MAE 552
Heuristic Optimization

Instructor: John Eddy

Lecture #31

4/17/02

Neural Networks

Neural Networks

References:
"Neural Networks – A Comprehensive Foundation"
Simon Haykin, 1994.

http://www.faqs.org/faqs/ai-faq/neural-nets/part2/section-21.html

http://www.pmsi.fr/neurin2a.htm#051

http://www.irit.fr/COSI/training/complexity-tutorial/non-linear-systems-neural-networks.htm

http://www.math.umn.edu/~wittman/faces/main.html

Neural Networks

Learning Cont:

Last time we ended with a description of 3 learning paradigms. Today we will begin by introducing some learning algorithms.

A learning algorithm will provide the specific implementation of the weight adjustment (how to determine Δw_{kj}.

Neural Networks

Error-Correction Learning:

This approach involved minimizing the deviation between actual and desired output.

So there is a prescribed desired output for each neuron at time step n ($d_k(n)$) and an actual output ($y_k(n)$).

Neural Networks

Thus the deviation between the actual and desired is given by:

And we have m such terms if there are m neurons in the network.

Neural Networks

So the formulation of our objective function which will attempt to simultaneously minimize all e_k's is:

Where E is the expected value (recall from statistics).

Neural Networks

So clearly, by using the expected value we are assuming that our environment is probabilistic in nature but to actually compute the expected value would require knowledge about just what that probability distribution is.

Since we do not have that information, we must alter our objective function to accommodate.

Neural Networks

We can eliminate the need for knowledge of our distribution by considering only the instantaneous value of the sum of squared errors given by:

Neural Networks

So finally, we come to an update relation like this:

Where **η** *is a positive constant that defines the rate of learning. It is a subjective value like our mutation rates etc. A very high* **η** *will cause your network to learn very quickly but also may cause divergence in the algorithm. A very low* **η** *will result in very slow learning.*

Neural Networks

So it is clear from our objective function that we want to drive each error term to zero. Looking at our update relation, it is clear that by driving our error terms to zero, we will be driving our changes-in-weights to zero.

Thus we have a convergence criteria for our learning algorithm.

Neural Networks

Point of interest:

A plot of our unabridged objective function (J) vs. the weights in our network produces a multidimensional surface that is:

- *parabolic if we have linear neurons*
- *multimodal if we have non-linear neurons.*

Neural Networks

Based on what we just learned, what category does error correction learning fall into (which of our 3 paradigms)?

Neural Networks

Hebbian Learning

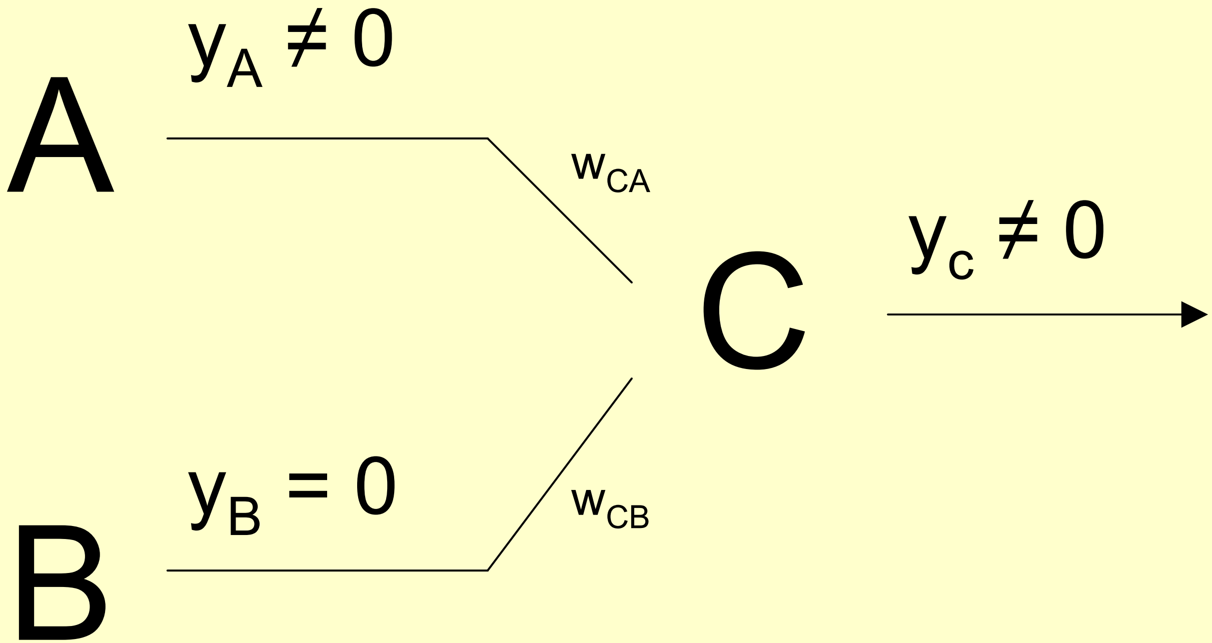
Hebbian learning is based on the concept that when 2 neurons in a brain are near enough to stimulate one another and do so often, then the connection between them grows stronger.

