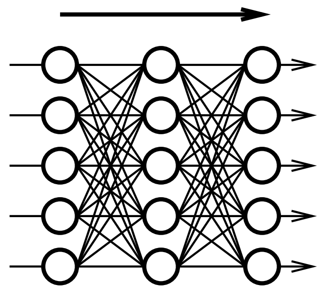
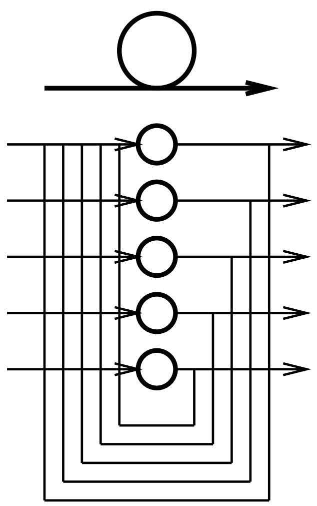
Recurrent Networks

Feed-forward networks: input transformed into output ($y = f(x)$), no memory, no feedback.



Recurrent networks: feedback loops, the network responds both to input and it's own state. It becomes a dynamical system ($y_{n+1} = f(y_n, x)$).



Neurodynamics

Dynamical Systems + Neural Networks = Neurodynamics

Neural networks as dynamical systems; function depends on the behavior in state space.

Tools from dynamical systems, statistics and statistical physics can be used.

Very rich field.

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
- Combinatorial optimization

Associative Memories

"Content Addressable Memory" (CAM)

Heteroassociative networks: Store a set of pattern pairs $\{(\xi^\mu, \zeta^\mu)\}$ in such a way that when presented with a pattern ξ^ν it will respond with the stored pattern ζ^ν (or the ζ of the pattern most similar to ξ^ν). Associates patterns of type A with patterns of type B, and possibly vice versa (e.g. image bitmaps with character codes).

Autoassociative memory: Store a set of patterns $\{\xi^\mu\}$ in such a way that when presented with a pattern ξ^ν it will respond with the stored pattern ξ^μ 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).

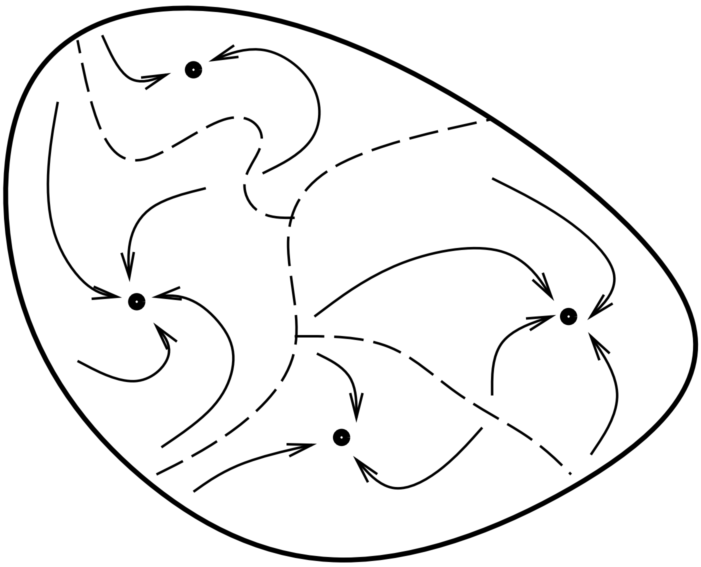
Attractors

How do we do pattern completion and noise removal?

- If shown the correct pattern do nothing.
- If shown a slightly distorted pattern, fix it.
- If the recalled pattern remains distorted, feed it back again and continue until the pattern doesn't change anymore.

This can be done using a recurrent autoassociative network.

The stored patterns are attractors. If the system starts outside any of the attractors it will begin to move towards one of them. The configuration space is divided into basins of attraction.



Note that the configuration space might be discrete (the Hopfield network has a hypercube configuration space) and very high dimensional (> 100 dimensions is quite common, since each neuron contributes a dimension).

The basin boundaries may be very complex.

Dynamical Systems

Continous dynamical system:

$$\frac{d}{dt}\vec{X}(t) = \vec{F}(\vec{x}(t))$$

where $x(t)$ is a vector-valued state and F a nonlinear vector-valued function.

If $\vec{F}(\vec{x}^*) = \vec{0}$, $\vec{x}^*$ is a fixed point (equilibrium point). The local properties of $\vec{F}$ around it will determine if trajectories approach it, circle it or escape. If trajectories approach, it is called an attractor.

