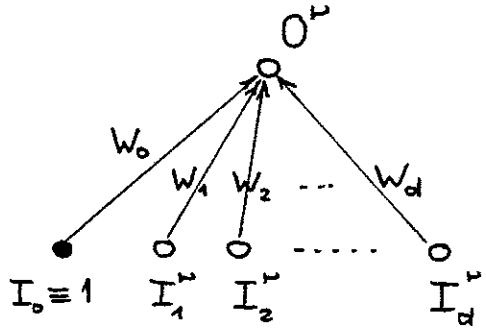
V. PERCEPTRONS

Literatur:
- Rosenblatt (1958, 1960), Minsky & Papert (1969)
- N.J. Nilsson, "Learning Machines", McGraw-Hill (1965)

V.1 Definition, Geometrische Interpretation

Ein Perceptron ist ein Feedforward-Netzwerk, das keine Hidden Units enthält. Es besteht also nur aus einer Schicht Input-Neuronen und einer Schicht Output-Neuronen.

Da die Outputs unabhängig voneinander sind, genügt es, sich auf Perceptrons mit nur einem Output zu beschränken:



W_i reell, $\gtrless 0$

W_0 = Schwelle $[I_0 = 1 \ \forall \mu]$

I_i^μ reell (ev. nur ± 1)

- Die Inputs sind Punkte $\{I_1^\mu, \dots, I_d^\mu\}$ in $\mathbb{R}^d$, d.h. im d-dim. Euklidischen Raum, $\mu = 1, \dots, N$.

- Vorgegebene Klassifizierung der N Inputpunkte:

$$D^\mu = \pm 1 \quad , \quad \mu = 1, \dots, N$$

[Klassifizierung in 2 Klassen: → DICHOTOMIE]

- Problem: Bestimme $W_1, \ldots, W_d$ und W_0 (Schwelle) so, dass $O^\nu = D^\nu$ für $\nu = 1, \ldots, N$.

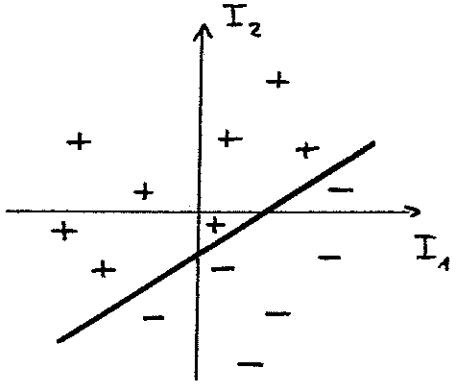
$$O^\nu = \operatorname{sign}\left(\sum_{i=1}^{d} W_i I_i^\nu + W_0\right)$$

$$= \begin{cases} +1 & \text{falls } \sum_{i=1}^{d} W_i I_i^\nu + W_0 \geq 0 \\ -1 & " \quad " \quad < 0 \end{cases}$$

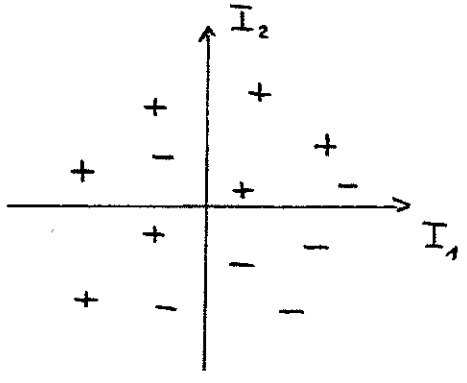
- Die Gleichung $\sum_{i=1}^{d} W_i I_i + W_0 = 0$ definiert eine $(d-1)$-dimensionale Hyperebene im d-dimensionalen Inputraum.

→ *Klassifizierung mit Perceptron nur möglich, wenn die Klassen "+" und "−" durch eine Hyperebene getrennt werden können.*
[LINEARE SEPARABILITÄT]

Beispiel $(d=2)$:



linear separabel



nicht linear separabel

V.2 Kapazität eines Perceptrons

- N Punkte $\underline{I}^1, \ldots, \underline{I}^N$ im d-dim Euklid'schen Raum

  → 2^N mögliche Dichotomien

  → Wieviele davon sind linear separabel?

- $L(N,d)$ = Zahl der linear separablen Dichotomien von N Punkten in <u>allgemeiner Lage</u> in d Dimensionen.

  <u>Allgemeine Lage</u>: $N > d$: Keine Untermenge von d+1 Punkten liegt auf einer (d-1)-dim. Hyperebene

  $N \leq d$: Keine (N-2)-dim. Hyperebene enthält die N Punkte

- <u>Berechnung von $L(N,d)$</u>:

  [Cover, IEEE Trans. on Electron. Computers, Vol. EC-14, pp. 326-334 (June 1965)]

