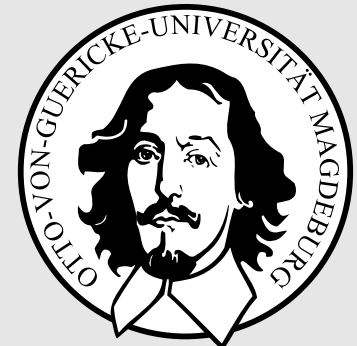


# Neuronale Netze

Prof. Dr. Rudolf Kruse

Computational Intelligence  
Institut für Wissens- und Sprachverarbeitung  
Fakultät für Informatik  
[kruse@iws.cs.uni-magdeburg.de](mailto:kruse@iws.cs.uni-magdeburg.de)



# Hopfield-Netze

# Hopfield-Netze

Ein **Hopfield-Netz** ist ein neuronales Netz mit einem Graphen  $G = (U, C)$ , das die folgenden Bedingungen erfüllt:

(i)  $U_{\text{hidden}} = \emptyset, U_{\text{in}} = U_{\text{out}} = U,$

(ii)  $C = U \times U - \{(u, u) \mid u \in U\}.$

- In einem Hopfield-Netz sind alle Neuronen sowohl Eingabe- als auch Ausgabeneuronen.
- Es gibt keine versteckten Neuronen.
- Jedes Neuron erhält seine Eingaben von allen anderen Neuronen.
- Ein Neuron ist nicht mit sich selbst verbunden.

Die Verbindungsgewichte sind symmetrisch, d.h.

$$\forall u, v \in U, u \neq v : \quad w_{uv} = w_{vu}.$$

# Hopfield-Netze

Die Netzeingabefunktion jedes Neurons ist die gewichtete Summe der Ausgaben aller anderen Neuronen, d.h.

$$\forall u \in U : f_{\text{net}}^{(u)}(\vec{w}_u, \vec{\text{in}}_u) = \vec{w}_u \vec{\text{in}}_u = \sum_{v \in U - \{u\}} w_{uv} \text{out}_v .$$

Die Aktivierungsfunktion jedes Neurons ist eine Sprungfunktion, d.h.

$$\forall u \in U : f_{\text{act}}^{(u)}(\text{net}_u, \theta_u) = \begin{cases} 1, & \text{falls } \text{net}_u \geq \theta_u, \\ -1, & \text{sonst.} \end{cases}$$

Die Ausgabefunktion jedes Neurons ist die Identität, d.h.

$$\forall u \in U : f_{\text{out}}^{(u)}(\text{act}_u) = \text{act}_u .$$

## Alternative Aktivierungsfunktion

$$\forall u \in U : f_{\text{act}}^{(u)}(\text{net}_u, \theta_u, \text{act}_u) = \begin{cases} 1, & \text{falls } \text{net}_u > \theta, \\ -1, & \text{falls } \text{net}_u < \theta, \\ \text{act}_u, & \text{falls } \text{net}_u = \theta. \end{cases}$$

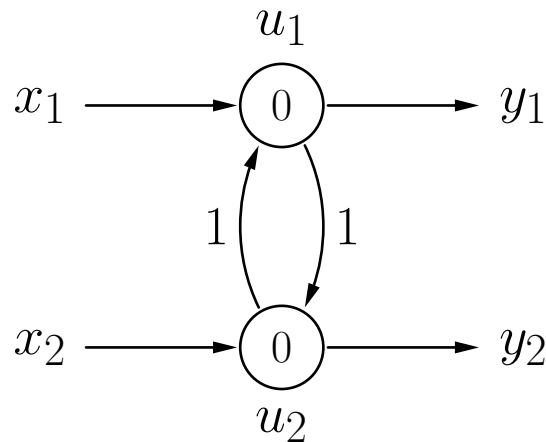
Diese Aktivierungsfunktion bietet einige Vorteile bei der späteren physikalischen Interpretation eines Hopfield-Netzes. Diese wird allerdings in der Vorlesung nicht weiter genutzt.

## Allgemeine Gewichtsmatrix eines Hopfield-Netzes

$$\mathbf{W} = \begin{pmatrix} 0 & w_{u_1 u_2} & \dots & w_{u_1 u_n} \\ w_{u_1 u_2} & 0 & \dots & w_{u_2 u_n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{u_1 u_n} & w_{u_1 u_n} & \dots & 0 \end{pmatrix}$$

# Hopfield-Netze: Beispiele

## Sehr einfaches Hopfield-Netz



$$\mathbf{W} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Das Verhalten eines Hopfield-Netzes kann von der Update-Reihenfolge abhängen.

- Die Berechnungen können oszillieren, wenn Neuronen synchron aktualisiert werden.
- Die Berechnung konvergiert immer, wenn die Neuronen asynchron in fester Reihenfolge aktualisiert werden.

## Parallele Aktualisierung der Neuronenaktivierungen

	$u_1$	$u_2$
Eingabephase	-1	1
Arbeitsphase	1	-1
	-1	1
	1	-1
	-1	1
	1	-1
	-1	1

- Die Berechnungen oszillieren, kein stabiler Zustand wird erreicht.
- Die Ausgabe hängt davon ab, wann die Berechnung abgebrochen wird.