Neural Networks

We can generalize this to say that if the input through a particular synapse is usually non-zero during time steps in which the neuron fires, then it is a good input and its corresponding weight should be increased.

By the same token, if an input through a particular synapse is usually 0 when the neuron fires, then that synaptic weight is correspondingly weakened.

Neural Networks



Here, we increase W_{CA}
and decrease W_{CB}

Neural Networks

Implementation:

So we are saying that our update relation for a synaptic weight is a function not only of the input at that synapse, but also of the output of the neuron.

Neural Networks

The general form of the update relation is:

This is similar to our error-correction update relation but now, since we have no target value we are considering y_k in place of e_k.

Neural Networks

Especially for a long learning process, a synaptic weight has the potential to grow without bound.

To combat this, another factor is typically introduced to limit the growth of synaptic weights. The modified update formulation is:

*Where **α** is another positive constant that defines the rate of "Forgetting".*

Neural Networks

Based on what we just learned, what category does Hebbian learning fall into (which of our 3 paradigms)?

Neural Networks

Network Architectures:

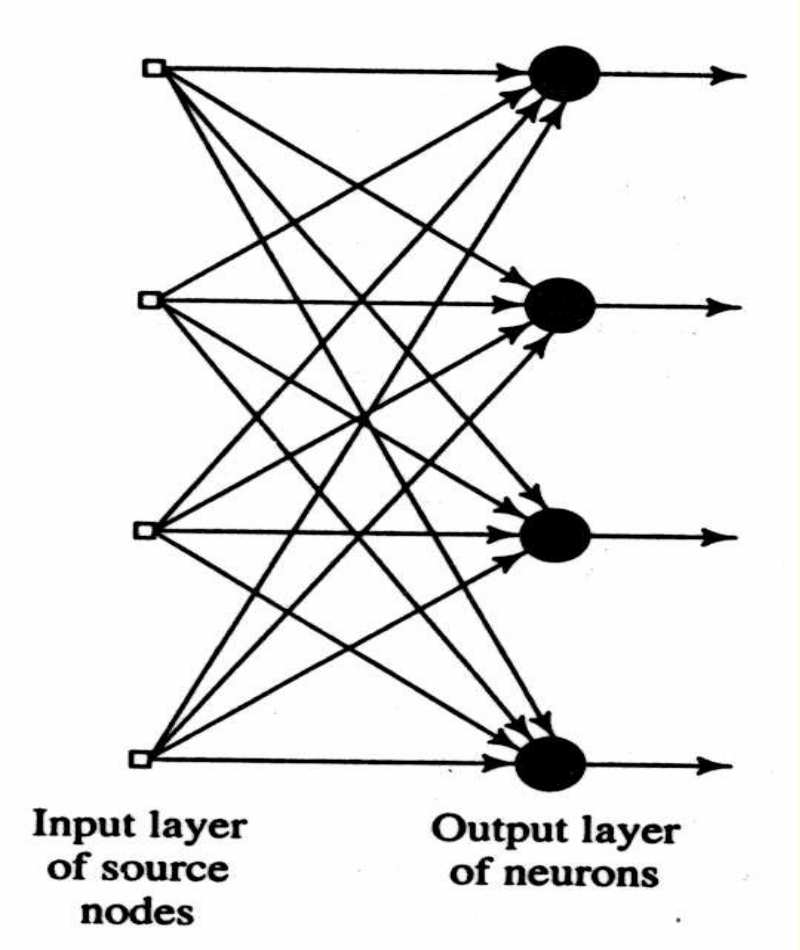
There are 3 general NN architectures that I will present.

Choice of an architecture is intimately linked with choice of a learning algorithm and both are highly dependent on the problem at hand.

Neural Networks

1 – Fully connected Single layer feed forward network

Note that there does not have to be the same number of input nodes and output nodes.

