Kapitel 8

Neuronale Netze

Ansätze zum Entwurf eines Klassifikators

Es gibt zwei prinzipiell unterschiedliche Ansätze zum Entwurf eines Klassifikators:

1. Statistische parametrische Modellierung der Klassenverteilungen, dann MAP
2. Lösung eines Abbildungsproblems durch Funktionsapproximation (nichtlineare Regression) der a-posteriori-Wahrscheinlichkeiten

$$\min E\left\{ \|\mathbf{f}(\mathbf{x}) - \mathbf{y}\|^2 \right\}$$

$\mathcal{X}$ – *Merkmalsraum*

$\mathcal{Y}$ – *Entscheidungsraum*

Zu 1.) Die bisher beschriebene Vorgehensweise beim Entwurf eines Klassifikators beruht darauf, dass man die *klassenspezifischen Verteilungsdichten*

$$p(\mathbf{x}|\omega)$$

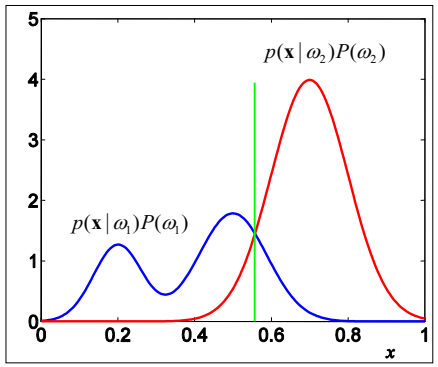
parametrisch durch statistische Modelle annähert (durch Schätzung der Parameter, z.Bsp. Einer Gauß-Verteilung) und durch eine Maximumselektion zu einer Entscheidung kommt. Lernen bedeutet hierbei: Verbesserung des Parameter-Fitting.

Zu 2.) Es gibt nun eine zweite Möglichkeit der Herangehensweise, welche auf die Auswertung der a-posteriori-Wahrscheinlichkeitsdichte

$$p(\omega|\mathbf{x})$$

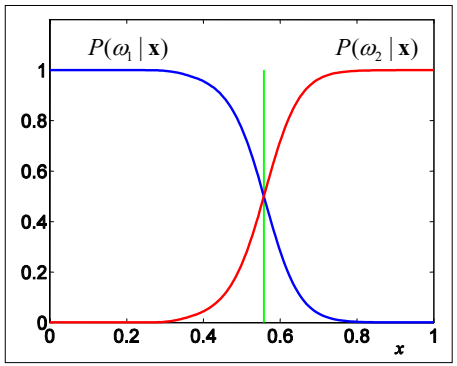
aufbaut und durch ein Problem der *Funktionsapproximation* beschrieben werden kann.

Die zwei Ansätze zum Klassifikatorentwurf



Modellierung der Klassenverteilungsdichten

$$P(\omega|\mathbf{x}) \propto p(\mathbf{x}|\omega)P(\omega)$$



Direkte Modellierung der A-posteriori-Verteilungsdichten

$$P(\omega_k|\mathbf{x})$$

Diese Funktionsapproximation kann z.Bsp. durch eine *nichtlineare Regression mit Polynomen* oder auch mit Hilfe eines *künstlichen Neuronalen Netzwerkes* (NN) durchgeführt werden. Im Rahmen dieser Veranstaltung sollen die Grundlagen für beide Vorgehensweisen behandelt werden.

Grundsätzlich ist die Suche nach einer besten Näherungsfunktion ein *Variationsproblem*, welches durch die Wahl von Basisfunktionen auf ein *parametrisches Optimierungsproblem* zurückgeführt werden kann. Lernen bedeutet dann auch hier: Parameterfitting.

Die Gleichwertigkeit der beiden genannten Vorgehensweisen ergibt sich aus dem Bayes-Theorem, also z.B.:

$$p(\omega \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega)\cancel{p(\omega)}}{\cancel{p(\mathbf{x})}}$$

$1/K$

unabhängig von ω

Die Gleichwertigkeit ergibt sich daraus, dass der Nenner unabhängig von ω ist und die a-priori-W. i.a. $1/K$ gesetzt werden kann.

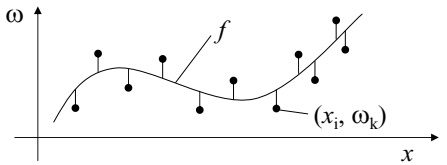
Im folgenden soll nun die Überführung in ein Funktionsapproximationsproblem begründet werden.

Bei bekannter a-posteriori-W. $P(\omega|\mathbf{x})$ würde für jedes kontinuierliche x bestmöglich ein ω zugeordnet werde (funktionale Zuordnung $f:\mathbf{x} \rightarrow \omega$). Gegeben sind jedoch nur *Stichproben* und man sucht nach einer Funktion f, welche bestmöglich die Einzelexperimente fittet und damit eine Abbildung realisert:

$$\mathbf{f}: \underset{\mathbf{x}}{\text{Merkmalsraum}} \rightarrow \underset{\omega}{\text{Bedeutungsraum}}$$

Diese Aufgabenstellung kann mit Hilfe der Variationsrechnung gelöst werden. Wählt man als Gütekriterium das minimale Fehlerquadrat, so geht es um die Minimierung von:

$$J = \min_{\mathbf{f}(\mathbf{x})} E\{\|\mathbf{f}(\mathbf{x}) - \mathbf{y}\|^2\}$$



Dabei entstehen die Zielvektoren $\{\mathbf{y}_i\}$ im *Entscheidungsraum* $\mathcal{Y}$ durch einfache Abbildung der skalaren Bedeutungswerte $\{\omega_i\}$

