DD2437 – Artificial Neural Networks and Deep Architectures (annda)

Lecture 8: **Hopfield networks**

Pawel Herman

Computational Science and Technology (CST)

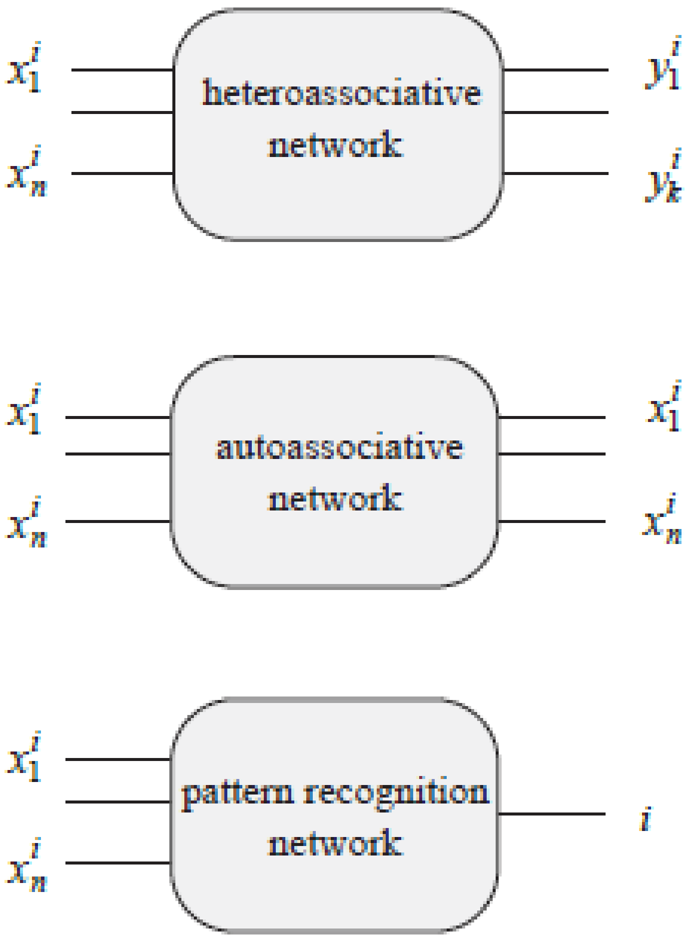
KTH Royal Institute of Technology

February 2019

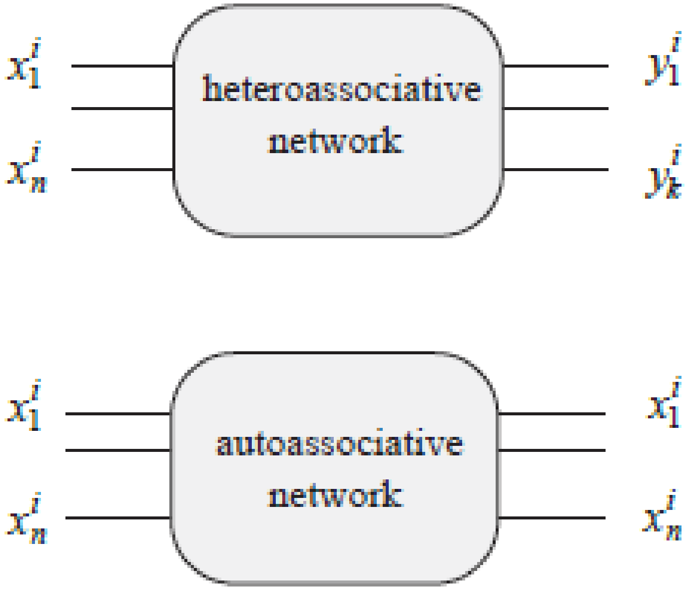
Lecture overview

- Associative memory, learning
- Hopfield networks
- Storage capacity
- Optimisation with Hopfield networks

Associative pattern recognition

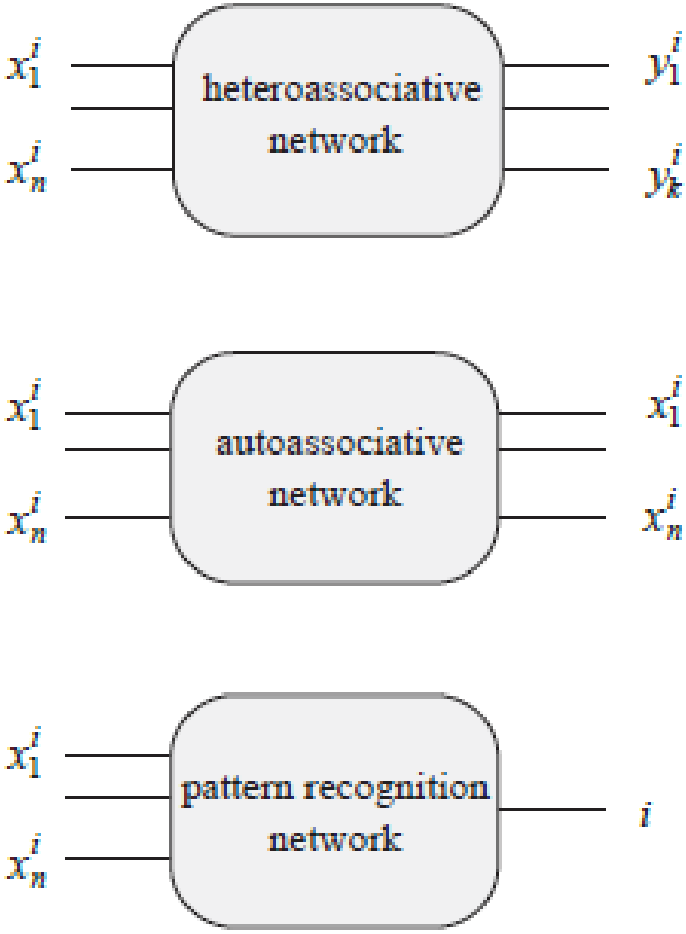


Associative pattern recognition



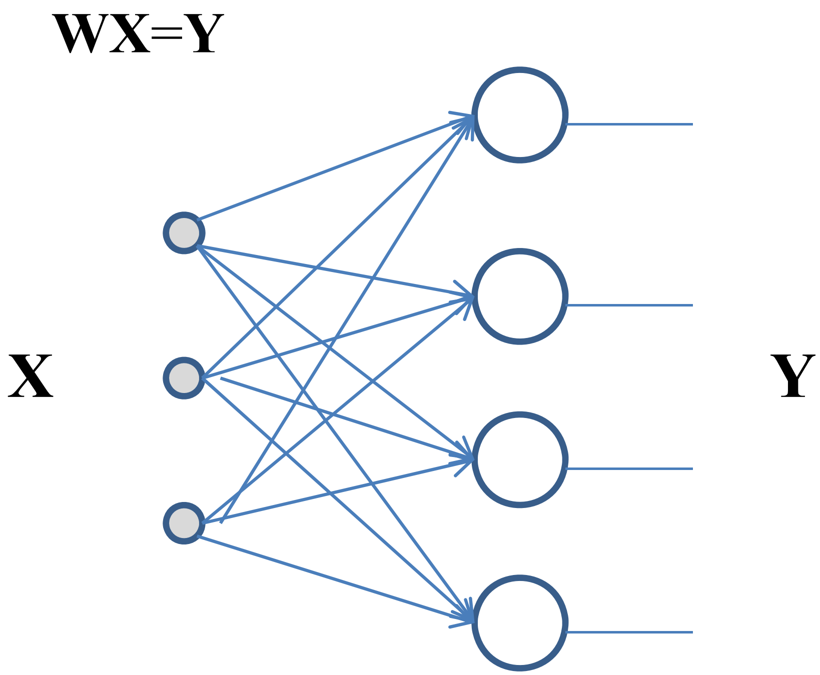
boat

Associative pattern recognition



Linear associative memory networks

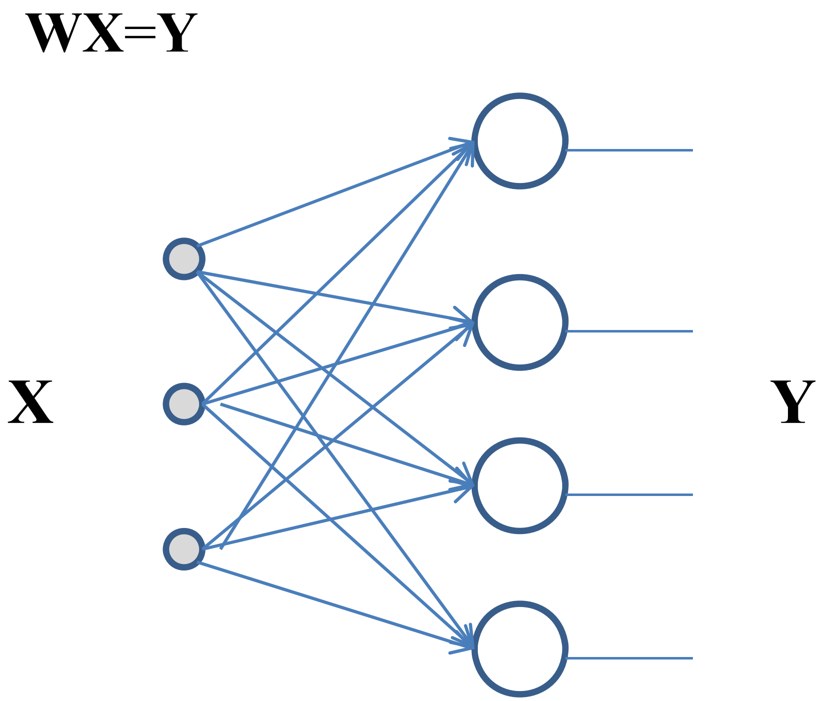
- Single layer networks (see lecture 2, correlation memory)

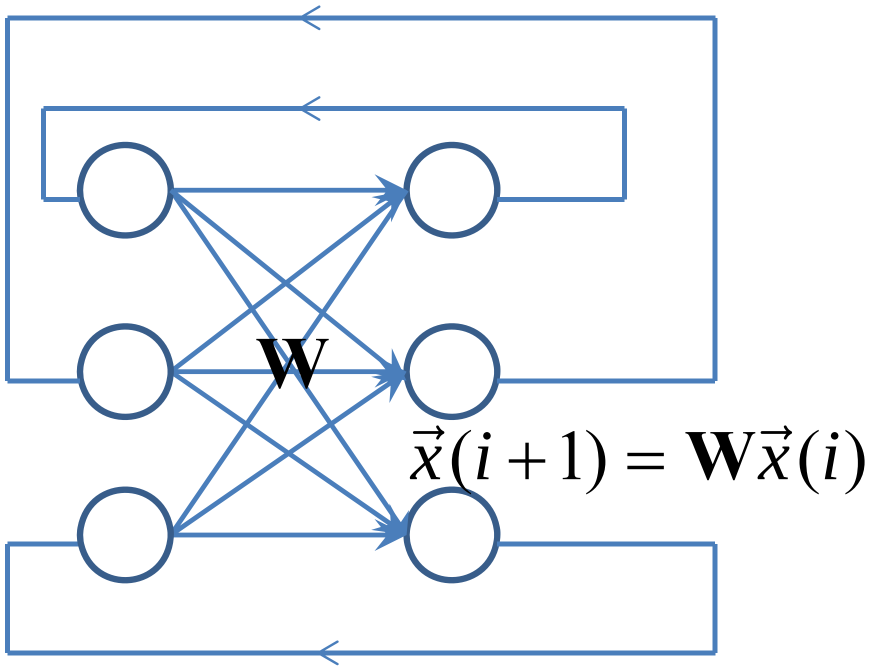


without feedback
(recall is a feedforward step)

Linear associative memory networks

- Simple single layer or recurrent networks





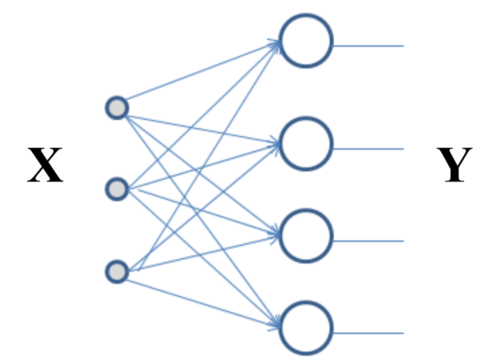
without feedback
(recall is a feedforward step)

