Neural Nets and Symbolic Reasoning

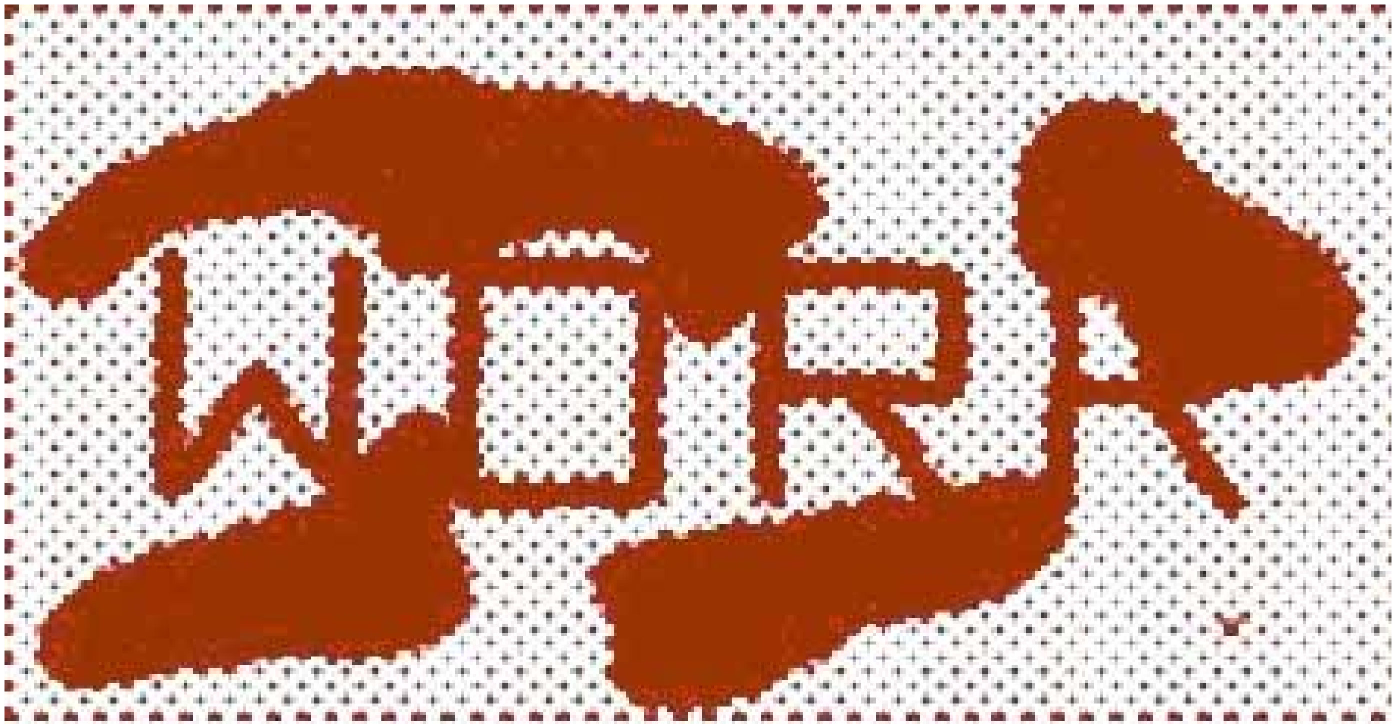
Hopfield Networks

0	−2	−2	2	−2	4	0	2	−2	2	0	2	0	0	0	−2
−2	0	4	0	0	−2	−2	0	0	0	2	0	−2	−2	−2	0
−2	4	0	0	0	−2	−2	0	0	0	2	0	−2	−2	−2	0
2	0	0	0	0	2	2	0	0	0	2	0	−2	2	2	0
−2	0	0	0	0	−2	2	0	4	−4	−2	0	−2	2	2	0
4	−2	−2	2	−2	0	0	2	−2	2	0	2	0	0	0	−2
0	−2	−2	2	2	0	0	−2	2	−2	0	−2	0	4	4	2
2	0	0	0	0	2	−2	0	0	0	−2	4	−2	−2	−2	−4
−2	0	0	0	4	−2	2	0	0	−4	−2	0	−2	2	2	0
2	0	0	0	−4	2	−2	0	−4	0	2	0	2	−2	−2	0
0	2	2	2	−2	0	0	−2	−2	2	0	−2	0	0	0	2
2	0	0	0	0	2	−2	4	0	0	−2	0	−2	−2	−2	−4
0	−2	−2	−2	−2	0	0	−2	−2	2	0	−2	0	0	0	2
0	−2	−2	2	2	0	4	−2	2	−2	0	−2	0	0	4	2
0	−2	−2	2	2	0	4	−2	2	−2	0	−2	0	4	0	2
−2	0	0	0	0	−2	2	−4	0	0	2	−4	2	2	2	0