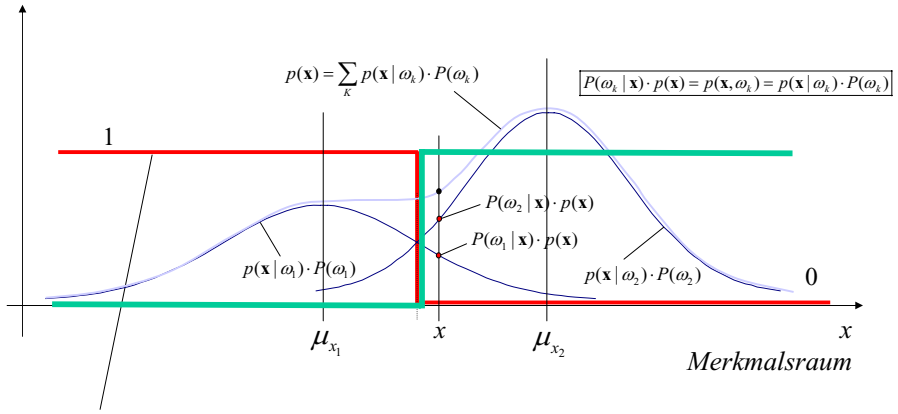
$$\boldsymbol{\Omega} := \{\omega_1, \omega_2, \ldots, \omega_K\}$$
$$\updownarrow \quad \updownarrow \quad \updownarrow$$
$$\mathcal{Y} := \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$$

mit:

$$\mathbf{y}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

i-ter Einheitsvektor

Zwei-Klassen-Problem mit Gaußverteilungsdichte



Regression mit Hilfe *künstlicher* Neuronaler Netze

Verwendung eines mehrschichtigen Perceptrons zur Funktionsapproximation (multilayer perceptron)

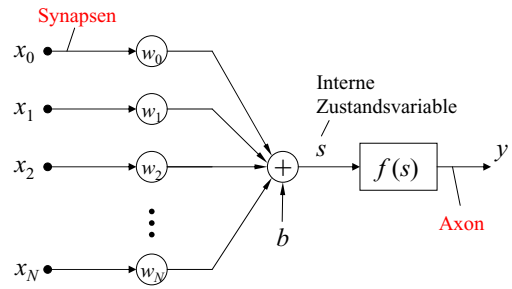
Motivation:

- Anlehnung an menschliche Vorgehensweise (?)
- Parallele Auswertung mit NN-Rechner (Hardware)

Menschliches Gehirn: 10^{11} Neuronen mit bis zu 10^4 Verbindungen pro Neuron

Modell eines Neurons

(McCulloch & Pitts, 1943)

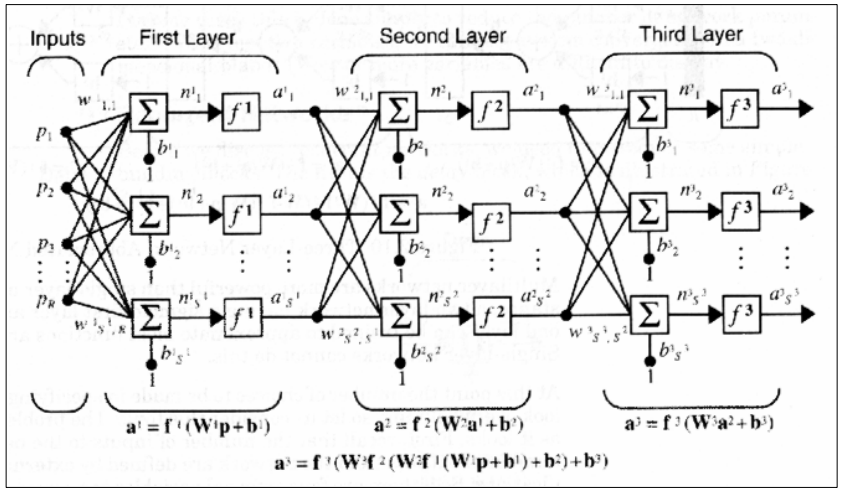


$$y = f(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

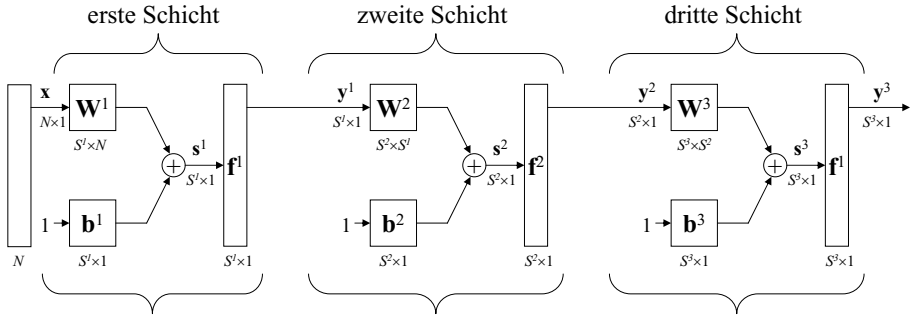
Die neuronale Aktivität wird durch ein Skalaprodukt zwischen dem Gewichtsvektor **w** und den Eingangskanälen x_i mit einer sich anschließenden nichtlinearen Aktivierungsfunktion $f(s)$ beschrieben. Dabei können die Erregungen x_i verstärkend oder hemmend wirken, was zu positiven oder negativen Gewichtswerten w_i korrespondiert.