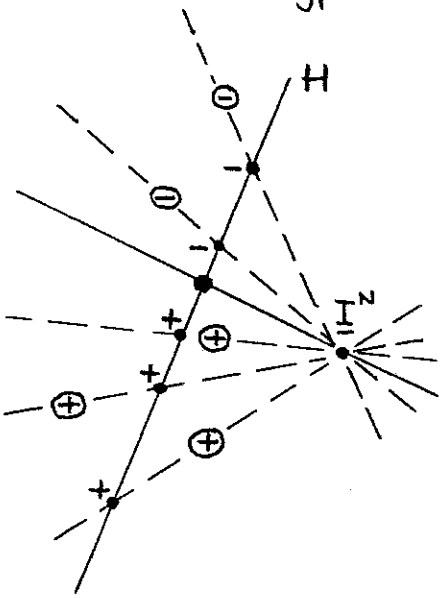
  Betrachte die $L(N-1,d)$ linear separablen Dichotomien der Punkte $\underline{I}^1, \ldots, \underline{I}^{N-1}$. Davon seien $L_{\underline{I}^N}(N-1,d)$ mit einer Hyperebene realisierbar, die durch den Punkt $\underline{I}^N$ geht.

  → $L(N,d) = \underbrace{\left[L(N-1,d) - L_{\underline{I}^N}(N-1,d)\right]}_{\underline{I}^N \text{ ist entweder in Klasse "+" oder "−"}} + \underbrace{2\, L_{\underline{I}^N}(N-1,d)}_{\underline{I}^N \text{ kann in beiden Klassen sein (beliebig kleine Verschiebung der Trennebene!)}}$

  → $L(N,d) = L(N-1,d) + L_{\underline{I}^N}(N-1,d)$

Behauptung: $L_{\underline{I}^N}(N-1, d) = L(N-1, d-1)$

Beweis: Schneide die N-1 Geraden, die durch $\underline{I}^N$ und $\underline{I}^i$ gehen ($i = 1, \ldots, N-1$), mit einer $(d-1)$-dim. Hyperebene H:



$\longrightarrow$ *N-1 Punkte im d-dim. Raum genau dann durch Hyperebene durch $\underline{I}^N$ separierbar, wenn die N-1 Projektionen auf H durch (d-2)-dim. Hyperebene trennbar sind.*

q.e.d.

$\longrightarrow$ **Rekursion:** $L(N, d) = L(N-1, d) + L(N-1, d-1)$

Offensichtliche Randbed.:

$L(1, d) = 2$, $L(N, 1) = 2N$

$\longrightarrow$

$$L(N, d) = \begin{cases} 2^N & \text{falls} \quad N \leq d \\ 2 \sum_{i=0}^{d} \binom{N-1}{i} & \text{falls} \quad N > d \end{cases}$$

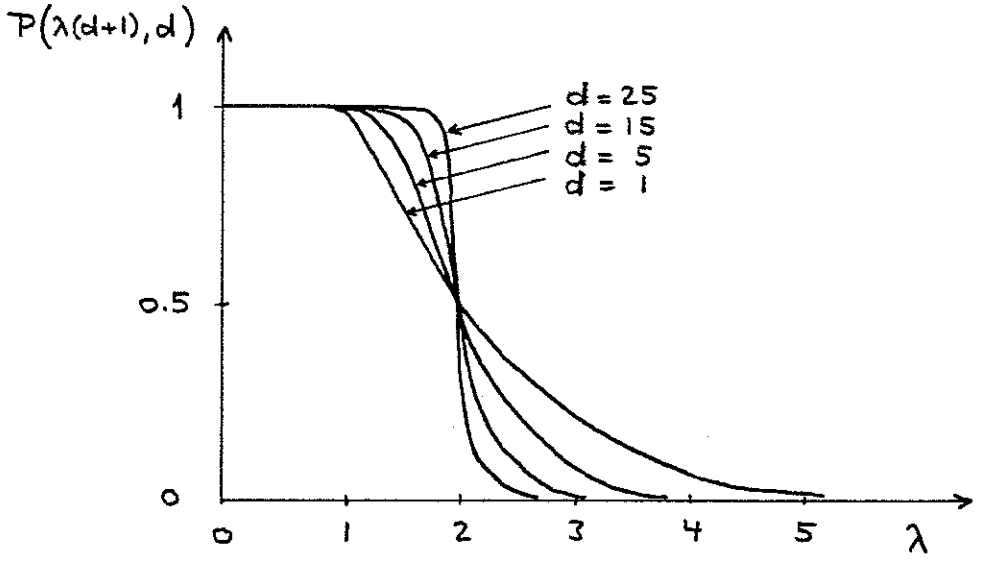
Definition:

$P(N, d)$ = Wahrsch.keit, dass eine zufällig gewählte Dichotomie von N Punkten im d-dim. Raum linear separabel ist

(d.h. durch ein Perceptron mit d Inputs und einer Schwelle realisiert werden kann).

$$\rightarrow \quad P(N,d) = \frac{L(N,d)}{2^N} = \begin{cases} 1 & \text{falls} \quad N \leq d \\ \frac{1}{2^{N-1}} \sum_{i=0}^{d} \binom{N-1}{i} & \text{falls} \quad N > d \end{cases}$$

Setze $N = \lambda(d+1)$ und betrachte $P(\lambda(d+1), d)$:



$$\rightarrow \quad P(\lambda(d+1), d) = \tfrac{1}{2} \qquad \text{für} \quad \lambda = 2$$

$$\lim_{d\to\infty} P(\lambda(d+1), d) = \begin{cases} 0 & \text{falls} \quad \lambda = 2+\varepsilon \\ 1 & \text{falls} \quad \lambda = 2-\varepsilon \end{cases}$$

