Assignment 1 - Sequential Neural Network

Ali Fenwick

November 2021

1 Back Propagation explained and mathematics

1.1 Explanation of the equations

The back propagation algorithm is an integral method of any sequential neural network. The back propagation algorithm helps to find the correct weights of a model to produce an expected output signal. In this section, I will provide the mathematical equations for a feedforward neural network (classification problem) with one input layer, one hidden layer, and one output layer.

A binary classification has the following formula in which the aim is to calculate the output of a model as 0 or 1 based on its inputs and parameters, interpreted as the probability of predicting class 1.

$$P(y=1|X,w,b) \equiv Y = \sigma(w^T \cdot X + b) \in \big[0,1\big]^{X \mathbf{x} M},$$

$$where \sigma(z) = 1/(1+e^{-z}).$$

Note: P is the probability, X the input variable, w the weights, b the bias, and Y the predicted outcome. In the assignment description, the tanh(z) activation was specifically requested to be used. This means that Y = tanh(w$^T \cdot X + b)$

Now let's look at the loss function. The following is the mathematical equation of the loss function of a binary classification (logistical regression) model.

$$\mathcal{L} = -\frac{1}{M}\sum_m \Big[y_{real}^{(m)} log Y^{(m)} + (1-y_{real}^{(m)}) log(1-Y^{(m)})\Big] = \frac{1}{M}\sum_m \mathcal{L}^{(m)}$$

As the loss function represents the sum of all training m-th examples, breaking down the equation into single training examples will help us to calculate the derivatives and thus find the correct parameters (w,b) for the back propagation. The following is the equation of the loss function for a single m-th training example.

$$\mathcal{L}^{(m)} = -\left(y_{real}^{(m)} log Y^{(m)} + (1 - y_{real}^{(m)}) log(1 - Y^{(m)})\right) = \frac{1}{M}\sum_m \mathcal{L}^{(m)}$$

To calculate the derivative, we will use the following equation:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w_i}, \frac{\partial \mathcal{L}^{(m)}}{\partial b}$$

To calculate the general expression of the outer layer, I will first calculate the $\mathcal{L}^{(m)}$ using the chain rule as mentioned in the video for both weights $\frac{\partial \mathcal{L}^{(m)}}{\partial w_i}$ and bias $\frac{\partial \mathcal{L}^{(m)}}{\partial b}$. These are expressed in the following ways:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w_i} = \frac{\partial \mathcal{L}^{(m)}}{\partial Y}\frac{\partial Y^{(m)}}{\partial w_i} = \frac{\partial \mathcal{L}^{(m)}}{\partial Y^{(m)}}\frac{\partial Y^{(m)}}{\partial z}\frac{\partial z}{\partial w_i}$$

$$\frac{\partial \mathcal{L}^{(m)}}{\partial b} = \frac{\partial \mathcal{L}^{(m)}}{\partial Y}\frac{\partial Y^{(m)}}{\partial b} = \frac{\partial \mathcal{L}^{(m)}}{\partial Y^{(m)}}\frac{\partial Y^{(m)}}{\partial z}\frac{\partial z}{\partial b}$$

Calculating the derivatives has led to the following expressions for w and b for a single training m-th. $a_h^{(m)}$ represents the activation function:

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w_i} = (1 + Y^{(m)})\left(1 - \frac{y_{real}^{(m)}}{Y^{(m)}}\right) a_h^{(m)}$$

which gives the following equation for the general expression $\mathcal{L}$:

$$\frac{\partial \mathcal{L}}{\partial w_i} = \sum_m \left((1 + Y^{(m)})\left(1 - \frac{y_{real}^{(m)}}{Y^{(m)}}\right) a_h^{(m)}\right)$$

Next, let's calculate the derivative for $\frac{\partial \mathcal{L}^{(m)}}{\partial b}$. In this part of the derivative we need to consider the tanh activation in relation to the bias:

$$\frac{\partial z}{\partial b} = \frac{\partial}{\partial b}(w^T \cdot X + b) = 1$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_m \left((1 + Y^{(m)}) \left(1 - \frac{y_{real}^{(m)}}{Y^{(m)}} \right) \right)$$

The chain rule for the hidden layer is expressed in the following way (u represents the pre-activated value of the hidden layer):

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w_i} = \frac{\partial \mathcal{L}^{(m)}}{\partial Y} \frac{\partial Y^{(m)}}{\partial w_i} = \frac{\partial \mathcal{L}^{(m)}}{\partial Y^{(m)}} \frac{\partial Y^{(m)}}{\partial z} \frac{\partial z}{\partial a_{hi}} \frac{\partial a_{hi}}{\partial u_{hi}} \frac{\partial u_{hi}}{\partial w_i}$$

$$\frac{\partial \mathcal{L}^{(m)}}{\partial w_i} = \frac{\partial \mathcal{L}^{(m)}}{\partial Y} \frac{\partial Y^{(m)}}{\partial w_i} = \frac{\partial \mathcal{L}^{(m)}}{\partial Y^{(m)}} \frac{\partial Y^{(m)}}{\partial z} \frac{\partial z}{\partial a_{hi}} \frac{\partial a_{hi}}{\partial u_{hi}} \frac{\partial u_{hi}}{\partial b_i}$$

With the tanh equation we can calculate $\frac{\partial \mathcal{L}^{(m)}}{\partial b}$ for the hidden layer:

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_m \left((1 + Y^{(m)}) \left(1 - \frac{y_{real}^{(m)}}{Y^{(m)}} \right) \right) w_i^{(out)} (1 - a_h^2) x_j^{(m)}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_m \left((1 + Y^{(m)}) \left(1 - \frac{y_{real}^{(m)}}{Y^{(m)}} \right) \right) w_i^{(out)} (1 - a_h^2)$$

2 Coding the neural network

See the attached python notebook for the code and explanation of the code.

3 Comparing the neural net with a SKLearn Linear Classifier

I also compared the performance of the neural net model to a linear classifier model (scikit learn - LogisticalRegression()) using the same dataset and split. The results of the comparison test show that both models provide similar results (approximately 94.4 - 94.8 percent accuracy for the training set and 95.0 percent for the validation set).