# Hopfield-Netze: Beispiele

## Sequentielle Aktualisierung der Neuronenaktivierungen

	$u_1$	$u_2$
Eingabephase	<b>-1</b>	<b>1</b>
Arbeitsphase	<b>1</b>	1
	1	<b>1</b>
	<b>1</b>	1
	1	<b>1</b>

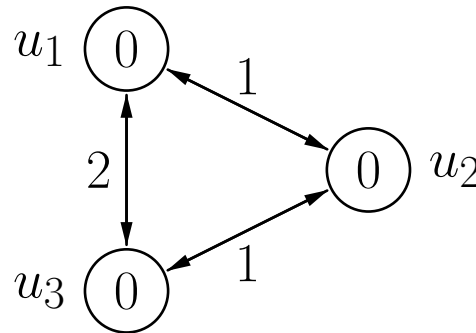
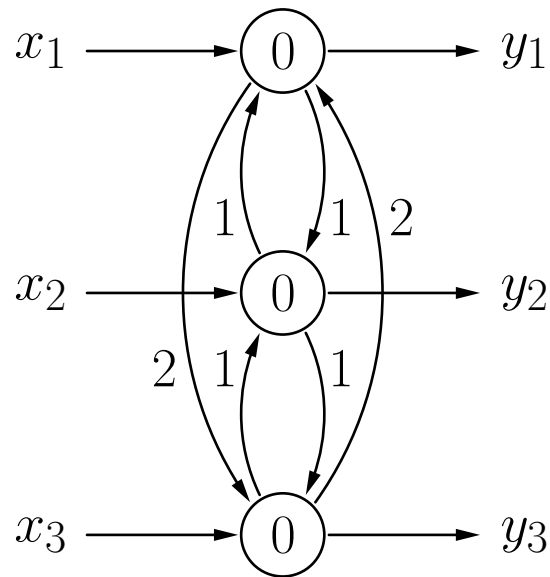
	$u_1$	$u_2$
Eingabephase	<b>-1</b>	<b>1</b>
Arbeitsphase	-1	<b>-1</b>
	<b>-1</b>	-1
	-1	<b>-1</b>
	<b>-1</b>	-1

- Aktivierungsreihenfolge  $u_1, u_2, u_1, \dots$  bzw.  $u_2, u_1, u_1, \dots$
- Unabhängig von der Reihenfolge wird ein stabiler Zustand erreicht.
- Welcher Zustand stabil ist, hängt von der Reihenfolge ab.



# Hopfield-Netze: Beispiele

## Vereinfachte Darstellung eines Hopfield-Netzes

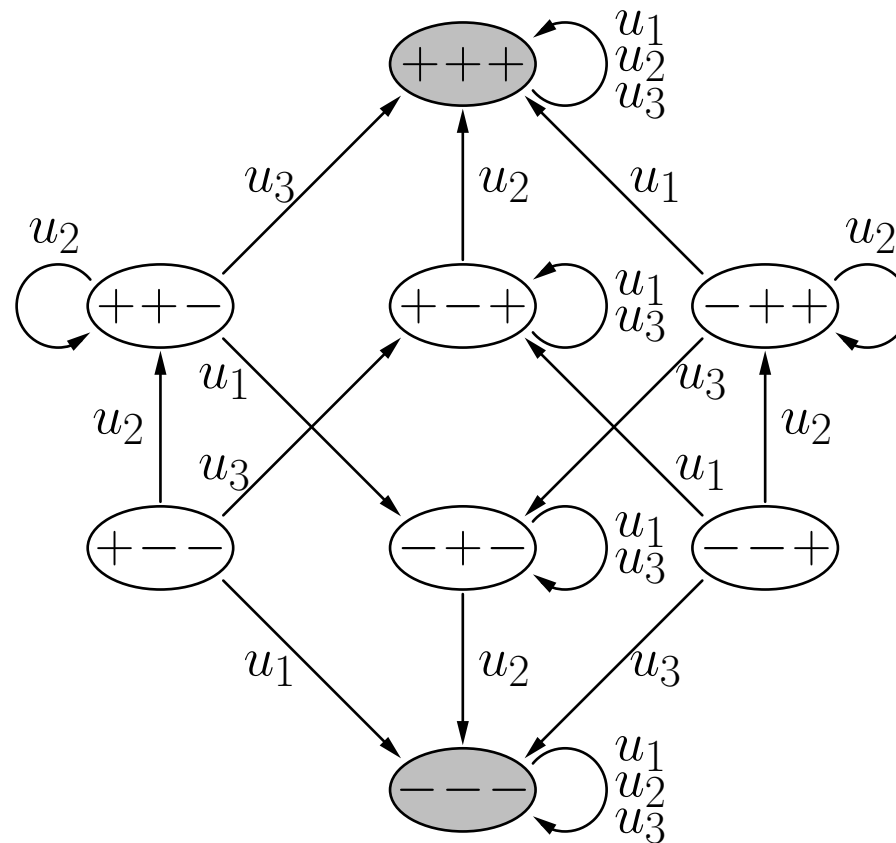


$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$

- Symmetrische Verbindungen zwischen Neuronen werden kombiniert.
- Eingaben und Ausgaben werden nicht explizit dargestellt.