<u>autoassociative</u> recurrent network, <u>with feedback</u>
(recall is an iterative process)

Associative learning in a single layer network

- Bipolar coding {-1, 1} with sign transform:

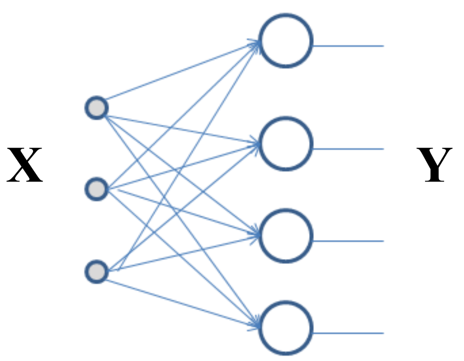
$$\operatorname{sgn}(x) = \begin{cases} 1 & , x \geq 0 \\ -1 & , x < 0 \end{cases}$$



Associative learning in a single layer network

- Bipolar coding {-1, 1} with sign transform:

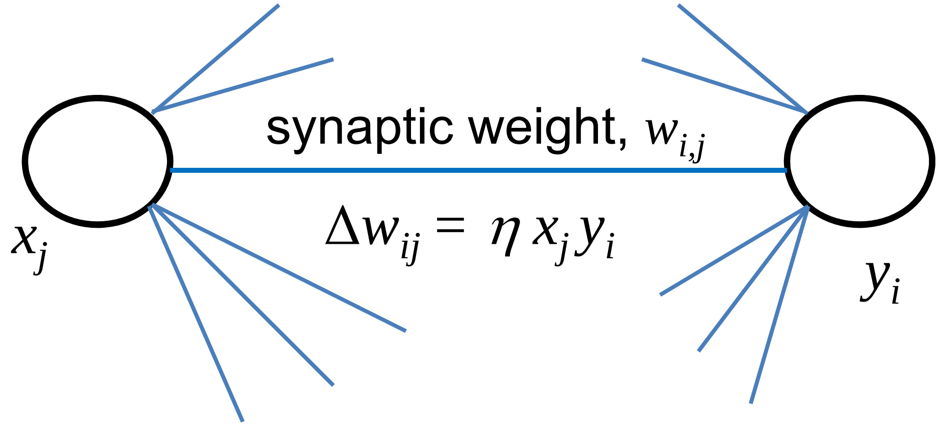
$$\operatorname{sgn}(x) = \begin{cases} 1 \;, x \geq 0 \\ -1 \,, x < 0 \end{cases}$$



- Hebbian learning (correlation learning, outer product)

$$\mathbf{W} = \mathbf{W}^1 + \mathbf{W}^2 + \ldots + \mathbf{W}^m$$

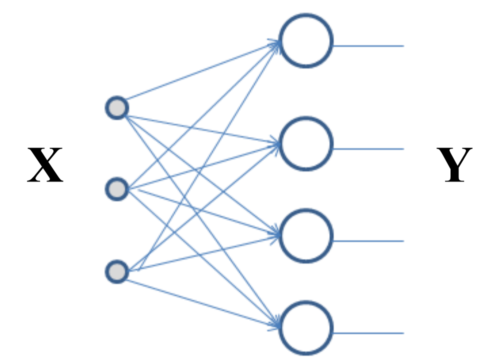
$$\mathbf{W}^k = [w_{ij}] = [x_j^k \; y_i^k] \quad (\textit{outer product})$$



Associative learning in a single layer network

- Bipolar coding {-1, 1} with sign transform:

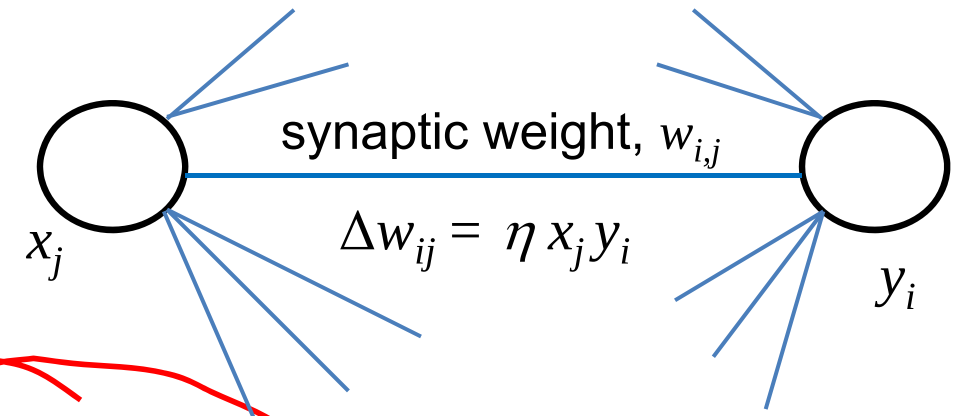
$$\mathrm{sgn}(x) = \begin{cases} 1 \ , x \geq 0 \\ -1 \ , x < 0 \end{cases}$$



- Hebbian learning (correlation learning, outer product)

$$\mathbf{W} = \mathbf{W}^1 + \mathbf{W}^2 + \ldots + \mathbf{W}^m$$

$$\mathbf{W}^k = [w_{ij}] = [x_j^k \ y_i^k] \quad \textit{(outer product)}$$



$$\vec{y}_p(\vec{x}_p^{\mathrm{T}}\vec{x}_p) + \sum_{p \neq k}^{m} \vec{y}_k(\vec{x}_p^{\mathrm{T}}\vec{x}_k)$$

crosstalk

Hebbian learning for associative memory

- Autoassociative case

$$\mathbf{W} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$$

$$\operatorname{sgn}(\mathbf{W}\vec{x}) = \vec{x}, \quad \operatorname{sgn}(\mathbf{W}\mathbf{X}) = \mathbf{X}$$

Essentially, $\vec{x}$ are the eigenvectors of nonlinear sgn operation so the idea is to find $\mathbf{W}$ for which sgn($\mathbf{W}\mathbf{X}$) has these patterns as eigenvectors,
<u>but we do not want $\mathbf{W}$ = $\mathbf{I}$ as a trivial solution of $\operatorname{sgn}(\mathbf{W}\mathbf{X}) = \mathbf{X}$</u>

$$\text{for } \mathbf{W} = \mathbf{X}\mathbf{X}^{\mathrm{T}}, \; \operatorname{sgn}(\mathbf{W}\mathbf{X}) = \operatorname{sgn}(\mathbf{X}\underbrace{\mathbf{X}^{\mathrm{T}}\mathbf{X}}) = \operatorname{sgn}(\mathbf{X}) = \mathbf{X}$$

For orthogonal $\mathbf{X}$ (or nearly),
$\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is a scaled identity $\mathbf{I}$ matrix

Hebbian learning for associative memory

- Autoassociative case

$$\mathbf{W} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$$

$$\operatorname{sgn}(\mathbf{W}\vec{x}) = \vec{x}, \quad \operatorname{sgn}(\mathbf{W}\mathbf{X}) = \mathbf{X}$$

Essentially, $\vec{x}$ are the eigenvectors of nonlinear sgn operation so the idea is to find $\mathbf{W}$ for which sgn($\mathbf{W}\mathbf{X}$) has these patterns as eigenvectors,

$$\mathbf{W} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$$

From a geometrical perspective:

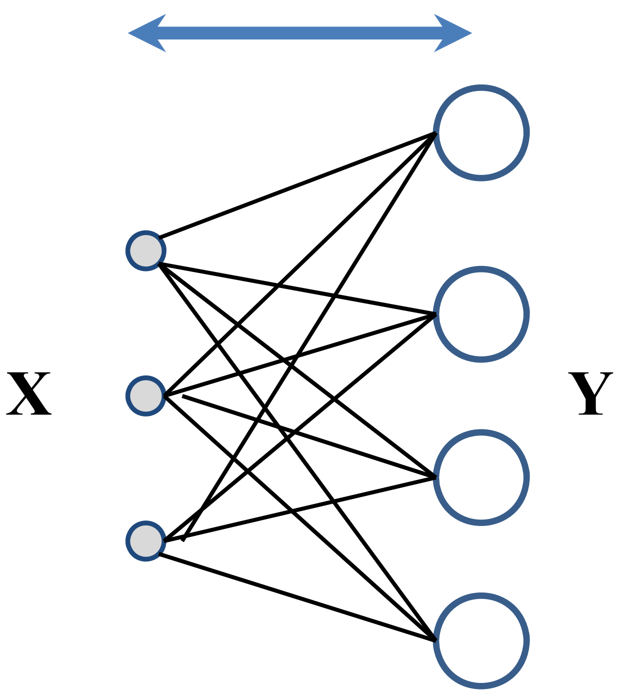
$\mathbf{W}$ describes *non-orthogonal* projection on the subspace spanned by $\vec{X}$.

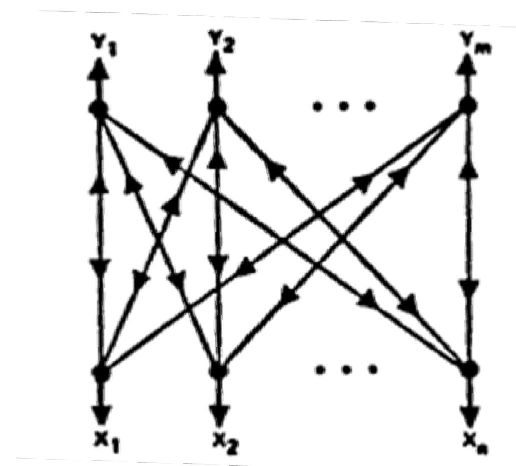
Bidirectional associative memory (resonance)

Builds on the concept of memory networks with feedback (recursive)

- bipolar {-1, 1} coding

- sign activation function





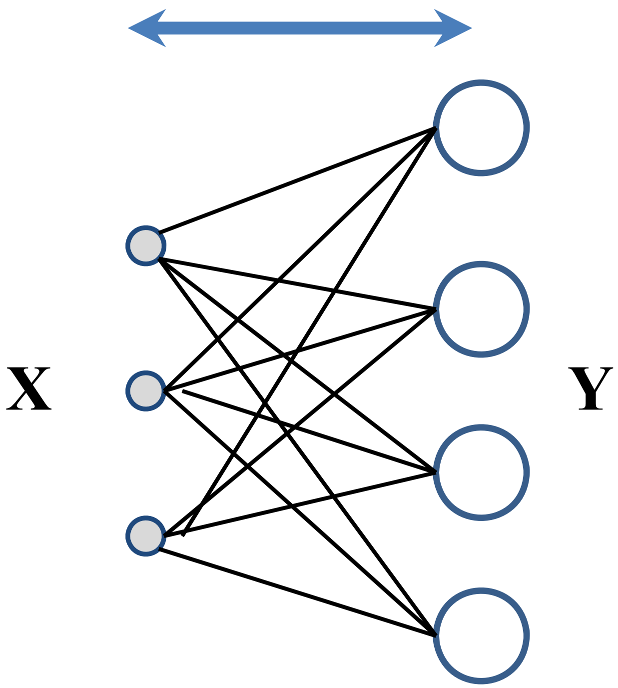
B. Kosko, 1988

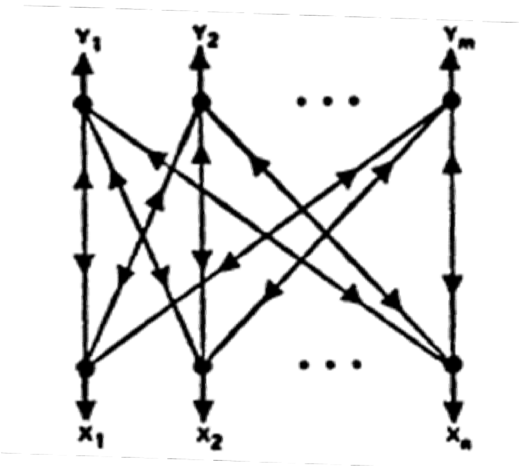
Bidirectional associative memory (resonance)

Builds on the concept of memory networks with feedback (recursive)

- bipolar {-1, 1} coding

- sign activation function





B. Kosko, 1988

Bidirectionality (feedback) imposes extra challenges

- synchronous vs asynchronous update
- different properties depending on updating mode

