Hopfield Networks

Hopfield Networks

- John Hopfield: Professor at Princeton, Caltech, then Princeton
- According to Terry Sejnowski (then Hopfield's graduate student), Hopfield nets may have been suggested by Sejnowski .

Approaches to Hopfield Nets

- Recurrent neural nets without sequential input, or
- Extend **linear associative memory** ideas by adding cyclic connections, or
- Special case of Kosko's BAM (Bi-Directional Associative Memory, proposed later), or
- Derive from Cohen-Grossberg theorem (not covered yet).

Hopfield Nets

- Generally considered to be **fixed-weight** models; **they don't learn**.
- However, one way to get the weights is through the supervised **Hebbian** outer-product summation as used in the Linear Associative Model.
- Some insensitivity to noise or network damage.
- Some extensions do learn: e.g.Boltzmann network.

Applications

- Associative or content-addressable memory.
- Model of memory as a dynamical system.
- A technique for finding solutions to certain optimization problems.
- The *practical* applications do not seem so plentiful.

Commentary from James Anderson's book (1995)

- "Theoretical physicists are an unusual lot, acting like gunslingers in the old West, anxious to prove themselves against a really good problem. And there aren't that many really good problems that might be solvable. As soon as Hopfield pointed out the connection between a new and important problem (network models of brain function) and an old and well-studied problem (the Ising model), many physicists rode into town, so to speak, with the intention of shooting the problem full of holes and then, the brain understood, riding off into the sunset looking for a newer, tougher problem. (Who was that masked physicist?)".

Commentary from James Anderson's book (1995)

- "Hopfield [1982] made the portentous comment: 'This case is isomorphic with an Ising model,' thereby allowing a deluge of physical theory (and physicists) to enter neural network modeling. This flood of new participants transformed the field. In 1974 Little and Shaw made a similar identification of neural network dynamics with the Ising model, but for whatever reason, their idea was not widely picked up at the time".

Commentary from James Anderson's book (1995)

- "Unfortunately, the problem of brain function turned out to be more difficult than expected, and it is still unsolved, although a number of interesting results about Hopfield nets were proved. At present, many of the traveling theoreticians have traveled on".

Hopfield Memory

- As with the Linear Associative Memory, the "stored patterns" are represented by the weights.
- To be effective, the patterns should be reasonably **orthogonal**.
- Theoretical lower bound on number of neurons needed to store p patterns:
  $n \geq 7p$
- Equivalently
  $p \leq .15n$

Model Variants

- Basic: Discrete state, discrete time, asynchronous
- Same as basic, but synchronous
- Continuous state, discrete time
- Continuous state, continuous time

Basic Model

- N neurons, fully connected in a cyclic fashion:
  - Values are +1, -1.
  - Each neuron has a weighted input from all **other** neurons.
  - Weights are **symmetric**: $w_{ij} = w_{ji}$ and self-weights $= w_{ii} = 0$
  - Activation function on each neuron i is
    $$f(net) = sgn(net) = \begin{cases} 1 \text{ if net} > 0 \\ -1 \text{ if net} < 0 \end{cases} \quad (net_i = \sum w_{ij}\, x_j)$$
  - If net = 0, then the output is the same as before, **by convention**.

Comment in Hertz, Krogh, and Palmer, 1991

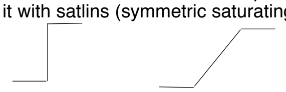
- "Gerard Toulouse has called Hopfield's use of symmetric connections a 'clever step backwards from biological realism'. The cleverness arises from the existence of an energy function".
- Toulouse, et al. Spin glass model of learning by selection. Proc. of the National Academy of Sciences, 83, 1695-1698, 1986.

Thresholds/Biases

- There are no thresholds or biases. However, these could be represented by units that have all weights = 0 and thus never change their output.