# Hopfield-Netze: Zustandsgraph

## Graph der Aktivierungen und Übergänge



(Zustandsgraph zum Netz auf der vorherigen Folie, Erläuterung nächste Folie)

## Graph der Aktivierungen und Übergänge

- +/-: Aktivierung der Neuronen (+ entspricht +1, - entspricht -1)
- Pfeile: geben die Neuronen an, deren Aktualisierung zu dem jeweiligen Zustandsübergang führt
- grau unterlegte Zustände: stabile Zustände
- beliebige Aktualisierungsreihenfolgen ablesbar

# Hopfield-Netze: Konvergenz der Berechnungen

**Konvergenztheorem:** Wenn die Aktivierungen der Neuronen eines Hopfield-Netzes asynchron (sequentiell) durchgeführt werden, wird ein stabiler Zustand nach einer endlichen Anzahl von Schritten erreicht.

Wenn die Neuronen zyklisch in einer beliebigen, aber festen Reihenfolge durchlaufen werden, sind höchstens  $n \cdot 2^n$  Schritte (Aktualisierungen einzelner Neuronen) notwendig, wobei  $n$  die Anzahl der Neuronen im Netz ist.

Der Beweis erfolgt mit Hilfe einer **Energiefunktion**.

Die Energiefunktion eines Hopfield-Netzes mit  $n$  Neuronen  $u_1, \dots, u_n$  ist

$$\begin{aligned} E &= -\frac{1}{2} \vec{\text{act}}^\top \mathbf{W} \vec{\text{act}} + \vec{\theta}^T \vec{\text{act}} \\ &= -\frac{1}{2} \sum_{u,v \in U, u \neq v} w_{uv} \text{act}_u \text{act}_v + \sum_{u \in U} \theta_u \text{act}_u. \end{aligned}$$

# Hopfield-Netze: Konvergenz

Man betrachte die Energieänderung die aus einer aktivierungsändernden Aktualisierung entsteht:

$$\begin{aligned}\Delta E = E^{(\text{new})} - E^{(\text{old})} &= \left( - \sum_{v \in U - \{u\}} w_{uv} \text{act}_u^{(\text{new})} \text{act}_v + \theta_u \text{act}_u^{(\text{new})} \right) \\ &- \left( - \sum_{v \in U - \{u\}} w_{uv} \text{act}_u^{(\text{old})} \text{act}_v + \theta_u \text{act}_u^{(\text{old})} \right) \\ &= \left( \text{act}_u^{(\text{old})} - \text{act}_u^{(\text{new})} \right) \underbrace{\left( \sum_{v \in U - \{u\}} w_{uv} \text{act}_v - \theta_u \right)}_{= \text{net}_u}.\end{aligned}$$

- $\text{net}_u < \theta_u$ : Zweiter Faktor kleiner als 0.  
 $\text{act}_u^{(\text{new})} = -1$  and  $\text{act}_u^{(\text{old})} = 1$ , daher erster Faktor größer als 0.

**Ergebnis:**  $\Delta E < 0$ .

- $\text{net}_u \geq \theta_u$ : Zweiter Faktor größer als oder gleich 0.  
 $\text{act}_u^{(\text{new})} = 1$  und  $\text{act}_u^{(\text{old})} = -1$ , daher erster Faktor kleiner als 0.

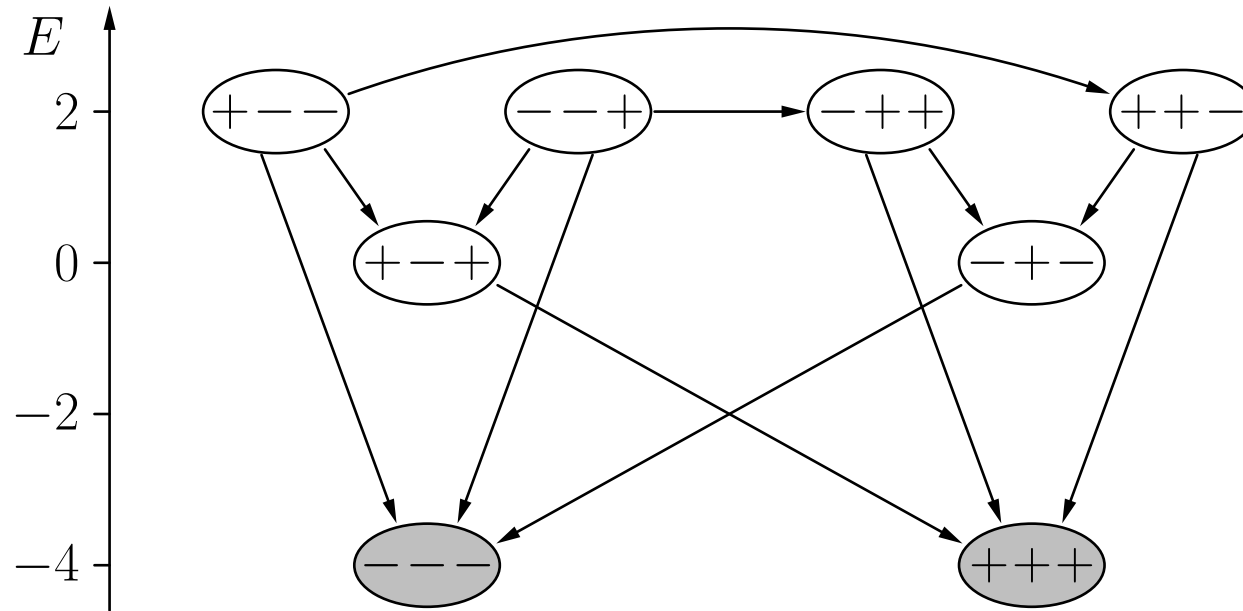
**Ergebnis:**  $\Delta E \leq 0$ .