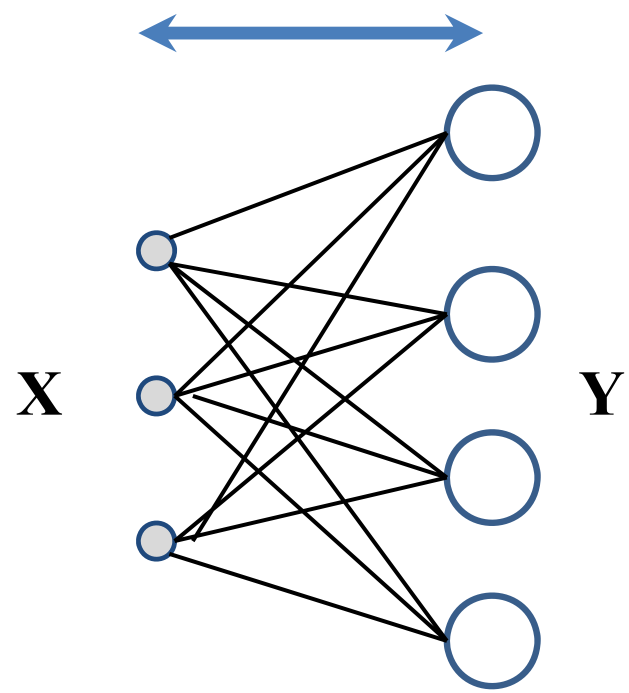
Bidirectional associative memory (resonance)

Builds on the concept of memory networks with feedback (recursive)

- bipolar {-1, 1} coding

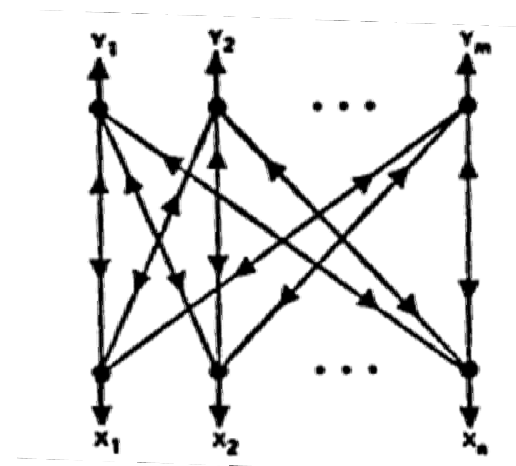
- sign activation function

$$\vec{y}(t) = \mathrm{sgn}(\mathbf{W}\vec{x}(t))$$
$$\vec{x}(t+1) = \mathrm{sgn}(\mathbf{W}\vec{y}(t))$$



Does it converge?

What are stable points?

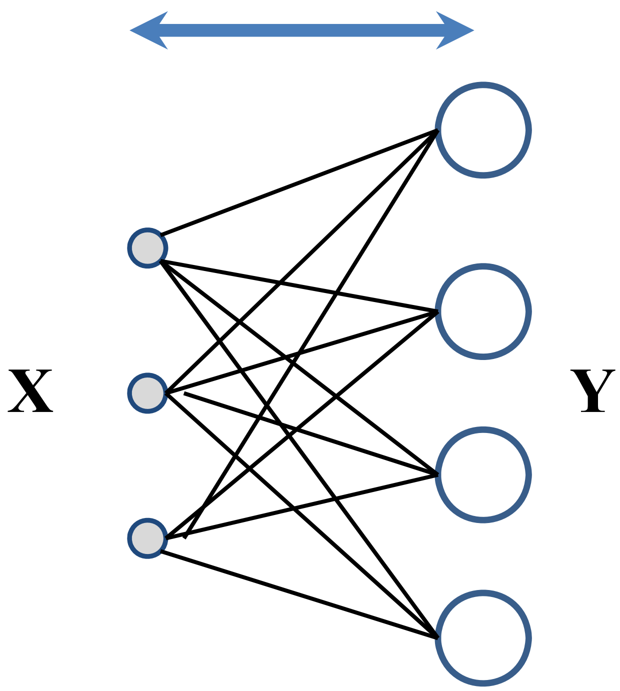


B. Kosko, 1988

Bidirectionality (feedback) imposes extra challenges

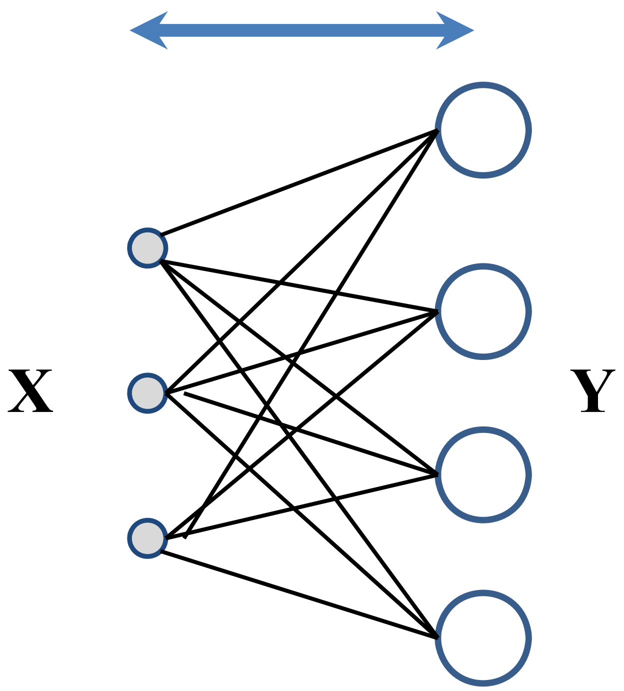
- synchronous vs asynchronous update
- different properties depending on updating mode

Concept of energy in BAM



If $(\vec{x}, \vec{y})$ is a stable point, then nearby points like $(\vec{x}_0, \vec{y}_0)$ should converge.

Concept of energy in BAM

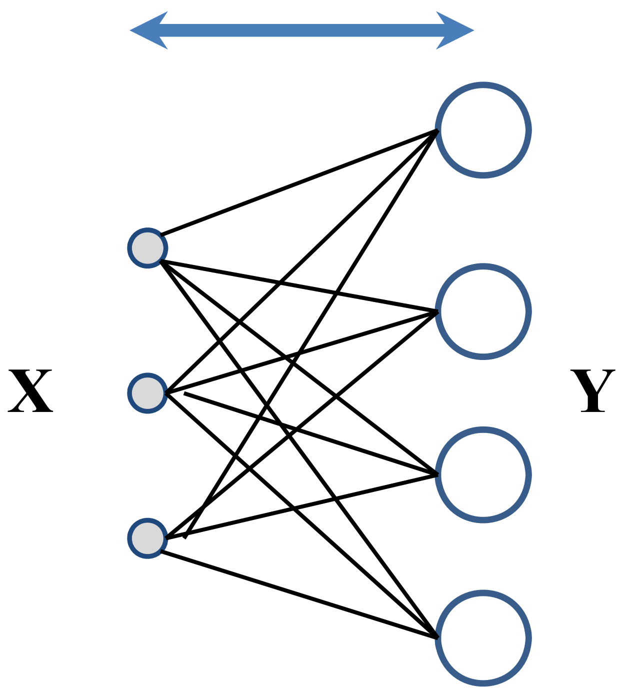


If $(\vec{x}, \vec{y})$ is a stable point, then nearby points like $(\vec{x}_0, \vec{y}_0)$ should converge.

$$\vec{y}_0 = \mathbf{W}\vec{x}_0 \text{ , next } \quad \vec{e} = \mathbf{W}^{\mathrm{T}}\vec{y}_0$$

How far is $\vec{e}$ from $\vec{x}_0$?

Concept of energy in BAM



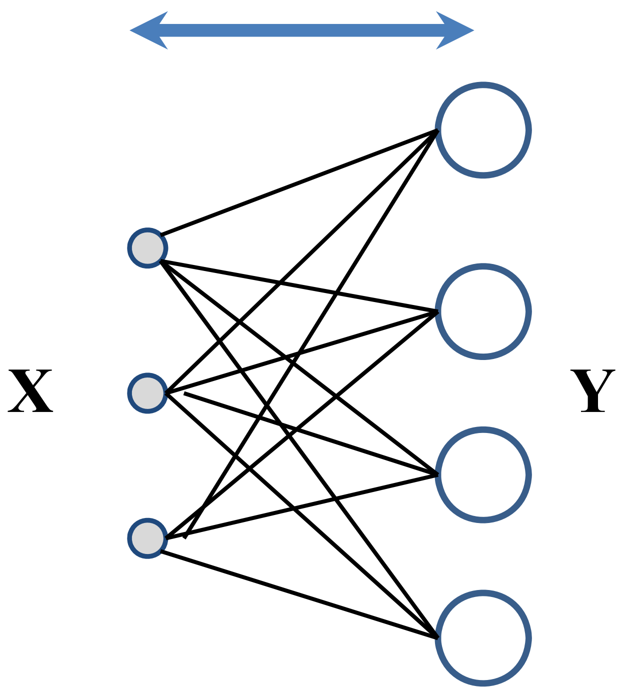
If $(\vec{x}, \vec{y})$ is a stable point, then nearby points like $(\vec{x}_0, \vec{y}_0)$ should converge.

$$\vec{y}_0 = \mathbf{W}\vec{x}_0 \text{ , next } \quad \vec{e} = \mathbf{W}^{\mathrm{T}}\vec{y}_0$$

How far is $\vec{e}$ from $\vec{x}_0$?

$$E = -\vec{x}_0^{\ \mathrm{T}}\vec{e} = -\vec{x}_0^{\ \mathrm{T}}\mathbf{W}^{\mathrm{T}}\vec{y}_0 = -\vec{y}_0^{\ \mathrm{T}}\mathbf{W}\vec{x}_0$$

Concept of energy in BAM



If $(\vec{x}, \vec{y})$ is a stable point, then nearby points like $(\vec{x}_0, \vec{y}_0)$ should converge.

$$\vec{y}_0 = \mathbf{W}\vec{x}_0 \text{ , next } \quad \vec{e} = \mathbf{W}^{\mathrm{T}}\vec{y}_0$$

How far is $\vec{e}$ from $\vec{x}_0$?

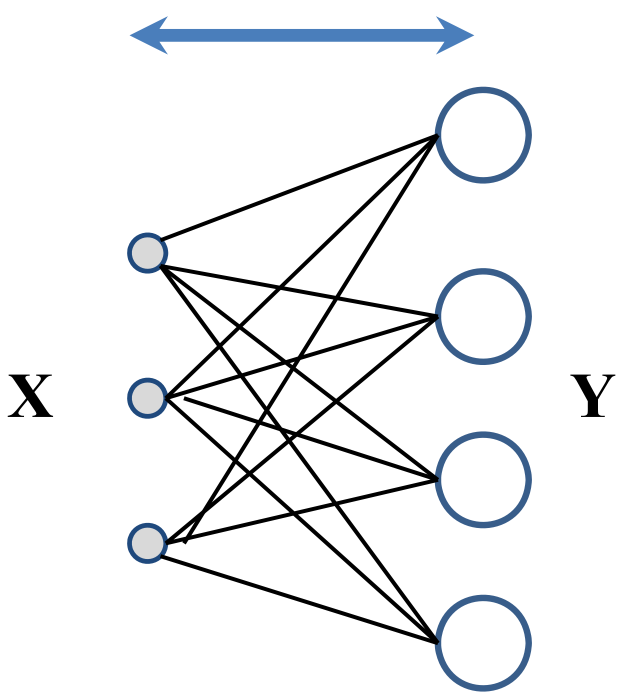
$$E = -\vec{x}_0{}^{\mathrm{T}}\vec{e} = -\vec{x}_0{}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\vec{y}_0 = -\vec{y}_0{}^{\mathrm{T}}\mathbf{W}\vec{x}_0$$

For the autoassociative BAM with $\mathbf{W}$, energy in the state $\vec{x}$:

$$E(\vec{x}, \vec{x}) = -\frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{W}\vec{x}$$

$$E(\vec{x}) = -\frac{1}{2}\sum_{i,j=1}^{n} w_{i,j} x_i x_j$$

Concept of energy in BAM



If $(\vec{x}, \vec{y})$ is a stable point, then nearby points like $(\vec{x}_0, \vec{y}_0)$ should converge.

$$\vec{y}_0 = \mathbf{W}\vec{x}_0 \text{ , next } \quad \vec{e} = \mathbf{W}^{\mathrm{T}}\vec{y}_0$$