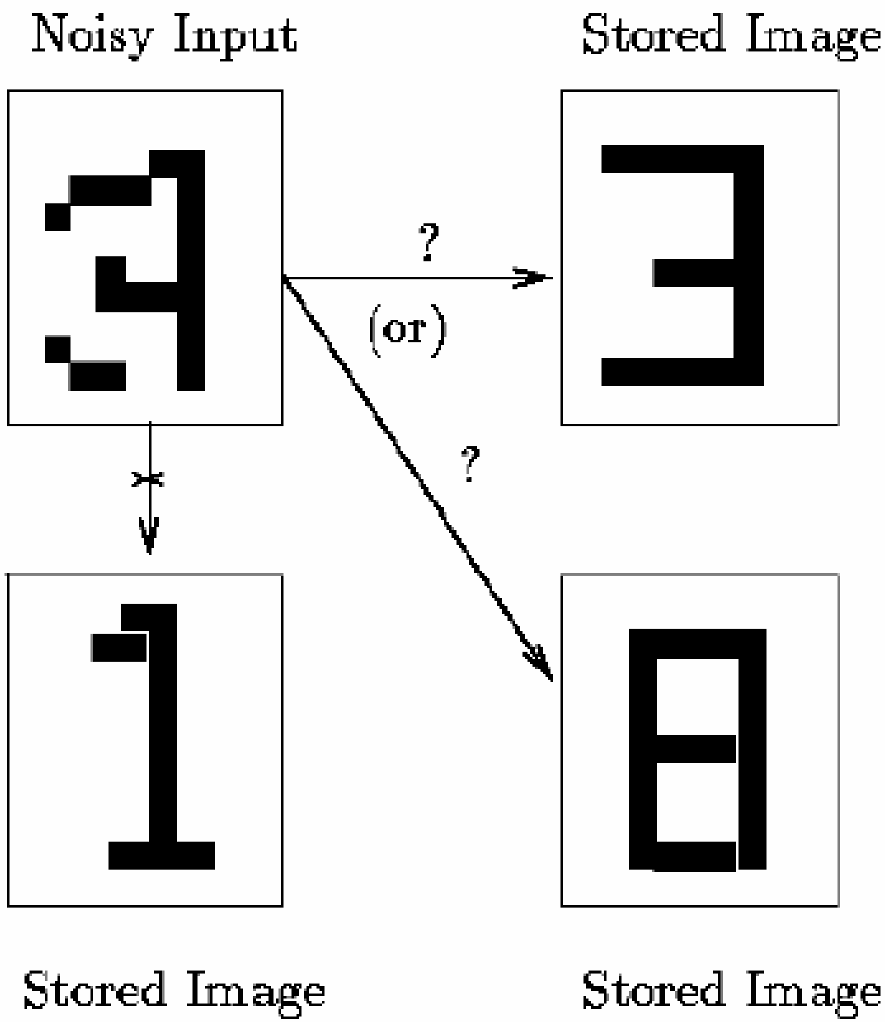
Outline

- The idea of pattern completion
- The fast dynamics of Hopfield networks
- Learning with Hopfield networks
- Emerging properties of Hopfield networks
- Conclusions

1 The idea of pattern completion

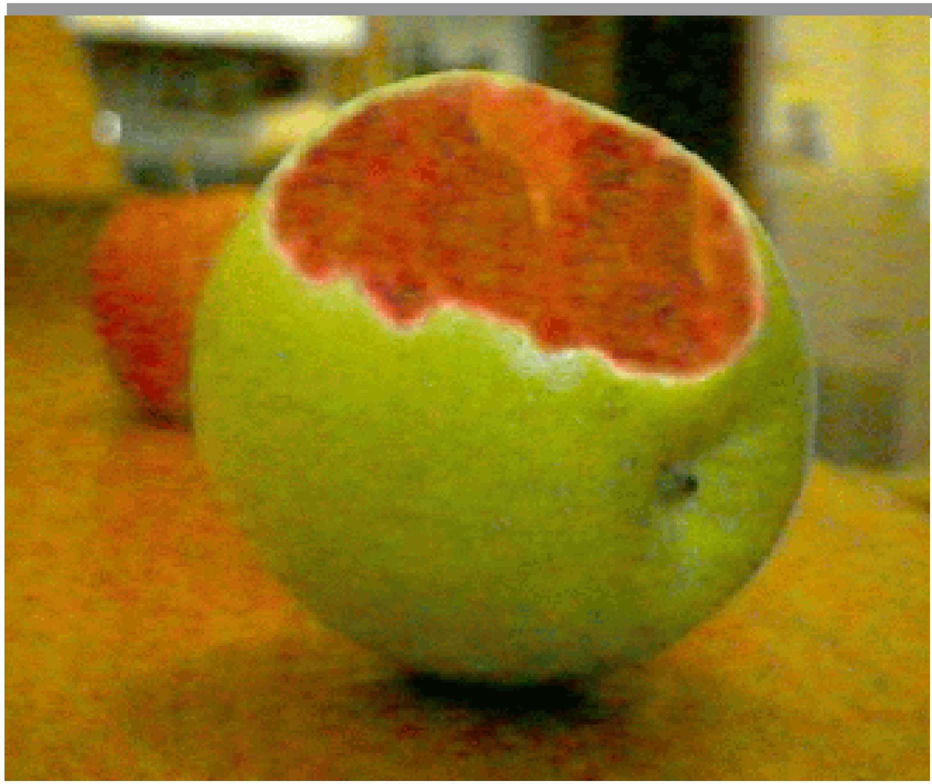


- Noisy or underspecified input
- Mechanism of pattern completion (using *stored* images)
- Stored patterns are addressable by *content*, not *pointers* (as in traditional computer memories)



a.	*a fast car*	[one that moves quickly]
b.	*a fast typist*	[a person that performs the act of typing quickly]
c.	*a fast book*	[one that can be read in a short time]
d.	*a fast driver*	[one who drives quickly]

Example from semantics: *A red apple*



A red apple?