## Höchstens $n \cdot 2^n$ Schritte bis zur Konvergenz:

- die beliebige, aber feste Reihenfolge sorgt dafür, dass alle Neuronen zyklisch durchlaufen und Neuberechnet werden
  - a) es ändert sich keine Aktivierung – ein stabiler Zustand wurde erreicht
  - b) es ändert sich mindestens eine Aktivierung: dann wurde damit mindestens einer der  $2^n$  möglichen Aktivierungszustände ausgeschlossen.
- Ein einmal verlassener Zustand kann nicht wieder erreicht werden (siehe vorherige Folien).
- D.h. nach spätestens  $2^n$  Durchläufen durch die  $n$  Neuronen ist ein stabiler Zustand erreicht.

# Hopfield-Netze: Beispiele

Ordne die Zustände im Graphen entsprechend ihrer Energie

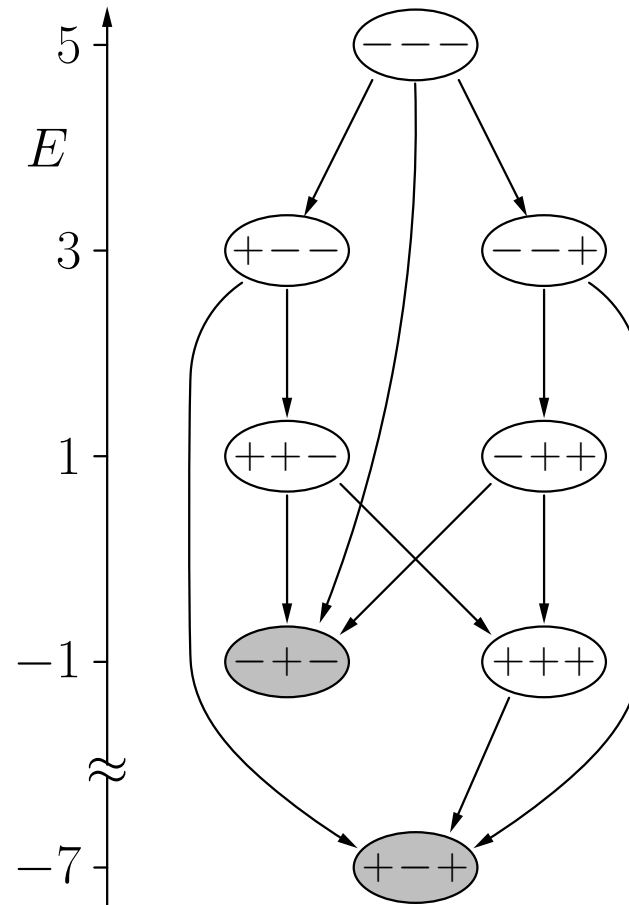
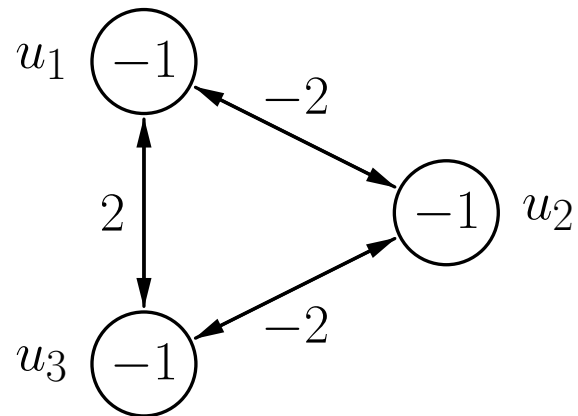


Energiefunktion für das Beispiel-Hopfield-Netz:

$$E = - \text{act}_{u_1} \text{act}_{u_2} - 2 \text{act}_{u_1} \text{act}_{u_3} - \text{act}_{u_2} \text{act}_{u_3} .$$

# Hopfield-Netze: Beispiele

Der Zustandsgraph muss nicht symmetrisch sein





# Hopfield-Netze: Physikalische Interpretation

## Physikalische Interpretation: Magnetismus

Ein Hopfield-Netz kann als (mikroskopisches) Modell von Magnetismus gesehen werden (sogenanntes Ising-Modell, [Ising 1925]).

physikalisch	neuronal
Atom	Neuron
Magnetisches Moment (Spin)	Aktivierungszustand
Stärke des äußeren Magnetfeldes	Schwellenwert
Magnetische Kopplung der Atome	Verbindungsgewichte
Hamilton-Operator des Magnetfeldes	Energiefunktion

# Hopfield-Netze: Assoziativspeicher

**Idee: Nutze stabile Zustände, um Muster zu speichern**

Zuerst: Speichere nur ein Muster  $\vec{x} = (\text{act}_{u_1}^{(l)}, \dots, \text{act}_{u_n}^{(l)})^\top \in \{-1, 1\}^n$ ,  $n \geq 2$ ,  
d.h. finde Gewichte, so dass der Zustand ein stabiler Zustand wird.

