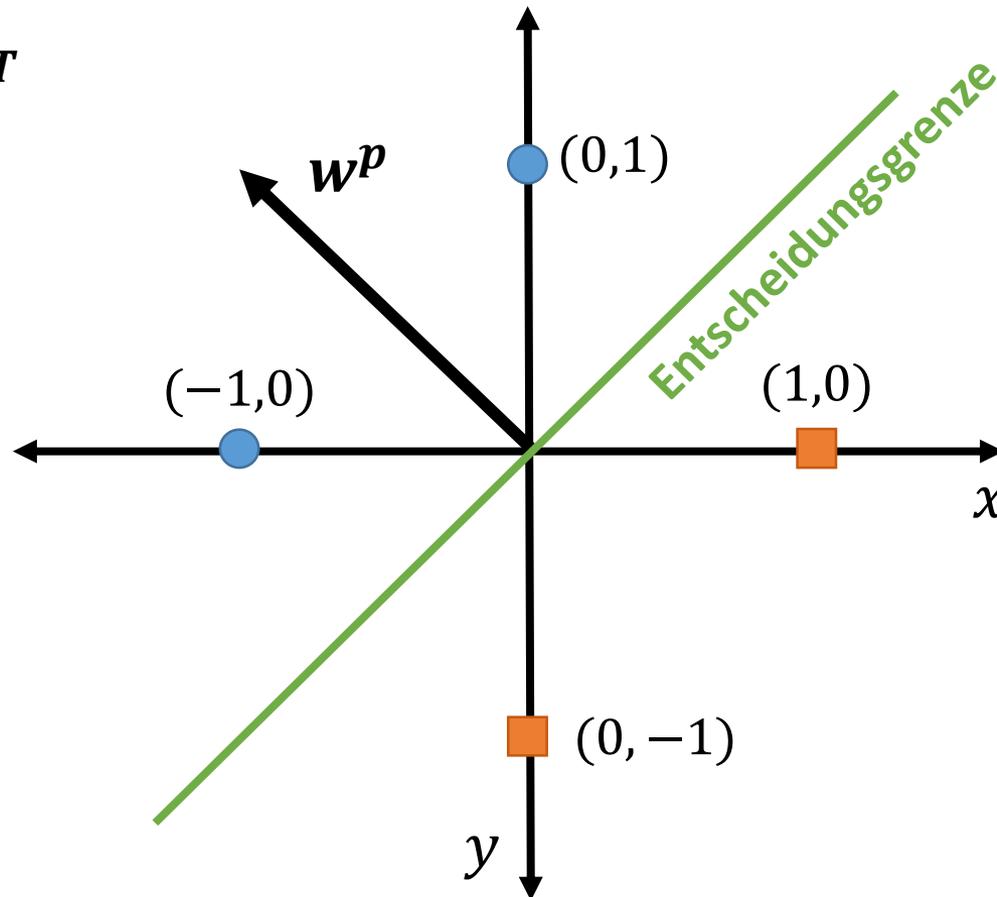
$$\mathbf{w}^{(2)T}(\mathbf{x}_3 y_3) = [-1 \quad 1 \quad 0] \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \cdot -1 = 1 > 0 \quad \checkmark$$