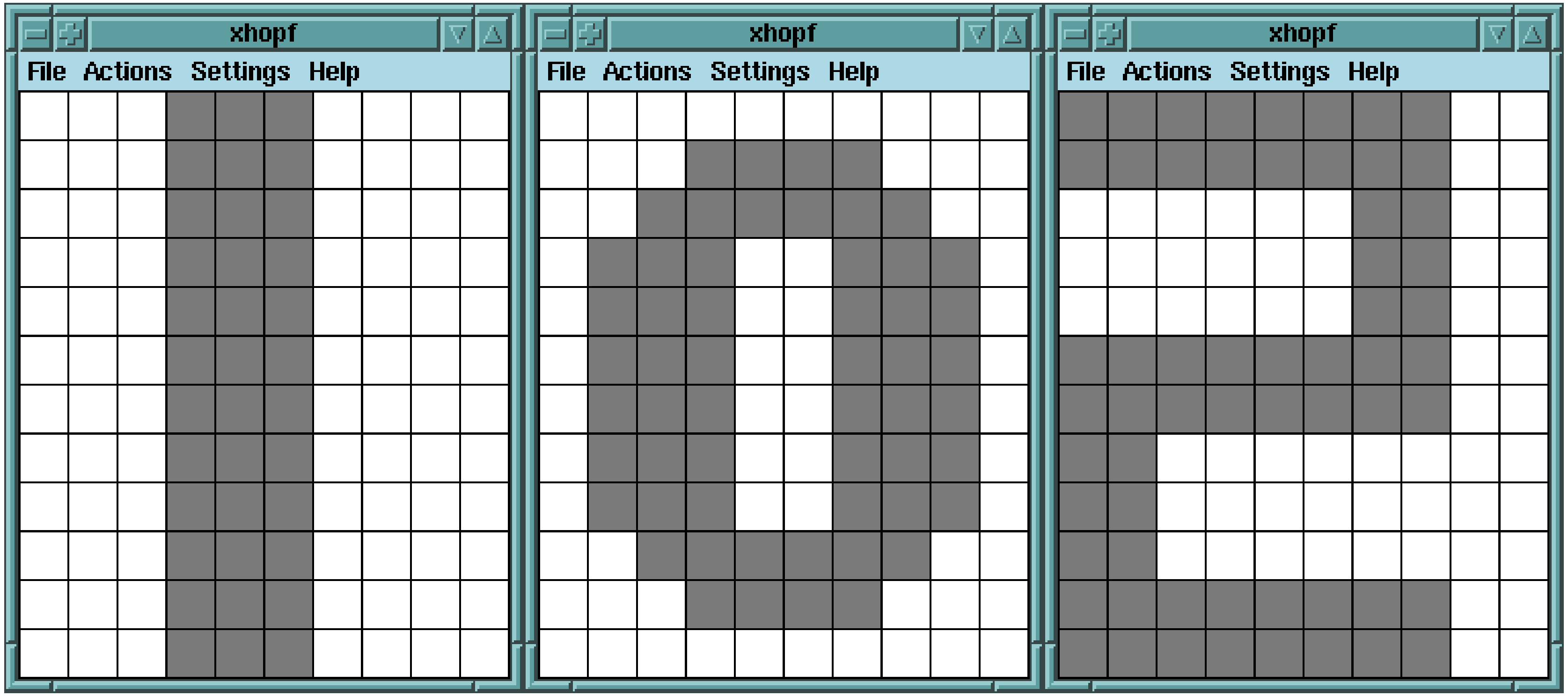
What color is an apple?

Q_1 What color is its peel?

Q_2 What color is its pulp?

a.	a red apple	[red peel]
b.	a sweet apple	[sweet pulp]
c.	a reddish grapefruit	[reddish pulp]
d.	a white room/ a white house	[inside/outside]

2 The fast dynamics of Hopfield networks



A neural network N can be defined as a quadruple <S,F,W,G>:

- S: Space of all possible states
- W: Set of possible configurations. $w \in W$ describes for each pair i,j of "neurons" the connection w_{ij} between i and j
- F: Set of activation functions. For a given configuration $w \in W$: the function $f_w \in F$ describes how the neuron activities spread through that network (fast dynamics)
- G: Set of learning functions (slow dynamics)

Discrete dynamic systems

$\vec{s}(t+1) = g(\vec{s}(t))$ ***s***(t) is a vector of the space of states S

Continuous dynamic systems

$\frac{d}{dt}\vec{s}(t) = g(\vec{s}(t))$ the function *g* describes the fast dynamics

Dynamical Systems + Neural Networks =Neurodynamics

- Tools from dynamical systems, statistics and statistical physics can be used. Very rich field.
- The triple <S,F,W> corresponds to the *fast* neurodynamics

What can recurrent networks do?

- Associative memories
- Pattern completion
- Noise removal
- General networks (can implement everything feedforward networks can do, and even emulate Turing machines)
- Spatio-temporal pattern recognition
- Dynamic reconstruction

J.J.Hopfield (1982), "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences* 79, 2554-2558.

An autoassociative, fully connected network with binary neurons, asynchronous updates and a Hebbian learning rule. The “classic” recurrent network

> Computational properties of use to biological organisms or to the construction of computers can emerge as collective properties of systems having a large number of simple equivalent components (or neurons). The physical meaning of content-addressable memory is described by an appropriate phase space flow of the state of the system. …