How far is $\vec{e}$ from $\vec{x}_0$?

$$E = -\vec{x}_0^{\ \mathrm{T}}\vec{e} = -\vec{x}_0^{\ \mathrm{T}}\mathbf{W}^{\mathrm{T}}\vec{y}_0 = -\vec{y}_0^{\ \mathrm{T}}\mathbf{W}\vec{x}_0$$

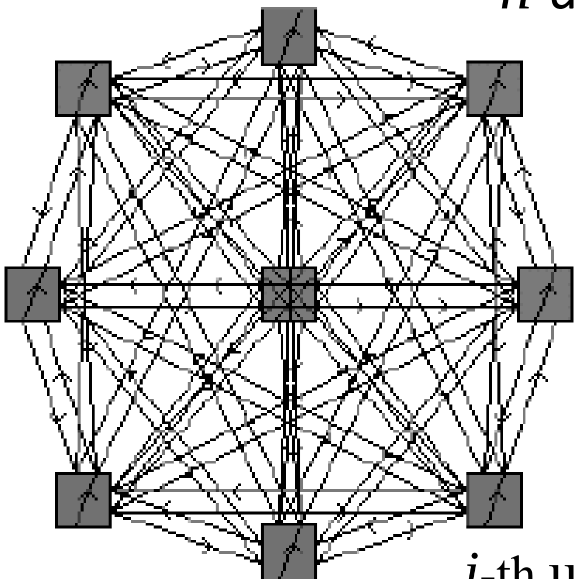
For the autoassociative BAM with $\mathbf{W}$, energy in the state $\vec{x}$:

$$E(\vec{x}, \vec{x}) = -\frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{W}\vec{x} + \vec{x}^{\mathrm{T}}\vec{\theta}$$

If bias is added

$$E(\vec{x}) = -\frac{1}{2}\sum_{i,j=1}^{n} w_{i,j}x_i x_j + \sum_{i=1}^{n}\theta_i x_i$$

Hopfield network



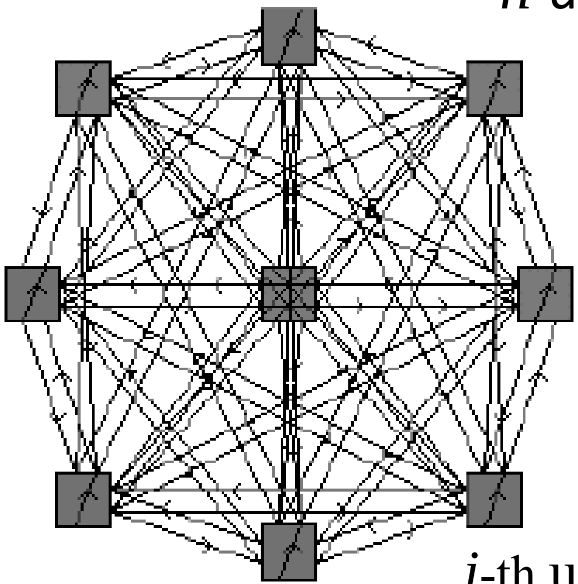
$\forall_i \, w_{i,i} = 0$ no self-connections

$$\vec{x}' = \mathrm{sgn}(\mathbf{W}\vec{x} + \vec{\theta})$$

$$E(state = \vec{x}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

Hopfield network

n units



j-th unit

$$\forall_i \, w_{i,i} = 0$$ no self-connections

$$\vec{x}\,' = \mathrm{sgn}(\mathbf{W}\vec{x} + \vec{\theta})$$

$$E(state = \vec{x}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

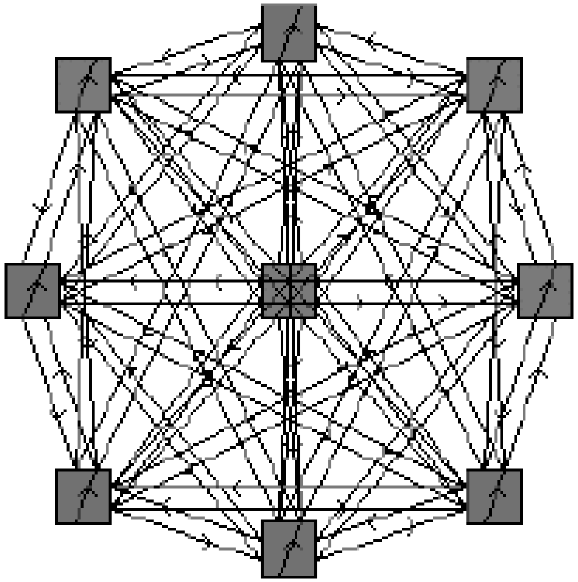
Iterative recall with asynchronous update

1) Apply input probe $\boldsymbol{\xi_p} = [\xi_{1,p}\,,\ \xi_{2,p}\,, \ldots,\ \xi_{n,p}]$, i.e. $x_j(0) = \xi_{j,p}$

2) Iterate *asynchronous* update until convergence (until the state $\boldsymbol{x}$ <u>remains unchanged</u>)

$$x_j(t+1) = \mathrm{sgn}\left(\sum_{i=1}^{n} w_{j,i}\, x_i(t)\right)$$ j=1,..,n is randomly selected one at a time

Hopfield network



$$\underset{i}{\forall} w_{i,i} = 0$$ no self-connections

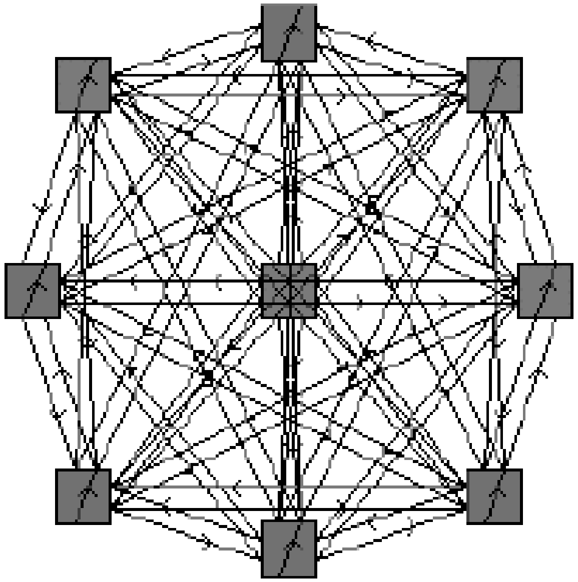
$$\vec{x}' = \mathrm{sgn}(\mathbf{W}\vec{x} + \vec{\theta})$$

$$E(state = \vec{x}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

Update occurs only when the state changes, so.....

$$\Delta E_{x_j \to x_j^*} = -\frac{1}{2}\left(\sum_{i}^{n} w_{i,j} x_i x_j^* - \sum_{i}^{n} w_{i,j} x_i x_j\right) = -\frac{1}{2}\left(x_j^* - x_j\right)\sum_{i}^{n} w_{i,j} x_i \le 0$$

Hopfield network



$$\forall_i \, w_{i,i} = 0$$ no self-connections

$$\vec{x}\,' = \mathrm{sgn}(\mathbf{W}\vec{x} + \vec{\theta})$$

$$E(state = \vec{x}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

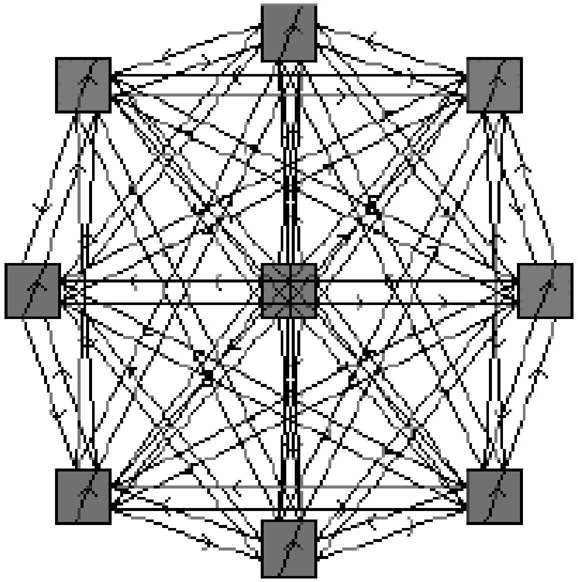
W should be symmetric with diag=0 for convergence

Update occurs only when the state changes, so.....

$$\Delta E_{x_j \to x_j^*} = -\frac{1}{2}\left(\sum_i^n w_{i,j} x_i x_j^* - \sum_i^n w_{i,j} x_i x_j\right) = -\frac{1}{2}\left(x_j^* - x_j\right)\sum_i^n w_{i,j} x_i \le 0$$

towards lower energy – convergence!

Hopfield network



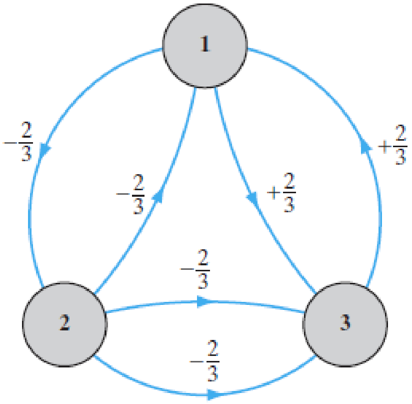
$$\forall_i \, w_{i,i} = 0$$ no self-connections

$$\vec{x}\,' = \operatorname{sgn}(\mathbf{W}\vec{x} + \vec{\theta})$$

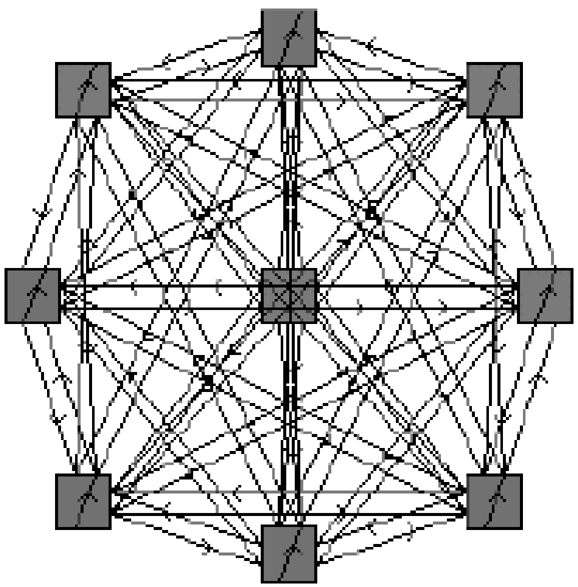
$$E(state = \vec{x}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

$\mathbf{W}$ should be symmetric with diag=0 for convergence

How many states are candidates for fixed states?