Ende Epoche 2 → Perceptron-Algorithmus konvergiert

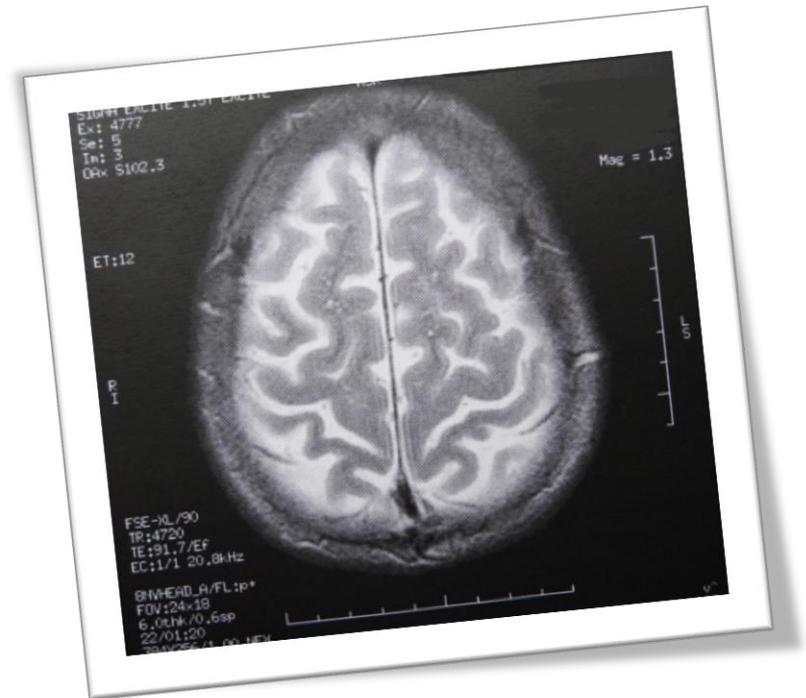
Beispiel

Ergebnis:

$$\mathbf{w}^p = (-1, 1)^T$$
$$\theta = 0$$



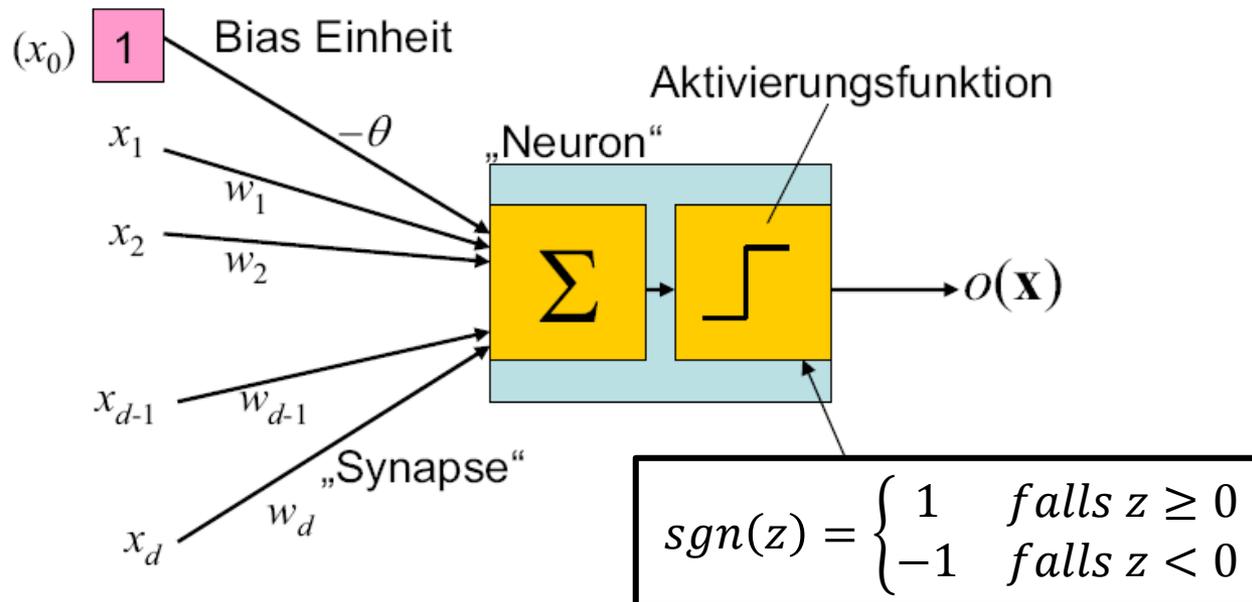
Perceptron als Neuron



Darstellung als Neuron

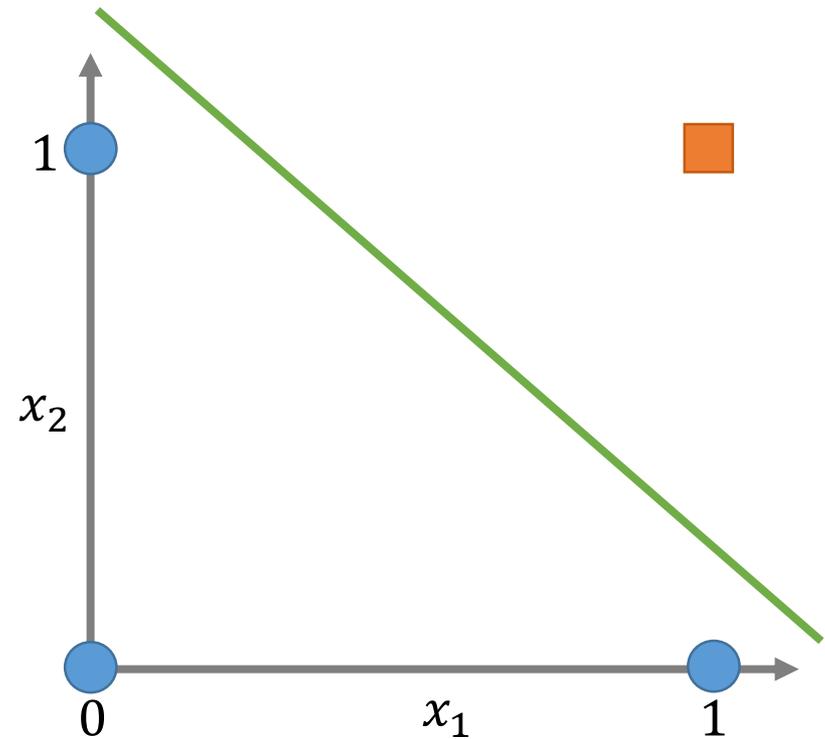
Erinnerung Signumfunktion: $o(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^d w_i x_i\right)$

Perceptron ist eine gute Annäherung an ein biologisches Neuron:



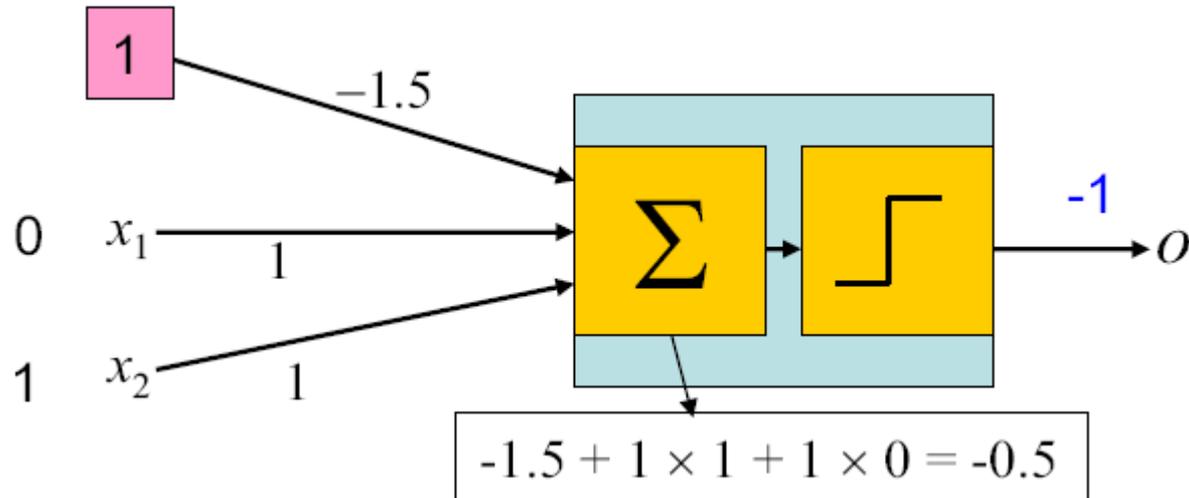
Beispiel: UND

x_1	x_2	$o(x)$
0	0	-1
0	1	-1
1	0	-1
1	1	1



Beispiel: UND

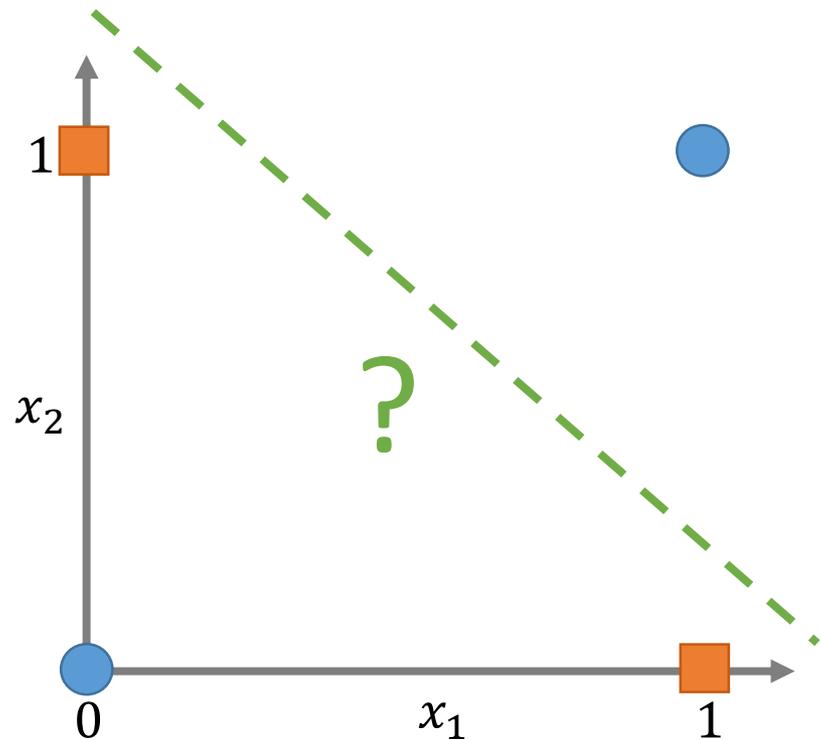
- das UND-Problem kann mit folgendem Perceptron gelöst werden:



Beispiel: XOR

Das XOR-Problem ist nicht linear separierbar!

x_1	x_2	$o(x)$
0	0	-1
0	1	1
1	0	1
1	1	-1



Wichtigster Nachteil

Ein **einstufiges Perceptron** kann nur linear separierbare Mengen, d.h. Mengen die durch eine Hyperebene trennbar sind, fehlerfrei klassifizieren.



Ein bisschen Geschichte ...

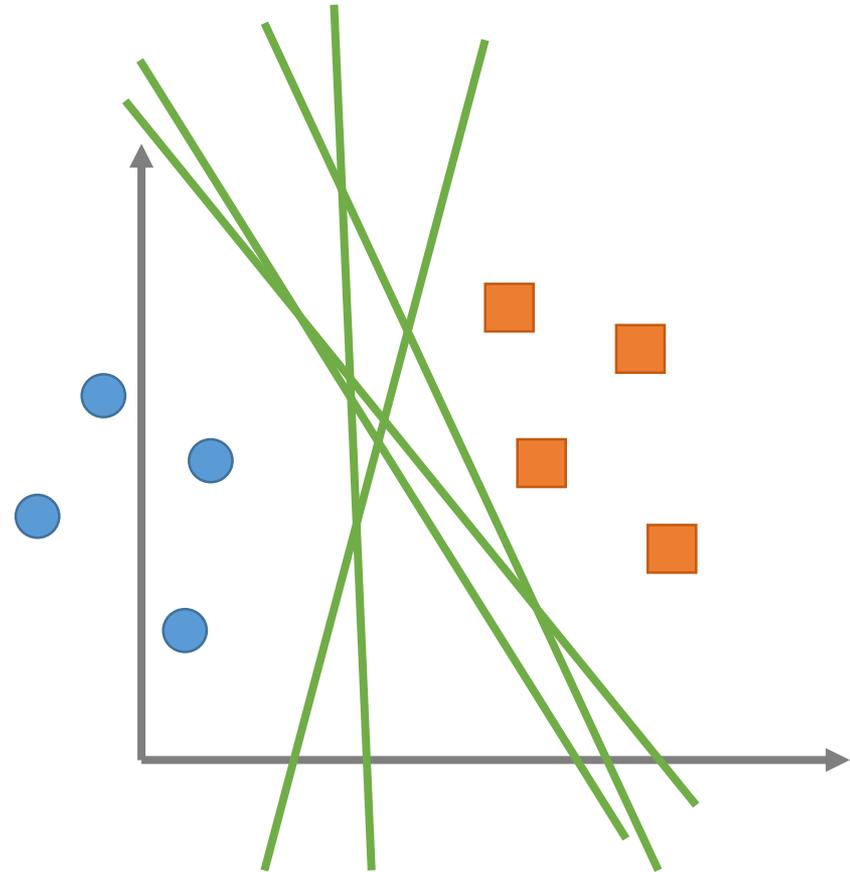
1969: M.L. **Minsky** und S.A. **Papert** zeigen unter anderem diesen Nachteil in ihrem **Buch mit dem Titel „Perceptrons“** auf. Nach der Veröffentlichung dieses Buches war der Enthusiasmus für Forschung auf dem Gebiet der Neuronalen Netze für etwa **15 Jahre gedämpft**.

