Consider some sort of network connectivity model with outputs F_i. Let our total error function look like

$$E=\frac{1}{2}\sum_I \left(T_I-F_I\right)^2$$

where

F_I - Output Activation (what we got): I is the target node index
T_I - Target Output Activation (what we want): I is the target-node index
E - Error function for a single training-set model

Where the ½ is there to remove an unneeded factor of two from the derivative and we want to find the minimum with respect to all the weights w (index labels α, β) for all the layers (index label γ), $w_{\alpha\beta\gamma}$, across all the training set data. I am using Greek index variables so as to not confuse this general indexing with the more specific version to be used later. To find a minimum we generally would take the gradient with respect to $w_{\alpha\beta\gamma}$, set it equal to zero and then solve for all the weights through the simultaneous equations. The problem is that there may not be a true zero. In general, we need to find the minimum in E with respect to the weights. We thus have a minimization problem. There are many ways to find a minimum in an N-dimensional phase space. The simplest approach is gradient (steepest) decent, where we go "down hill" in weight space. We modify each weight by the negative derivative and put in a "learning factor", which I am calling λ, to allow us to control how far things can move down hill in each step:

$$\Delta w_{\alpha\beta\gamma}=-\lambda\frac{\partial E}{\partial w_{\alpha\beta\gamma}}.$$

Note that you can also make λ adaptive by making it dependent on the value of the error function E or the individual weights. You could simply set $\lambda = E/10$ for example.

You could also apply the **Newton–Raphson method** to converge on the correct weights. For a single variable function we can find a zero using

$$x^1=x^0-\frac{f(x)}{f'(x)},$$

so if the error function is E then we can find the minimum by looking for zeros in the first derivative (which is therefore the $f(x)$ used in the method) to locate a minimum:

$$w^1_{\alpha\beta\gamma}=w^0_{\alpha\beta\gamma}-\frac{\partial E/\partial w_{\alpha\beta\gamma}}{\partial^2 E/\partial w^2_{\alpha\beta\gamma}}.$$

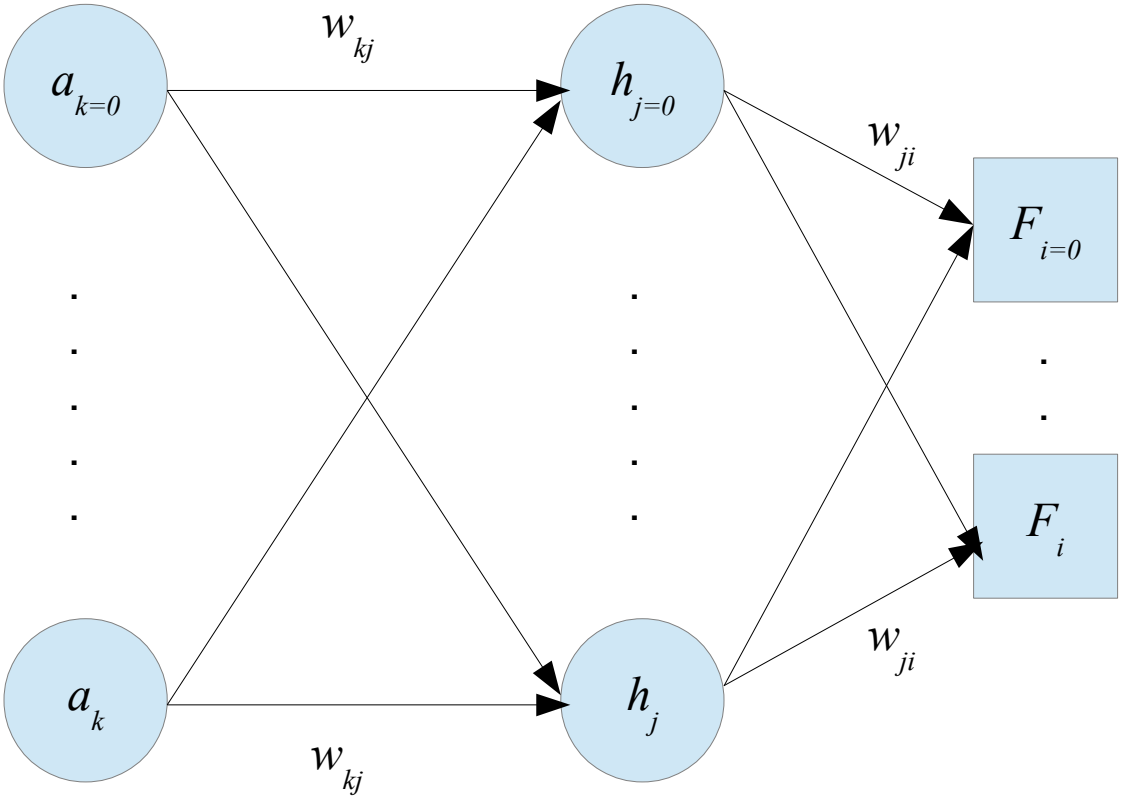
Or you could use the method of Davidon-Fletcher-Powell