$\rightarrow$ KAPAZITÄT EINES PERCEPTRONS: $\boxed{N_c = 2(d+1)}$

d.h. für grosse d gilt:

- Eine vorgegebene Dichotomie kann fast sicher durch ein Perceptron realisiert werden, falls $N < 2(d+1)$
- und fast sicher nicht, falls $N > 2(d+1)$

[N = Anzahl Lernbeispiele
d = Dimension des Inputraums (Anzahl Inputs ohne Schwelle)]

V.3 Das Perceptron-Lernverfahren

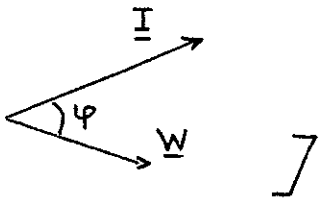
Notation: $\underline{I}^\nu = (I_0=1, I_1^\nu, I_2^\nu, \dots, I_d^\nu)$, $\nu = 1, \dots, N$

$\underline{W} = (W_0, W_1, W_2, \dots, W_d)$

$$\longrightarrow \quad O^\nu = \text{sign}\left(\underline{W}\cdot\underline{I}^\nu\right) = \begin{cases} +1 & \underline{W}\cdot\underline{I}^\nu \geq 0 \\ -1 & \underline{W}\cdot\underline{I}^\nu < 0 \end{cases}$$

[$\underline{W}\cdot\underline{I} = \sum_{i=0}^{d} W_i I_i = |\underline{W}|\cdot|\underline{I}|\cdot\cos\varphi$

→ Skalarprodukt von $(d+1)$-dimensionalen Vektoren



]

- Wie können die Gewichte $W_0, W_1, \dots, W_d$ "gelernt" werden, sodass $O^\nu = D^\nu$ $(\nu = 1, \dots, N)$, für vorgegebene Klassifizierung D^ν der Inputmuster?

→ PERCEPTRON-LERNVERFAHREN:

- Die Inputbeispiele $\underline{I}^\nu$ $(\nu = 1, \dots, N)$ werden in zufälliger Reihenfolge immer wieder präsentiert.
- Bei jedem Schritt k wird $O^\nu = \text{sign}\left(\underline{W}\cdot\underline{I}^\nu\right)$ berechnet und das aktuelle $\underline{W} = \underline{W}(k)$ wie folgt geändert:

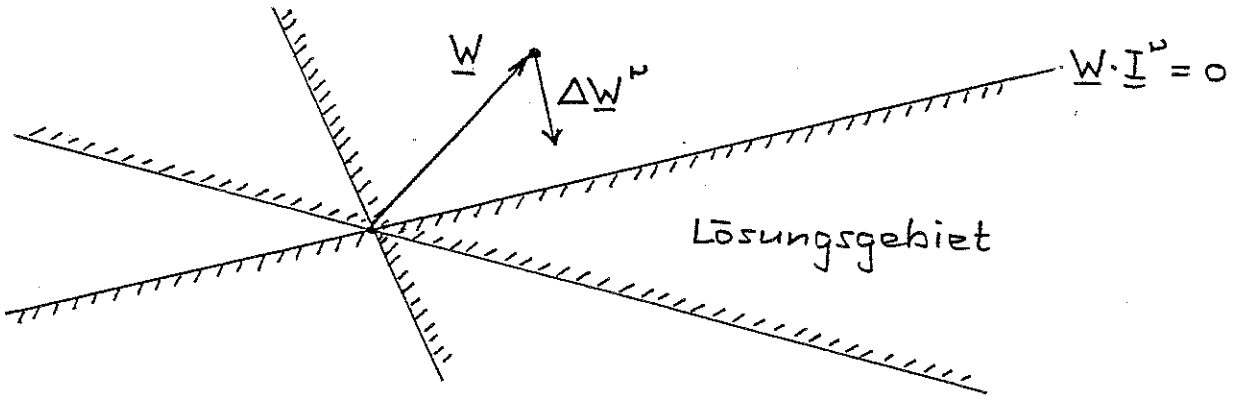
$$\underline{W}(k) \longrightarrow \underline{W}(k+1) = \underline{W}(k) + \Delta\underline{W}^\nu$$

wobei $\Delta\underline{W}^\nu = \eta\left(D^\nu - O^\nu\right)\underline{I}^\nu$, $\eta = \text{const}$

$$\left[\ \Delta W_i^\nu = \eta\left(D^\nu - O^\nu\right) I_i^\nu \ , \quad i = 0, 1, \dots, d\ \right]$$

Konvergenzbeweis (geometrisch):

Voraussetzung: Es existiere ein Lösungsgebiet im $\underline{W}$-Raum. (Dieses ist durch Hyperebenen $\underline{W} \cdot \underline{I}^{\nu} = 0$ begrenzt.)



