# Mathematical Review

Adam Woods - W17014929

KF5012 Software Engineering Practice - AI Stream
Word Count: 1061

February 2020

# 1 Introduction

The project makes use of a lot of functions that come with python libraries such as keras and numpy. These libraries allow neural networks to be developed efficiently and with ease as most of the functionality comes from these libraries. Functions such as "spmatrix" allows for very specific pre-processing of data and keras does a huge amount of work resulting in a neural network being functional with only a few lines of code. These libraries give huge advantages when developing machine learning tools but all rely on the mathematical theory behind them. This section aims to expand on some of this theory such as the loss functions used in the model and how the model itself is updated for each epoch as well as give a problem definition and solution in terms of mathematical formulation.

# 2 Problem formulation

The task of filtering spam is simply a classification task aimed around the content of a message, there are many ways to conduct classification such as decision trees and K-Nearest Neighbour but we have used the Neural network approach for our task as well as experimenting with Multinomial Naive Bayes early in development. The problem is therefore one of optimization of the loss function, due to our problem being binary classification we use binary cross entropy as our loss function when compiling the model. This function represents the difference between the estimated and true values of the data, or the error of our model and thus minimizing this will result in a more accurate network.
This function can be written as

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i \cdot \log(1-p(y_i))$$

where y is the label of the data, p(y) is the predicted probability of the data being 1 (or spam in our case) and N is all of the data points. The way we minimize this function is by decreasing the difference between the predicted values and the actual values of the data. This will be done using the model over many iterations of training. The best way we can help is by ensuring that the data we give the model is as accurate to real scenarios as possible. (Godoy, 2020)

# 3 Parameter Justification

As we use keras to implement our model there are many choices to make as to the parameters we enter such as the activation functions and optimizer. We use sigmoid and ReLU as our activation functions, the purpose of these is to determine if a neuron should fire or not and to normalise the data for the next part of the network. The difference between the two is that sigmoid is $\frac{1}{1+e^{-x}}$, this will take the inputs and fit them between 0 and 1, this is very useful for binary classification such as ours but doesn't work with all layers, this is when we use ReLU which is defined as max(0,x) this means that if the input is less than zero it is reset to zero and if it is greater than zero then it is left alone.
The sigmoid function has been used on the final layer due to its effectiveness with binary classification, the reason it works so well for this task can be seen when you look at how it represents the chance of a value between two normal curves. As you can see in figure 1 the sigmoid function is the result of dividing the green line by the sum

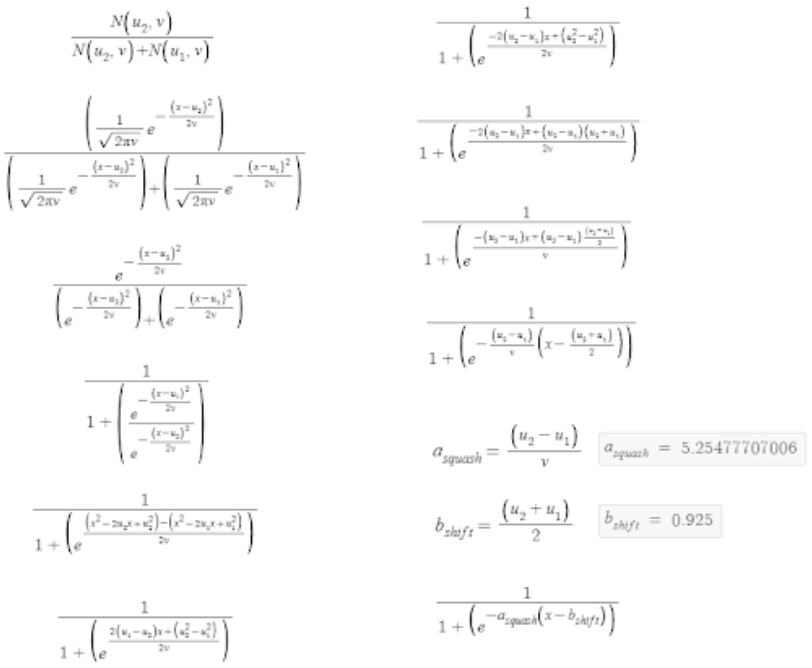

Figure 1: Derivation of two normal distributions to get the sigmoid function. (Waite, E. 2020)

of both normal curves or the chance that a given value will be in either category, thus the sigmoid function reaches 0.5 or 50% in the center of the two curves. This useful relationship is due to the fact that the sigmoid function can be derived from two normal distributions as seen in figure 2. However, there are many functions that give us very similar "S" shaped curves such as the error function but there are two major reasons why we use sigmoid. The first is because it is easy to calculate, due to the simplicity of the function is it fast to calculate and this is very helpful as we use it at the end of every epoch. The other reason is due to the linear gradient we get when we look at the loss function. Other functions that create an "S" shape don't produce good loss as seen in figure 3 and thus when the model learns it will not learn as quickly. (Waite, 2020)
The optimizer used for the model was "adam" This combines two other methods: root mean square propagation and adaptive gradient algorithm to create an efficient and effective optimizer that doesn't take up much memory and is straightforward to implement. (Brownlee, 2020)