http://en.wikipedia.org/wiki/Davidon-Fletcher-Powell_formula.

The problem with these methods is that you must calculate the error function three times for each weight because of the derivative, which is computationally intensive. If you have a fast system, then it is not a problem. On the other hand, although they might be slow they can be used for ANY connectivity model.

For all these methods, at first it looks like we still need to make three determinations of the error function for each weight because we have the derivative to deal with, but with a judicious choice of the connectivity model and activation rule along with a little grunt work shows how we can optimize the learning into a "back propagation" algorithm which significantly reduces the amount of computation. Back propagation is just an optimized version of steepest decent, so we will first need to implement steepest decent and then later optimize it.

We need to formalize our notation and our connectivity model. The initial use of our index letters was to help see the relations between the elements as they moved through the network. We now want to modify our notation to view things from right to left (yes, backwards trough the layers) and start our index lettering on the output side:



What we want to know is how the error depends on the weights in each layer so we can determine how to modify the weights to minimize the error. We are going to limit our derivation to a network with only a single hidden activation layer which I shall call a two-layer network since it has two layers of weights. The hidden layer is fully connected to both its input and output layer.

We start with the expressions for the output values and an error function which is now simply defined for only a single training event. $F_i = f\left(\sum_J h_J w_{Ji}\right)$, $E = \frac{1}{2}\sum_I \left(T_I - F_I\right)^2$, where $f(x)$ is the activation function. We should also note that by symmetry, the value of the hidden activation nodes looks a lot like the output activations in that $h_j = f\left(\sum_K a_K w_{Kj}\right)$.

First we want to see the effects of the error on the right-most set of weights coming out of the hidden layer (the *ji* weights)

$$\frac{\partial E}{\partial w_{ji}} = \sum_I \left[\left(T_I - F_I\right)\frac{\partial\left(T_I - F_I\right)}{\partial w_{ji}}\right].$$

The T_i values are constants so we have

$$\frac{\partial E}{\partial w_{ji}} = -\sum_I \left[\left(T_I - F_I\right)\frac{\partial F_I}{\partial w_{ji}}\right],$$

but we know that $F_i = f\left(\sum_J h_J w_{Ji}\right)$, so we have

$$\frac{\partial E}{\partial w_{ji}} = -\sum_I \left[\left(T_I - F_I\right)\frac{\partial f\left(\sum_J h_J w_{JI}\right)}{\partial w_{ji}}\right],$$

or

$$\frac{\partial E}{\partial w_{ji}} = -\sum_I \left[\left(T_I - F_I\right) f'\left(\sum_J h_J w_{JI}\right)\frac{\partial \sum_J h_J w_{JI}}{\partial w_{ji}}\right].$$

The values of the activations in the hidden layer, h_j, are not dependent on the values of the weights to the right that feed into the output layer, w_{ji}, so the h_j terms look like constants to the derivative. We also need to note that we are going to differentiate for a single value of the index variables *i* and *j* on both sides, and since there are no cross-dependencies in our connectivity model then both the summations outside of the activation function reduce to only one term since all the other derivative terms will be zero, so we can see that $\frac{\partial \sum_J h_J w_{JI}}{\partial w_{ji}} = h_j$, when $I = i$ and $J = j$ and is zero otherwise, so we have

$$\boxed{\frac{\partial E}{\partial w_{ji}} = -\left(T_i - F_i\right) f'\left(\sum_J h_J w_{Ji}\right) h_j .}$$

Now let's move to the left most connectivity layer. We want to see the effects of the error on the first set of weights coming out of the initial input layer (the *kj* weights):

$$\frac{\partial E}{\partial w_{kj}} = \sum_I \left[\left(T_I - F_I \right) \frac{\partial \left(T_I - F_I \right)}{\partial w_{kj}} \right].$$

