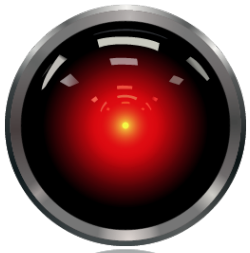


INTELLIGENT COMPUTING

(COMPUTATIONAL INTELLIGENCE)
Multilayer Neural Networks (Backpropagation)

S. Sheridan

MULTILAYER NEURAL NETWORKS

- In our previous lecture we saw how adding a hidden layer to a perceptron allowed it to overcome problems such as the XOR operator.
- We also covered the simple ADALINE network which used the Delta Rule to update weights, thus allowing the network to learn problems that were linearly separable.
- It would be nice if we could combine the idea of adding a hidden layer and apply the delta rule to update weights in order to get neural networks to solve more complicated non-linearly separable problems such as the XOR and more.
- The Backpropagation learning algorithm allows us to do just that!

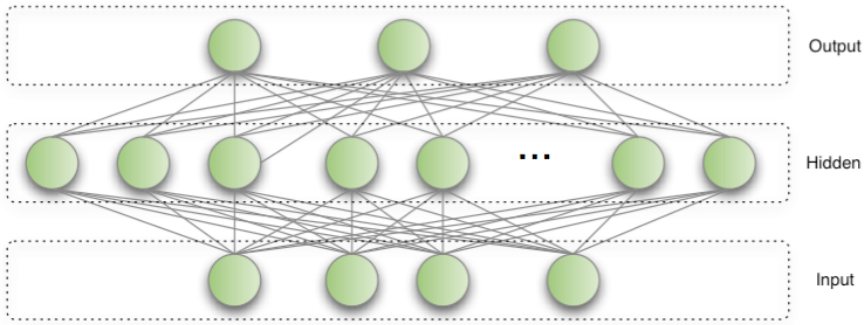
BACKPROPAGATION LEARNING ALGORITHM

- The backpropagation (**backprop**) neural network first formalised by Werbos, Rumelhart and Hinton (1974) is an extension of the ADALINE network. It is essentially a solution to the **credit assignment** problem encountered with multilayer perceptrons and networks.
- Instead of having one processing node, the backprop network is a collection of nodes (each very similar to the ADALINE node) organised into interconnected layers.
- The layered structure of the backprop network and its weight update scheme allows it to escape the ADALINE's linear separability limitation making it a much more powerful problem solving tool. The backprop network is ideal for problems involving classification, projection, interpretation and generalisation.

BACKPROP LEARNING ALGORITHM

- The nodes in the backprop network are interconnected via weighted links, and each node in a layer is generally connected to each node in the succeeding layer leaving the output layer node to provide output for the network.
- When an input pattern is presented to the backprop network, each input layer node is assigned one of the input pattern component values. The nodes in the next layer receive the input node values through the links and compute output values of their own to pass to the next layer.
- This process continues until each output layer node has produced an output for the network.

BACKPROP LEARNING ALGORITHM



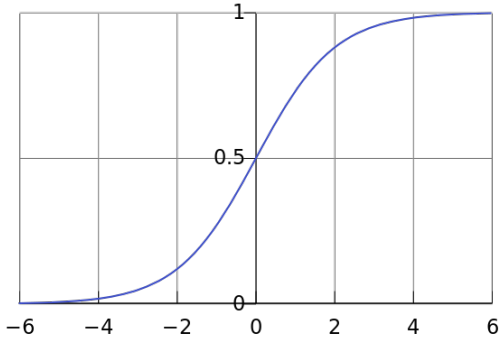
Number of **Inputs** nodes = dependant on problem
Number of **hidden** layer nodes = decided through experimentation
Number of **output** layer nodes = dependant on problem

BACKPROP LEARNING ALGORITHM

- Input patterns are identical to the ones used in the ADALINE network except they are usually scaled between 0.0 and 1.0.
- The output layer is not limited to one node and its output values are not limited to a binary output like the ADALINE network, these nodes (as well as the rest of the nodes in the backprop network) produce values between 0.0 and 1.0 (continuous).
- This is because the backprop algorithm uses a sigmoid curve for its transfer/ threshold function. The next slide shows a plot of the most commonly used backprop **sigmoid** function.

