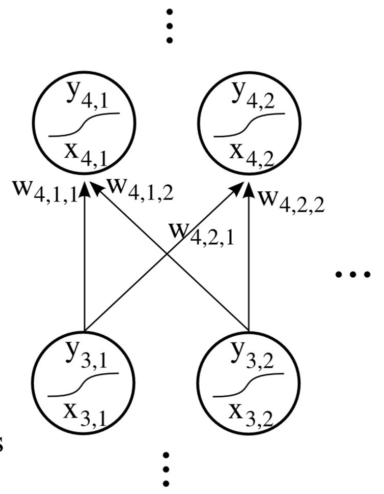
A thorough derivation of back-propagation

for people who *really* want to understand it
by: Mike Gashler, September 2010

Define the problem:
Suppose we have a 5-layer feed-forward neural network. (I intentionally made it big so that certain repeating patterns will be obvious.)



I will refer to the input pattern as "layer 0". Thus, layer 1 is the first hidden layer in the network in feed-forward order. Layer 5 is the output layer. y_{ij} is the output of layer *i*, node *j*. So, y_{0j} is input *j* into the network. $y_{5,j}$ is output *j* of the network. f_i is the activation function used in layer *i*. w_{ijk} is the weight that feeds into layer *i*, node *j*, from node *k* of the previous layer. x_{ij} is the net value that feeds into layer *i*, node *j*. So, $x_{ij} = \sum_k w_{ijk} y_{i-1,k}$ and $y_{i,j} = f(x_{i,j})$. f_i' is the derivative of f_i.

Our goal is to compute the partial derivative of the sum-squared error with respect to the weights. (We'll skip the explanation of why sum-squared error is the maximum likelihood estimator given the assumption of Gaussian noise.) For this example, we will arbitrarily compute the gradient for the weight feeding into layer 1, node 7 from layer 0 (the input layer), node 3.

A Quick Calculus review:
There are really only two big concepts from calculus that you need to remember. The first is the chain rule. To take the derivative of a function, do this: $\frac{\partial f(x)}{\partial y} \rightarrow \frac{f'(x)\partial x}{\partial y}$. You might remember this as "the derivative of the function times the derivative of the inside". The second thing you need to remember is that constant terms can be dropped from a derivative. For example, if *y* is a function of *z*, and *x* is not a function of *z*, then you can do this: $\frac{\partial(x+y)}{\partial z} \rightarrow \frac{\partial y}{\partial z}$. You'll also need to remember basic algebra, but I will not review that in this document.

Okay, let's begin:

1. Given. $$\frac{\partial \frac{1}{2}\sum_i (t_i - y_{5i})^2}{\partial w_{173}}$$

In English, this means "the partial derivative of the sum-squared-error with respect to the weight to layer 1 node 7, from layer 0 node 3. In order to implement it (efficiently), we need to get rid of the "∂" symbol. So, we're going to do a bunch of math in order to get it out of the formula.

2. Apply the chain rule. $$= \frac{\sum_i (t_i - y_{5i})\partial(t_i - y_{5i})}{\partial w_{173}}$$

3. Simplify. $$= \frac{-\sum_i (t_i - y_{5i})\partial y_{5i}}{\partial w_{173}}$$

4. Expand a term. $$= \frac{-\sum_i (t_i - y_{5i})\partial f_5(x_{5i})}{\partial w_{173}}$$

5. Apply the chain rule. $= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \partial x_{5i}}{\partial w_{173}}$

6. Expand a term. $= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \partial \sum_j w_{5ij} y_{4j}}{\partial w_{173}}$

7. Simplify. $= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} \partial y_{4j}}{\partial w_{173}}$

8. Expand a term. $= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} \partial f_4(x_{4j})}{\partial w_{173}}$

9. Repeat steps 5 through 8. $= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} \partial f_3(x_{3k})}{\partial w_{173}}$

10. Repeat steps 5 through 8.

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} \partial f_2(x_{2l})}{\partial w_{173}}$$

11. It should be clear that we could keep doing this (repeating steps 5 through 8) for an arbitrarily deep network. But now we're approaching the layer of interest, so we'll slow down again. Now we'll apply the chain rule (as in step 5.)

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) \partial x_{2l}}{\partial w_{173}}$$

12. Expand a term (as in step 6).

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) \partial \sum_m w_{2lm} y_{1m}}{\partial w_{173}}$$

13. Simplify. (All of the terms in the summation over *m* evaluate to constants, except when *m*=7.)

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) \partial w_{2l7} y_{17}}{\partial w_{173}}$$

14. Simplify (as in step 7).

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} \partial y_{17}}{\partial w_{173}}$$

15. Expand a term (as in step 8).

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} \partial f_1(x_{17})}{\partial w_{173}}$$

16. Apply the chain rule (as in step 5).

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} f_1'(x_{17}) \partial x_{17}}{\partial w_{173}}$$

17. Expand a term (as in step 6).

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} f_1'(x_{17}) \partial \sum_n w_{17n} y_{0n}}{\partial w_{173}}$$

18. Simplify. (All of the terms in the summation over *n* evaluate to constants, except when *n*=3.)

$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} f_1'(x_{17}) \partial w_{173} y_{03}}{\partial w_{173}}$$