Continuous-State Variant

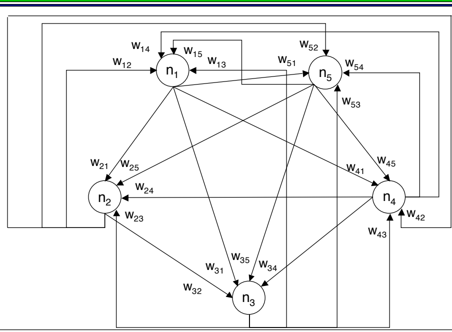
- On the previous slides, sgn is the same as hardlims (symmetric hard-limiter).
- We could allow continuous neuron outputs and replace it with satlins (symmetric saturating limiter)



discrete continuous

- One advantage of the continuous version is that it makes it easier to visualize certain phenomena such as attractors.

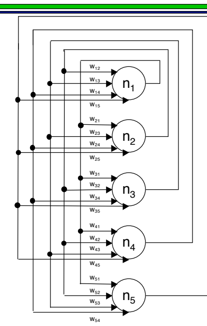
Hopfield Net



Hopfield Net

$$\begin{pmatrix} 0 & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & 0 & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & 0 & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & 0 & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & 0 \end{pmatrix}$$

$$w_{ij} = w_{ji}$$



Operation (Basic Version)

- Each neuron's output is initially **forced** to a specified value; this is the "input" state.
- Repeat forever:
  A neuron that has f(net) ≠ current output is "fired", changes its output to 1 or -1 according to the definition of f.
- The firable neuron is chosen arbitrarily.
- When and if the network stabilizes, the current state is the "output".

Operation: Synchronous Variation

- All **firable** neurons are first identified, then all change their state **simultaneously**.
- While this may be viewed as an expedient, it may create behavioral anomalies such as oscillations.

Operational Principle

- Energy Minimization:
  - For an appropriate definition of “energy”, **each single firing** can be show to **decrease** the energy.
  - Energy cannot be decreased forever; there is a definite minimum.
  - Therefore operation must eventually **terminate**.

Final State

- For **asynchronous** (basic) behavior, a **unique** final state is **not** guaranteed: it could be a **local minimum**.
- For **synchronous** behavior, **if** there is a final state, it still is a **local minimum** (it is also reachable by asynchronous firing). However, the network could instead **oscillate** forever.

Weights

- Similar to the Linear Associative Memory, weights can be computed by summing the **outer product** of the pattern vectors.
- However, after computing the sum of the outer products, the diagonal element are forced to 0.

Working an Example

- Two patterns: (1, -1, 1) and (-1, 1, -1)
- Compute the **outer products**, sum, normalize, and set diagonals to 0:
  - $(1, -1, 1)^T * (1, -1, 1)$ +

$(-1, 1, -1)^T * (-1, 1, -1)$ =

Force Diagonal to 0

$$\begin{pmatrix} 2 & -2 & 2 \\ -2 & 2 & -2 \\ 2 & -2 & 2 \end{pmatrix} \Rightarrow \frac{1}{3}\begin{pmatrix} \cancel{0} & -2 & 2 \\ -2 & \cancel{0} & -2 \\ 2 & -2 & \cancel{0} \end{pmatrix}$$

$\frac{1}{3}$ = 1/(number of neurons)

Working an Example

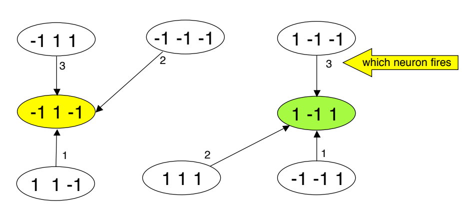
- Eight states total: (-1, -1, -1) … (1, 1, 1)
- For each state, compute the possible next states using the firing rule and the weight matrix:

$$\frac{1}{3}\begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix}$$

- Then plot the transitions, noting where the patterns occur.

