We use the squared error function

\[ E = \frac{1}{2} \sum_{p=1}^{N} (y^p - t^p)^2 \]

(superscript p refers to pattern p, we will drop this notation below)
Remember the gradient vector points in the direction of increasing function value.
The gradient descent learning rule is

\[ w_1(\text{new}) = w_1(\text{old}) - \eta \frac{\partial E}{\partial w_1} \]

\[ y_3 = f(x_1 \ast w_{31} + x_2 \ast w_{32} + b_3) = f(\text{net}_3) \]

The \( f \) function is called the activation function. We will assume that \( f(x) = \sigma(x) \) for the rest of these notes.

(\text{the below summation is over pattern presentations (superscripts dropped)})

\[
\frac{\partial E}{\partial w_{31}} = \sum \frac{\partial E}{\partial y_3} \frac{\partial y_3}{\partial w_{31}} = \sum \frac{\partial E}{\partial y_3} \frac{\partial \text{net}_3}{\partial w_{31}} = \sum \frac{\partial E}{\partial y_3} \frac{\partial \text{net}_3}{\partial w_{31}}
\]

\[ = \sum (y_3 - t_3) y_3 (1 - y_3) x_1 = \sum (y_3 - t_3) y_3 (1 - y_3) x_1 \]

\[ = \sum_{p=1}^{P} \delta_3 \ast x_1 = \sum_{p=1}^{P} -\delta_3 \ast x_1 \]

We define

\[ -\frac{\partial E}{\partial \text{net}_j} = \delta_j \]
1 Multilayer Network

\[ y_5 = f(\text{net}_5) \]
\[ = f(w_{53} \times y_3 + w_{54} \times y_4 + b_5) \]
\[ = f(w_{53} \times f(w_{31} \times y_1 + w_{32} \times y_2 + b_3) + w_{54} \times f(w_{41} \times y_1 + w_{42} \times y_2 + b_4) + b_5) \]

\[ E = \frac{1}{2} \sum_{p=1}^{P} (y^p - t^p)^2 \]

\[ \frac{\partial E}{\partial w_{53}} = \sum_p (y_5 - t_5)y_5(1 - y_5)y_3 \]

(Aalogous to the one layer net above)

For the next layer

\[ \frac{\partial E}{\partial w_{31}} = \sum_p \frac{\partial E}{\partial y_5} \frac{\partial y_5}{\partial \text{net}_5} \frac{\partial \text{net}_5}{\partial y_3} \frac{\partial y_3}{\partial \text{net}_3} \frac{\partial \text{net}_3}{\partial w_{31}} \]
\[ = \sum_p (y_5 - t_5)y_5(1 - y_5)w_{53}(y_3)(1 - y_3)y_1 \]
\[ = \sum_p \delta_5 w_{53}f'(\text{net}_3)y_1 \]
If there are multiple output units then we must sum over the back-propagated contribution from each higher level unit (k) that it sends connections too.

Consider a situation where a unit j projects to several higher level units (indexed by k)

\[ \delta_j = - \frac{\partial E}{\partial net_j} = \sum_k \delta_k \frac{\partial net_k}{\partial net_j} \]

\[ = \sum_k \delta_k w_{kj} y_j (1 - y_j) \]

\[ = f'(net_j) \sum_k w_{kj} \delta_k \]

Therefore \( \delta \)'s can be propagated backwards just like activations (y's) are propagated forward. (make sure you can see this from the above equation) (This is why the algorithm is called back-propagation.) This makes the algorithm quite efficient (you don’t have to recompute the effect of all the higher weights/units on the error derivatives for the lower weights) Therefore the batch weight update rule (accumulated for each pattern and then summed and implemented at once) is

\[ w_{ji}(new) = w_{ji}(old) + \eta \sum_p \delta_j y_i \]

\[ w_{ji}(new) = w_{ji}(old) + \eta \delta_j y_i \]

Or an update of

\[ w_{ji}(new) = w_{ji}(old) + \eta \delta_j y_i \]

After each pattern presentation for the on-line version

Basic back-propagation algorithm is
• propagate forward activations
• compute error
• back-propagate deltas
• update weights
• repeat for next pattern