# 4 Network Mathematics

Feed-forward and back-propagation are the main parts of the neural network. Depending on how many epochs are run determines the number of times the model must do these. Passing the data forward is done by using the input, hidden layer, the weights and the bias in the following formula:

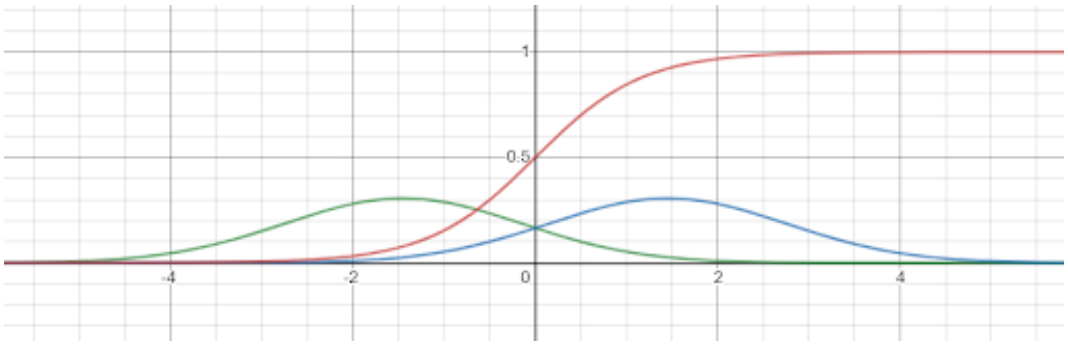


Figure 2: Two normal distributions with sigmoid function. (Waite, E. 2020)

$h = W^T x + b$ where h = the hidden layer, W is the weights, x is the input and b is the bias. This is done for every layer in the network until the data reaches the end. Then to update the weights in the network in order to improve accuracy we must go back through it trying to guess what data we originally started with. To calculate the value for the updated weights is rather complex but can be broken down into three parts. "How does a change to the next layer, effects the loss", "how a change in the current layer effects the next layer" and "How does a change to the weights affect the current layer" the first step is calculated using $\frac{\partial L}{\partial O_{L+1}}$ this is the derivation of the loss with regards to the weight, the second using $\frac{\partial O_{L+1}}{\partial O_L}$ which is the derivation of the loss in regards to the next layer and the third using $\frac{\partial O_L}{\partial W_L}$ which is the derivation of the next layer in regards to the current layer. This gives us $\frac{\partial L}{\partial W_L}$ which can then be used to update the weights in preparation for the next feed-forward. Below in figure 4 is a colour coded diagram showing what these calculations are related to in the network, so for W100 the weight is calculated using the last weight which is propagated back, the last layer and the current layer. (Chein, 2020)