Hopfield network

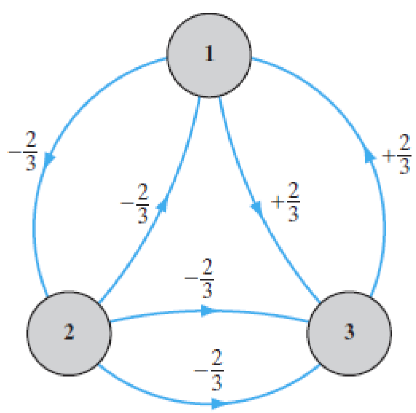


$$\forall_i \, w_{i,i} = 0$$ no self-connections

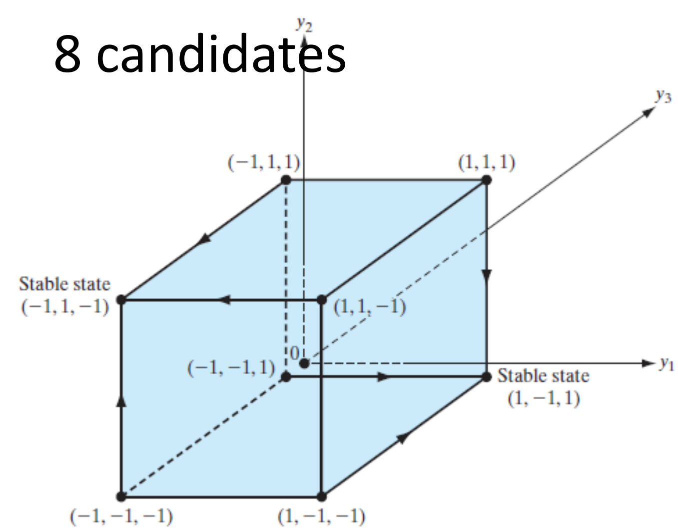
$$\vec{x}\,' = \mathrm{sgn}(\mathbf{W}\vec{x} + \vec{\theta})$$

$$E(state = \vec{x}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

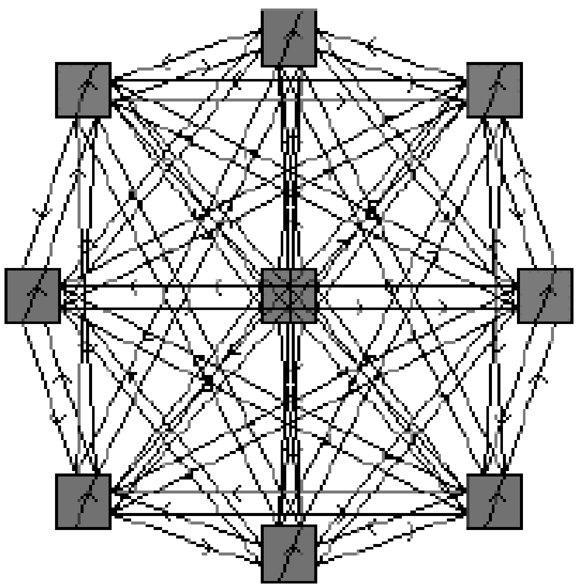
$\mathbf{W}$ should be symmetric with diag=0 for convergence



8 candidates



Hopfield network

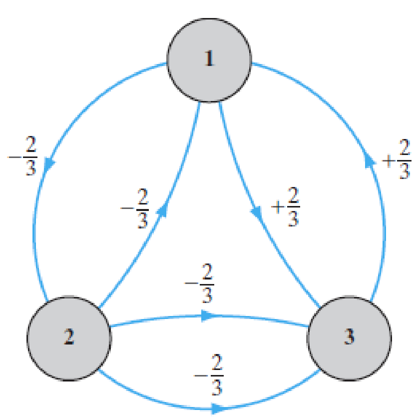


$$\forall_i \, w_{i,i} = 0$$ no self-connections

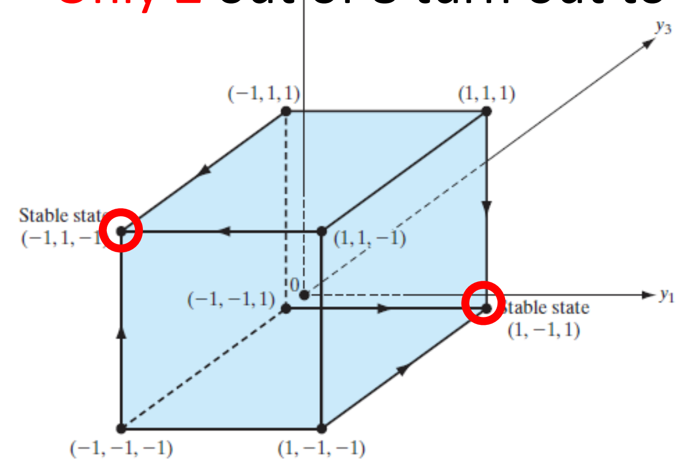
$$\vec{x}\,' = \mathrm{sgn}(\mathbf{W}\vec{x} + \vec{\theta})$$

$$E(state = \vec{x}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$$

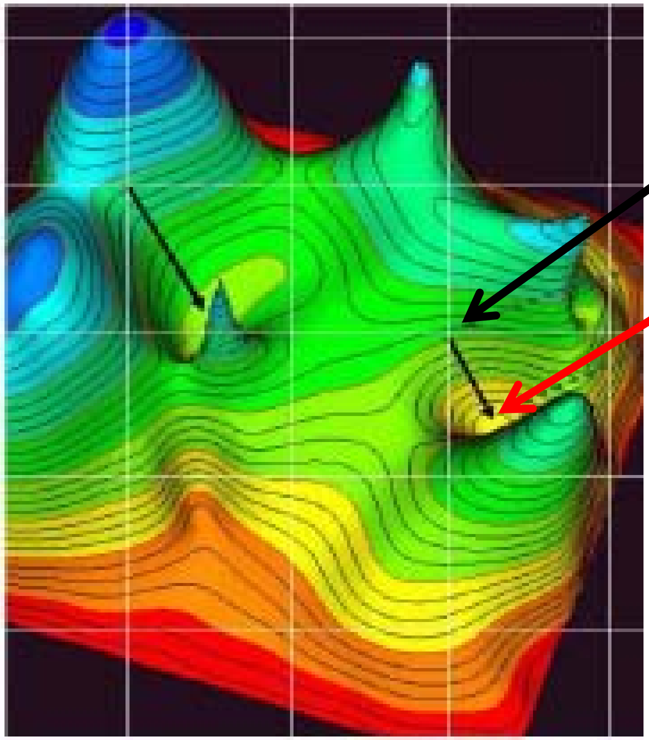
$\mathbf{W}$ should be symmetric with diag=0 for convergence



Only **2** out of 8 turn out to be stable!



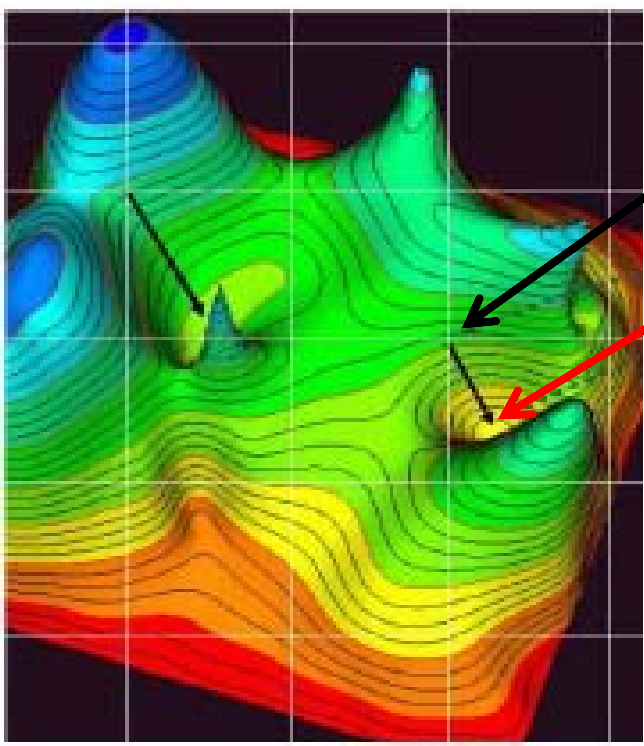
Attractor dynamics



Memory cue
(within the basin of attractor)

Memory state
(local energy minimum,
stable point, attractor)

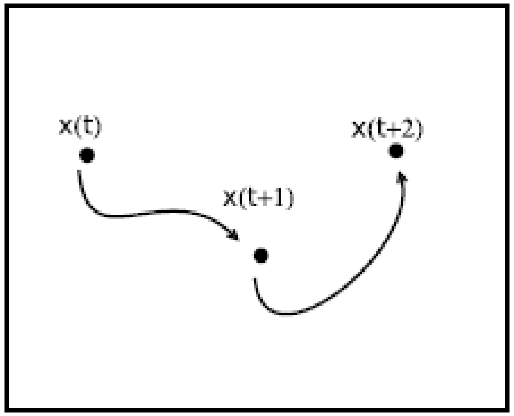
Attractor dynamics



Memory cue
(within the basin of attractor)

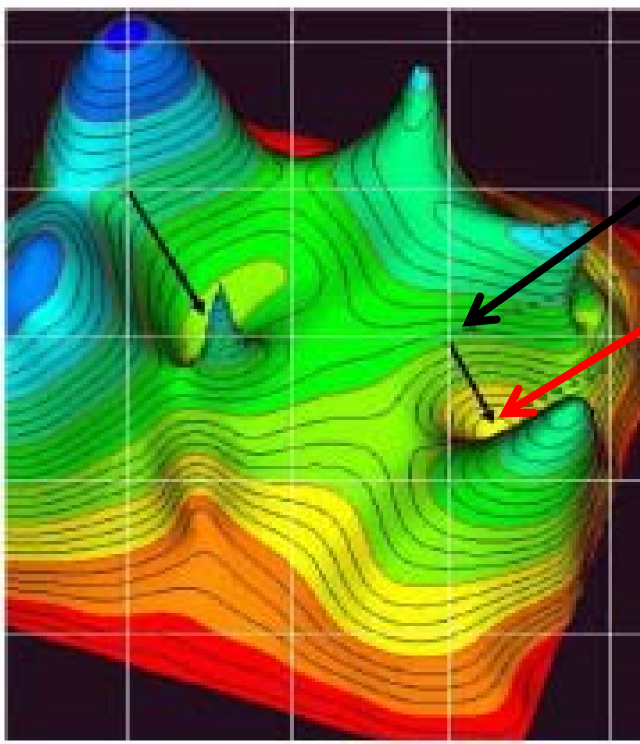
Memory state
(local energy minimum,
stable point, **fixed-point attractor**)

Dynamics travelling in the energy landscape and attracted to the <u>energy minimum</u>



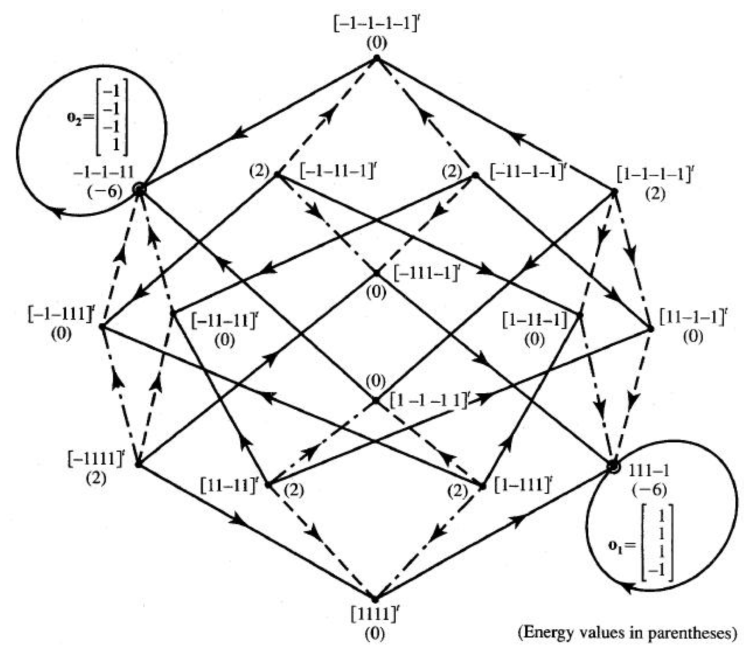
In *discrete* Hopfield network,
the energy landscape is discrete!

Attractor dynamics



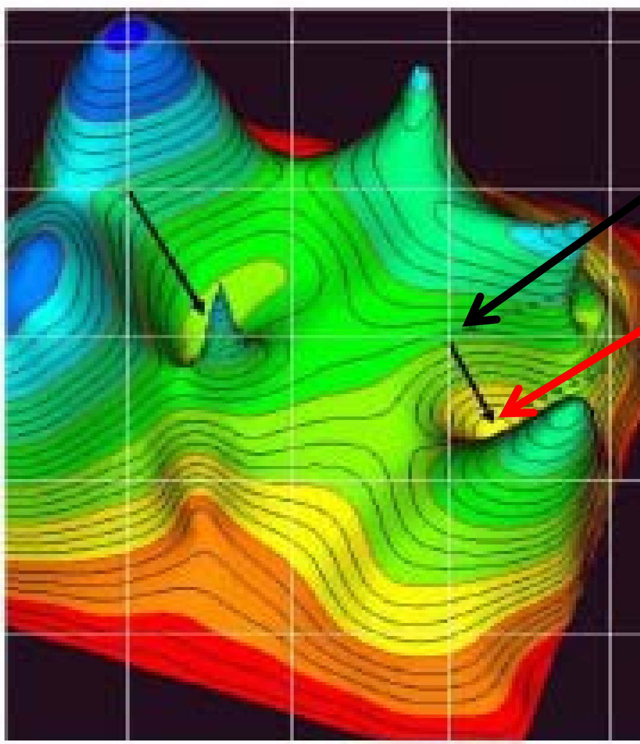
Memory cue
(within the basin of attractor)

Memory state
(local energy minimum,
stable point, **fixed-point attractor**)



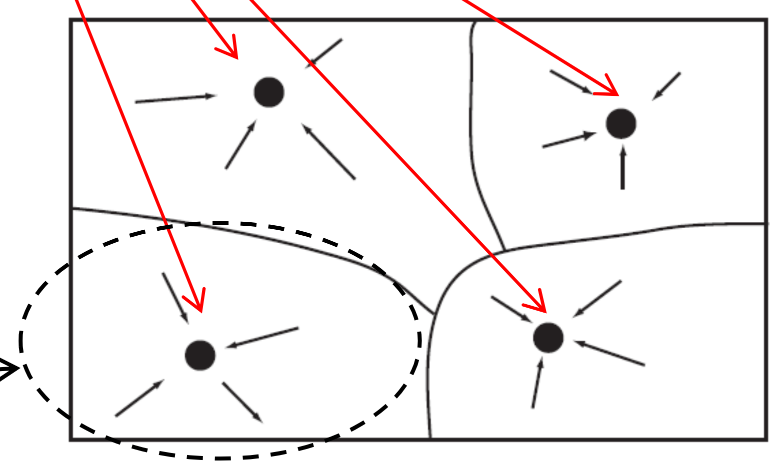
In *discrete* Hopfield network,
the energy landscape is discrete!

