Lernen und Klassifizieren

AS1-2

www.uni-frankfurt.de

Assoziatives Lernen

Lineare Klassifikation

Lernen linearer Klassifikation

Lernen in Multilayer-Netzen

Anwendungen

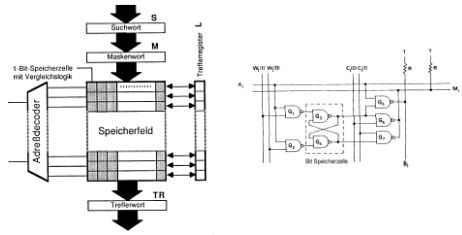
Assoziatives Lernen

- **Informationssystem:**
  Speichern und Abrufen von Information
- **RAM-Speicher**
  Speichern: Adresse A → Speicherinhalt
  Abrufen: Adresse A → Speicherinhalt

Adresse	Inhalt
1004	Text3
1003	Daten
1002	Text2
1001	Text1
1000	Text1

- **Assoziativspeicher**
  Speichern: (Adresse A, Speicherinhalt) *Assoziatives Lernen*
  Abrufen: (? , Speicherinhalt)

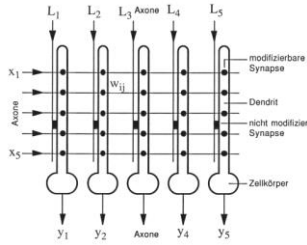
Konventionelle Assoziativspeicher



Eingabe: Suchwort, Ausgabe: Treffer in Daten (auch mehrere!)

Problem: Teile des Suchworts unbekannt oder falsch (unbekannte Maske)

Neuro-Modell des Assoziativspeichers



Funktion:

Jede Komp.ist lin. Summe

$$z_i = w_i x$$

Nichtlin. Ausgabe:

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

Lernen von **W** ?

Lernen: Hebbsche Regel

Beobachtung des Physiologen Hebb (1949):

"Wenn ein Axon der Zelle A nahe genug ist, um eine Zelle B zu erregen und wiederholt oder dauerhaft sich am Feuern beteiligt, geschieht ein Wachstumsprozess oder metabolische Änderung in einer oder beiden Zellen dergestalt, dass A's Effizienz, als eine der auf B feuernden Zellen, anwächst."

Also: $w_{AB}(t) - w_{AB}(t\text{-}1) =: \Delta w \sim x_A y_B$

oder $w_{ij}(t) = w_{ij}(t\text{-}1) + \gamma_i(t)\, y_i x_j$

Vektor: $\mathbf{w}_i(t) = \mathbf{w}_i(t\text{-}1) + \gamma_i(t)\, y_i \mathbf{x}$ *Iterative Hebb'sche Lernregel*

Matrix: $\mathbf{W}(t) = \mathbf{W}(t\text{-}1) + \gamma(t)\, \mathbf{y}\mathbf{x}^T$ *Speichern eines Tupels (x,y)*

Frage: Ist auch eine andere Form der Hebb'schen Lernregel denkbar?

Lernen im Assoziativspeicher

- **Speichern aller *N* Muster**

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

- **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.

Code eines Assoziativspeichers

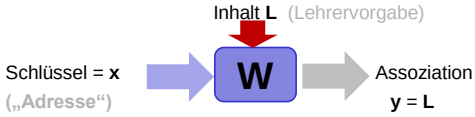
```
AMEM:   (* Implementiert einen Korrelationsspeicher *)
VAR   (* Datenstrukturen *)
   x:   ARRAY[1..n] OF REAL;          (* Eingabe *)
   y,L: ARRAY[1..m] OF REAL;          (* Ausgaben *)
   w:   ARRAY[1..m,1..n] OF REAL;     (* Gewichte *)
   γ:   REAL; (* Lernrate *);   Auslesen : BOOLEAN;
BEGIN
    γ := 0.1;                  (* Lernrate festlegen: |x|²=10 *)
   initWeights( w,0.0);        (* Gewichte initialisieren *)
   AlleMusterSpeichern ( );
   SpeicherAuslesen ( );
END AMEM.
```


Code eines Assoziativspeichers

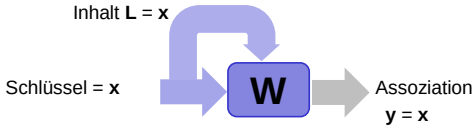
```
REPEAT                          Alle Muster speichern
   Read(PatternFile, x, L)   (* Eingabeschlüssel, gewünschte Ausgabe *)
   FOR i:=1 TO m DO          (* Für alle Neuronen *)
       FOR j:=1 TO n DO      (* ihre Gewichte verändern *)
          w[i,j] := w[i,j] + γ*L[i]*x[j]
       ENDFOR;
   ENDFOR;
UNTIL EndOf( PatternFile)

Speicher auslesen   (* zu Schlüssel x das gespeicherte y assoziieren *)
Input (x)
FOR i:=1 TO m DO      (* Ausgabe für alle Neuronen *)
    y[i] := S(z(w[i],x))
ENDFOR;
Print (y)
```

Speicherarten

- **Heteroassoziativer Speicher**

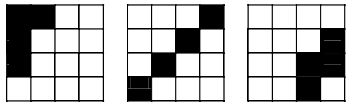


- **Autoassoziativer Speicher**



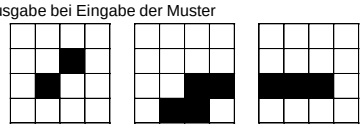
Beispiel Autoassoziative Ergänzung

Beispiel: N = 3 gespeicherte, orthogonale Muster



x^1 = (1100 1000 1000 0000) x^2 = (0001 0010 0100 1000) x^3 = (0000 0001 0011 0010)

Ausgabe bei Eingabe der Muster



x = (0000 0010 0100 0000) x = (0000 0000 0011 0110) x = (0000 0000 1110 0000)

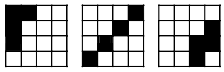
=?