Once again, the T_I values are constants so we have

$$\frac{\partial E}{\partial w_{kj}} = -\sum_i \left[\left(T_I - F_I \right) \frac{\partial F_I}{\partial w_{kj}} \right],$$

but we know that $F_i = f\left(\sum_J h_J w_{Ji} \right)$, so we have

$$\frac{\partial E}{\partial w_{kj}} = -\sum_I \left[\left(T_I - F_I \right) \frac{\partial f\left(\sum_J h_J w_{JI} \right)}{\partial w_{kj}} \right],$$

so

$$\frac{\partial E}{\partial w_{kj}} = -\sum_I \left[\left(T_I - F_I \right) f'\left(\sum_J h_J w_{JI} \right) \frac{\partial \sum_J h_J w_{JI}}{\partial w_{kj}} \right].$$

In the previous case we only had the *i* and *j* index values on both sides, so only a single differential element would survive the summations. In this case however, we have the *k* index and there are dependencies of h_j on w_k, so we cannot drop the all the sums up front this time. Define $\psi_i = \left(T_i - F_i \right) f'\left(\sum_J h_J w_{Ji} \right)$, so we have

$$\frac{\partial E}{\partial w_{kj}} = -\sum_I \left[\psi_I \frac{\partial \sum_J h_J w_{JI}}{\partial w_{kj}} \right].$$

The weights on the right connectivity layer are independent of the weights in the left connectivity layer and each activation h_j will depend on only one w_k for the matching values of *j*, so the right summation reduces to a single term, therefore

$$\frac{\partial \sum_J h_J w_{Ji}}{\partial w_{kj}} = w_{ji} \frac{\partial h_j}{\partial w_{kj}}, \text{ so we have}$$

$$\frac{\partial E}{\partial w_{kj}} = -\sum_I \psi_I w_{jI} \frac{\partial h_j}{\partial w_{kj}},$$

but we know from the diagram that $h_j = f\left(\sum_K a_K w_{Kj}\right)$, so we have

$$\frac{\partial E}{\partial w_{kj}} = -\sum_I \psi_I w_{jI} \frac{\partial f\left(\sum_K a_K w_{Kj}\right)}{\partial w_{kj}},$$

or

$$\frac{\partial E}{\partial w_{kj}} = -\sum_I \psi_I w_{jI} f'\left(\sum_K a_K w_{Kj}\right) \frac{\partial \sum_K a_K w_{Kj}}{\partial w_{kj}}.$$

We have seen things like these relations before, and we should see that

$$\frac{\partial \sum_K a_K w_{Kj}}{\partial w_{kj}} = a_k,$$

so

$$\frac{\partial E}{\partial w_{kj}} = -\sum_I \psi_I w_{jI} f'\left(\sum_K a_K w_{Kj}\right) a_k,$$

or after pulling terms outside the summation over *i* and recalling that $\psi_i = \left(T_i - F_i\right) f'\left(\sum_J h_J w_{Ji}\right)$, we have

$$\frac{\partial E}{\partial w_{kj}} = -a_k f'\left(\sum_K a_K w_{Kj}\right) \sum_I \left[\left(T_I - F_I\right) f'\left(\sum_J h_J w_{JI}\right) w_{jI}\right].$$

You can now use steepest descent, $\Delta w_{\alpha\beta\gamma} = -\lambda \frac{\partial E}{\partial w_{\alpha\beta\gamma}}$, with some learning factor.

The Activation Function

To get *f*′(*x*) let us use a simple sigmoid so that $f(x) = \frac{1}{1+e^{-x}}$. In that case:

$$f'(x) = \frac{d}{dx} f(x) = \frac{d}{dx}\left(1+e^{-x}\right)^{-1} = -\left(1+e^{-x}\right)^{-2} e^{-x} = -f^2(x) \frac{d}{dx} e^{-x} = f^2(x) e^{-x},$$

but we can write $e^{-x} = \frac{1}{f(x)} - 1$, so $\frac{d}{dx} f(x) = f^2(x)\left(\frac{1}{f(x)} - 1\right) = f(x)\left(1 - f(x)\right)$,

so we finally have $f'(x) = f(x)\left(1 - f(x)\right)$ if $f(x) = \frac{1}{1+e^{-x}}$.