Theorie neuronaler Netze

Friedhelm Schwenker

October 16, 2012

Organisation

- Vorlesung (3h) mit Übung (1h)
  Mi 12-14 H21 und Fr 10-12 Uhr 123
  MatLab: 19.10.2012
  1. Übung: 9.11.2012
  Schein: 50% der Punkte (7 Übungsblätter) + aktive Übungsteilnahme; Bonusregel gilt!

Theorie neuronaler Netze

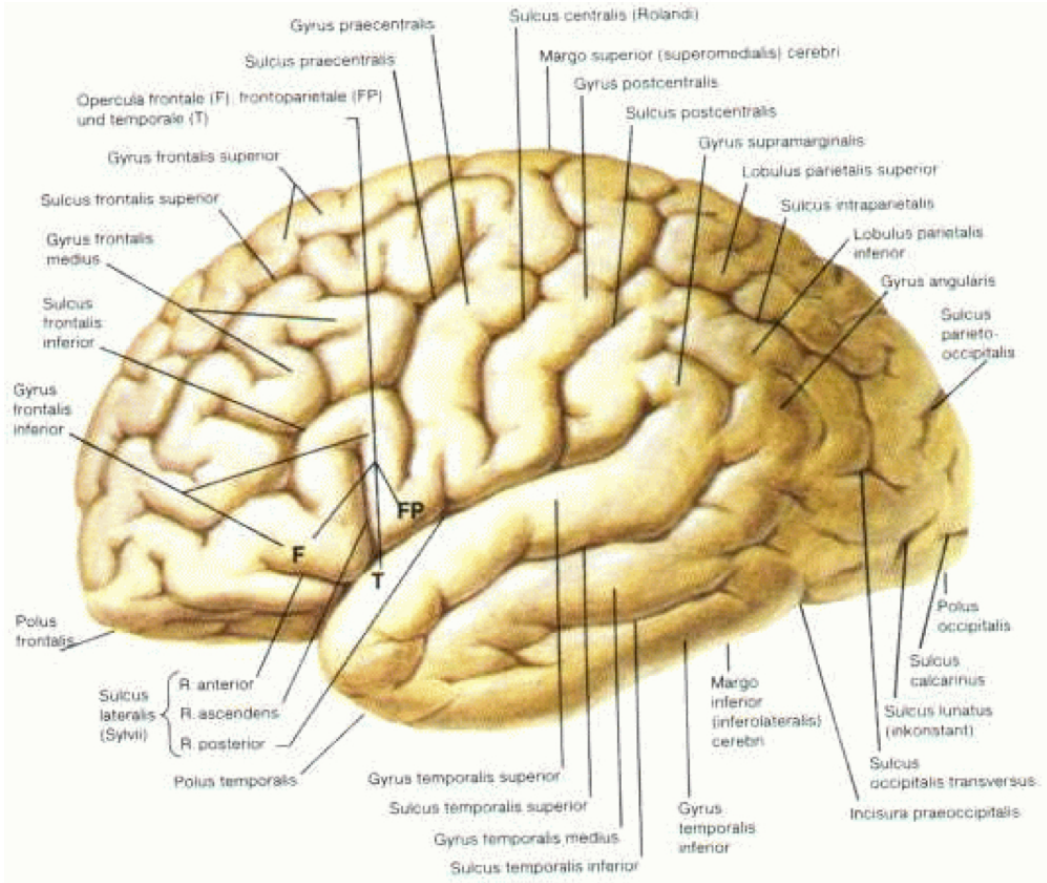
1. Modellierung neuronaler Netze
2. Stabilität neuronaler Netze
3. Lernen in Einschichtnetzen
4. Lernen in Mehrschichtnetzen
5. Rekurrente Netze
6. Darstellung mit neuronalen Netzen
7. Komplexität neuronaler Netze

1. Modellierung neuronaler Netze

1. Biologische Grundlagen
2. Allgemeines Kanalmodell
3. Vereinfachte Neuronenmodelle
4. Quasi-stationäre Lösungen

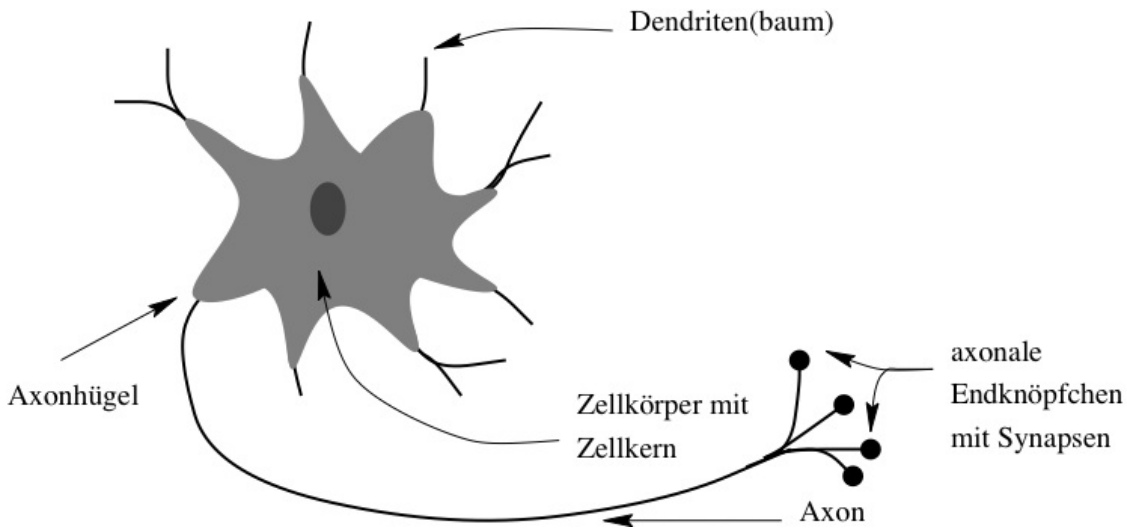
1.1 Biologische Grundlagen

Der Kortex des menschlichen Gehirns mit verschiedenen Arealen.



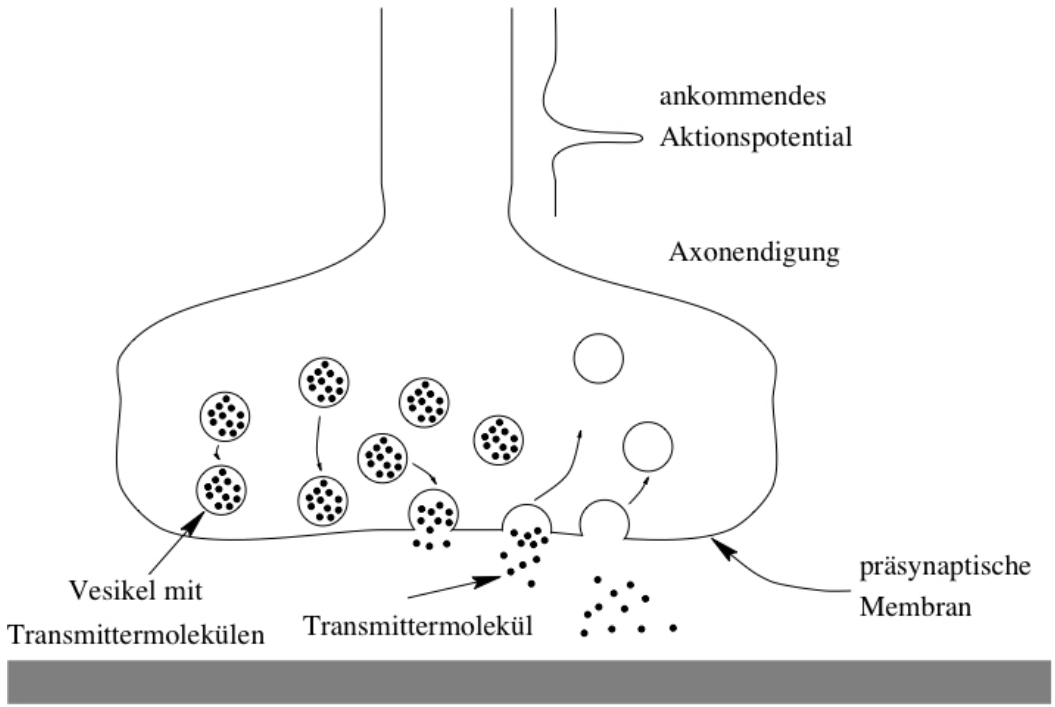
Bestandteile eines typischen Neurons

- Dendriten bzw. Dendritenbaum
- Zellkörper (Soma)
- Nervenfaser (Axon) mit dem axonalen Baum



Aufbau einer Synapse

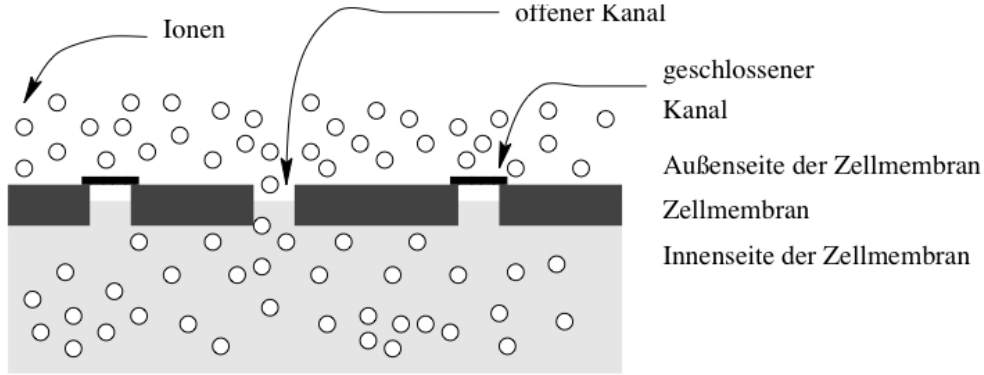
- Synapse mit der prä- und postsynaptischen Membran.
- Ausschüttung der Neurotransmittermoleküle in den synaptischen Spalt.



postsynaptische Membran mit Rezeptoren für die Transmittermoleküle

Zellmembran / Ionen

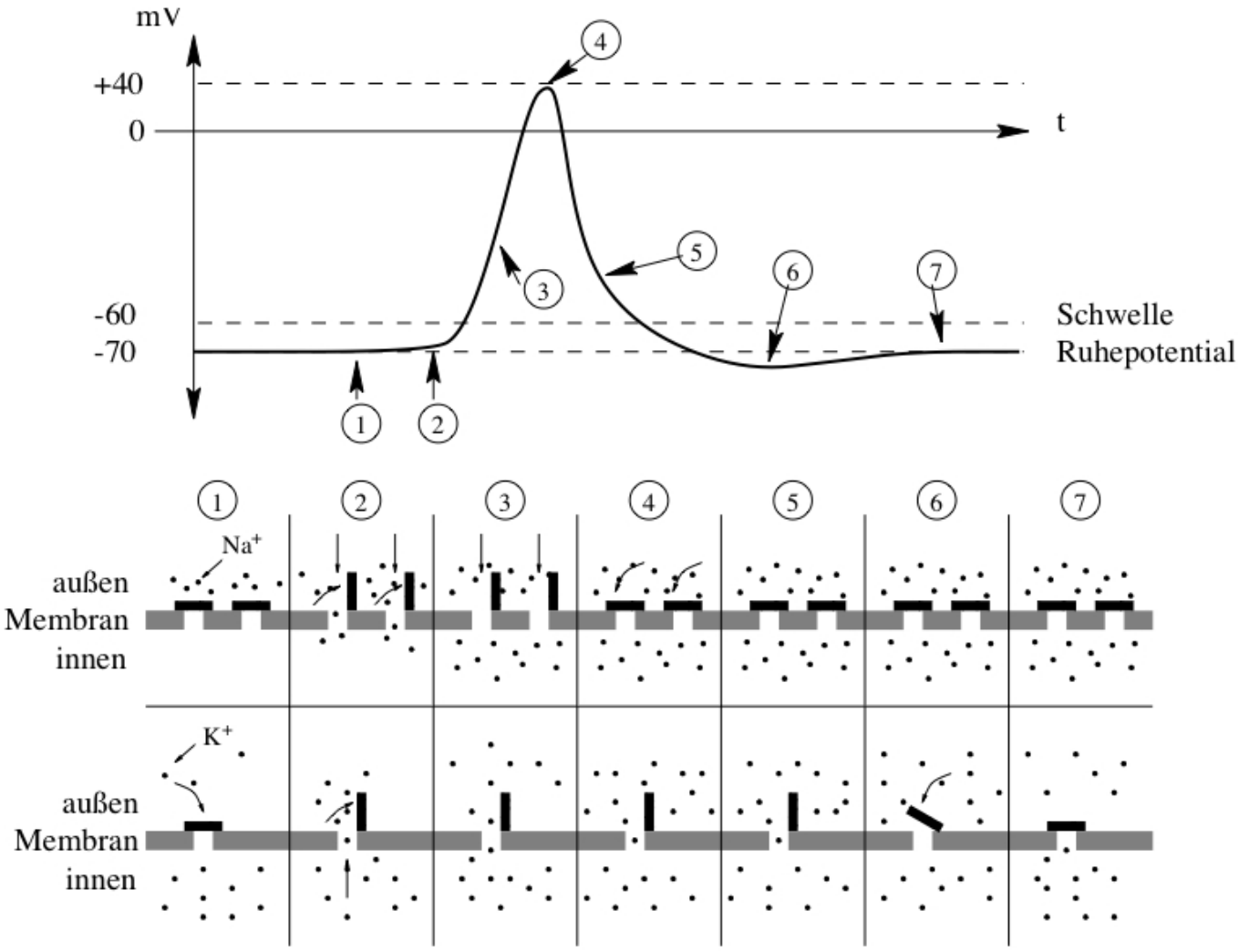
Zellmembran - offene und verschließbare Ionenkanäle sind abgebildet.



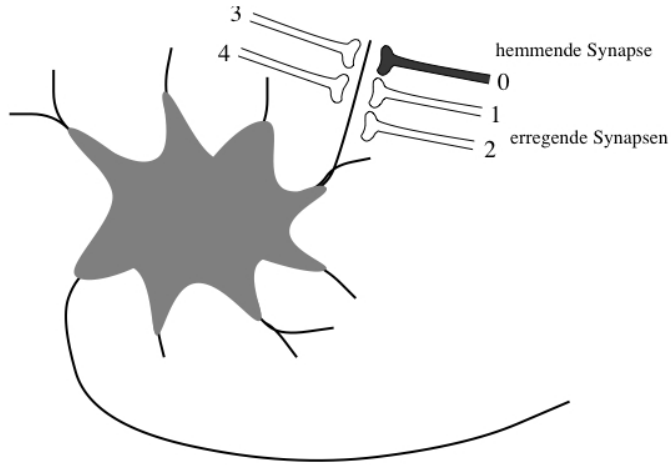
Typische Ionenkonzentrationen (für (Tintenfisch-)Axon in Millimol/Liter) im Intra- und Extrazellulärraum

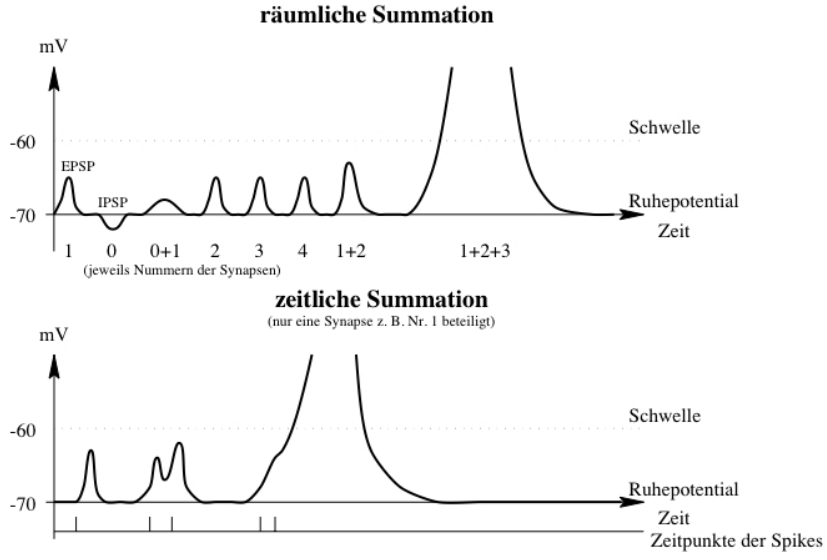
Typ	**Innen**	**Außen**
K^+	400	20
Cl^-	30	590
Na^+	60	436

Aktionspotential



Räumlich-zeitliche Summation am Dendriten





Dynamik der neuronalen Verarbeitung

- **Summation am Zellkörper** EPSP/IPSP von vorgeschalteten Neuronen führen zu Potentialveränderungen am Dendritenbaum/Zellkörper (räumlich/zeitliche Summation der EPSP/IPSP)
- **Auslösung eines Aktionspotentials** Ändert sich das Membranpotential am Zellkörper nur schwach, wird kein Aktionspotential ausgelöst (unterschwelligen Erregung). Bei hinreichend starker Depolarisation der Membran wird im Bereich des Axonhügels ein Aktionspotential ausgelöst.
- **Ausbreitung des Aktionspotentials** Das Aktionspotential breitet über das Axon bis in den axonalen Baum mit konstanter Amplitude aus
- **Synaptische Übertragung** Das Aktionspotential führt in den axonalen Endknöpfchen zu einer Ausschüttung von Neurotransmittern, die auf der postsynaptischen Membran zu einer Änderung des Membranpotentials führen.

Anatomische Größen

	Maus	**Mensch**
N/mm^3	10^5	$3 \cdot 10^4$
D/N	3 mm	2 cm
A/N	2 cm	6 cm
S/N	10^4	$2 \cdot 10^4$
D/mm^3	$3 \cdot 10^2$ m	$6 \cdot 10^2$ m
A/mm^3	$2 \cdot 10^3$ m	$2 \cdot 10^3$ m
S/mm^3	10^9	$6 \cdot 10^8$
S/D	3 / μm	1 / μm
S/A	1 / 2 μm	1 / 3 μm

N = Neuronenzahl ; D = Dendritenlänge; S = Synapsenzahl; A = Axonlänge

Computer vs. Neuronale Netze

Computer	**Neuronale Netze**
kaum fehlertolerant	fehlertolerant, robust gegenüber verrauschtenDaten
Totalausfall bei Hardwarestörungen	System funktioniert mit eingeschränkter Funktionalität
explizite Programmierung	Lernen durch Beispiele
wenige, aber komplexe Prozessoren (typischerweise 10^0)	sehr viele einfache Neuronen (menschliches Gehirn ca. 10^{11})
niedriger Vernetzungsgrad (Verbindungen $\sim 10^2$ am Prozessor)	hoher Vernetzungsgrad (bis zu 10^4 Synapsen pro Neuron)
kurze Schaltzeiten der Prozessoren (heute im 10^8 Hz-Bereich)	lange Schalt– und Laufzeiten (im 10^2 Hz-Bereich)

1.2 Allgemeines Kanalmodell

Modellvoraussetzungen für das allgemeine Kanalmodell:

- $I = \{1, \ldots, N\}$ die Menge der Neuronen.
- Jedes Neuron j ist von einem bestimmten (Membran-)Typ l und einem bestimmten (Übertragungs-)Kanaltyp k.
- $\tau_l > 0$ Zeitkonstante und $\vartheta_l : \mathbb{R}_+ \to \mathbb{R}_+$ die Schwellenfunktion für den Membrantyp l.
- $r_k : \mathbb{R}_+ \to \mathbb{R}_+$ die Responsefunktion und U_k Umkehrpotentiale für den Übertragungskanaltyp k.
- $C = (c_{ij})$ synaptische Kopplungsmatrix und $D = (d_{ij})$ Matrix der Delays.

Neuronenmodelle (1/2)

(1) Dendritisches Potential u_j des Neurons j vom Membrantyp l

$$\tau_l \dot{u}_j(t) = -u_j(t) + \sum_k a_j^k(t)(U_k - u_j(t)) + x_j(t)$$

(V) Variante: u_j nach Spike von j auf Ruhepotential U_0 gesetzt.

(2) Gesamtinputaktivität a_j^k im Übertragungskanal k des Neurons j

$$a_j^k(t) = \sum_{i \in I_k} a_{ij}(t) \quad I_k \text{ Neuronen vom Kanaltyp } k$$

übertragene Aktivität a_{ij} vom Neuron i zum Neuron j im Kanal k

$$a_{ij}(t) = \sum_{s \in T_t^i} r_k(t - (s + d_{ij}))c_{ij} = \sum_{s<t} r_k(t - (s + d_{ij}))y_i(s)c_{ij}$$

Neuronenmodelle (2/2)

(3) Outputaktivität y_j des Neurons j

$$y_j(t) = \mathbf{1}_{[u_j(t) \geq \theta_j(t)]}$$

(4) Schwellenfunktion θ_j und Spikezeitpunkte T_t^j von Neuron j (vom Membrantyp l)

$$\theta_j(t) = \max_{s \in T_t^j} \vartheta_l(t-s) \qquad T_t^j = \{s < t \mid y_j(s) = 1\}$$

(5) Diskretisierung von (1) mit $\varrho_l := \frac{\Delta t}{\tau_l}$ und $\varrho_l \in (0, 1]$

$$u_j(t + \Delta t) = (1 - \varrho_l) \cdot u_j(t) + \varrho_l \sum_k a_j^k (U_k - u_j(t)) + \varrho_l x_j(t)$$

Vereinfachungen

a) Response Funktion durch Gesamtwirkung w_{ij} beschreiben:

$$(2') \qquad a_j^k(t) = \sum_{i \in I_k} w_{ij} y_i(t - d_{ij})$$

b) $U_k - u$ ersetzen durch U_k (unterhalb der Schwelle ist u klein):

$$(1') \qquad \tau_l \dot{u}_j(t) = -u_j(t) + \sum_k a_j^k(t) U_k + x_j(t)$$

Mit $(2')$ folgt:

$$(1'') \qquad \tau_l \dot{u}_j(t) = -u_j(t) + \sum_i w_{ij} y_i(t - d_{ij}) + x_j(t)$$

c) Einfache zweiwertige Schwellenfunktion $\vartheta(t) \in \{\vartheta_{ref}, \vartheta_{rest}\}$.

d) Schwellenmechanismus (mit ϑ_l) durch Ratenfunktion f_l ersetzen:

$$(3') \qquad y_j(t) = f_l(u_j(t))$$

e) Alle Delays $= 1$; alle Delays $= 0$.