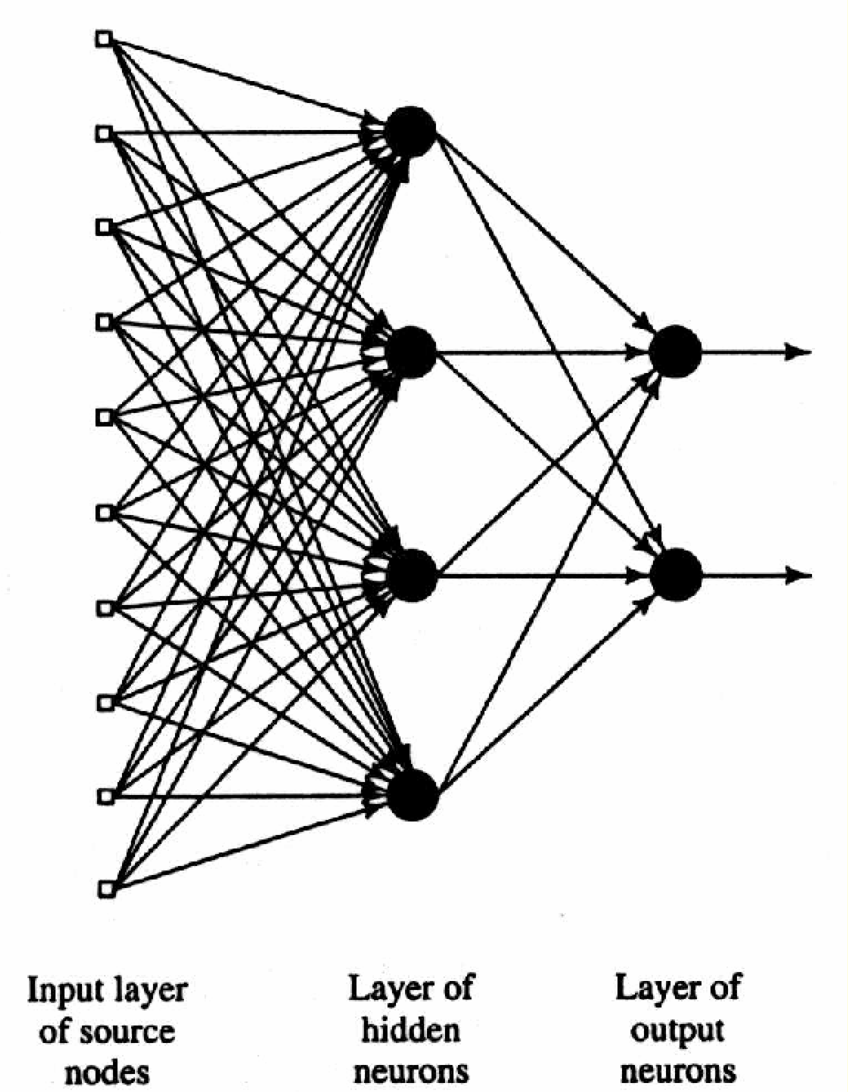


Neural Networks

2 – Fully connected Multi layer feed forward network

Notice that here we have a "hidden" layer of neurons.

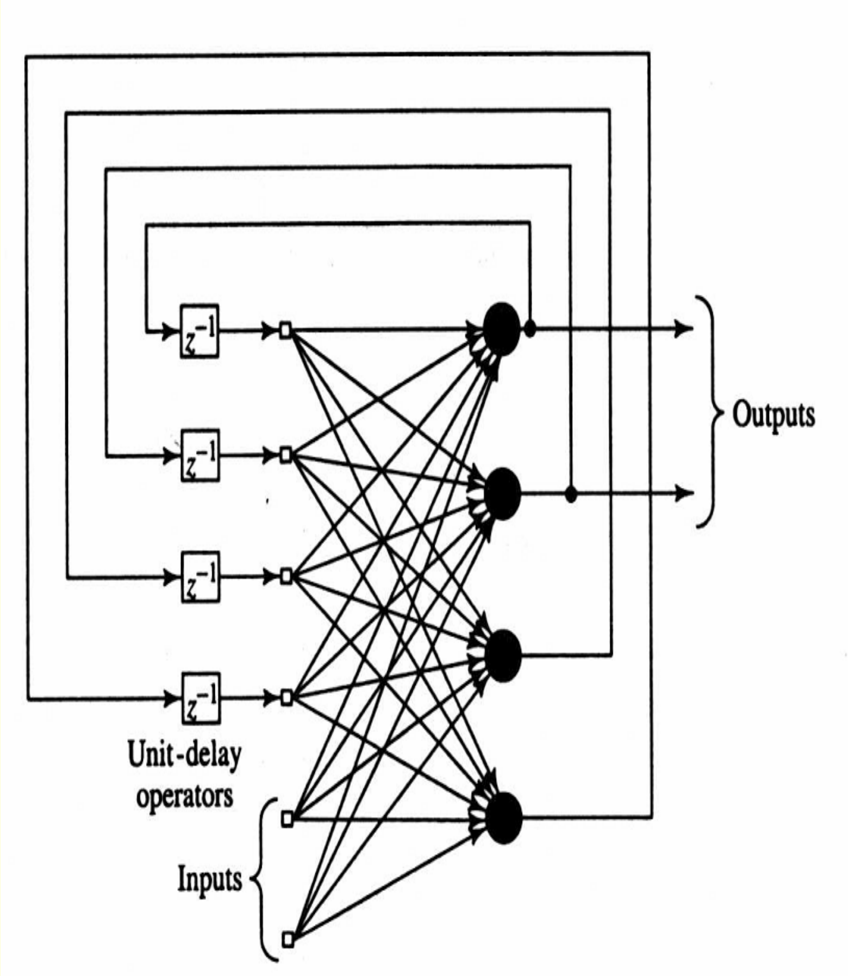
(Hidden b/c not seen by input nodes or end effectors).



Neural Networks

3 – recurrent network (with self-feedback loops).

Note that by virtue of the feedback, this network is considered to have hidden nodes.



Neural Networks

Feedback increases the dynamical nature of the NN.

And because the output of a neuron is usually the result of a non-linear function, the result is increased non-linearity for the net.