Choosing the Right Weights...Iteratively!

Choosing the right weights directly is too difficult. An alternative approach is to iteratively improve the weights by descending the error function, taking small steps. Each step is taken in the direction of the greatest downward slope from your current position.

Now let's move forward! If you've had your coffee, you may have realized this means the error function didn't need to sum over all the output nodes in the first place. We've seen the reason is that the output of a node only depends on the connected links and hence their weights. This is often glossed over in many texts which simply state the error function without explaining it. Anyway, we now have a simpler expression.

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} (t_k - o_k)^2$$

Now we will do a bit of calculus. Remember, you can refer to the Appendix if you're unfamiliar with differentiation. That t_k part is a constant, and so doesn't vary as w_{jk} varies. That is t_k isn't a function of w_{jk}. If you think about it, it would be really strange if the truth examples which provide the target values did change depending on the weights! That leaves the o_k part which we know does depend on w_{jk} because the weights are used to feed forward the signal to become the outputs o_k. We'll use the chain rule to break apart this differentiation task into more manageable pieces. Again refer to the *Appendix* for an introduction to the chain rule.

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial w_{jk}}$$

Now we can attack each simpler bit in turn. The first bit is easy as we're taking a simple derivative of a squared function. This gives us the following:

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot \frac{\partial o_k}{\partial w_{jk}}$$

The second bit needs a bit more thought, but not too much. That o_k is the output of the node k which, if you remember, is the sigmoid function applied to the weighted sum of the connected incoming signals. So let's write that out to make it clear.

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot \frac{\partial}{\partial w_{jk}} sigmoid\left(\sum_j w_{jk} \cdot o_j\right)$$

That o_j is the output from the previously hidden layer node, not the output from the final layer o_k. How do we differentiate the Sigmoid function? We could do it the long hard way, using the fundamental ideas in the Appendix, but others have already done that work. We can just use the well-known answer, just like mathematicians all over the world do every day.

$$\frac{\partial}{\partial x} sigmoid(x) = sigmoid(x)\left(1 - sigmoid(x)\right)$$

Some functions turn into horrible expressions when you differentiate them. This sigmoid has a nice simple and easy to use the result. It's one of the reasons the sigmoid is popular for activation functions in neural networks. So let's apply this cool result to get the following.

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot sigmoid\left(\sum_j w_{jk} \cdot o_j\right)\left(1 - sigmoid\left(\sum_j w_{jk} \cdot o_j\right)\right) \cdot \frac{\partial}{\partial w_{jk}}\left(\sum_j w_{jk} \cdot o_j\right)$$

$$= -2(t_k - o_k) \cdot sigmoid\left(\sum_j w_{jk} \cdot o_j\right)\left(1 - sigmoid\left(\sum_j w_{jk} \cdot o_j\right)\right) \cdot o_j$$

What's that extra last bit? It's the chain rule again applied to the sigmoid

derivative because the expression inside the sigmoid() function also needs to

be differentiated with respect to w_{jk}. That too is easy, and the answer is simply o_j. Before we write down the final answer, let's get rid of that **2** at the front. We can do that because we're only interested in the direction of the slope of the error function so we can descend it. It doesn't matter if there is a constant factor of **2**, **3** or even **100** in front of that expression, as long we're consistent about which one we stick to. So let's get rid of it to keep things simple. Here's the final answer we've been working towards, the one that describes the slope of the error function so we can adjust the weight w_{jk}.

$$\frac{\partial E}{\partial w_{jk}} = -(t_k - o_k) \cdot sigmoid(\Sigma_j w_{jk} \cdot o_j)(1 - sigmoid(\Sigma_j w_{jk} \cdot o_j)) \cdot o_j$$

Phew! We did it! That's the magic expression we have been looking for. The key to training neural networks. It's worth a second look, and the color coding helps show each part. The first part is simply the $(target - actual)$ error we know so well. The sum expression inside the sigmoids is simply the signal into the final layer node; we could have called it i_k to make it look simpler. It's just the signal into a node before the activation squashing function is applied. That last part is the output from the previous hidden layer node *j*. It is worth viewing these expressions in these terms because you get a feel for what physically is involved in that slope, and ultimately the refinement of the weights. That's a fantastic result, and we should be really pleased with ourselves. Many people find getting to this point really hard.