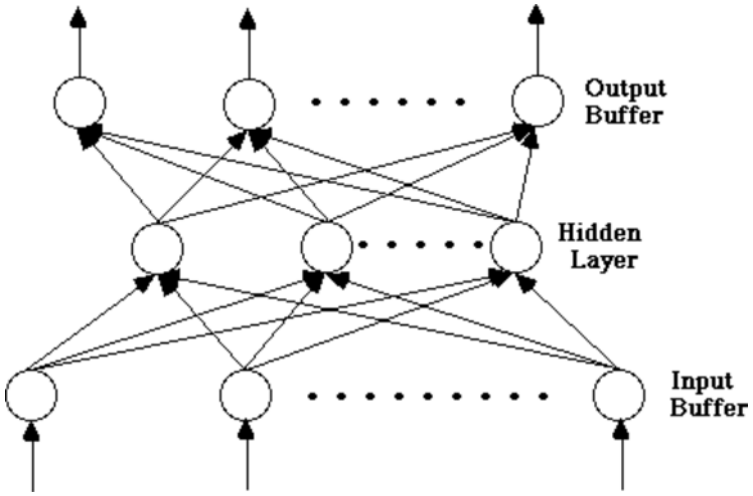
TECHNIQUES: THE MULTI-LAYER PERCEPTRON

- Learning
- Back propagation



The General Multi-Layer Perceptron

- one of the most important and widely used network models
- links together processing units into a network made up of layers
  - input (set by problem data)
  - output (of solution values)
  - typically one or two hidden layers (units model patterns in input data)
- Each layer is fully connected to the succeeding layer

The network is ***feedforward***. When using or testing a trained network, the input values set the values of elements in the first hidden layer, which influence the next layer, and so on until it sets values for the output layer elements. See a JavaScript model of the feedforward calculation in the Neural Planner diggers example.

What if a known input pattern produces a wildly incorrect output signal? Then we need to train the network, through a learning process.

Learning

in neural networks is done by changing the weighting factors (weights) at each element to reduce output errors. I've put together a simple JavaScript demonstration of learning in a single-layer perceptron.

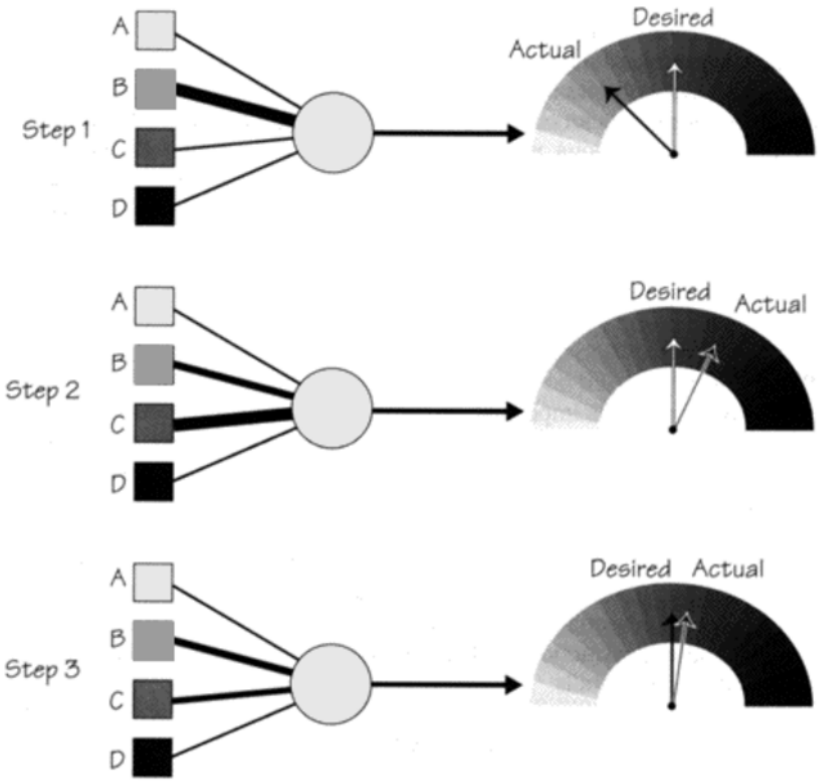


Figure 2.8 Neural networks—learning from experience.

In MLPs, learning is ***supervised***, with separate training and recall phases. For an example of how you train and then test a network, see Bob Mitchell's handwriting recognizer

During training the nodes in the hidden layers organise themselves such that different nodes learn to recognise different features of the total input space.

During the recall phase of operation the network will respond to inputs that exhibit features similar to those learned during training. Incomplete or noisy inputs may be completely recovered by the network.

