UNIVERSITÄT FRANKFURT AM MAIN

www.uni-frankfurt.de

Lernen und Klassifizieren

AS2-2

UNIVERSITÄT FRANKFURT AM MAIN

Assoziativspeicher

Lineare Klassifikation

Lernen linearer Klassifikation

Lernen und Zielfunktion

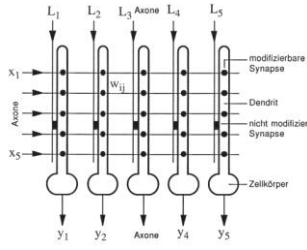
Stochast. Klassifikation

Lernen in Multilayer-Netzen

Backpropagation-Lernen

Neuro-Modell des Assoziativspeichers

UNIVERSITÄT FRANKFURT AM MAIN



Funktion:

Jede Komp.ist lin. Summe

$$\mathbf{z_i = w_i x}$$

Nichtlin. Ausgabe:

$$y_i = S_B(z_i) = \begin{cases} 1 & z > \theta \\ 0 & z \le \theta \end{cases}$$

Lernen von **W** ?

Lernen im Assoziativspeicher

- **Speichern aller *N* Muster mit Hebbscher Regel**

$$w_{ij} = \sum_{k=1}^{N} \Delta w_{ij} = \sum_{k=1}^{N} \gamma_k L_i^k x_j^k \qquad w_{ij}(0) := 0$$

- **Auslesen eines Musters r**

$$\mathbf{y} = \mathbf{W}\mathbf{x}^r = \mathbf{z} = \gamma_r \mathbf{L}^r (\mathbf{x}^r)^T \mathbf{x}^r + \sum_{k \neq r} \gamma_k \mathbf{L}^k \left(\mathbf{x}^k\right)^T \mathbf{x}^r$$

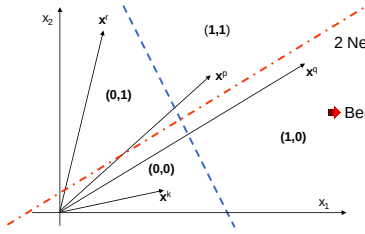
assoziierte Antwort + Übersprechen von anderen Mustern

- **Orthogonale** Muster $\mathbf{x}^r$: Übersprechen = 0, exakte Reproduktion.
- **Nicht-orthogonale** Muster: **Schwellwerte** nötig zum Unterdrücken des Übersprechens.

Trennung mehrerer Klassen

Erinnerung: Lineare Separierung

1 Neuron: 1 Trennlinie (Ebene)

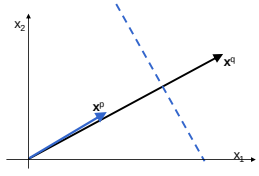


2 Neurone: 2 Trennlinien (Ebenen)

➡ Bereiche trennbar

Trennung mehrerer Klassen

Problem: Klassenentscheidung über Korrelationsgröße



Entscheidung über **x**:

Klasse p: $\mathbf{x}\mathbf{x}^p > \mathbf{x}\mathbf{x}^q$
Klasse q: $\mathbf{x}\mathbf{x}^p < \mathbf{x}\mathbf{x}^q$

Frage: $\mathbf{x} = \mathbf{x}^p$:
In welche Klasse?

Antwort:
in Klasse q !

Lösung

$(\mathbf{x}\text{-}\mathbf{y})^2 = \mathbf{x}^2 - 2\mathbf{x}\mathbf{y} + \mathbf{y}^2$ ist minimal $\Leftrightarrow$ $\mathbf{x}\mathbf{y}$ ist maximal
genau dann, wenn
Konstante Länge $c = |\mathbf{x}| = |\mathbf{y}|$ (normierte Musteraktivität)

Trennung mehrerer Klassen

UNIVERSITÄT FRANKFURT AM MAIN

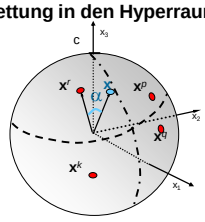
Erweiterung der Mustertupel

$\mathbf{x} \rightarrow \mathbf{x}' = (x_0, x_1, x_2, \ldots, x_n)$ mit $|\mathbf{x}'| = \text{const}$

weil $x_0^2 = c^2 - |(x_1, x_2, \ldots, x_n)|^2 \geq 0$ (!)

≅ Einbettung in den Hyperraum

Beispiel: 2-dim → 3-dim



Entscheidung durch

$$\cos(\alpha) = \frac{\mathbf{x}^T\mathbf{x}^r}{|\mathbf{x}||\mathbf{x}^r|} = c^{-2}\,\mathbf{x}^T\mathbf{x}^r$$

cos(α) monoton fallend ⇒ Winkel als Distanzmaß

min α ⇔ max Korrelation

Assoziativspeicher: Speicherkapazität

UNIVERSITÄT FRANKFURT AM MAIN

M Tupel (**x**,**y**) gegeben:

Wie viele können zuverlässig gespeichert werden?

$\mathbf{x}^1 = \mathbf{x}^2 = \ldots = \mathbf{x}^M$: nur ein Muster speicherbar.

$\mathbf{y}^1 = \mathbf{y}^2 = \ldots = \mathbf{y}^M$: beliebig viele Muster speicherbar, da Antwort y immer richtig.

Problem der **Kodierung** der Muster ! Sei $|\mathbf{x}| = a$.

- **Maximaler** Musterabstand
  max $d(\mathbf{x}^p,\mathbf{x}^q) = \min \mathbf{x}^p\mathbf{x}^q = 0$ bei orthogonalen Mustern
  Reelle Komponenten: *n* Dimensionen ⇒ *n* orthogonale Basisvektoren
  Binäre Komponenten:
  $M_{max} = \left\lfloor \frac{n}{a} \right\rfloor$ z.B. n=100, a=10, also max M=10
- **Mittlere** Abstand maximal

$$\frac{\partial\left\langle d(\mathbf{x}^r,\mathbf{x}^k)\right\rangle}{\partial a}\Big|_{a=a^*} \overset{!}{=} 0 \quad \Rightarrow \quad M = \binom{n}{a^*} = \binom{n}{n/2}$$

z.B. n = 100
max $M \approx 2^n/n^{-0.5} \approx 10^{29}$

Assoziativspeicher: Binärspeicher

UNIVERSITÄT FRANKFURT AM MAIN

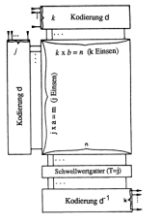
Spärliche Kodierung

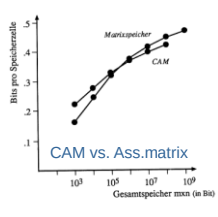
Binäre Muster

Speichern: $w_{ij} = V_p\, y_i^p x_j^p = \max_p\, y_i^p x_j^p$

Kapazität: $H_B = \ln 2 = 0{,}693$ Bit pro Speicherzelle
vergleichbar mit CAM-Speicher

Palm 1980





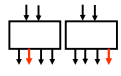
CAM vs. Ass.matrix

Kodierung

$k = a_x = \text{ld } m$

$j = a_y = O(\log n)$

Konstante Zahl von 1 durch eine Leitung pro Eingabecode



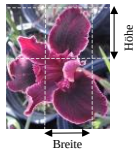
Klassenbildung heute

Objekte werden durch **Merkmale** beschrieben

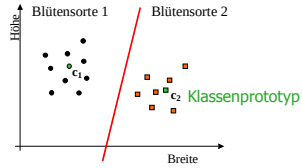
- z.B. ***qualitativ*** Mensch = (groß, braune Augen, dunkle Haare, nett, ...)
- ***quantitativ*** Mensch = (Größe=1,80m, Augenfarbe=2, Haarfarbe=7, ...)