19. Simplify.

$$= -\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} f_1'(x_{17}) y_{03}$$

20. We now have a formula to compute the gradient with respect to a certain weight. We could simply use this formula to compute the gradient for all of the weights, but that would involve a lot of redundant computation. To implement it efficiently, we need to cache the redundant parts of the formula. So, let $e_{5i} = (t_i - y_{5i}) f_5'(x_{5i})$. This is a value that we will cache because we'll use it again when computing the gradient for other weights. This is referred to as the "error" term on the nodes in the output layer.

$$= -\sum_i e_{5i} \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} f_1'(x_{17}) y_{03}$$

21. Let $e_{4j} = \sum_i e_{5i} w_{5ij} f_4'(x_{4j})$. (Now we're computing the "error term" of a node in a hidden layer.)

$$= -\sum_j e_{4j} \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} f_1'(x_{17}) y_{03}$$

22. Let $e_{3k} = \sum_j e_{4j} w_{4jk} f_3'(x_{3k})$. (This is just repeating step 21 to compute the "error terms" in the next layer.)

$$= -\sum_k e_{3k} \sum_l w_{3kl} f_2'(x_{2l}) w_{2l7} f_1'(x_{17}) y_{03}$$

23. Let $e_{2l} = \sum_k e_{3k} w_{3kl} f_2'(x_{2l})$. (This is repeating step 21 again for the next layer.)

$$= -\sum_l e_{2l} w_{2l7} f_1'(x_{17}) y_{03}$$

24. Let $e_{17} = \sum_l e_{2l} w_{2l7} f_1'(x_{17})$. (This is repeating step 21 again, but only for a particular node.)

$$= -e_{17} y_{03}$$

And thus we see that the gradient of the error with respect to a particular weight in a neural network is simply the negative error of the node into which that weight feeds times the value that feeds in through the weight. If you implement steps 20 and 21, you will find that you have implemented back-propagation. All that remains is to discuss how we update weights. Since the gradient points in the direction that makes error bigger, we want to subtract the gradient direction from the weights. Also, it would be too big of a step if we subtracted the whole gradient, so we multiply it by a learning rate in order to take small steps.

Okay, now let's compute the gradient with respect to the inputs (instead of the weights), just for fun.

1. Given. $$\frac{\partial \frac{1}{2}\sum_i (t_i - y_{5i})^2}{\partial y_{05}}$$

2. Chain rule. $$= \frac{\sum_i (t_i - y_{5i}) \partial (t_i - y_{5i})}{\partial y_{05}}$$

3. Simplify. $$= \frac{-\sum_i (t_i - y_{5i}) \partial y_{5i}}{\partial y_{05}}$$

4. Expand. $$= \frac{-\sum_i (t_i - y_{5i}) \partial f_5(x_{5i})}{\partial y_{05}}$$

5. Chain rule. $$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \partial x_{5i}}{\partial y_{05}}$$

6. Expand. $$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \partial \sum_j w_{5ij} y_{4j}}{\partial y_{05}}$$

7. Simplify. $$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} \partial y_{4j}}{\partial y_{05}}$$

8. Expand. $$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} \partial f_4(x_{4j})}{\partial y_{05}}$$

9. Steps 5-8. $$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} \partial f_3(x_{3k})}{\partial y_{05}}$$

10. Steps 5-8. $$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} \partial f_2(x_{2l})}{\partial y_{05}}$$

11. Steps 5-8.
$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{1l}) \sum_m w_{2lm} \partial f_1(x_{1m})}{\partial y_{05}}$$

12. Chain rule.
$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{1l}) \sum_m w_{2lm} f_1'(x_{1m}) \partial x_{1m}}{\partial y_{05}}$$

13. Expand.
$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{1l}) \sum_m w_{2lm} f_1'(x_{1m}) \partial \sum_n w_{1mn} y_{0n}}{\partial y_{05}}$$

14. Simplify.
$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{1l}) \sum_m w_{2lm} f_1'(x_{1m}) \partial w_{1m5} y_{05}}{\partial y_{05}}$$

15. Simplify.
$$= \frac{-\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{1l}) \sum_m w_{2lm} f_1'(x_{1m}) w_{1m5} \partial y_{05}}{\partial y_{05}}$$

16. Simplify.

$$= -\sum_i (t_i - y_{5i}) f_5'(x_{5i}) \sum_j w_{5ij} f_4'(x_{4j}) \sum_k w_{4jk} f_3'(x_{3k}) \sum_l w_{3kl} f_2'(x_{1l}) \sum_m w_{2lm} f_1'(x_{1m}) w_{1m5}$$

17. Now we just need to turn this into an algorithm. We can save ourselves a lot of work by observing that this formula is almost identical to the one in step 19 of the derivation of back-propagation. We can, therefore, use the work we've already done to jump to the following result:

$$= -\sum_m e_{1m} w_{1m5}$$

And thus we see that back-propagation computes an “error” term for each node in the network, and these “error” terms are useful for computing the gradient with respect to either the weights or the inputs. Further, by subtracting a small portion of that gradient, we can move the system closer to the target values.