BACKPROP THRESHOLD FUNCTION

Sigmoid function



$$f(x) = \frac{1.0}{1.0 + \exp(-x)}$$

BACKPROP TRAINING DATA

- Training the a backprop network consists of presenting the network with a set of input patterns and desired output patterns. The input pattern and the desired output pattern is often called the training pair. The set of all training pairs is known as the training set.
- A training set for the XOR problem would look as follows:

Pattern No.	Input 1	Input 2	Desired Output
1	0	0	0
2	1	0	1
3	0	1	1
4	1	1	0

BACKPROP TRAINING DATA

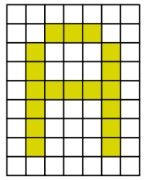
- In most cases the data sets that are required for successful network training are very large and must be presented to the network many times.
- Baum & Haussler (1989) provide a guide as to the size of the training set required.

$$N > \frac{W}{\varepsilon}$$

*Where **N** is the number of training patterns, **W** is the number of network weights and **ε** is the proportion of allowed errors.*

BACKPROP TRAINING DATA

- So for a character recognition problem with 63 input units (9x7 bitmap), 30 hidden layer nodes and an output layer of 26 units (1 to categorise each alpha char) the amount of training patterns is calculated as:



= on

= off

$$N > \frac{2{,}580}{10} = N > 258$$

- Where

  Number of weights = (63x30)+(30x26) = 2,580

  Acceptable error of 10%

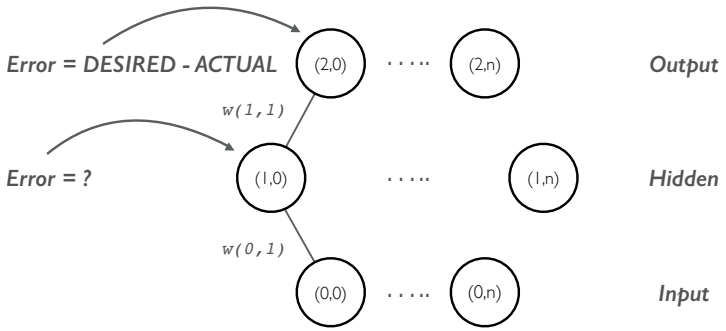
BACKPROP ALGORITHM

- Training a backprop network can be broken down into the following steps:

```
Initialise network weights (small random values)
do
For Each training example
prediction // forward pass
compute error (desired - actual) at the output units
compute Δw for all weights from hidden layer to output layer // backward pass
compute Δw for all weights from input layer to hidden layer
update network weights
until all examples classified correctly or another stopping criterion satisfied
```

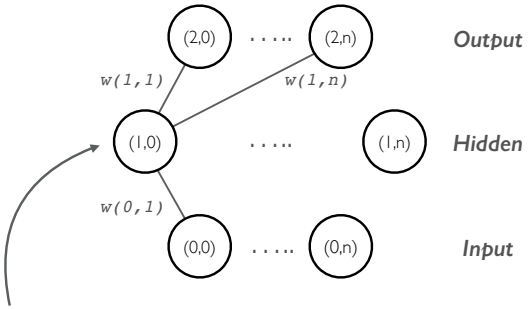

BACKPROP WEIGHT UPDATE

- Computing the error for the output layer nodes is straightforward (DESIRED-ACTUAL).
- However, there are no desired outputs for the MIDDLE layer nodes. So how do we calculate the ERROR in order to adjust the hidden weights?



BACKPROP WEIGHT UPDATE

- To calculate the error values for the hidden nodes we simply run the network backwards. That is we take the errors on the output nodes and back propagate them through the network. In much the same was as we propagated the input values forward.