Beispiel Autoassoziative Ergänzung

Mit der Hebb'schen Regel wird die Gewichtsmatrix

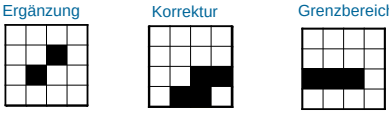
$$\mathbf{W} = \sum_{k=1}^{3} \gamma_k \mathbf{L}^k \mathbf{x}^{Tk} = \mathbf{x}^1\mathbf{x}^{1T} + \mathbf{x}^2\mathbf{x}^{2T} + \mathbf{x}^3\mathbf{x}^{3T}$$ und die Ausgabe

$$\mathbf{z} = \sum_{k=1}^{3} \gamma_k \mathbf{L}^k \left(\mathbf{x}^k\right)^T \mathbf{x} = \mathbf{x}^1(\mathbf{x}^{1T}\mathbf{x}) + \mathbf{x}^2(\mathbf{x}^{2T}\mathbf{x}) + \mathbf{x}^3(\mathbf{x}^{3T}\mathbf{x})$$



Testmuster 1: $= \mathbf{x}^1 \cdot 0 + \mathbf{x}^2 \cdot 2 + \mathbf{x}^3 \cdot 0$
Testmuster 2: $= \mathbf{x}^1 \cdot 0 + \mathbf{x}^2 \cdot 0 + \mathbf{x}^3 \cdot 3$
Testmuster 3: $= \mathbf{x}^1 \cdot 1 + \mathbf{x}^2 \cdot 1 + \mathbf{x}^3 \cdot 1$

Ergänzung Korrektur Grenzbereich



Beispiel Autoassoziative Ergänzung

Setze **L**(**x**) = **x**, lerne alle Muster (→ symm. Matrix W).

Beispiel: Buchstaben, kodiert mit 0 und 1

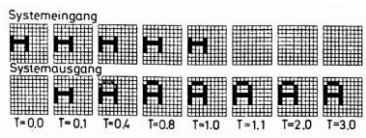
x_A = (0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 ....)

Beispiel Autoassoziative Ergänzung

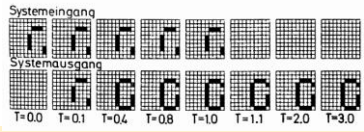
Gebe **Teil**muster von **x** ein ⇒ erhalte **Gesamt**muster L=**x**

Teil von

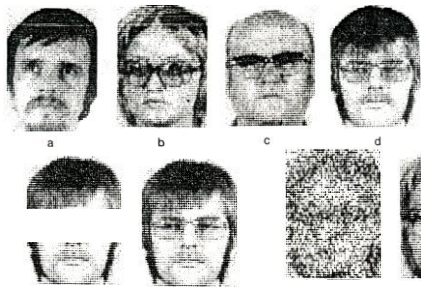
A



G



Beispiel Autoassoziative Ergänzung



Kohonen et al. 1977

3024 pixel je 3 Bit, 100 Prototypen, orthogonalisiert

Gebe **Teil**muster von **x** ein ⇒ erhalte **Gesamt**muster L=**x**

Gebe **gestörtes** Muster von **x** ein ⇒ erhalte **Gesamt**muster L=**x**

UNIVERSITÄT FRANKFURT AM MAIN

Assoziatives Lernen

Lineare Klassifikation

Lernen linearer Klassifikation

Lernen in Multilayer-Netzen

Anwendungen

Klassenbildung

UNIVERSITÄT FRANKFURT AM MAIN

Erfahrung: Es gibt **ähnliche Dinge**, „Arten“, „Klassen“,
z.B. Buchstabe A

? Woher **kommt das** ?

Plato: ***Ideen angeboren***	**Aristoteles:** ***Ideen erworben***
Ideenlehre: **Dinge in Reinstform von der Seele im Jenseits gesehen,** **Erfahrung davon = „wie Schatten an einer Wand“** **(Höhlenmetapher)**	***Zuerst*** werden Dinge mit den Sinnen erfaßt, ***dann*** die Idee dazu entwickelt

Klassenbildung: Beispiel Iris

UNIVERSITÄT FRANKFURT AM MAIN

Klasse1: ***Iris Setosa***

Klasse 2: ***Iris Virginica***

Klassenbildung heute

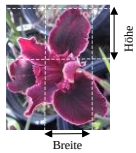
Objekte werden durch **Merkmale** beschrieben

z.B. ***qualitativ*** Mensch = (groß, braune Augen, dunkle Haare, nett, ...)