Notwendige und hinreichende Bedingung:

$$S(\mathbf{W}\vec{x} - \vec{\theta}) = \vec{x},$$

wobei

$$\begin{aligned} S : \mathbb{R}^n &\rightarrow \{-1, 1\}^n, \\ \vec{x} &\mapsto \vec{y} \end{aligned}$$

mit

$$\forall i \in \{1, \dots, n\} : y_i = \begin{cases} 1, & \text{falls } x_i \geq 0, \\ -1, & \text{sonst.} \end{cases}$$

# Hopfield-Netze: Assoziativspeicher

Falls  $\vec{\theta} = \vec{0}$ , dann kann eine passende Matrix  $\mathbf{W}$  leicht berechnet werden. Es reicht eine Matrix  $\mathbf{W}$  zu finden mit

$$\mathbf{W}\vec{x} = c\vec{x} \quad \text{mit } c \in \mathbb{R}^+.$$

Algebraisch: Finde eine Matrix  $\mathbf{W}$  die einen positiven Eigenwert in Bezug auf  $\vec{x}$  hat.

Wähle

$$\mathbf{W} = \vec{x}\vec{x}^T - \mathbf{E}$$

wobei  $\vec{x}\vec{x}^T$  das sogenannte **äußere Produkt** von  $\vec{x}$  mit sich selbst ist.

Mit dieser Matrix erhalten wir

$$\begin{aligned} \mathbf{W}\vec{x} &= (\vec{x}\vec{x}^T)\vec{x} - \underbrace{\mathbf{E}\vec{x}}_{=\vec{x}} \stackrel{(*)}{=} \vec{x} \underbrace{(\vec{x}^T\vec{x})}_{=|\vec{x}|^2=n} - \vec{x} \\ &= n\vec{x} - \vec{x} = (n-1)\vec{x}. \end{aligned}$$

# Hopfield-Netze: Assoziativspeicher

## Hebb'sche Lernregel [Hebb 1949]

In einzelnen Gewichten aufgeschrieben lautet die Berechnung der Gewichtsmatrix wie folgt:

$$w_{uv} = \begin{cases} 0, & \text{falls } u = v, \\ 1, & \text{falls } u \neq v, \text{act}_u^{(p)} = \text{act}_u^{(v)}, \\ -1, & \text{sonst.} \end{cases}$$

- Ursprünglich von biologischer Analogie abgeleitet.
- Verstärkt Verbindungen zwischen Neuronen, die zur selben Zeit aktiv sind.

Diese Lernregel speichert auch das Komplement des Musters:

$$\text{Mit } \mathbf{W}\vec{x} = (n-1)\vec{x} \quad \text{ist daher auch} \quad \mathbf{W}(-\vec{x}) = (n-1)(-\vec{x}).$$

# Hopfield-Netze: Assoziativspeicher

## Speichern mehrerer Muster

Wähle

$$\begin{aligned}\mathbf{W}\vec{x}_j &= \sum_{i=1}^m \mathbf{W}_i \vec{x}_j = \left( \sum_{i=1}^m (\vec{x}_i \vec{x}_i^T) \right) \vec{x}_j - m \underbrace{\mathbf{E} \vec{x}_j}_{=\vec{x}_j} \\ &= \left( \sum_{i=1}^m \vec{x}_i (\vec{x}_i^T \vec{x}_j) \right) - m \vec{x}_j\end{aligned}$$

Wenn die Muster orthogonal sind, gilt

$$\vec{x}_i^T \vec{x}_j = \begin{cases} 0, & \text{falls } i \neq j, \\ n, & \text{falls } i = j, \end{cases}$$

und daher

$$\mathbf{W}\vec{x}_j = (n - m)\vec{x}_j.$$

## Speichern mehrerer Muster

Ergebnis: So lange  $m < n$ , ist  $\vec{x}_j$  ein stabiler Zustand des Hopfield-Netzes.

Man beachte, dass die Komplemente der Muster ebenfalls gespeichert werden.

Mit  $\mathbf{W}\vec{x}_j = (n - m)\vec{x}_j$  ist daher auch  $\mathbf{W}(-\vec{x}_j) = (n - m)(-\vec{x}_j)$ .

Aber: die Speicherkapazität ist verglichen mit der Anzahl möglicher Zustände sehr klein ( $2^n$ ).

## Nicht-orthogonale Muster:

$$\mathbf{W}\vec{x}_j = (n - m)\vec{x}_j + \underbrace{\sum_{\substack{i=1 \\ i \neq j}}^m \vec{x}_i (\vec{x}_i^T \vec{x}_j)}_{\text{“Störterm”}}.$$

- $p_j$  kann trotzdem stabil sein, wenn  $n - m \geq 0$  gilt und der “Störterm” hinreichend klein ist.
- Dieser Fall tritt ein, wenn die Muster “annähernd” orthogonal sind.
- Je größer die Zahl der zu speichernden Muster ist, desto kleiner muß der Störterm sein.
- Die theoretische Maximalkapazität eines Hopfield-Netzes wird praktisch nie erreicht.