Attractor states such as limit cycles or strange attractors are also possible.

The set of points that are eventually attracted to an attractor is called its basin of attraction.

Lyapunov functions can be used to prove the existence of attractor points.

Neurodynamics

Create attractors to solve problems (e.g. store memories or optimize something) by setting the weights (supervised and unsupervised learning, as well as ad hoc).

Usually an encoding phase where information is stored/trained, followed by a retrieval phase. For retrieval, unit activities are either set to a suitable state and allowed to evolve, or some units are clamped and the others allowed to change.

Questions: Will it converge? And to what? How much can be stored?

The Hopfield Network

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

N neurons with states $S_i \in \{-1, 1\}$ and a (symmetric) weight matrix w_{ij}.

Update rule for state i: $S_i \leftarrow \sigma\left(\sum_{j=0}^{N} w_{ij} S_j\right)$ where $\sigma(x)$ is the threshold function:

$$\sigma(x) = \begin{cases} -1 & x \leq 0 \\ 1 & x > 0 \end{cases}$$

Asynchronous updating (choose one i randomly for each update) has nice properties as we will see.

Sometimes a bias term β_i is used:
$S_i \leftarrow \sigma\left(\sum_{j=0}^{N} w_{ij} S_j + \beta_i\right)$

Weights, Stability and Convergence

How to find the right weights?

One training pattern ξ^1: let $w_{ij} = \xi_i^1 \xi_j^1$ (outer product).

If the network starts in state $S_i = \xi_i^1$, does it change state?

$$
\begin{aligned}
S_i \leftarrow & \sigma\left(\sum_{j=0}^{N} w_{ij}\xi_j^1\right) = \\
& \sigma\left(\sum_{j=0}^{N} \xi_i^1\xi_j^1\xi_j^1\right) = \\
& \sigma\left(\sum_{j=0}^{N} \xi_i^1\right) = \\
& \sigma\left(N\xi_i^1\right) = \\
& \xi_i^1
\end{aligned}
$$

Stable!

Several patterns: add together the outer products: $w_{ij} = \frac{1}{N} \sum_{\mu}^{p} \xi_i^\mu \xi_j^\mu$

Note that $w_{ij} = w_{ji}$. The symmetry of the weights is very important (and nonbiological).

What happens if one unit is in the wrong state? $S_k = -\xi_k^\mu$, $S_i = \xi_i^\mu$ for $i \neq k$

$$S_i \leftarrow \sigma \left(\sum_{j=0}^{N} w_{ij} S_j \right) =$$

$$\sigma \left(w_{ik} S_k + \sum_{j \neq k}^{N} w_{ij} S_j \right)$$

But $|w_{ik} S_k| = O(1)$ is likely much smaller than $|\sum_{j \neq k}^{N} w_{ij} S_j| = O(N)$, so the total sum will have the same sign regardless of the sign of S_k, and hence S_i will have the right sign.

It seems that the Hopfield network usually can correct single bit errors.

In practice it can correct many-bit errors.

How does the dynamics look?

Attractor states corresponding to the trained patterns $\Rightarrow$ stable memories. Flow towards them $\Rightarrow$ pattern completion, noise removal.

But there are monsters in state space:

If ξ^μ is an attractor state, then so is $-\xi^\mu$:

$$\begin{aligned}
&\sigma\left(\sum_{j=0}^{N} w_{ij}(-\xi_j^\mu)\right) = \\
&\sigma\left(-\sum_{j=0}^{N} w_{ij}\xi_j^\mu\right) = \\
&-\sigma\left(\sum_{j=0}^{N} w_{ij}\xi_j^\mu\right) = \\
&-\xi_i^\mu
\end{aligned}$$

Mixed states: $\xi_i^\nu = \sigma(\xi_i^1 + \xi_i^2 + \xi_i^3)$

Spurious states: not correlated to any learned patterns.

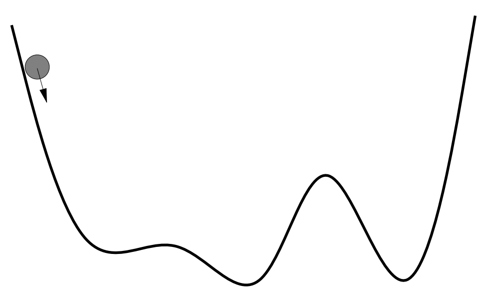
Energy and Convergence

How can we be sure the dynamics converges to any attractor? Why cannot it enter an endless loop $\xi^1 \to \xi^2 \to \xi^3 \to \xi^1 \to \ldots$?