1.3 Vereinfachte Neuronenmodelle

Kanalmodell: Gleichungen $1, 2, 3$ und 4

Spike-Response-Modell: $1', 2, 3$ und 4

Dynamic-Threshold-Modell: $1, 2', 3$ und 4

Integrate-and-Fire-Modell: $1''$ mit Variante (V), $3, 4$ und mit einfacher Schwelle (d)

Ratenmodell: $1, 2$ und $3'$

Grundmodell: $1''$ und $3'$

Lineares Modell: $1'$, $3'$ und mit linearer Ratenfunktion f

Einfaches lineares Modell: $1'', 3'$, Delays alle $= 0$ und mit linearer Ratenfunktion f

1.4 Asynchrone quasistationäre Zustände (1/5)

Asynchrone quasistationäre Zustände - als Basis für die neuronale Informationsverarbeitung.

(1) **Ratenfunktion** $f_l(u)$ aus Schwellenfunktion $\vartheta_l(t)$ bestimmen.

$$y_j = f_l(u_j) = \frac{1}{\vartheta_l^{-1}(u_j)} \quad \text{fr } j \text{ vom Membrantyp } l$$

(2) **Gesamtwirkung** $\hat{r}_k := \int r_k(t)dt$ aus der Responsefunktion $r_k(t)$.
Diese beiden Vereinfachungen ergeben das *Grundmodell* mit:

$$a_j^k = \sum_{i \in I_k} \hat{r}_k \; y_i \; c_{ij} = \sum_{i \in I_k} \hat{r}_k \; f_{l_i}(u_i) \; c_{ij}$$

l_i Membrantyp des Neurons i.

... (2/5)

(3) **Berechnung der Fixpunkte** mit dem Ansatz $0 \approx \dot{u}_j$

$$0 \approx \tau_k \dot{u}_j = -u_j + \sum_k a_j^k \, (U_k - u_j) + x_j \quad \Longleftrightarrow \quad u_j \approx \frac{x_j + \sum_k a_j^k U_k}{1 + \sum_k a_j^k}$$

(4) **Linearisierung** ergibt ein homogenes einfaches lineares Modell
DGL des dendritischen Potentials lautet mit den Vereinfachungen:

$$\tau_l \dot{u}_j = -u_j + \sum_k (U_k - u_j) \hat{r}_k \sum_{i \in I_k} f_{l_i}(u_i) c_{ij} + x_j$$

Betrachten nun DGL für $u_j = u_j + e_j$:

$$\tau_l (\dot{u}_j + \dot{e}_j) = -(u_j + e_j) + \sum_k (U_k - (u_j + e_j)) \hat{r}_k \sum_{i \in I_k} f_{l_i}(u_i + e_i) c_{ij} + x_j$$

... (3/5)

Nun setzen wir ein $f_{l_i}(u_i + e_i) \approx f_{l_i}(u_i) + e_i f'_{l_i}(u_i)$

$$\tau_l(\dot{u}_j + \dot{e}_j) = -(u_j + e_j) + x_j + \sum_k (U_k - (u_j + e_j))\hat{r}_k \sum_{i \in I_k} (f_{l_i}(u_i) + e_i f'_{l_i}(u_i))c_{ij}$$

$$\begin{aligned}
\tau_l(\dot{u}_j + \dot{e}_j) \quad = \quad & -(u_j + e_j) + x_j + \sum_k (U_k - u_j)\hat{r}_k \sum_{i \in I_k} f_{l_i}(u_i)c_{ij} \\
& + \sum_k (U_k - u_j)\hat{r}_k \sum_{i \in I_k} e_i f'_{l_i}(u_i)c_{ij} \\
& - \sum_k e_j \hat{r}_k \sum_{i \in I_k} f_{l_i}(u_i)c_{ij} - \sum_k e_j \hat{r}_k \sum_{i \in I_k} e_i f'_{l_i}(u_i)c_{ij}
\end{aligned}$$

...(4/5)

Die Summanden $\sum_k \hat{r}_k \sum_{i \in I_k} e_j e_i f'_{l_i}(u_i) c_{ij}$ sind quadratisch in den Störungstermen e_j und können bei der Linearisierung vernachlässigt werden:

$$\tau_l \dot{e}_j = -e_j + \sum_k (U_k - u_j) \hat{r}_k \sum_{i \in I_k} e_i f'_{l_i}(u_i) c_{ij} - e_j \sum_k \hat{r}_k \sum_{i \in I_k} f_{l_i}(u_i) c_{ij}$$

Mit $\bar{C}_{ij} = \hat{r}_k f_{l_i}(u_i) c_{ij}$ und $\hat{C}_{ij} = (U_k - u_j) \hat{r}_k f'_{l_i}(u_i) c_{ij}$ folgt die lineare DGL:

$$\tau_l \dot{e}_j = -e_j - e_j \sum_k \sum_{i \in I_k} \bar{C}_{ij} + \sum_k \sum_{i \in I_k} \hat{C}_{ij} e_i$$

...(5/5)

Man erhält man ein homogenes lineares DGl-System:

$$\dot{e} = Ae$$

mit $e = (e_1, \ldots e_n)^T$, $\dot{e} = (\dot{e}_1, \ldots \dot{e}_n)^T$ und mit der reellen $n \times n$ Matrix $A = (a_{ij})$ definiert durch

$$a_{jj} = -\frac{1}{\tau_I}(1 + \sum_k \sum_{i \in I_k} \bar{C}_{ij}) \qquad \text{und} \qquad a_{ji} = \frac{1}{\tau_I} \sum_k \sum_{i \in I_k} \hat{C}_{ij}, \qquad i \neq j$$

Die Stabilität dieses Systems ist zu untersuchen, dazu betrachten wir einfache lineare neuronale Netze.

2. Stabilität in neuronalen Netzen

1. Lineare Netze in kontinuierlicher Zeit
2. Lineare Netze in diskreter Zeit
3. Äquivalenz der Stabilitätsbedingungen
4. Hopfield Netze in kontinuierlicher Zeit
5. Hopfield Netze in diskreter Zeit

2.1 Stabilität linearer Netze (kontinuierliche Zeit)

Wir untersuchen das System von n linearen Modellneuronen:

$$\tau \dot{u}_j = -u_j + x_j + \sum_{i=1}^{n} c_{ij} u_i \qquad j = 1, \ldots, n$$

$\tau > 0$ Zeitkonstante, u_j dendritisches Potential, x_j externe Eingabe, c_{ij} synaptische Kopplungen.
Für Modelle ohne externen Input, also $x_j = 0$, erhalten wir ein homogenes DGL-System

$$\tau \dot{u} = -u + uC = u(C - Id)$$

mit Einheitsmatrix Id.

...

Also

$$\dot{u} = Au \qquad \text{mit} \qquad A = \frac{1}{\tau}(C^T - \mathrm{Id}).$$

Lösungsansatz für (komplexe) Lösungen:

$$u(t) = ve^{\lambda t} \qquad \text{mit} \qquad \lambda \in \mathbb{C} \text{ und } v \in \mathbb{C}^n.$$

Hierfür muss dann gelten

$$Au = Ave^{\lambda t} = \dot{u} = v\lambda e^{\lambda t}$$

also muss gelten

$$Av = \lambda v$$

d.h. $\lambda \in \mathbb{C}$ ist *Eigenwert* mit zugehürigem *Eigenvektor* $v \in \mathbb{C}^n$ der (reellen) $n \times n$ Matrix A

...

Für eine $n \times n$ Matrix A sind die Eigenwerte nach Definition genau die Nullstellen des *charakteristischen Polynoms* von A. Es ist definiert durch:

$$p_A(\lambda) = \det(A - \lambda Id)$$

Für einen Eigenwert $\lambda \in \mathbb{C}$ von A bestimmt man zugehörige Eigenvektoren $v \in \mathbb{C}^n$ durch Lösen des LGS:

$$(A - \lambda Id)v = 0$$

Es gelten die folgenden Eigenschaften:

- Sei u Lösung von $\dot{u} = Au$, dann ist auch $\bar{u}$ Lösung, denn:

$$\dot{\bar{u}} = \overline{\dot{u}} = \overline{Au} = A\bar{u}$$

  denn A ist eine reelle Matrix.

- Seien u_1, u_2 Lösungen von $\dot{u} = Au$ und $a_1, a_2 \in \mathbb{C}$ dann sind auch $a_1u_1 + a_2u_2$ Lösungen:

$$\dot{(a_1u_1 + a_2u_2)} = a_1\dot{u}_1 + a_2\dot{u}_2 = a_1Au_1 + a_2Au_2 = A(a_1u_1 + a_2u_2)$$

...

- Sei u Lösung von $\dot{u} = Au$, dann sind auch $Re(u)$ und $Im(u)$ Lösungen:

$$Re(u) = \frac{1}{2}(u + \bar{u}) \qquad \text{und} \qquad Im(u) = \frac{1}{2i}(u - \bar{u})$$

- Explizite Darstellung von Real- und Imaginärteil sind:

$$Re(u) = e^{\mu t}(a\cos(\nu t) - b\sin(\nu t))$$

und

$$Im(u) = e^{\mu t}(a\sin(\nu t) + b\cos(\nu t))$$

hierbei sei $\lambda = \mu + i\nu$ und $v = a + ib$ mit $\mu, \nu \in \mathbb{R}$ und $a, b \in \mathbb{R}^n$.

....

Stabilität von $\dot{u} = Au$ bei $t \to \infty$ offenbar gdw alle Eigenwerte $\lambda \in \mathbb{C}$ von $A = \frac{1}{\tau}(C^T - \mathrm{Id})$ negative Realteile haben bzw. die Eigenwerte der Kopplungsmatrix c Realteile kleiner als 1 haben.
Es gelten folgende Aussagen für Eigenwerte/Eigenvektoren von Matrizen:

1. λ Eigenwert zum Eigenvektor v von Matrix C, gdw λ Eigenwert zum Eigenvektor v von Matrix C^T
2. Es sei $\alpha, \beta \in \mathbb{C}$. Dann gilt λ Eigenwert zum Eigenvektor v von Matrix C, gdw $\alpha\lambda + \beta)$ Eigenwert zum Eigenvektor v von Matrix $\alpha C + \beta\mathrm{Id}$.

Es sei λ Eigenwert von C dann gilt:

$$\begin{aligned} 0 &= det(\alpha\lambda + \beta\mathrm{Id} - \alpha C + \beta\mathrm{Id}) && (1)\\ &= det(\alpha(\lambda\mathrm{Id} + \mu\mathrm{Id} - C - \mu\mathrm{Id})) && (2)\\ &= det(\lambda\mathrm{Id} - C) && (3) \end{aligned}$$

2.2 Stabilität linearer Netze (diskrete Zeit)

Wir betrachten nun das zeitlich diskretisierte Modell für $j = 1, \ldots, n, \Delta t > 0$

$$\tau \frac{u_j(t + \Delta t) - u_j(t)}{\Delta t} = -u_j(t) + x_j(t) + \sum_{i=1}^{n} c_{ij} u_i(t)$$

Also mit $\varrho = \Delta t / \tau$

$$u_j(t + \Delta t) = (1 - \varrho) u_j(t) + \varrho x_j(t) + \varrho \sum_{i=1}^{n} c_{ij} u_i(t)$$

in Matrixnotation erhalten wir

$$u(t + \Delta t) = (1 - \varrho) u(t) + \varrho x(t) + \varrho u(t) C$$

mit $F = (1 - \varrho)\mathrm{Id} + \varrho C$ folgt:

$$u(t + \Delta t) = u(t) F + \varrho x(t)$$

...

Hieraus folgt:

$$\begin{aligned} u(0) &= u_0 \\ u(\Delta t) &= u_0 F + \varrho x(0) \\ u(2\Delta t) &= u_0 F^2 + \varrho x(0) F + \varrho x(1) \\ u(3\Delta t) &= u_0 F^3 + \varrho x(0) F^2 + \varrho x(1) F + \varrho x(2) \\ \ldots &\quad \ldots \end{aligned}$$

Induktiv folgt $k > 0$:

$$u(k\Delta t) = u_0 F^k + \varrho \sum_{i=1}^{k-1} x(i) F^{k-(i+1)}$$

Falls $x(t) = 0$ für alle t so folgt:

$$u(k\Delta t) = u_0 F^k$$

...

Falls $x(t) = x \in \mathbb{R}^n$ für alle t so folgt:

$$u(k\Delta t) = u_0 F^k + \varrho x \sum_{i=0}^{k-1} F^i$$

Verhalten bei $k \to \infty$:
Es seien $\lambda_1, \ldots, \lambda_n$ die Eigenwerte von F mit den entsprechenden Eigenvektoren $v_1, \ldots, v_n$ und V die Matrix deren Spalten aus den Eigenvektoren v_i besteht.
Sind die Eigenvektoren linear unabhängig so gilt

$$F = VDV^{-1}$$

dabei ist $D = diag(\lambda_1, \ldots, \lambda_n)$ die Diagonalmatrix mit den Eigenwerten in der Diagonalen.

...

Dann gilt:
$F^k = (VDV^{-1})^k = \cdot VDV^{-1} \cdots VDV^{-1} = VD^kV^{-1}$
Also

$$F^k = V \cdot diag(\lambda_1^k, \ldots, \lambda_n^k) \cdot V^{-1}$$

Es gilt offenbar

$$(I - F)\sum_{i=0}^{k-1} F^i = (\sum_{i=0}^{k-1} F^i)(I - F) = I - F^k$$

Falls nun $I - F$ invertierbar ist (gdw alle Eigenwerte von $F \neq 1$ sind) dann gilt:

$$\sum_{i=0}^{k-1} F^i = (I - F^k)(I - F)^{-1}$$

...

Falls alle EW von F vom Betrag kleiner als 1 sind, so folgt

$$\sum_{i=0}^{k-1} F^i \to (I - F)^{-1}, \qquad \text{bei } k \to \infty$$

Damit gilt für:

$$u(k\Delta t) = u_0 F^k + \varrho x \sum_{i=0}^{k-1} F^i$$

falls alle Eigenwerte λ von F gilt $|\lambda| < 1$ so folgt $u(n\Delta t) \to 0$ bei $t \to \infty$.
Sind die Stabilitätsbedingungen für das kontinuierliche und das diskretisierte lineare System äquivalent?

2.3 Äquivalenz der Stabilitätsbedingungen

Es sei $a \in \mathbb{C}$ ein Eigenwert von Kopplungsmatrix C und damit $1 + \varrho(a-1)$ ein Eigenwert von $\mathrm{Id} + \varrho(C - \mathrm{Id})$ mit

$$|1 + \varrho(a-1)| < 1$$

Es sei nun $a = \alpha + i\beta$ mit $\alpha, \beta \in \mathbb{R}$. Dann gilt:

$$1 + \varrho(a-1) = 1 + \varrho(\alpha - 1) + i\varrho\beta$$

Ferner ist $|1 + \varrho(a-1)| < 1$ gdw. $|1 + \varrho(a-1)|^2 < 1$.
Es gilt weiter:

$$|1+\varrho(a-1)|^2 = (1+\varrho(\alpha-1))^2+\varrho^2\beta^2 = 1+2\varrho(\alpha-1)+\varrho^2(\alpha-1)^2+\varrho^2\beta^2$$

Also $|1 + \varrho(a-1)|^2 < 1$ gdw

$$2(\alpha - 1) + \varrho(\alpha - 1)^2 + \varrho\beta^2 < 0$$

Dies ist nun äquvalent zu

$$\varrho < \frac{2(1-\alpha)}{(\alpha-1)^2 + \beta^2} =: b^*$$

...

Hieraus folgt sofort $\alpha = \mathrm{Re}(a) < 1$, denn für $\alpha \geq 1$ folgt $\varrho = \frac{\Delta t}{\tau} \leq 0$.
Falls nun $\alpha < 1$, so gilt $b^* > 0$ und somit gibt es ein ϱ mit der Eigenschaft $0\varrho < b^*$.
Die Betrachtung gilt für beliebiges $\beta = \mathrm{Im}(a) \in \mathbb{R}$.
Die Forderung $\alpha = \mathrm{Re}(a) < 1$ für alle Eigenwerte der Kopplungsmatrix C war gerade die Stabilitätsbedingung an das kontinulierliche lineare System.
Stabilitätsbedingungen für das kontinuierliche und das diskretisierte System sind äquivalent und es muss gelten

$$0 < \Delta t < \tau \frac{2(1-\alpha)}{(\alpha-1)^2 + \beta^2}$$

für alle Eigenwert $a = \alpha + i\beta$ von C.

Einschub: Lyapunov-Funktionen

Betrachten DGL $\dot{x} = g(x)$ mit $x = x(t) = (x_1(t), \ldots, x_n(t))$,
wobei $x_i : \mathbb{R} \to \mathbb{R}$, $t \mapsto x_i(t)$
Eine stetig differenzierbare Funktion $H : D \to \mathbb{R}$ mit $D \subset \mathbb{R}^n$ heißt eine *Lyapunov-Funktion* für die DGL $\dot{x} = g(x)$, wenn gilt:

$$\dot{H}(x) = \langle grad(H(x)), g(x) \rangle = \sum_{i=1}^{n} \frac{\partial H}{\partial x_i} \cdot g_i(x) \leq 0$$

für alle $x \in D$.
Satz: Es ein $\dot{x} = g(x)$ ein DGL System mit isoliertem Fixpunkt x^* (oBdA gelte $x^* = 0$) und H ein nach unten beschränke Ljapunov-Funktion. Dann ist der Fixpunkt asymptotisch lokal stabil.

2.4 Stabilität des kontinuierlichen Hopfieldmodells

Wir wollen zum Grundmodell in kontinuierlicher Zeit

$$\begin{aligned} \tau \dot{u}(t) &= -u(t) + x(t) + y(t)C \quad (4) \\ y(t) &= f(u(t)) \quad (5) \end{aligned}$$

eine Lyapunov-Funktion konstruieren. Der Input sei
$x(t) = x = \textit{konstant}$.
Für den Zustandsvektor $y(t) = (y_1(t), \ldots, y_n(t))$ gilt

$$\begin{aligned} \frac{dH(y(t))}{dt} &= \sum_{i=1}^{n} \frac{\partial H}{\partial y_i}(y(t))\dot{y}_i(t) \\ &= \sum_{i=1}^{n} \frac{\partial H}{\partial y_i} \cdot \frac{dy_i}{du_i} \cdot \dot{u}_i(t) \\ &= \sum_{i=1}^{n} \frac{\partial H}{\partial y_i} \cdot \underbrace{f'(u_i(t))}_{\geq 0} \underbrace{\frac{1}{\tau}}_{>0} (-u_i(t) + x + (yC)_i) \quad (6) \end{aligned}$$

...

Wir werden Lyapunov-Funktion H so konstruieren, dass

$$-\frac{\partial H}{\partial y_i} = -u_i + x_i + (yC)_i$$

ist, dann gilt nämlich

$$\frac{dH(y(t))}{dt} \leq 0.$$

Konstruktion von H aus drei Summanden H_1, H_2, H_3, also $H = H_1 + H_2 + H_3$.

...

1) Die Transferfunktion f sei streng monoton wachsend und somit invertierbar, somit gilt $u_i = f^{-1}(y_i)$. Deshalb setzen wir

$$H_1 = \sum_i \int_0^{y_i} f^{-1}(s)ds.$$

Damit erhalten wir als Ableitung von H_1 nach y_i

$$\frac{\partial H_1}{\partial y_i} = f^{-1}(y_i) = u_i$$

2) Für den zweiten Summanden setzen wir an:

$$H_2 = -\sum_{i=1}^{n} x_i \cdot y_i,$$

Dann ist offenbar

$$\frac{\partial H_2}{\partial y_i} = -x_i.$$

...

3) Es bleibt noch der Anteil $(yC)_i = \sum_{j=1}^{n} y_j c_{ji}$ Wir setzen an:

$$H_3 = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_j c_{ji} y_i,$$

dann ist

$$\frac{\partial H_3}{\partial y_i} = -\frac{1}{2} \left(2 y_i c_{ii} + \sum_{j=1, j \neq i}^{n} c_{ji} y_j + \sum_{j=1, j \neq i}^{n} y_j c_{ij} \right)$$

Sei die Kopplungsmatrix C symmetrisch, d.h. $c_{ij} = c_{ji}$ für alle i, j. Dann gilt:

$$\frac{\partial H_3}{\partial y_i} = -\frac{1}{2} \left(\sum_{j=1}^{n} c_{ji} y_j + \sum_{j=1}^{n} y_j c_{ij} \right) = -\sum_{j=1}^{n} c_{ji} y_j$$

...

Zusammenfassung: $H = H_1 + H_2 + H_3$ eine Lyapunov-Funktion des Grundmodells

$$\tau \dot{u}(t) = -u(t) + x + y(t)C \qquad y(t) = f(u(t))$$

Denn es gilt

$$\begin{aligned} \frac{\partial H}{\partial y_i} &= \frac{\partial H_1}{\partial y_i} + \frac{\partial H_2}{\partial y_i} + \frac{\partial H_3}{\partial y_i} \\ &= u_i - x_i - \sum_j y_j c_{ji} \\ &= -\frac{1}{\tau}\dot{u}_i \end{aligned}$$

und

$$\frac{dH(y(t))}{dt} = -\frac{1}{\tau}\sum_i \dot{u}_i^2(t) \cdot f'(u_i(t)) \leq 0.$$

2.5 Stabilität des diskreten Hopfieldmodells

Diskrete Zeit und ohne Gedächtnis $\Delta t = 1$

$$u_j(t+1) = \sum_{i=1}^{n} c_{ij} y_j(t) + x_j(t)$$

Binäre Ausgabe

$$y_j(t) = \mathrm{sgn}(u_j(t))$$

wobei $\mathrm{sgn}(a) = 1$ falls $a \geq 0$ und $\mathrm{sgn}(a) = -1$ falls $a < 0$.
Voraussetzung: Asynchrone neuronale Dynamik, d.h. es wird zur Zeit t genau ein Neuron j ausgewählt - gemäß der positiven Wahrscheinlichkeiten $p_1, \ldots, p_n$ auf den n Neuronen.

...