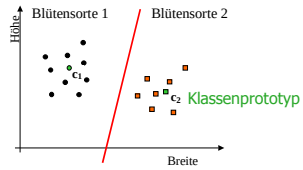
quantitativ Mensch = (Größe=1,80m, Augenfarbe=2, Haarfarbe=7, ...)

Idee = Form = „Klassenprototyp“

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



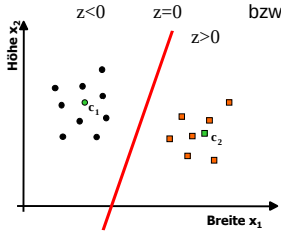
Trennung von **Klassen**



Klassifizierung = Ermitteln der Geradengleichung bzw Parameter c_1, c_2.

Klassentrennung

Klassentrennung durch **Trenngerade**



mit $f(x_1) = x_2 = w_1 x_1 + w_3$

bzw. $z := w_1 x_1 + w_2 x_2 + w_3 x_3 = 0$

mit $w_2 := -1$, $x_3 := 1$

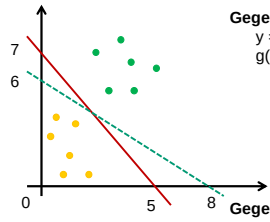
$$\text{Mit } z = \sum_{i=1}^{n} w_i x_i = \mathbf{w}^T \mathbf{x}$$

Klassenentscheidung

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

Beispiel

Klassengrenze ⇔ Gewichtsvektor



Gegeben: Klassengrenze. Gewichte?

$y = ax + b = \Delta y/\Delta x\, x + b = -7/5\, x + 7$

$g(x,y) = 0 = -7/5\, x - 1y + 7$

$= (-7/5, -1, 7) \cdot (x, y, 1)^T$

$= (7, 5, -35) \cdot (x, y, 1)^T$

$\mathbf{w}^T \cdot \mathbf{x}$

Gegeben: Gewichte. Klassengrenze?

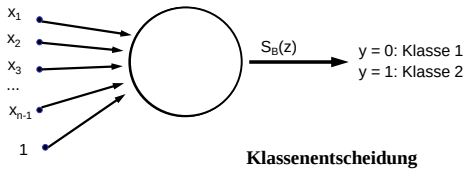
$\mathbf{w} = (6, 8, -48)$

$g(x_1,x_2) = 0 = 6x_1 + 8x_2 - 48$

$= -6/8\, x_1 - 1x_2 + 6 \Rightarrow a=-6/8,\ b=6$

Klassentrennung durch formales Neuron

Klassentrennung durch binäres Neuron



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

Klassenentscheidung

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

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 $g(\mathbf{x}) > 0$.

***Frage**: Und welche Klasse haben die $\mathbf{x}^*$?*

ABER:

WIE erhalten wir die richtigen Gewichte,

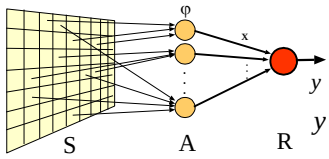
d.h. die richtige Klassifizierung

?

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

UNIVERSITÄT FRANKFURT AM MAIN

Lernen

Ziel: Wähle $\mathbf{w}$ so, dass für alle $\mathbf{x}$ gilt

$$y = \begin{cases} \text{TRUE} & \text{wenn } x \in \Omega_1 \\ \text{FALSE} & \text{sonst} \end{cases} \quad \Leftrightarrow \quad y = \begin{cases} \text{TRUE} & \text{wenn } \mathbf{w}^T\mathbf{x} > 0 \\ \text{FALSE} & \text{wenn } \mathbf{w}^T\mathbf{x} \leq 0 \end{cases}$$

Methode: Für alle x aus Ω_1 und $\mathbf{w}^T\mathbf{x} < 0$

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

Erhöhung von $\mathbf{w}^T\mathbf{x}$!

Das Perzeptron: Pseudo-code 1

UNIVERSITÄT FRANKFURT AM MAIN

PERCEPT1:

Wähle zufällige Gewichte $\mathbf{w}$ zum Zeitpunkt t:=0.
REPEAT
  Wähle zufällig ein Muster $\mathbf{x}$ aus $\Omega_1 \cup \Omega_2$; `t:= t+1;`
  IF ($\mathbf{x}$ aus Klasse Ω_1)
    THEN **IF** $\mathbf{w}^T\mathbf{x} \leq 0$
      THEN $\mathbf{w} = \mathbf{w} + \gamma\mathbf{x}$
      ELSE $\mathbf{w} = \mathbf{w}$
    END
    ELSE **IF** $\mathbf{w}^T\mathbf{x} > 0$
      THEN $\mathbf{w} = \mathbf{w} - \gamma\mathbf{x}$
      ELSE $\mathbf{w} = \mathbf{w}$
    END
  ENDIF
UNTIL (alle $\mathbf{x}$ richtig klassifiziert)

Das Perzeptron: Pseudo-code 2

UNIVERSITÄT FRANKFURT AM MAIN

DEF $\Omega^- := \{\mathbf{x} \mid -\mathbf{x} \text{ aus Klasse } \Omega_2\}$

$\Rightarrow$ statt $\mathbf{w}^T\mathbf{x} \leq 0$ gilt für Ω^- die Relation $\mathbf{w}^T\mathbf{x} \geq 0$

PERCEPT2:

