Back-propagation and its derivation

Lucas Charpentier

November 7, 2018

1 Variables and equations

We have the following variables and equations for the neural network:

X Input vector of N inputs, each containing p different features (the x_i)
$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X)$ Hidden layer node m, $Z = (Z_1, Z_1, \ldots, Z_M)$.
$T_k = \beta_{0k} + \beta_k^T Z$ Output k before "activation", $T = (T_1, T_2, \ldots, T_K)$.
$\beta_k = (\beta_{k1}\, \beta_{k2}\, \ldots\, \beta_{kM})$ The weights applied to the hidden layer for output k.
β_{0k} The bias added to output k before going through the activation function.
$\alpha_m = (\alpha_{m1}\, \alpha_{m2}\, \ldots\, \alpha_{mp})$ The weights applied to the input layer for hidden node m.
α_{0m} The bias added to node m of the hidden layer before going through the activation function.

2 Derivation

Before starting, we will (as in [HTF09]) re-define β_k and α_m as:

$$\beta_k = (\beta_{0k}\, \beta_{k1}\, \beta_{k2}\, \ldots\, \beta_{kM})$$
$$\alpha_m = (\alpha_{0m}\, \alpha_{m1}\, \alpha_{m2}\, \ldots\, \alpha_{mp})$$

Now that we have redefined the weights, we can introduce our error function, we will use the squared error:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2. \tag{1}$$

Where y_{ik} is the true value of output k for input i and $f_k(x_i)$ is our prediction for y_{ik}. So if we define:

$$R(i) = \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2. \tag{2}$$

We get:

$$R(\theta) = \sum_{i=1}^{N} R(i). \tag{3}$$

To be able to do back-propagation for each input i, we need to derive $R(i)$. We first need to do the derivation:

$$\frac{\delta R_i}{\delta \beta_{km}} \quad \forall k, m$$

Using 2, we get the equation:

$$\frac{\delta \sum_{l=1}^{K} (y_{il} - f_l(x_i))}{\delta \beta_{km}}$$

From here there are multiple ways to get $f_k(x_i)$ that lead into two ways of doing the derivation.

2.1 g_k defined only in terms of t_{ik}

$$f_k(x_i) = g_k(t_{ik})$$

where g_k is the "activation" function for output k, $t_{ik} = \beta_k^T z_i$ and $z_i = (1, z_{i1}, z_{i2}, \ldots, z_{iM})$. This means that the output function for node k is defined by the weights sending the hidden layer to itself and not by the weights sending the hidden layer to other output nodes. Now we can start deriving R_i:

$$\begin{aligned}
\frac{\delta R_i}{\delta \beta_{km}} &= \frac{\delta \sum_{l=1}^{K} (y_{il} - f_l(x_i))^2}{\delta \beta_{km}} \\
&= \sum_{l=1}^{K} 2(y_{il} - f_l(x_i)) \frac{\delta y_{il} - f_l(x_i)}{\delta \beta_{km}} \\
&= -2 \sum_{l=1}^{K} (y_{il} - f_l(x_i)) \frac{\delta g_l(t_{il})}{\delta \beta_{km}} \\
&= -2 \sum_{l=1}^{K} (y_{il} - f_l(x_i)) \frac{\delta g_l(t_{il})}{\delta \beta_k} \frac{\delta \beta_l^T z_i}{\delta \beta_{km}} \\
&= -2(y_{ik} - f_k(x_i)) \frac{\delta g_k(t_{ik})}{\delta \beta_k} z_{im}.
\end{aligned}$$

We will rewrite:

$$\frac{\delta g_k(t_{ik})}{\delta \beta_k} = g_k'(\beta_k^T z_i).$$

So we have:

$$\frac{\delta R_i}{\delta \beta_{km}} = -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) z_{im}. \tag{4}$$

