# 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.

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

# Hopfield-Netze

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

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

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

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

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

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

**Alternative Aktivierungsfunktion**

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

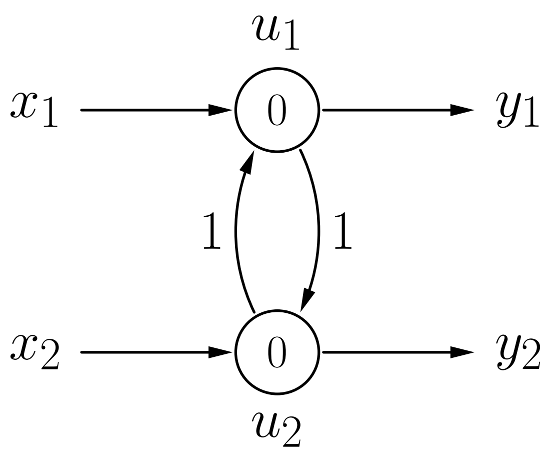
Diese Aktivierungsfunktion bietet einige Vorteile bei der späteren physikalischen Interpretation eines Hopfield-Netzes. Diese wird allerdings in der Vorlesung nicht weiter genutzt.

**Allgemeine Gewichtsmatrix eines Hopfield-Netzes**

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

# 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**

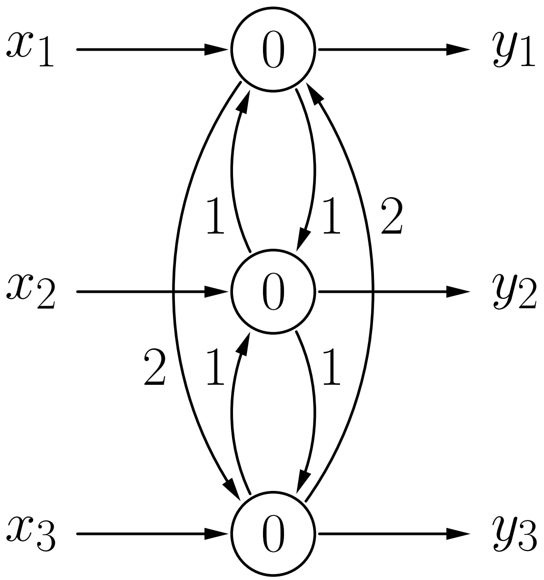
| | $u_1$ | $u_2$ |
|---|---|---|
| Eingabephase | $\mathbf{-1}$ | $\mathbf{1}$ |
| Arbeitsphase | $\mathbf{1}$ | $1$ |
| | $1$ | $\mathbf{1}$ |
| | $\mathbf{1}$ | $1$ |
| | $1$ | $\mathbf{1}$ |

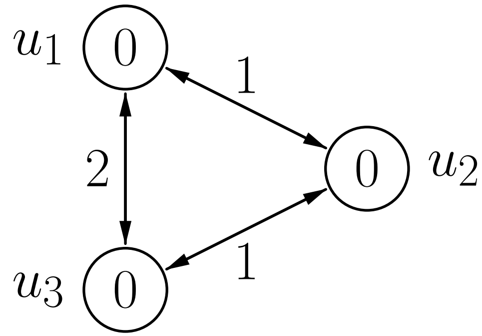
| | $u_1$ | $u_2$ |
|---|---|---|
| Eingabephase | $\mathbf{-1}$ | $\mathbf{1}$ |
| Arbeitsphase | $-1$ | $\mathbf{-1}$ |
| | $\mathbf{-1}$ | $-1$ |
| | $-1$ | $\mathbf{-1}$ |
| | $\mathbf{-1}$ | $-1$ |

- Aktivierungsreihenfolge $u_1, u_2, u_1, \ldots$ bzw. $u_2, u_1, u_1, \ldots$
- 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