Wähle zufällige Gewichte $\mathbf{w}$ zum Zeitpunkt t:=0.
REPEAT
  Wähle zufällig ein Muster $\mathbf{x}$ aus $\Omega_1 \cup \Omega^-$; t:= t+1;
  IF $\mathbf{w}^T\mathbf{x} \leq 0$
    THEN $\mathbf{w}(t) = \mathbf{w}(t-1) + \gamma\mathbf{x}$
    ELSE $\mathbf{w}(t) = \mathbf{w}(t-1)$
  END
UNTIL (alle $\mathbf{x}$ richtig klassifiziert)

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

Satz:

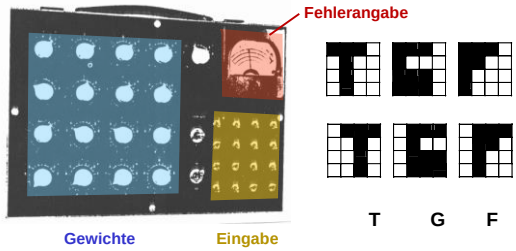
Ein Perzeptron kann nur Klassen trennen, wenn sie linear separierbar sind.

Warum?

Adaline

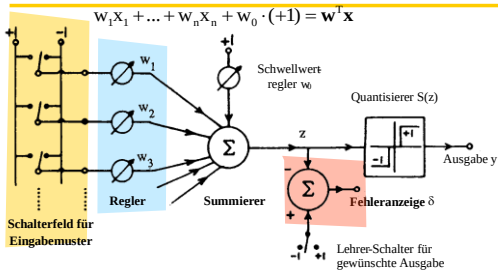
***ADA*ptive *LIN*ear *E*lement** Widrow und Hoff (1960)

Diskrete „Lernmaschine“ aus Widerständen und Leitungen



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: Lernalgorithmus

Minimierung des erwarteten Fehlers durch Anpassung der Parameter (Drehwiderstände):

- Wenn der Fehler größer wird, drehe Knopf zurück und in die entgegengesetzte Richtung
- Wenn der Fehler am kleinsten ist, wende dich nächstem Knopf zu

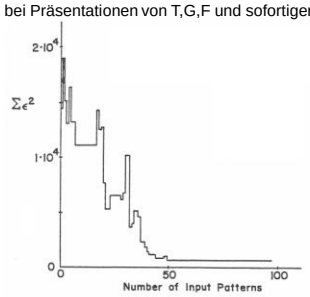
Automatische Anpassung: *Lerngleichung*

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \gamma(t)\underbrace{(L(\mathbf{x}) - \mathbf{w}^T\mathbf{x})}_{Fehler}\frac{\mathbf{x}}{|\mathbf{x}|^2}$$

Widrow-Hoff Lernregel

Adaline: Aktivität

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



Adaline: Pseudocode

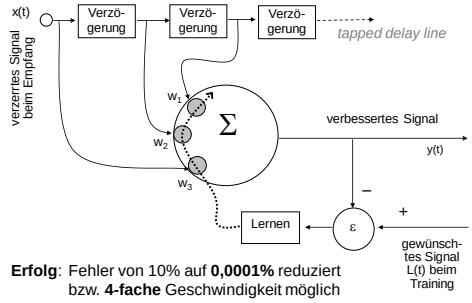
```
VAR   (* Datenstrukturen *)
x:      ARRAY[1..n] OF REAL;          (* Eingabe *)
z,y,L:  ARRAY[1..m] OF REAL;          (* IST und SOLL-Ausgaben *)
w:      ARRAY[1..m,1..n] OF REAL;     (* Gewichte *)
γ:      REAL                          (* Lernrate *);  x2:  REAL;

BEGIN
γ:= 0.1;                              (* Lernrate festlegen *)
initWeights(w,0.0);                   (* Gewichte initialisieren *)
```


Adaline: Pseudocode

```
REPEAT
  Read( PatternFile,x,L)              (* Eingabe *)
  x2 := Z(x,x)                        (* |x|²*)
(* Aktivität bilden im Netz *)
  FOR i:=1 TO m DO                    (* Ausgabe für alle Neuronen *)
    z[i] :=  Z(w[i],x)                (*  Aktivität errechnen*)
    y[i] :=  S(z[i])                  (*  Nicht-lin. Ausgabe *)
  END;
(* Lernen der Gewichte *)
  FOR i:=1 TO m DO                    (* Für alle Neuronen *)
    FOR j:=1 TO n DO                  (* und alle Dimensionen *)
      w[i,j] := w[i,j]-γ*(z[i]-L[i])*x[j]/x2    (* Gewichte verändern *)
    END;
  END;
UNTIL EndOf(PatternFile)
```


Adaline: Anwendung

Aufgabe: Korrektur des Signals bei Transatlantik-Kommunikation

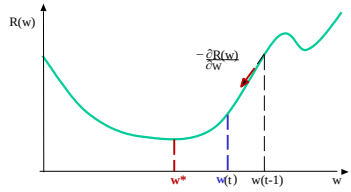


Erfolg: Fehler von 10% auf **0,0001%** reduziert bzw. **4-fache** Geschwindigkeit möglich