# Assoziativspeicher: Beispiel

Beispiel: Speichere Muster  $\vec{x}_1 = (+1, +1, -1, -1)^\top$  und  $\vec{x}_2 = (-1, +1, -1, +1)^\top$ .

$$\mathbf{W} = \mathbf{W}_1 + \mathbf{W}_2 = \vec{x}_1 \vec{x}_1^T + \vec{x}_2 \vec{x}_2^T - 2\mathbf{E}$$

wobei

$$\mathbf{W}_1 = \begin{pmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{pmatrix}, \quad \mathbf{W}_2 = \begin{pmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{pmatrix}.$$

Die vollständige Gewichtsmatrix ist:

$$\mathbf{W} = \begin{pmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \\ 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \end{pmatrix}.$$

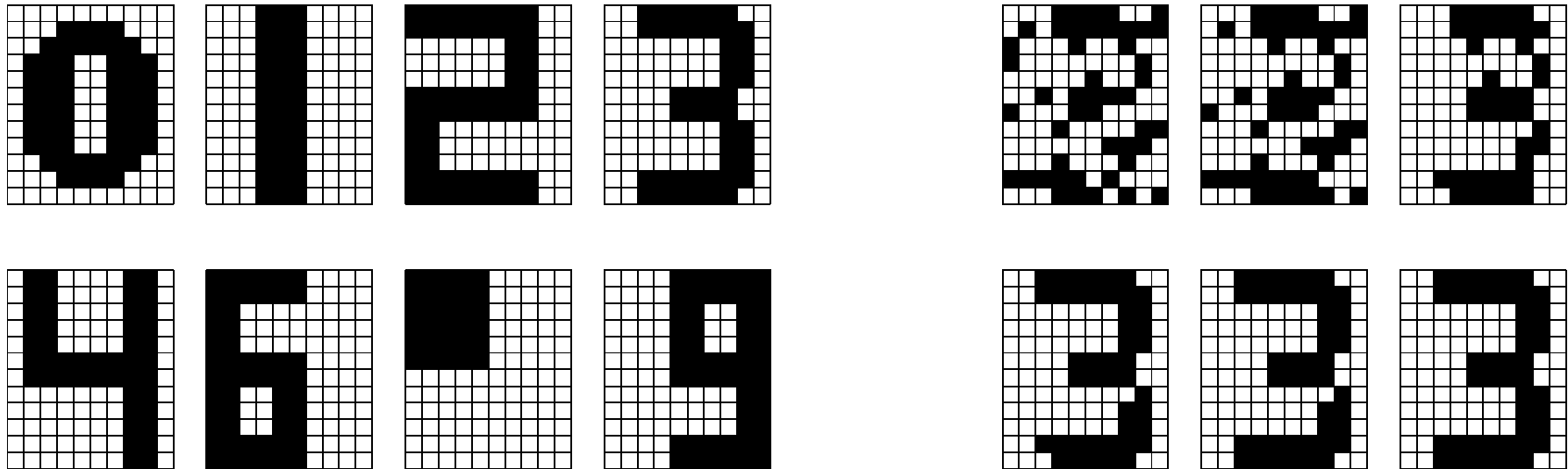
Daher ist

$$\mathbf{W}\vec{x}_1 = (+2, +2, -2, -2)^\top \quad \text{und} \quad \mathbf{W}\vec{x}_2 = (-2, +2, -2, +2)^\top.$$



# Assoziativspeicher: Beispiele

## Beispiel: Speichere Bitmaps von Zahlen



- Links: Bitmaps, die im Hopfield-Netz gespeichert sind.
- Rechts: Rekonstruktion eines Musters aus einer zufälligen Eingabe.

# Hopfield-Netze: Assoziativspeicher

## Trainieren eines Hopfield-Netzes mit der Delta-Regel

Notwendige Bedingung, dass ein Muster  $\vec{x}$  einem stabilen Zustand entspricht:

$$\begin{array}{rccccccc} s(0 & & + w_{u_1 u_2} \text{act}_{u_2}^{(p)} & + \dots & + w_{u_1 u_n} \text{act}_{u_n}^{(p)} & - \theta_{u_1} & = \text{act}_{u_1}^{(p)}, \\ s(w_{u_2 u_1} \text{act}_{u_1}^{(p)} & + 0 & & + \dots & + w_{u_2 u_n} \text{act}_{u_n}^{(p)} & - \theta_{u_2} & = \text{act}_{u_2}^{(p)}, \\ \vdots & & \vdots & & \vdots & & \vdots \\ s(w_{u_n u_1} \text{act}_{u_1}^{(p)} & + w_{u_n u_2} \text{act}_{u_2}^{(p)} & + \dots & + 0 & & - \theta_{u_n} & = \text{act}_{u_n}^{(p)}. \end{array}$$

mit der standardmäßigen Schwellenwertfunktion

$$s(x) = \begin{cases} 1, & \text{falls } x \geq 0, \\ -1, & \text{sonst.} \end{cases}$$

## Trainieren eines Hopfield-Netzes mit der Delta-Regel

Überführe Gewichtsmatrix in einen Gewichtsvektor:

$$\vec{w} = \left( \begin{array}{cccc} w_{u_1 u_2}, & w_{u_1 u_3}, & \dots, & w_{u_1 u_n}, \\ & w_{u_2 u_3}, & \dots, & w_{u_2 u_n}, \\ & & \ddots & \vdots \\ & & & w_{u_{n-1} u_n}, \\ -\theta_{u_1}, & -\theta_{u_2}, & \dots, & -\theta_{u_n} \end{array} \right).$$