Working an Example (**asynchronous**)

$$\frac{1}{3}\begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{pmatrix}$$



Working an Example (**synchronous**)

- states as columns

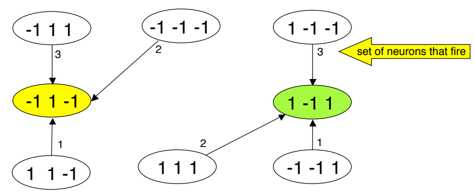
$$\frac{1}{3}\begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix}\begin{pmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{pmatrix}$$

- $$= \frac{1}{3}\begin{pmatrix} 0 & 4 & 0 & 0 & 0 & 4 & -4 & 0 \\ 4 & 0 & 4 & 0 & 0 & -4 & 0 & -4 \\ 0 & 0 & -4 & -4 & 4 & 4 & 0 & 0 \end{pmatrix}$$

- next states = $$\begin{matrix} -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \end{matrix}$$

Working an Example (synchronous)

- next states = $$\begin{pmatrix} -1 & -1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 \end{pmatrix}$$

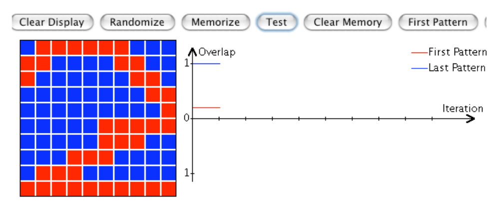


Comparison

- In this example, the asynchronous and synchronous behaviors worked out to be the same.
- This won't always be the case.
- Firing a neuron in the asynchronous *could* **disable** one of the neurons that would have fired simultaneously in the synchronous case.
- Conceivably, the synchronous case could therefore have **cycles** in its behavior.
- See if you can find an example.

Possible Demos

- http://diwww.epfl.ch/mantra/tutorial/english/hopfield/html/index.html



- matlab: demohop1, 2, 3 (uses continuous activation with satlins)

Proving that an Asynchronous Hopfield Net Terminates

- Define an **energy function**:

$$E(y_1, y_2, \ldots, y_n) = -\sum \sum w_{ij} y_i y_j$$

where $(y_1, y_2, \ldots, y_n)$ is the vector of neuron outputs, w_{ij} is the weight from neuron j to neuron i, and the double sum is over i and j.

- Remember that w is symmetric ($w_{ij} = w_{ji}$) and diagonal terms are 0.

Proving that an Asynchronous Hopfield Net Terminates

- Observation: The energy function is bounded from below.
- **Claim**: Firing any transition *decreases* the value of the energy function.

$$E(y_1, y_2, \ldots, y_n) = -\sum \sum w_{ij} y_i y_j$$

- Therefore the net cannot fire forever.

Proof of Claim

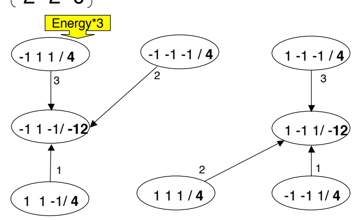
- When a neuron i fires, the *increase* (new-old) in energy is entirely due to the contribution of y_i to $-\sum w_{ij} y_i y_j$. Since w is symmetric, the amount of this increase is $-\sum w_{ij} y_i' y_j$ - $-\sum w_{ij} y_i y_j$ where y_i' represents the new value of y_i and the sum is over $i \neq j$ only.
- Since neuron i changes, $y_i' = -y_i$, so the energy increase is
  $2\sum w_{ij} y_i y_j = 2y_i \sum w_{ij} y_j$
  where the right-hand summation is over j, where $j \neq i$ only.

Proof of Claim

- The energy *increase* is
  $2y_i \sum w_{ij} y_j$
- If $y_i = 1$ ($y_i' = -1$), then we must have $\sum w_{ij} y_j < 0$ in order to activate the neuron, so the increase is 2*1*(negative) which is *negative*.
- If $y_i = -1$ ($y_i' = 1$), then we must have $\sum w_{ij} y_j > 0$ in order to activate the neuron, so the increase is 2*(-1)*(postive), which is negative.
- So there is a net energy **decrease** either way.

Checking Energy

- $$\frac{1}{3}\begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{pmatrix}$$



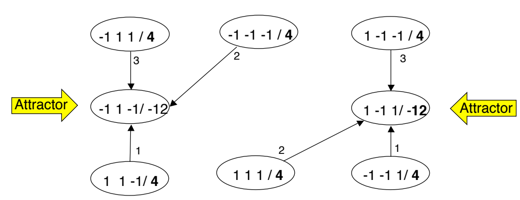
Note on Synchronous Firing

- In contrast to asynchronous firing, synchronous firing may increasing the energy.
- The analysis doesn't go through if several neurons fire at the same time.