Lernen durch Iteration

UNIVERSITÄT FRANKFURT AM MAIN

Gradientenabstieg



$\nabla_{\cdot} := (\partial/\partial w_1, \ldots, \partial/\partial w_n)^T$

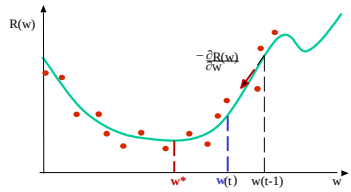
$$\Delta \mathbf{w} := (\mathbf{w}(t-1) - \mathbf{w}(t)) \sim -\nabla_{\mathbf{w}} R(\mathbf{w}(t-1))$$ *multi-dim. Ableitung*

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

Lernen durch Iteration

UNIVERSITÄT FRANKFURT AM MAIN

Problem: stochastischer Gradientenabstieg



Zielfunktion abhängig von stochastischer Beobachtung x(t)

Stochastisches Lernen

UNIVERSITÄT FRANKFURT AM MAIN

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

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

wird ersetzt durch

Lernen mit stochast. Zielfunktion r(w,x)

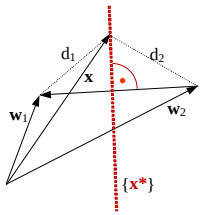
$$\mathbf{w}(t) = \mathbf{w}(t\text{-}1) - \gamma(t)\, \nabla_{\mathbf{w}} r(\mathbf{w}(t\text{-}1), \mathbf{x}(t))$$ *stochastisches Lernen*

Stochastisches Lernen

UNIVERSITÄT FRANKFURT AM MAIN

Beispiel Klassentrennung

Zielfunktion $r(\mathbf{w},\mathbf{x}) := \frac{1}{2}(\mathbf{w}-\mathbf{x})^2, \quad \gamma(t) := 1/t$



Klassifizierung

$r(\mathbf{w}_1,\mathbf{x}) < r(\mathbf{w}_2,\mathbf{x}) \quad \Rightarrow \mathbf{x}$ aus Klasse 1
$r(\mathbf{w}_1,\mathbf{x}) > r(\mathbf{w}_2,\mathbf{x}) \quad \Rightarrow \mathbf{x}$ aus Klasse 2

Klassengrenze $\{\mathbf{x}^*\}$

$r(\mathbf{w}_1,\mathbf{x}^*) = r(\mathbf{w}_2,\mathbf{x}^*)$
$|\mathbf{w}_1-\mathbf{x}^*| = d_1 = d_2 = |\mathbf{w}_2-\mathbf{x}^*|$

Lernen *für x aus Klasse i*

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

Codebeispiel Klassentrennung

UNIVERSITÄT FRANKFURT AM MAIN

```
float w[ ][ ] = new float[2][2];    float x[2];   float γ;  int t, k;

t = 1;                          (* erster Zeitschritt *)
do {
(* Eingabe oder Generation des Trainingsmusters *)
    read(PatternFile, x);
    γ = 1/t;                    (* zeitabh. Lernrate  *)
(*suche Klasse mit minimalem Abstand *)
   if (Abstand(x,w[0]) > Abstand(x,w[1]))
            k= 1;
     else   k= 0;
(* verändere den Gewichtsvektor *)
   for (int i=0; i<=1; i++)  (* Approximation des Mittelwerts *)
      w[k,i] = w[k,i] - γ *(w[k,i]-x[i]);
    t = t+1;                    (* nächster Zeitschritt, nächstes Muster *)
} while ( !EndOf (PatternFile) );
```

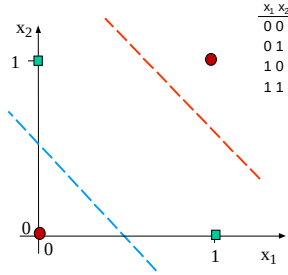

UNIVERSITÄT FRANKFURT AM MAIN

Das XOR-Problem

UNIVERSITÄT FRANKFURT AM MAIN

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

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

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

Klassen **nicht** linear separierbar!

Das XOR-Problem

UNIVERSITÄT FRANKFURT AM MAIN

Lösung

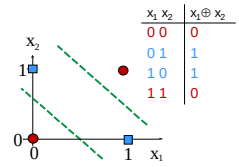
Trennung durch **zwei** Schichten

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

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

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

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



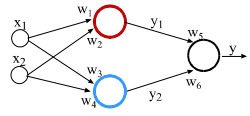
x_1 x_2	$x_1 \oplus x_2$
0 0	0
0 1	1
1 0	1
1 1	0

z.B. formale binäre Neuronen
$S(z \geq s) = 1$, $S(z < s) = 0$

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

$w_2 = w_3 = -1/3$

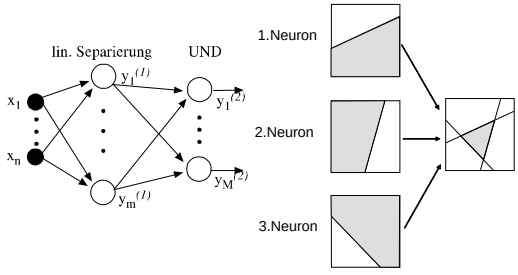
$s_1 = s_2 = 0$, $s = 1/2$



Multilayer-Klassifikation

UNIVERSITÄT FRANKFURT AM MAIN

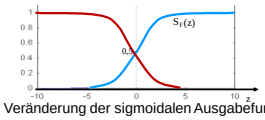
Visualisierung: Separierung von Klassen



Multilayer-Klassifikation

UNIVERSITÄT FRANKFURT AM MAIN

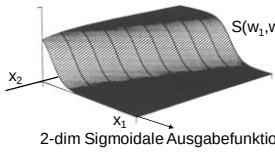
„weiche“ Separierung von Klassen



$$S_F(z) := \frac{1}{1+e^{-kz}}$$

K > 0 K < 0

Veränderung der sigmoidalen Ausgabefunktion



$z = \mathbf{x}^T\mathbf{w}$

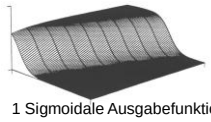
$\mathbf{x} = (x_1,x_2,1)$
$\mathbf{w} = (w_1,w_2,w_3)$