Answer (and the secret of the Hopfield network's popularity): the energy 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.



Energy in Hopfield Networks

An energy function for the Hopfield network:

$$E = -\frac{1}{2} \sum_{ij} w_{ij} S_i S_j$$

This always decreases when units are updated asynchronously.

Note that the w_{ii} terms just contribute a constant factor to the energy and can be ignored; it turns out that self-connections worsen performance slightly.

$E = C - \sum_{(ij)} w_{ij} S_i S_j$ where (ij) are all the *distinct* pairs of indices.

The symmetry of the weights is crucial.

Proof of stability

Let S_i be the unit selected for an update, and S_i' its new value. If $S_i = S_i'$, then E will not change. Otherwise, $S_i' = -S_i$. Take the difference in energy, cancelling all the terms not involving S_i':

$$
\begin{aligned}
E' - E &= -\sum_{j \neq i} w_{ij} S_i' S_j + \sum_{j \neq i} w_{ij} S_i S_j \\
&= 2S_i \sum_{j \neq i} w_{ij} S_j \\
&= 2S_i \sum_{j} w_{ij} S_j - 2w_{ii} \\
&= 2S_i S_i' - 2w_{ii}
\end{aligned}
$$

The first term will be negative since the factors have different signs, the second term is zero or negative. Thus for all transitions, the energy difference will be negative.

Note that for synchronous updating the above derivation does not hold; the network may end up flipping between two (but not more) states.

Capacity

How many patterns can a N unit network store?

The more patterns added, the more crosstalk and spurious states. The larger the network, the greater the capacity.

Load parameter $\alpha = \frac{M}{N}$.

It turns out that the capacity is roughly proportional to N.

Probe with ξ^ν. The local field h_i of neuron i is

$$
\begin{aligned}
h_i &= \sum_{j=0}^{N} w_{ij}\xi_j^\nu \\
&= \sum_{j=0}^{N}\left(\frac{1}{N}\sum_{\mu=1}^{M}\xi_i^\mu\xi_j^\mu\right)\xi_j^\nu \\
&= \sum_{j=0}^{N}\frac{1}{N}\left(\xi_i^\nu\xi_j^\nu + \sum_{\mu=1,\mu\neq\nu}^{M}\xi_i^\mu\xi_j^\mu\right)\xi_j^\nu \\
&= \frac{1}{N}\sum_{j=0}^{N}\xi_i^\nu\xi_j^\nu\xi_j^\nu + \frac{1}{N}\sum_{j=0}^{N}\left(\sum_{\mu=1,\mu\neq\nu}^{M}\xi_i^\mu\xi_j^\mu\right)\xi_j^\nu \\
&= \frac{1}{N}\sum_{j=0}^{N}\xi_i^\nu + \frac{1}{N}\sum_{j=0}^{N}\left(\sum_{\mu=1,\mu\neq\nu}^{M}\xi_i^\mu\xi_j^\mu\right)\xi_j^\nu \\
&= \xi_i^\nu + \frac{1}{N}\sum_{j=0}^{N}\left(\sum_{\mu=1,\mu\neq\nu}^{M}\xi_i^\mu\xi_j^\mu\right)\xi_j^\nu \\
&= \text{Memory} + \text{Crosstalk}
\end{aligned}
$$

When M is small the second term is likely small.

What is the probability that h_i has the right sign if $\xi_i^\nu = 1$? Lets call it $P = P(h_i > 0 | \xi_i^\nu = 1)$. The local field can be viewed as a gaussian random variable with mean 1 and variance $\sigma^2 = (M-1)/N$:

$$P = \frac{1}{\sqrt{2\pi\sigma}} \int_0^\infty \exp\left(-\frac{(h_i-1)^2}{2\sigma^2}\right) dh_i$$
$$= \frac{1}{2}\left(1 + \operatorname{erf}\left(\sqrt{\frac{1}{2\sigma}}\right)\right)$$

(where $\operatorname{erf}(x) = 2/\sqrt{\pi}\int_0^x \exp(-t^2)dt$)

For $M \to \infty$ the probability goes to $1/2$.

Since there are N bits in a pattern, the probability that the pattern ξ^ν is stable is $p_{\text{stable}} = (P(h_i > 0 | \xi_i^\nu = 1))^N$.

If we want $p_{\text{stable}} = 0.99$, then $\alpha = 0.15N$. This is an upper bound, since if one unit flips, more may flip and we get an avalanche. At $M = 0.138N$ patterns the network suddenly breaks down and cannot recall anything useful ("catastrophic forgetting").

