CS188 Fall 2018 Section 12: Neural Networks

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:

\[
\begin{align*}
  z_1 &= x \ast w_1 \\
  a_1 &= g_1(z_1) \\
  z_2 &= a_1 \ast w_2 \\
  a_2 &= g_2(z_2)
\end{align*}
\]

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

   Recursively substituting the values computed above, we have:

   \[
   \text{Loss}(a_2, y^*) = \text{Loss}(g_2(w_2 \ast g_1(w_1 \ast x)), y^*)
   \]

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.)

\[
\frac{\partial \text{Loss}}{\partial w_2} = \frac{\partial \text{Loss}}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial w_2}
\]
4. Suppose the loss function is quadratic, $Loss(a_2, y^*) = \frac{1}{2}(a_2 - y^*)^2$, 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 $\frac{\partial Loss}{\partial w_2}$ 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:

$$\frac{\partial Loss}{\partial a_2} = (a_2 - y^*)$$

$$\frac{\partial a_2}{\partial z_2} = \frac{\partial g_2(z_2)}{\partial z_2} = g_2(z_2)(1 - g_2(z_2)) = a_2(1 - a_2)$$

$$\frac{\partial z_2}{\partial w_2} = a_1$$

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

$$\frac{\partial Loss}{\partial w_2} = \frac{\partial Loss}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial w_2}$$

$$= (a_2 - y^*) * a_2(1 - a_2) * a_1$$

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

$$\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial a_1} \frac{\partial a_1}{\partial z_1}$$

6. Finally, write $\frac{\partial Loss}{\partial w_1}$ in terms of $x, y^*, w_1, a_1, z_1$: 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:

$$\frac{\partial Loss}{\partial w_1} = \frac{\partial Loss}{\partial a_2} \frac{\partial a_2}{\partial z_2} \frac{\partial z_2}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}$$

$$= (a_2 - y^*) * a_2(1 - a_2) * w_2 * a_1(1 - a_1) * x$$

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

$$w_1 \leftarrow w_1 - \alpha(a_2 - y^*) * a_2(1 - a_2) * w_2 * a_1(1 - a_1) * x$$
2 Vectorized Gradients

Let’s compute the backward step for a node that computes $x^TAx$, 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 $\frac{\partial f}{\partial A_{ij}}$?
   
   $\frac{\partial f}{\partial A_{ij}} = \frac{\partial f}{\partial c} \frac{\partial c}{\partial A_{ij}} = \frac{\partial f}{\partial c} x_i x_j$

2. What is $\frac{\partial f}{\partial A}$?
   
   $\frac{\partial f}{\partial A} = \frac{\partial f}{\partial x} x^T$

3. What is $\frac{\partial f}{\partial x}$?
   Use the Product Rule:
   
   $$\frac{\partial f}{\partial x_k} = \frac{\partial f}{\partial c} \frac{\partial c}{\partial x_k} = \frac{\partial f}{\partial c} \left( \frac{d}{dx_k} \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij} x_i x_j \right) = \frac{\partial f}{\partial c} \left( \frac{d}{dx_k} \sum_{j=1}^{m} A_{kj} x_j + \frac{d}{dx_k} \sum_{i=1}^{m} A_{ik} x_i \right)$$

   $$\frac{\partial f}{\partial x_k} = \frac{\partial f}{\partial c} \left( \sum_{j=1}^{m} A_{kj} x_j + \sum_{i=1}^{m} A^T_{ki} x_i \right) = \frac{\partial f}{\partial c} \left( \sum_{j=1}^{m} (A_{kj} + A^T_{kj}) x_j \right)$$

4. What is $\frac{\partial f}{\partial x}$?
   
   $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial c} (A + A^T)x$