Idee = Form = „Klassenprototyp“

Muster eines Objekts ≡ (Breite, Höhe) = **x**



Trennung von **Klassen**



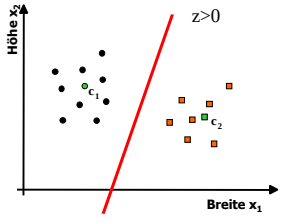
Klassifizierung = Ermitteln der Geradengleichung bzw Parameter c_1,c_2.

Klassentrennung

Klassentrennung durch Trenngerade mit $f(x_1) = x_2 = w_1x_1+w_3$

z<0 z=0 z>0

bzw. $z = w_1x_1+w_2x_2+w_3x_3 = 0$ mit $x_3 := 1$



$$\text{Mit } z = \sum_{i=1}^{n} w_i x_i = \mathbf{w}^T\mathbf{x}$$

Klassenentscheidung

$$y = S(z) = \begin{cases} 0 & z \le 0 \quad \mathbf{x} \text{ aus Klasse 1} \\ 1 & z > 0 \quad \mathbf{x} \text{ aus Klasse 2} \end{cases}$$

Trennung mehrerer Klassen

•**DEF Lineare Separierung**

Seien Muster **x** und Parameter **w** gegeben.
Zwei Klassen Ω_1 und Ω_2
des Musterraums $\Omega = \Omega_1 \cup \Omega_2$ mit $\Omega_1 \cap \Omega_2 = \varnothing$

heißen ***linear separierbar***,

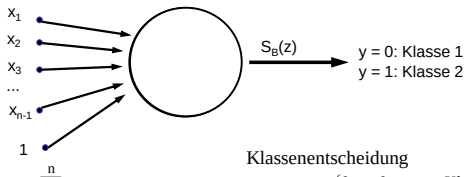
falls eine Hyperebene $\{\mathbf{x}^*\}$ existiert mit $g(\mathbf{x}^*) = \mathbf{w}^T\mathbf{x}^* = 0$,

so daß für alle $\mathbf{x} \in \Omega_1$ gilt $g(\mathbf{x})<0$

und für alle $\mathbf{x} \in \Omega_2$ gilt $g(\mathbf{x})>0$.

Klassentrennung durch formales Neuron

Klassentrennung durch binäres Neuron



$$z = \sum_{i=1}^{n} w_i x_i = \mathbf{w}^T\mathbf{x}$$

Klassenentscheidung

$$y = S_B(z) = \begin{cases} 0 & z \le 0 \quad \mathbf{x} \text{ aus Klasse 1} \\ 1 & z > 0 \quad \mathbf{x} \text{ aus Klasse 2} \end{cases}$$

ALSO :

WIE erhalten wir die richtigen Gewichte,

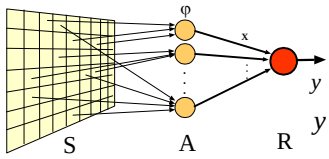
d.h. die richtige Klassifizierung

?

Lernen !

Das Perzeptron

Idee: Reize wiedererkennen
Rosenblatt 1958



- Künstliche Retina
- Assoziations-Schicht
- Response-Schicht

$$y = \sum_i w_i \varphi_i(S)$$

- Verbindungen zu A fix (zufällig): $\mathbf{x} = (x_1,...,x_n)^T = (\varphi_1(S),...,\varphi_n(S))^T$
- Stärke der Verbindungen zu R veränderbar: $\mathbf{w} = (w_1,...,w_n)^T$

Das Perzeptron

Entscheiden

$\Omega := \{\mathbf{x}\}$ alle Muster, $\Omega = \Omega_1 + \Omega_2$

Ω_1 : Menge aller **x** aus Klasse 1
Ω_2 : Menge aller **x** aus Klasse 2

DEF Log. Prädikat $y = \begin{cases} \text{TRUE} & \text{wenn } \mathbf{w}^T\mathbf{x} > s \\ \text{FALSE} & \text{wenn } \mathbf{w}^T\mathbf{x} \le s \end{cases}$ Schwelle

Mit den Erweiterungen $\mathbf{x} = (x_1,...,x_n,1)^T$

$\mathbf{w} = (w_1,...,w_n,s)^T$

wird

$$y = \begin{cases} \text{TRUE} & \text{wenn } \mathbf{w}^T\mathbf{x} > 0 \\ \text{FALSE} & \text{wenn } \mathbf{w}^T\mathbf{x} \le 0 \end{cases}$$

Das Perzeptron: Pseudo-code 3

DEF numerische Werte

$$y = \begin{cases} 1 & \text{wenn} \mathbf{w}^T\mathbf{x} > 0 \\ 0 & \text{wenn} \mathbf{w}^T\mathbf{x} \le 0 \end{cases}$$

$$L(\mathbf{x}) = \begin{cases} 1 & \text{wenn } \mathbf{x} \in \Omega_1 \\ 0 & \text{sonst} \end{cases}$$

PERCEPT3:

Wähle zufällige Gewichte $\mathbf{w}$ zum Zeitpunkt t:=0.

REPEAT

t:= t+1;

$\mathbf{w}(t) = \mathbf{w}(t-1) + \gamma\,(L(\mathbf{x}) - y(\mathbf{x}))\,\mathbf{x}(t)$ *Fehler-Lernregel*

UNTIL (alle $\mathbf{x}$ richtig klassifiziert)

Sogar ohne Umdefinition der Muster aus Ω_2!