Falls $O^{\nu}(\underline{W}) \neq D^{\nu}$, so wird $\underline{W}$ bei Präsentation des Lernbeispiels ν um $\Delta\underline{W}^{\nu} = \eta (D^{\nu} - O^{\nu}) \underline{I}^{\nu}$ geändert.

Es ist leicht einzusehen, dass $\Delta\underline{W}^{\nu}$ senkrecht zur Hyperebene $\underline{W} \cdot \underline{I}^{\nu} = 0$ steht und in die Richtung des Lösungshalbraums für das ν-te Inputbeispiel zeigt.

⟶ Da jedes ν immer wieder auftritt, und da das Lösungsgebiet ein offener "Keil" ist, konvergiert das Verfahren immer in einer endlichen Anzahl Schritten gegen eine Lösung (mit Wahrscheinlichkeit 1).

Aber: Wenn keine Lösung für $\underline{W}$ existiert, d.h. wenn das Problem nicht linear separabel ist, konvergiert das Verfahren nicht!
(Auch nicht gegen eine optimale Lösung mit möglichst wenig Fehlklassifizierungen!)

⟶ "POCKET-ALGORITHMUS"

(Während Lernprozess immer die Gewichte der bis anhin besten Lösung abspeichern.)

ANHANG: VOLLSTÄNDIGER KONVERGENZBEWEIS FÜR DAS PERCEPTRON-LERNVERFAHREN

Wahl der Anfangsgewichte: $\underline{W} = \underline{W}^0 = \underline{0}$

(Der Beweis kann leicht auf beliebige Anfangsgewichte verallgemeinert werden!)

Gewichtsänderung beim Perceptron-Lernverfahren:

$$\Delta \underline{W} = \eta\,(D^\nu - O^\nu)\,\underline{I}^\nu = \eta\,(1 - D^\nu O^\nu)\,\underline{X}^\nu$$

wobei: $D^\nu = \pm 1$, $O^\nu = \text{sign}\,(\underline{W}\cdot\underline{I}^\nu) = \pm 1$

$$\underline{X}^\nu \equiv D^\nu \cdot \underline{I}^\nu \quad , \quad \nu = 1, \ldots, N$$

$$\rightarrow \Delta \underline{W} = \begin{cases} 2\eta\,\underline{X}^\nu & \text{falls } O^\nu \neq D^\nu, \text{ d.h. falls } \underline{W}\cdot\underline{X}^\nu < 0 \\ 0 & \text{- } O^\nu = D^\nu, \text{ - - } \underline{W}\cdot\underline{X}^\nu > 0 \end{cases}$$

$$\rightarrow \underline{W} = 2\eta \sum_{\nu=1}^{N} M^\nu \underline{X}^\nu \quad ,$$

wobei M^ν = Anzahl Präsentationen des Lernbeispiels ν, bei denen $O^\nu \neq D^\nu$ war.

Sei $\underline{W}^*$ eine Lösung $[\underline{W}^*\cdot\underline{X}^\nu > 0 \;,\; \nu = 1, \ldots, N]$:

$$\rightarrow \underline{W}\cdot\underline{W}^* = 2\eta \sum_{\nu=1}^{N} M^\nu \underline{X}^\nu\cdot\underline{W}^* \geq 2\eta\, M \cdot D(\underline{W}^*)\cdot|\underline{W}^*| \qquad (a)$$

wobei: $M \equiv \sum_{\nu=1}^{N} M^\nu$, $D(\underline{W}^*) = \frac{1}{|\underline{W}^*|} \min_\nu \underline{W}^*\cdot\underline{X}^\nu > 0$

Für eine Gewichtsänderung aufgrund des Lernbeispiels μ gilt andererseits:

$$\Delta|\underline{W}|^2 = (\underline{W} + 2\eta \underline{X}^\mu)^2 - \underline{W}^2 = 4\eta^2 (\underline{X}^\mu)^2 + 4\eta\, \underline{W}\cdot\underline{X}^\mu$$

$$\rightarrow \Delta|\underline{W}|^2 \leq 4\eta^2 (\underline{X}^\mu)^2 \quad , \quad \text{da } \underline{W}\cdot\underline{X}^\mu < 0$$

$$\rightarrow |W|^2 \leq \sum_{\mu=1}^{N} M^\mu \cdot 4\eta^2 (\underline{X}^\mu)^2 \leq 4\eta^2 M \cdot X_{max}^2 \qquad (b)$$

$$\text{wobei: } X_{max}^2 = \max_{\mu} (\underline{X}^\mu)^2$$

Aber:

$$\underline{W}\cdot\underline{W}^* = |\underline{W}|\cdot|\underline{W}^*|\cdot\cos\phi$$

wobei ϕ = Winkel zwischen $\underline{W}$ und $\underline{W}^*$

$$\rightarrow 1 \geq \cos\phi = \frac{\underline{W}\cdot\underline{W}^*}{|\underline{W}|\cdot|\underline{W}^*|} \geq \frac{2\eta M \cdot D(\underline{W}^*)\cdot|\underline{W}^*|}{2\eta\sqrt{M}\sqrt{X_{max}^2}\cdot|\underline{W}^*|}$$

siehe (a) und (b)

$$\rightarrow 1 \geq \frac{\sqrt{M}\cdot D(\underline{W}^*)}{\sqrt{X_{max}^2}}$$

$$\rightarrow M \leq \frac{X_{max}^2}{D(\underline{W}^*)^2} \quad , \quad \text{d.h. die Anzahl M der Gewichtsänderungen während des Lernprozesses ist endlich!}$$

