Backpropagation

Now we've come to the problem of how to make a multilayer neural network *learn*. Before, we saw how to update weights with gradient descent. The backpropagation algorithm is just an extension of that, using the chain rule to find the error with the respect to the weights connecting the input layer to the hidden layer (for a two layer network).

To update the weights to hidden layers using gradient descent, you need to know how much error each of the hidden units contributed to the final output. Since the output of a layer is determined by the weights between layers, the error resulting from units is scaled by the weights going forward through the network. Since we know the error at the output, we can use the weights to work backwards to hidden layers.

For example, in the output layer, you have errors δ_k^o attributed to each output unit k. Then, the error attributed to hidden unit j is the output errors, scaled by the weights between the output and hidden layers (and the gradient):

$$\delta_j^h = \sum W_{jk} \delta_k^o f'(h_j)$$