

Data Science 210 Final Project

Thomas Kwashnak

Fall 2021

1 Background

2 Design

During this section, a variety of variables and variable syntax will be used. The following is a list that explains each variable. Remember that for a given "layer", the output is considered the 0th layer, the hidden layer is considered the 1st layer, and the input is considered the 2nd layer.

\bar{y}^n The output values of the nodes in the n^{th} hidden layer.

y_i^n The output value of node i in the n^{th} hidden layer.

y By default, if y is left alone, it represents the output layer, known as y_0^0

\hat{y} The expected final output

W^n The weight matrix for a given layer n , as it applies to the values \bar{y}^{n+1} (the outputs of the previous layer)

$w_{i,j}^n$ The weight value for the value passed from node j of layer $n + 1$ as it is passed to node i of layer n

2.1 Derivations and Equations

2.1.1 The σ function

The sigmoid activation function that will be used for all nodes (apart from the input layer, which uses a linear activation function) can be explained as follows:

$$\sigma(n) = \frac{1}{1 + e^{-n}}$$

$$\sigma'(n) = \sigma(n) * (1 - \sigma(n))$$

When using this function on a matrix or vector, we simply just apply the sigma function for all individual values in the matrix/vector

2.1.2 Forward Feeding Equation

This equation represents what the output will be for a given input \vec{y}^2 . The basic principal is derived from the following relationship for the outputs of layer n :

$$\vec{y}^n = \sigma(W_n * \vec{y}^{n+1})$$

We can chain this rule to find the output vector. Since the output vector is a 1x1 matrix, it will result in a scalar value

$$y = \sigma(W_0 * \sigma(W_1 * \vec{y}^2))$$

We can then simplify the first layer by writing the manual multiplication:

$$y = \sigma\left(\sum_i (w_{0,i}^0 * \sigma(W^1 * \vec{y}^2)_{i,0})\right)$$

2.1.3 The Loss Function

The loss function is the overall function we wish to minimize, and is as follows:

$$L = (y_0^0 - \hat{y})^2$$

2.1.4 Layer 1 Derivative

In back propagation, we need to find the derivatives for each of the weights in the first layer. The derivative can be simplified using the chain rule as follows:

$$\frac{\delta L}{\delta w_{0,i}^0} = \frac{\delta L}{\delta y} * \frac{\delta y}{\delta w_{0,i}^0}$$

We can factor this out to be the following. Note that $\sigma(\dots) = \sigma(\sum_i (w_{0,i}^0 * \sigma(W^1 * \vec{y}^2)_{i,0})) = y$.

$$\begin{aligned} \frac{\delta L}{\delta w_{0,i}^0} &= 2(y - \hat{y}) * (\sigma'(\dots)) * (\sigma((W^1 * \vec{y}^2)_{i,0})) \\ \frac{\delta L}{\delta w_{0,i}^0} &= 2(y - \hat{y}) * (\sigma(\dots)(1 - \sigma(\dots))) * (\sigma(\sum_h (w_{i,h}^1 * y_h^2))) \\ \frac{\delta L}{\delta w_{0,i}^0} &= 2(y - \hat{y}) * (y * (1 - y)) * (\sigma(\sum_h (w_{i,h}^1 * y_h^2))) \\ \frac{\delta L}{\delta w_{0,i}^0} &= 2(y - \hat{y}) * (y - y^2) * (\sigma(\sum_h (w_{i,h}^1 * y_h^2))) \end{aligned}$$

And thus, we have the derivative of a given weight in relation to the network.

2.1.5 Layer 2 Derivative

The last Layer 2 derivative is based on the derivative of the 1st layer's derivative.

$$\frac{\delta L}{\delta w_{i,j}^1} = \frac{\delta L}{\delta w_{0,i}^0} * \frac{\delta w_{0,i}^0}{\delta y_i^1} * \frac{\delta y_i^1}{\delta w_{i,j}^1}$$

Thus, with using $\frac{\delta L}{\delta w_{0,i}^0}$, we can derive $\frac{\delta L}{\delta w_{i,j}^1}$. We will keep $\frac{\delta L}{\delta w_{0,i}^0}$ in as itself, since in our algorithms we will be able to store this value from the previous step. In this instance, $\sigma(\dots) = \sigma(W^1 * \vec{y}^2) = y_i^1$.

$$\begin{aligned}\frac{\delta L}{\delta w_{i,j}^1} &= \frac{\delta L}{\delta w_{0,i}^0} * (2(y - \hat{y})(y - y^2)(\sigma'(\dots))) * (w_{i,j}^1 * y_j^2) \\ \frac{\delta L}{\delta w_{i,j}^1} &= \frac{\delta L}{\delta w_{0,i}^0} * (2(y - \hat{y})(y - y^2)(\sigma(\dots)(1 - \sigma(\dots)))) * (w_{i,j}^1 * y_j^2) \\ \frac{\delta L}{\delta w_{i,j}^1} &= \frac{\delta L}{\delta w_{0,i}^0} * (2(y - \hat{y})(y - y^2)(y_i^1(1 - y_i^1))) * (w_{i,j}^1 * y_j^2) \\ \frac{\delta L}{\delta w_{i,j}^1} &= \frac{\delta L}{\delta w_{0,i}^0} * (2(y - \hat{y})(y - y^2)((y_i^1) - (y_i^1)^2) * (w_{i,j}^1 * y_j^2)\end{aligned}$$

2.2 Data Importing

Since the data was given in a string csv format, an additional pytohn script was created to import the data into two matrices, one contianing the inputs and one contianing the expected outputs.

3 Implementation

4 Validation

5 Reflection

6 Notes

Forward Feeding Algorithm

$$S(W_0 * f(W_i * \vec{\text{In}})) = \text{Out}$$

7 Equations

$$s(x) = \frac{1}{1 + e^{-x}}$$