Concise description of the fast dynamics

Let the interval [-1,+1] be the *working range* of each neuron

+1: maximal firing rate

0: resting

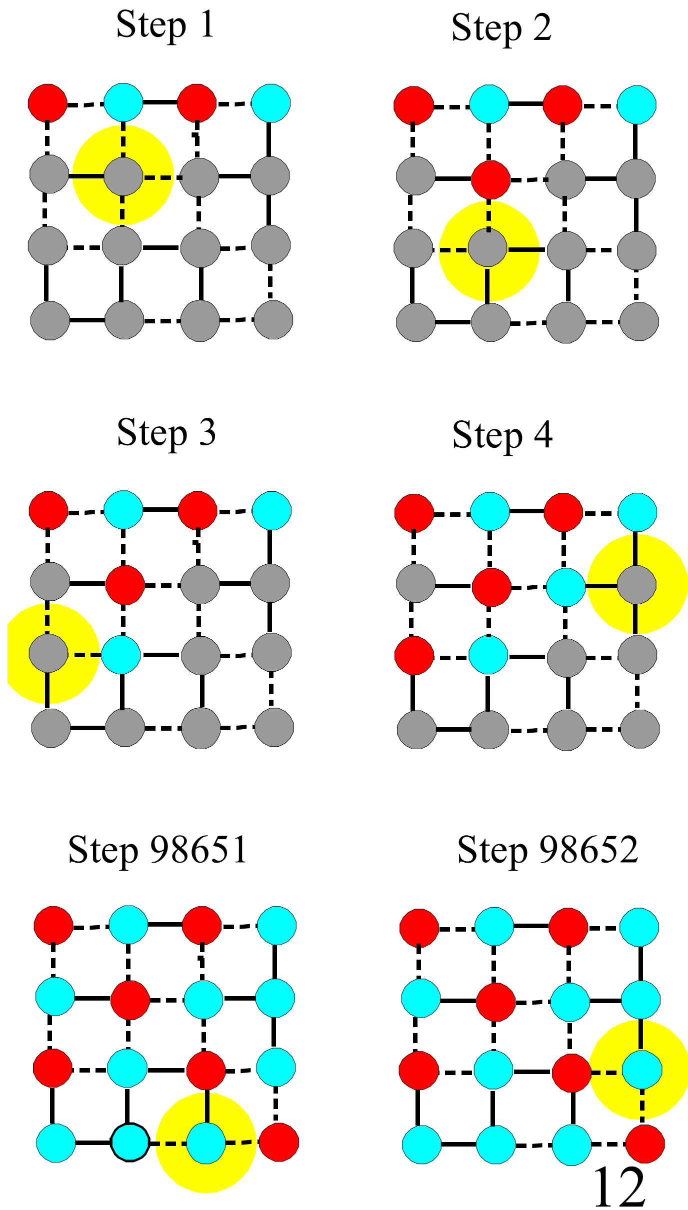
-1 : minimal firing rate)

$S = [-1, 1]^n$

$w_{ij} = w_{ji}$, $w_{ii} = 0$

ASYNCHRONOUS UPDATING:

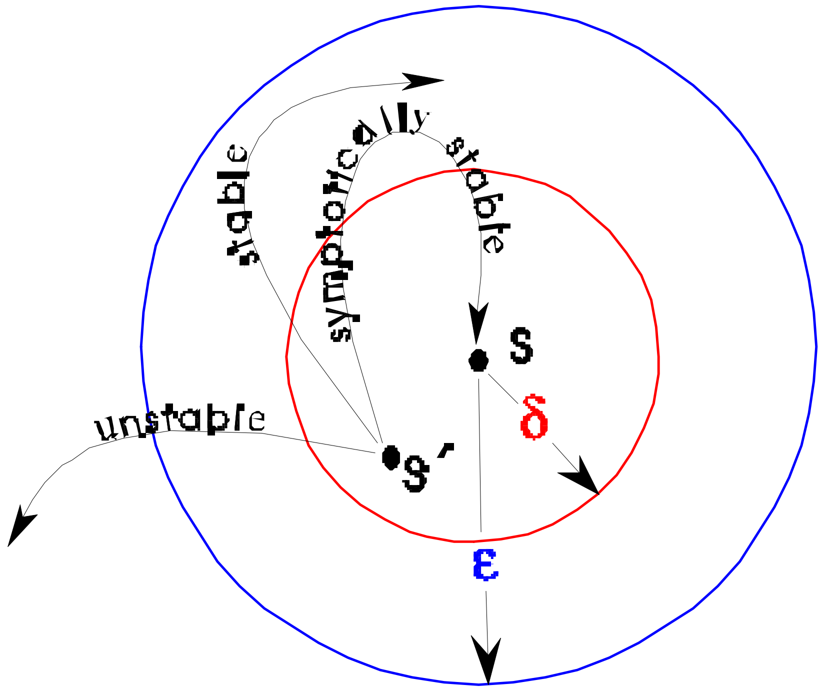
$$s_i(t+1) = \begin{cases} \theta\left(\Sigma_j\, w_{ij} \cdot s_j(t)\right), & \text{if } i = \text{rand}(1,n) \\ s_i(t), & \text{otherwise} \end{cases}$$



A state s in S is called a resonance of a dynamic system [S, f] iff

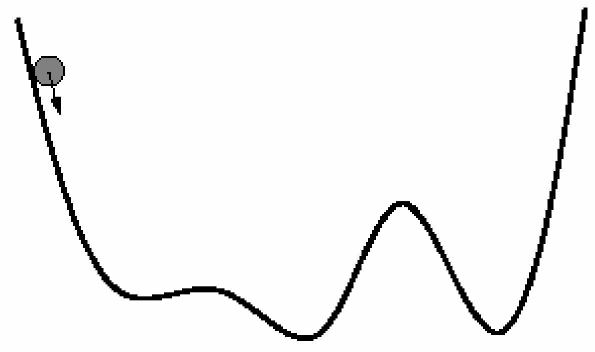
1. $f(s) = s$ (equilibrium)

2. For each $\varepsilon>0$ there exists a $0<\delta\leq\varepsilon$ such that for all $n\geq1$ $|f_n(s')-s| < \varepsilon$ whenever $|s'-s| < \delta$
   (stability)



3. For each $\varepsilon>0$ there exists a $0<\delta\leq\varepsilon$ such that $\lim_{n\to\infty} f^n(s') = s$ whenever $|s'-s| < \delta$
   (asymptotic stability)

- How can we be sure the dynamics converges to any attractor? Why cannot it enter an endless loop $\mathbf{s}^1 \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^3 \rightarrow \mathbf{s}^1 \rightarrow \ldots$?
- Answer (and the secret of the Hopfield network's popularity): the energy function (also called *Ljapunov* function).
- An energy function (Lyapunov function) always decreases monotonically as we change state and is bounded below. The descent to lower energy levels will have to end eventually at a local minimum.
- Energy landscapes are a popular (and somewhat dangerous) analogy.



Definition A neural network [S,W,F] is called a resonance system iff $\lim_{n\to\infty} f^n(s)$ exists and is a resonance for each $s\in S$ and $f\in F$.