Das Perzeptron: Konvergenz

Perzeptron - Konvergenztheorem **(Minsky Papert 1988)**

Wenn die Mustermenge Ω_i **linear separierbar** ist, so konvergiert der Algorithmus bei $t \to \infty$

Problem: Wenn Klassen sich überlappen, so wird die Grenzlinie bei $\gamma = 1$ immer hin und her geschoben

Das Perzeptron: Zielfunktion

Ziel: Verallgemeinerung der Lernregel

Hier: Minimierung **aller** Fehlentscheidungen

DEF $R(\mathbf{w}) = \sum_{\mathbf{x}\in\Omega_F} -\mathbf{w}^T\mathbf{x}$ ***Perzeptron-Zielfunktion „Energie“***

Neuformulierung erwartetes Lernen:

$\mathbf{w}(t) = \mathbf{w}(0) - \gamma\,\nabla_{\mathbf{w}} R(\mathbf{w})$ *Gradient*

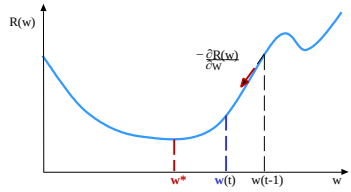
d.h.

$$\mathbf{w}(t) = \mathbf{w}(0) + \gamma \sum_{\mathbf{x}\in\Omega_F} \mathbf{x}$$

$\mathbf{w}(t) = \mathbf{w}(t-1) - \gamma\,\nabla_{\mathbf{w}_{t-1}} r(\mathbf{w}(t-1), \mathbf{x})$ Stochast. Lernen

Lernen durch Iteration

Gradientenabstieg einer Zielfunktion R(w)



$$\Delta \mathbf{w} := (\mathbf{w}(t\text{-}1) - \mathbf{w}(t)) \sim -\nabla_{\mathbf{w}} R(\mathbf{w}(t\text{–}1))$$

$$\mathbf{w}(t) = \mathbf{w}(t\text{–}1) - \gamma(t)\, \nabla_{\mathbf{w}} R(\mathbf{w}(t\text{–}1))$$

Was kann ein Perzeptron ?

Erwartung: „Intelligente Leistungen“ durch Wahl von φ(S)

Abbildung der Merkmale auf linear separierbare Mustermengen

Perzeptronarten

- ***diameter-limited perceptrons***
  nur Bildpunkte aus einem begrenzten Radius
- ***order-restricted perceptrons***
  von maximal *n* (beliebigen) Bildpunkten abhängig
- ***random perceptrons***
  eine zufällige Auswahl aller Bildpunkte

Was kann ein Perzeptron ?

Topologische Prädikate, z.B.

- „X ist ein Kreis“ ?
- „X ist eine konvexe Figur“ ?
- „X ist eine zusammenhängende Figur“ ?
- …

Tatsache: ***keine*** korrekte Klassifizierung von Punktmengen X (Bildpixeln) dieser Arten

Nur „X hat Eulerzahl E“

E(X) : = K(X) – Anzahl der Löcher

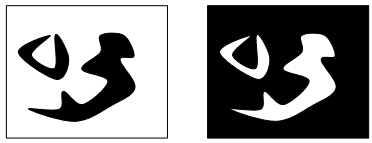
Was kann ein Perzeptron ?

Eulerzahl E

E(X) : = K(X) – Anzahl der Löcher

K(X) : = zusammenhängende Komponenten

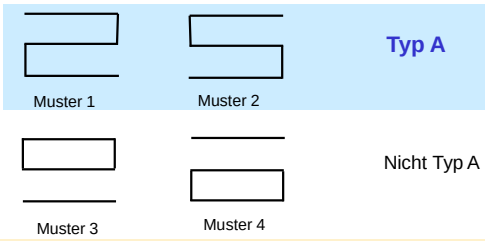
Loch := zusamm. Komponente der komplementären Menge



K(x) = 2, Löcher = 1 $\Rightarrow$ E(x) = 1

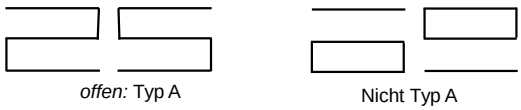
Was kann ein Perzeptron ?

Beispiel: *keine* korrekte Klassifizierung von Punktmengen X (Bildpixeln) für Prädikat „X ist Typ A“ möglich mit „diameter-limited“ Perzeptron



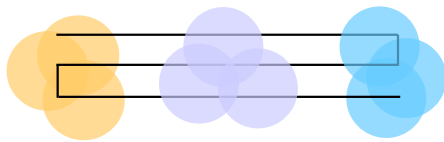
Was kann ein Perzeptron ?

Beweis:



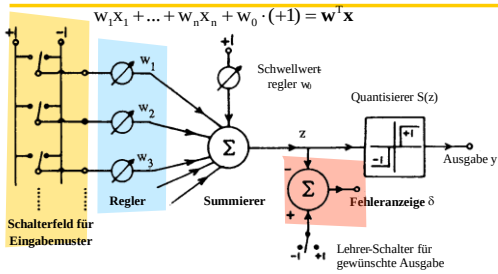
offen: Typ A

Nicht Typ A



Adaline: Aktivität

$$w_1 x_1 + \ldots + w_n x_n + w_0 \cdot (+1) = \mathbf{w}^T \mathbf{x}$$



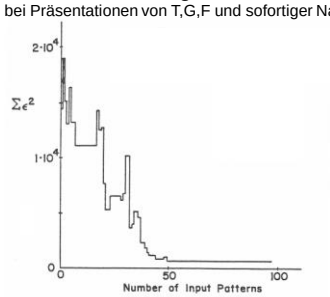
$w_i \in \Re^+$

$x_i \in \{-1, +1\}$

$$y = \begin{cases} +1 & \text{wenn } \mathbf{w}^T\mathbf{x} > 0 \\ -1 & \text{wenn } \mathbf{w}^T\mathbf{x} \le 0 \end{cases}$$

Adaline: Aktivität

Verlauf des Klassifizierungsfehlers für „Klasse T liegt vor"
bei Präsentationen von T,G,F und sofortiger Nachregelung



Adaline: Lernalgorithmus

Minimierung des erwarteten quadratischen Fehlers

$$R(\mathbf{w},L) := \left\langle (z(\mathbf{x}) - L(\mathbf{x}))^2 \right\rangle_{\mathbf{x}} = \left\langle (\mathbf{w}^T\mathbf{x} - L(\mathbf{x}))^2 \right\rangle_{\mathbf{x}}$$

durch Anpassung der Parameter

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \gamma(t) \frac{\partial}{\partial \mathbf{w}} R(\mathbf{w}(t-1))$$

$$\mathbf{w}(t) = \mathbf{w}(t\text{-}1) - \gamma(t)(\mathbf{w}^T\mathbf{x} - L(\mathbf{x}))\mathbf{x}$$ *stochastische Approximation*

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \gamma(t)(\mathbf{w}^T\mathbf{x} - L(\mathbf{x}))\frac{\mathbf{x}}{|\mathbf{x}|^2}$$ *Widrow-Hoff Lernregel*