PERCEPTRONS (ERGÄNZUNGEN):

Perceptron: $O^\nu = \text{sign}\left(\sum_{i=0}^{d} W_i I_i^\nu\right)$

$$\longrightarrow \quad O^\nu = D^\nu \,, \quad \nu = 1,\dots,N \qquad [O^\nu, D^\nu = \pm 1]$$

$$\Updownarrow$$

$$E^\nu = D^\nu \sum_{i=0}^{d} W_i I_i^\nu > 0 \quad , \quad \nu = 1,\dots,N$$

Perceptron-Lernverfahren:

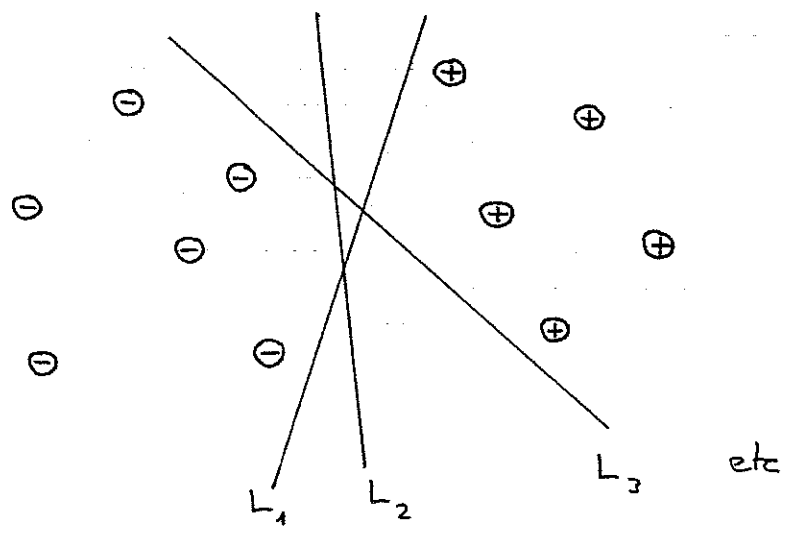
- ν zufällig oder der Reihe nach ausgewählt
- $\Delta W_i = \eta\,(D^\nu - O^\nu)\, I_i^\nu$

$$\longrightarrow \quad \Delta E^\nu = D^\nu \sum_{i=0}^{d} \Delta W_i I_i^\nu = \eta\,(1 - D^\nu O^\nu) \sum_{i=0}^{d} (I_i^\nu)^2$$

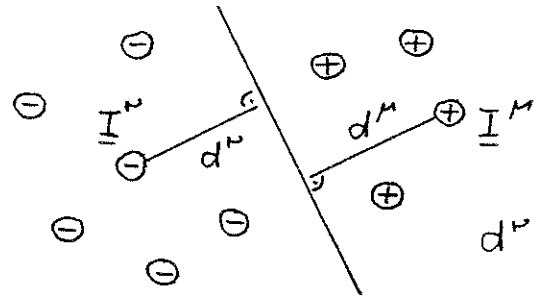
$$\longrightarrow \quad \Delta E^\nu > 0 \quad \text{falls} \quad O^\nu \neq D^\nu \qquad [= 0 \ \text{falls} \ O^\nu = D^\nu]$$

d.h. E^ν wird immer wieder vergrössert, bis schliesslich alle $E^\nu > 0$ *[falls Lösung existiert!]*

Es gibt aber viele verschiedene Lösungen:



Perceptron mit maximaler Stabilität



$$d^\mu = \frac{E^\mu}{|\underline{W}|} = \frac{D^\mu \sum_{i=0}^{d} W_i I_i^\mu}{\sqrt{\sum_{i=1}^{d} W_i^2}}$$

→ *Maximale Stabilität:* $\boxed{\min_\mu \{d^\mu\} = \max}$

Lösungsmethoden:

(i) Minover-Lernalgorithmus (für $I_i^\mu = \pm 1$)

[Krauth & Mézard, J. Phys. A 20, L745-752 (1987)]

(ii) Formulierung als Optimierungsproblem

- $\sqrt{\sum_{i=1}^{d} W_i^2} = \min$
- Nebenbedingungen: $D^\mu \sum_{i=0}^{d} W_i I_i^\mu \geq 1$, $\mu = 1, \ldots, N$

→ "Support Vector Classifiers"

[siehe z.B.: T. Hastie et al
The Elements of Statistical Learning
Springer Series in Statistics, 2001
Chapters 4 and 12]

→ Verallgemeinerung auf nicht-lineare Trennflächen, nicht perfekt separierbare Klassifizierungsprobleme, und auf Regressionsprobleme:

"Support Vector Machines"

V.4 Erweiterung des Inputraums

- Ursprünglicher Inputraum: I_i , $i = 1, \ldots, d$

- Erweiterter Inputraum:

  z.B.: I_i , $I_i I_j$, $I_i I_j I_k$, ...

  oder: I_i , $\sin(\pi I_i)$, $\cos(\pi I_i)$, $\sin(2\pi I_i)$, ...