Annahme: Zur Zeit t wird Neuron j ausgewählt (Update).
Ferner betrachten wir die Funktion

$$H(y(t)) = H_2(y(t)) + H_3(y(t)) = -\sum_{i=1}^{n} x_i \cdot y_i(t) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_j(t) c_{ji} y_i(t)$$

zur Zeit t, d.h. vor dem Update, und zur Zeit $t+1$, d.h. nach dem Update.
Falls $y_j(t+1) = y_j(t)$, dann ist auch
$\Delta H = H(y(t+1)) - H(y(t)) = 0$.
Falls nun $y_j(t+1) = -y_j(t)$, dann :

$$\Delta H_2 = -\sum_{i=1}^{n} x_i \cdot y_i(t+1) + \sum_{i=1}^{n} x_i \cdot y_i(t) = 2 x_j y_j(t)$$

und

$$\Delta H_3 = -\frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} y_k(t+1) c_{ki} y_i(t+1) + \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} y_k(t) c_{ki} y_i(t)$$

...

Falls C symmetrisch ist, gilt:

$$
\begin{aligned}
\Delta H_3 &= -\sum_{i=1}^{n} y_j(t+1) c_{ij} y_i(t+1) + \sum_{i=1}^{n} y_j(t) c_{ij} y_i(t) \\
&= -y_j(t+1) \sum_{i=1}^{n} c_{ij} y_i(t+1) + y_j(t) \sum_{i=1}^{n} c_{ij} y_i(t) \\
&= -y_j(t+1) \sum_{i=1, i\neq j}^{n} c_{ij} y_i(t+1) - y_j(t+1) c_{jj} y_j(t+1) \\
&\quad + y_j(t) \sum_{i=1, i\neq j}^{n} c_{ij} y_i(t) + y_j(t) c_{jj} y_j(t) \\
&= -y_j(t+1) \sum_{i=1, i\neq j}^{n} c_{ij} y_i(t+1) + y_j(t) \sum_{i=1, i\neq j}^{n} c_{ij} y_i(t) \\
&= 2 y_j(t) \sum_{i=1, i\neq j}^{n} c_{ij} y_i(t) = 2 y_j(t) \sum_{i=1}^{n} c_{ij} y_i(t) - 2 c_{jj}
\end{aligned}
$$

...

Damit folgt nun:

$$\Delta H = \Delta H_2 + \Delta H_3 = 2x_j y_j(t) + 2y_j(t) \sum_{i=1}^{n} c_{ij} y_i(t) - 2c_{jj}$$

Also

$$\Delta H = 2y_j(t)\Big(\sum_{i=1}^{n} c_{ij} y_i(t) + x_j\Big) - 2c_{jj} = 2y_j(t)u_j(t+1) - 2c_{jj}$$

Da das Neuron j den Zustand ändert gilt:

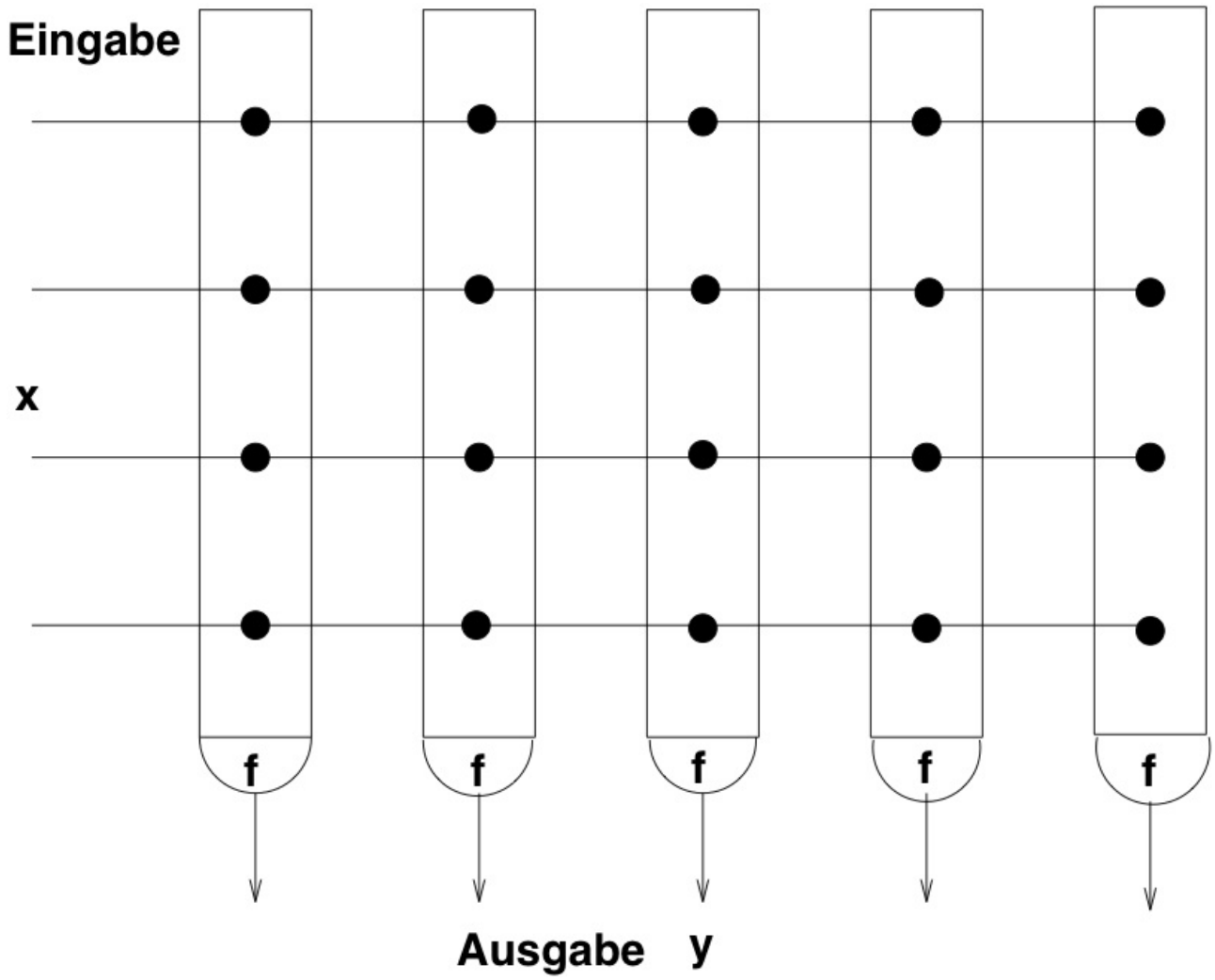
$$\text{sgn}(u_j(t+1)) = y_j(t+1) = -y_j(t)$$

Somit ist $y_j(t)u_j(t+1) \leq 0$ und Gleichheit gdw. $u_j(t+1) = 0$.
Es gelte weiterhin: $c_{jj} \geq 0$ für alle j, d.h. keine negativen Selbstrückkopplungen.
Dann gilt $\Delta H \leq 0$ und Gleichheit gdw. $u_j(t+1) = 0$ und $c_{jj} = 0$

3. Lernen in einschichtigen neuronalen Netzen

1. Architektur und Lernproblem
2. Lineare Assoziativspeicher
3. Binäre Assoziativspeicher
4. Perzeptron-Lernen
5. Support-Vektor-Lernen
6. Neuronale PCA

3.1 Architektur und Lernproblem



Lernproblem

Merkmalsvektor: $x \in \mathbb{R}^d$ bzw $x \in \{0,1\}^d$
Lehrersignal: $T \in \mathbb{R}^n$ oder $T \in \{0,1\}^n$
Ausgabe: $y_j = f(x \cdot w_j)$, $j = 1, \ldots, n$ (ggf. erweiterte Gewichts- und Eingabevektoren)
f eine Transferfunktion, z.B. Sprung-, Signum-, Logistische Funktion
Trainingsmaterial:
$\mathcal{M} = \{(x^\mu, T^\mu) \ : \ \mu = 1, \ldots, M\}$ bzw $\mathcal{M} = \{x^\mu \ : \ \mu = 1, \ldots, M\}$
Gesucht sind Gewichtsvektoren $w_j^* \in \mathbb{R}^d$, $j = 1, \ldots, n$ so dass

$$E(w_1^*, \ldots, w_n^*) \to \min$$

für eine definierte Fehlerfunktionen $E : \mathbb{R}^{nd} \to \mathbb{R}$.

3.2 Lineare Assoziativspeicher

- Lösung durch Pseudo-Inverse
- Approximative Lösung durch Gradientenabstieg

Lösung durch Pseudo-Inverse

Wir betrachten ein Einschichtnetz mit n Neuronen:

$$y_j = f(\sum_{i=1}^{d} x_i w_{ij}) \quad j = 1, \ldots n$$

f eine stetig differenzierbare Funktion mit $f' > 0$.
Dann ist f umkehrbar und statt T^μ benutzt man $f^{-1}(T^\mu)$ als Lehrersignale. Somit neben wir ohne Einschränkung an, $f = \mathrm{id}$ und

$$y_j = \sum_{i=1}^{d} x_i w_{ij} \quad j = 1, \ldots n$$

Quadratische Fehlerfunktion: $E(w) = \sum_\mu \| T^\mu - y^\mu \|_2^2$, dabei ist y^μ die Ausgabe für Input x^μ.

Wir setzen ohne Einschränkung $n = 1$. Damit hat die Fehlerfunktion die Gestalt:

$$E(w) = \sum_{\mu} (T^{\mu} - y^{\mu})^2 = \sum_{\mu} (T^{\mu} - \sum_{i=1}^{d} x_i^{\mu} w_i)^2 \to \min$$

Definiere $T = (T^{\mu})_{1 \leq \mu \leq M}$ und $M \times d$ Matrix $X = (x_i^{\mu})_{\substack{1 \leq \mu \leq M \\ 1 \leq i \leq d}}$.
Fehlerfunktion lässt sich nun schreiben als

$$E(w) = \| T - Xw \|_2^2 \to \min$$

Fehler $= 0$ ist möglich, falls X invertierbar ist

$$Xw = T \quad \text{durch} \quad w = X^{-1} T$$

Diese Lösung ist nur für $M = d$ möglich (uninteressante Lernaufgabe)

..

Minimierung von E_w, d.h. es muss gelten

$$\frac{\partial E_w}{\partial w_k} = 0 \quad \text{für alle } k = 1, \ldots, d$$

$$\frac{\partial E_w}{\partial w_k} = -2 \sum_{\mu} (T^{\mu} - \sum_{i=1}^{d} x_i^{\mu} w_i) x_k^{\mu}$$

Somit folgt

$$\sum_{\mu} (T^{\mu} - \sum_{i=1}^{d} x_i^{\mu} w_i) x_k^{\mu} = 0$$

Hieraus folgt weiter

$$\sum_{\mu} x_k^{\mu} \sum_{i=1}^{d} x_i^{\mu} w_i = \sum_{\mu} x_k^{\mu} T^{\mu} \quad \text{für alle } k = 1, \ldots, d$$

Mit oben definierten Matrizen folgt: $X^t X w = X^t T$

Ist die (symmetrische) $d \times d$ Matrix $X^t X$ invertierbar, so gilt:

$$w = (X^t X)^{-1} X^t T$$

Matrix $(X^t X)^{-1} X^t$ nennt man *Pseudoinverse* von X.
Matrix $X^t X$ ist invertierbar, gdw es d linear unabhängige Eingabevektoren in X gibt.
$X^t X$ nicht invertierbar, so ist $E(w) = \| T - Xw \|_2^2 \to \min$ nicht eindeutig lösbar.
Eindeutigkeit für veränderte Fehlerfunktion

$$E(w) = \| T - Xw \|_2^2 + \alpha^2 \| w \|_2^2 \to \min$$

mit $\alpha > 0$. (Norm des Gewichtsvektor minimal)

Dann folgt offenbar

$$\frac{\partial E}{\partial w_k} = -2\sum_{\mu}(T^{\mu} - \sum_{i=1}^{d} x_i^{\mu} w_i) x_k^{\mu} + 2\alpha^2 w_k \quad \text{fr alle } k = 1, \ldots, d$$

und

$$\sum_{\mu} x_k^{\mu} \sum_{i=1}^{d} x_i^{\mu} w_i + \alpha^2 w_k = \sum_{\mu} x_k^{\mu} T^{\mu} \quad \text{fr alle } k = 1, \ldots, d$$

Mit den oben definierten Matrizen folgt hieraus

$$(X^t X + \alpha^2 I) w = X^t T$$

Für $\alpha \neq 0$ ist $X^t X + \alpha^2 I$ invertierbar (da positiv definit) und es gilt

$$w_\alpha = (X^t X + \alpha^2 I)^{-1} X^t T$$

Zusammenfassung:

1. Für beliebige Matrizen X existiert die Pseudoinverse X^+ :

$$X^+ = \lim_{\alpha \to 0} (X^t X + \alpha^2 I)^{-1} X^t$$

2. Falls $X^t X$ invertierbar ist, so gilt $X^+ = (X^t X)^{-1} X^t$
3. Falls sogar X invertierbar ist, so gilt $X^+ = X^{-1}$
4. In jedem Fall ist

$$w = X^+ T$$

Lösung der Minimierungsaufgabe

$$E(w) = \| T - Xw \|_2^2 \to \min$$

Gradientenabstieg (Delta-Lernregel)

Lernregel als Gradienten-Verfahren:

1. grad $E(w_1, \ldots, w_n)$ ausrechnen (Kettenregel zweimal anwenden) (hier für das Gewicht w_{ij}) :

$$\frac{\partial}{\partial w_{ij}} E = \sum_{\mu=1}^{M} 2 \cdot (T_j^\mu - y_j^\mu) \cdot (-f'(w_j \cdot x^\mu)) \cdot x_i^\mu.$$

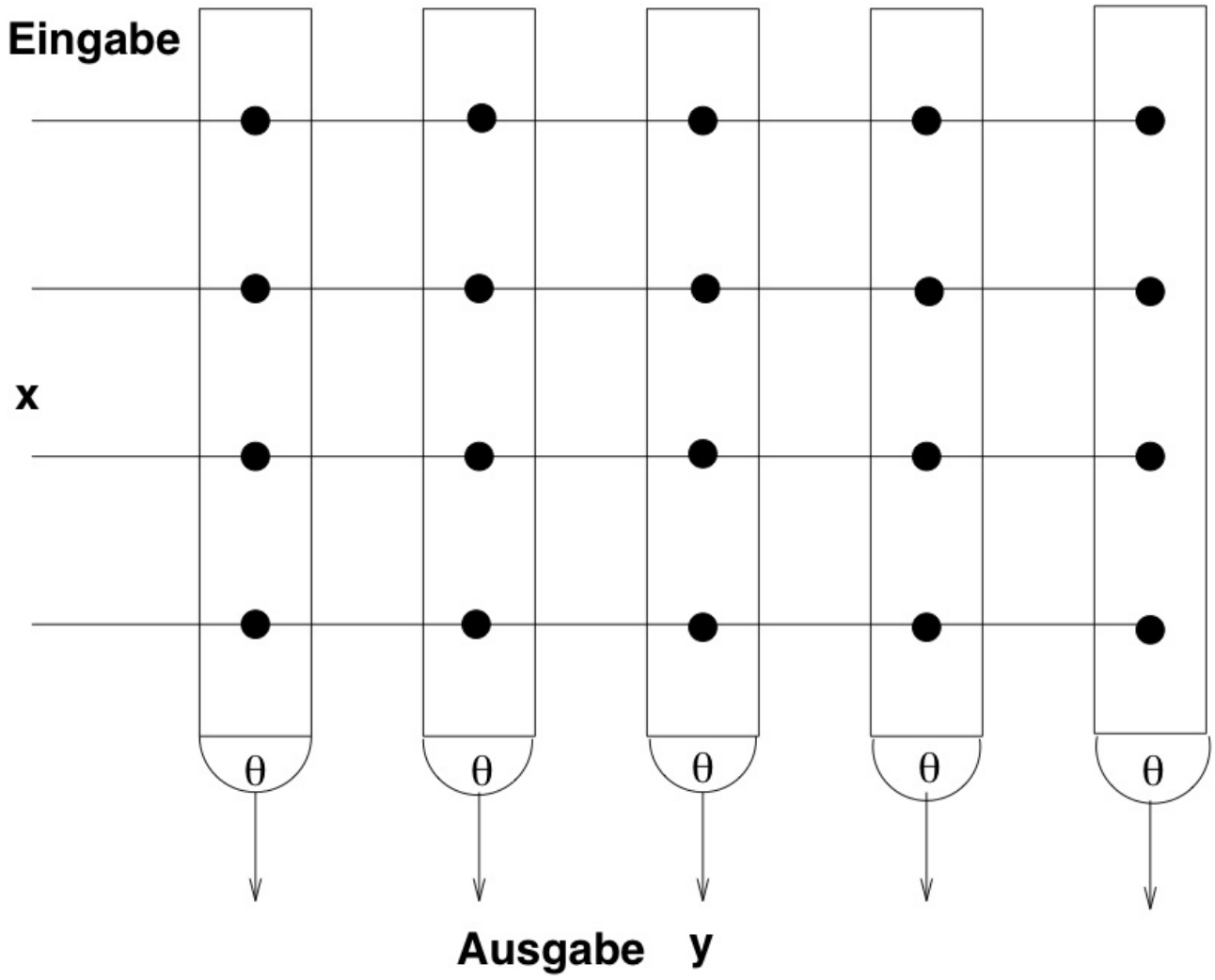
2. Ableitung in die allgemeine Gradientenformel einsetzen liefert:

$$\Delta w_{ij} = l(t) \sum_{\mu=1}^{M} (T_j^\mu - y_j^\mu) f'(w_j \cdot^\mu) x_i^\mu$$

3.3 Binäre Assoziativspeicher

- Architektur, Musterspeicherung
- Hetero-Assoziation
- Auto-Assoziation

Architektur



Einschichtnetz aus n Schwellenneuronen mit Schwelle θ

Musterspeicherung

M binäre Musterpaare (x^μ, T^μ) $(\mu = 1, \ldots, M)$ werden gespeichert durch Hebb-Lernregeln

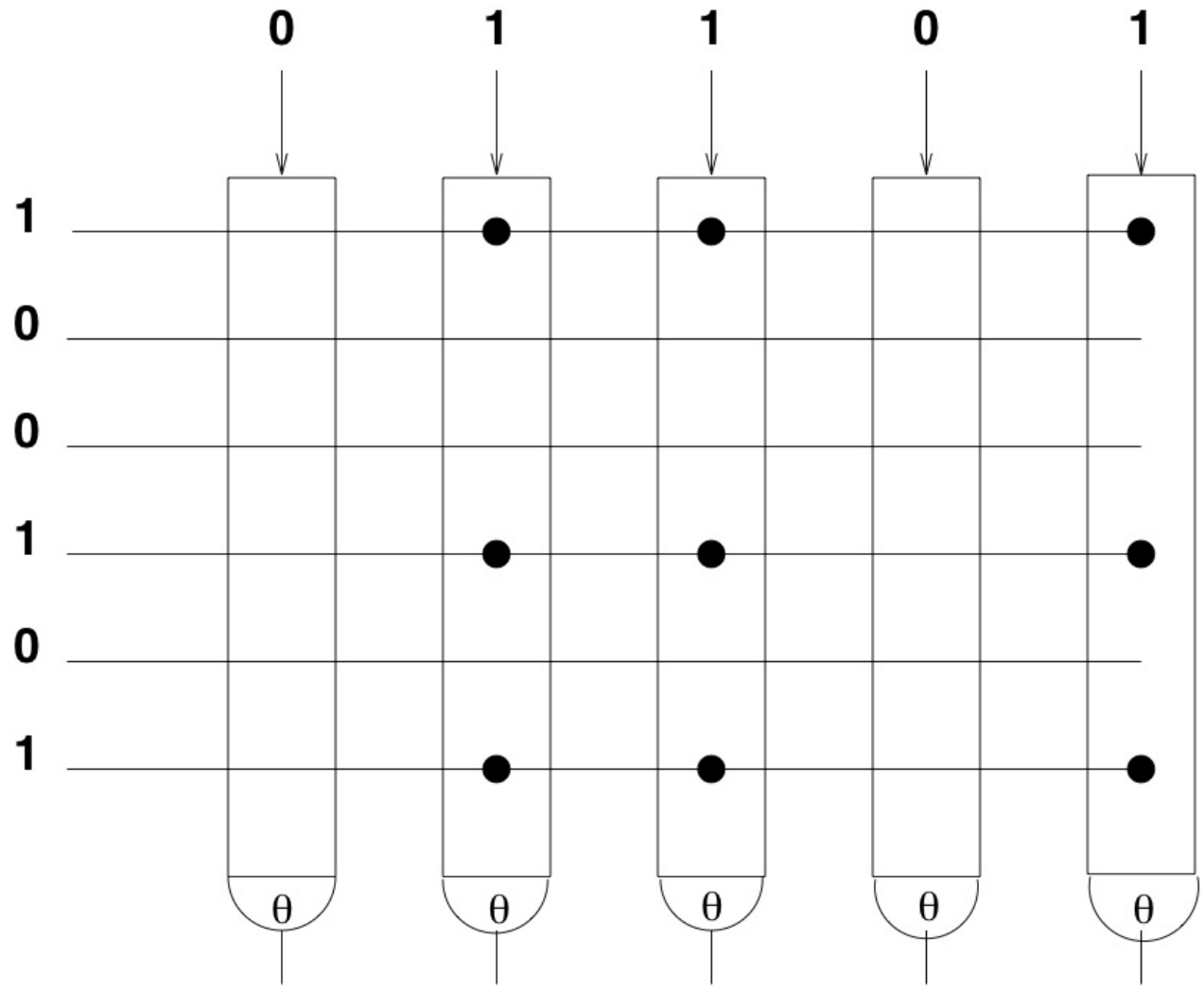
$$c_{ij} = \bigvee_{\mu=1}^{M} x_i^\mu T_j^\mu \qquad \textbf{binäre Hebbregel}$$