Übersicht: Lernen

UNIVERSITÄT FRANKFURT AM MAIN

Assoziativspeicher

1. Muster $\mathbf{x}^k$ eingespeichert

$\mathbf{w}_i(1) = L_i^k \mathbf{x}^k$ *(Hebb'sche Regel)*

Perzeptron

$\mathbf{w}_i(t) = \mathbf{w}_i(t\text{-}1) + \gamma(L_i(\mathbf{x})\text{-}y_i)\mathbf{x}$ *(Fehler-Lernregel)*

$\mathbf{w}_i(1) = (L_i(\mathbf{x}^k)\text{-}y_i)\mathbf{x}^k = L_i^k \mathbf{x}^k$ bei $w_i(0) = 0 \Rightarrow y_i^k(0) = 0.$

Adaline

$\mathbf{w}_i(t) = \mathbf{w}_i(t\text{-}1) + \gamma(t)(L(\mathbf{x})\text{-}z_i)\mathbf{x}$ *(Gradientenabstieg)*

$\mathbf{w}_i(1) = (L_i(\mathbf{x}^k)\text{-}z_i)\mathbf{x}^k = L_i^k \mathbf{x}^k$ bei $w_i(0) = 0 \Rightarrow z_i^k(0) = 0.$

Assoziativspeicher = Grundfunktion von Netzen

UNIVERSITÄT FRANKFURT AM MAIN

Übersicht Lernarten

UNIVERSITÄT FRANKFURT AM MAIN

- **Beispiel-basiertes Lernen** *(example based learning, feedback learning)*
  Gegeben: (**Eingabe** **x**, gewünschte **Ausgabe** *L*)
  Ziel: Differenz zwischen *y* und *L* im Laufe des Lernens klein machen.
- **Erklärungs-basiertes Lernen** *(explanation based learning EBL)*
  Gegeben: **Beispielpaare, Ziel** sowie **Regeln**, es zu erreichen.
  Lernen: Generalisierung der Beispiele.
  (regelbasierte Systeme, nicht bei neuronalen Netzen)
- **Score-basiertes Lernen** *(reinforcement learning)*
  Gegeben: **skalares** Gütemaß ("gut", "schlecht", mit Abstufungen dazwischen) für Lernleistung.
  Lernen: **??** *Der Lernende muss daraus selbst sehen, was an der Ausgabe zu ändern ist.*
- **Unüberwachtes Lernen** *(observation based learning, emotion based learning, similarity learning)*
  Gegeben: **keine explizite** Rückmeldung über die Güte seines Lernens
  Lernen: Vergleich gewünschte Auswirkungen mit beobachteten Auswirkungen. Folgerung für geeignete Verhaltensänderung.

Lernen durch Iteration

UNIVERSITÄT FRANKFURT AM MAIN

Modifikationen Gradientenabstieg

- **Taylorentwicklung**

$$f(x+\Delta x) = f(x) + \frac{\partial}{\partial x} f(x)\,\Delta x + \frac{1}{2!}\frac{\partial^2}{\partial x^2} f(x)(\Delta x)^2 + \ldots$$

$$R(\mathbf{w}+\Delta\mathbf{w}) - R(\mathbf{w}) = \nabla_w R(\mathbf{w})^T\,\Delta\mathbf{w} + \tfrac{1}{2}\Delta\mathbf{w}^T\boldsymbol{R}\,\Delta\mathbf{w} + \ldots$$

$$\text{mit } \boldsymbol{R} = \frac{\partial^2 R}{\partial w_i \partial w_j} \qquad \textit{Hesse-Matrix}$$

- **Conjugate gradient**

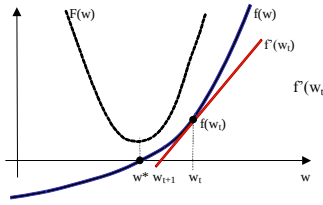
$$R(\mathbf{w}+\Delta\mathbf{w}) - R(\mathbf{w}) = (\nabla_w R(\mathbf{w})^T + \tfrac{1}{2}\Delta\mathbf{w}^T\boldsymbol{R})\,\Delta\mathbf{w} = \mathbf{0}$$

löse *n*-dim Gleichungssystem für $\Delta\mathbf{w}$

Lernen durch Iteration

UNIVERSITÄT FRANKFURT AM MAIN

Newton-Iteration



$$f'(w_t) = \frac{f(w_t) - f(w_{t+1})}{w_t - w_{t+1}} = \frac{f(w_t)}{w_t - w_{t+1}}$$

$$w_{t+1} = w_t - \frac{f(w_t)}{f'(w_t)} \qquad \textit{Newton-Verfahren}$$

$$w_{t+1} = w_t - \frac{F'(w_t)}{F''(w_t)}$$

Lernen durch Iteration

UNIVERSITÄT FRANKFURT AM MAIN

Konvergenz des Gradientenverfahrens

Es ist $R(t)$ = *Ljapunov-Funktion* mit Konvergenz, **wenn**

- $R(t+1) \leq R(t)$ bzw. $\frac{\partial R}{\partial t} \leq 0$ — *monoton fallend*
- Ex. endliches $R_0 \leq R(t)$ für jedes t — *Ljapunov-Bedingung*

Also: Wenn $\frac{\partial R}{\partial t}(w(t)) = \nabla_w R(w)\,\frac{\partial \mathbf{w}}{\partial t} \leq 0$ **dann Konvergenz**

Hinreichend dafür: $\frac{\partial \mathbf{w}}{\partial t} = -\gamma \nabla_w R(\mathbf{w})$ mit $\gamma > 0$

weil $\frac{\partial R}{\partial t} = -\gamma(\nabla_w R(\mathbf{w}))^2 \leq 0$

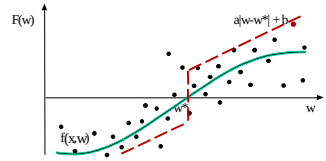
Mit $\frac{\partial \mathbf{w}}{\partial t} \approx \frac{\Delta \mathbf{w}}{\Delta t}$ und $\Delta t = 1$ ist

$$\boxed{\mathbf{w}(t) - \mathbf{w}(t\text{-}1) = -\gamma \nabla_w R(\mathbf{w})}$$

Gradientenabstieg

Stochastische Approximation

Gesucht: Nullstelle einer stochast. Funktion f(x,w) = R'(x,w)



Methode 1: Alle Ereignisse x abwarten und dann F(w) = $\langle f(x,w) \rangle_x$ bilden

$$\mathbf{w}(t) = \mathbf{w}(t\text{-}1) - \gamma(t)\, F(\mathbf{w}(t\text{-}1))$$

Methode 2: Einfach f(x,w) verwenden *Robbins, Monro 1951*

$$\mathbf{w}(t) = \mathbf{w}(t\text{-}1) - \gamma(t)\, f(\mathbf{w}(t\text{-}1), \mathbf{x}(t))$$ *stochastische Approximation*