Theorem 1 (Cohen & Großberg 1983)
Hopfield networks are resonance systems.
(The same holds for a large class of other systems: The McCulloch-Pitts model (1943), Cohen-Grossberg models (1983), Rumelhart's Interactive Activation model (1986), Smolensky's Harmony networks (1986), etc.)

Lemma (Hopfield 1982)

The function $E(s) = -\frac{1}{2} \sum_{i,j} w_{ij}\, s_i\, s_j$ is a *Ljapunov-function* of the system in the case of asynchronous updates.

That means:

- when the activation state of the network changes, E can either decrease or remain the same.

Consequence: The output states $\lim_{n\to\infty} f^n(s)$ can be characterized as *the local minima* of the Ljapunov-function.

Remark: $E(s) = -\frac{1}{2} \sum_{i,j} w_{ij}\, s_i\, s_j = -\sum_{i<j} w_{ij}\, s_i\, s_j$ (symmetry!)

For the proof we assume a discrete working space ($s_i = \pm 1$)

Let node 1 be selected for update:

$s_1(t+1) = \theta(\Sigma_j w_{1j} \cdot s_j(t))$; $s_j(t+1) = s_j(t)$ for $j \neq 1$

Case 1: $s_1(t+1) = s_1(t)$, then $E(t+1) - E(t) = 0$

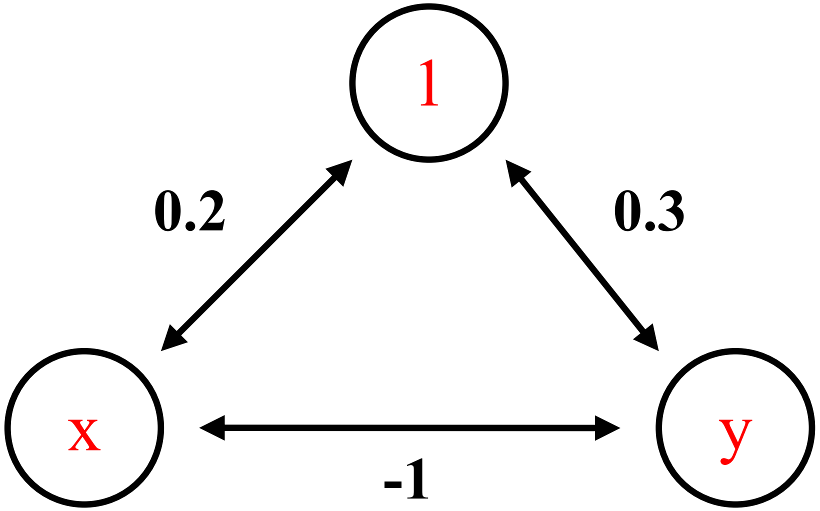
Case 2: $s_1(t+1) = -s_1(t)$ [binary threshold!, working space {-1, +1}]

We have $E(\mathbf{s}(t)) = -\sum_{i<j} w_{ij}\, s_i(t)\, s_j(t)$. For the difference $E(t+1)–E(t)$ only the terms with index $i=1$, $1<j$ matter. Consequently,

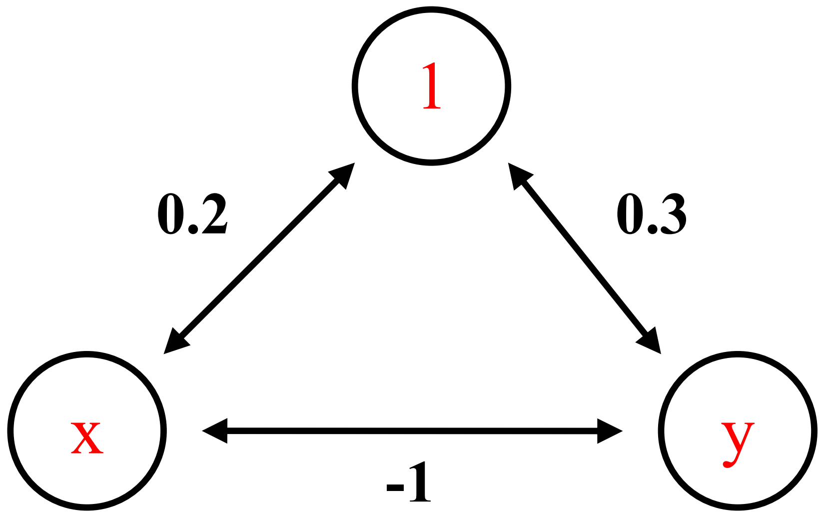
$E(t+1)–E(t) = -\sum_{j>1} w_{1j}\, s_1(t+1)\, s_j(t) + \sum_{j>1} w_{1j}\, s_1(t)\, s_j(t) =$

$-\sum_j w_{1j}\, s_1(t+1)\, s_j(t) - \sum_j w_{1j}\, s_1(t+1)\, s_j(t) = -2\, s_1(t+1) \cdot \sum_j w_{1j}\, s_j(t) < 0$

because the two factors have the same sign!



- What is the Ljapunov-function E(x,y) for this system?
- What is the global minimum? (assuming binary activations ±1)



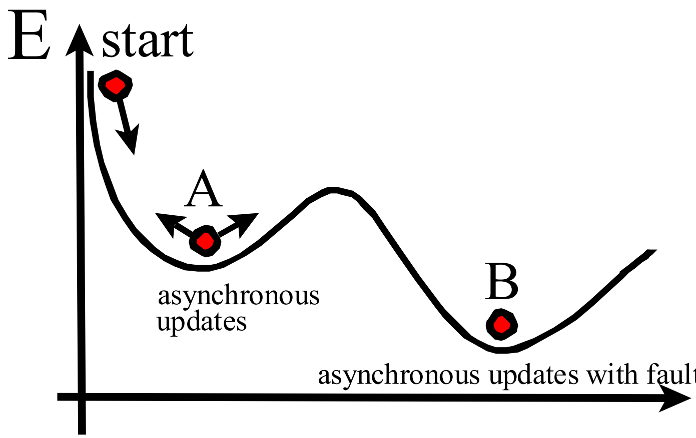
$E = -\sum_{i<j} w_{ij} s_i s_j$

$E(x,y) = -0.2x - 0.3y + xy$

x	y	E
1	1	.5
1	-1	-.9
-1	1	-1.1
-1	-1	1.5

Theorem 2 (Hopfield 1982)

The output states $\lim_{n\to\infty} f^n(s)$ can be characterized as *the global minima* of the Ljapunov-function if certain stochastic update functions f are considered ("simulated annealing").