Das Multilagen-Perceptron mit 3 Schichten

(Rosenblatt 1958)



Symbolische Darstellung eines Perceptrons mit 3 Schichten



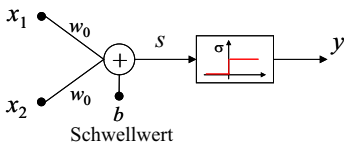
$\mathbf{y}^1=\mathbf{f}^1(\mathbf{W}^1\mathbf{x}+\mathbf{b}^1)$ $\mathbf{y}^2=\mathbf{f}^2(\mathbf{W}^2\mathbf{y}^1+\mathbf{b}^2)$ $\mathbf{y}^3=\mathbf{f}^3(\mathbf{W}^3\mathbf{y}^2+\mathbf{b}^3)$

$$\mathbf{y}^3=\mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{x}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

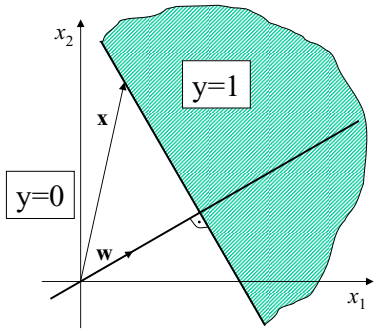
Das Perceptron mit einem Neuron und zwei Eingängen
(Rosenblatt, 1958)

In den Anfangsjahren der NN-Forschung benutzte man vorzugsweise Schwellwertlogiken. D.h. der Eingangsvektor war *zweiwertig* und als Aktivierungsfunktion wurde die *Sprungfunktion* ($f(s)=\sigma(s)=1$ für $s\geq 0$ und $\sigma(s)=0$ für $s<0$).

Da jede Boole'sche Funktion als disjunktive oder konjunktive Normalform geschrieben werden kann, ist es möglich, mit einem *zweischichtigen* Perceptron *jede logische Funktion* zu realisieren.



$$y = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b) = \sigma(g(\mathbf{x}))$$



Funktionsweise bei rellwertigen Eingängen

Zur Funktionsweise des Perceptrons

Das Neuron unterteilt den Eingansraum von **x** mit einer *Hyperebene* (hier: *Gerade*) in zwei Hälften. Die Hyperebene ist der *geometrische Ort*, auf dem alle Vektoren liegen, deren Projektion auf **w** konstant ist:

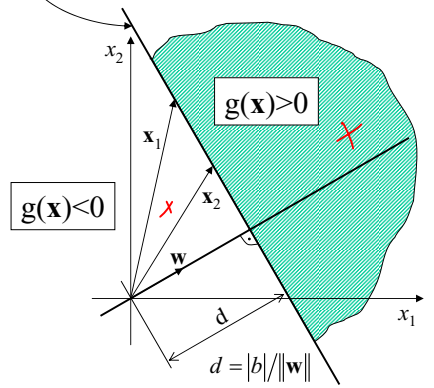
$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = 0$$

Orthogonale Projektion:

$$\frac{\langle \mathbf{w}, \mathbf{x} \rangle}{\|\mathbf{w}\|} = \|\mathbf{x}\| \cos\varphi \gtrless^{?} \frac{|b|}{\|\mathbf{w}\|}$$

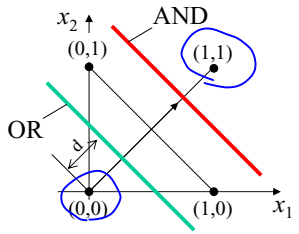
Für zwei Punkte auf der Trennfläche gilt:

$$0 = \langle \mathbf{w}, \mathbf{x}_1 \rangle + b = \langle \mathbf{w}, \mathbf{x}_2 \rangle + b$$
$$\Rightarrow \langle \mathbf{w}, (\mathbf{x}_1 - \mathbf{x}_2) \rangle = 0$$
$$\Rightarrow \mathbf{w} \perp (\mathbf{x}_1 - \mathbf{x}_2)$$

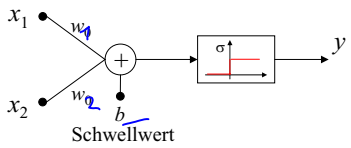


Nichtlinearer Klassifikatorentwurf - das XOR-Problem

Es ist offensichtlich, dass sich die Booleschen Funktionen AND und OR durch einen linearen Klassifikator und damit mit einem einschichtigen Perceptron lösen lassen:



x_1	x_2	AND	Klasse	OR	Klasse
0	0	0	ω_1	0	ω_1
0	1	0	ω_1	1	ω_2
1	0	0	ω_1	1	ω_2
1	1	1	ω_2	1	ω_2



$$\mathbf{w} = \tfrac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

AND	$b = -\tfrac{3}{4}\sqrt{2}$
OR	$b = -\tfrac{1}{4}\sqrt{2}$

oder auch vereinfacht:

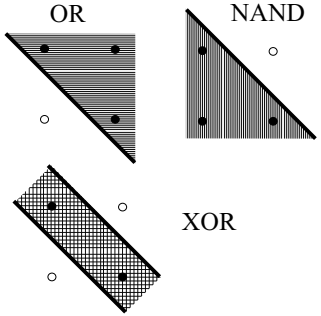
AND: $g(\mathbf{x}) = x_1 + x_2 - \tfrac{3}{2} = 0$

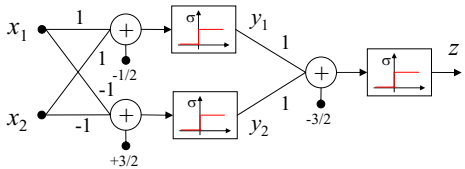
OR: $g(\mathbf{x}) = x_1 + x_2 - \tfrac{1}{2} = 0$

Lösung des XOR-Problems mit einem Zwei-Lagen-Perceptron

Das Exklusiv-Oder-Problem lässt sich hingegen im Gegensatz zu AND und OR *nicht* durch einen linearen Klassifikator lösen.

Eine Lösung ergibt sich jedoch durch Kombination zweier Trennlinien. Dies führt auf ein zweilagiges Perceptron!