→ Perceptrons mit erweitertem Inputraum können nichtlineare Trennflächen realisieren!

→ Probleme, die im ursprünglichen Inputraum nicht linear separabel sind, können im erweiterten Inputraum linear separabel werden!

→ Einfaches Perceptron-Lernverfahren anwendbar! (d.h. schnelleres Lernen als bei Verwendung von komplexeren Netzwerk-Architekturen)

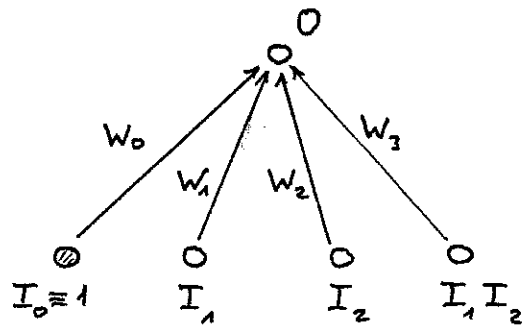
Aber: Geeignete Form der nichtlinearen Erweiterung nicht immer offensichtlich!

Beispiel: XOR-Problem

I_1	I_2	O
0	0	-1
1	0	+1
0	1	+1
1	1	-1

→ nicht linear separabel!

Mit erweitertem Inputraum aber linear separabel:



z.B. $W_0 = -1$

$W_1 = 2 \quad W_2 = 2$

$W_3 = -4$

$$\left[O = \operatorname{sign}\left(\sum_{i=0}^{3} W_i I_i \right) \right]$$

V.5 Minimierung des mittleren quadratischen Fehlers

$$F = \frac{1}{N} \sum_{\mu=1}^{N} (D^\mu - O^\mu)^2 = \min \quad , \quad O^\mu = f\left(\sum_i W_i I_i^\mu\right)$$

- EINFACHES GRADIENTENVERFAHREN:

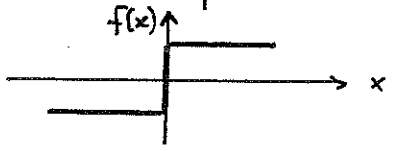
  (zur Minimierung von F)

$$W_i \longrightarrow W_i - \eta \frac{\partial F}{\partial W_i}$$

$$\frac{\partial F}{\partial W_i} = -\frac{1}{N} \sum_{\mu=1}^{N} 2 (D^\mu - O^\mu) \cdot f'\left(\sum_j W_j I_j^\mu\right) \cdot I_i^\mu$$

Im Gegensatz zum Perceptron-Lernverfahren konvergiert dieses Verfahren immer (gegen ein lokales Minimum von F), falls η nicht zu gross gewählt wird.

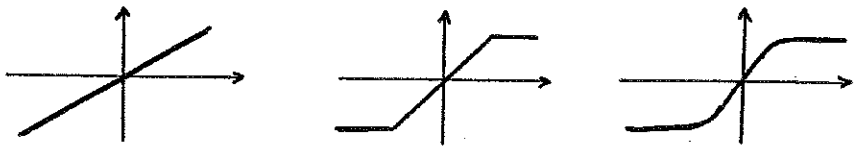
Beim ursprünglichen Perceptron ist aber $f = \text{sign}$:



→ f' existiert nicht, d.h. das Gradientenverfahren ist nicht anwendbar!

→ Änderung der Wahl der Aktivierungsfunktion f für das Output-Neuron:

z.B.



(oder Wahl eines anderen Optimierungsverfahrens zur Minimierung von F
[z.B. stochastisches Optimierungsverfahren]
(siehe Kapitel VIII))

V.6 Analyse von Gradientenverfahren

A) Lineares Output-Neuron [ADALINE]

$\rightarrow O^\mu = \sum_i W_i I_i^\mu$

$\rightarrow F = \frac{1}{N} \sum_{\mu=1}^{N} [D^\mu - \sum_i W_i I_i^\mu]^2 = \frac{1}{N} \sum_{\mu=1}^{N} F_\mu$

ist eine quadratische Funktion der W_i

$\rightarrow$ Es existiert genau 1 Minimum!

Eindimensionales Beispiel:

$$F = A - 2BW + CW^2$$

$$\left[\rightarrow F_\mu = a_\mu - 2b_\mu W + c_\mu W^2 \qquad A = \frac{1}{N} \sum_{\mu=1}^{N} a_\mu = \langle a_\mu \rangle \text{ , etc.} \right]$$

$$\rightarrow F_{min} = F(W^*) = A - \frac{B^2}{C} \quad , \quad W^* = \frac{B}{C}$$

- **Gradientenmethode ("Batch"):**

$\rightarrow$ Änderung von W ist proportional zum negativen Gradienten von F
(F = Summe der quadratischen Outputfehler über alle Lernbeispiele)

$$W_{k+1} = W_k - \eta \left. \frac{\partial F}{\partial W} \right|_{W=W_k} = W_k + 2\eta (B - CW_k)$$

(k = Iterationsindex)