What we need is a probability distribution for the states P(**s**) for each time and a stochastic update rule which respects that there is some stochastic disturbance during updating the activation vectors.

How to escape local minima?

- One idea: add randomness, so that we can go uphill sometimes. Then we can escape shallow local minima and more likely end up in deep minima. We can use a *probabilistic update rule*.
- Too little randomness: we end up in local minima. Too much, and we jump around instead of converging.
- Solution by analogy from thermodynamics: annealing through slow cooling. Start the network in a high temperature state, and slowly decrease the temperature according to an annealing schedule.

Deriving a probability distribution

What is a plausible probability distribution for activation states?

1. Assume that the probability of an activation state is an function of its *energy*: $P(\mathbf{s}) = f(E(\mathbf{s}))$
2. Assume independent probability distributions for independent subnets $E(\mathbf{s} \oplus \mathbf{s'}) = E(\mathbf{s}) + E(\mathbf{s'})$; $P(\mathbf{s} \oplus \mathbf{s'}) = P(\mathbf{s}) \cdot P(\mathbf{s'})$

   (*) $f(E+E') = f(E) \cdot f(E')$

Assume f is a continuous function that satisfies f, then it must be an exponential function. Hence,

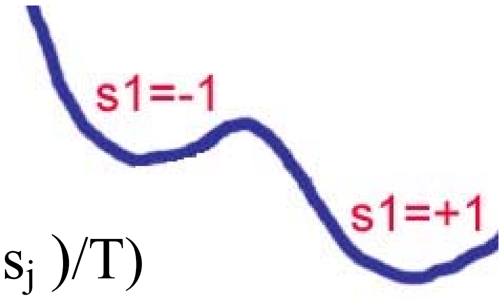
$P(\mathbf{s}) = \text{const} \cdot e^{k \cdot E(\mathbf{s})}$, or with $k = -1/T$:

$P(\mathbf{s}) = \text{const} \cdot e^{-E(\mathbf{s})/T}$

A probabilistic update rule

Assume (without restricting generality) that unit 1 is selected for updating at time t and has activity –1 at time *t*. Should it flip from –1 to +1 at time t+1? The energy $E = -\sum_{i<j} w_{ij}\, s_i\, s_j$ is relevant!

t:	$s_1 = -1, s_2, \ldots, s_n$	
t+1	$s_1 = -1, s_2, \ldots, s_n$	$s_1 = +1, s_2, \ldots, s_n$
Energy	E	$E - 2\sum_j w_{1j}\, s_j$
Prob	$c \cdot \exp(-E/T)$	$c \cdot \exp((-E + 2\sum_j w_{1j}\, s_j)/T)$



If $\sum_j w_{1j}\, s_j$ is positive, then flip the activation with a probability that increases exponentially with the energy difference $2\sum_j w_{1j}\, s_j$:

$P(s_1 = +1) = \sigma(2(\sum_j w_{1j}\, s_j)/T)$ with the sigmoid function σ.

Classical rule for $T \to 0$

At each step, select a unit and calculate the energy difference ΔE between its current state and its flipped state.

If $\Delta E \geq 0$ don't flip the unit.

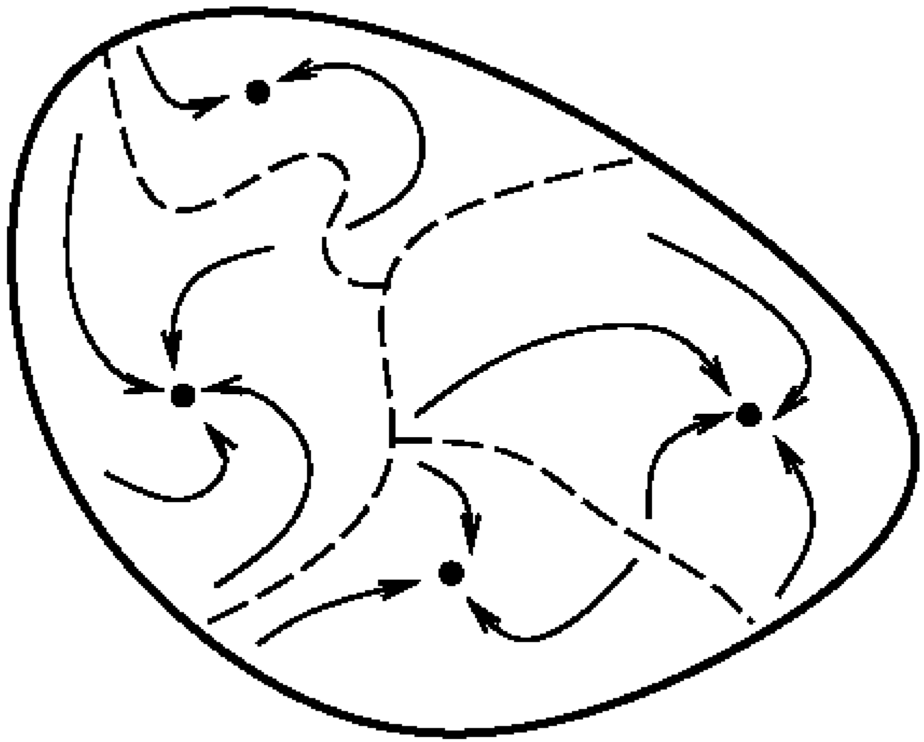
If $\Delta E < 0$, flip the unit with probability $\sigma(-\Delta E/T)$.