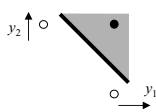




		erste Schicht		zweite Schicht	
x_1	x_2	y_1	y_2	XOR	Klasse
0	0	0	1	0	ω_1
0	1	1	1	1	ω_2
1	0	1	1	1	ω_2
1	1	1	0	0	ω_1

$$z = (x_1 \vee x_2) \wedge \overline{(x_1 \wedge x_2)} = y_1 \wedge \overline{y_2} \quad \text{XOR}$$

Nach der Abbildung der ersten Schicht ergibt sich ein linear trennbares Problem!!



$$y_1 = \sigma(x_1 + x_2 - \tfrac{1}{2}) \quad \text{OR}$$
$$y_2 = \sigma(-x_1 - x_2 + \tfrac{3}{2}) \quad \text{NAND}$$
$$z = \sigma(y_1 + y_2 - \tfrac{3}{2}) \quad \text{AND}$$

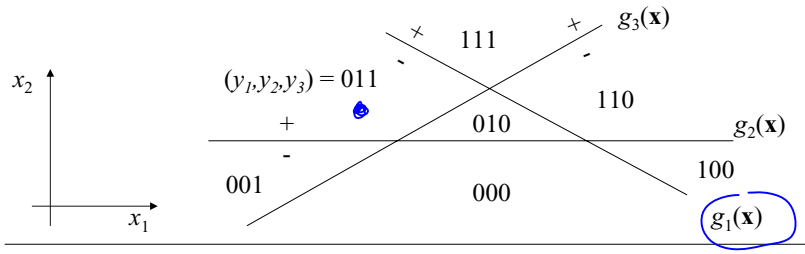
Funktionsweise der ersten Schicht: Abbildung des reellwertigen Eingangsraums auf die Eckpunkte eines Hyperwürfels

In der ersten Schicht eines Perceptrons reagiert ein Neuron mit 0 oder 1, in Abhängigkeit davon, in welcher Hälfte der von der Hyperfläche erzeugten beiden Halbräume sich der Eingangsvektor befindet.

Jedes weitere Neuron erzeugt eine zusätzliche Trennebene. Die erste Schicht eines Perceptrons bildet somit den reellwertigen Eingangsraum auf die Eckpunkte eines Hyperwürfels ab.

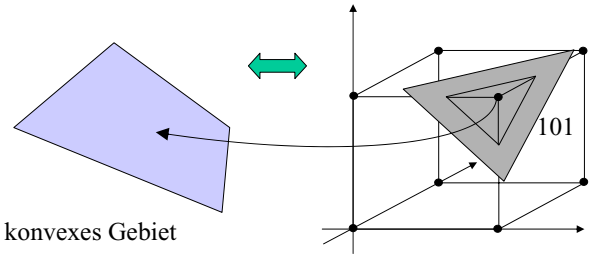
Die sich schneidenden Trennebenen bilden linear begrenzte (konvexe) ***Gebiete***, sogenannte ***Polyeder***. Jedes Gebiet korrespondiert zu einem Eckpunkt des Hyperwürfels.

Nachfolgend ist eine Grafik zu sehen für 3 Neuronen und einen zweidimensionalen Eingangsraum **x**:



Funktionsweise der zweiten Schicht: Abtrennung eines Eckpunktes des Hyperwürfels

In der zweiten Schicht kann durch Abtrennung eines Eckpunktes des Hyperwürfels mit einer neuen Trennfläche ein eindeutiges Merkmal für ein konvexes Gebiet erzeugt werden:



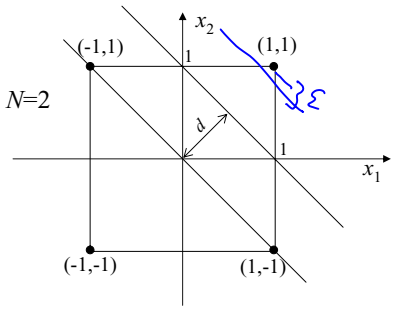
konvexes Gebiet

Eckpunkte eines Hyperwürfels

Optimale Trennebene zur Abtrennung eines Eckpunktes von einem Hyperwürfel

Wird als nichtlineare Funktion im Neuron die Signum-Funktion verwendet, so liegt der daraus resultierende Hyperwürfel zentriert im Ursprung. Die optimale Trennebene liegt in der Mitte zwischen einem Eckpunkt und einer Ebene die von den nächsten Nachbarn aufgespannt wird.

Ohne Einschränkung der Allgemeinheit, können wir den Eckpunkt entlang der ersten Raumdiagonale **u** betrachten. Dafür gilt:



$$\mathbf{u}^T = [1 \quad 1 \quad 1 \quad \dots \quad 1] \quad \text{mit:} \quad \dim(\mathbf{x}) = N$$

$$\Rightarrow \mathbf{e}_u = \mathbf{u} / \|\mathbf{u}\| = \mathbf{u} / \sqrt{N}$$

Eigentlich genügt es, die Trennebene in einem Abstand $d=N^{1/2}-\varepsilon$ mit einem hinreichend kleinen Abstand ε zu platzieren. Problem: über ε in Abhängigkeit von N kann keine genaue Aussage gemacht werden.

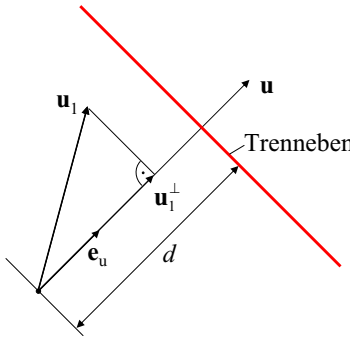
Die Trennebene liegt genau zwischen **u** und der orthogonalen Projektion eines der Nachbarecken $\mathbf{u}_1$ auf **u**:

$$\mathbf{u}_1^T = [\underbrace{1 \quad 1 \quad \dots \quad 1}_{N-1} \quad -1]$$

$$\mathbf{u}_1^{\perp} = \underbrace{\langle \mathbf{u}_1, \mathbf{u} \rangle}_{N-2} \Big/ \sqrt{N} \, \mathbf{e}_u^T = \frac{N-2}{N}\mathbf{u}$$

und damit der Fußpunkt der Trennebene bei:

$$\frac{\mathbf{u}_1^{\perp} + \mathbf{u}}{2} = \tfrac{1}{2}\left(\frac{N-2}{N} + 1\right)\mathbf{u} = \left(1 - \frac{1}{N}\right)\mathbf{u} = \frac{N-1}{N}\mathbf{u} = \underbrace{\frac{N-1}{\sqrt{N}}}_{d}\mathbf{e}_u$$



z. Bsp. $N = 3 \Rightarrow \frac{2}{3}\mathbf{u}$

$N = 4 \Rightarrow 3/4\mathbf{u}$

$N = 100 \Rightarrow 0{,}99\mathbf{u}$

Ein zu **u** benachbarter Eckpunkt des Würfels produziert im Skalarprodukt $\langle \mathbf{u}_1, \mathbf{u} \rangle$ genau den Wert (N-2) !