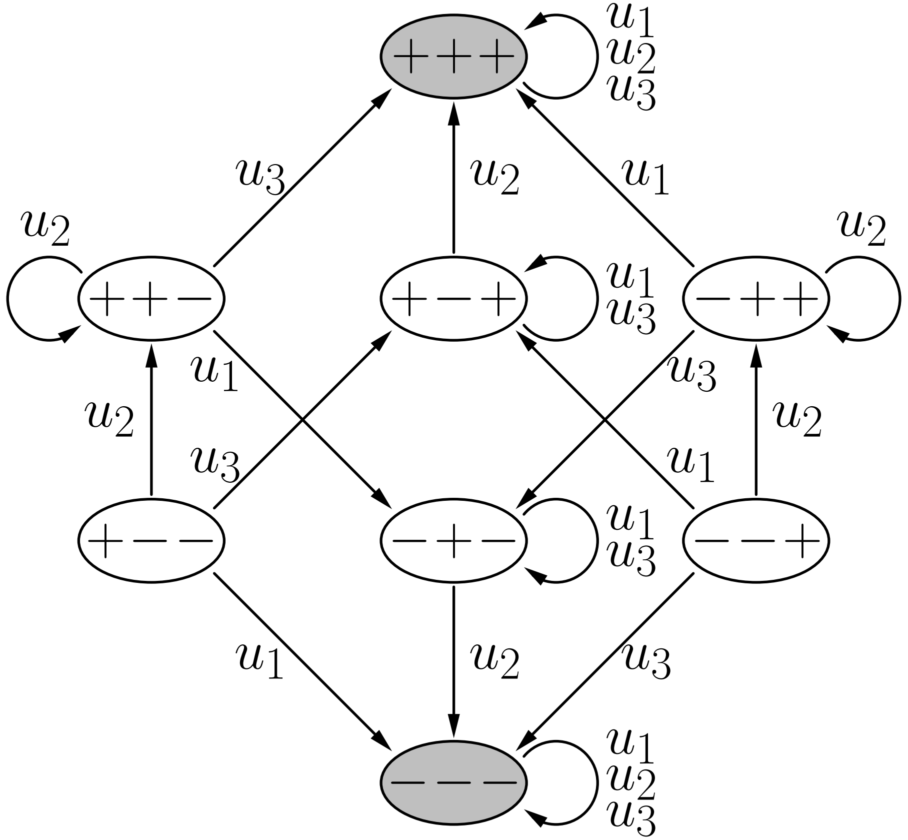



$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$

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

## 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

**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 $u_1, \ldots, u_n$ ist

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

# Hopfield-Netze: Konvergenz

Man betrachte die Energieänderung die aus einer aktivierungsändernden Aktualisierung entsteht:

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

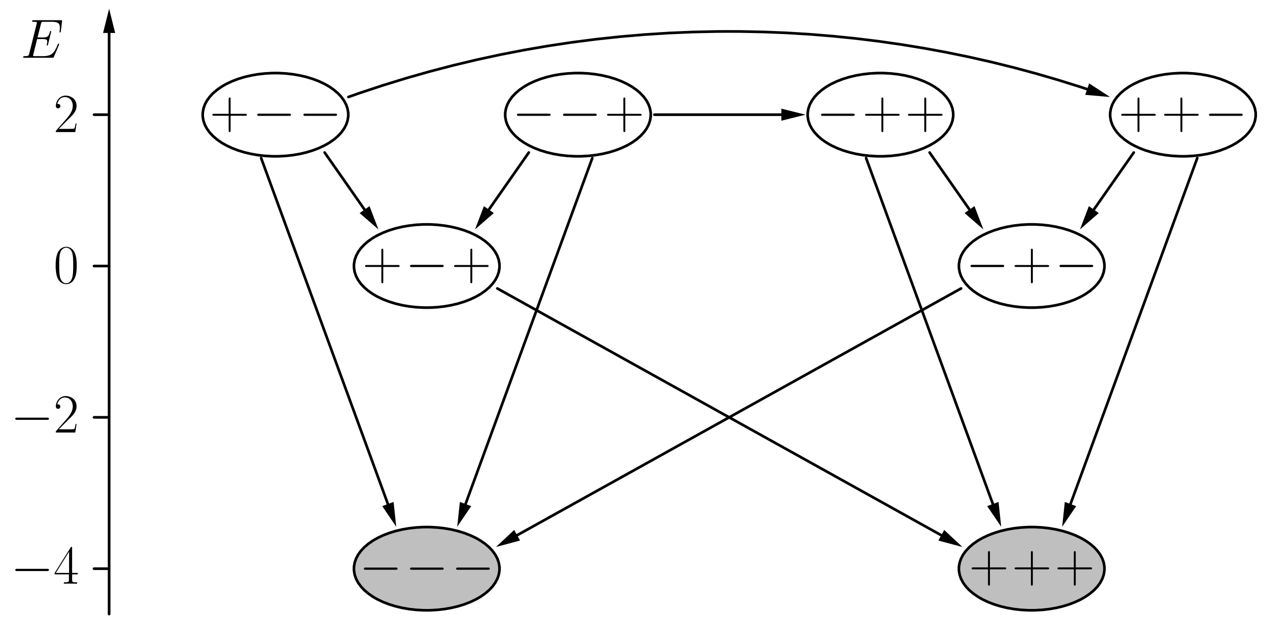
- $\operatorname{net}_u < \theta_u$: Zweiter Faktor kleiner als 0.
  $\operatorname{act}_u^{(\text{new})} = -1$ and $\operatorname{act}_u^{(\text{old})} = 1$, daher erster Faktor größer als 0.
  **Ergebnis:** $\Delta E < 0$.
- $\operatorname{net}_u \geq \theta_u$: Zweiter Faktor größer als oder gleich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