Attractor dynamics



Memory cue
(within the basin of attractor)

Memory state
(local energy minimum,
stable point, **fixed-point attractor**)



Around each fixed point (attractor), there is
a ***basin of attraction***

How do we learn memories for storage?

Hopfield network as a content addressable memory

A set of memory patterns $\{\boldsymbol{\xi_1}, \boldsymbol{\xi_2}, \ldots \boldsymbol{\xi_M}\}$ to be learnt.

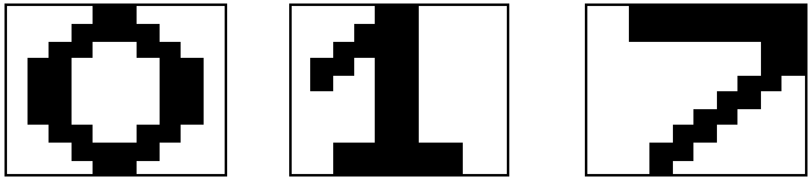
$$\boldsymbol{\xi_k} = [\xi_{k,1}, \xi_{k,2}, \ldots \xi_{k,n}], \quad k=1,.., M$$

Outer product rule (Hebbian-like learning) is used to compute **W**:

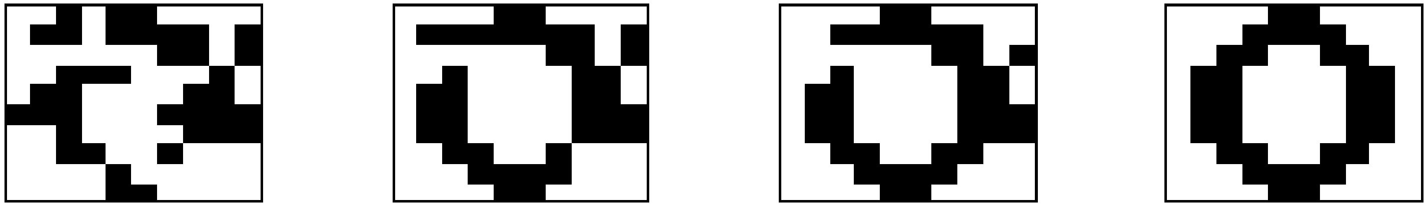
$$w_{j,i} = \begin{cases} \frac{1}{n} \sum_{k=1}^{M} \xi_{k,j} \cdot \xi_{k,i}, & j \neq i \\ 0, & j = i \end{cases}$$

Pattern storage and recall example

- The following patterns ξ^1, ξ^2, ξ^3 were stored in the weight matrix W:

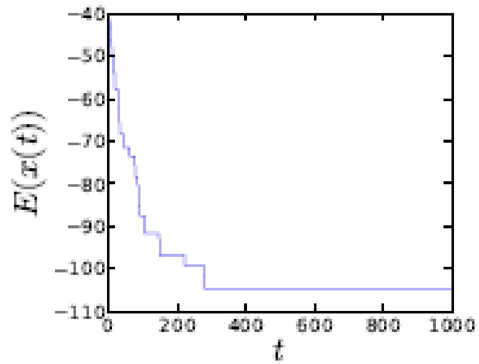


- Four snapshots of the state evolution $x(t)$:



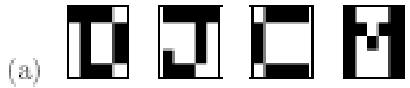
$t = 0$ $t = 50$ $t = 100$ $t = 300$

- Evolution of the energy $E(x(t))$:

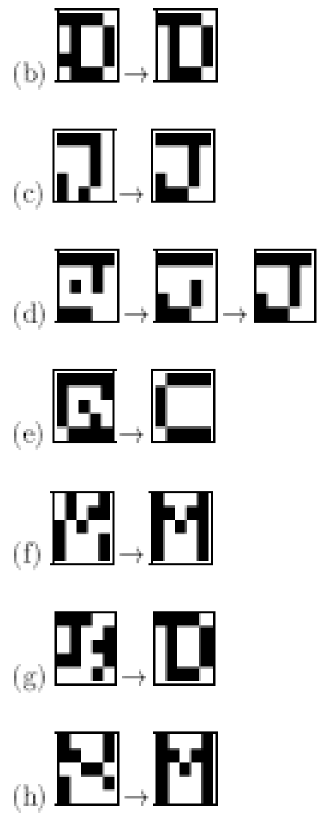


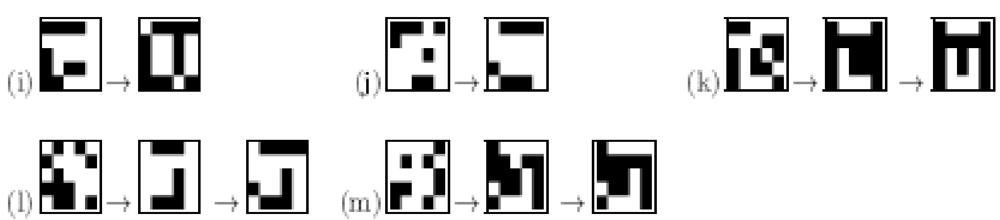
adapted from L. Busing (TU Graz)

Pattern storage and recall example



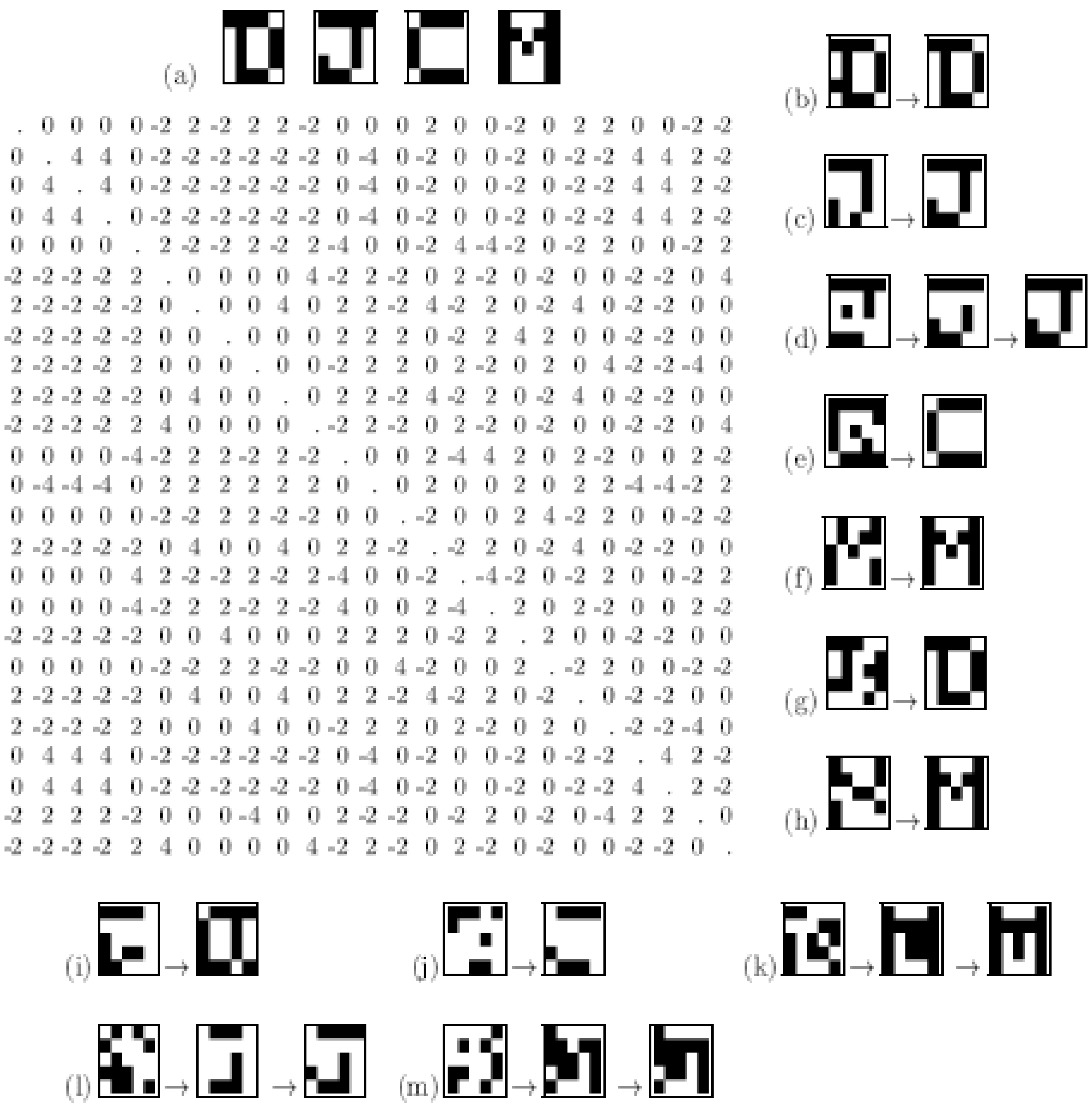
.	0	0	0	0	-2	2	-2	2	2	-2	0	0	0	2	0	0	-2	0	2	2	0	0	-2	-2
0	.	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	4	.	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	4	4	.	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	0	0	0	.	2	-2	-2	2	-2	2	-4	0	0	-2	4	-4	-2	0	-2	2	0	0	-2	2
-2	-2	-2	-2	2	.	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
2	-2	-2	-2	-2	0	.	0	0	4	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	-2	0	0	.	0	0	0	2	2	2	0	-2	2	4	2	0	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	.	0	0	-2	2	2	0	2	-2	0	2	0	4	-2	-2	-4	0
2	-2	-2	-2	-2	0	4	0	0	.	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	2	4	0	0	0	0	.	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
0	0	0	0	-4	-2	2	2	-2	2	-2	.	0	0	2	-4	4	2	0	2	-2	0	0	2	-2
0	-4	-4	-4	0	2	2	2	2	2	2	0	.	0	2	0	0	2	0	2	2	-4	-4	-2	2
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	.	-2	0	0	2	4	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	.	-2	2	0	-2	4	0	-2	-2	0	0
0	0	0	0	4	2	-2	-2	2	-2	2	-4	0	0	-2	.	-4	-2	0	-2	2	0	0	-2	2
0	0	0	0	-4	-2	2	2	-2	2	-2	4	0	0	2	-4	.	2	0	2	-2	0	0	2	-2
-2	-2	-2	-2	-2	0	0	4	0	0	0	2	2	2	0	-2	2	.	2	0	0	-2	-2	0	0
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	4	-2	0	0	2	.	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	4	-2	2	0	-2	.	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	4	0	0	-2	2	2	0	2	-2	0	2	0	.	-2	-2	-4	0
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	.	4	2	-2
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	.	2	-2
-2	2	2	2	-2	0	0	0	-4	0	0	2	-2	-2	0	-2	2	0	-2	0	-4	2	2	.	0
-2	-2	-2	-2	2	4	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	.