After around n such steps, lower the temperature further and repeat the cycle again.

Geman and Geman (1984)

If the temperature in cycle *k* satisfies $T_k \geq \frac{T_0}{\log(1+k)}$ for every *k* and T_0 is large enough, then the system will with probability one converge to the minimum energy configuration.

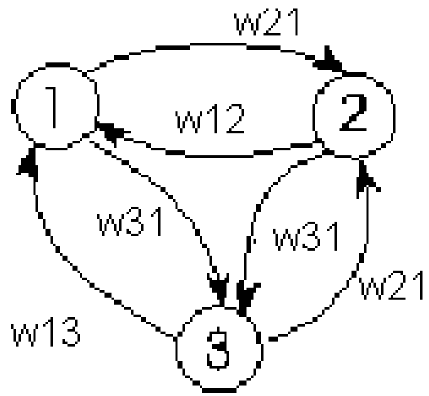
3 Learning with Hopfield networks



Generalized Hebbian rule for a single neuron confronted with a input vector $\mathbf{s}^d$, working space $S = \{-1, 1\}$

$\Delta\mathbf{w} = \eta \cdot \mathbf{s}^d \cdot r^d \quad d \in D$

Hopfield used this rule for his networks:



$\Delta w_{ij} = \eta \cdot s_j^d \cdot r_i^d$ or equivalently

$\Delta w_{ij} = \eta \cdot s_j^d \cdot s_i^d \quad d \in D$ and $i \neq j$!

In this case, the resulting connection matrix can be shown to be

$w_{ij} = {}^1/_N \cdot \Sigma_{d \in D}\, s_j^d \cdot s_i^d$ for $i \neq j$; zero for $i = j$

- Consider a network with 3 neurons. Teach the system with the input vector (1 1 1). What is the weight matrix?

- Take the same system, but now teach the system with two input vectors (1 1 1) and (-1 -1 -1). Is there a behavioural difference that corresponds by adding the second input vector? Take the activation function to be a binary threshold.

- Consider a network with 3 neurons. Teach the system with the input vector (1 1 1). What is the weight matrix?

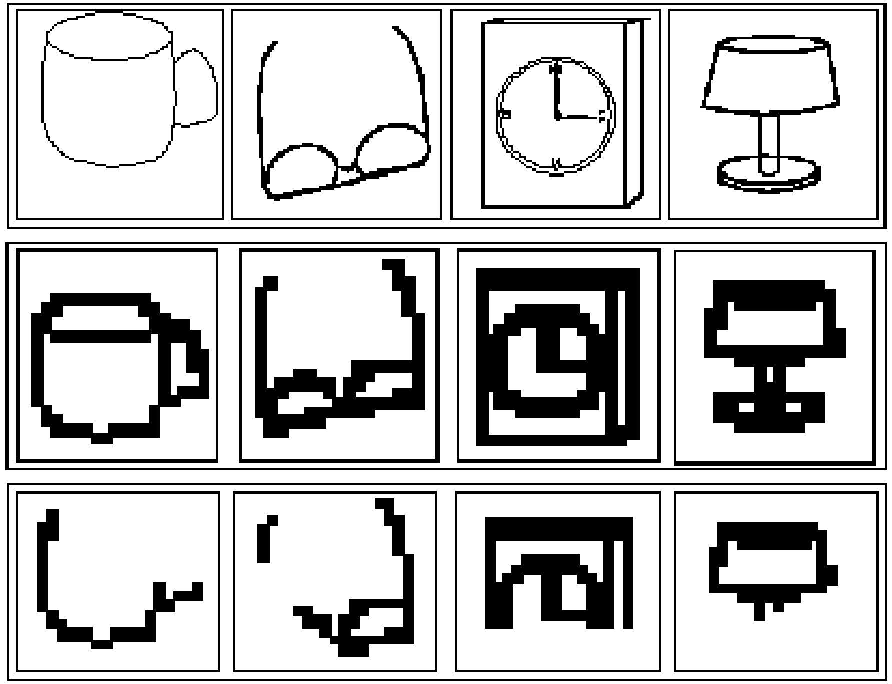
$$w = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

- Take the same sytem, but now teach the system with two input vectors (1 1 1) and (-1 -1 -1). Is there a behavioural difference that corresponds by adding the second input vector? Take the activation function to be a binary threshold.

$$w = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

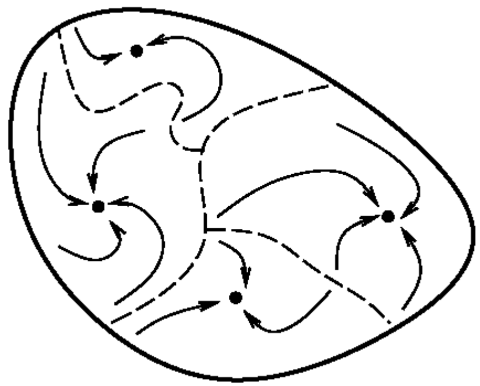
No behavioural difference to the case before