# Hopfield-Netze: Konvergenz

**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.

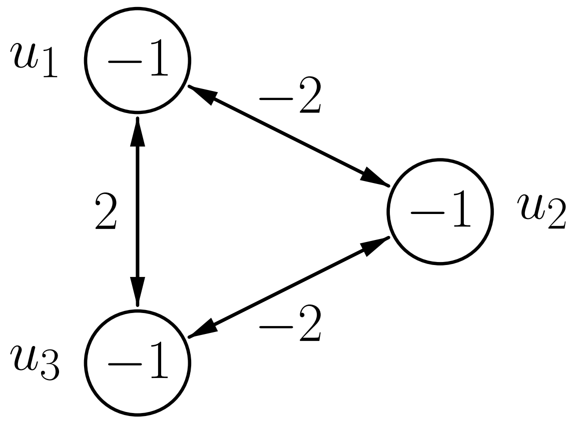
**Ordne die Zustände im Graphen entsprechend ihrer Energie**

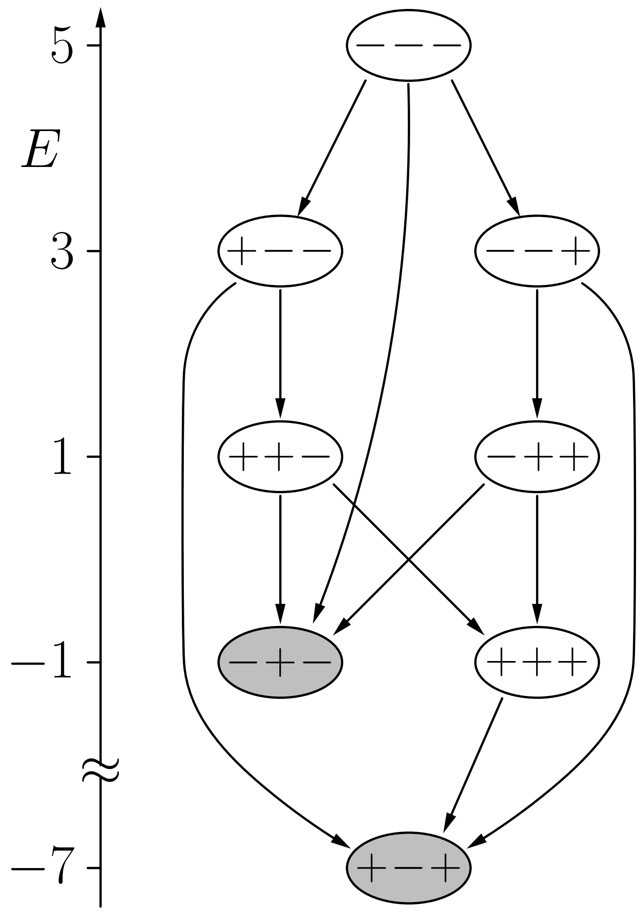



Energiefunktion für das Beispiel-Hopfield-Netz:

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

**Der Zustandsgraph muss nicht symmetrisch sein**



## 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 |

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

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

Notwendige und hinreichende Bedingung:

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

wobei

$$\begin{array}{rcl} S : \mathbb{R}^n & \to & \{-1, 1\}^n, \\ \boldsymbol{x} & \mapsto & \boldsymbol{y} \end{array}$$

mit

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

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

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

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

Wähle

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

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

Mit dieser Matrix erhalten wir

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

**Hebb'sche Lernregel** [Hebb 1949]

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

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

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

Diese Lernregel speichert auch das Komplement des Musters:

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

**Speichern mehrerer Muster**

Wähle

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

Wenn die Muster orthogonal sind, gilt

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

und daher

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

**Speichern mehrerer Muster**

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

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

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

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

**Nicht-orthogonale Muster:**

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

- $p_j$ 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 $\boldsymbol{x}_1 = (+1, +1, -1, -1)^\top$ und $\boldsymbol{x}_2 = (-1, +1, -1, +1)^\top$.

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

wobei

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

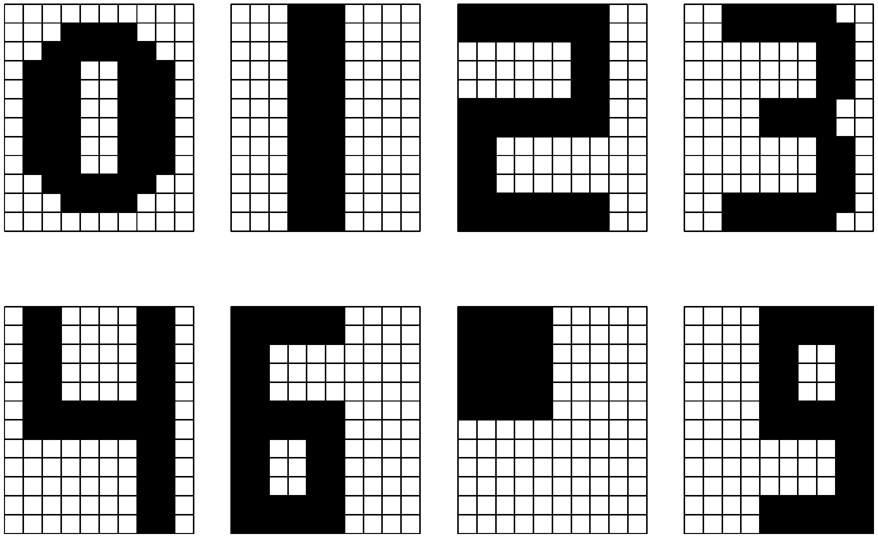
Die vollständige Gewichtsmatrix ist:

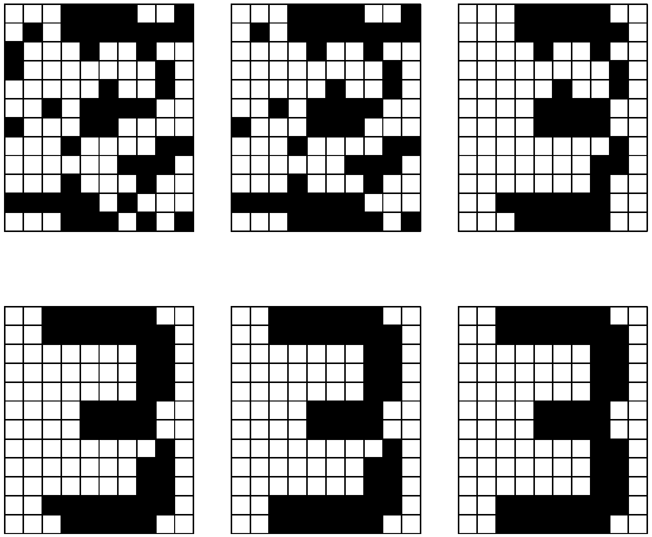
$$\mathbf{W} = \begin{pmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \\ 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \end{pmatrix}.$$