Setze $V_k = W_k - W^* \quad (W^* = \frac{B}{C})$:

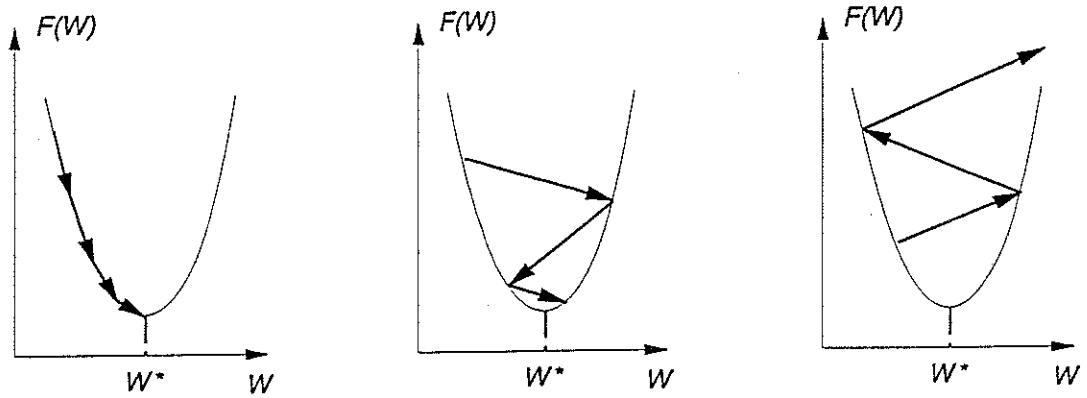
$$\rightarrow V_{k+1} = (1 - 2\eta C) V_k$$

$\longrightarrow$ Konvergenz falls $\lim_{k\to\infty} (1-2\eta C)^k = 0$

d.h. falls $|1-2\eta C| < 1$

$\longrightarrow$ $\boxed{0 < \eta < \frac{1}{2C}}$ oder $\boxed{\frac{1}{2C} \leq \eta < \frac{1}{C}}$

Im ersten Fall strebt W_k stetig gegen die Lösung W^*, im 2. Fall wird W^* in einer "Zickzackbewegung" angenähert ($V_k = W_k - W^*$ wechselt bei jedem Iterationsschritt das Vorzeichen). Für $\eta > \frac{1}{C}$ divergiert das Verfahren.



$0 < \eta < 1/2C$ $\quad$ $1/2C < \eta < 1/C$ $\quad$ $\eta > 1/C$

- <u>LMS-Algorithmus ("Example-by-Example"):</u>

  [Widrow & Hoff 1960] (LMS = "least mean squares")

  $\longrightarrow$ *Lernbeispiele ν werden in zufälliger Reihenfolge immer wieder präsentiert und W jedesmal um einen Betrag geändert, der proportional zum negativen Gradienten von F_ν ist:*

$$W_{k+1} = W_k - \eta \frac{\partial F_\nu}{\partial W} = W_k + 2\eta (b_\nu - c_\nu W_k)$$

(ν zufällig)

$\longrightarrow$ Wenn η genügend klein ist, dann strebt $\{W_k\}$ gegen einen stationären Zufallsprozess mit:

$$\langle F \rangle = \langle a \rangle - 2\langle b \rangle \langle W \rangle + \langle c \rangle \langle W^2 \rangle$$

$$\langle W \rangle = \frac{\langle b \rangle}{\langle c \rangle} = W^*$$

$$\langle W^2 \rangle = \frac{\langle b \rangle^2 + \eta \left[\langle b^2 \rangle \langle c \rangle - 2 \langle b \rangle \langle bc \rangle \right]}{\langle c \rangle \left[\langle c \rangle - \eta \langle c^2 \rangle \right]}$$

$$\longrightarrow \langle F \rangle - F_{min} \propto \eta \quad \text{falls} \quad \eta \ll \frac{\langle c \rangle}{\langle c^2 \rangle}$$

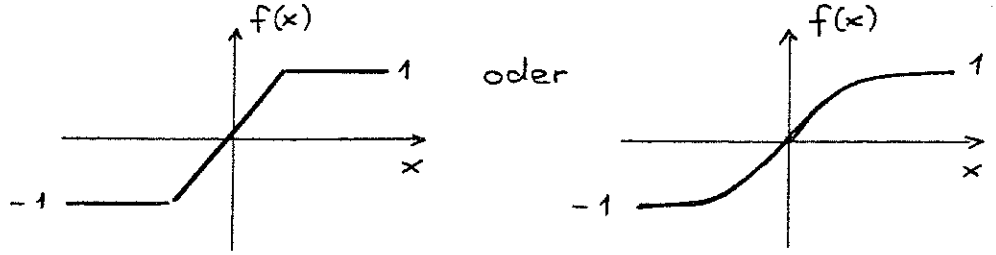
Bemerkung:

Die Konvergenzbeweise für die Gradientenmethode und für den LMS-Algorithmus lassen sich analog führen, wenn F von mehreren W_i abhängt.
[Vektor/Matrix-Notation!]

