1 Neural Nets

Consider the following computation graph for a simple neural network for binary classification. Here $x$ is a single real-valued input feature with an associated class $y^*$ (0 or 1). There are two weight parameters $w_1$ and $w_2$, and non-linearity functions $g_1$ and $g_2$ (to be defined later, below). The network will output a value $a_2$ between 0 and 1, representing the probability of being in class 1. We will be using a loss function $Loss$ (to be defined later, below), to compare the prediction $a_2$ with the true class $y^*$.

1. Perform the forward pass on this network, writing the output values for each node $z_1, a_1, z_2$ and $a_2$ in terms of the node’s input values:

   $z_1 = x * w_1$
   $a_1 = g_1(z_1)$
   $z_2 = a_1 * w_2$
   $a_2 = g_2(z_2)$

2. Compute the loss $Loss(a_2, y^*)$ in terms of the input $x$, weights $w_i$, and activation functions $g_i$:

   Recursively substituting the values computed above, we have:

   $$Loss(a_2, y^*) = Loss(g_2(w_2 * g_1(w_1 * x)), y^*)$$

3. Now we will work through parts of the backward pass, incrementally. Use the chain rule to derive $\frac{\partial Loss}{\partial w_2}$. Write your expression as a product of partial derivatives at each node: i.e. the partial derivative of the node’s output with respect to its inputs. (Hint: the series of expressions you wrote in part 1 will be helpful; you may use any of those variables.)

   $$\frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial w_2}$$
4. Suppose the loss function is quadratic, \( \text{Loss}(a_2, y^*) = \frac{1}{2}(a_2 - y^*)^2 \), and \( g_1 \) and \( g_2 \) are both sigmoid functions \( g(z) = \frac{1}{1 + e^{-z}} \) (note: it's typically better to use a different type of loss, cross-entropy, for classification problems, but we'll use this to make the math easier).

Using the chain rule from Part 3, and the fact that \( \frac{\partial g(z)}{\partial z} = g(z)(1 - g(z)) \) for the sigmoid function, write \( \frac{\partial \text{Loss}}{\partial w_2} \) in terms of the values from the forward pass, \( y^* \), \( a_1 \), and \( a_2 \):

First we'll compute the partial derivatives at each node:

\[
\frac{\partial \text{Loss}}{\partial a_2} = (a_2 - y^*) \\
\frac{\partial a_2}{\partial z_2} = \frac{\partial g_2(z_2)}{\partial z_2} = g_2(z_2)(1 - g_2(z_2)) = a_2(1 - a_2) \\
\frac{\partial z_2}{\partial w_2} = a_1
\]

Now we can plug into the chain rule from part 3:

\[
\frac{\partial \text{Loss}}{\partial w_2} = \frac{\partial \text{Loss}}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial w_2} = (a_2 - y^*) * a_2(1 - a_2) * a_1
\]

5. Now use the chain rule to derive \( \frac{\partial \text{Loss}}{\partial w_1} \) as a product of partial derivatives at each node used in the chain rule:

\[
\frac{\partial \text{Loss}}{\partial w_1} = \frac{\partial \text{Loss}}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1} = (a_2 - y^*) * a_2(1 - a_2) * a_1(1 - a_1)
\]

6. Finally, write \( \frac{\partial \text{Loss}}{\partial w_1} \) in terms of \( x, y^*, w_i, a_i, z_i \): The partial derivatives at each node (in addition to the ones we computed in Part 4) are:

\[
\frac{\partial z_2}{\partial a_1} = w_2 \\
\frac{\partial a_1}{\partial z_1} = \frac{\partial g_1(z_1)}{\partial z_1} = g_1(z_1)(1 - g_1(z_1)) = a_1(1 - a_1) \\
\frac{\partial z_1}{\partial a_1} = x
\]

Plugging into the chain rule from Part 5 gives:

\[
\frac{\partial \text{Loss}}{\partial w_1} = \frac{\partial \text{Loss}}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1} = (a_2 - y^*) * a_2(1 - a_2) * w_2 * a_1(1 - a_1) * x
\]

7. What is the gradient descent update for \( w_1 \) with step-size \( \alpha \) in terms of the values computed above?

\[
w_1 \leftarrow w_1 - \alpha (a_2 - y^*) * a_2(1 - a_2) * w_2 * a_1(1 - a_1) * x
\]