Daher ist

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

**Beispiel: Speichere Bitmaps von Zahlen**





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

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

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

$$\begin{array}{lllll}
s(0 & + w_{u_1u_2}\,\mathrm{act}_{u_2}^{(p)} & + \ldots + w_{u_1u_n}\,\mathrm{act}_{u_n}^{(p)} & - \theta_{u_1}) & = \mathrm{act}_{u_1}^{(p)}, \\
s(w_{u_2u_1}\,\mathrm{act}_{u_1}^{(p)} & + 0 & + \ldots + w_{u_2u_n}\,\mathrm{act}_{u_n}^{(p)} & - \theta_{u_2}) & = \mathrm{act}_{u_2}^{(p)}, \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
s(w_{u_nu_1}\,\mathrm{act}_{u_1}^{(p)} & + w_{u_nu_2}\,\mathrm{act}_{u_2}^{(p)} & + \ldots + 0 & - \theta_{u_n}) & = \mathrm{act}_{u_n}^{(p)}.
\end{array}$$

mit der standardmäßigen Schwellenwertfunktion

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

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

Überführe Gewichtsmatrix in einen Gewichtsvektor:

$$\boldsymbol{w} = \begin{pmatrix} w_{u_1u_2}, & w_{u_1u_3}, & \ldots, & w_{u_1u_n}, \\ & w_{u_2u_3}, & \ldots, & w_{u_2u_n}, \\ & & \ddots & \vdots \\ & & & w_{u_{n-1}u_n}, \\ -\theta_{u_1}, & -\theta_{u_2}, & \ldots, & -\theta_{u_n} \end{pmatrix}.$$

Konstruiere Eingabevektoren für ein Schwellenwertelement

$$\boldsymbol{z}_2 = (\mathrm{act}_{u_1}^{(p)}, \underbrace{0, \ldots, 0,}_{n-2 \text{ Nullen}} \mathrm{act}_{u_3}^{(p)}, \ldots, \mathrm{act}_{u_n}^{(p)}, \ldots 0, 1, \underbrace{0, \ldots, 0}_{n-2 \text{ Nullen}}).$$

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

## 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 $\mathrm{act}_u \in \{-1, 1\}$ in $\mathrm{act}_u \in \{0, 1\}$:

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

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

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

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

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

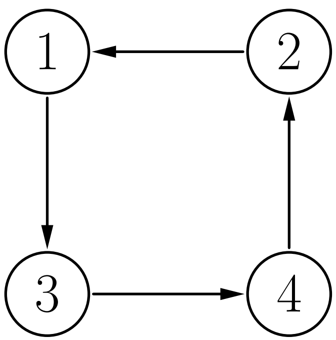
$i = 1, 2$. Weiterhin sei $a, b \in \mathbb{R}$. Dann ist $E = aE_1 + bE_2$ die Energiefunktion des Hopfield-Netzes auf den Neuronen in $U$ das die Gewichte $w_{uv} = aw_{uv}^{(1)} + bw_{uv}^{(2)}$ und die Schwellenwerte $\theta_u = a\theta_u^{(1)} + b\theta_u^{(2)}$ hat.

Beweis: Einfach Berechnungen durchführen.

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

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

Idee: Stelle Tour durch Matrix dar.




$$\begin{array}{c} \text{Stadt} \\ \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \quad \begin{array}{l} 1. \\ 2. \\ 3. \\ 4. \end{array} \ \text{Schritt}$$

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 $i$):

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

wobei

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

Symmetrische Version der Energiefunktion:

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

# Hopfield-Netze: Lösen von Optimierungsproblemen

Zusätzliche Bedingungen, die erfüllt werden müssen:

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

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

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

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

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

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

Diese Bedingungen werden erfüllt durch zusätzlich zu optimierende Funktionen.

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 $i$):

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

Sich ergebende Energiefunktion:

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

Die zweite zusätzliche Bedingung wird analog gehandhabt:

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

Kombinieren der Energiefunktionen:

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

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.