This gives us the update scheme for the weights and biases sending the hidden layer to the output nodes. Now we need to find how we update the weights and biases sending the input layer to the hidden layer. For that, we do the derivation:

$$\frac{\delta R_i}{\delta \alpha_{ml}}$$

It is not immediately obvious where the α_{ml} comes in, but if we rewrite:

$$z_{im} = \sigma(\alpha_m^T x_i), \quad \forall m \in [1, M]$$

where $\alpha_m = (\alpha_{0m}, \alpha_{m1}, \alpha_{m2}, \ldots, \alpha_{mp})$, σ is the activation function for the hidden layer and $x_i = (1, x_{i1}, x_{i2}, \ldots, x_{ip})$. Now if we do the derivation and using our previous derivation:

$$\begin{aligned}
\frac{\delta R_i}{\delta \alpha_{ml}} &= -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \frac{\delta \beta_k^T z_i}{\delta \alpha_{ml}} \\
&= -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \frac{\delta z_{im}}{\delta \alpha_{ml}} \\
&= -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \frac{\delta \sigma(\alpha_m^T x_i)}{\delta \alpha_{ml}} \\
&= -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) \frac{\delta \alpha_m^T x_i}{\delta \alpha_{ml}} \\
&= -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{il}.
\end{aligned}$$

Therefore:

$$\frac{\delta R_i}{\delta \alpha_{ml}} = -2 \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{il}. \tag{5}$$

If we re-define 4 as:

$$\frac{\delta R_i}{\delta \beta_{km}} = \delta_{km} z_{im}. \tag{6}$$

where $\delta_{km} = -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i)$. Then by combining 6 and 5, we get:

$$\frac{\delta R_i}{\delta \alpha_{ml}} = \sum_{k=1}^{K} \delta_{ki} \beta_{km} \sigma'(\alpha_m^T x_i) x_{il}. \tag{7}$$

Which concludes the derivation for this case.

2.2 g_k defined in terms of t_i (usually when doing multi-class classification)

The main difference to before is that we define the output function as:

$$f_k(x_i) = g_k(t_i)$$

where g_k is the "activation" function for output k, and $t_i = (t_{i1}, t_{i2}, \ldots, ti_{iK}$ and the rest of variables defined same as before. This means that the output function for node k is defined by all the weights sending the hidden layer to the output layer. Now we can start deriving R_i:

$$\begin{aligned}
\frac{\delta R_i}{\delta \beta_{km}} &= \frac{\delta \sum_{l=1}^{K} (y_{il} - f_l(x_i))^2}{\delta \beta_{km}} \\
&= \sum_{l=1}^{K} 2(y_{il} - f_l(x_i)) \frac{\delta y_{il} - f_l(x_i)}{\delta \beta_{km}} \\
&= -2 \sum_{l=1}^{K} (y_{il} - f_l(x_i)) \frac{\delta g_l(t_i)}{\delta \beta_{km}} \\
&= -2 \sum_{l=1}^{K} (y_{il} - f_l(x_i)) \frac{\delta g_l(t_i)}{\delta \beta_k} \frac{\delta \beta_k^T z_i}{\delta \beta_{km}} \\
&= -2 \sum_{l=1}^{K} (y_{ik} - f_k(x_i)) \frac{\delta g_l(t_i)}{\delta \beta_k} z_{im}.
\end{aligned}$$

We will rewrite:

$$\frac{\delta g_l(t_i)}{\delta \beta_k} = g_l'(\beta_k^T z_i).$$

So we have:

$$\frac{\delta R_i}{\delta \beta_{km}} = -2 \sum_{l=1}^{K} (y_{il} - f_l(x_i)) g_l'(\beta_k^T z_i) z_{im}. \tag{8}$$

This gives us the update scheme for the weights and biases sending the hidden layer to the output nodes. Now we need to find how we update the weights and biases sending the input layer to the hidden layer. For that, we do the derivation:

$$\frac{\delta R_i}{\delta \alpha_{ml}}$$

It is not immediately obvious where the α_{ml} comes in, but if we rewrite:

$$z_{im} = \sigma(\alpha_m^T x_i), \quad \forall m \in [1, M]$$

where the variables are defined as before. Now if we do the derivation and using our previous derivation:

$$\begin{aligned}
\frac{\delta R_i}{\delta \alpha_{ml}} &= -2\sum_{j=1}^{K}(y_{ij} - f_j(x_i))\frac{\delta g_j(t_i)}{\delta \alpha_{ml}} \\
&= -2\sum_{j=1}^{K}(y_{ij} - f_j(x_i))\sum_{k=1}^{K} g_j'(\beta_k^T z_i)\beta_{km}\frac{\delta z_{im}}{\delta \alpha_{ml}} \\
&= -2\sum_{j=1}^{K}(y_{ij} - f_j(x_i))\sum_{k=1}^{K} g_j'(\beta_k^T z_i)\beta_{km}\frac{\delta \sigma(\alpha_m^T x_i)}{\delta \alpha_{ml}} \\
&= -2\sum_{j=1}^{K}(y_{ij} - f_j(x_i))\sum_{k=1}^{K} g_j'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)\frac{\delta \alpha_m^T x_i}{\delta \alpha_{ml}} \\
&= -2\sum_{j=1}^{K}(y_{ij} - f_j(x_i))\sum_{k=1}^{K} g_j'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{il} \\
&= -2\sum_{k=1}^{K}\sum_{j=1}^{K}(y_{ij} - f_j(x_i))g_j'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{il}.
\end{aligned}$$

Therefore:

$$\frac{\delta R_i}{\delta \alpha_{ml}} = -2\sum_{k=1}^{K}\sum_{j=1}^{K}(y_{ij} - f_j(x_i))g_j'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{il}. \tag{9}$$

If we re-define 8 as:

$$\frac{\delta R_i}{\delta \beta_{km}} = \delta_{km} z_{im}. \tag{10}$$

where $\delta_{km} = -2\sum_{l=1}^{K}(y_{il} - f_l(x_i))g_l'(\beta_k^T z_i)$. Then by combining 10 and 9, we get:

$$\frac{\delta R_i}{\delta \alpha_{ml}} = \sum_{k=1}^{K}\delta_{ki}\beta_{km}\sigma'(\alpha_m^T x_i)x_{il}. \tag{11}$$

Which is the same equation as before. Therefore the difference in the two methods resides in the derivation of the activation function of the output layer. This concludes the two derivations for the second case.

2.3 Doing the back-propagation

Now that we have shown how to do both types of derivation, we can do the back-propagation, which is done the same way for both. Since we have found the two derivatives, we can apply

gradient descent on the weights to do the back-propagation:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\delta R_i}{\delta \beta_{km}} \tag{12}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\delta R_i}{\delta \alpha_{ml}} \tag{13}$$

where γ_r is the *learning rate* for the $r+1$-update. Once this is done for each weight, we have completed one pass of the back-propagation algorithm for the case of a single hidden layer.

3 Multi-layer Neural Network

Now that we are able to do back-propagation for a single hidden layer neural network, we can expand the algorithm to a multi-layer neural network. Let us assume we have a neural network with D hidden layer, the hidden layer d has M_d nodes $\forall d \in [1, D]$. So if re-define some of our variables and define new ones, we have the following variables:

X Input vector of N inputs, each containing p different features (the x_i)
$Z_m^{(d)} = \sigma(\alpha_{0m}^{(d)} + \alpha_m^{(d)T} X)$ Hidden layer d node m, $Z^{(d)} = (Z_1^{(d)}, Z_1^{(d)}, \ldots, Z_{M_d}^{(d)})$.
$T_k = \beta_{0k} + \beta_k^T Z$ Output k before "activation", $T = (T_1, T_2, \ldots, T_K)$.
$\beta_k = (\beta_{0k}\, \beta_{k1}\, \beta_{k2}\, \ldots\, \beta_{kM_D})$ The weights applied to the last hidden layer for output k.
β_{0k} The bias added to output k before going through the activation function.
$\alpha_m^{(d)} = (\alpha_{0m}^{(d)}\, \alpha_{m1}^{(d)}\, \alpha_{m2}^{(d)}\, \ldots\, \alpha_{mp}^{(d)})$ The weights applied to the hidden layer $d-1$ (if $d = 1$, the input layer) for hidden layer d node m.
$\alpha_{0m}^{(d)}$ The bias added to node m of the hidden layer d before going through the activation function.