Variants

- Sparse connections
- Sparse activity
- Binary activity
- Continuous unit activations
- Forgetful learning
  - Marginalist learning
  - Clipped weights
  - Incremental learning
- “Designer energy functions” for optimization
- Stochastic neurons

Continous Hopfield Model

$$C_i \frac{d}{dt} x_i(t) \;=\; -\frac{v_i(t)}{R_i} + \sum_{i=1}^{N} w_{ij}\theta(v_j(t)) + I_i$$

where v_i is the support, $\theta(x)$ is a sigmoid function of some kind (e.g. $\tanh(gx)$), $x_i = \theta(v_i)$ the output and I_i the external input. The weights are set just like in the discrete case.

The energy is defined as

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_i x_j$$
$$+\sum_{j=1}^{N}\frac{1}{R_j}\int_0^{x_j}\theta_j^{-1}(x)dx - \sum_j = 1^N I_j x_j$$

The energy minima are not corners of the $-1, 1^N$ hypercube, but lie inside.

The Cohen-Grossberg Theorem

If

$$\frac{d}{dt}u_j = a_j(u_j)\left[b_j(u_j) - \sum_{i=1}^{N} c_{ji}\phi_i(u_i)\right]$$

is the network update rule and $c_{ij} = c_{ji}$ (symmetric), $a_j(u_j) \geq 0$ (non-negative) and $\phi_j'(u_j) \geq 0$ (monotonicity), then the network has a Lyapunov function

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ji}\phi_i(u_i)\phi_j(u_j) - \sum_{j=1}^{N}\int_0^{u_j} b_j(\lambda)\phi_j'(\lambda)d\lambda$$

with

$$\frac{dE(t)}{dt} \leq 0$$

as the dynamics evolves.

Network dynamics as optimization

The dynamics tries to minimize the energy function. Can we use this for optimization?

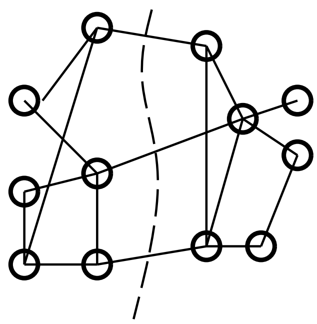
Problem $\Rightarrow$ cost function + constraints $\Rightarrow$ energy function $\Rightarrow$ weight matrix.

This works for problems with a quadratic symmetric cost function $C(S)$ and a set of quadratic constraints $c(S)$ on a set of binary variables S_i.

Construct the energy function $E[S] = aC(S) + bc(S)$. Rewrite it as $-\frac{1}{2}\sum_{ij} w_{ij} S_i S_j - \sum_i \beta_i S_i$ which gives the weights w_{ij} and biases β_i.

Graph Bipartitioning

Problem: Divide a graph with an even number N of vertices into two equal size sets, minimizing the number of edges that cross the partition.



Let S_i be 1 if vertex i is in set 1 and -1 if vertex i is in set 2. Let $A = (a_{ij})$ be the adjacency matrix ($a_{ij} = 1 \Leftrightarrow$ node i and j linked, zero otherwise).

Then we want to minimize $-\sum_{ij} a_{ij} S_i S_j$ (increases with the number of crossings) subject to $\sum_i S_i = 0$ (set 1 and 2 have equal size).

Energy function with soft constraint:

$$E[n] = -\sum_{ij} a_{ij} S_i S_j + \mu \left(\sum_i S_i \right)^2$$

Multiply out the last term:

$$E[n] = N\mu - \sum_{ij} (a_{ij} - 2\mu) S_i S_j$$

So we get $w_{ij} = a_{ij} - 2\mu$.

Simulated Annealing

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 the 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.

Stochastic neurons

Analogy with spin models in condensed matter physics. 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).

A probabilistic update rule:

$$S_i \leftarrow \begin{cases} -1 & \text{with probability } P(h_i) \\ 1 & \text{with probability } 1 - P(h_i) \end{cases}$$

$P(h_i) = \frac{1}{1+\exp(-2h_i/T)}$ where h_i is the "local field" $\sum_j w_{ij} S_j$.

For $T = 0$ we get the original deterministic Hopfield rule. For $T = \infty$ the unit states flip completely randomly. In between $P(h_i)$ is sigmoid.

The Metropolis algorithm

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

If $\Delta E \leq 0$, flip the unit.

If $\Delta E > 0$, flip the unit with probability $\exp\left(-\frac{\Delta E}{T}\right)$

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

Geman and Geman: 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.

Good news: we get to the minimum. Bad news: it is s-l-o-w…