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 $\text{Loss}$ (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:

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

3. Now we will work through parts of the backward pass, incrementally. Use the chain rule to derive $\frac{\partial \text{Loss}}{\partial w_2}$. 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.)
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 \( a_2 \): 

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:

6. Finally, write \( \frac{\partial \text{Loss}}{\partial w_1} \) in terms of \( x, y^*, w_i, a_i, z_i \):

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

Let’s compute the backward step for a node that computes $x^T Ax$, where $x$ is a vector with $m$ values, and $A$ is a matrix with shape $m \times m$. Thus, $c = \sum_{i=1}^{m} x_i \sum_{j=1}^{m} A_{ij} x_j = \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij} x_i x_j = \sum_{j=1}^{m} x_j \sum_{i=1}^{m} A_{ij} x_i$.

1. What is $\frac{\partial f}{\partial A_{ij}}$?

2. What is $\frac{\partial f}{\partial A}$?

3. What is $\frac{\partial f}{\partial x}$?

4. What is $\frac{\partial f}{\partial x}$?