Therefore, the first derivation $\frac{\delta R_i}{\delta \beta_{km}}$ is the same as before, where we have 6 or 10 depending on which activation function we have for the output layer, where all we need to change is the z_{im} to $z_{im}^{(D)}$, therefore we have:

$$\frac{\delta R_i}{\delta \beta_{km}} = \delta_{ki} z_{im}^{(D)} \tag{14}$$

If we re-write 7 as:

$$\frac{\delta R_i}{\delta \alpha_{ml}^{(D)}} = \sum_{k=1}^{K} \delta_{ki} \beta_{km} (\sigma^{(D)})'((\alpha_m^{(D)})^T z_i^{(D-1)}) z_{il}^{(D-1)} \tag{15}$$

where $(\sigma^{(D)})$ is the activation function for the last hidden layer. Then we have the derivation need for the update scheme of the last hidden layer. If we define re-write 15 as:

$$\frac{\delta R_i}{\delta \alpha_{ml}^{(D)}} = s_{mi}^{(D)} z_{il}^{(D-1)} \tag{16}$$

where $s_{mi}^{(D)} = \sum_{k=1}^{K} \delta_{ki}\beta_{km}(\sigma^{(D)})'((\alpha_m^{(D)})^T z_i^{(D-1)})$. Then if we do a similar derivation as when finding $\frac{\delta R_i}{\delta \alpha^{ml}}$, then we get:

$$\frac{\delta R_i}{\delta \alpha_{ml}^{(d)}} = \sum_{k=1}^{M_{d+1}} s_{ki}^{(d+1)} \alpha_{km}^{(d+1)} (\sigma^{(d)})'((\alpha_m^{(d)})^T z_i^{(d-1)}) z_{il}^{(d-1)} \quad \forall d \in [2, D-1] \tag{17}$$

where $s_{ki}^{(d+1)} = (\frac{\delta R_i}{\delta \alpha_{kl}^{(d+1)}})/z_{il}^{(d)}$ and $(\sigma^{(d)})$ is the activation function of hidden layer d. The $s_{ki}^{(d+1)}$ can be found by doing the derivation starting at $D-1$ and going till 2 (we start at $D-1$ because we know the value of $s_{ki}^{(D)}$). Now we have done all the derivations of the update schemes between the hidden layer. All that is left is the derivation for the update scheme between the first hidden layer and the input. This is very similar to 17 with just a difference in the last variable, therefore we get:

$$\frac{\delta R_i}{\delta \alpha_{ml}^{(1)}} = \sum_{k=1}^{M_2} s_{ki}^{(2)} \alpha_{km}^{(2)} (\sigma^{(1)})'((\alpha_m^{(1)})^T x_i) x_{il} \tag{18}$$

where $(\sigma^{(1)})'$ is the activation function of the first hidden layer. Now that we have done all the derivations, we can write the update scheme for each weight:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\delta R_i}{\delta \beta_{km}}$$

$$(\alpha_{ml}^{(d)})^{(r+1)} = (\alpha_{ml}^{(d)})^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\delta R_i}{\delta \alpha_{ml}^{(d)}} \quad \forall d \in [1, D]$$

where γ_r is the learning rate for update $r+1$. Once all the weights are updated, we have done one pass of the back-propagation.

4 Common activation functions and their derivatives.

1. Sigmoid function ($a_k(x) = \frac{1}{1+e^{-x}}$):

$$a_k' = a_k(1 - a_k).$$

2. Hyperbolic Tangent function ($a_k(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$):

$$a_k' = 1 - a_k^2.$$

3. Softmax function (usually only used as an output layer activation and in classification problems) $g_k(t_i) = \frac{e^{t_{ik}}}{\sum_{l=1}^{K} e^{t_{ikl}}} = \delta_k$):

$$g_k'(\beta_l^T z_i^{(D)}) = \begin{cases} -\delta_k \delta_l & \text{if } k \neq l \\ \delta_k(1 - \delta_k) & \text{if } k = l \end{cases}.$$

4. Rectified Linear Unit (ReLU) $\left(a_k(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}\right)$:

$$a'_k = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

5. Leaky ReLU $\left(a_k(x) = \begin{cases} \alpha x & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}\right)$, where $\alpha \leq 1$ and chosen before-hand:

$$a'_k = \begin{cases} \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

6. Exponential Linear Unit (ELU) $\left(a_k(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}\right)$, where $\alpha \leq 1$ and chosen before-hand:

$$a'_k = \begin{cases} a_k + \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

References

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009. URL: `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`.