Derivation of Backpropagation Equations

Jesse Hoey
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, CANADA, N2L3G1
jhoey@cs.uwaterloo.ca

In this note, I consider a feedforward deep network comprised of L layers, interleaved complete linear layers and activation layers (e.g. sigmoid or rectified linear layers). I follow the convention adopted in Poole and Mackworth [2] that linear layers and activation layers are kept separate. Usually, treatments of backpropagation lump the two into a single layer [1, 3], but this way makes the exposition easier.

Thus, the complete linear layer k has $N^k + 1$ input units $input^k[0 \dots N^k]$ and M^k output units $output^k[1 \dots M^k]$, and each node $j \in output^k$ is connected to each node $i \in input^k$ with weight w^k_{ji} (superscripts are indices, not a power). Thus, we can compute $output^k[j] = \sum_{i=0}^{N^k} w^k_{ji} input^k[i]$, assuming $input^k[0] = 1$ for each linear layer. The activation layer l has N^l inputs $input^l$ and N^l outputs $output^l$, and each node $j \in output^l$ is connected to one node $j \in intput^l$ through an activation function f^l, such that $output^l = f^l(input^l)$. The layers are interleaved, such that a linear layer's $input^k$ is the same as the next lower activation layer's $output^{k-1}$, and the activation layer's $input^l$ is the same as the next lower linear layer's $output^{l-1}$. The input features are fed to the linear layer $input^0 = X(e)$, and the output (target) features are extracted from an activation layer $\hat{Y}(e) = output^L$. Therefore, we have that

$$
\begin{aligned}
\hat{Y}(e)[j] &= output^L[j] \\
&= f^L(input^L[j]) \\
&= f^L(output^{L-1}[j]) \\
&= f^L\left(\sum_i w^{L-1}_{ji} input^{L-1}[i]\right) \qquad (1) \\
&= f^L\left(\sum_i w^{L-1}_{ji} f^{L-2}(input^{L-2}[i])\right) \\
&= f^L\left(\sum_i w^{L-1}_{ji} f^{L-2}\left(\sum_k w^{L-3}_{ik} input^{L-3}[k]\right)\right) \\
&= \dots \\
&= f^L\left(\sum_i w^{L-1}_{ji} f^{L-2}\left(\sum_k w^{L-3}_{ik} f^{L-4}\left(\dots f^1\left(\sum_m w^0_{lm} input^0[m]\right)\dots\right)\right)\right) \\
&= f^L\left(\sum_i w^{L-1}_{ji} f^{L-2}\left(\sum_k w^{L-3}_{ik} f^{L-4}\left(\dots f^1\left(\sum_m w^0_{lm} X(e)[m]\right)\dots\right)\right)\right) \qquad (2)
\end{aligned}
$$

Now, we wish to evaluate the error and use stochastic gradient descent to compute updates to the weights. The squared error on example e, with weights w is:

$$error(e, w) = \sum_j (output^L[j] - Y(e)[j])^2$$

and we will update the weight between input node m and output node l at layer k as

$$w^k_{lm} \leftarrow w^k_{lm} - \eta \frac{\partial error(e, w)}{\partial w^k_{lm}}$$

which is

$$w^k_{lm} \leftarrow w^k_{lm} - \eta \frac{\partial \left(\sum_j (Y(e)[j] - output^L[j])^2 \right)}{\partial w^k_{lm}}$$

that is,

$$w^k_{lm} \leftarrow w^k_{lm} + \eta \sum_j (Y(e)[j] - output^L[j]) \frac{\partial \left(output^L[j] \right)}{\partial w^k_{lm}} \tag{3}$$

where we have set $\eta \leftarrow 2\eta$ since η has aribitrary scale. Our objective is now to compute this derivative and to establish a recursive formula to do so that can be used with propagation (backwards) in the network. Inserting the definition for $output^L[j]$ from Equation (1), we have

$$\begin{aligned} \frac{\partial \left(output^L[j] \right)}{\partial w^k_{lm}} &= \frac{\partial}{\partial w^k_{lm}} f^L \left(\sum_i w^{L-1}_{ji} input^{L-1}[i] \right) \\ &= f^{L'}(output^{L-1}[j]) \sum_i w^{L-1}_{ji} \frac{\partial input^{L-1}[i]}{\partial w^k_{lm}} \end{aligned} \tag{4}$$

where we have used the chain rule

$$\frac{\partial f(x(w))}{\partial w} = f'(x(w)) \frac{\partial x(w)}{\partial w}.$$

Putting this back into Equation (3), and gathering the sums over inputs, i, and outputs, j, and replacing $input^{L-1}[i]$ with its equivalent $output^{L-2}[i]$, we obtain:

$$w^k_{lm} \leftarrow w^k_{lm} + \eta \sum_i \left[\sum_j (Y(e)[j] - output^L[j]) f^{L'}(output^{L-1}[j]) w^{L-1}_{ji} \right] \frac{\partial (output^{L-2}[i])}{\partial w^k_{lm}}. \tag{5}$$

We now identify the quantity in square brackets in this equation,

$$\sum_j (Y(e)[j] - output^L[j]) f^{L'}(output^{L-1}[j]) w^{L-1}_{ji},$$

as a sum over all output units j of layer L, which is the weight to be passed back recursively to output unit i in layer $L-2$. This i unit "collects" all the terms from the summation that it is "responsible for", and then will recursively pass this back to the next lower layer in exactly the same fashion. The remaining derivative in Equation (5), $\frac{\partial(output^{L-2}[i])}{\partial w_{lm}^k}$, is then computed in the same way as in Equation (4), yielding

$$\frac{\partial output^{L-2}[i]}{\partial w_{lm}^k} = f^{L-2'}(output^{L-3}[j]) \sum_i w_{ji}^{L-3} \frac{\partial input^{L-3}[i]}{\partial w_{lm}^k}.$$

Again, the partial sums corresponding to input units in the next lower layer are collected, and the process repeats. Finally, at the k^{th} layer, the derivative find its match and the derivative for the l^{th} output unit at layer $k+1$ is

$$\begin{aligned} \frac{\partial output^{k+1}[l]}{\partial w_{lm}^k} &= f^{k+1'}(output^k[l]) \frac{\partial \sum_i w_{li}^k input^k[i]}{\partial w_{lm}^k} \\ &= f^{k+1'}(output^k[l]) input^k[m] \end{aligned}$$

which is then used for the update to the weights from the m^{th} input unit to the l^{th} output unit in layer k. This process thus requires a single pass through the network to compute all $output^k$ values, and then a backwards pass to propagate all weights backwards, updating each layer as the pass proceeds, and computing the error to pass back to the next lower layer until the input layer is reached.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *DeepLearning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] David Poole and Alan Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, 2010.

[3] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.