B) Nichtlineares Output-Neuron

$$F = \frac{1}{2N} \sum_{\nu=1}^{N} (D^\nu - O^\nu)^2 = \min, \quad O^\nu = f\left(\sum_i W_i I_i^\nu\right)$$

wobei z.B.:



$$\longrightarrow \quad \Delta W_i = -\eta \frac{\partial F}{\partial W_i} = \eta \frac{1}{N} \sum_\mu (D^\mu - O^\mu) \cdot f'\left(\sum_j W_j I_j^\mu\right) \cdot I_i^\mu$$

bzw. $$\Delta W_i = -\eta \frac{\partial F_\mu}{\partial W_i} = \eta (D^\mu - O^\mu) \cdot f'\left(\sum_j W_j I_j^\mu\right) \cdot I_i^\mu$$

- Auch im nichtlinearen Fall konvergieren die Verfahren, falls η genügend klein gewählt wird.
- Aber wenn f nichtlinear ist, können lokale Minima auftreten!

  → *Nur Konvergenz gegen nächstliegendes lokales Minimum garantiert!*

Bemerkungen:

i) Wenn die "gewünschten" Outputs $D^\mu = \pm 1$ sind, und die nichtlineare Aktivierungsfunktion f zwischen -1 und $+1$ variiert, dann

$$O^\mu = f\left(\sum_i W_i I_i^\mu\right) = \pm 1$$

nur möglich, wenn gewisse $|W_i| \longrightarrow \infty$.

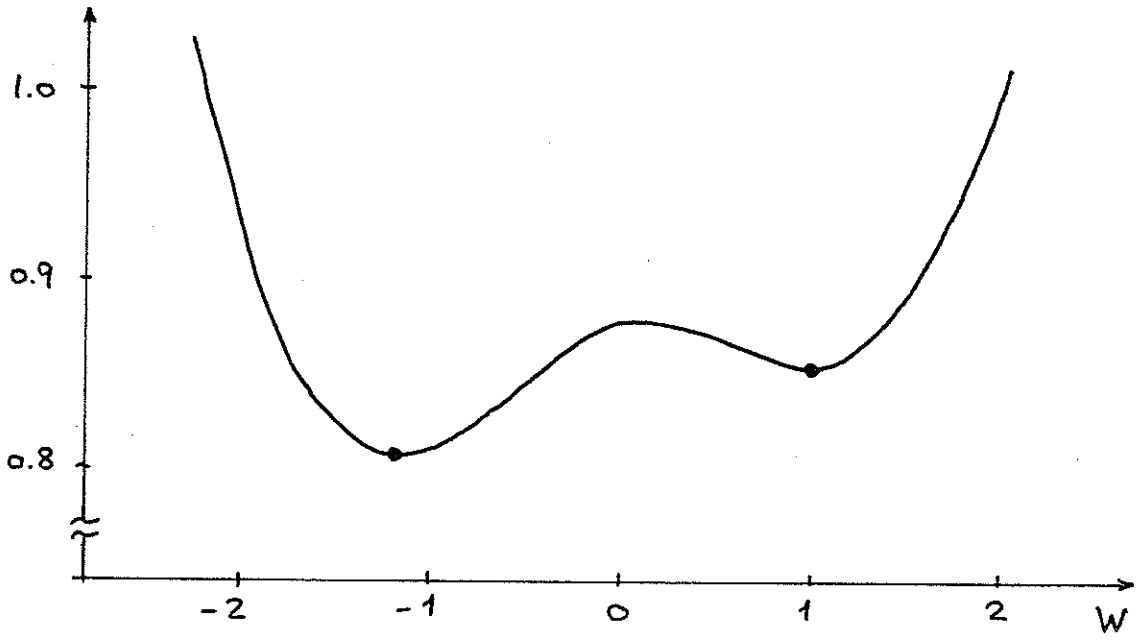
Abhilfe:
- D^μ von ± 1 auf ± 0.9 reduzieren
- f in y-Richtung strecken, sodass ihre Maximalwerte z.B. ± 1.1 sind
- Cutoff: $$F_\mu = \begin{cases} \frac{1}{2}(D^\mu - O^\mu)^2 & \text{if } |D^\mu - O^\mu| > \varepsilon \\ 0 & \text{if } |D^\mu - O^\mu| \leq \varepsilon \end{cases}$$

ii) Das Perceptron-Lernverfahren entspricht dem LMS-Algorithmus mit $f' = 1$ (obwohl $f = \text{sign}$!)
[→ Keine Konvergenz wenn $F_{min} > 0$!]

Beispiel mit 2 lokalen Minima:

$$F = F_1 + F_2 = f(W-1.52)^2 + [f(W+1.52)+0.04]^2 = \min$$

$$f(x) = \frac{1-e^{-x}}{1+e^{-x}}$$



a) "Batch": → Immer Konvergenz gegen dasjenige Minimum, das dem Anfangswert für W am nächsten liegt. (wenn η genügend klein)

b) "Example-by-Example":

→ Mit grosser Wahrscheinlichkeit Konvergenz gegen das tiefere Minimum (selbst wenn bei W=1 gestartet wird!).

→ *Example-by-Example Verfahren hat Vorteile bezüglich Steckenbleiben in schlechten lokalen Minima!*