Die Gleichung der Trennebene ergibt sich zu:

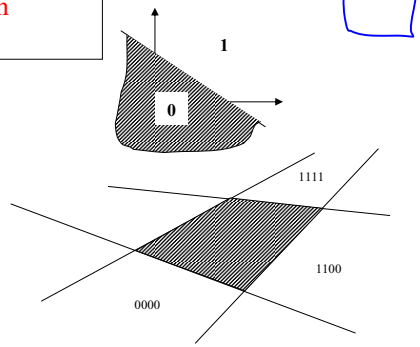
$$\langle \mathbf{x}, \mathbf{e}_u \rangle = d = \frac{N-1}{\sqrt{N}}$$

$$\Rightarrow \langle \mathbf{x}, \mathbf{u} \rangle \frac{1}{\sqrt{N}} = \frac{N-1}{\sqrt{N}}$$

$$\Rightarrow \boxed{\langle \mathbf{x}, \mathbf{u} \rangle = (N-1)}$$

Das 3-Lagen-Perceptron mit Schwellwertfunktion

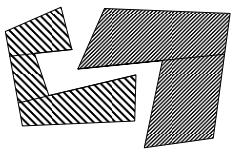
Mit einem 3-L-S-MLPC lassen sich beliebige linear begrenzte Cluster in Merkmalsräumen klassifizieren!!



1. Schicht: sich schneidende Hyperebenen bilden ***konvexe Polyeder*** . Diese werden auf die ***Eckpunkte eines Hyperwürfels*** abgebildet.

2. Schicht: Durch abtrennen von Eckpunkten des Hyperwürfels werden ***konvexe Polyeder*** durch Schnittmengenbildung von Halbräumen selektiert (AND).

3. Schicht: ***beliebige linear begrenzte Gebiete*** entstehen durch Vereinigung von konvexen Gebieten (OR).

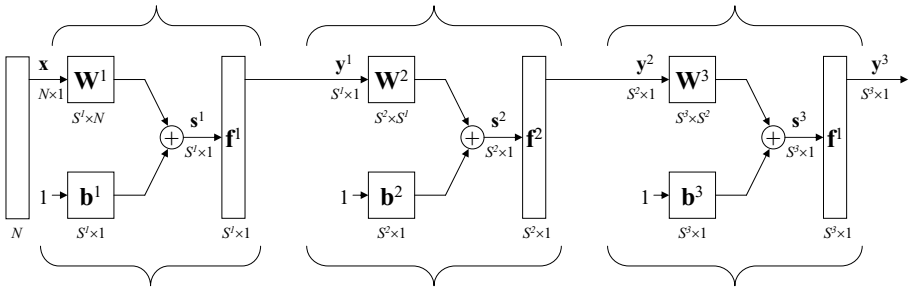


Perceptron mit 3 Lagen zur Realisierung eines Klassifikators für beliebig linear begrenzte Gebiete (Polyeder)

erste Schicht: linear begrenzte Gebiete werden auf Hyperwürfel abgebildet

zweite Schicht: Schnittmengen von Halbräumen (Polyeder) werden Booleschen Vektoren zugeordnet (AND)

dritte Schicht Vereinigung von konvexen Gebieten (OR)



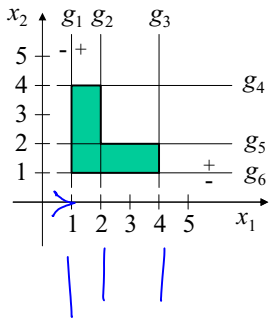
$\mathbf{y}^1=\mathbf{f}^1(\mathbf{W}^1\mathbf{x}+\mathbf{b}^1)$ $\mathbf{y}^2=\mathbf{f}^2(\mathbf{W}^2\mathbf{y}^1+\mathbf{b}^2)$ $\mathbf{y}^3=\mathbf{f}^3(\mathbf{W}^3\mathbf{y}^2+\mathbf{b}^3)$

$$\mathbf{y}^3=\mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{x}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Beispiel: Selektion eines nichtkonvexen Gebietes mit einem dreilagigen Perceptron mit $f(s)$=sign(s)

$$\mathbf{y}^3=\mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{x}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Aufgabe:



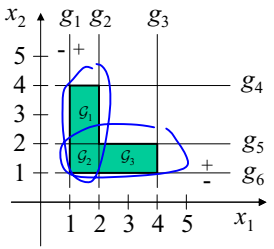
Entwurf der ersten Schicht: Definition der Hyperebenen

$$\mathbf{W}^1=\begin{bmatrix} w_{1,1}=1 & w_{1,2}=0 \\ w_{2,1}=1 & w_{2,2}=0 \\ w_{3,1}=1 & w_{3,2}=0 \\ w_{4,1}=0 & w_{4,2}=1 \\ w_{5,1}=0 & w_{5,2}=1 \\ w_{6,1}=0 & w_{6,2}=1 \end{bmatrix} \begin{matrix} \leftarrow g_1 \\ \leftarrow g_2 \\ \leftarrow g_3 \\ \leftarrow g_4 \\ \leftarrow g_5 \\ \leftarrow g_6 \end{matrix} \quad \mathbf{b}^1=\begin{bmatrix} -1 \\ -2 \\ -4 \\ -4 \\ -2 \\ -1 \end{bmatrix}$$

z.Bsp.: $g_1: \langle \mathbf{w}^1, \mathbf{x} \rangle + b^1 = 0$ mit: $\mathbf{w}^1=\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $b^1=-1$