Stochastisches Lernen

Lernen mit Zielfunktion R(w) = $\langle r(w,x) \rangle_x$

$$\mathbf{w}(t) = \mathbf{w}(t\text{-}1) - \gamma(t)\, \nabla_w R(\mathbf{w}(t\text{-}1))$$

wird ersetzt durch

Lernen mit stochast. Zielfunktion r(w,x)

$$\mathbf{w}(t) = \mathbf{w}(t\text{-}1) - \gamma(t)\, \nabla_w r(\mathbf{w}(t\text{-}1), \mathbf{x}(t))$$ *stochastisches Lernen*

Stochastische Approximation

Voraussetzungen *das klein Gedruckte...*

- die Funktion F(w) := $\langle f(x,w) \rangle_x$ ist *zentriert*, d.h. F(w*) = 0
- F(w) ist *ansteigend*, d.h. F(w<w*) < 0, F(w>w*) > 0 .
- F(w) ist *beschränkt* mit $|F(w)| \leq a|w\text{-}w^*|+b < \infty$ a,b > 0
- f(x,w) hat *endliche Varianz*, d.h. $\sigma^2(w) = \langle (F(w) - f(x,w))^2 \rangle_x < \infty$
- $\gamma(t)$ *verschwindet*, $\gamma(t) \to 0$
- $\gamma(t)$ wird nicht zu schnell *klein* $\sum_{t=1}^{\infty} \gamma(t) = \infty$
- $\gamma(t)$ wird nicht zu *groß* $\sum_{t=1}^{\infty} \gamma(t)^2 < \infty$

Dann ex.

$\lim_{t\to\infty} \langle (w(t) - w^*)^2 \rangle = 0$ mittl. quadr. Konv. *Robbins-Monro*

$P(\lim_{t\to\infty} w(t) = w^*) = 1$ *Blum*

Stochastische Iteration: Konvergenz

Beispiel

Sei die Zufallsvariable X gegeben, geschätzt durch w.

Abweichung bei der Schätzung ist

$R(w) = \langle r(w,x)\rangle_x = \langle (w-x)^2\rangle_x$ *mean squared error*

$\mathbf{w}(t) = \mathbf{w}(t-1) - \gamma(t)\, \nabla_{\mathbf{w}} r(\mathbf{w}(t-1), \mathbf{x}(t))$ *stoch. Gradient*

$\mathbf{w}(t) = \mathbf{w}(t-1) - \gamma(t)(\mathbf{w}(t-1) - \mathbf{x}(t))$

Zeitabhängigkeit

$R(w) \rightarrow R(w^*)$ bei $w \rightarrow w^*$

stoch. und erwarteter Verlauf?

Stochastische Iteration: Konvergenz

Stochastische Iteration

$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) - \gamma(t)(\mathbf{w}_i(t-1) - \mathbf{x}(t))$

Behauptung

Bei $\gamma(t) := 1/t$ ist immer $\mathbf{w}(t) = \langle \mathbf{x}\rangle_x$

Beweis durch vollständige Induktion $w(0) \equiv 0$ Kap.2.3.2

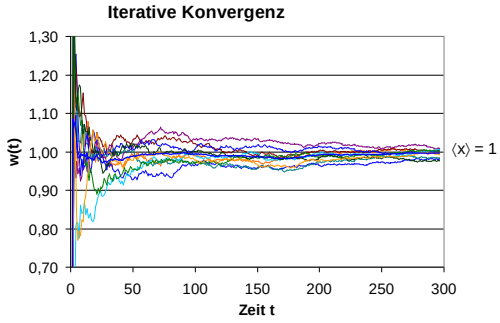
✓ $\mathbf{w}(t=1) = 0 - \gamma(t)(0-x) = x = \langle \mathbf{x}\rangle_x$ *Induktionsverankerung*

Mit $\mathbf{w}(t-1) = \langle \mathbf{x}\rangle_{t-1} = \sum_{i=1}^{t-1} \mathbf{x}(i)$ *Induktionsvoraussetzung*

✓ gilt $\mathbf{w}(t) = \ldots = \langle \mathbf{x}\rangle_t$ *Induktionsschritt*

q.e.d.

Konvergenzverlauf



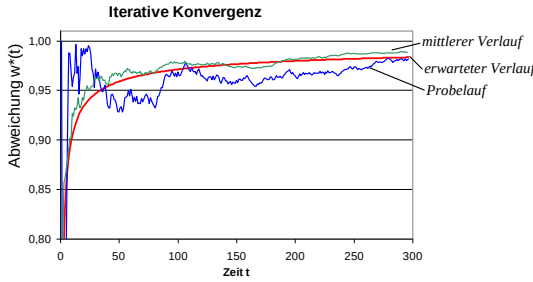
Erwarteter Konvergenzverlauf

Rechnung Anhang D.4

- mittl. quadrat. Abweichung
- Erwartungswert aller Verläufe
- Abweichung durch Standardabweichung beschreibbar

$$\langle\, |w^* - w(t)| \,\rangle = \sigma_t = \sigma_x / \sqrt{t}$$

Konvergenzverlauf



$w^* = 1, \quad \sigma_x = 0{,}288$

Stochastisches Lernen

Beispiel Klassentrennung

$$\mathbf{w}_i(t) = \mathbf{w}_i(t\text{-}1) - \gamma(t)(\mathbf{w}_i(t\text{-}1)\text{-}\mathbf{x}(t))$$

Behauptung

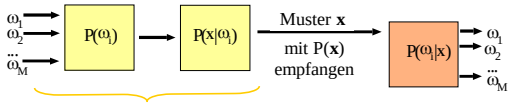
Bei $\gamma(t) := 1/t$ ist immer $\mathbf{w}(t) = \langle \mathbf{x} \rangle_{\mathbf{x}}$ *Klassenprototyp*

Beweis durch vollständige Induktion $w(0) \equiv 0$

Problem: $\langle \mathbf{x} \rangle_{\mathbf{x}}$ ist abhängig von der Klassenentscheidung für x

Stochastische Musterklassifikation

Grundsituation der Erkennung



Quelle, Sender — *a priori*

Empfänger — *a posteriori*

Notation: Mustermenge Ω = {**x**}, unterteilt in Klassen Ω_i

ω_k = " Klasse *k* liegt vor "

Klassifikation

Ω_k: $P(\omega_k|x) = \max_j P(\omega_j|x)$ *Bayes-Klassifikation*

Wie erhalten wir $\mathbf{P(\omega_j|x)}$ **?**

Stochastische Klassifikation

$P(\omega_j|\mathbf{x}) = ?$

Bekannte Quellen

Sei *a-priori* $P(\omega_j)$, $P(\mathbf{x}|\omega_j)$, $P(\mathbf{x})$ bekannt

und $P(\mathbf{x}, \omega_j) = P(\omega_j|\mathbf{x})P(\mathbf{x}) = P(\mathbf{x}|\omega_j)P(\omega_j)$

so ist $P(\omega_j|\mathbf{x}) = P(\mathbf{x}|\omega_j)P(\omega_j) / P(\mathbf{x})$ mit $P(\mathbf{x}) = \sum_j P(\mathbf{x}|\omega_j)P(\omega_j)$

Aufgabe !