Konstruiere Eingabevektoren für ein Schwellenwertelement

$$\vec{z}_2 = \left( \text{act}_{u_1}^{(p)}, \underbrace{0, \dots, 0}_{n-2 \text{ Nullen}}, \text{act}_{u_3}^{(p)}, \dots, \text{act}_{u_n}^{(p)}, \dots, 0, 1, \underbrace{0, \dots, 0}_{n-2 \text{ Nullen}} \right).$$

Wende die Deltaregel auf diesen Gewichtsvektor und die Eingabevektoren an, bis sich Konvergenz einstellt.

# Hopfield-Netze: Lösen von Optimierungsproblemen

## Nutze Energieminimierung, um Optimierungsprobleme zu lösen

Allgemeine Vorgehensweise:

- Transformiere die zu optimierende Funktion in eine zu minimierende.
- Transformiere Funktion in die Form einer Energiefunktion eines Hopfield-Netzes.
- Lies die Gewichte und Schwellenwerte der Energiefunktion ab.
- Konstruiere das zugehörige Hopfield-Netz.
- Initialisiere das Hopfield-Netz zufällig und aktualisiere es solange, bis sich Konvergenz einstellt.
- Lies die Lösung aus dem erreichten stabilen Zustand ab.
- Wiederhole mehrmals und nutze die beste gefundene Lösung.

# Hopfield-Netze: Aktivierungstransformation

Ein Hopfield-Netz kann entweder mit Aktivierungen  $-1$  und  $1$  oder mit Aktivierungen  $0$  and  $1$  definiert werden. Die Netze können ineinander umgewandelt werden.

Von  $\text{act}_u \in \{-1, 1\}$  in  $\text{act}_u \in \{0, 1\}$ :

$$\begin{aligned}w_{uv}^0 &= 2w_{uv}^- & \text{und} \\ \theta_u^0 &= \theta_u^- + \sum_{v \in U - \{u\}} w_{uv}^- \end{aligned}$$

Von  $\text{act}_u \in \{0, 1\}$  in  $\text{act}_u \in \{-1, 1\}$ :

$$\begin{aligned}w_{uv}^- &= \frac{1}{2}w_{uv}^0 & \text{und} \\ \theta_u^- &= \theta_u^0 - \frac{1}{2} \sum_{v \in U - \{u\}} w_{uv}^0. \end{aligned}$$

# Hopfield-Netze: Lösen von Optimierungsproblemen

**Kombinationslemma:** Gegeben seien zwei Hopfield-Netze auf derselben Menge  $U$  Neuronen mit Gewichten  $w_{uv}^{(i)}$ , Schwellenwerten  $\theta_u^{(i)}$  und Energiefunktionen

$$E_i = -\frac{1}{2} \sum_{u \in U} \sum_{v \in U - \{u\}} w_{uv}^{(i)} \text{act}_u \text{act}_v + \sum_{u \in U} \theta_u^{(i)} \text{act}_u,$$

$i = 1, 2$ . Weiterhin sei  $a, b \in \mathbb{R}$ . Dann ist  $E = aE_1 + bE_2$  die Energiefunktion des Hopfield-Netzes auf den Neuronen in  $U$  das die Gewichte  $w_{uv} = aw_{uv}^{(1)} + bw_{uv}^{(2)}$  und die Schwellenwerte  $\theta_u = a\theta_u^{(1)} + b\theta_u^{(2)}$  hat.

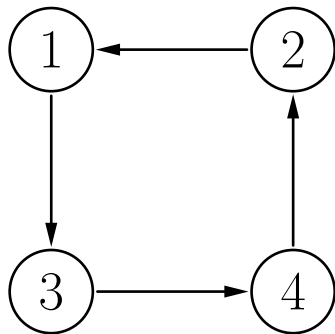
Beweis: Einfach Berechnungen durchführen.

Idee: Zusätzliche Bedingungen können separat formuliert und später mit einbezogen werden.

# Hopfield-Netze: Lösen von Optimierungsproblemen

## Beispiel: Problem des Handlungsreisenden (TSP – Traveling Salesman Problem)

Idee: Stelle Tour durch Matrix dar.



Stadt				
1	2	3	4	
1	0	0	0	1.
0	0	1	0	2. Schritt
0	0	0	1	3.
0	1	0	0	4.

Ein Element  $m_{ij}$  der Matrix ist 1 wenn die  $i$ -te Stadt im  $j$ -ten Schritt besucht wird und 0 sonst.

Jeder Matrixeintrag wird durch ein Neuron repräsentiert.

