Hopfield Nets

Topic 10

Note: lecture notes by Bob Keller (Harvey Mudd College, CA) are used

Main idea: learning without learning

- 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.

Approaches to Hopfield networks

- 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).

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.

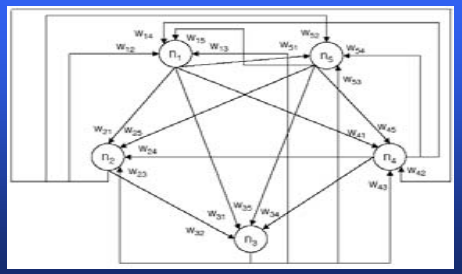
Basic model

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

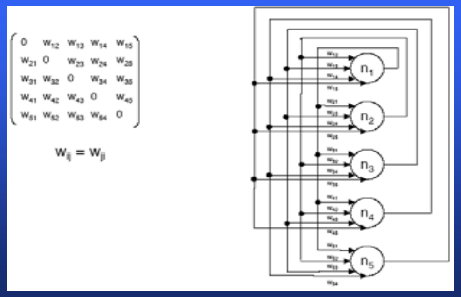
$$f(net) = sgn(net) = \begin{cases} 1 \text{ if net} > 0 \\ -1 \text{ if net} < 0 \end{cases} \qquad (net_i = \Sigma w_{ij} x_j)$$

If net = 0, then the output is the same as before, **by convention**.

Hopfield Net



Hopfield Net



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.

Operation: main principle

- Energy Minimization:

  For an appropriate definition of "energy", **each single firing** can be shown to **decrease** the energy.

  Energy cannot be decreased forever; there is a definite minimum.

  Therefore operation must eventually **terminate**.

Operation: 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.

Operation: 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.