Generalisierungsfähigkeit des Perceptrons

Lineare Diskriminantenfunktion

Es gibt theoretisch **unendlich viele lineare Entscheidungsgrenzen**, um dieses Klassifikationsproblem zu lösen.

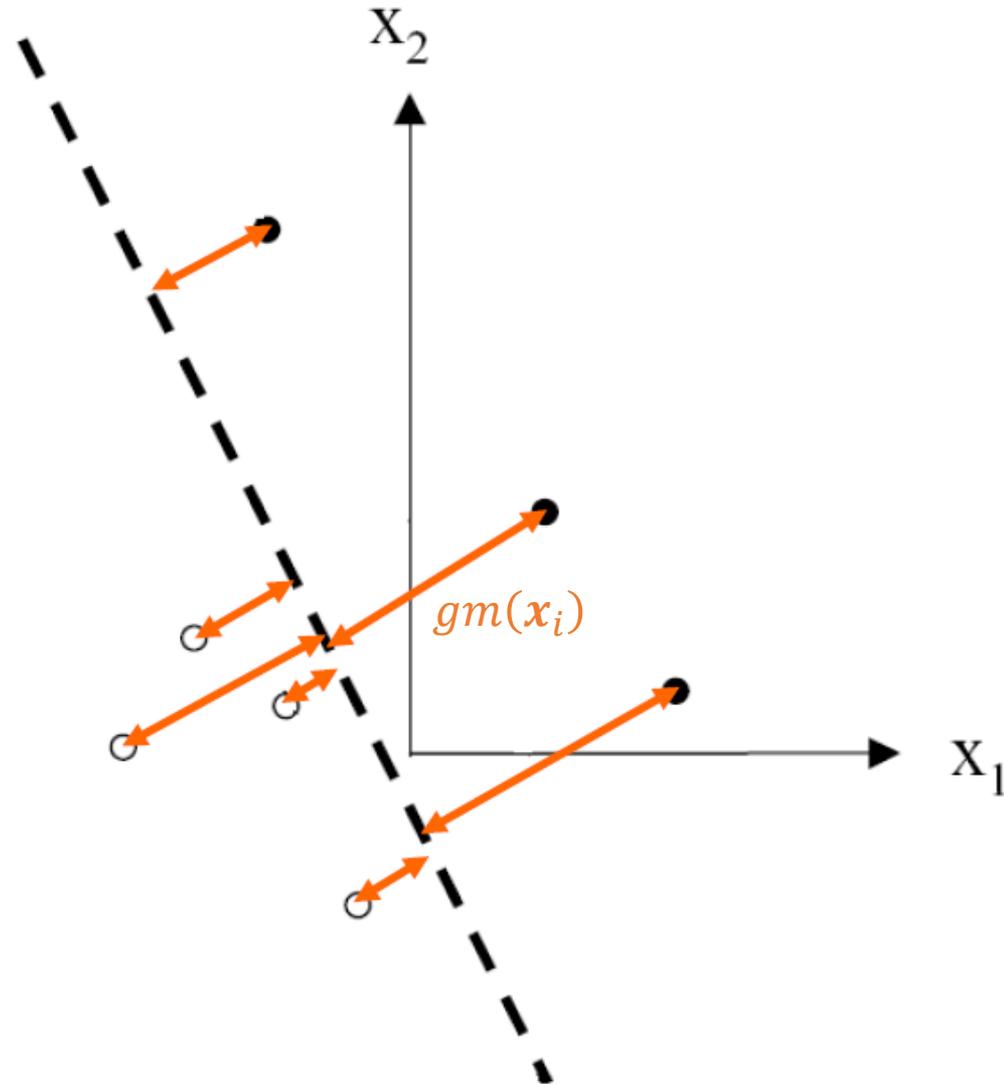
Welche ist die Beste?