$$c_{ij} = \sum_{\mu=1}^{M} x_i^\mu T_j^\mu \qquad \textbf{additive Hebbregel}$$

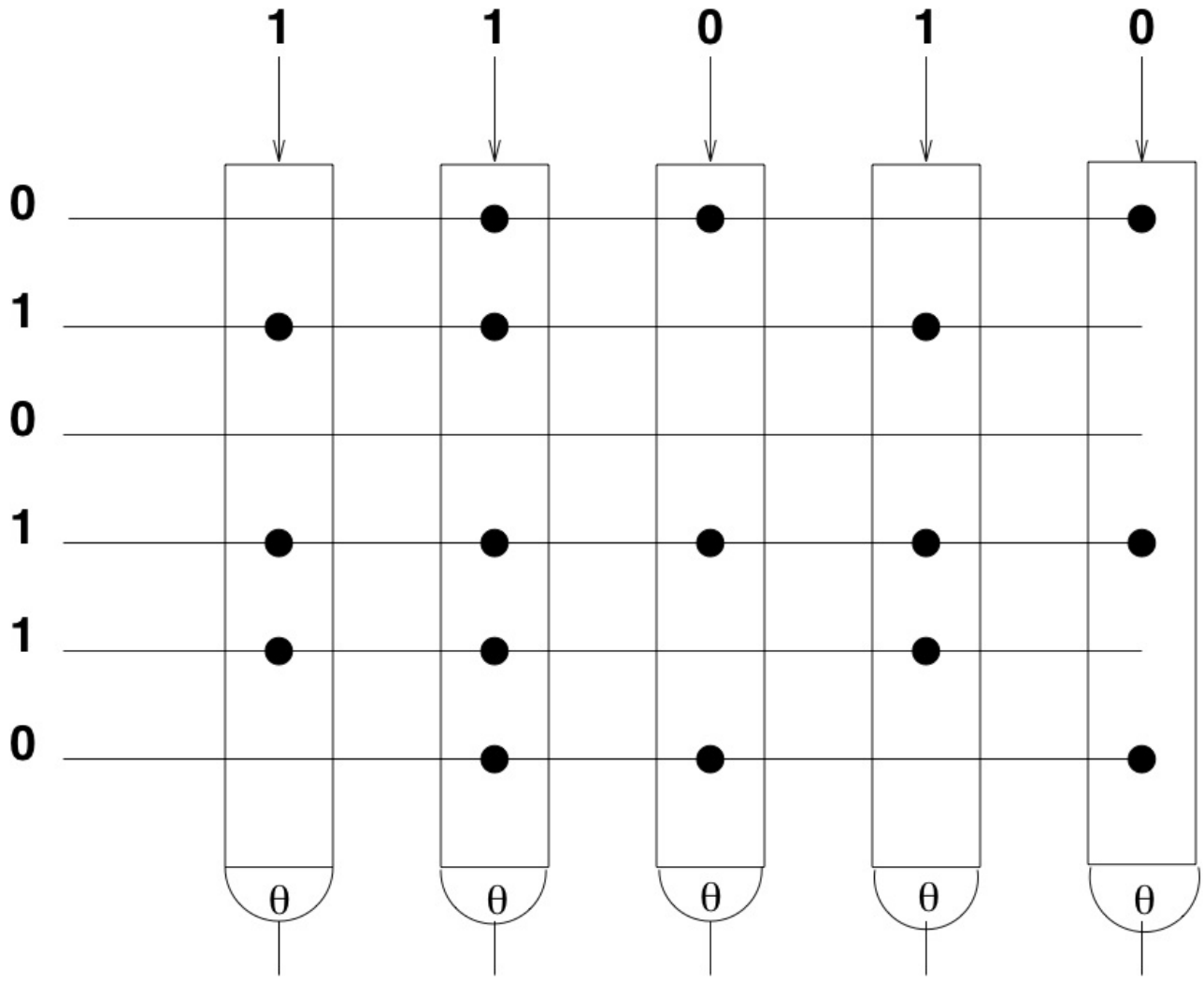
Auslesen des Antwortmusters zur Eingabe x^μ

$$y_j = \begin{cases} 1 & \sum_i x_i^\mu c_{ij} \geq \theta := |x^\mu| \\ 0 & \text{sonst} \end{cases}$$

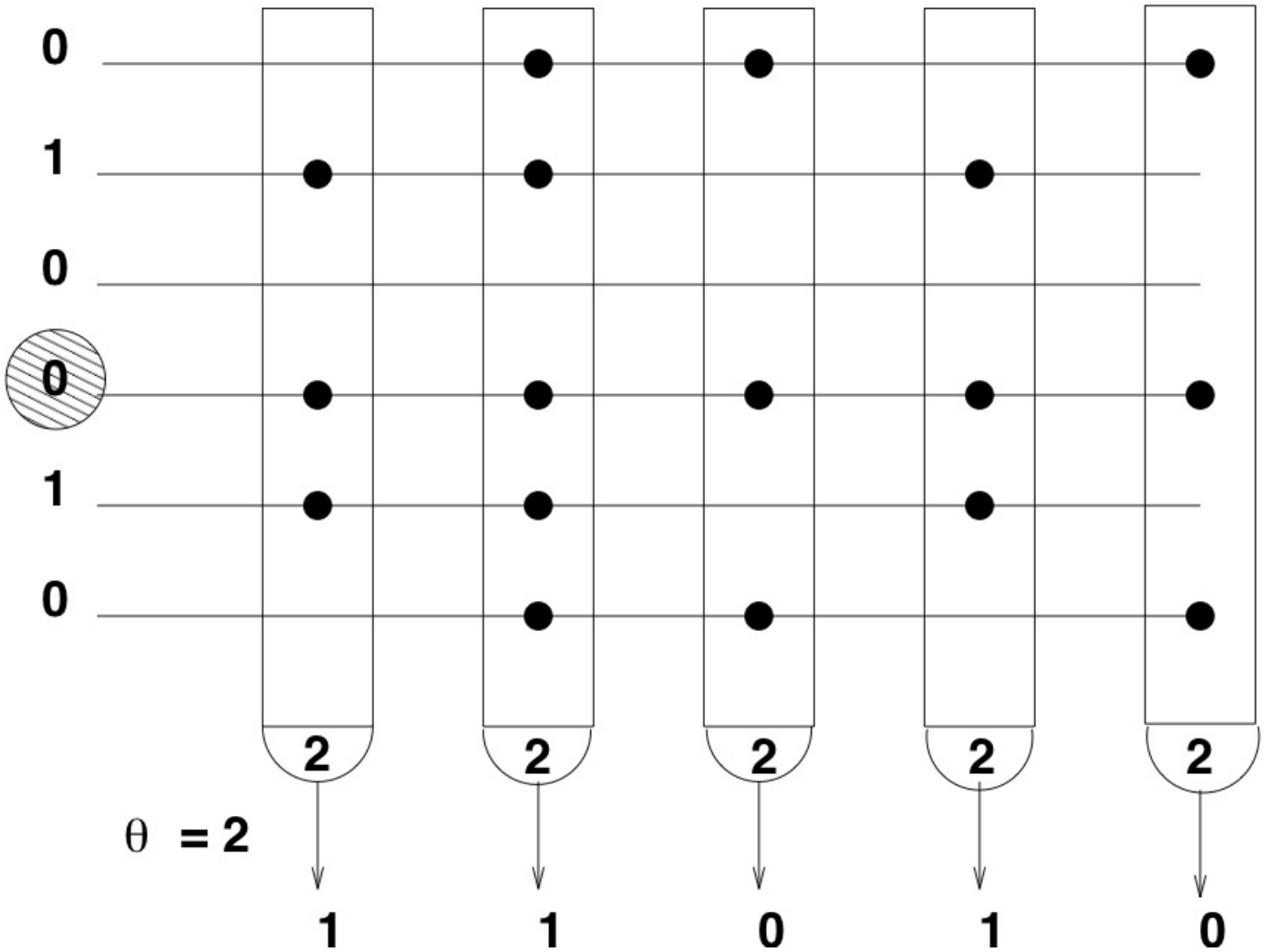
	Input x		Output y
A:	1 0 0 1 0 1	→	0 1 1 0 1
B:	0 1 0 1 1 0	→	1 1 0 1 0



	Input x		Output y
A:	1 0 0 1 0 1	→	0 1 1 0 1
B:	0 1 0 1 1 0	→	1 1 0 1 0



	Input x		Output y
A:	1 0 0 1 0 1	→	0 1 1 0 1
B:	0 1 0 1 1 0	→	1 1 0 1 0



Hetero-Assoziation

Muster $\{(x^\mu, T^\mu) \ : \ \mu = 1, \ldots, M\}$, $x^\mu \in \{0,1\}^m_k$, $T^\mu \in \{0,1\}^n_l$, $p := \frac{k}{m}$, $q = \frac{l}{n}$

Lernregel: $c_{ij} = \bigvee_{\mu=1}^{M} x_i^\mu T_j^\mu$
Retrieval: $z_j = \mathbf{1}_{[x \cdot C \geq \theta]}$ dabei ist $\theta = k$

Fehler: $f_0 = p[z_j = 0 \mid T_j = 1] = 0$ und

$$f_1 = p[z_j = 1 \mid T_j = 0] = (1 - p_0)^k \qquad \Longrightarrow pm = k = \frac{\ln f_1}{\ln(1 - p_0)} \tag{7}$$

Dabei ist

$$p_0 = p[c_{ij} = 0] = (1 - pq)^M \approx e^{-Mpq} \implies M = \frac{-\ln p_0}{pq} \tag{8}$$

Das Retrieval ist sehr genau, wenn $f_1 \leq \delta \cdot q$ mit $\delta < 1$. δ: Gütekriterium

Kapazität bei Hetero-Assoziation

Falls δ klein, dann ist die Information über das Ausgabemuster T^μ:

$$I_\mu \approx n \cdot (-q \log_2 q - (1-q) \log_2 (1-q)) \approx -nq \cdot \log_2 q$$

Relative Speicherkapazität:

$$C = \frac{M \cdot I_\mu}{m \cdot n} = \frac{-I_\mu \ln p_0}{pqmn} = \frac{\ln p_0 \ln(1-p_0)}{qn \cdot \ln f_1} nq \cdot \log_2 q$$

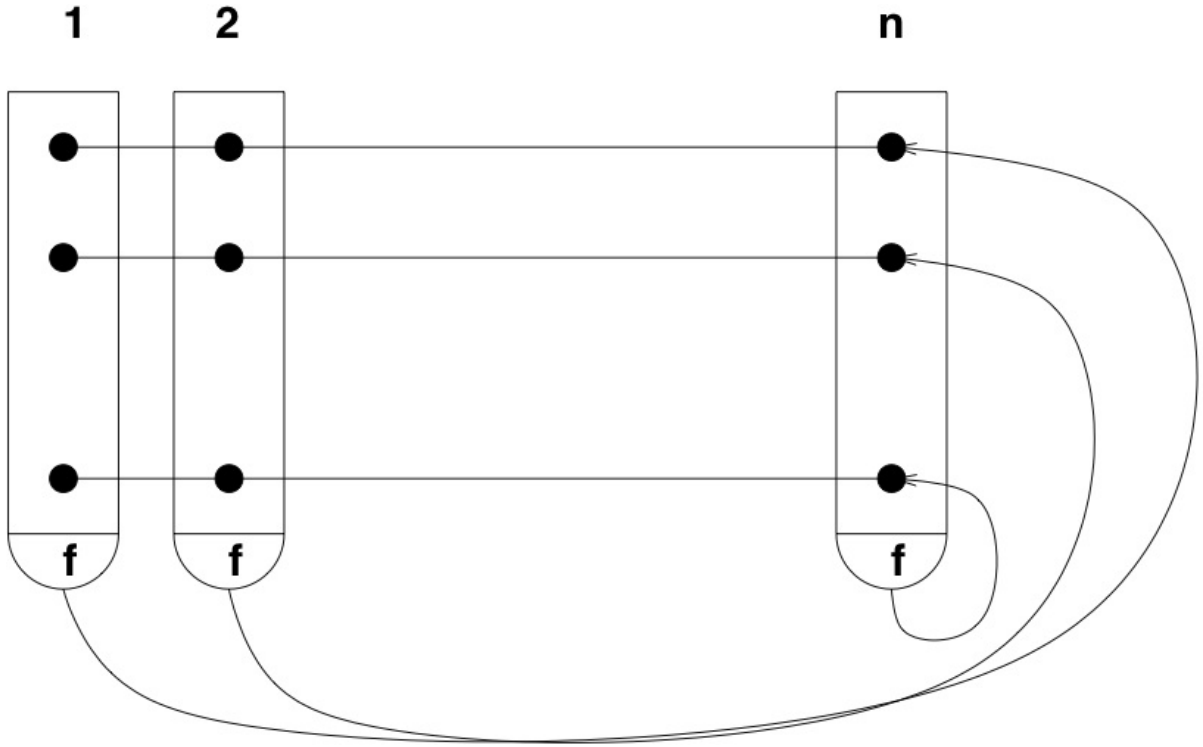
Maximieren nach p_0: $\Longrightarrow p_0 = \frac{1}{2}$ und damit

$$C_{max} = \frac{(\ln 2)^2 \log_2 q}{\ln q + \ln \delta} = \ln 2 \frac{\ln q}{\ln q + \ln \delta} \longrightarrow \ln 2$$

Der Limes wird erreicht für $q \to 0$ und $\delta \to 0$, aber δ langsamer, so dass $\frac{\ln \delta}{\ln q} \to 0$.
Für große Matrizen ist $C \approx \ln 2$ bei kleinem $\delta > 0$ erreichbar, wenn q sehr klein gewählt wird. $\Longrightarrow$ Spärlichkeit.

Auto-Assoziativspeicher

Hetero-Assoziation: $x \neq T$ (pattern mapping)
Auto-Assoziation: $x = T$ (pattern completion)
Für Auto-Assoziation iteratives Retrieval durch Rückkopplung der Netzausgabe:



Speichern und Retrieval

Muster $\{(x^\mu) \ : \ \mu = 1, \ldots, M\}$, $x^\mu \in \{0,1\}_k^m$, , $p := \frac{k}{m}$.

Lernregel: $c_{ij} = \bigvee_{\mu=1}^{M} x_i^\mu x_j^\mu$

Retrieval: $z_j^{t+1} = \mathbf{1}_{[z^t \cdot C \geq \theta^t]}$ dabei $\theta = |z^t|$, $t = 1$ und $\theta^t = k$

Abbruch: $z^t \subset z^{t+1}$ für $t > 1$

Fehler: $f_0 = p[z_j = 0 \mid x_j = 1] = 0$ und

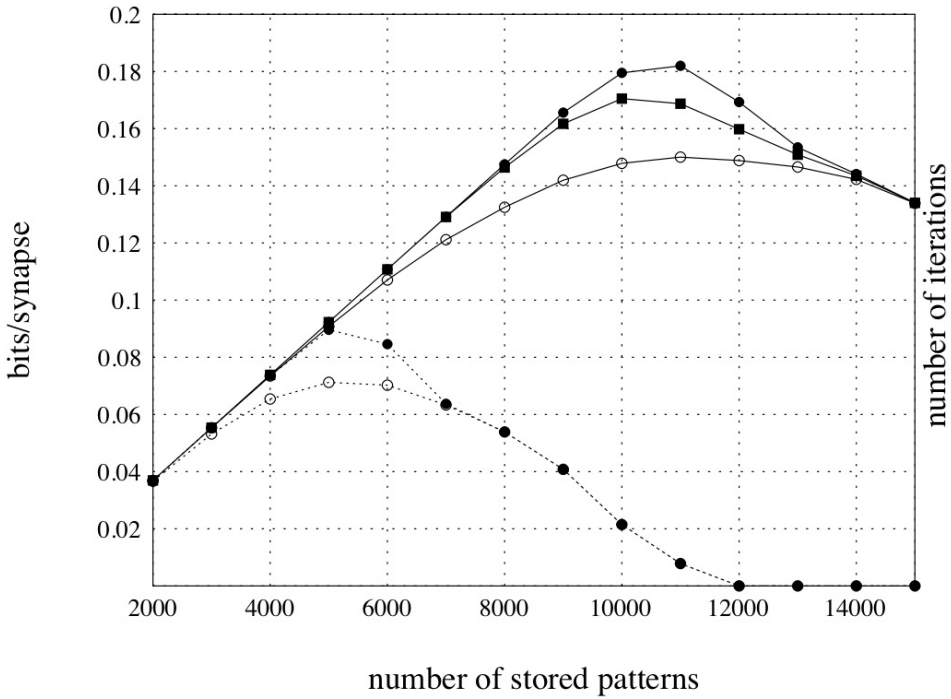
$$f_1 = p[z_j = 1 \mid T_j = 0] = (1 - p_0)^k \qquad \Longrightarrow pm = k = \frac{\ln f_1}{\ln(1 - p_0)} \tag{9}$$

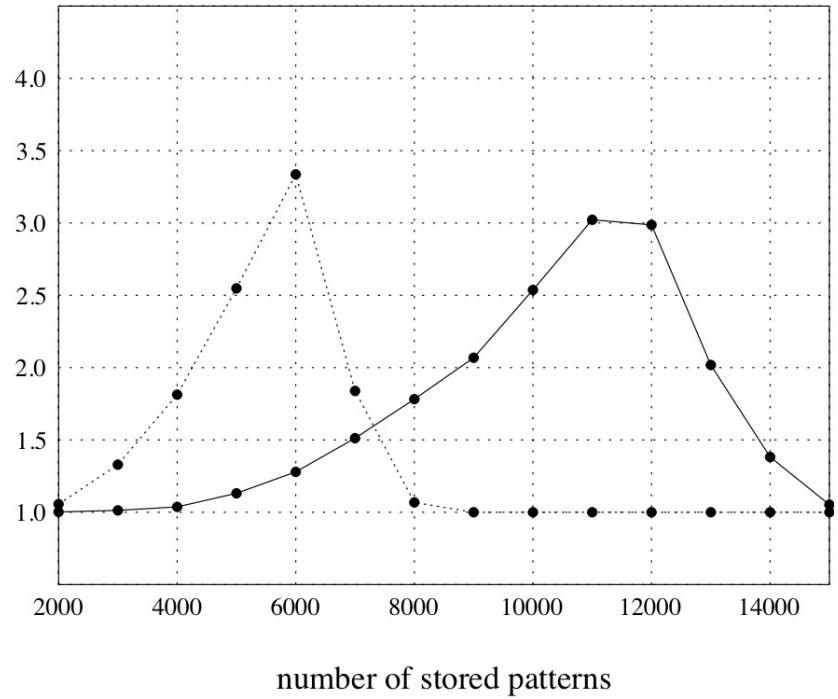
Dabei ist

$$p_0 = p[c_{ij} = 0] = (1 - p^2)^M \approx e^{-Mp^2} \implies M = \frac{-\ln p_0}{p^2} \tag{10}$$

Das Retrieval ist sehr genau, wenn $f_1 \leq \delta \cdot p$ mit $\delta < 1$. δ: Gütekriterium

Speicherkapazität und mittlere Iterationszeit



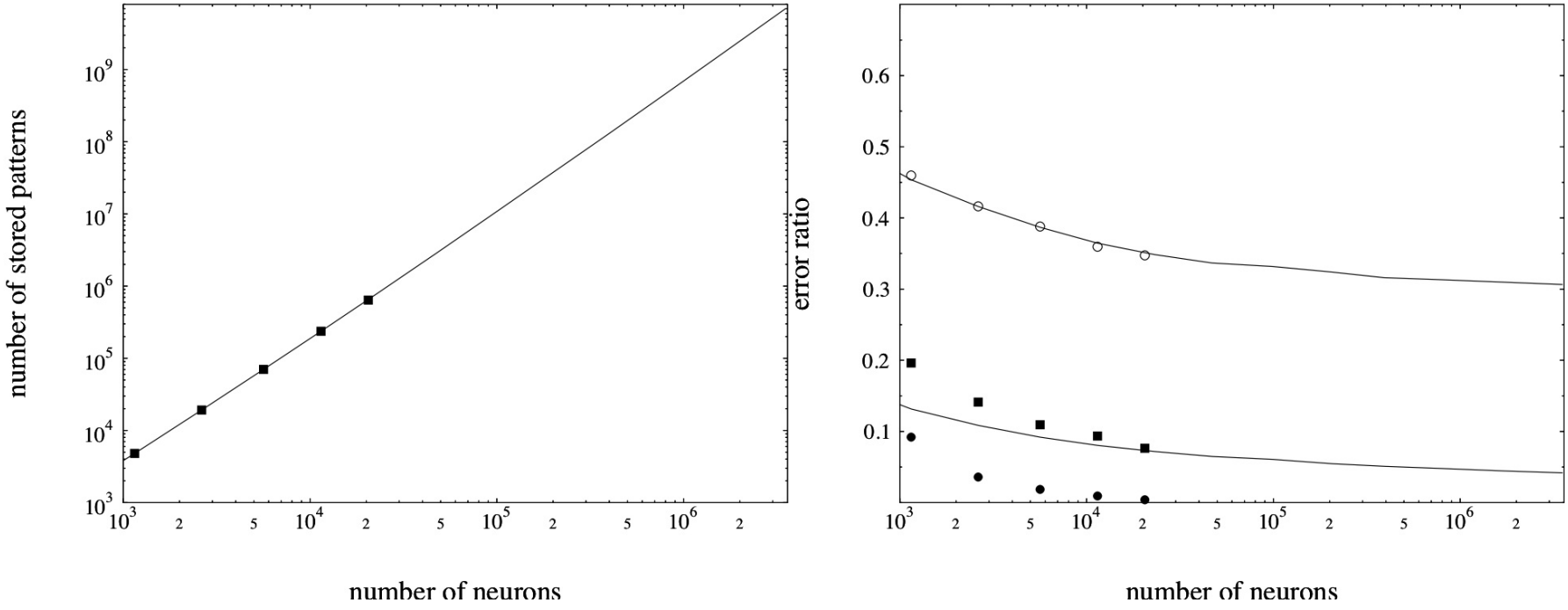


- Autoassoziation mit n=2048 Neuronen; Binäre (durchgezogene Linie)/additive Hebb-Lernregel.
- 1-Schritt- (○), 2-Schritt- (■), und Fixpunkt-Retrieval (●)

Resultate

- Höhere Speicherkapazität mit binärer Hebbregel !
- Nur wenige Iterationsschritte (≈ 3) notwendig.