Entwurf der zweiten Schicht:

Aufgabe:



Entwurf der zweiten Schicht:
Abtrennen von Eckpunkten des Hyperwürfels

Kennzeichnung der Gebiete:

	g_1	g_2	g_3	g_4	g_5	g_6
$\mathcal{G}_1$	+	−	−	−	+	+
$\mathcal{G}_2$	+	−	−	−	−	+
$\mathcal{G}_3$	+	+	−	−	−	+

$$\mathbf{W}^2=\begin{bmatrix} 1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 \end{bmatrix} \begin{matrix} \leftarrow \mathcal{G}_1 \\ \leftarrow \mathcal{G}_2 \\ \leftarrow \mathcal{G}_3 \end{matrix} \quad \mathbf{b}^2=\begin{bmatrix} -5 \\ -5 \\ -5 \end{bmatrix}$$

Die Zeilenvektoren von $\mathbf{W}^2$ zeigen genau auf die abzutrennenden Eckpunkte; der Schwellwert hat den Wert N-1

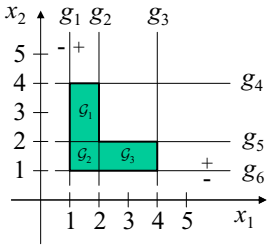
Entwurf der dritten Schicht:

In der dritten Schicht ist die Vereinigungsmenge der Gebiete zu realisieren. Dies geschieht einfach mit einer OR-Verknüpfung. Das bedeutet, dass der Eckpunkt [–1 –1 –1] abgetrennt werden muss. Dies wiederum geschieht mit Hilfe von:

$$\mathbf{W}^3 = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix} \qquad b^3 = -2$$

Vereinfachter Entwurf der zweiten und dritten Schicht:

Aufgabe:



Entwurf der zweiten Schicht:
Abtrennen von Eckpunkten des Hyperwürfels

Kennzeichnung der Gebiete: die Tabelle enthält „don't care"-Elemente (∗). Die dazugehörigen Neuronen können unberücksichtigt bleiben (Verbindung auftrennen)

	g_1	g_2	g_3	g_4	g_5	g_6
$\mathcal{F}_1 = \mathcal{G}_1 \cup \mathcal{G}_2$	+	–	∗	–	∗	+
$\mathcal{F}_2 = \mathcal{G}_2 \cup \mathcal{G}_3$	+	∗	–	∗	–	+

$$\mathbf{W}^2 = \begin{matrix} g_1 & g_2 & g_3 & g_4 & g_5 & g_6 \end{matrix} \begin{bmatrix} 1 & -1 & 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 & -1 & 1 \end{bmatrix} \begin{matrix} \leftarrow \mathcal{F}_1 \\ \leftarrow \mathcal{F}_2 \end{matrix} \qquad \mathbf{b}^2 = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$$

Die Dimension des reduzierten Würfels beträgt nur N=4 (ohne „don't care Elemente!)

Die Zeilenvektoren von $\mathbf{W}^2$ zeigen genau auf die abzutrennenden Eckpunkte; der Schwellwert hat den Wert N-1 des reduzierten Würfels!

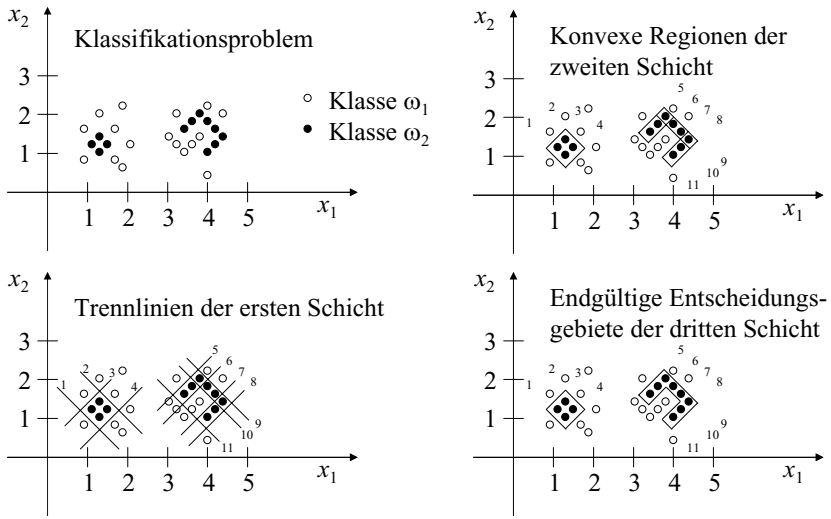
Vereinfachter Entwurf der dritten Schicht:

In der dritten Schicht ist die Vereinigungsmenge der Gebiete zu realisieren. Dies geschieht einfach mit einer OR-Verknüpfung. Das bedeutet, dass der Eckpunkt [–1 –1] abgetrennt werden muss. Dies wiederum geschieht mit Hilfe von:

$$\mathbf{W}^3 = \begin{bmatrix} -1 & -1 \end{bmatrix} \qquad b^3 = -1$$

Beispiel für ein nicht linear separierbares Zwei-Klassenproblem

(M.T. Hagan, H.B. Demuth, M. Beale „Neural Network Design", PWS Publishing Company, Boston, 1995)



Beispiel für ein nicht linear separierbares Zwei-Klassenproblem

Erste Schicht: es werden 11 Geraden zur Abtrennung der Gebiete als Funktion von zwei Eingangsvariablen benötigt!

$$\mathbf{W}^{1^T} = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{b}^{1^T} = \begin{bmatrix} -2 & 3 & 0.5 & 0.5 & -1.75 & 2.25 & -3.25 & 3.75 & 6.25 & -5.75 & -4.75 \end{bmatrix}$$