Abstand

... Margin (im Englischen)

$gm(x_i)$...der geometrische Abstand (Margin) vom Vektor x_i zur Entscheidungsgrenze (Hyperebene)

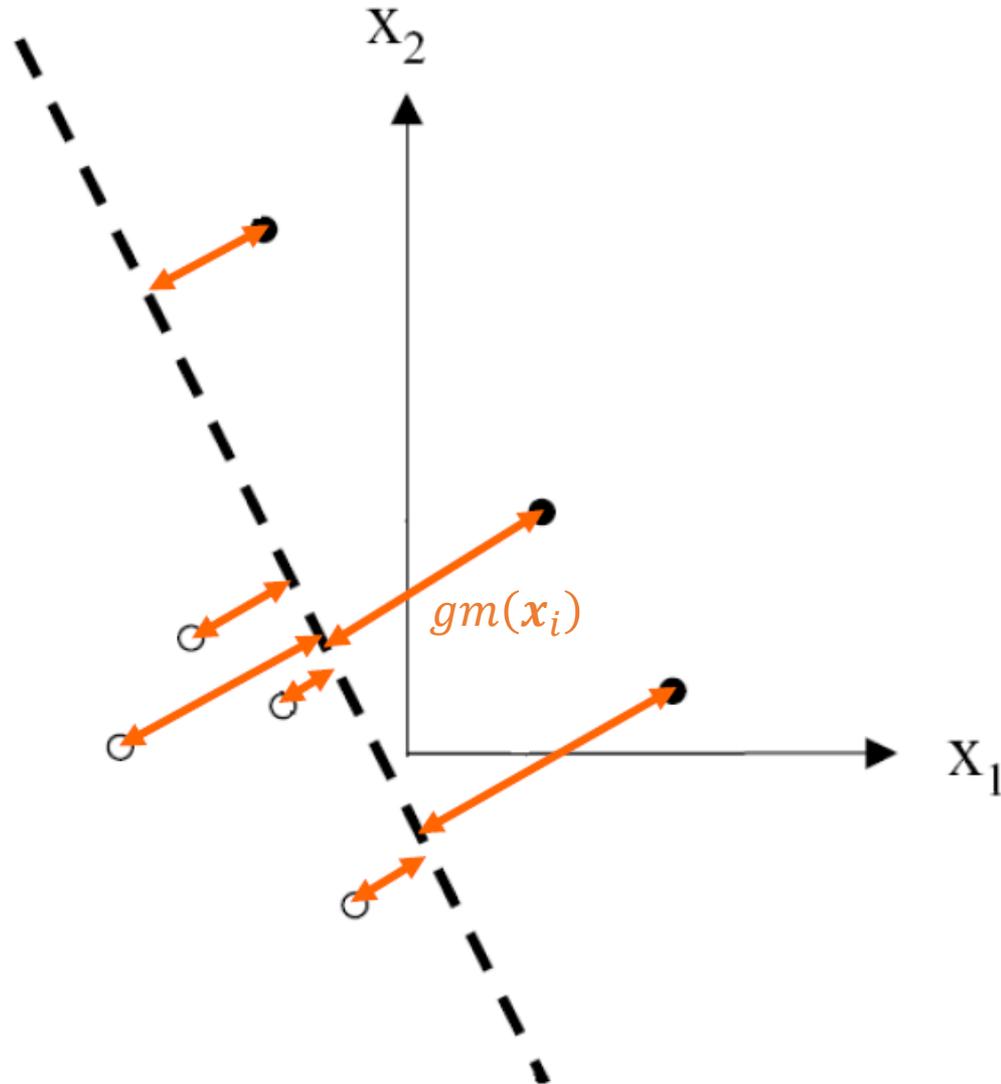


Abstand

... Margin (im Englischen)

$gm(\mathbf{w}) = gm(\mathbf{x}_j)$... der geometrische Abstand der Hyperebene bezüglich der Trainingsdaten $\{\mathbf{X}, \mathbf{y}\}$

$gm(\mathbf{x}_j)$... Vektor mit geringstem geometrischen Abstand: $j = \arg \min_i gm(\mathbf{x}_i)$
wobei $1 \leq i \leq N$