Fehlerwahrscheinlichkeit und Musterzahl



- Näherungsrechung für 1-Schritt- und 2-Schritt-Retrieval
- Experimentelle Ergebnisse für 1-Schritt- (○), 2-Schritt- (■) und Fixpunkt-Retrieval (●)

Resultate

- Fehlerwahrscheinlichkeit ist klein (für große Netze $\rightarrow 0$)
- Auslesegüte wird durch iteratives Retrieval verbessert

3.4 Perzeptron Lernen

Trainingsmaterial :

$$\mathcal{T} = \{(x^\mu, T^\mu) \ : \ \mu = 1, \ldots, M\} \subset \mathbb{R}^d \times \{-1, 1\}$$

Wir untersuchen die Perzeptron-Lernregel:

$$\Delta w = l\,(T - y) \cdot x \text{ mit Lehrersignal } T \in \{-1, 1\}$$

mit Eingabe $x \in \mathbb{R}^d$, Lernrate l und $y = \text{sign}(x \cdot w)$
(Schwelle θ als Gewicht w_1 mit konstanter Eingabe $x_1^\mu = 1$ für alle Muster μ).
Andere Schreibweise der Perzeptron-Lernregel:

$$\Delta w = -l\,\text{sign}(x \cdot w) \cdot x = l\,T \cdot x \quad \text{falls } T \neq y \text{ (Änderungsschritt)}$$

- Das Problem ist lösbar, falls ein Gewichtsvektor w mit $\text{sign}(x^\mu \cdot w) = T^\mu$ für alle μ exisitiert d.h. $T^\mu (x^\mu \cdot w) > 0$ $\forall \mu$, d.h. $D(w) := \min_{\mu=1}^{M} T^\mu (x^\mu \cdot w) > 0$.
- Die Funktion $D(w)$ nimmt auf der Einheitskugel $K = \{w \ : \ w \cdot w = 1\}$ das Maximum d an.
- Also gibt es w^* mit $w^* \cdot w^* = 1$ und $D(w^*) = d$.
- Separierungsproblem ist lösbar, falls $d > 0$.
- Setze $c := \max_{\mu=1}^{M}(x^\mu \cdot x^\mu)$.

Zu bestimmen ist die Anzahl der Änderungsschritte S.

Betrachten dazu das Gewicht w_S nach S Änderungsschritten:
$w_S = \sum_{i=1}^{S} (\Delta w)_i$ mit Startwert $w = 0$
Dann gilt (wegen der alternativen Formulierungen der Lernregel)

$$(\Delta w) \cdot w^* = l \, T^\mu \, (x^\mu \cdot w^*) \geq l \, D(w^*) = l \, d$$

und

$$(w + \Delta w) \cdot (w + \Delta w) - w \cdot w = 2 \, ((\Delta w) \cdot w) + (\Delta w) \cdot (\Delta w) = \ldots$$

$$= -2 \, l \, \mathrm{sign}(x^\mu \cdot w) \, (x^\mu \cdot w) + l^2 \, (x^\mu \cdot x^\mu) \leq l^2 \, (x^\mu \cdot x^\mu) \leq l^2 \, c$$

Also gilt: $w_S \cdot w_S \leq S l^2 c$ und $w_S \cdot w^* \geq S l d$.
Daraus folgt dann mit Hilfe der Cauchy-Schwarz-Ungleichung:

$$S\, l\, d \overset{(3)}{\leq} w_S \cdot w^* \leq \sqrt{(w_S \cdot w_S)(w^* \cdot w^*)} = \sqrt{w_S \cdot w_S} \leq \sqrt{S\, l^2\, c}$$

$$\Longrightarrow S \leq c/d^2$$

- Also konvergiert der Perzeptron-Lernalgorithmus nach endlich vielen (echten) Lernschritten.
- Der Gewichtsvektor kann durch $w = 0$ initialisiert werden.
- Der Konvergenz-Beweis gilt für beliebige konstante positive Lernrate.

3.5 Support Vektor Lernen

Ist eine spezielle Form des Perzeptron-Lernverfahrens.
Lernverfahren entsteht durch eine Kombination von 2 Zielen; diese legen dann im Fall linear separierbarer Mengen eine eindeutige Trennhyperebene fest.
Gegeben Trainingsdaten

$$\mathcal{T} = \{(x^\mu, T^\mu) \ : \ \mu = 1, \ldots, M\} \subset \mathbb{R}^d \times \{-1, 1\}$$

Wir nehmen zunächst einmal an, die Mengen

$$P = \{x^\mu \mid T^\mu = 1\} \quad \text{und} \quad N = \{x^\mu \mid T^\mu = -1\}$$

seien linear separierbar.
Das Perzeptron-Lerntheorem sichert die Konvergenz nach endlich vielen Schritten gegen eine Lösung w (erweiterter Gewichtsvektor).

Die Lösung w^* beim SV Lernen soll erfüllen:

1. Separationsbedingungen:

$$T^\mu(\langle w, x^\mu \rangle + w_0) > 0 \quad \text{fr alle } \mu = 1, \ldots, M$$

2. Möglichst großen Abstand den Mengen N und P hat (*maximal margin*)

Es sein

$$\min_\mu \quad T^\mu(\langle w, x^\mu \rangle + w_0) = \delta > 0$$

Nun reskalieren wir und erhalten $w = \frac{1}{\delta} w$ und $w_0 = \frac{1}{\delta} w_0$, sowie

$$T^\mu(\langle w, x^\mu \rangle + w_0) \geq 1 \quad \text{fr alle } \mu = 1, \ldots, M$$

Offenbar gibt es mindestens einen Punkt $x^\nu \in P$ und $x^\mu \in N$ mit

$$\langle w, x^\nu \rangle + w_0 = 1$$

und mit

$$\langle w, x^\mu \rangle + w_0 = -1$$

Daraus folgt $\langle w, x^\nu - x^\mu \rangle = 2$ und damit ist $D(w)$ die Breite des Randes der separierenden Hyperebene gegeben durch

$$D(w) = \langle \frac{w}{\|w\|_2}, (x^\nu - x^\mu) \rangle = \frac{2}{\|w\|_2}$$

Maximierung des Randes bedeutet Minimierung von

$$\varphi(w) = \frac{\|w\|_2^2}{2} \to \min$$

unter den Nebenbedingungen

$$T^\mu(\langle w, x^\mu \rangle + w_0) \geq 1 \quad \text{für alle } \mu = 1, \ldots, M$$

Dies ist ein quadratisches Optimierungsproblem unter Nebenbedingungen.
Einführung von Lagrange Multiplikatoren $\alpha_\mu \geq 0$ für jeder Separationsbedingung $\mu = 1, \ldots, M$ ergibt:

$$L(w, w_0, \alpha) = \frac{\|w\|_2^2}{2} - \sum_{\mu=1}^{M} \alpha_\mu \left(T^\mu(\langle w, x^\mu \rangle + w_0) - 1 \right)$$

Setzt man nun die partiellen Ableitungen $\frac{\partial L}{\partial w} = 0$ und $\frac{\partial L}{\partial w_0} = 0$, so folgen die Bedingungen

$$\sum_{\mu=1}^{M} \alpha_\mu T^\mu = 0 \quad \text{und} \quad w = \sum_{\mu=1}^{M} \alpha_\mu T^\mu x^\mu$$

Außerdem folgt aus den Optimierungsbedingungen

$$\alpha_\mu \left[T^\mu \left(\langle w, x^\mu \rangle + w_0 \right) - 1 \right] = 0 \quad \text{für alle } \mu = 1, \ldots, M$$

Falls nun $\alpha_\mu \neq 0$ so folgt: $T^\mu \left(\langle w, x^\mu \rangle + w_0 \right) = 1$, d.h. für solche Trainingsbeispiele liegt x^μ genau auf dem Rand.
Diese Vektoren heißen **Support Vektoren**. Offensichtlich ist w eine Linearkombination der Support Vektoren (geometrisch ist dies (jedenfalls im $\mathbb{R}^2$) klar).

$$w = \sum_{x^\mu \in SV} \alpha_\mu T^\mu x^\mu$$

Einsetzen in L ein gibt quadratische Funktion:

$$W(\alpha) = \sum_{\mu=1}^{M} \alpha_\mu - \frac{1}{2} \sum_{\nu=1}^{M} \sum_{\mu=1}^{M} \alpha_\nu \alpha_\mu T^\nu T^\mu \langle x^\nu, x^\mu \rangle$$

das mit $\alpha_\mu \geq 0$ für alle $\mu = 1, \ldots, M$ zu maximieren ist.
Die Lösung α^* steht nun fest:

$$w^* = \sum_{\mu=1}^{M} \alpha_\mu^* T^\mu x^\mu$$

Schwelle $w_0^* \in \mathbb{R}$ mit Hilfe eines Supportvektors x^{μ_0} bestimmen, denn hierfür gilt $\alpha_{\mu_0} \neq 0$ und wegen KKT-Bedingung:

$$T^{\mu_0} \left(\langle w, x^{\mu_0} \rangle + w_0 \right) = 1 \Longrightarrow w_0^* = \frac{1}{T^{\mu_0}} - \langle w, x^{\mu_0} \rangle$$

damit liegt die Entscheidungsfunktion fest:

$$F(x) = \text{sgn} \ \left(\langle w^*, x \rangle + w_0^* \right) = \text{sgn} \ \left(\sum_{x^\mu \in SV} \alpha_\mu^* T^\mu \langle x^\mu, x \rangle + w_0^* \right).$$

Linear nichtseparierbares Problem

$P = \{x^\mu \mid T^\mu = 1\}$ und $N = \{x^\mu \mid T^\mu = -1\}$ seien linear nicht separierbar

Soft Separationsbedingungen durch Schlupfvariable $\delta_\mu \geq 0$ (*slack variables*)

$$T^\mu \left(\langle w, x^\mu \rangle + w_0 \right) \geq 1 - \delta_\mu \quad \text{für alle } \mu = 1, \ldots, M$$

Nun minimieren wir mit $C > 0$

$$\varphi(w, \delta) = \frac{1}{2} \|w\|_2^2 + \frac{C}{M} \sum_{\mu=1}^{M} \delta_\mu$$

Dies gibt wiederum auf die quadratische Funktion

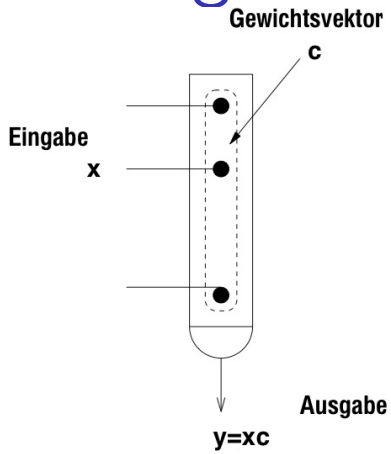
$$W(\alpha) = \sum_{\mu=1}^{M} \alpha_\mu - \frac{1}{2} \sum_{\nu=1}^{M} \sum_{\mu=1}^{M} \alpha_\nu \alpha_\mu T^\nu T^\mu \langle x^\nu, x^\mu \rangle$$

die mit $0 \leq \alpha_\mu \leq C/M$ für alle $\mu = 1, \ldots, M$ zu maximieren ist.

3.6 Neuronale PCA

1. Hebb-Lernregel
2. Oja-Lernregel
3. Sanger-Lernregel

Hebb-Lernregel



Lineares Neuron mit Hebb-Lernregel

- Lineare Verrechnung der Eingabe x und dem Gewichtsvektor w

$$y = \langle x, w \rangle = \sum_{i=1}^{n} x_i w_i$$

- Hebb-Lernregel

$$w := w + lxy = w + lxx^t w$$

- Normierte Hebb-Lernregel

Hauptachsentransformation

- Die Gesamt-Varianz in Richtung v ist dann

$$\sigma_v^2 = (Xv)^t(Xv) = v^t X^t X v = v^t C v$$

  mit $C = X^t X$.
- Bezglich der Matrix C soll nun σ_v^2 maximiert werden.
- Ohne Randbedingungen an v ist eine Maximierung nicht mglich.
- Normierung als Bedingung: $v^t v = \|v\|^2 = 1$
- Maximierung unter Nebenbedingungen fhrt auf die Maximierung der Funktion.

$$\varphi(v) = v^t C v - \lambda(v^t v - 1)$$

  mit dem **Lagrange Multiplikator** $\lambda \in \mathbb{R}$.
- Differenzieren von φ nach v und Nullsetzen liefert:

$$\frac{\partial \varphi}{\partial v} = 2Cv - 2\lambda v = 0$$

- Dies fhrt direkt auf die Matrixgleichung in Eigenvektorform

$$Cv = \lambda v$$

Analyse der Hebb-Lernregel

Gegeben seien also Eingabevektoren $x^\mu \in \mathbb{R}^d$, die einzelnen Merkmale (= Spaltenvektoren in der Datenmatrix X) haben den Mittelwert $E(x_i) = 0$. sonst Mittelwerttranslation durchführen.
Lineares Neuron: $y^\mu := \langle x^\mu, w \rangle = (x^\mu)^t w$.
Dieses Neuron realisiert eine Projektion von x^μ auf w. Somit ist Xw der Vektor der Datenpuntprojektionen.
Hebb-Regel : $\Delta w = \alpha x y = \alpha x (x^t w) = \alpha (x x^t) w$, mit $\alpha > 0$
Wir setzen ferner : $J(w) := -\frac{1}{2} y^2 = -\frac{1}{2} (x^t w)^2$
dann ist offenbar $\frac{\partial J}{\partial w_i} = -\frac{1}{2} 2 x_i y$
also ist $\Delta w = -\alpha \frac{\partial J}{\partial w}$.

...

Wir nehmen zunächst an, es gibt eine Gleichgewichtslösung für w, dann gilt $0 = E(\Delta w)$ also folgt dann

$$0 = E(\Delta w) = \alpha E(xx^t)w = \alpha Cw$$

Also muss gelten $Cw = 0$. Dabei ist C die Korrelationsmatrix der Datenmatrix X mit $C_{ij} = E(x_i x_j)$.
w ist dann Eigenvektor von C zum Eigenwert 0, diese Lösung kann aber nicht stabil sein, denn es gibt sicher positive Eigenwerte von C.

- C ist i.A. **ungleich** der Kovarianzmatrix
  $Cov := E((x - \mu)(x - \mu)^t)$, es sei denn $\mu = E(x) = 0$.
- C ist symmetrisch und positiv semi-definit, d.h. alle Eigenwert sind $\in \mathbb{R}_{\geq 0}$ und die Eigenvektoren orthogonal und

$$u^t Cu = u^t E(xx^t)u = E((u^t x)(x^t u)) = E((u^t x)^2) \geq 0.$$

Oja-Lernregel

Lernregel nach Oja

$$\Delta w = l_t(yx - y^2 w) = l_t y(x - yw)$$

Satz von Oja (1985): Gewichtsvektor w konvergiert gegen die 1. Hauptachse v_1, hierbei muss gelten: $l_t \to 0$ bei $t \to \infty$, $\sum_t l_t = \infty$ und $\sum_t l_t^2 < \infty$.

Sanger-Lernregel

- Verallgemeinerung auf $d' \leq d$ lineare Neuronen mit d' Gewichtsvektor w_j. Die Ausgabe des j-ten Neurons ist dabei:

$$y_j = \langle x, w_j \rangle$$

- Lernregel nach Sanger

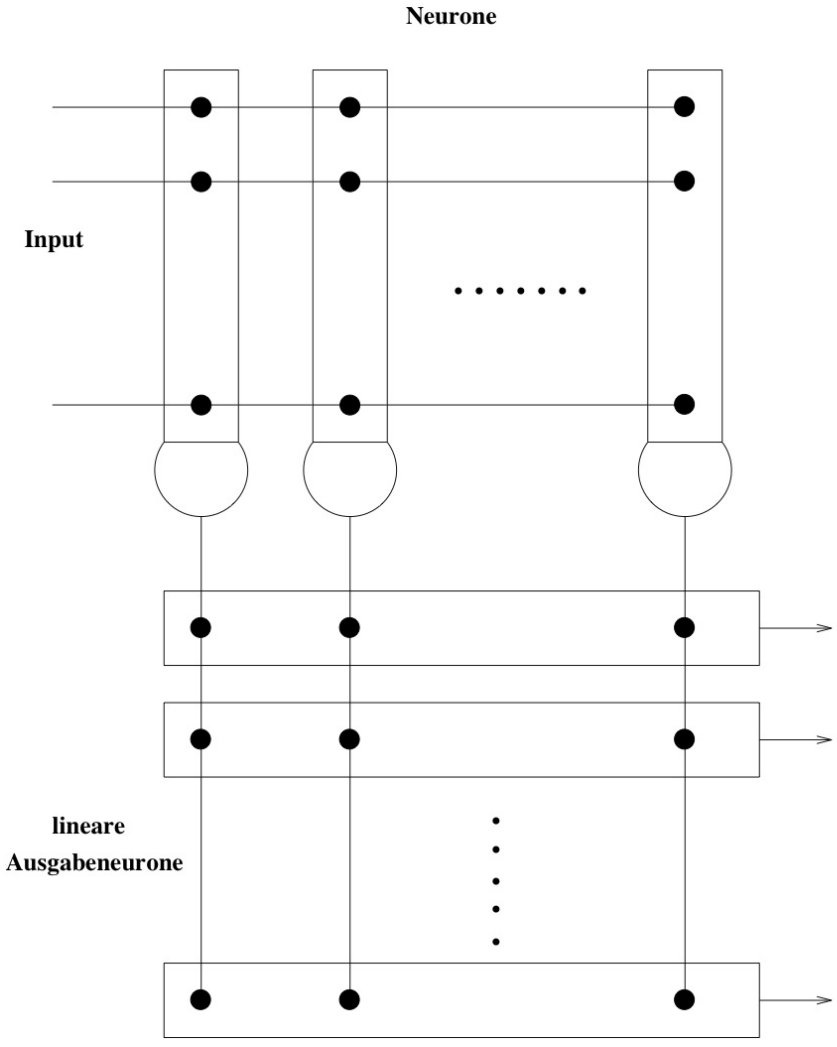
$$\Delta w_{ij} = l_t y_j (x_i - \sum_{k=1}^{j} y_k w_{ik})$$

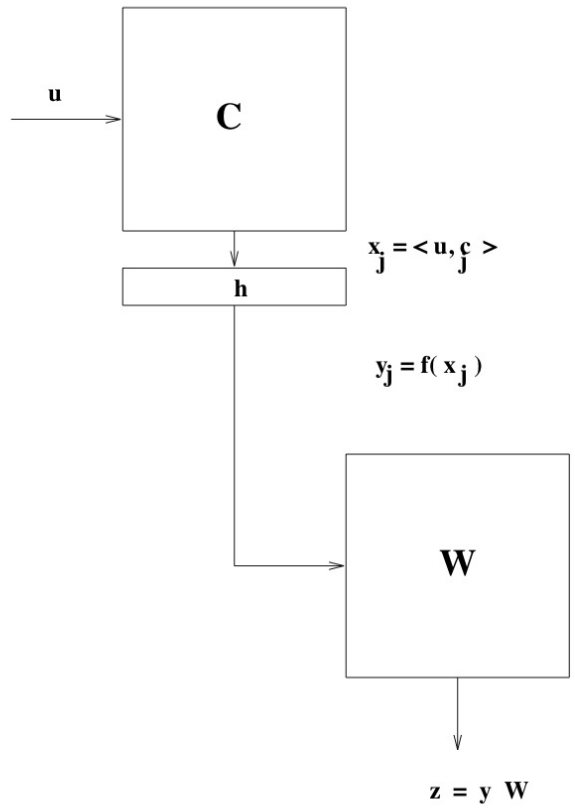
- **Satz von Sanger** (1989): w_l konvergiert gegen die Hauptachsen der Eingabevektoren x^μ. Es muss gelten $l_t \to 0$ bei $t \to \infty$, $\sum_t l_t = \infty$ und $\sum_t l_t^2 < \infty$.