Attractors

- Minimal energy states are known as "attractors" in the theory of dynamical systems.
- There can also be "repellors" and "saddles" (aka "meta-stable states").

Attractors



Stored Patterns Correspond to Attractors

- When the Hebb rule is used with ***orthogonal*** patterns, stored patterns correspond to attractors (stable, or minimum-energy, states).
- The reasoning is analogous to the case with the linear associative memory.

Stored Patterns Correspond to Attractors

- The supervised Hebb weight matrix is given by
  $W = \Sigma p^T p$ (with diagonals forced to 0)
  where the summation is over all patterns p as row vectors ($p^T p$ is the outer product).
- Let q be a pattern. Assuming **linear** activation functions for the moment, we have *stability* (i.e. minimum energy) if $Wq = q$ (actually satlin(Wq) = q).
- Also, stored patterns are **eigenvectors** of W, since $Wq = \lambda q$ is the equation determining eigenvalues λ and eigenvectors q.

Spurious Attractors

- The converse may be false, i.e. not every attractor is necessarily a pattern.
- For example, if **p** is an attractor, then so is -**p** (i.e. the *negative* of an image).
- Also, certain linear combinations of attractors may be attractors themselves.

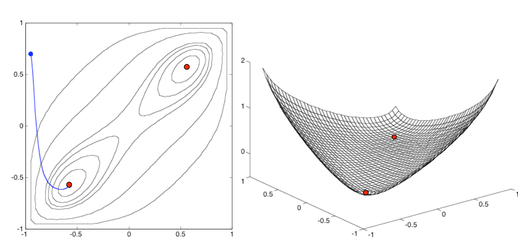
Spurious Attractors

- These aspects limit the applicability of Hopfield nets as pattern retrieval devices.
- The following paper presents a weight setting technique for minimizing the number of spurious attractors:
  Li, Michel, and Porod, Analysis and synthesis of a class of neural networks: linear systems operating on a closed hypercube, IEEE Trans. on Circuits and Systems, **36**, 11, 1405-1422, Nov. 1989.

Unlearning

- Hopfield, et al. proposed “unlearning” as a way to get rid of spurious attractors.

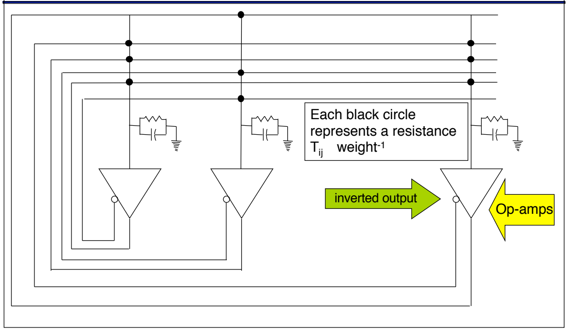
Attractors in a Continuous Analog of the Example



Lyapunov Functions

- For the continuous case, the energy function is called a Lyapunov function.
- The Hopfield network minimizes the value of the Lyapunov function.

Physical Realization of a Continuous Hopfield Net



Equations of Operation

$$C\frac{dn_i(t)}{dt} = \sum_{j=1}^{S} T_{i,j}a_j(t) - \frac{n_i(t)}{R_i} + I_i$$

n_i - input voltage to the ith amplifier
a_i - output voltage of the ith amplifier
C - amplifier input capacitance
I_i - fixed input current to the ith amplifier

$$|T_{i,j}| = \frac{1}{R_{i,j}} \qquad \frac{1}{R_i} = \frac{1}{\square} + \sum_{j=1}^{S}\frac{1}{R_{i,j}} \qquad n_i = f^{-1}(a_i) \qquad a_i = f(n_i)$$

Commercial Success?

- At least one company, Attrasoft
  http://attrasoft.com/new.htm
  claims to have products based on Hopfield nets and Boltzmann machines (to be discussed next).

Related Topics

- Boltzmann machine
- Cauchy machine
- Helmholtz machine
- Wilshaw nets