In its learning phase, you give it a training set of examples with known inputs and outputs.

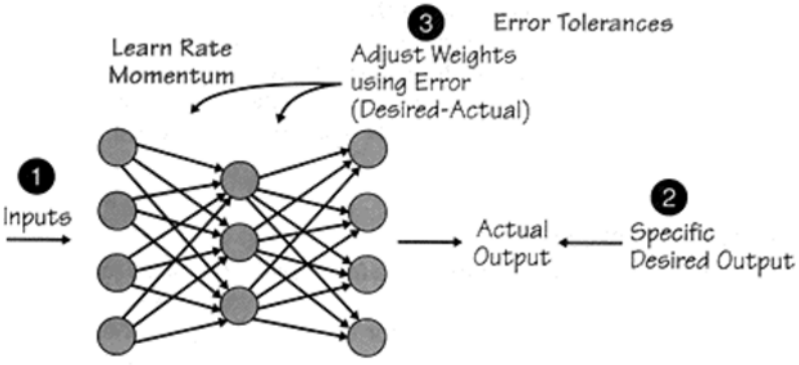


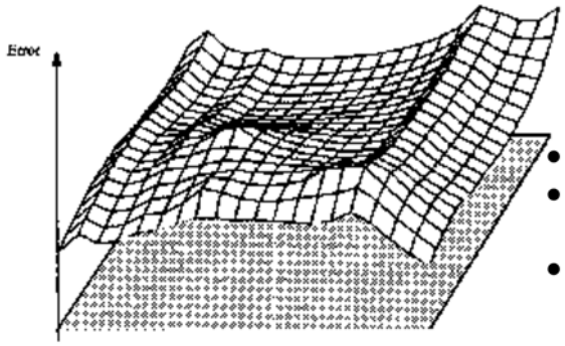
Figure 4.7 Back propagation networks.

1. For each input pattern, the network produces an output pattern.
2. It compares the actual output and the desired one from the training set and calculates an error.
3. It adjusts its weights a little to reduce the error (sliding down the slope).
4. It repeats 1-3 many times for every example in the training set until it has minimised the errors.

For a graphical visualisation of how a neural network gradually adjusts

itself to get closer and closer to the input patterns, see Jochen Fröhlich's Java simulation of a self-organising Kohonen feature map. (Note that this uses a different learning algorithm, not back propagation, but it still gradually gets closer to the training set values.)

There are many weights to be adjusted, so consider a multi-dimensional surface constructed by plotting the total network error in weight space (i.e. over all the possible changes in weight). A 3-D approximation could look like this:



During training:

- objective: find the global minimum on the error surface.
- solution: gradient descent, adjust weights to follow the steepest downhill slope.
- don't know surface in advance, so explore it in many small steps.

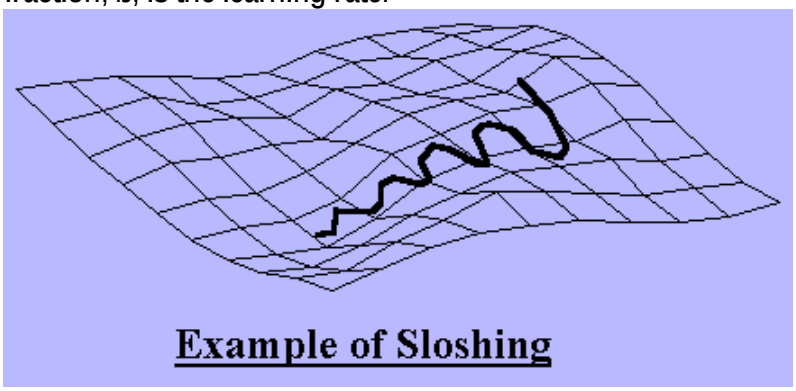
Back propagation

- During training, information is propagated back through the network and used to update connection weights. How?
- Different neural network architectures use different algorithms to calculate the weight changes.
- Backpropagation (BP) is a commonly used (but inefficient) algorithm in MLPs.
- We know the errors at the output layer, but not at the hidden layer elements.
- BP solves the problem of how to calculate the hidden layer errors (it propagates the output errors back to the previous layer using the output element weights).

The mathematics of this algorithm are given in several textbooks and on-line tutorials. For a detailed explanation of the back propagation algorithm, see Carling, Alison (1992) *Introducing Neural Networks*, Wilmslow: Sigma Press, pp. 147-154.

It helps to know some features of it when training neural networks.