4. Lernen in Mehrschichtnetzen

1. Multilayer Perzeptrone
2. Radiale Basisfunktionen Netze
3. Nichtlineares Support-Vektor-Lernen
4. Zusammenfassung

4.1 Multilayer Perzeptrone





Lernregeln für Multilayer Perzeptrone

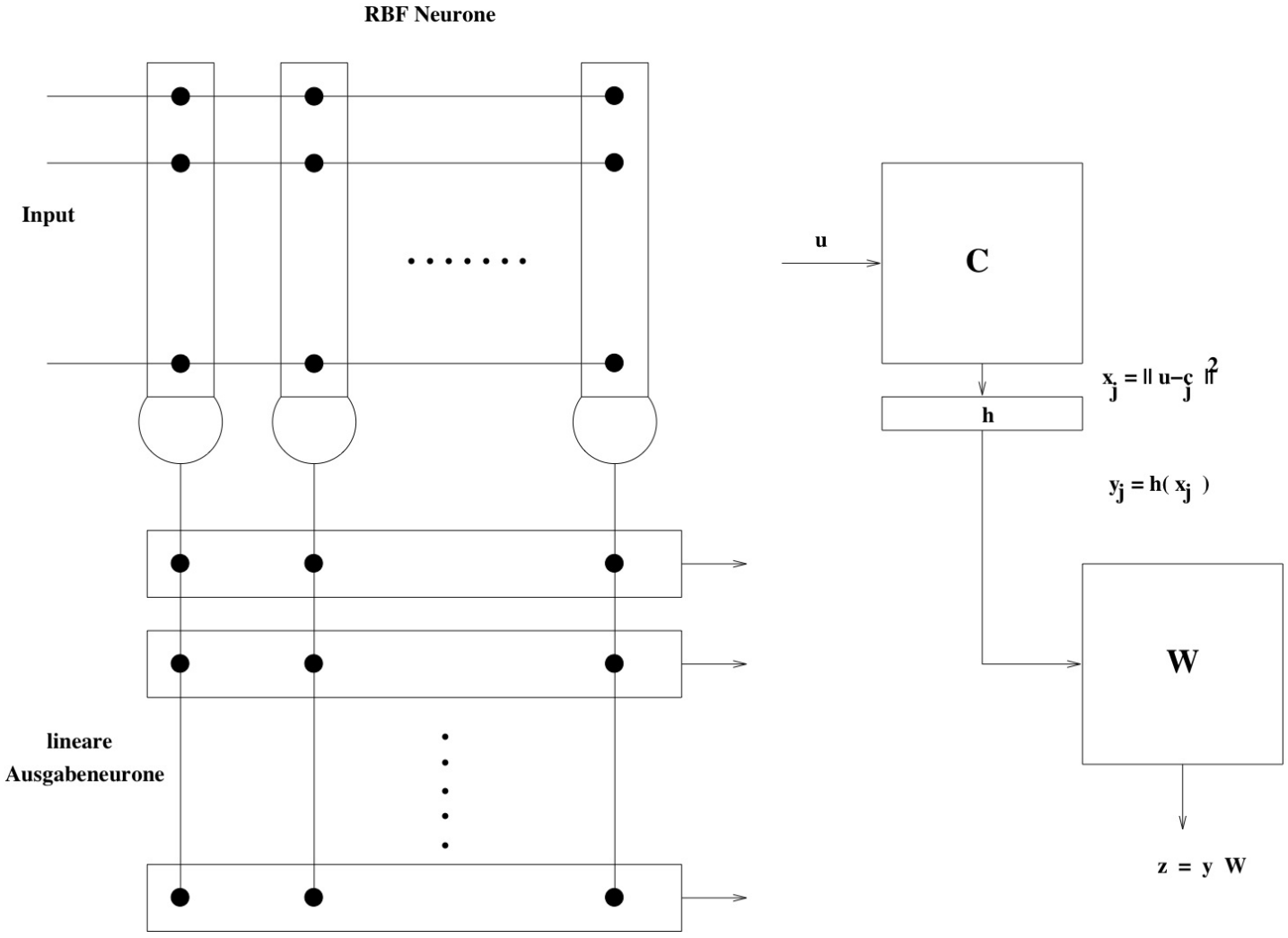
Merkmalsvektor: $x \in \mathbb{R}^n$ Ausgabe: $z_k = \sum_j w_{jk} f(\|x - c_j\|^2)$.
Material: $\mathcal{M} = \{(x^\mu, T^\mu) \ : \ \mu = 1, \ldots, M\}$ $x^\mu \in \mathbb{R}^n$, $T^\mu \in \mathbb{R}^m$.
Lernregel für die Ausgabeschicht:

$$\Delta w_{jk} = l(T_k^\mu - z_k^\mu) \cdot y_j^\mu$$

Lernregel für die Neuronen der Zwischenschicht:

$$\Delta c_{ij} = l \sum_{k=1}^{m} (T_k^\mu - z_k^\mu) \cdot w_{jk} \cdot f'(u_j^\mu) \cdot c_{ij}.$$

4.2 Radiale Basisfunktionen Netze



Lernregeln für Radiale Basisfunktionen

Merkmalsvektor: $x \in \mathbb{R}^n$ Ausgabe: $z_k = \sum_j w_{jk} h(\|x - c_j\|^2)$.
Material: $\mathcal{M} = \{(x^\mu, T^\mu) \ : \ \mu = 1, \ldots, M\}$ $x^\mu \in \mathbb{R}^n$, $T^\mu \in \mathbb{R}^m$.
Lernregel für die Ausgabeschicht:

$$\Delta w_{jk} = l(T_k^\mu - z_k^\mu) \cdot y_j^\mu$$

Lernregel für die RBF Neuronen

$$\Delta c_{ij} = l \sum_{k=1}^{m} (T_k^\mu - z_k^\mu) \cdot w_{jk} \cdot (-h'(u_j^\mu)) \cdot (x_i^\mu - c_{ij}).$$

Interpolation mit RBF

$\mathcal{M} = \{(x^\mu, t^\mu) \ : \ \mu = 1, \ldots, M\}$, $x^\mu \in \mathbb{R}^n$, $t^\mu \in \mathbb{R}^m$. OBdA sei $m = 1$. Gesucht ist $G : \mathbb{R}^n \to \mathbb{R}$ mit

$$G(x^\mu) = t^\mu \quad \forall \mu = 1, \ldots, M. \tag{11}$$

Lösung bei RBF: G als Linearkomination von Funktionen $H_\mu : \mathbb{R}^n \to \mathbb{R}$ mit

$$H_\mu(x) := h(||x - x^\mu||^2) \quad \mu = 1, \ldots, M.$$

Dabei ist $h : \mathbb{R}^+ \to \mathbb{R}$ eine (beliebig oft differenzierbare) Funktion. Eine Lösung des Interpolationsproblems hat die Form

$$G(x) = \sum_{\mu=1}^{M} c_\mu h(\|x - x^\mu\|^2) \quad \text{mit } c \in \mathbb{R}^M$$

Die Interpolationsbedingungen $G(x^\nu) = t^\nu$, $\forall \nu$ geben das lineare Gleichungssystem:

$$\sum_{\mu=1}^{M} c_\mu H_\mu(x^\nu) = \sum_{\mu=1}^{M} c_\mu h(\|x^\nu - x^\mu\|^2) = t^\nu$$

Matrixnotation mit $H_{\nu\mu} := H_\mu(x^\nu)$, $H := (H_{\mu\nu})$, $c = (c_1, \ldots, c_M)$ und $t = (t^1, \ldots, t^M)$:

$$Hc = t$$

Falls H invertierbar ist, so ist $c = H^{-1}t$

Die Funktion $h : \mathbb{R}^+ \to \mathbb{R}$ heißt *radiale Basisfunktion*, wenn die Matrix H invertierbar ist, d.h. wenn das Interpolationsproblem eindeutig lösbar ist. Notwendig: x^μ, $\mu = 1, \ldots, M$ paarweise verschiedene Punkte.

Die symmetrische Matrix H ist invertierbar, wenn sie *positiv definit* ist, d.h. wenn für alle $c \in \mathbb{R}^M$ gilt:

$$\sum_{i=1}^{M} \sum_{j=1}^{M} c_i H_{ij} c_j > 0$$

Theorem: Ist $h(x^2)$ eine positiv definite Funktion, so ist $h(x)$ eine radiale Basisfunktion.

Eine Funktion $h : \mathbb{R}^+ \to \mathbb{R}$ heißt *vollständig monoton* auf $(0, \infty)$, wenn h beliebig oft differenzierbar ist und wenn $(-1)^{(l)} h^{(l)}(x) \geq 0$ für alle $x \in (0, \infty)$ und alle $l \geq 0$ gilt.

Theorem (Schoenberg 1938): Eine Funktion $h(x)$ ist vollständig monoton, gdw. $h(x^2)$ positiv definit ist.

Theorem (Micchelli 1986): Ist h' vollständig monoton und h positiv, so ist h eine radiale Basisfunktion.

Beispiele radialer Basisfunktionen

$$
\begin{aligned}
h(r) &= e^{-r/\sigma^2}, \text{ mit } \sigma > 0 \\
h(r) &= (c^2 + r)^{-\alpha}, \text{ mit } c > 0 \text{ und } \alpha > 0 \\
h(r) &= (c^2 + r)^{\beta}, \text{ mit } c > 0 \text{ und } 0 < \beta < 1 \\
h(r) &= r
\end{aligned}
$$

4.3 Nichtlineares Support Vektor Lernen

$P = \{x^\mu \mid T^\mu = 1\}$ und $N = \{x^\mu \mid T^\mu = -1\}$ linear nicht separierbar

Transformieren x^μ mit nichtlinearer Transformation $\phi : \mathbb{R}^d \to \mathcal{H}$, in einen Vektorraum mit Skalarprodukt (genauer ein Hilbertraum), z.B. $\mathcal{H}$ kann endlichdimensional sein, also $\mathcal{H} = \mathbb{R}^N$, aber auch ein unendlichdimensonaler Raum, etwa der Folgenraum $l^2(\mathbb{R})$.

Idee: Zuerst Transformation $z^\mu := \phi(x^\mu)$ nach $\mathbb{R}^N$ durchführen und dann das Support-Vektor-Lernproblem in $\mathbb{R}^N$ lösen. Gesucht ist also $w \in \mathbb{R}^N$ und $w_0 \in \mathbb{R}$ für die Entscheidungsfunktion

$$F(x) = \text{sgn } (\langle w, \phi(x) \rangle + w_0)$$

Dann ergibt sich für $w^* \in \mathbb{R}^N$

$$w = \sum_{\phi(x^\mu) \in SV} \alpha_\mu^* T^\mu \phi(x^\mu)$$

(und $w_0^* \in \mathbb{R}$ wie bereits beschrieben.)

Die Entscheidgungsfunktion hat dann die Gestalt

$$F(x) = \text{sgn}\left(\sum_{\phi(x^\mu)\in SV} \alpha_\mu^* T^\mu \langle \phi(x^\mu), \phi(x)\rangle + w_0^*\right).$$

Abbildungen der Form

$$(x, y) \in \mathbb{R}^d \times \mathbb{R}^d \to (\phi(x), \phi(y)) \in \mathcal{H} \times \mathcal{H} \to \langle \phi(x), \phi(y)\rangle_{\mathcal{H}} \in \mathbb{R}$$

lassen sich u.U. durch sogenannte **Mercer Kernfunktionen** $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ direkt darstellen.

Satz von Mercer: Sei $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ symmetrisch und gelte

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} f(x)k(x, y)f(y)dxdy > 0$$

für alle $f \in L^2$ (quadratische integrierbare Funktionen). Dann gibt es einen Hilbertraum $\mathcal{H}$ und eine Abbildung $\phi : \mathbb{R}^d \to \mathcal{H}$ mit

$$k(x, y) = \langle \phi(x), \phi(y)\rangle \quad \text{für alle } x, y \in \mathbb{R}^d$$

Damit läßt sich die Entscheidungsfunktion darstellen durch

$$F(x) = \text{sgn} \left(\sum_{\mu=1}^{M} \alpha_\mu^* T^\mu k(x^\mu, x) + w_0^* \right).$$

Die Koeffizienten ergeben sich durch Maximierung von

$$W(\alpha) = \sum_{\mu=1}^{M} \alpha_\mu - \frac{1}{2} \sum_{\nu=1}^{M} \sum_{\mu=1}^{M} \alpha_\nu \alpha_\mu T^\nu T^\mu k(x^\nu, x^\mu)$$

die mit $0 \leq \alpha_\mu \leq C/M$ für alle $\mu = 1, \ldots, M$ erreichen

Beispiele für Mercer Funktionen (eine kleine Auswahl)

1.
$$k(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right) \quad \sigma^2 > 0$$

2.
$$k(x,y) = \tanh\left(\langle x,y\rangle + \theta\right) \quad \theta \in \mathbb{R}$$

3.
$$k(x,y) = \left(\langle x,y\rangle + 1\right)^d \quad d \geq 2$$

4.4 Zusammenfassung

Trainingsdaten $\mathcal{M} = \{(x^\mu, T^\mu) \ : \ \mu = 1, \ldots, M\} \subset \mathbb{R}^d \times \{-1, 1\}$

- Lineare Netze (Delta-Lernregel)
  Fehlerfunktion lautet:

$$E(w) = \| T - Xw \|_2^2 \to \min$$

  w kann iterativ bestimmt werden durch (Batch Lernregel)

$$\Delta w = l \sum_\mu (T^\mu - \langle w, x^\mu \rangle) x^\mu \quad (l > 0) \text{ Lernrate}$$

  Die inkrementelle Lernregel lautet

$$\Delta w = l (T^\mu - \langle w, x^\mu \rangle) x^\mu$$

- Lineare Netze - Pseudoinverse
  Analytische Lösung falls es d linear unabhängige Trainingsbeispiele gibt, ist die eindeutige Lösung gegeben durch:

$$w = (X^t X)^{-1} X^t T$$

  Falls es nicht d linear unabhängige Trainingsbeispiele gibt, lässt ebenfalls die eindeutige Lösung angeben durch

$$w = X^+ T$$

  hierbei ist

$$X^+ = \lim_{\alpha \to 0} (X^t X + \alpha^2 I)^{-1} X^t$$

  X^+ ergibt sich aus der kombinierten Fehlerfunktion (Regularisierung) beim Grenzübergang $\alpha \to 0$.

$$E(w) = \| T - Xw \|_2^2 + \alpha^2 \| w \|_2^2 \to \min$$

- Perzeptron-Lernen
  Ziel ist es die Anzahl der Fehlklassifikationen zu minimieren:

$$E(w) = -\sum_{x^\mu \in \mathcal{M}} T^\mu \langle w, x^\mu \rangle \to \min$$

  hier sei $\mathcal{M}$ die Menge der fehlklassifizierten Muster x^μ (w und x^μ als erweiterte Vektoren).
  Inkrementelle Perzeptron-Lernregel:

$$\Delta w = l(T^\mu - \operatorname{sgn}\langle w, x^\mu \rangle) x^\mu$$

- Neuronale Assoziativspeicher
  Spezialfall: Muster binär mit Einschrittlernen.
  Additive Hebbregel:

$$w = X^t T$$

  Auch die binäre Hebbregel ist gebräuchlich

$$w = \min(\mathbf{1}, X^t T)$$

  (komponentenweises Minimum), $\mathbf{1}$ die Matrix/Vektor mit Einsen in allen Komponenten.

- Support Vektor Lernen (linear)
  Ziel: Trainingsdaten durch linearen Klasssifikator richtig klassifizieren und die Trennebene soll maximalen Rand haben.

$$L(w, w_0, \alpha) = \frac{\|w\|_2^2}{2} - \sum_{\mu=1}^{M} \alpha_\mu \left(T^\mu (\langle w, x^\mu \rangle + w_0) - 1 \right)$$

Die Lösung w ist eindeutig und liegt fest durch

$$w = \sum_{x^\mu \in SV} \alpha_\mu T^\mu x^\mu$$

Die Entscheidungsfunktion lautet dann

$$F(x) = \text{sgn} \left(\sum_{x^\mu \in SV} \alpha_\mu T^\mu \langle x^\mu, x \rangle + w_0 \right).$$

- Backpropagation-Lernen in Mehrschichtnetze (MLP und RBF)
  Fehlerfunktion :

$$E(w) = \| T - Y \|_2^2 \rightarrow \min$$

  Keine analytische Lösung gegeben. Iterative Bestimmung der Parameter (synaptische Kopplungsmatrizen) notwendig, z.B. Gradientenverfahren oder ähnliche Optimierungsverfahren (siehe Neuroinformatik I Vorlesung).

- Interpolation in RBF-Netzen
  Ziel: Fehler auf den Trainingsdaten soll gleich Null sein.
  Gesucht ist eine Funktion $\phi : \mathbb{R}^d \to \mathbb{R}^M$ mit
  $\phi(x) = (h_1(x), \ldots, h_M(x))$, wobei $h_\nu : \mathbb{R}^d \to \mathbb{R}$, für alle $\nu = 1, \ldots, M$. mit

$$T^\mu = \langle w, \phi(x) \rangle = \sum_{\nu=1}^{M} w_\nu h_\nu(x^\mu)$$

  für alle $\mu = 1, \ldots, M$. Lösung ist:

$$w = H^{-1} T$$

  mit $H_{\mu,\nu} = h_\nu(x^\mu) = h(\|x^\nu - x^\mu\|)$.

- Support Vektor Lernen (nichtlinear)
  Zielfunktion:

$$L(w, w_0, \alpha) = \frac{\|w\|_2^2}{2} - \sum_{\mu=1}^{M} \alpha_\mu \left(T^\mu (\langle w, \phi(x^\mu) \rangle + w_0) - 1 \right)$$

  $\phi : \mathbb{R}^d \to \mathbb{R}^N$ und $w \in \mathbb{R}^n$ (und $w_0 \in \mathbb{R}$) gesucht.
  Die Entscheidungsfunktion lautet:

$$F(x) = \text{sgn} \left(\sum_{\phi(x^\mu) \in SV} \alpha_\mu T^\mu \langle \phi(x^\mu), \phi(x) \rangle + w_0 \right).$$

  Für Mercer-Kernfunktionen $k(x, y)$:

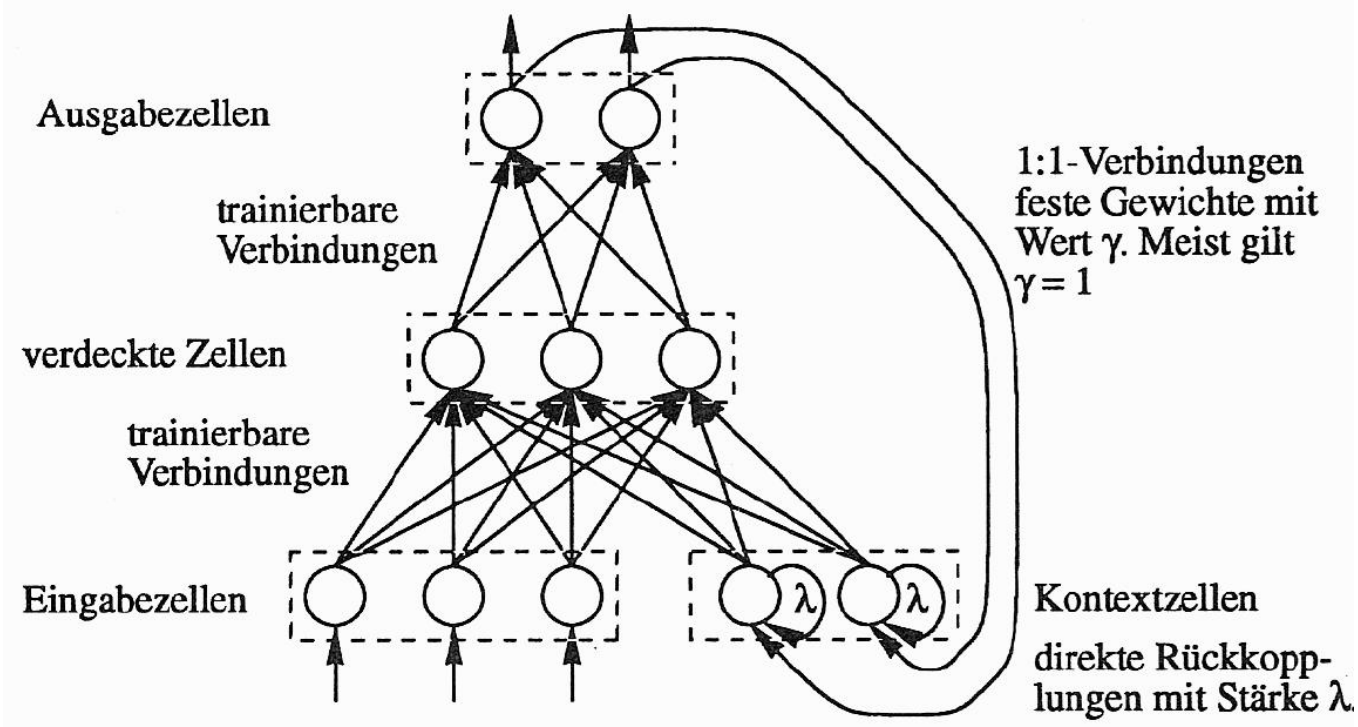
$$F(x) = \text{sgn} \left(\sum_{\phi(x^\mu) \in SV} \alpha_\mu^* T^\mu k(x^\mu, x) + w_0^* \right).$$

5. Rekurente Netze

- Jordan Netze
- Elman Netze
- BPTT Algorithmus
- Echo-State Netze

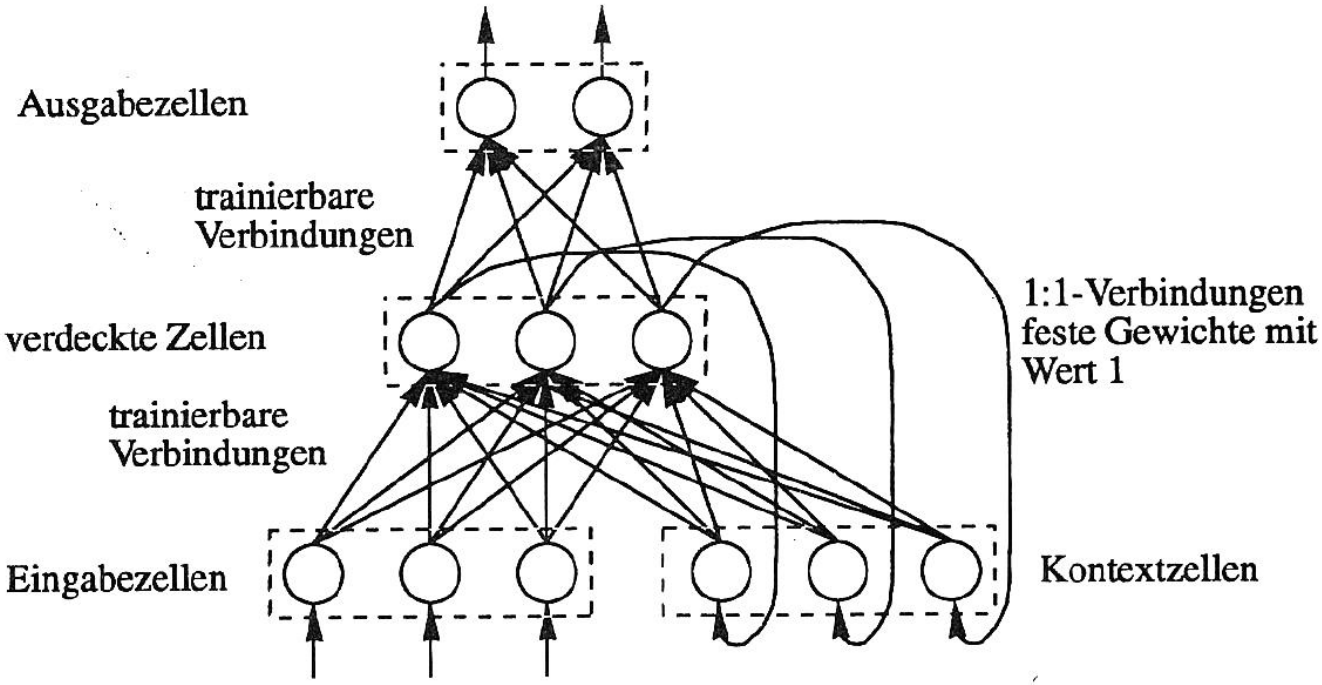
Jordan Netzwerke

Feedback von der Ausgabeschicht auf die Eingabeschicht;
Feedback-Kopplungen werden nicht trainiert (Jordan, 1986)



Elman Netzwerk

Feedback von einem hidden layer (Elman 1990):



Vorteile: interne Repräsentation einer Sequenz ist unabhängig von der Ausgabe $\mathbf{y}$, Zahl der Kontextzellen ist unabhängig von der Ausgabedimension!

Training eines partiell rekurrenten Netzes

Möglichkeit A: Modifikation eines Lernverfahrens für nichtrekurrente Netze, z.B. Error Backpropagation.