4 Emerging properties of Hopfield networks



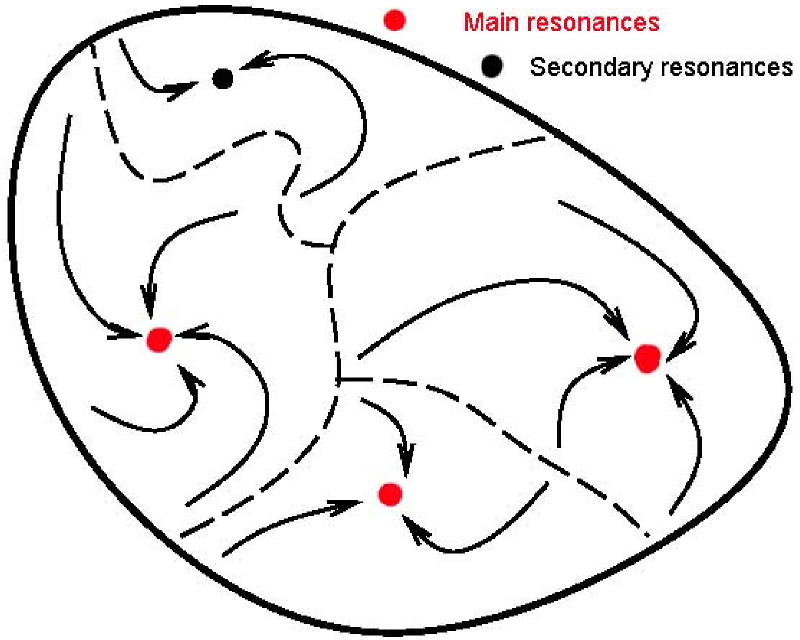
Store a set of patterns { $\mathbf{s}^{d}$ } in such a way that when presented with a pattern $\mathbf{s}^{x}$ it will respond with the stored pattern that is most similar to it. Maps patterns to patterns of the same type. (e.g. a noisy or incomplete record maps to a clear record).

- Mechanism of pattern completion: The stored patterns are attractors. If the system starts outside any of the attractors it will begin to move towards one of them.
- Stored patterns are addressable by *content*, not *pointers* (as in traditional computer memories)

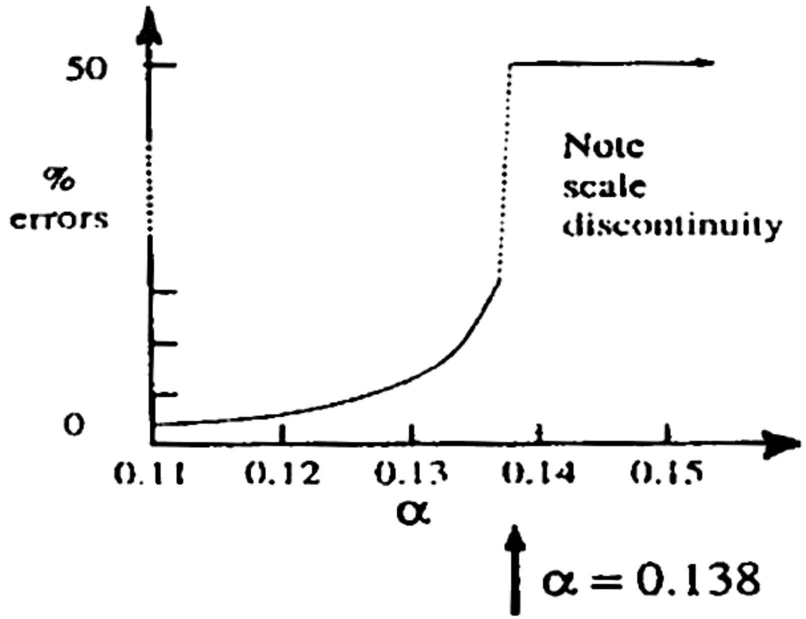


See the link Hopfield network as associative memory on the website

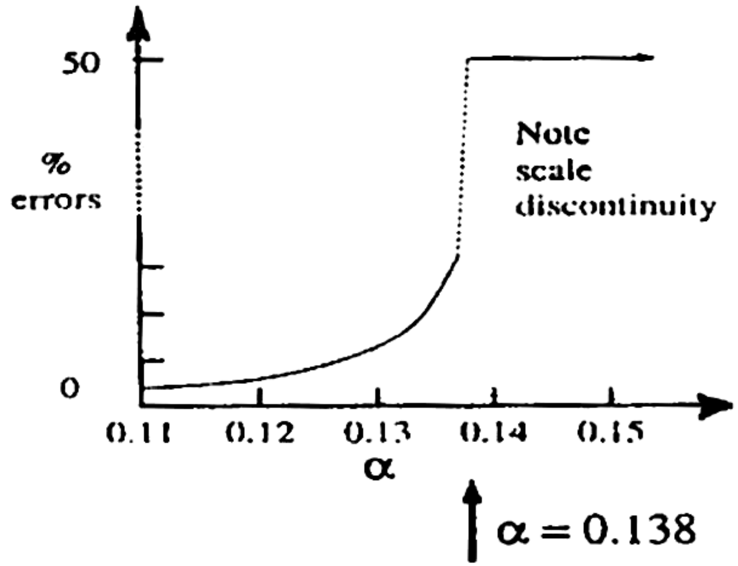
- The inputted patterns of activation are resonances. However, they are not the only resonances of the system
- The state (-1, -1, …, -1) is always a resonance
- If **s** is a resonance, so is -**s**



- How many patterns can a *n* unit network store?
- The more patterns added, the more crosstalk and spurious states (second. resonances). The larger the network, the greater the capacity.
- It turns out that the capacity is roughly proportional to n: $M = \alpha \cdot n$, where M is the number of inputted pattern that can be correctly reproduced (with an error probability of p)



- If we want $p_{error} = 0.01$, then $M = 0.105 \cdot n$. This is an upper bound.
- At $M = 0.138 \cdot n$ patterns the network suddenly breaks down and cannot recall anything useful ("catastrophic forgetting").
- The behaviour around $\alpha = 0.138$ corresponds to a phase transition in physics (solid – liquid).
- Physical analogy: spin glasses. Unit states correspond to spin states in a solid. Each spin is affected by the spins of the others plus thermal noise, and can flip between two states (1 and -1).



5 Conclusions

- Hopfield nets are "the harmonic oscillator" in modern neurodynamics
- the idea of resonance systems
- the idea of simulated annealing
- the idea of content addressable memory
- very simple learning theory based on the generalized Hebbian rule.