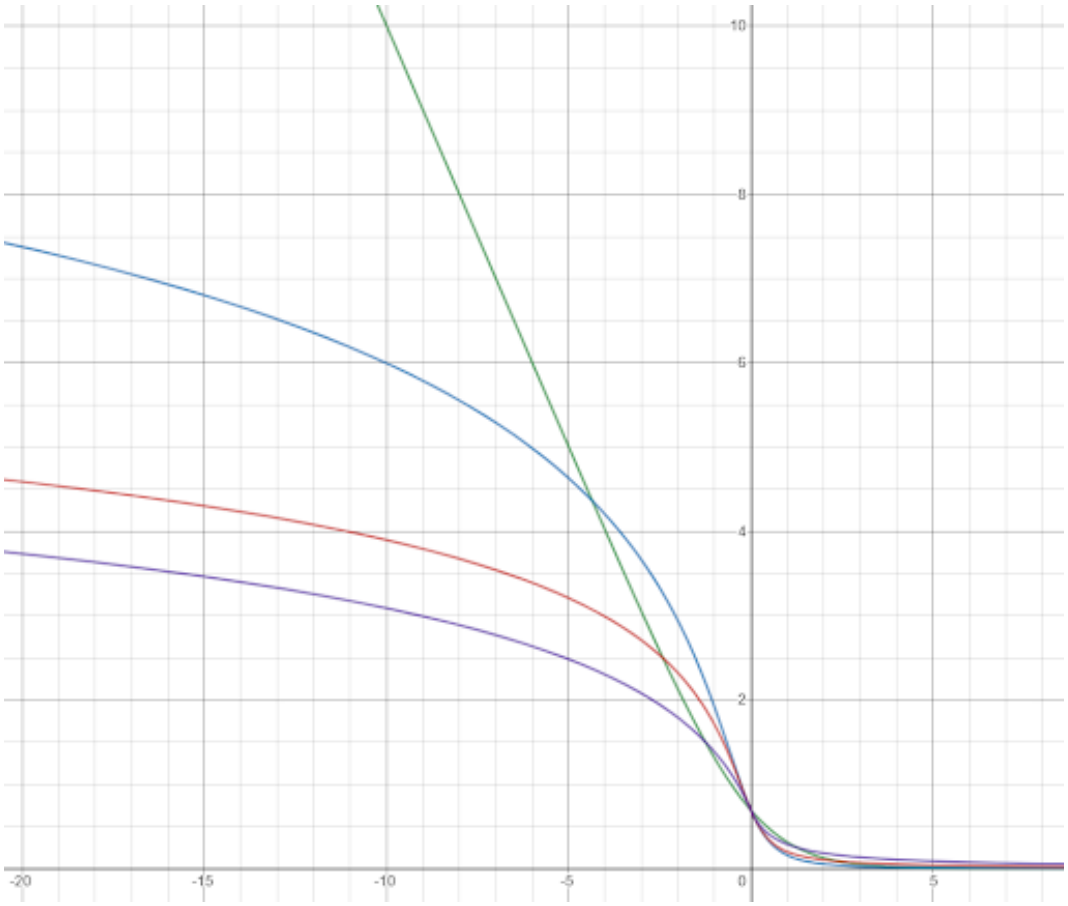


Figure 3: the loss of sigmoid against other S shaped functions (Waite, E. 2020)

# 5 References

Brownlee, J., 2020. Gentle Introduction To The Adam Optimization Algorithm For Deep Learning. [online] Machine Learning Mastery. Available at: ¡https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/¿ [Accessed 6 May 2020].

Chein, M., 2020. Backpropagation For People Who Are Afraid Of Math. [online] Medium. Available at: ¡https://towardsdatascience.com/backpropagation-for-people-who-are-afraid-of-math-936a2cbebed7¿ [Accessed 6 May 2020].

Godoy, D., 2020. Understanding Binary Cross-Entropy / Log Loss: A Visual Explanation. [online] Medium. Available at: ¡https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a¿ [Accessed 6 May 2020].

Waite, E., 2020. Sigmoid Function Vs Alternatives. [online] Desmos. Available at: ¡https://www.desmos.com/calculator [Accessed 6 May 2020].

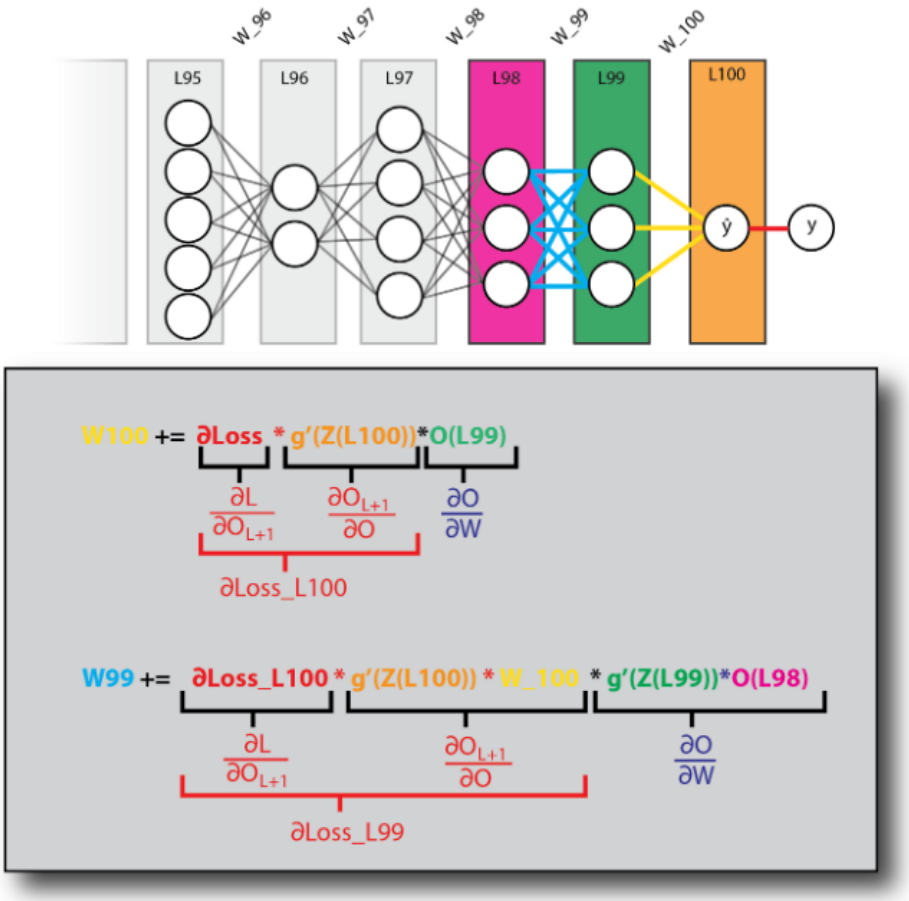


Figure 4: Colour coded example of back-propagation mathematics. (Chein, 2020)