- *Algorithmus* (für Elman-Netzwerk):
  Seien w_{ki} und v_{ki} die Gewichte von Eingabeknoten u_k bzw. Kontextknoten s_k zum verdeckten Neuron i und c_{ij} die Gewichte der zweiten Netzwerkschicht
  1) `Setze` $t = t_0$, `initialisiere Kontextzellen` $\mathbf{s}(t_0) = 0$
  2) `Berechne` $\Delta w_{ki}(t)$, $\Delta v_{ki}(t)$ `und` $\Delta c_{ij}(t)$ `gemäß Lernregel für eine Eingabe` $\mathbf{x}(t)$ `mit Sollwert` $\mathbf{T}(t)$ `ohne Beachtung rekurrenter Verbindungen`
  3) `Setze` $t = t + 1$, `aktualisiere die Ausgabe` $\mathbf{s}(t)$ `der Kontextzellen und gehe zu 2)`
- *Eigenschaften:* Fehler von $\mathbf{y}(t) = f(\mathbf{x}(t))$ wird minimiert, keine Klassifikation von Sequenzen möglich.

Möglichkeit B:

Verwendung eines Lernverfahrens für rekurrente Netze
(z.B. BPTT [Rumelhart 86], RTRL [Willliams 89])

- Idee von BPTT (*"Backpropagation Through Time"*):
  Entfaltung des Netzwerks in der *Zeit* !
- Gradientenabstieg zur Minimierung von $E = \sum_{t=t_0}^{t_{\max}} E(t)$

  mit $E(t) =$
  $\begin{cases} ||\mathbf{T}(t) - \mathbf{y}(t)|| & \text{falls } \mathbf{T}(t) \text{ zum Zeitpunkt } t \text{ vorliegt} \\ 0 & \text{sonst} \end{cases}$
- *Eigenschaften:* Fehler von $\mathbf{y}(t_{\max}) = f(\mathbf{x}(t_0), \ldots, \mathbf{x}(t_{\max}))$ wird minimiert, auch Klassifikation von Sequenzen variabler Länge möglich!

BPTT Algorithmus für Elman Netzwerk

Gegeben sei ein $(m+h)-h-n$ Elman Netzwerk

mit:

w_{ki}	:	Gewichte von Eingabeknoten u_k zum verdeckten Neuron i
v_{ki}	:	Gewichte von Kontextknoten s_k zum verdeckten Neuron i
c_{ij}	:	Gewichte vom verdeckten Neuron i zum Ausgabeneuron j
$\delta_j^{(y)}$	:	Fehler am Ausgabeneuron j
$\delta_i^{(s)}$	:	Fehler am verdeckten Neuron i

Lineare Ausgabeneuronen $j = 1, \ldots, n$:

Annahme: $E(t) = 0$ für $t \neq t_{\max}$

für $t = t_{max}$ gilt:
$$\delta_j^{(y)}(t) = T_j(t) - y_j(t)$$
$$\Delta c_{ij} = s_i(t+1)\,\delta_j^{(y)}(t)$$

für $t < t_{max}$ gilt:
$$\delta_j^{(y)}(t) = 0$$
$$\Delta c_{ij} = 0$$

Sigmoide verdeckte Neuronen $i = 1, \ldots, h$:

für $t = t_{max}$

$$\delta_i^{(s)}(t) = \sum_{j=1}^{n} c_{ij}\, \delta_j^{(y)}(t) \cdot s_i'(t+1)$$

$$\Delta v_{ki}(t) = s_k(t)\, \delta_i^{(s)}(t)$$

$$\Delta w_{ki}(t) = x_k(t)\, \delta_i^{(s)}(t)$$

für $t_0 \leq t < t_{max}$

$$\delta_i^{(s)}(t) = \sum_{k=1}^{h} v_{ki}\, \delta_k^{(s)}(t+1) \cdot s_i'(t+1)$$

$$\Delta v_{ki}(t) = s_k(t)\, \delta_i^{(s)}(t)$$

$$\Delta w_{ki}(t) = x_k(t)\, \delta_i^{(s)}(t)$$

Resultierende Lernregeln:

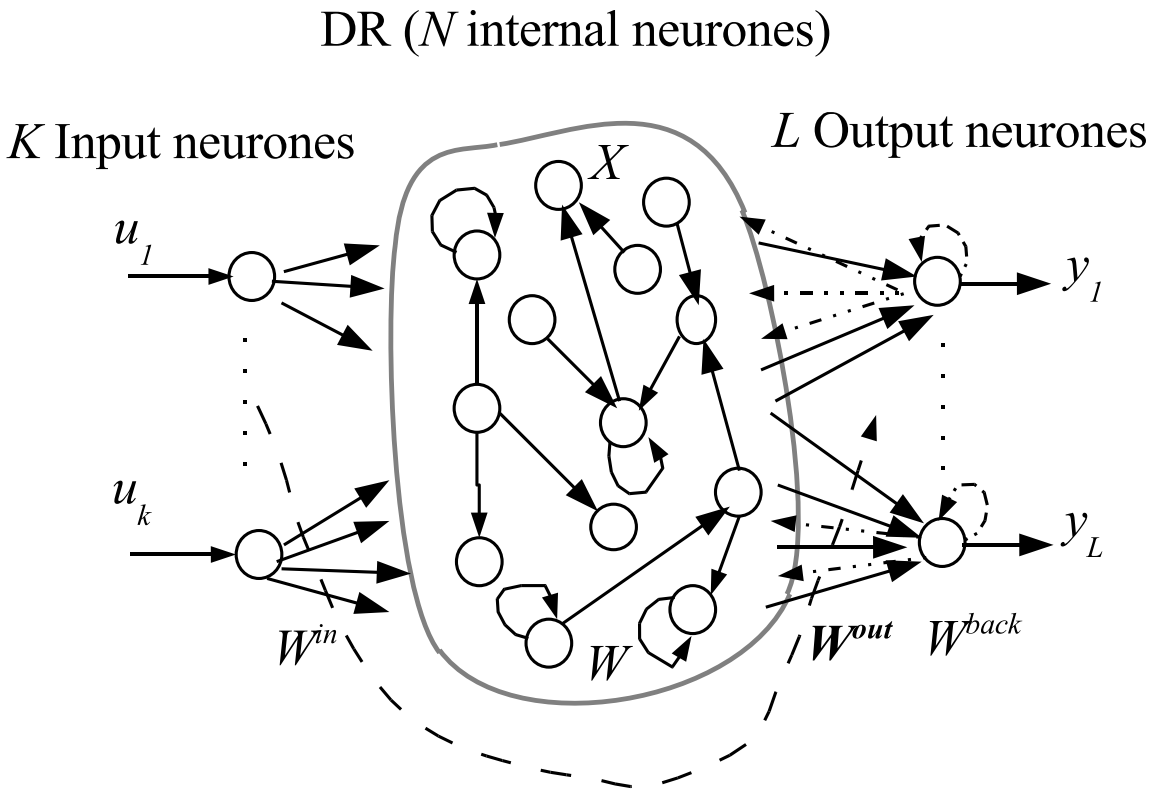
$$c_{ij} = c_{ij} + \eta_1 \Delta c_{ij}$$

$$w_{ki} = w_{ki} + \eta_2 \sum_{t=t_0}^{t_{max}} \Delta w_{ki}(t)$$

$$v_{ki} = v_{ki} + \eta_2 \sum_{t=t_0}^{t_{max}} \Delta v_{ki}(t)$$

Echo-State Netzwerke

ESN wurden von Herbert Jäger entwickelt (Jäger, 2004).



Für Zeitschritte $n = 1, 2, \ldots$

$U(n) = (u_1(n), \ldots, u_K(n))$ Eingabe zur Zeit n

$X(n) = (x_1(n), \ldots, x_N(n))$ Aktivität der Poolneuronen zur Zeit n

$Y(n) = (y_1(n), \ldots, y_L(n))$ Ausgabe zur Zeit n

Kopplungsmatrizen:
$N \times K$ Eingabematrix $W^{in} = (w_{ij}^{in})$
$N \times N$ Kopplungsmatrix der Poolneuronen $W = (w_{ij})$
$L \times (K + N + L)$ Ausgabematrix $W^{out} = (w_{ij}^{out})$
$N \times L$ Feedback-Matrix $W^{back} = (w_{ij}^{back})$ von der Ausgabe zu den Poolneuronen.
Berechnung der Aktivierung der Poolneuronen:

$$X(n+1) = f(W^{in}U(n+1) + WX(n) + W^{back}Y(n)) \qquad (12)$$

mit $f = (f_1, \ldots, f_N)$
Ausgabe des ESN:

$$Y(n+1) = f^{out}(W^{out}(U(n+1), X(n+1), Y(n))) \qquad (13)$$

$f^{out} = (f_1^{out}, \ldots, f_L^{out})$ sigmoide Funktionen der Ausgabeneuronen.
$(U(n+1), X(n+1), Y(n))$
nur W^{out} wird trainiert.

Training von Echo-State Netzwerken

1. Gegeben Trainingssequenz $(U(n), D(n))$
2. Bilde Zufallsmatrizen (W^{in}, W, W^{back}).
3. Skaliere W , dass alle Eigenwerte $|\lambda_{max}| \leq 1$.
4. Netzwerk laufen lassen

$$X(n+1) = f(W^{in}U(n+1) + WX(n) + W^{back}D(n)) \quad (14)$$

5. Für jeden Zeitschritt n sammle als Eingaben $X(n)$ in einer Matrix M und $tanh^{-1}D(n)$ als Matrix der Lehrersignale T.
6. Berechne die Pseudo-Inverse von M und setze

$$W^{out} = (M^{+}T)^{t} \quad (15)$$

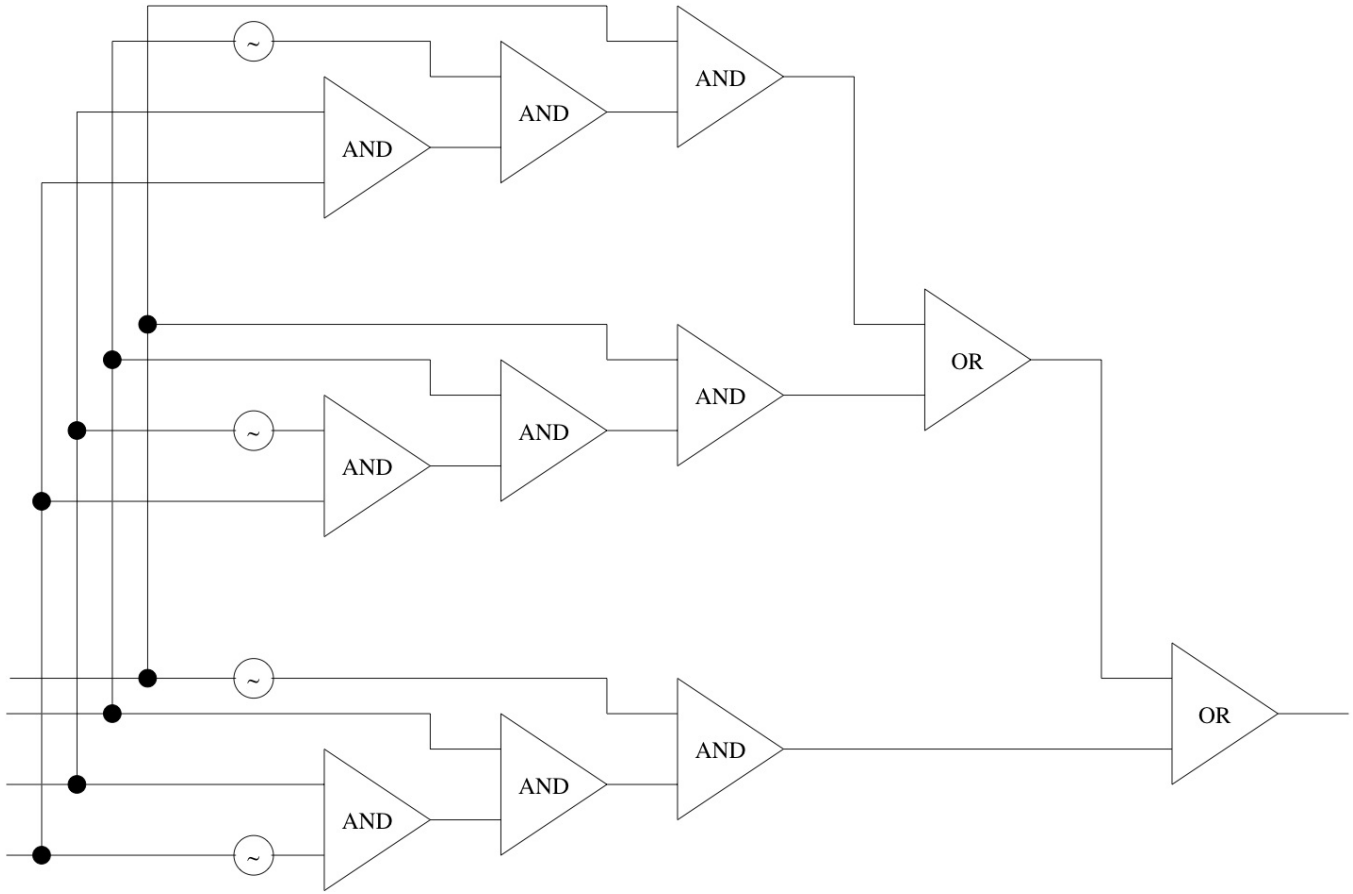
t sei die transponierte Matrix.

6. Komplexität der Netze und des Lernens

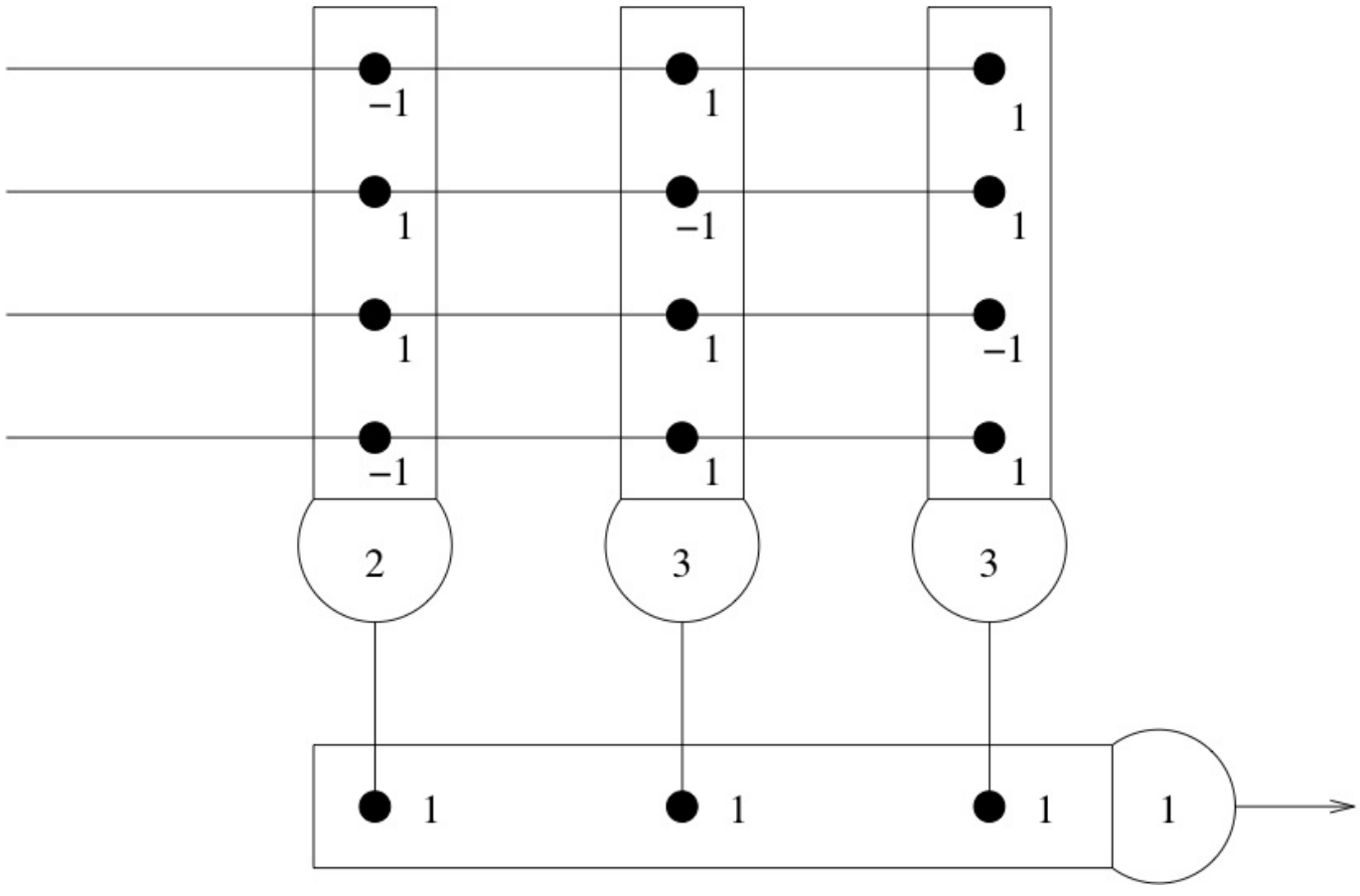
1. Neuronale Schaltungen
2. NP-Vollständigkeit des Lernens

6.1 Neuronale Schaltungen

Boolesche Funktion mit AND, OR und NOT:



Neuronale disjunktive Normalform



Exkurs: McCulloch-Pitts Neuron

- Struktur eines Neurons (McCulloch and Pitts, 1943):
- $\mathbf{x} \in \{0,1\}^m$, $\mathbf{w} \in \{-1,1\}^m$, $\theta \in \mathbb{Z}$

$$u = \sum_{i=1}^{m} x_i w_i = \mathbf{x} \cdot \mathbf{w} = \langle \mathbf{x}, \mathbf{w} \rangle$$

$$y = \begin{cases} 1 & \text{für} \quad u \geq \theta \\ 0 & \text{für} \quad \text{sonst} \end{cases}$$

- **Satz:** Jede beliebige logische Funktion ist mit Netzen aus McCulloch-Pitts Neuronen realisierbar.
  Beweis: mittes isjunktiver Normalform

Aussagen über Schaltnetze

Schaltungen von polylogarithmischer Tiefe für Boole'sche Funktionen $\{0,1\}^n \to \{0,1\}$.

Schaltungsklassen

- $\mathcal{AC}$ (**A**lternating **C**ircuits) besteht aus UND-, ODER- und NICHT-Gattern (von beliebiger Stelligkeit).
- $\mathcal{RC}$ (**R**eal threshold **C**ircuits) besteht aus Schwellenneuronen mit beliebigen reellen Gewichten.
- $\mathcal{TC}$ (**T**hreshold **C**ircuits) besteht aus Schwellenneuronen mit polynomiellen (ganzzahligen) Gewichten.
- $\mathcal{UC}$ (**U**nitary threshold **C**ircuits) besteht aus Schwellenneuronen mit Gewichten aus $\{-1,1\}$.

Unterklassen

Für Klasse X bezeichnet X^k Schaltungen der Tiefe $O(log^k(n))$ und X_k Schaltungen der Tiefe $\leq k$.
Beobachtungen (1): Für $X = \mathcal{AC}, \mathcal{RC}, \mathcal{TC}, \mathcal{UC}$ gilt:

$$\bigcup_{k \geq 1} X_k = X^0 \quad \bigcup_{k \geq 1} X^k = X$$

(2): $\mathcal{AC}_k \subseteq \mathcal{UC}_k \subseteq \mathcal{TC}_k \subseteq \mathcal{RC}_k$
Spezielle Prädikate:

$$\text{PAR}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \bmod 2$$

$$\text{SPR}(x_1, \ldots, x_{2n}) = \sum_{i=1}^{n} x_i x_{i+n} \bmod 2$$

Vergleich von Neuronen mit Logik

Theorem: Für alle k gilt:

$$\mathcal{AC}^k \subseteq \mathcal{UC}^k = \mathcal{TC}^k = \mathcal{RC}^k \subseteq \mathcal{AC}^{k+1}$$

Theorem: PAR $\notin \mathcal{AC}^0$ PAR $\in \mathcal{TC}^0 \subseteq \mathcal{AC}^1$.

Vergleich verschiedener Neuronen bei endlicher Tiefe:
Theorem: Für alle k gilt:

$$\mathcal{RC}_k \subseteq \mathcal{TC}_{k+1} \subseteq \mathcal{UC}_{k+2}$$

Beobachtung: $\Longrightarrow \mathcal{UC}^k = \mathcal{TC}^k = \mathcal{RC}^k$ für alle k

Theorem: PAR $\in \mathcal{TC}_2 \setminus \mathcal{TC}_1$ und SPR $\in \mathcal{TC}_3 \setminus \mathcal{TC}_2$. (Schwierig zu zeigen ist, daß SPR $\notin \mathcal{TC}_2$.)

Bisher ist keine Funktion aus $\mathcal{TC}^1$ bekannt, die nicht in $\mathcal{TC}_3$ ist.

6.2 NP-Vollständigkeit des Lernens

Three-Unit-Training Problem: Es sei $G : \{0,1\}^n \to \{0,1\}$ eine Boole'sche Funktion, gegeben durch Paare (x_i, y_i), $i = 1, \ldots, M$ mit $x_i \in \{0,1\}^n$ und $y_i \in \{0,1\}$.
Frage: Gibt es ein neuronales 2-schichtiges Netz mit 3 Neuronen (2 in der versteckten Schicht, 1 Ausgabeneuron) mit $F(x_i) = y_i$ alle $i = 1, \ldots, M$.
$F : \{0,1\}^n \to \{0,1\}$ sei hierbei die Funktion, die das neuronale Netz realisiert.
F ist durch $2n + 2$ Gewichte und 3 Schwellwerte definiert

Entscheidungsprobleme:

- Entscheidungsproblem D heißt in Polynomzeit entscheidbar, kurz $D \in P$, wenn es eine Turingmaschine M und ein Polynom p gibt, so dass für jedes $I \in D$ die Instanz I in höchstens $p(|I|)$ Schritten entscheidbar ist.
- Entscheidungsproblem D heißt in verifizierbarbar, kurz $D \in NP$, wenn es eine deterministische Turingmaschine M und ein Polynom p gibt, so dass für jedes $I \in D$ eine Lösung für die Instanz I in höchstens $p(|I|)$ Schritten verifiziert ist.