adapted from McKay

Pattern storage and recall example



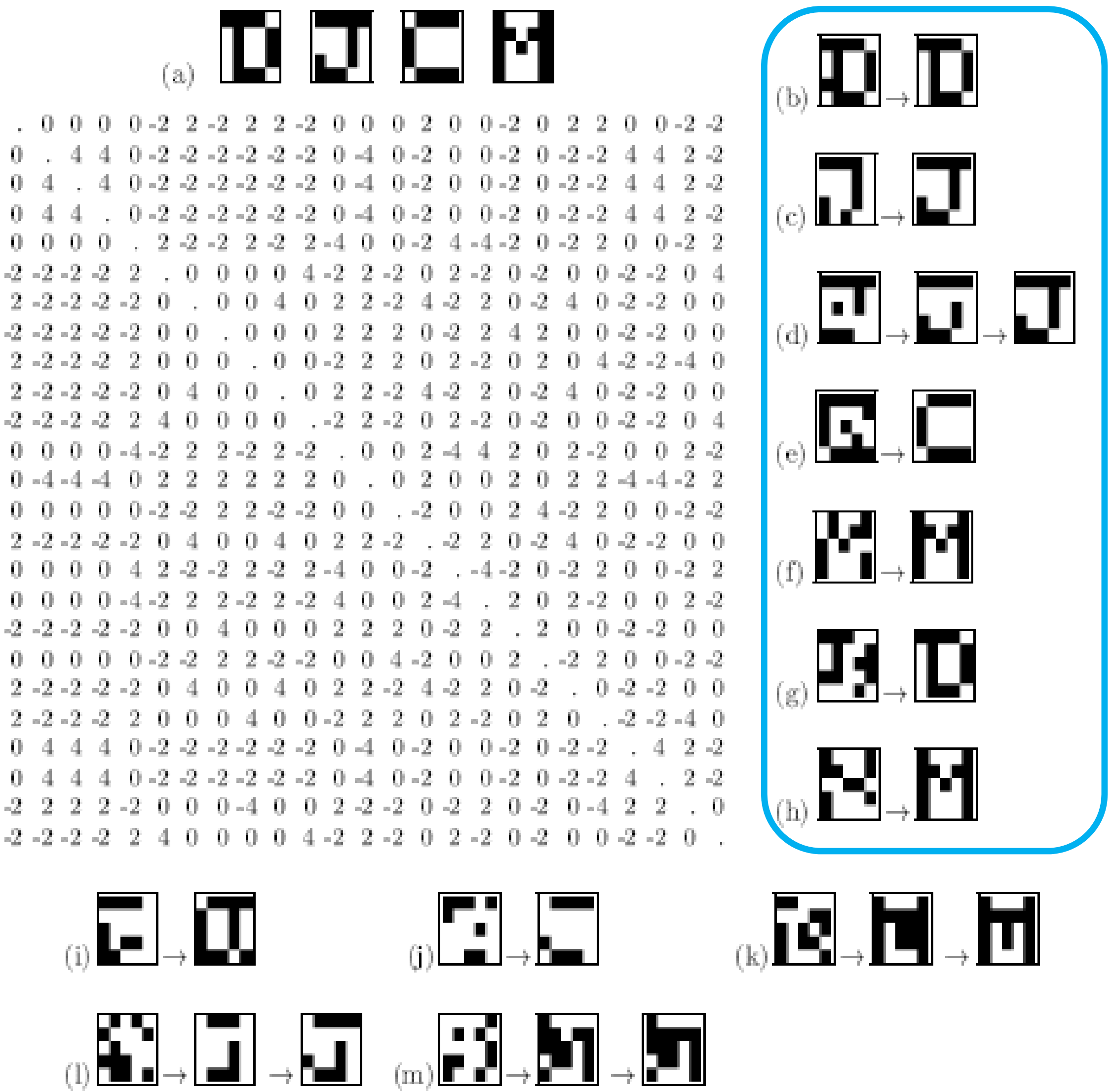
.	0	0	0	0	-2	2	-2	2	2	-2	0	0	0	2	0	0	-2	0	2	2	0	0	-2	-2
0	.	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	4	.	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	4	4	.	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	0	0	0	.	2	-2	-2	2	-2	2	-4	0	0	-2	4	-4	-2	0	-2	2	0	0	-2	2
-2	-2	-2	-2	2	.	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
2	-2	-2	-2	-2	0	.	0	0	4	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	-2	0	0	.	0	0	0	2	2	2	0	-2	2	4	2	0	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	.	0	0	-2	2	2	0	2	-2	0	2	0	4	-2	-2	-4	0
2	-2	-2	-2	-2	0	4	0	0	.	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	2	4	0	0	0	0	.	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
0	0	0	0	-4	-2	2	2	-2	2	-2	.	0	0	2	-4	4	2	0	2	-2	0	0	2	-2
0	-4	-4	-4	0	2	2	2	2	2	2	0	.	0	2	0	0	2	0	2	2	-4	-4	-2	2
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	.	-2	0	0	2	4	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	.	-2	2	0	-2	4	0	-2	-2	0	0
0	0	0	0	4	2	-2	-2	2	-2	2	-4	0	0	-2	.	-4	-2	0	-2	2	0	0	-2	2
0	0	0	0	-4	-2	2	2	-2	2	-2	4	0	0	2	-4	.	2	0	2	-2	0	0	2	-2
-2	-2	-2	-2	-2	0	0	4	0	0	0	2	2	2	0	-2	2	.	2	0	0	-2	-2	0	0
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	4	-2	0	0	2	.	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	4	-2	2	0	-2	.	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	4	0	0	-2	2	2	0	2	-2	0	2	0	.	-2	-2	-4	0
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	.	4	2	-2
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	.	2	-2
-2	2	2	2	-2	0	0	0	-4	0	0	2	-2	-2	0	-2	2	0	-2	0	-4	2	2	.	0
-2	-2	-2	-2	2	4	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	.

Common problems

1. Corruption of individual bits.
2. Lack of encoded memory or a very small basin of attraction.
3. Appearance of spurious additional memories.

adapted from McKay

Pattern storage and recall example

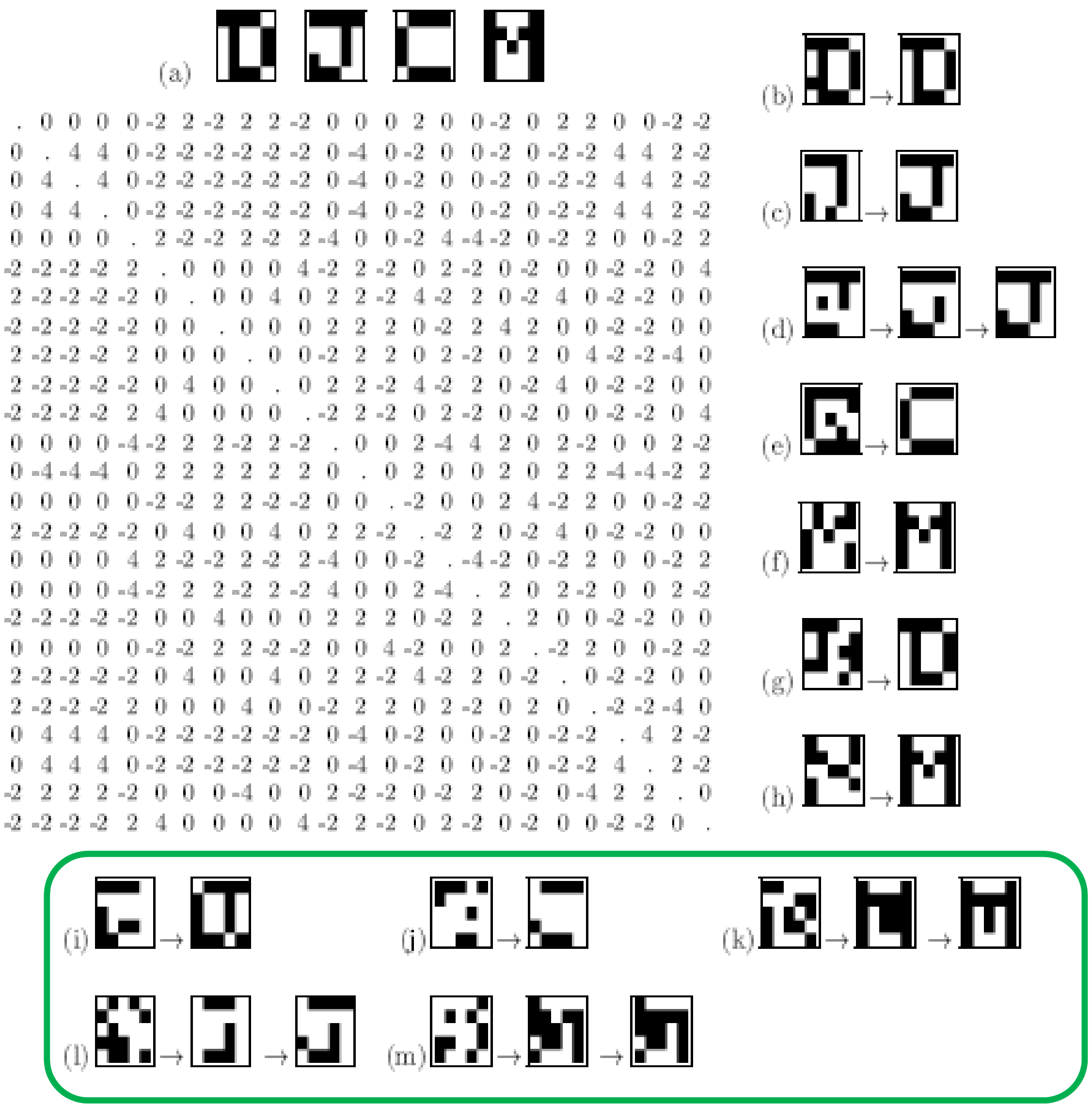


Common problems

1. Corruption of individual bits.
2. Lack of encoded memory or a very small basin of attraction.
3. Appearance of spurious additional memories.

adapted from McKay

Pattern storage and recall example



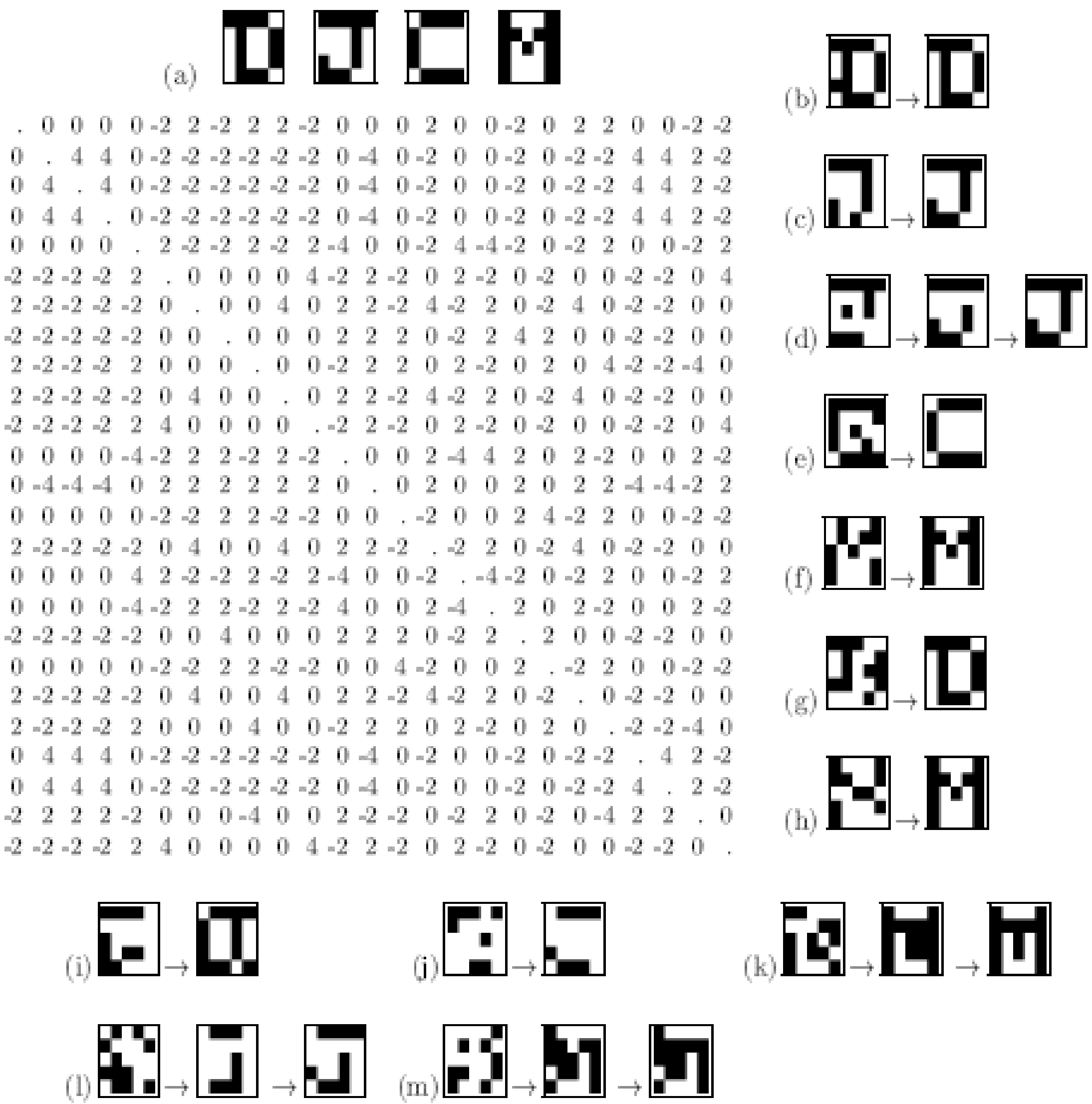
Common problems

1. Corruption of individual bits.
2. Lack of encoded memory or a very small basin of attraction.
3. Appearance of spurious additional memories.