Error(1,0) = Value(1,0) * (1.0 - Value(1,0)) * weighted_error

weighted_error = (Error(2,0)*w(1,1) + …+ Error(2,n)*w(1,n))

BACKPROP WEIGHT UPDATE

- There are two parameters which are crucial to the backprop algorithm, the Learning Rate (β) and the Momentum Term (α). The learning rate governs the speed at which the network learns.

- The learning rate value is chosen by the user and can be of critical importance in the training process, as too high of a value can lead to network instability and unsatisfactory training. Choose a value which is too low and it can cause excessively slow training. Typical values of (β) are usually between 0.1 and 0.75.

- The momentum coefficient α was introduced in 1986 by Rumelhart & McClelland as a method of producing a more stable network. It is an extra parameter that is added to the training algorithm so that adjustments are made in proportion to the previous weight changes of the network. A common value for is usually 0.9.

BACKPROP WEIGHT UPDATE

- Once the error for a node has been calculated we can adjust the weights on the layer below according to a slightly modified delta rule that includes the additional momentum term.

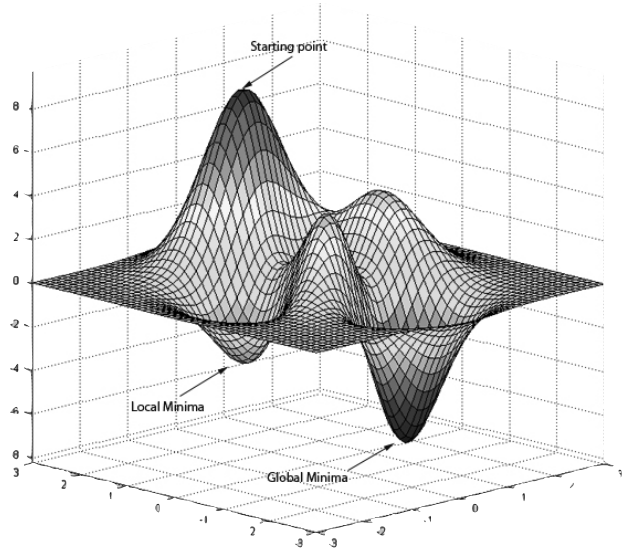
Weight_Delta(t+1) = β * Error(t+1) * Input_Value

Weight(t+1) = Weight(t) + Weight_Delta(t+1) + (α * Weight_Delta(t))

- As with the ADALINE network, the Backprop network is based in the idea of **gradient descent** or **steepest descent**. Ideally we would like the weights in the network to settle in a spot that represents a **global minima** for the problem being solved.

- Since the backprop learning algorithm can adjust weights in a multilayer network it is capable of solving problems that have complex error surfaces.

BACKPROP WEIGHT UPDATE



BACKPROP DEFICIENCIES

- Although the backprop algorithm has been successfully applied to a variety of problems it suffers from a number of deficiencies. One drawback is the long training process. This is usually caused by a non optimum learning rate and momentum term. More advanced algorithms apply an adaptive learning rate independently to each weight in the network.
- The addition of the momentum term helps the backprop algorithm overcome local minima. However, really complex problems can still cause the network to get stuck on sub optimal solutions.
- While the backprop algorithm is well understood and represents a very good starting point for solving problems with artificial neural networks, it has been surpassed by more efficient algorithms such as QuickProp (Scott E. Fahlman, 1988).
- There is no convergence theorem for backprop.

BACKPROP ALTERNATIVES

- The goal of backprop is to adjust a set weights given a set of example data for a given problem.
- So this is an optimisation problem.
- Therefore, we can use optimisation techniques such as **Genetic Algorithms** and **Particle Swarm Optimisation** to adjust network weights instead of backprop.
- In fact we can also use GA's and PSO to optimise the network topology (No. of hidden nodes and layers) and parameters such as the learning rate and momentum term.