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.

- $\vec{y}^n$ 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 $\vec{y}^{n+1}$ (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 ouputs 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:

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

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:

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

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

$$\frac{\delta L}{\delta w^0_{0,i}} = 2(y - \hat{y}) * (\sigma'(\ldots)) * (\sigma((W^1 * \vec{y^2})_{i,0}))$$

$$\frac{\delta L}{\delta w^0_{0,i}} = 2(y - \hat{y}) * (\sigma(\ldots)(1 - \sigma(\ldots))) * (\sigma(\sum_h (w^1_{i,h} * y^2_h))$$

$$\frac{\delta L}{\delta w^0_{0,i}} = 2(y - \hat{y}) * (y * (1 - y)) * (\sigma(\sum_h (w^1_{i,h} y^2_h))$$

$$\frac{\delta L}{\delta w^0_{0,i}} = 2(y - \hat{y}) * (y - y^2) * (\sigma(\sum_h (w^1_{i,h} * y^2_h))$$

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.

$$\frac{\delta L}{\delta w^1_{i,j}} = \frac{\delta L}{\delta w^0_{0,i}} * \frac{\delta w^0_{0,i}}{\delta y^1_i} * \frac{\delta y^1_i}{\delta w^1_{i,j}}$$

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

$$\frac{\delta L}{\delta w^1_{i,j}} = \frac{\delta L}{\delta w^0_{0,i}} * (2(y - \hat{y})(y - y^2)(\sigma'(\ldots))) * (w^1_{i,j} * y^2_j)$$

$$\frac{\delta L}{\delta w^1_{i,j}} = \frac{\delta L}{\delta w^0_{0,i}} * (2(y - \hat{y})(y - y^2)(\sigma(\ldots)(1 - \sigma(\ldots)))) * (w^1_{i,j} * y^2_j)$$

$$\frac{\delta L}{\delta w^1_{i,j}} = \frac{\delta L}{\delta w^0_{0,i}} * (2(y - \hat{y})(y - y^2)(y^1_i(1 - y^1_i))) * (w^1_{i,j} * y^2_j)$$

$$\frac{\delta L}{\delta w^1_{i,j}} = \frac{\delta L}{\delta w^0_{0,i}} * (2(y - \hat{y})(y - y^2)((y^1_i) - (y^1_i)^2) * (w^1_{i,j} * y^2_j)$$

3 Implementation

3.1 Data Importing

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

4 Validation

5 Reflection

6 Notes

Forward Feeding Algorithm

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

7 Equations

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