Spurious states often arise out of degenerate eigenvectors.

adapted from McKay

Pattern storage and recall example



Common problems

1. Corruption of individual bits.
2. Lack of encoded memory or a very small basin of attraction.
3. Appearance of spurious additional memories.

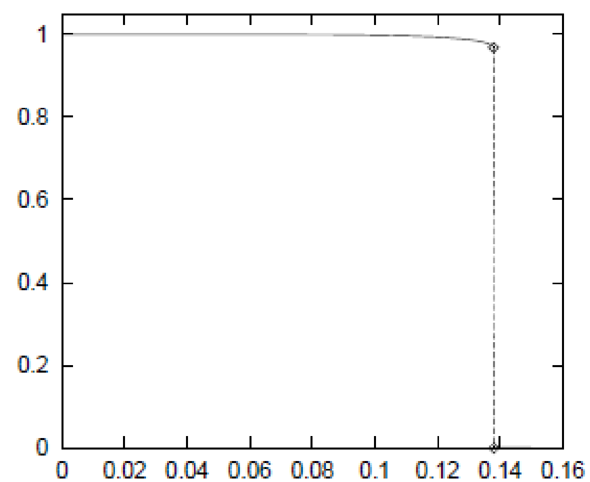
Generally, Hopfield network is robust to noise, data corruption and “brain damage” (zeroed subset of weights).

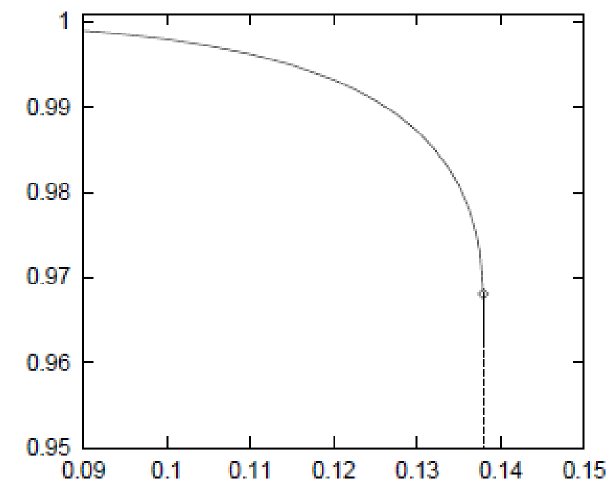
adapted from McKay

Memory capacity

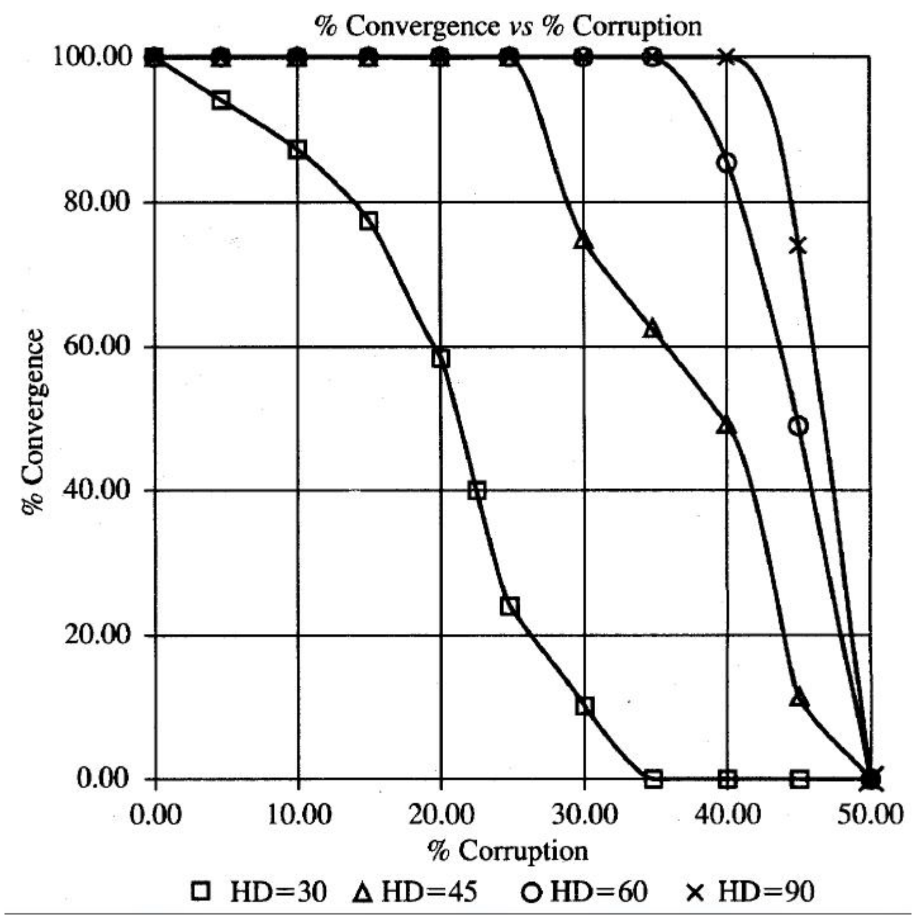
- Cross-talk between memory patterns is key to limited capacity
- Memory capacity is usually tested on independent random patterns
  - Hopfield network can store roughly M<=0.138 n of such random patterns (sharp discontinuity)
  - for large M/n, unstable bits may unfold into an avalanche effect
  - for sparse patterns in the order of n*log(n)
- To guarantee stability of all patterns with high probability, we must ensure

$$M \leq \frac{n}{4 \ln n}$$





Catastrophic forgetting effect



Convergence rate is defined based on the convergence criterion, often expressed as the upper bound on *Hamming distance*.

Network properties are not robust for synchronous updates.

Also, problems for continuous networks.

$$a_i = \sum_j w_{ij} x_j \qquad x_i = \tanh(a_i).$$

Better behaviour for continuous continuous –time Hopfield network

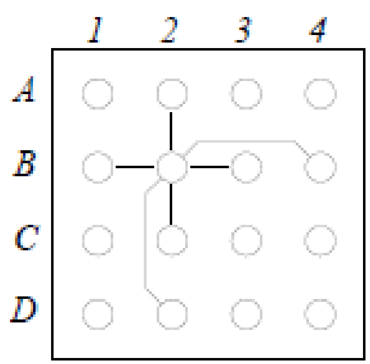
$$a_i(t) = \sum_j w_{ij} x_j(t). \qquad \frac{\mathrm{d}}{\mathrm{d}t} x_i(t) = -\frac{1}{\tau}(x_i(t) - f(a_i)),$$

Hopfield networks for optimisation problems

- Hopfield network’s dynamics minimises an energy function
- Some optimisation problems could be mapped to the quadratic energy function (particularly constrain satisfaction problems(CSPs))

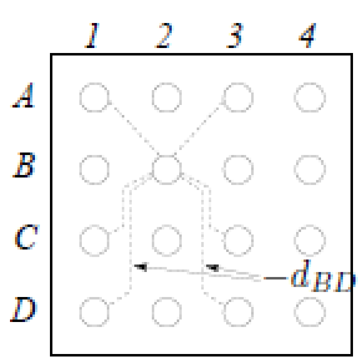
Hopfield networks for optimisation problems

- Hopfield network's dynamics minimises an energy function
- Some optimisation problems could be mapped to the quadratic energy function (particularly constrain satisfaction problems(CSPs))
- Travelling salesman problem (TSP) as a classic CSP problem



validity

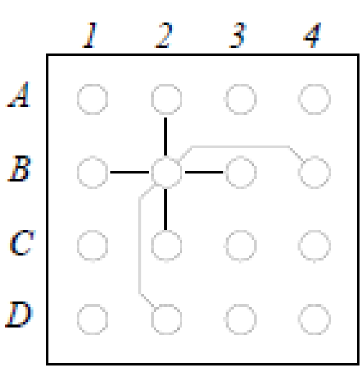
distances



$$E = \underbrace{\frac{1}{2}\sum_{i,j,k}^{n} d_{ij} x_{ik} x_{j,k+1}}_{\text{sum of distances}} + \underbrace{\frac{\gamma}{2}\Big(\sum_{j=1}^{n}\big(\sum_{i=1}^{n} x_{ij} - 1\big)^2 + \sum_{i=1}^{n}\big(\sum_{j=1}^{n} x_{ij} - 1\big)^2\Big)}_{\text{validity: single 1s in each column and row}}$$

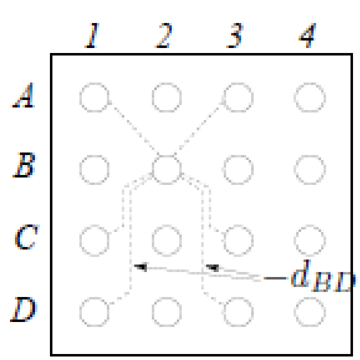
Hopfield networks for optimisation problems

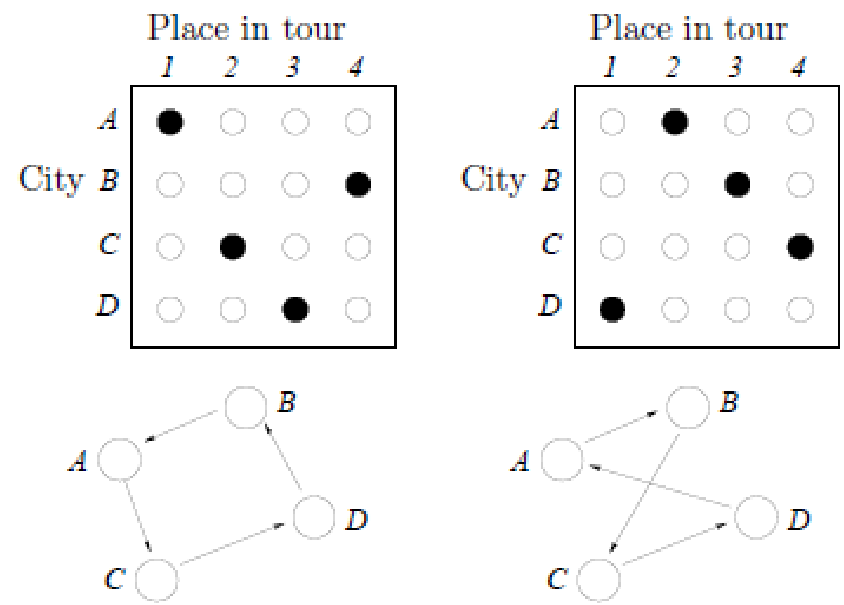
- Hopfield network's dynamics minimises an energy function
- Some optimisation problems could be mapped to the quadratic energy function (particularly constrain satisfaction problems(CSPs))
- Travelling salesman problem (TSP) as a classic CSP problem



validity

distances





Hopfield networks

In summary

- Hopfield network is a nice model for memory with biological features including Hebbian learning
- It is a very simple, stable and mathematically tractable model
- It has limited capacity and assumes near orthogonal patterns
- It does not allow for storing time series
- The attractor dynamics is limited to fixed points