2-dim Sigmoidale Ausgabefunktion

Multilayer-Klassifikation

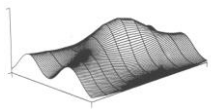
UNIVERSITÄT FRANKFURT AM MAIN

„weiche“ Separierung von Klassen

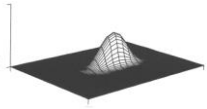


1 Sigmoidale Ausgabefunktion

2 überlagerte sigm.Ausgaben



4 Sigmoidale Ausgaben

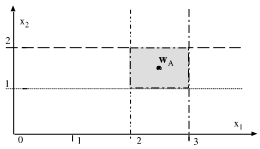


mit Schwelle

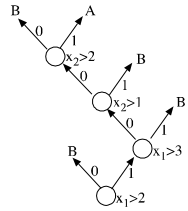
Lernen von Klassifikation

UNIVERSITÄT FRANKFURT AM MAIN

Wenn Daten bekannt:
Erstellen von Entscheidungsbäumen



Klassentrennung durch Hyperbenen



Klassentrennung durch Entscheidungsbaum

Lernen von Klassifikation

UNIVERSITÄT FRANKFURT AM MAIN

Wenn Daten unbekannt: Sequentielle Netzerstellung

Vorschlag 1 *ohne Training*

- Ordne Muster x(t) ein. Falsche Klasse: Erzeuge neues Neuron so, dass richtig klassifiziert wird.
- Sind gleiche Klassen benachbart, verschmelze sie.

Vorschlag 2 *mit Einzeltraining*

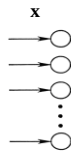
- Trainiere Neuron 1 mit Ω_1. Bilde $\Omega_1/\{x|$ x wurde für Klasse 1 erkannt$\}$
- Trainiere Neuron 2 mit Ω_1. Bilde $\Omega_1/\{x|$ x wurde für Klasse 1 erkannt$\}$
- ... bis Ω_1 leer.
- Trainiere Neuron n_1+1 mit Ω_2. Bilde $\Omega_2/\{x|$ x wurde für Klasse 2 erkannt$\}$
- ... bis Ω_2 leer, usw.
- Identifiziere **y** der Ausgabe mit **x** der nächsten Schicht.
- STOP, wenn für jede Klasse nur noch ein Neuron ex.

Backpropagation

UNIVERSITÄT FRANKFURT AM MAIN

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

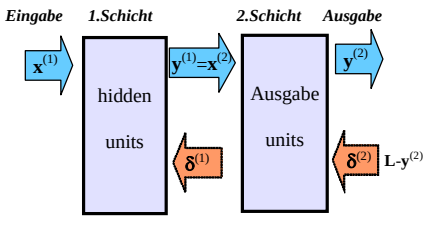
$$y_i^{(2)} = S_i\left(\sum_{j=1}^{m} w_j\, S\left(z_j(\mathbf{x})\right)\right)$$

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

Backpropagation-Grundidee

UNIVERSITÄT FRANKFURT AM MAIN

Netzarchitektur und Lernen



Schichtweise Verbesserung durch Rückführung des Fehlers

Backpropagation-Lernregel

Lernziel: $R(\mathbf{w}^*) = \min \langle (y(\mathbf{w}) - L(\mathbf{x}))^2 \rangle$ *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 $\delta_i^{(2)} := -(y_i^{(2)} - L(x))\,S'(z_i)$ **Schicht 2**

ist

$$w_{ij}(t+1) = w_{ij}(t) - \gamma\,\delta_i\,x_j$$ ***Delta-Regel***

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

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

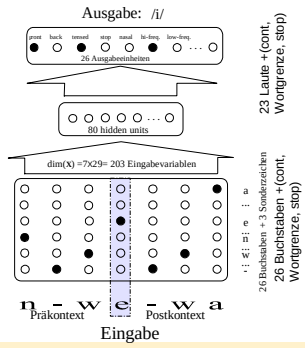
Adaptives Programm statt neu programmieren!