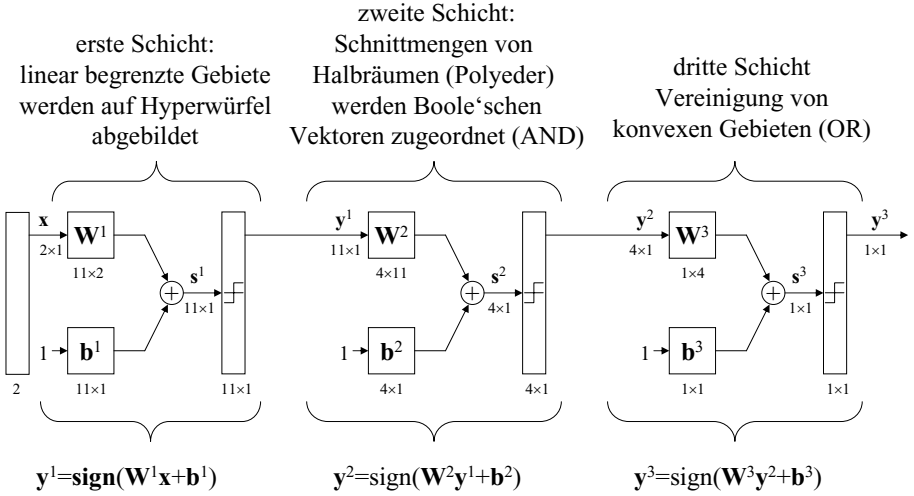
Zweite Schicht: es werden vier konvexe Gebiete selektiert!

$$\mathbf{W}^2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad \mathbf{b}^2 = \begin{bmatrix} -3 \\ -3 \\ -3 \\ -3 \end{bmatrix}$$

Dritte Schicht:

$$\mathbf{W}^3 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \quad b^3 = 3$$

Beispiel für ein nicht linear separierbares Zwei-Klassenproblem



$$\mathbf{y}^3 = \text{sign}(W^3 \text{sign}(W^2 \text{sign}(W^1 x + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

Auf dem Weg zum automatischen Entwurf eines Neuronalen Netzes

Die bisher skizzierte Vorgehensweise ist sehr anschaulich und im zweidimensionalen Fall noch intuitiv handhabbar, aber für höherdimensionale Fälle ist sie nicht brauchbar. Alles was wir in der Praxis zur Verfügung haben, sind die Lernstichproben. Benötigt wird ein automatischer Algorithmus, welcher davon ausgehend in einem Lernprozess die optimalen Gewichte des NN automatisch ermittelt.

Will man für diese nichtlineare Optimierungsaufgabe iterative Algorithmen einsetzen, so benötigt man differenzierbare Nichtlinearitäten in den Neuronen (die Schwellwertfunktionen sind dafür nur begrenzt geeignet!).

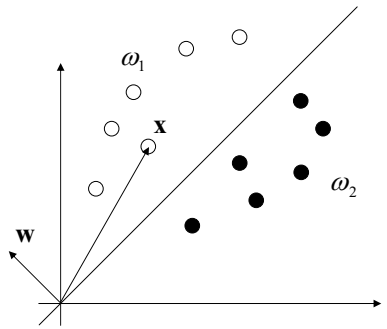
Linear separierbare Klassen – der Perceptron-Algorithmus

Ziel ist, ein Algorithmus zur *automatischen* Anpassung der unbekannten Gewichtsmatrizen $\mathbf{W}^i$ eines Perceptrons an ein Klassifikationsproblem, gegeben eine endliche Stichprobe $\{(\mathbf{x}_i, \omega_k)\}$.
Für den Fall linear separierbarer Klassen gab Rosenblatt einen iterativen Algorithmus zur Lösung dieses Problems an. Es wird angenommen, dass eine lineare Trennfläche, definiert durch die Gleichung $\mathbf{w}^{*T}\mathbf{x}=0$ für das Zweiklassenproblem existiert, derart dass:

$$\mathbf{w}^{*T}\mathbf{x} > 0 \quad \forall \mathbf{x} \in \omega_1$$
$$\mathbf{w}^{*T}\mathbf{x} < 0 \quad \forall \mathbf{x} \in \omega_2$$

Dies schließt auch den Fall ein, dass die Trennfläche nicht wie hier durch den Ursprung läuft, nämlich $\mathbf{w}^{*T}\mathbf{x}+b^*=0$, da dieser durch Erweiterung von $\mathbf{x}$ auf die obige Formulierung zurückgeführt werden kann:

$$\mathbf{x}'^T = \begin{bmatrix} 1, \mathbf{x}^T \end{bmatrix} \quad \mathbf{w}'^{*T} = \begin{bmatrix} b^*, \mathbf{w}^{*T} \end{bmatrix}$$
$$\Rightarrow \mathbf{w}'^{*T}\mathbf{x}' = \mathbf{w}^{*T}\mathbf{x} + b^*$$



Linear separierbares Problem

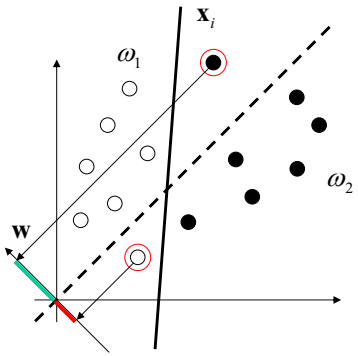
Gelöst wird das Problem durch die Wahl eines geeigneten Gütekriteriums und die Angabe eines Optimierungsalgorithmus. Das zu minimierende Perceptron-Gütemaß sei gegeben durch:

$$J(\mathbf{w}) = \sum_{\mathbf{x}\in\mathbb{Y}} (\delta_x \mathbf{w}^T \mathbf{x}) = \text{—} + \text{—}$$

mit: $\delta_x = +1$ für $\mathbf{x} \in \omega_1$

und: $\delta_x = -1$ für $\mathbf{x} \in \omega_2$

wobei $\mathbb{Y}$ die Menge der Trainingsvektoren enthält, welche durch die gegebene Hyperfläche *falsch* klassifiziert werden.
Offensichtlich ist obige Summe immer positiv und sie wird Null, wenn $\mathbb{Y}$ die leere Menge ist, d.h. alle Elemente *richtig* zugeordnet werden. Nämlich für den Fall eines falsch zugeordneten Vektors $\mathbf{x}\in\omega_1$ erhalten wir mit $\mathbf{w}^T\mathbf{x} > 0$ und $\delta_x > 0$ ein positives Produkt der beiden Terme. Das gleiche gilt für Vektoren der Klasse ω_2 . Das Gütekriterium nimmt seinen Minimalwert Null an, falls alle Stichproben richtig klassifiziert werden.