Unbekannte Quellen

A-posteriori $P(\omega_j|\mathbf{x})$ lernen !

Zielfunktion: Messung der Klassifikationsleistung ?

Klassifikationsleistung

Diagnose-Situation

(*Diagnose*, *Realität*)	*Name*	*Wahrscheinlichkeit*
$(D(x) = \omega \mid \omega)$	Sensitivität **TP**	$P_K = P(D(x) = \omega \mid \omega)$
$(D(x) = \neg\omega \mid \omega)$	Ignoranz **FP**	$P_I = P(D(x) = \neg\omega \mid \omega)$
$(D(x) = \omega \mid \neg\omega)$	Fehlalarm **FN**	$P_A = P(D(x) = \omega \mid \neg\omega)$
$(D(x) = \neg\omega \mid \neg\omega)$	Spezifität **FP**	$P_L = P(D(x) = \neg\omega \mid \neg\omega)$

$P_K + P_I = 1$ FP = FRR *false rejection rate*

$P_A + P_L = 1$ FN = FAR *false acceptance rate*

Klassifikationsleistung

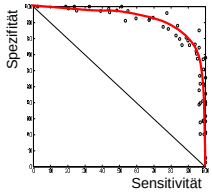
Diagnose-Situation („confusion matrix“)

	ω	$\neg\omega$
D(x) = ω	**Sensitivität** $P(D(x) = \omega \mid \omega)$ *true positive*	**Fehlalarm** $P(D(x) = \omega \mid \neg\omega)$ *false negative*
D(x) = ¬ω	**Ignoranz** $P(D(x) = \neg\omega \mid \omega)$ *false positive*	**Spezifität** $P(D(x) = \neg\omega \mid \neg\omega)$ *true negative*

ROC -Kurven von Diagnosesystemen

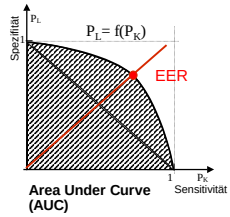
Wechselseit. Abhängigkeit Sensitivität / Spezifität

Beispiel med. Diagnose



Receiver Operating Characteristic (ROC)

Leistung eines Diagnosesystems



Area Under Curve (AUC)

ROC -Kurven von Diagnosesystemen

Aufgabe:

Ex. ein Diagnosesystem mit

D(x) > c Klasse A liegt vor

D(x) < c Klasse A liegt *nicht* vor

Frage:

Wie wird ROC und AUC davon gemessen?

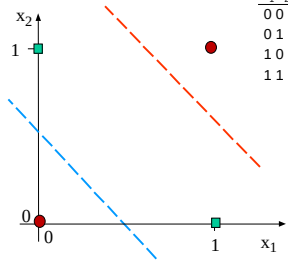
Antwort:

- Für festes c über alle x die Leistung (P_k ,P_L) messen, einen Punkt der Grafik einzeichnen
- c variieren, und jeweils Punkt zeichnen
- ROC in die Punkte einpassen, AUC davon berechnen

Das XOR-Problem

Aufgabe

Trennung zweier Klassen durch eine Gerade – wie ?



x_1 x_2	$x_1 \oplus x_2$
0 0	0
0 1	1
1 0	1
1 1	0

$\Omega_0 = \{$ ● $\}$
$= \{(0,0), (1,1)\}$

$\Omega_1 = \{$ ■ $\}$
$= \{(1,0), (0,1)\}$

Klassen **nicht** linear separierbar!

Das XOR-Problem

UNIVERSITÄT FRANKFURT AM MAIN

Lösung

Trennung durch **zwei** Schichten

$\mathbf{y} = \neg(x_1 \oplus x_2)$ *negiertes XOR*

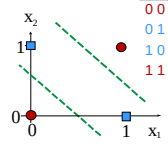
$= (x_1 \text{OR} \neg x_2) \text{ AND } (\neg x_1 \text{OR } x_2)$

$y_1 := x_1 \text{ OR } \bar{x}_2$

$y_2 := \bar{x}_1 \text{ OR } x_2$

$y_{XOR} := y_1 \text{ AND } y_2$

x_1 x_2	$x_1 \oplus x_2$
0 0	0
0 1	1
1 0	1
1 1	0



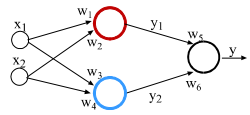
z.B. formale binäre Neuronen

S(z≥s) = 1, S(z<s) = 0

$\Rightarrow w_1 = w_4 = w_5 = w_6 = 1/3$

$w_2 = w_3 = -1/3$

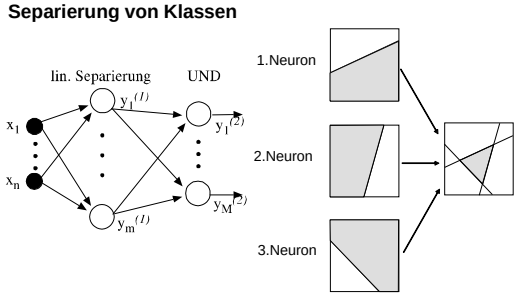
$s_1 = s_2 = 0,\ s = 1/2$



Multilayer-Klassifikation

UNIVERSITÄT FRANKFURT AM MAIN

Separierung von Klassen



Fähigkeiten der Multilayer-Netzwerke

UNIVERSITÄT FRANKFURT AM MAIN

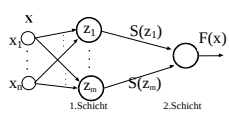
Approximationsnetze

Interpolation anhand von Beispielen (Stützstellen)

Typ. Netz *Linearkombinationen von Basisfunktionen S(.)*

$$F(\mathbf{x}) = \sum_{j=1}^{m} w_j S\left(z_j(\mathbf{x})\right)$$

Sigma-Funktion F: $\Re^n \rightarrow \Re$



wobei

$\{ z \mid z(\mathbf{x}) = \mathbf{w}^{(1)T}\mathbf{x} + b \}$ *affine Funktionen* $\Re^n \rightarrow \Re$

$\lim_{x \to \infty} S(x) = 1, \quad \lim_{x \to -\infty} S(x) = 0$ S *ist Quetschfunktion*

