Following on from Catastrophic interference, this section looks more closely at the nature of BP and why it is affected by CI. BP is compared against Cascade-Correlation, an algorithm that suffers much less from CI than BP.
There are several reasons why backpropagation is a popular choice for study and use.

It is easy to understand.

Many simulators are available.

It has been shown to approximate all ‘well-behaved’ functions and thus is often felt to qualify as a generic learning mechanism. In particular, it can learn non-linearly separable pattern sets.

It has only a few parameters and these are not very critical for the final results.

It possesses most of the basic elements of a ‘prototypical’ neural network: distributed representations, graceful degradation, pattern completion and adequate generalization of learned behavior.

but

The algorithm is often found to be very slow. The parameters may require a lot of tuning, especially the number of hidden units to use; and of course suffers profoundly from CI.
Moving Target Problem

1. Consider two tasks A & B

   —A = patterns of category A
   —B = patterns of category B

   —The error signal is thus constantly changing
   —I.e. target is always moving.


The moving target problem as it is called, refers really to the nature of training samples used in association with supervised training algorithms. Briefly, the desired output defined for each sample in the training set can be viewed as a target. As each sample is presented, the net is required to adjust weights to a different target.
The Herd Effect

1. When presented with A
   — All hidden units solve for A

2. When presented with B
   — All hidden units now try to solve for B

3. Sometimes called the herd effect.

When a category A sample is presented, an error signal is presented based on A. This error signal is passed back to all units and all weights. Likewise for B. One may expect that the hidden units will evolve into feature detectors some of which respond positively to category A patterns and others to category B patterns. This is however not simple, because when presented with A, all hidden units will try to move towards A and when presented with B they will all try to move towards B. There is no information available to the hidden units to help them select a particular category to solve for, other than the error signal passed back down from the output layer. The net is caught by indecision! Not surprisingly, this is sometimes called the herd effect and is further accentuated by the fact that the target is constantly changing.
Hidden Layer & Feature Detectors

1. Eventually hidden neurons will develop different behaviour

   —Some should develop detectors for task A

   —Some for task B

   —But how does a neuron decide which?

Hidden layer and feature detectors

The desired solution of course, is that the hidden layer units know how to decisively select which target they will attempt to solve. But how does a hidden unit decide which target? In the normal backpropagation algorithm there is no provision made to appoint assignments to individual units. The Cascade-Correlation algorithm however, solves this problem quite elegantly.
More on the Herd Effect

1. Explanation for the Herd Effect

   —Hidden units do not inter-communicate

   —Thus hidden units often *dance* around

   —Task is independently chosen

   —Random initial weights necessary.

Explanation for the *herd effect*

The main reason for the herd effect is that the hidden layer units cannot intercommunicate. There is no mechanism for these units to at least know not to try to solve for category A, if one of its neighbours has already selected A.

In backpropagation, this indecision is often seen down as a slow down in the learning process as the units dance around trying to settle on a solution. You may recall that when starting off a BP training sessions the weights of the MLP should be set to random initial weights. This is one reason why weights are often initialized to some small random values. The randomness breaks the symmetry which leads to neurons naturally exhibiting preference to the tasks. In fact, if random initial values are not used, backpropagation theoretically cannot converge! During the early epochs of the training cycle, this prevents the units of the net from behaving identically thus helping to mitigate the herd effect.
Cascade-Correlation

1. Developed by Fahlman and Lebiere (1990)

1. Key Differences from BP
   — Only output unit weights changed
   — Hidden units automatically inserted
   — Correlation used to train hidden weights
   — Once added, a hidden unit is frozen
   — Once single layer learning method is needed.

When comparing BP and CC, the following significant differences can be noted:

**Only output unit weights are changed**
The CC net is a multi-layer net just as with BP, but not all of the weights are trained at the same time. Once a hidden unit is added, only the output layer weights are trained. The hidden unit weights are left intact.

**Hidden units automatically added**
Training in CC commences with a single (output) layer net. If after training the output layer, the performance of the net is not deemed to be satisfactory (this can be done exactly as for BP), another hidden unit will be added to the net. Each new hidden unit is in fact a new layer rather than an addition to a single flat hidden layer.

**Correlation used to train hidden weights**
When selecting a hidden unit for inclusion into the net, a candidate unit is created which receives trainable weights from all inputs and all previously added hidden units. The output of the unit is not yet connected to the output layer. The hidden unit weights are trained in much the same way as the output layer, however the aim of the training is to correlate the performance of the unit with residual error at the output. This correlation is performed by maximizing the sum of the correlation between the candidate unit’s value and the residual output error observed at the output layer.

**Once added, the hidden unit is frozen**
Several hidden units may be available as candidates, but as soon as one is selected, the unit is added to the net and its weights are frozen. They will no longer change. This hidden unit becomes a permanent member of the network and will act as a feature detector. Although CC still has to deal with the moving target problem, the herd effect, a manifestation of the problem is avoided. (The downside to this is that if a hidden unit is selected that performs poorly, it added permanently to the net. Thus it is advantageous to work with several candidates, in parallel if possible, before updating the net.)
Adding Hidden Neurons

Adding Hidden Neurons

CC and CI

1. Consider two training sets $X$ & $Y$

1. Training with $X$
   — Hidden units created to handle set $X$

1. Training later with $Y$
   — Hidden units for $X$, unaffected
   — New ones added for set $Y$

Cascade-correlation and Catastrophic Interference

The CC algorithm is claimed to have some degree of robustness to CI. Consider two training sets $X$ & $Y$. $X$ and $Y$ represent two completely different training sets. They are to be used to train a net in sequence. In BP, following from our discussion of catastrophic interference, one would expect that the weights arrived at by BP would only be able to handle the last problem it was trained on.

In CC, something different happens. The net is trained on, say, set $X$ first using CC. As learning progresses, hidden units will be added to the net and frozen. Learning will cease when the net is observed to perform to some prescribed criteria. The hidden units will represent a set of feature detectors which were found by the algorithm useful in solving $X$. The net is later trained on set $Y$. In BP, all weights would be re-adjusted to solve for $Y$, consequently disrupting all knowledge of the previous task, $X$. In CC, that cannot happen because the hidden unit weights are all frozen. Units were added to help solve $X$ and they will remain intact. As the net is trained on $Y$, additional hidden units will be added which provide the necessary feature detectors required to solve $Y$. 

The net has now learned set Y, but what has become of its ability to handle set X patterns? Only one set of output weights for X and Y. There is only one set of output weights and just like in BP, when the second set is presented, these will change and it is likely that this will affect performance on set X in favour of performance on set Y. However, the hidden units that were learned for set X are still part of the net. To force the net to learn both X and Y, it should be possible to train again using a combined training set that includes samples from both X and Y. This should theoretically be easy and may not even involve the addition of further hidden units. After all, the hidden units to solve for both tasks are already part of the net. Of course, it is also possible to simply train BP with a combined X+Y training set, but this would be expected to take much longer than the approach adopted by the CC algorithm.