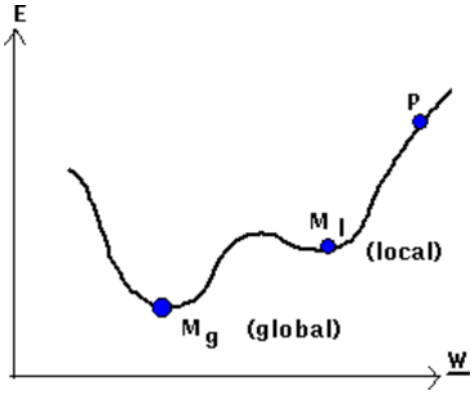
1. Internally most BP networks work with values between 0 and 1. If your inputs have a different range, NN simulators like Neural Planner will scale each input variable minimum to 0 and maximum to 1.
2. They change the weights each time by some fraction of the change needed to completely correct the error. This fraction, ß, is the learning rate.



   a. High learning rates cause the learning algorithm to take large steps on the error surface, with the risk of missing a minimum, or unstably oscillating across the error minimum ('sloshing')
   b. Small steps, from a low learning rate, eventually find a minimum, but they take a long time to get there.

c. Some NN simulators can be set to reduce the learning rate as the error decreases.
d. Also, sloshing can be reduced by mixing in to the weight change a proportion of the last weight change, so smoothing out small fluctutions. This proportion is the momentum term.

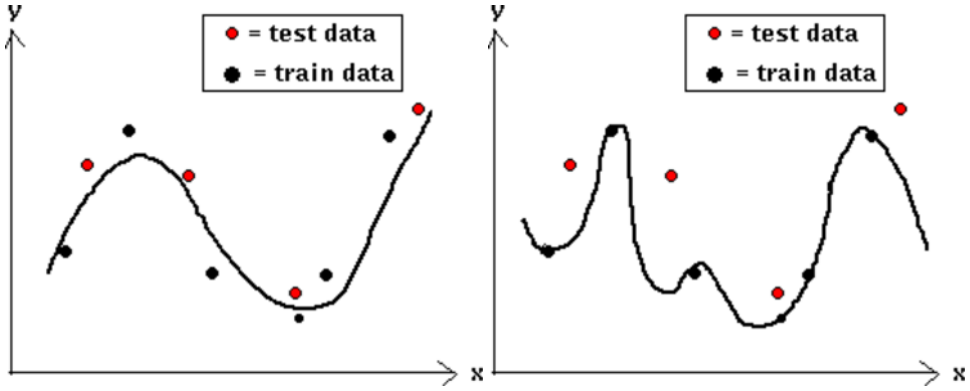
3. The algorithm finds the nearest local minimum, not always the lowest minimum.



One solution commonly used in backpropagation is to:

1. restart learning every so often from a new set of random weights (i.e. somewhere else in the weight space).
2. find the local minimum from each new start
3. keep track of the best minimum found

4. Overfitting is when the NN learns the specific details of the training set, instead of the general pattern found in all present and future data



There can be two causes:

a. Training for too long. Solution?
    1. Test against a separate test set every so often.
    2. Stop when the results on the test set start getting worse.

b. Too many hidden nodes
    - One node can model a linear function
    - More nodes can model higher-order functions, or more input patterns
    - Too many nodes model the training set too closely, preventing generalisation.

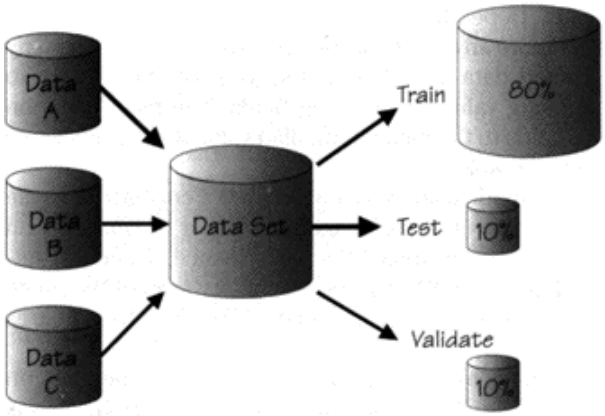


Figure 3.6 Data set management.

Learning parameters for Neural Networks

A summary of the parameters used in BP networks to control the learning behaviour.

Parameter	Models	Function
Learning rate	All	Controls the step size for weight adjustments. Decreases over time for some types of NN.
Momentum	Back propagation	Smooths the effect of weight adjustments over time.
Error tolerance	Back propagation	Specifies how close the output value must be to the desired value before the error is considered
Activation function	All	The function used at each neural processing unit to generate the output signal from the weighted average of inputs. Most common is the sigmoid function.

Adapted from: Joseph P. Biggus (1996) *Data mining with Neural Networks.* New York: McGraw-Hill, p. 82.

Prepared by Dr. David R. Newman.