Fähigkeiten der Multilayer-Netzwerke

Satz Hornik, Stinchkombe, White 1989

Für die Funktionswerte *jeder beliebigen Funktion* f(**x**) : $\Re^n \rightarrow \Re$ von *N* Mustern $\mathbf{x}^1$ .. $\mathbf{x}^N$ ex. eine Sigma-Funktion F, so dass für alle Muster $\mathbf{x}^i$ mit i = 1..N gilt

$F(\mathbf{x}^i) = f(\mathbf{x}^i)$ Gilt auch für Schicht $\{F_i\}$

Assoziativspeicher

Satz

Jede beliebige, stetige Funktion f(x) in einem kompakten Intervall ("kompakte Teilmenge des $\Re^n$ ") kann beliebig dicht (*uniform dicht* im Sinne der L_2-Norm in der Menge C^n aller stetigen Funktionen und ρ_p-*dicht* in der Menge der Borel meßbaren Funktionen) durch eine Sigma-Funktion F(x) approximiert werden

Anmerkung: Gilt auch für S = stetig, begrenzt, nicht-konstant (RBF)

Fähigkeiten der Multilayer-Netzwerke

Frage:

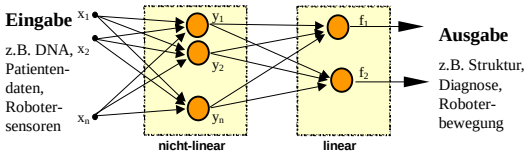
Wieviel Schichten muss ein Netzwerk mindestens haben, um eine beliebige Funktion beliebig gut zu approximieren?

? Antworten:

- eine
- zwei
- drei
- unendlich viele

Mehrschichten-Netze

Fähigkeiten von Mehrschicht-Netzen



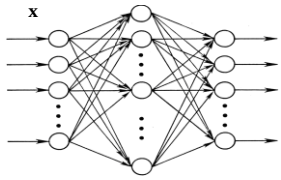
- Ein 2-Schichtennetzwerk mit nicht-linearer Ausgabefunktion S(z) kann JEDE beliebige Funktion so genau wie gewünscht approximieren, wenn genügend Neuronen ex.

Neuronenzahl gegeben. **Lernalgorithmus=?**

Backpropagation

Netzarchitektur und Aktivität

Eingabe *hidden units* Ausgabe $y_k^{(2)} = S_k\left(\sum_{j=1}^{m} w_j^{(2)}\, y_j^{(1)}\right)$



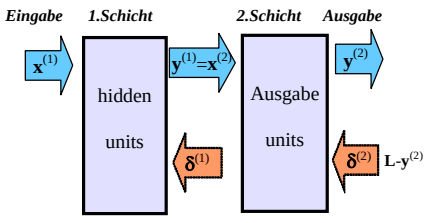
Gesamtaktivität

$$y_i^{(2)} = S_i\left(\sum_{j=1}^{m} w_j\, S\left(z_j(\mathbf{x})\right)\right)$$

$$y_j^{(1)} = S_j\left(\sum_{i=1}^{n} w_i^{(1)}\, x_i^{(1)}\right)$$

Backpropagation-Grundidee

Netzarchitektur und Lernen



Schichtweise Verbesserung durch Rückführung des Fehlers

Backpropagation-Lernregel letzte Schicht

Lernziel: $R(\mathbf{w}^*) = \min E(y(\mathbf{w}) - L(\mathbf{x}))^2$ *min.mittl. quadr. Fehler*

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \gamma \frac{\partial R}{\partial w_i}$$ *Gradienten-Lernregel*

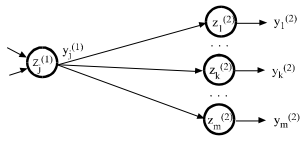
$$w_{ij}(t+1) = w_{ij}(t) - \gamma\,(y_i(w_{ij}) - L(x)) \frac{\partial S(z_i)}{\partial w_{ij}}$$ *stoch. Approximation*

mit $$\frac{\partial S(z_i)}{\partial w_{ij}} = S'(z_i)\frac{\partial z_i}{\partial w_{ij}} = S'(z_i)\frac{\partial}{\partial w_{ij}}\sum_k w_{ik}x_k = S'(z_i)x_j$$

Mit $\delta_i := -(y_i(w_{ij}) - L(x))\, S'(z_i)$

ist $\Delta w_{ij}(x) = \gamma\, \delta_i\, x_j$ *Delta-Regel*

Fehler-Backpropagation



Beeinflussung voriger Schichten $z_i^{(1)} \rightarrow R$

Delta-Regel für Schicht 1, *unabh. von R*

$$\delta_i^{(1)} = \frac{-\partial R_x}{\partial y_i^{(1)}} \frac{\partial y_i^{(1)}}{\partial z_i^{(1)}} = \left(\sum_{k=1}^{m} \delta_k^{(2)} w_{ki}^{(2)} \right) S'\left(z_i^{(1)}\right)$$

Anwendung BP

Gegeben DECtalk

Ausgabe Text → Sprache der Fa. Digital Eq. (DEC)
Aufwand 20 PJ für 95% Genauigkeit

Beispiel NetTalk

Sejnowsky, Rosenberg 1986

16 CPU-Stunden BP-Training für 98% Genauigkeit

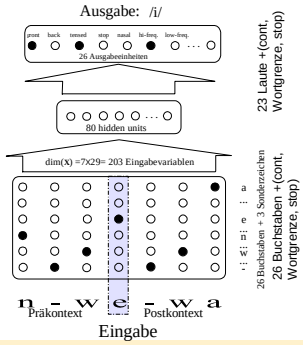
Adaptives Programm statt neu programmieren!

NetTalk: Kodierung

Ausgabekodierung
Binäre Kodierung der 26 Laute

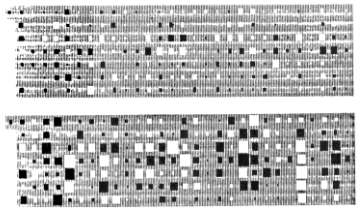
Eingabekodierung
Binäre Kodierung der 29 Buchstaben

Lauffenster der Trainingsbuchstaben



Analyse der Neuronengewichte

Visualisierung der Gewichte *Hinton Diagramm*



Gewichte von Neuron 1

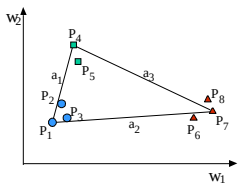
Gewichte von Neuron 2

pos. Gewichte
neg. Gewichte

Sinn = ?