Das Gütemaß ist *stetig und stückweise linear*. Falls wir den Gewichtsvektor wenig verändern, ändert sich $J(\mathbf{w})$ linear bis zu dem Punkt, wo sich die Zahl der falsch zugeordneten Vektoren ändert. An dieser Stelle ist der Gradient von J nicht definiert und der Gradient von J ist unstetig.
Die iterative Minimierung der Kostenfunktion geschieht durch einen Algorithmus, welcher dem Gradientenabstieg ähnlich ist:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \rho_i \left.\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}_i}$$

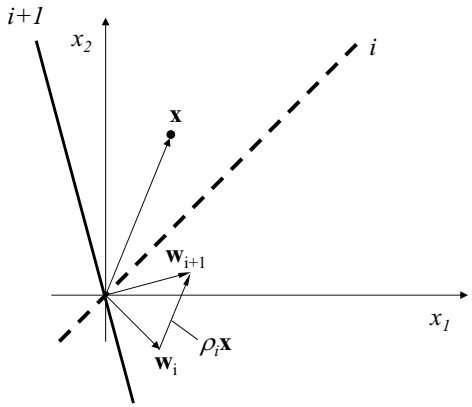
Dabei ist $\mathbf{w}_i$ der Gewichtsvektor für die i-te Iteration und ρ_i eine Sequenz von positiven Zahlen.
Vorsicht ist geboten an den Unstetigkeitsstellen! Aus dem Gütekriterium ergibt sich:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_{\mathbf{x}\in\mathbb{Y}} \delta_x \mathbf{x} \quad \text{wegen:} \quad \frac{\partial \langle \mathbf{w}, \mathbf{x} \rangle}{\partial \mathbf{w}} = \mathbf{x}$$

und damit:

$$\boxed{\mathbf{w}_{i+1} = \mathbf{w}_i - \rho_i \sum_{\mathbf{x}\in\mathbb{Y}} \delta_x \mathbf{x}}$$ Perceptron-Algorithmus

Die Iteration ist fortzusetzen bis der Algorithmus konvergiert.
Das folgende Bild vermittelt einen Eindruck der Funktionsweise. Es wird angenommen, dass beim Iterationsschritt *i* nur ein **x** falsch klassifiziert wird ($\mathbf{w}^T\mathbf{x} > 0$) und es sei ρ_i=0,5. Der Perceptron-Algorithmus korrigiert den Gewichtsvektor *in die Richtung von* **x**. Damit dreht sich die Trennlinie und **x** wird korrekt zugeordnet ($\mathbf{w}^T\mathbf{x} > 0$) .



In dem folgenden Beispiel wird angenommen, dass wir beim vorletzten Iterationsschritt angekommen sind und lediglich noch zwei Merkmalsvektoren falsch zugeordnet werden:

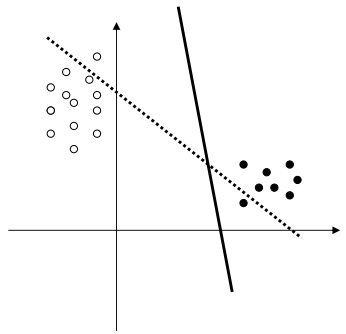
Trennlinie: $x_1 + x_2 - 0,5 = 0 \Rightarrow$ Gewichtsvektor: $\mathbf{w}^T = \begin{bmatrix}1 & 1 & -0,5\end{bmatrix}^T$

Falsch klassifizierte Vektoren: $\begin{bmatrix}0,4 & 0,05\end{bmatrix}^T$ $\begin{bmatrix}-0,2 & 0,75\end{bmatrix}^T$

Mit dem Perceptron-Algorithmus ergibt sich der nächste Gewichtsvektor mit ρ_i=0,7 zu:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - 0,7(-1)\begin{bmatrix}0,4\\0,05\\1\end{bmatrix} - 0,7(+1)\begin{bmatrix}-0,2\\0,75\\1\end{bmatrix} = \begin{bmatrix}1,42\\0,51\\-0,5\end{bmatrix}$$

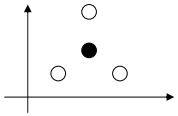
Die neue Trennlinie: $1,42x_1 + 0,51x_2 - 0,5 = 0$ trennt alle Vektoren korrekt. Der Algorithmus terminiert.



Eigenschaften des Perceptron-Algorithmus

- Man kann beweisen, dass der Algorithmus in einer endlichen Anzahl von Iterationen zu einer Lösung konvergiert (bei geeignet gewählten Werten von ρ_i, was bei Gradientenalgorithmen immer ein Problem ist), falls die Voraussetzung erfüllt ist, dass die Klassen linear trennbar sind.
- Die gefundene Lösung ist allerdings nicht eindeutig.

- Für nicht linear-separierbare Probleme erhält man keine Lösung. Die Korrektur in der Iteration hat immer von Null verschiedene Beiträge und kann somit nicht terminieren.



- Später werden wir Multilagen-NN kennenlernen, welche beliebige Klassifikationsprobleme (auch nichtlinear
Backpropagation-Algorithmus kann verw
Netzwerke zu trainieren.

Demo aus Matlab: NN-Toolbox

-Perceptron Learning rule

-Linearly non-separable vectors