Beispiele

- **Mengensplitting:** Es sei $\{s_1, \ldots, s_n\}$ eine Menge und $C = \{c_1, \ldots, c_k\}$ mit $x_j \subset S$.
  Frage: Gibt es A, B disjunkte, nichtleereTeilmengen von S mit $A \cup B = S$ und $c_j \not\subset A$ und $c_j \not\subset B$ für alle $j = 1, \ldots, k$?
- **Bilineare Beschränkung:** Es sei $\{(x_i, y_i)\}$, $i = 1, \ldots, M$ mit $x_i \in \{0, 1\}^n$ und $y_i \in \{0, 1\}$ eine Menge von Paaren.
  Frage: Gibt es Halbräume Z_1 und Z_2 in $\mathbb{R}^n$, so dass der Durchschnitt $Z_1 \cap Z_2$ genau die positiven Beispiele (mit $y_i = 1$) enthält ?

D heisst NP-vollständig wenn gilt:

1. $D \in NP$
2. Jedes Problem $D' \in NP$ ist polynomiell reduzierbar auf D.

$D' \in NP$ heißt polynomiell reduzierbar auf D, gdw es eine Abbildung $f : D \to D'$ gibt (f mittels einer deterministischen Turingmaschine in Polynomzeit berechenbar), so dass für jede Instanz $I \in D'$ gilt:
I hat Antwort *ja/nein* bzgl D' gdw $f(i)$ Antwort *ja/nein* bzgl D hat

Satz: **Three-Unit-Training** (TUT) ist NP-vollständig.
Beweis:

1. $TUT \in NP$
2. *Bilineare Beschränkung* (BE) ist polynomiell reduzierbar auf TUT
3. *Mengensplitting* ist polynomiell reduzierbar auf BE

Beweis:

1. $TUT \in NP$
2. *Bilineare Beschränkung* (BE) ist polynomiell reduzierbar auf TUT
3. *Mengensplitting* ist polynomiell reduzierbar auf BE

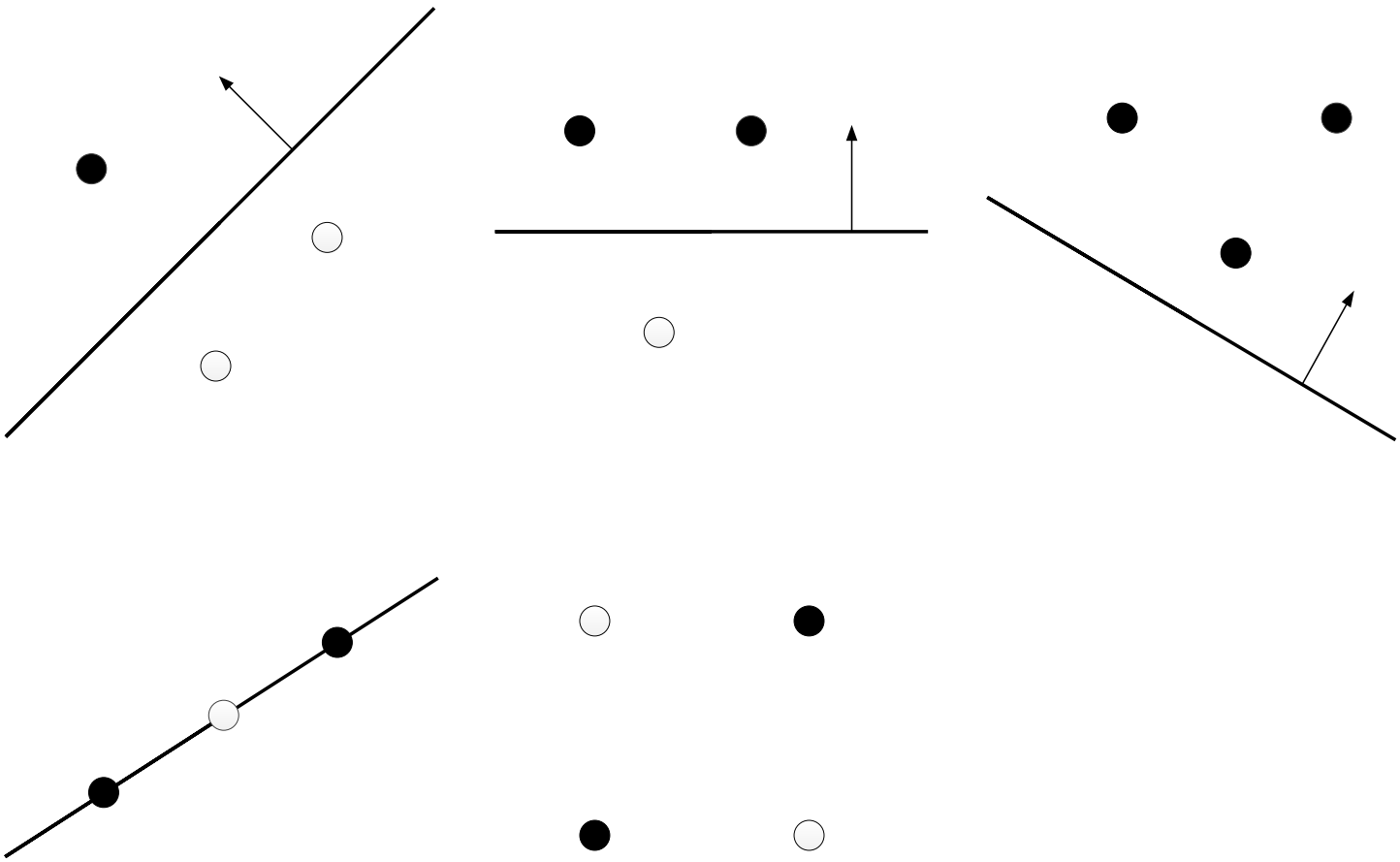
7. Darstellung mit neuronalen Netzen

1. Satz von Cover
2. Satz von Cybenko/Hornik

7.1 Satz von Cover (1964)

- P und N heißen linear trennbare Mengen, gdw es $w \in \mathbb{R}^n$ und $\alpha \in \mathbb{R}$ gibt mit $\langle w, x \rangle > \alpha$ falls $x \in P$ und $\langle w, x \rangle < \alpha$ falls $x \in N$
- Hyperebene ist dann die Menge der x mit $\langle w, x \rangle = \alpha$.
- P und N heißen 0-trennbare Mengen, falls sie linear trennbar sind mit $\alpha = 0$.
- M Punkte im $\mathbb{R}^n$ sind in allgemeiner Lage, falls jeweils k Punkte aus M linear unabhängig sind für $k = 2, 3, \ldots, n$. ($x_1, \ldots, x_k$ heißen linear unabhängig, gdw aus $\sum_{i=1}^{k} \alpha_i x_i = 0$ stets $\alpha_1 = \cdots \alpha_k = 0$ folgt).

Separierbarkeit / Allgemeine Lage : $M = 3$, $M = 4$ Punkte im $\mathbb{R}^2$



...

Betrachten M paarweise verschiedene Punkte im $\mathbb{R}^n$ *in allgemeiner Lage*.
$C(M,n)$ Zahl der 0-1-Belegungen die *linear trennbar* sind
$C_0(M,n)$ Zahl der 0-1-Belegungen die 0-*trennbar* sind.
Dann ist

$$C_0(M,n) = 2\sum_{k=0}^{n-1} \binom{M-1}{k}$$

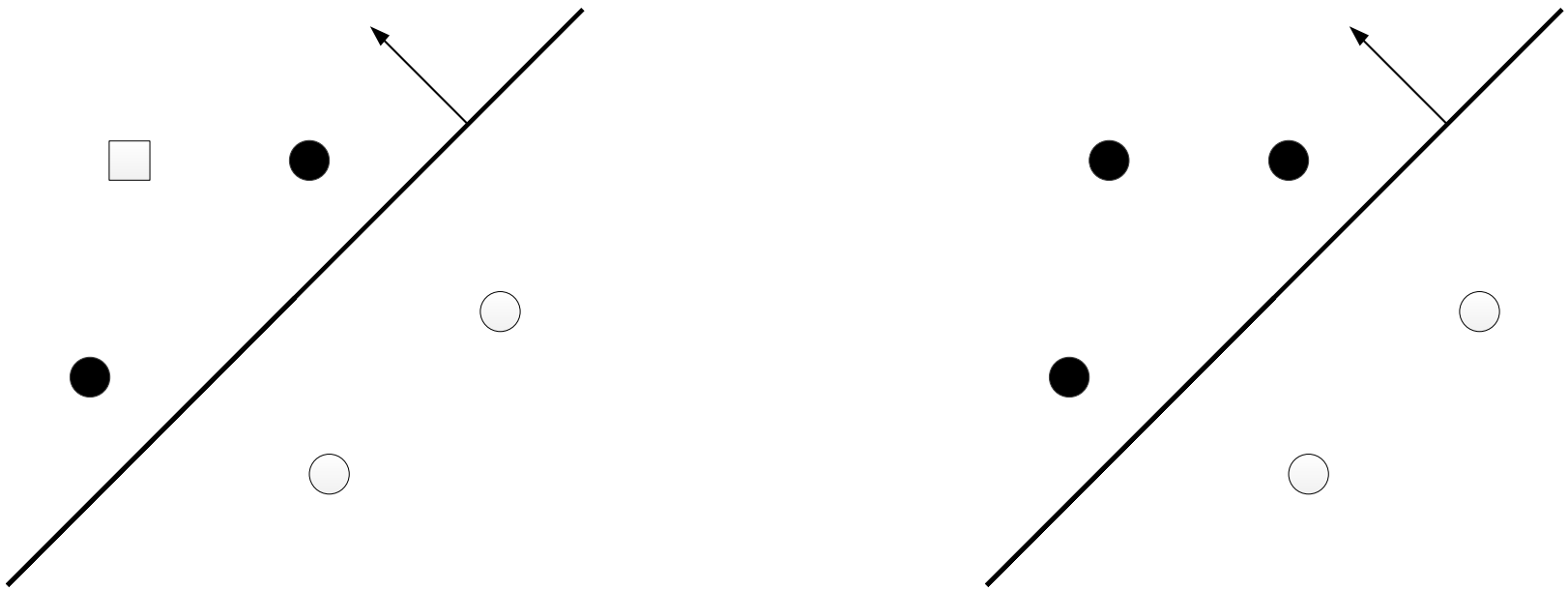
und

$$C(M,n) = 2\sum_{k=0}^{n} \binom{M-1}{k}.$$

Hierbei ist der Binomialkoeffizient $\binom{n}{k} = \begin{cases} \frac{n!}{(n-k)!k!} & \text{für } 0 \leq k \leq n \\ 0 & \text{für } k > n\,. \end{cases}$

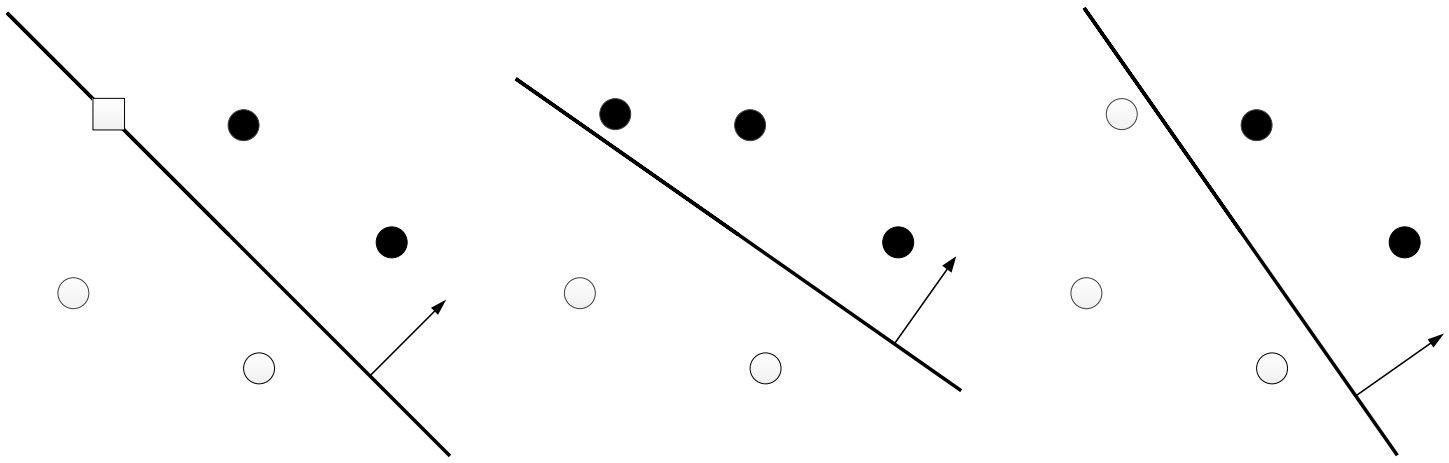
Rekursionsgleichung

1. $C(M+1,n) = C(M,n) + C_0(M,n)$
2. $C_0(M+1,n) = C_0(M,n) + C_0(M,n-1)$



Links: $M = 4$ Punkte (linear trennbar) und ein neuer Punkt (Quadrat) kommt hinzu
Rechts: Belegung der 4 Punkte bzw die Trennebene ist so, dass die Belegung des neuen Punktes festgelegt ist

...



Belegung der 4 Punkte bzw Trennebene ist so, dass Belegung des Punktes nicht festgelegt ist; (OE neuer Punkt = Nullpunkt)

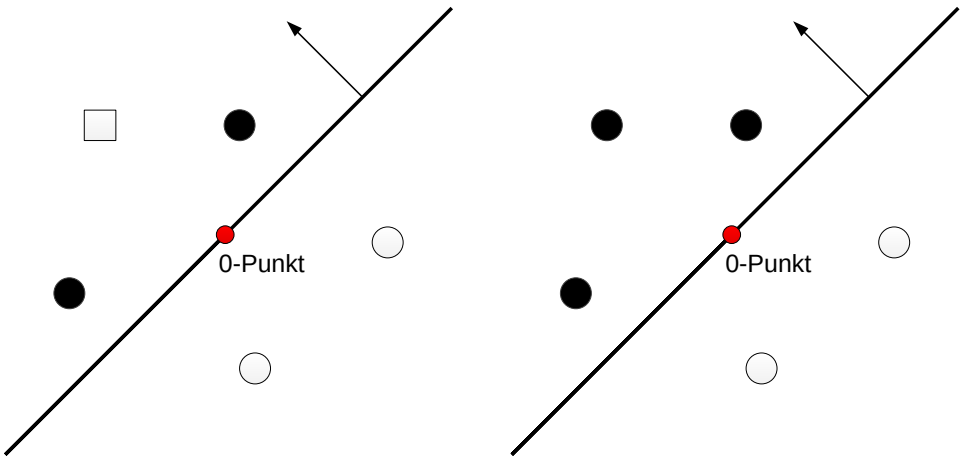
$$C(M+1,n) = \#\{\text{Trennebene legt neuen Punkt fest}\} + 2\#\{\text{Trennebene legt neuen Punkt } \textbf{nicht} \text{ fest}\}$$

- $\#\{\text{Trennebene legt neuen Punkt } \textbf{nicht} \text{ fest}\}$, d.h. es sind 2 Belegungen möglich, d.h. neuer Punkt = Nullpunkt. Dies sind $C_0(M,n)$ Belegungen.
- $\#\{\text{Trennebene legt neuen Punkt fest}\}$, dass sind dann $C(M,n) - C_0(M,n)$

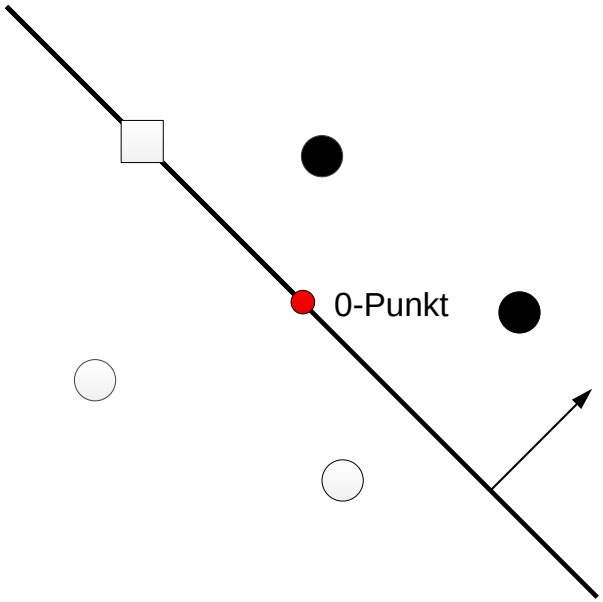
Damit gilt:

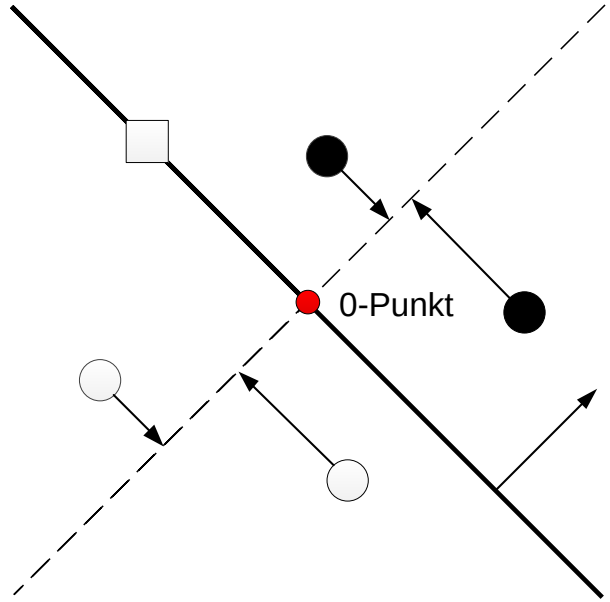
$$\begin{aligned} C(M+1,n) &= C(M,n) - C_0(M,n) + 2C_0(M,n) \\ &= C(M,n) + C_0(M,n)\,. \end{aligned}$$

Wir betrachten nun die Rekursionsgleichung für $C_0(M,n)$:



Links: $M = 4$ Punkte (0 trennbar) und ein neuer Punkt (Quadrat) kommt hinzu
Rechts: Belegung der 4 Punkte bzw die Trennebene ist so, dass die Belegung des neuen Punktes festgelegt ist





Links: Belegung der 4 Punkte bzw Trennebene ist so, dass Belegung des Punktes nicht festgelegt ist (OE Trennebene geht durch den neuen Punkt und den 0-Punkt
Rechts: Projektion auf den Orthogonalraum, der von 0 und dem neuen Punkt definiert wird. Projektionen sind wegen der allgemeinen Lage der Punkt, wieder in allgemeiner Lage und linear trennbar.

Es gilt für $M = 1$: $C(1, n) = C_0(1, n) = 2$ für alle $n \in \mathbb{N}$
Außerdem gilt $C_0(M, 1) = 2$
$M \Rightarrow M + 1$:

$$\begin{aligned} C_0(M+1, n) &= C_0(M, n) + C_0(M, n-1) \\ &= 2\left(\sum_{k=0}^{n-1} \binom{M-1}{k} + \sum_{k=0}^{n-2} \binom{M-1}{k}\right) \\ &= 2\left(1 + \sum_{k=1}^{n-1} \binom{M-1}{k} + \binom{M-1}{k-1}\right) \\ &= 2\left(1 + \sum_{k=1}^{n-1} \binom{M}{k}\right) = 2\sum_{k=0}^{n-1} \binom{M}{k} \end{aligned}$$

Damit ist die Formel für $C_0(M, n)$ bewiesen.

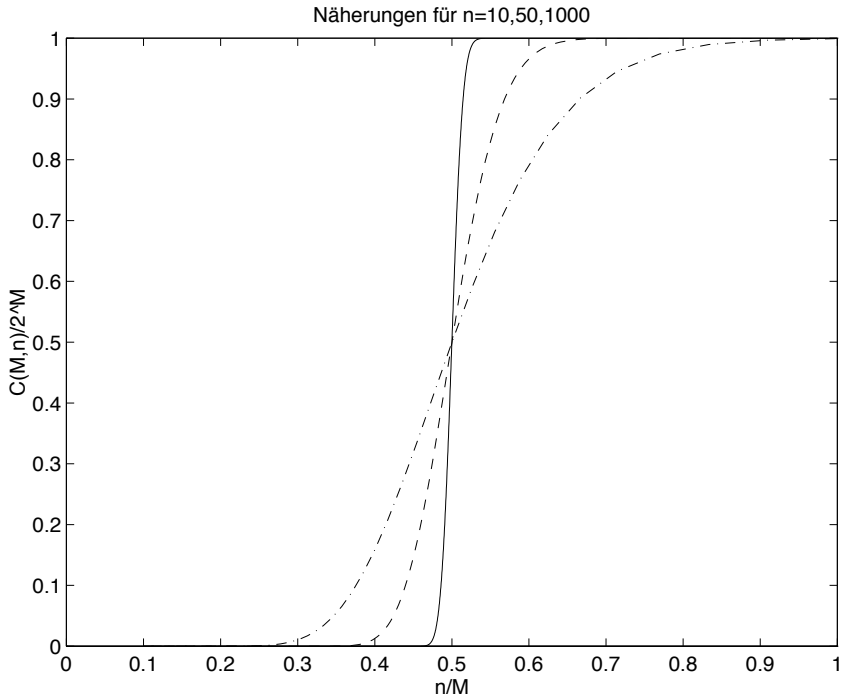
Es gilt ferner $C(M,n) = C_0(M,n+1)$ durch Induktion nach M:
$M = 1$: $C(1,n) = C_0(1,n+1) = 2 \; \forall \; n$
$M \Rightarrow M+1$:

$$\begin{aligned} C(M+1,n) &= C(M,n) + C_0(M,n) \\ &= C_0(M,n+1) + C_0(M,n) \\ &= C_0(M+1,n+1) \end{aligned}$$

Damit ist die Formel $C(M,n)$ bewiesen.

Häufigkeit linear separierbarer Dichotomien

$X = \{x_1, \ldots, x_M\} \subset \mathbb{R}^n$ mit M Punkten in *allgemeiner Lage*
$C(M, d)$ die Anzahl der linear separierbaren Dichotomien von X
(insgesamt gibt es 2^M Dichotomien).



Für große n steiler Verlauf der Binomial-Verteilung.
Falls $M \leq n + 1$, dann immer linear trennbar.
Falls $M \leq 2n$ dann mit hoher Wahrscheinlichkeit trennbar.