Bezug zum Perceptron

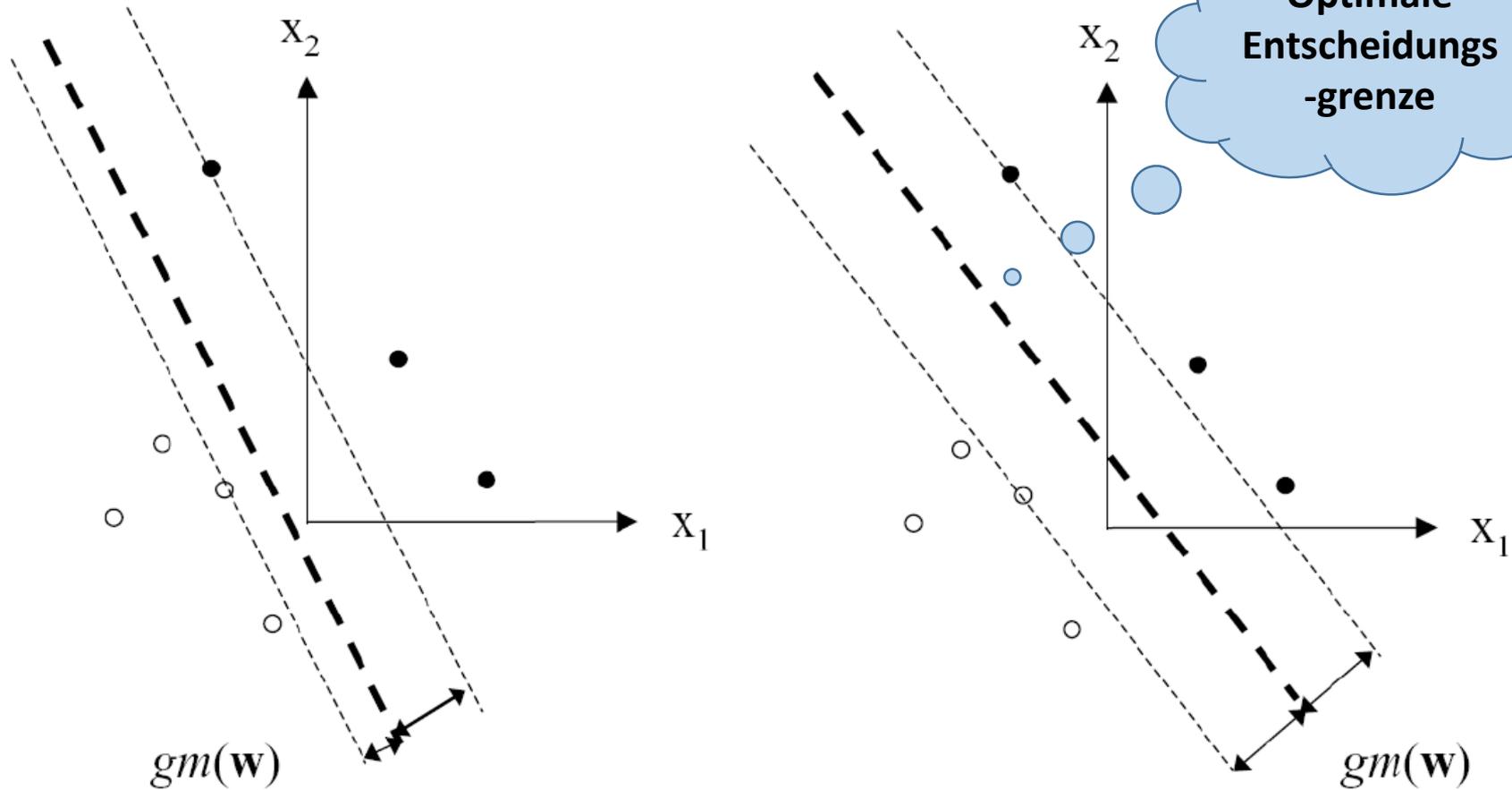
Die **Generalisierungsfähigkeit** des Perceptrons wird umso besser je größer $gm(\mathbf{w})$ ist. Es ist auch zu erwarten, dass ein solcher Klassifikator die Klassifikationsfehler minimiert.



Die Idee den Abstand des Trainingsdatensatzes von der Hyperebene zu maximieren ist die Grundidee des beliebten **SVM-Klassifikators** (SVM=Support Vector Machine).

Solche Klassifikatoren werden auch als „**maximum margin classifiers**“ bezeichnet.

Beispiel



Zusammenfassung

- Schätzung der **Parameter von linearen Diskriminantenfunktionen** durch Perceptron-Algorithmus
- Perceptron kann als **Neuron** dargestellt werden → Grundlage für Neuronale Netze
- Perceptron, wie auch lineare Diskriminantenfunktionen allgemein, sind ein **guter Ansatz für einen Prototyp** eines ME-Systems