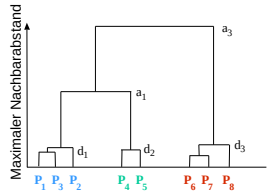
Analyse der Neuronengewichte

Clusteranalyse



- d(x,Nachbar) < d_N
  ⇒gleicher Cluster

Hierarchische Clusteranalyse



Dendrogramm

- Sukzessives Zusammenfassung
- Reihenfolge durch Cluster-Abstandsmaß

Analyse der Neuronengewichte

Clusteranalyse

- Clusterung der Muster im Eingaberaum („in vitro“)
- Clusterung der Ausgabewerte bei äquidistanten Testmustern („in vivo“)

→ Funktionsgruppen von Neuronen

Analyse der Neuronengewichte

Sensitivitätsanalyse

Aufstellen der Abhängigkeit des Fehlers des Netzes von der Eingabe bzw. den Gewichten.

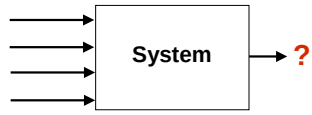
→ Wichtigkeitsliste der Eingabevariablen

Aber:

- Fehler hängt ab von Signalgrösse

→Normierung d. Signale

- Grosse Gewichte auch bei *random*-Eingabe
- Abhängigkeit von Eingabevar. nicht erfasst



Verbesserungen des BP-Algorithmus

Problem

Das System kann in einem ***lokalen*** Optimum "stecken" bleiben

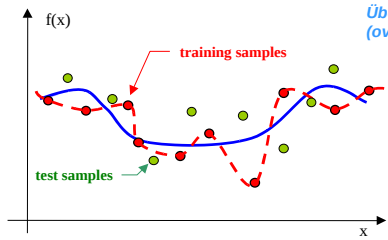
Lösung

- Gewichte der *hidden units* als Eigenvektoren initialisieren
- Mehrere Durchgänge mit zufallsveränderten Gewichten
- Regelmässige Störung der Gewichte & Neulernen
- Mit kleiner Wahrscheinlichkeit auch gegen das Optimum verändern
- Sequentieller Netzaufbau, gesteuert durch Kriterium (Ausgabeentropie, Fehler, ...)

Verbesserungen des BP-Algorithmus

Problem

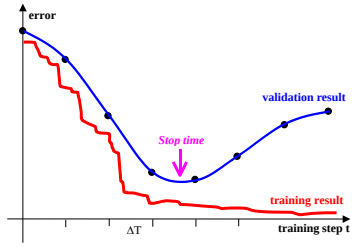
- Trotz guter Trainingsleistung zeigt der Test ***schlechte Ergebnisse***



Überanpassung (overfitting) !

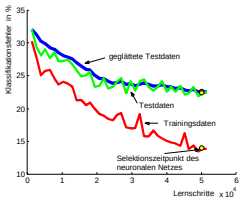
Verbesserungen des BP-Algorithmus

Lösung: ***Stopped Training***

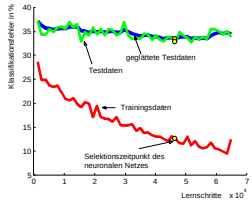


Training und Testen

- **Problem: Partition der Daten**



zufällige Einteilung der Datenmenge aller Patienten



Aufteilung der Daten nach Patienten

Verbesserungen des BP-Algorithmus

Problem $\Delta w_{ij}(x) = \gamma\, \delta_i\, x_j = \gamma\, (..) S'(z_i)\, x_j$

Die Konvergenz ist relativ ***langsam***, besonders bei mehreren Schichten

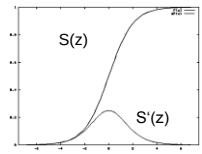
Problem Ausgabefunktion

Bei Wahl von S = **Fermifunktion**

ist die Ableitung eine Glocken-Funktion mit $S'(-\infty) = 0 = S'(\infty)$

und damit bei sehr großem oder kleinem x

$\delta(x) = 0$



$\Rightarrow$ Kein Lernen mehr möglich!

$$S'(z) = \frac{\partial}{\partial z}\frac{1}{1+e^{-z}} = \frac{+1-1+e^{-z}}{\left(1+e^{-z}\right)^2} = (1-S(z))S(z)$$

Verbesserungen des BP-Algorithmus

Abhilfen für die Ausgabefunktion:

- Andere Lernfunktion wählen, z.B. Ergänzung durch Backpropagation *Trägheitsmoment*

  $\delta(t+1) = \alpha\delta(t) + (1-\alpha)\delta(t-1)$ z.B. $\alpha = 0.9$

- **Quickprop**:
  - Addiere 0,1 zu S'(z) = (1-S)S
  - Ist die Veränderung zu klein, wird sie Null gesetzt
  - Ergänzung durch einen *Abklingterm* und *Schrittinterpolation*

  $\Delta w_{ij}(t) = -\gamma(t) R_x(t)/\partial w_{ij} - \gamma(t)\alpha w_{ij}(t-1) + \beta(t)\Delta w_{ij}(t-1)$

- Andere Ausgabefunktion wählen (z.B. Sinus)

Verbesserungen des BP-Algorithmus

Problem

Die Konvergenz ist relativ ***langsam***, besonders bei mehreren Schichten

Lösung

- Abhilfe bei Ausgabefunktion
- Wahl von $\gamma(t)$ als Hauptdiagonal-Matrix
- Änderung von $\gamma(t)$ bei Vorzeichen des Gradienten
- Lernen der Einzelschichten, falls möglich (z.B. zuerst EV bei hidden units verwenden und 2. Schicht lernen, dann 1. Schicht)
- Andere Zielfunktion wählen (Information statt erwart. quadr. Fehler MSE)

Wechsel der Zielfunktion

UNIVERSITÄT FRANKFURT AM MAIN

Zielfunktion Information

$P_{kj} = P(\omega_j|x^k)^{L_j} \, (1-P(\omega_j|x^k))^{1-L_j}$

Wahrsch. für Klassifik. von Muster *k* in Klasse *j* bei Lehrerurteil L

$$P_k = \prod_{j=1}^{M} P_{kj} = \prod_{j=1}^{M} P\left(\omega_j \middle| \mathbf{x}^k\right)^{L_j} \left(1 - P\left(\omega_j \middle| \mathbf{x}^k\right)\right)^{1-L_j}$$

Wahrsch. bei *M* Entscheidng., Muster *k* richtig zu klassifizieren

DEF $R_x := I(x^k) = -\log P_k$ *Zielfunktion* *Kap.2.6.9*

$$= -\sum_{j=1}^{M} L_j \log P(\omega_j|x^k) + (1-L_j) \log (1-P(\omega_j|x^k)) \qquad y_j = P(\omega_j|xk)$$

$$= -\sum_{j=1}^{M} L_j \log y_j + (1-L_j) \log (1-y_j) \qquad \Rightarrow \delta^{(2)} = (y-L)$$

Wechsel der Zielfunktion

UNIVERSITÄT FRANKFURT AM MAIN

Beispiel: Klassifikation von Phonemen

MSE 68% ok **Information 78% ok**