## Minimierung der Tourlänge

$$E_1 = \sum_{j_1=1}^n \sum_{j_2=1}^n \sum_{i=1}^n d_{j_1 j_2} \cdot m_{i j_1} \cdot m_{(i \bmod n)+1, j_2}.$$

Doppelsumme über die benötigten Schritte (Index  $i$ ):

$$E_1 = \sum_{(i_1, j_1) \in \{1, \dots, n\}^2} \sum_{(i_2, j_2) \in \{1, \dots, n\}^2} d_{j_1 j_2} \cdot \delta_{(i_1 \bmod n)+1, i_2} \cdot m_{i_1 j_1} \cdot m_{i_2 j_2},$$

wobei

$$\delta_{ab} = \begin{cases} 1, & \text{falls } a = b, \\ 0, & \text{sonst.} \end{cases}$$

Symmetrische Version der Energiefunktion:

$$E_1 = -\frac{1}{2} \sum_{\substack{(i_1, j_1) \in \{1, \dots, n\}^2 \\ (i_2, j_2) \in \{1, \dots, n\}^2}} -d_{j_1 j_2} \cdot (\delta_{(i_1 \bmod n)+1, i_2} + \delta_{i_1, (i_2 \bmod n)+1}) \cdot m_{i_1 j_1} \cdot m_{i_2 j_2}$$



# Hopfield-Netze: Lösen von Optimierungsproblemen

Zusätzliche Bedingungen, die erfüllt werden müssen:

- Jede Stadt wird in genau einem Schritt der Tour besucht:

$$\forall j \in \{1, \dots, n\} : \sum_{i=1}^n m_{ij} = 1,$$

d.h. jede Spalte der Matrix enthält genau eine 1.

- In jedem Schritt der Tour wird genau eine Stadt besucht:

$$\forall i \in \{1, \dots, n\} : \sum_{j=1}^n m_{ij} = 1,$$

d.h. jede Zeile der Matrix enthält genau eine 1.

Diese Bedingungen werden erfüllt durch zusätzlich zu optimierende Funktionen.

# Hopfield-Netze: Lösen von Optimierungsproblemen

Formalisierung der ersten Bedingung als Minimierungsproblem:

$$\begin{aligned} E_2^* &= \sum_{j=1}^n \left( \left( \sum_{i=1}^n m_{ij} \right)^2 - 2 \sum_{i=1}^n m_{ij} + 1 \right) \\ &= \sum_{j=1}^n \left( \left( \sum_{i_1=1}^n m_{i_1 j} \right) \left( \sum_{i_2=1}^n m_{i_2 j} \right) - 2 \sum_{i=1}^n m_{ij} + 1 \right) \\ &= \sum_{j=1}^n \sum_{i_1=1}^n \sum_{i_2=1}^n m_{i_1 j} m_{i_2 j} - 2 \sum_{j=1}^n \sum_{i=1}^n m_{ij} + n. \end{aligned}$$

Doppelsumme über benötigte Städte (Index  $i$ ):

$$E_2 = \sum_{(i_1, j_1) \in \{1, \dots, n\}^2} \sum_{(i_2, j_2) \in \{1, \dots, n\}^2} \delta_{j_1 j_2} \cdot m_{i_1 j_1} \cdot m_{i_2 j_2} - 2 \sum_{(i, j) \in \{1, \dots, n\}^2} m_{ij}.$$

# Hopfield-Netze: Lösen von Optimierungsproblemen

Sich ergebende Energiefunktion:

$$E_2 = -\frac{1}{2} \sum_{\substack{(i_1, j_1) \in \{1, \dots, n\}^2 \\ (i_2, j_2) \in \{1, \dots, n\}^2}} -2\delta_{j_1 j_2} \cdot m_{i_1 j_1} \cdot m_{i_2 j_2} + \sum_{(i, j) \in \{1, \dots, n\}^2} -2m_{ij}$$

Die zweite zusätzliche Bedingung wird analog gehandhabt:

$$E_3 = -\frac{1}{2} \sum_{\substack{(i_1, j_1) \in \{1, \dots, n\}^2 \\ (i_2, j_2) \in \{1, \dots, n\}^2}} -2\delta_{i_1 i_2} \cdot m_{i_1 j_1} \cdot m_{i_2 j_2} + \sum_{(i, j) \in \{1, \dots, n\}^2} -2m_{ij}$$

Kombinieren der Energiefunktionen:

$$E = aE_1 + bE_2 + cE_3 \quad \text{wobei} \quad \frac{b}{a} = \frac{c}{a} > 2 \quad \max_{(j_1, j_2) \in \{1, \dots, n\}^2} d_{j_1 j_2}$$

# Hopfield-Netze: Lösen von Optimierungsproblemen

Aus der resultierenden Energiefunktionen können wir die Gewichte

$$w_{(i_1, j_1)(i_2, j_2)} = \underbrace{-ad_{j_1 j_2} \cdot (\delta_{(i_1 \bmod n)+1, i_2} + \delta_{i_1, (i_2 \bmod n)+1})}_{\text{von } E_1} \underbrace{-2b\delta_{j_1 j_2}}_{\text{von } E_2} \underbrace{-2c\delta_{i_1 i_2}}_{\text{von } E_3}$$

und die Schwellenwerte:

$$\theta_{(i, j)} = \underbrace{0a}_{\text{von } E_1} \underbrace{-2b}_{\text{von } E_2} \underbrace{-2c}_{\text{von } E_3} = -2(b + c)$$

ablesen.

Problem: die zufällige Initialisierung und die Aktualisierung bis zur Konvergenz führen nicht immer zu einer Matrix, die tatsächlich eine Tour repräsentiert, geschweige denn eine optimale Tour.