NetTalk: Kodierung

Ausgabekodierung
Binäre Kodierung der 26 Laute

Eingabekodierung
Binäre Kodierung der 29 Buchstaben

Lauffenster der Trainingsbuchstaben



NetTalk: Training

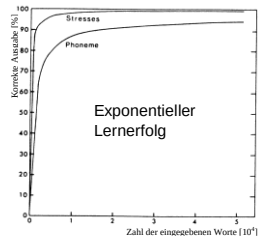
UNIVERSITÄT FRANKFURT AM MAIN

Training

- transkribiertes Wörterbuch 20.000 Einträge
- Protokollierte Kindersätze

Ergebnis

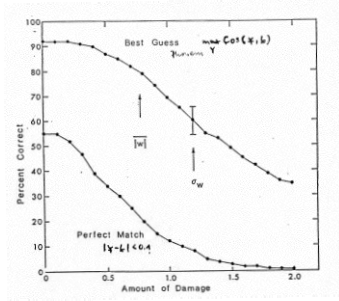
- Trennung der Konsonanten von Vokalen („Babbeln“)
- Entwicklung der Wortgrenzen („Pseudoworte“)
- Verständliche Sprache (10xTraining pro Wort)



NetTalk: gestörte Gewichte

UNIVERSITÄT FRANKFURT AM MAIN

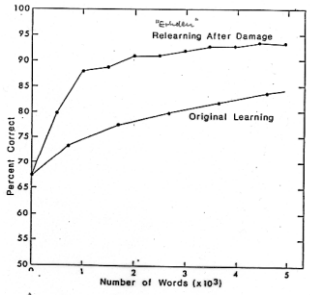
Störung durch normalverteiltes Rauschen



Neulernen der Gewichte

UNIVERSITÄT FRANKFURT AM MAIN

Schnelleres Lernen „verlernter“ Inhalte

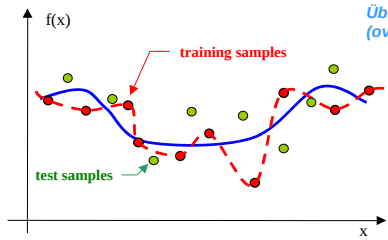


Verbesserungen des BP-Algorithmus

UNIVERSITÄT FRANKFURT AM MAIN

Problem

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

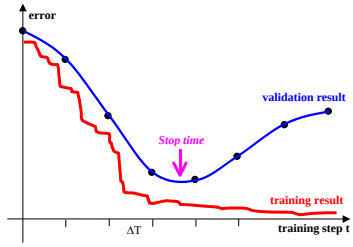


Überanpassung (overfitting) !

Verbesserungen des BP-Algorithmus

UNIVERSITÄT FRANKFURT AM MAIN

Lösung: ***Stopped Training***



UNIVERSITÄT FRANKFURT AM MAIN

Assoziatives Lernen

Lineare Klassifikation

Lernen linearer Klassifikation

Lernen in Multilayer-Netzen

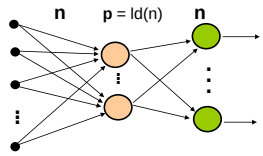
Anwendungen

Anwendung BP

UNIVERSITÄT FRANKFURT AM MAIN

Binäre Kodierung

Ziel:
n Zustände in ld(*n*) Bits kodieren



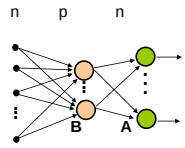
Ergebnis:

10000000	**0,5** 0 0	10000000
01000000	0 1 0	01000000
00100000	1 1 0	00100000
00010000	1 1 1	00010000
00001000	0 1 1	00001000
00000100	**0,5** 0 1	00000100
00000010	1 0 **0,5**	00000010
00000001	0 0 **0,5**	00000001

Analyse der Neuronengewichte

UNIVERSITÄT FRANKFURT AM MAIN

Hauptkomponentenanalyse



Lin. Approximation
(1. Glied Taylorentwicklung)

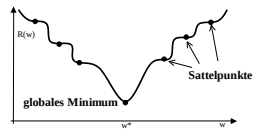
Beispiel: n-p-n Netz Kodierer

$\mathbf{y} = \mathbf{A}_{pxn}\ \mathbf{B}_{nxp}\ \mathbf{x}$

- Min. quadr. Fehler bei globalem Minimum $\Rightarrow$ **A**, **B**
- **B** besteht aus *p* **Eigenvektoren** der **Kovarianzmatrix**

$$C_{xx} = \langle (\mathbf{x}-\langle\mathbf{x}\rangle)(\mathbf{x}-\langle\mathbf{x}\rangle)^T \rangle$$

$$(C_{ij}) = \left(\frac{1}{N} \sum_{k=1}^{N} \left(x_i^k - \langle x_i^k \rangle \right) \left(x_j^k - \langle x_j^k \rangle \right) \right)$$



SNOOPE

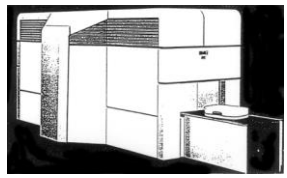
UNIVERSITÄT FRANKFURT AM MAIN

System for **N**uclear **O**n-line **O**bservation of **P**otential **E**xplosives, Science Appl. Int. Corp SAIC, 1989

Entdeckung von Plastiksprengstoff in Fluggepäck

Eingabe: Gepäckstück

Ausgabe: Diagnose „gefährlich“ oder „ungefährlich“

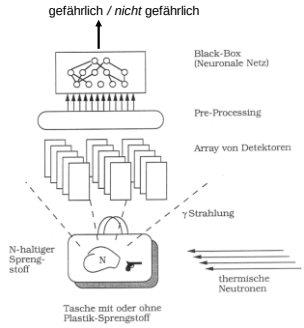


SNOOPE

UNIVERSITÄT FRANKFURT AM MAIN

Funktionsprinzip

- Thermische Neutronen: ungefährlich für Filme und Magnetmedien, aber regt Stickstoff an.
- Training + Test eines BP-Netzwerks mit Gepäckstücken
- Keine menschl. Interpretation nötig



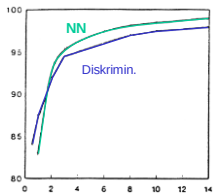
SNOOPE

UNIVERSITÄT FRANKFURT AM MAIN

Ergebnisse

Leistung des BP-NN vs. Diskriminantenanalyse

Prob. Entdeckung (Spezifität)



Prob. Fehlalarm %
= 1-Sensitivität

Problembereich: Sehr kleiner % Fehlalarme (sehr kleine Aktivitätsfunktion)

UNIVERSITÄT FRANKFURT AM MAIN

www.uni-frankfurt.de

Backprop-Anwendung

Roboter-fahrzeuge

ALVINN

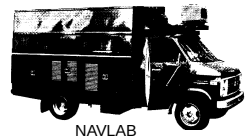
UNIVERSITÄT FRANKFURT AM MAIN

Autonomous **L**and **V**ehicle **i**n a **N**eural **N**etwork

Touretzky, Pomerleau 1989

Team: Carnegie-Mellon University, Pittburgh, USA

Methode: 2-Schicht BP



NAVLAB

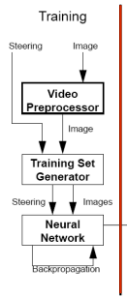
Resultat: Automatisches Fahren auf Uni-Gelände mit ca. 5 km/h (3,5 mph), besser als tradit. Methoden (3 mph)

ALVINN

UNIVERSITÄT FRANKFURT AM MAIN

Training/Testschema

- **Training** auf stationärem Supercomputer mit Aufzeichnungen
- **Test** in der Realität (SUN-3/160

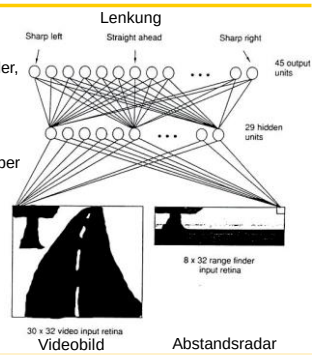


ALVINN

UNIVERSITÄT FRANKFURT AM MAIN

Training

- 1200 simulierte Strassenbilder, 40x präsentiert
- Training auf Supercomputer (100M FLOPS)
- 20 Mill. connection updates per second CUPS



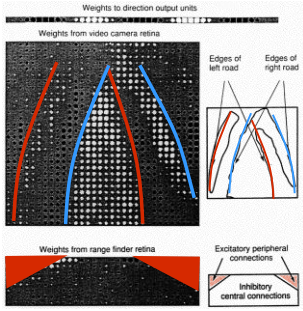
Videobild

Abstandsradar

ALVINN

UNIVERSITÄT FRANKFURT AM MAIN

Analyse der *hidden units*



Visualisierung der Gewichte einer unit

- Einprägung der Fahrbahnmarkierung
- Einprägung der Hindernisse vom Rand

Roboterwettbewerb

UNIVERSITÄT FRANKFURT AM MAIN

DARPA Grand Challenge 2005: 2M$

Autonome Fahrt 211,1km

(Mojave-Wüste, Las Vegas, Nevada)

durch Tunnel, über Sandwege, Serpentinen, Gleise, Kuhpassagen, an fahrenden Autos vorbei ....

Hauptkonkurrenten

UNIVERSITÄT FRANKFURT AM MAIN

Team: „Red Team“

Carnegie-Mellon-University, Pittsburgh, Hummer-Geländewagen

7:04 Std.

Methode: Sensordaten des Terrain für 3200km registriert + GPS verwendet.

Finanzen: **12 M$:** Boing, Caterpillar, Science Int.SAIC, etc., 20 Angestellte (50P),

Technik: 75m Laser-scan Kohlenfaserkuppel, gekühlt, 6 Scanner, 7 Pentium-M+3 PC/104-Module für 300MB/min Daten, 80 Module, 1,5M.Codezeilen (Linux).

http://www.redteamracing.org/

Sieger

Team: „Stanford Racing Team“ Stanford University, VW Touareg 6:53 Std.

Methode: Route: GPS-Punkte + Realtime-Daten dazwischen.

Adapt.Prob. Modell für Fahrtechnik: trainiert an einem menschl. Fahrer.

12%Fehler ->0,02%

Technik: 5 Sick-Sensoren, GPS, 2 Radarsensoren + 1 Dachkamera, 6 Pentium-M Systeme für 4MB/Min Daten. 31 Module, (Linux), 0,1M Codezeilen

Finanzen: **0,5M$** VW, Intel, ...

http://www.stanfordracing.org/

Ausblick Autonome Fahrzeuge

- **Mehr Sicherheit** – aber: wer ist schuld bei Unfall?
- **Weniger Staus**
- **Weniger Spritverbrauch** – 10-15%, im Stau 30-40%
- **Car Sharing** – statt 43 Mio nur 4 Mio Fahrzeuge

Bereits im Jahr 2010 hängte ein führerloser Audi auf einer Bergstrecke im US-Bundesstaat Colorado menschliche Spitzenrennfahrer ab.