# COS 314 : Project 3 (Neural Networks)

Werner Mostert
13019695

University of Pretoria
Computer Science Department

# A.) Introduction

This report entails experimentally finding the optimal set of parameters for a neural network who's goal is to classify a piece of text as either Afrikaans or English. The neural network more specifically is a feed forward neural network which is trained by the backpropagation algorithm using gradient descent.

# B.) The Training Data

## I. Choosing the data

When choosing data for a neural network to train on, arguably the most important decision is being made in terms of the applicability of the neural network to the actual real world problem. Since this neural network uses the "learning by example" principle, the phenomenon called "Garbage In, Garbage Out" strongly applies. Therefore, great care and consideration was taken w.r.t the choices of training data. The following considerations were made:

### *Origin of the data:*

Training data was obtained from web pages written in Afrikaans and English respectively. The majority of the data was acquired from random articles on Wikipedia, the free encyclopaedia, which has both Afrikaans and English versions of the site. Wikipedia was mostly used due to the following reasons:

- Many different authors contribute to a Wikipedia article, thus allowing for different styles of writing in the training examples.
- Wikipedia articles are generally syntactically correct and should be almost completely spelling-error free since these articles are peer reviewed on a very regular basis.
- Wikipedia is a very accessibly domain for finding large chunks of Afrikaans text, which could otherwise prove quite problematic since Afrikaans as a language does not exactly have the most prominent online presence.

### *Size of the data sets:*

One main restriction applied to the validity of a data set as a training example for the neural network was based on the size of the data sets. Data sets have to contain at the very least roughly 800 words. The largest data set chosen contains around 3000 words. Extremely large texts are avoided due to the computation time constraint on the project (it is being run on normal desktop computers).

It is assumed that this minimum size of the chunk of text will be sufficiently indicative of the language, by using the frequency at which characters occur in the text, since all the characters will appear ,if not multiple times, at least once in the data

set.

Finally, after taking the above into consideration 15 Afrikaans and 15 English texts were chosen, thus 30 in total. Taking an average word count per file as 1500 word, the rough total amount of words analysed would then amount to 45 000. The amount of training examples for a neural network is normally much larger than the amount used in this case. However, the problem in this case is much simpler than for example, pattern recognition from an image.

The data sets are stored in .txt files in the Training Data folder, where they are separated as Afrikaans or English, by being placed in the respective folders. The format of these files are a type of raw format. They appear directly as they were extracted from the original web pages and thus contain various non-alphabet characters.

## II. Pre-Processing the Data

At this current moment in time, only the frequency at which alphabet characters (irrespective of case) appear in the text will be considered as indicative attributes of the language in which a piece of text is written.

It is extremely common for text in Afrikaans to contain diacritical marks, such as ê ; ï ; etc., on characters (represented by extended ASCII or UNICODE characters) of a word. This is in reality a very important language feature very unique to Afrikaans w.r.t English since it is much less common for English words to contain diacritical marks. In fact, as a matter of interest, if an English word contains a diacritical mark it shows that the word has a French root, such as the word “cliché”.

It was however decided to discard these diacritical marks to maintain the constant amount of input nodes to the neural network as 26 (one for each alphabet character). If these were included, provision would ideally need to be made for every possible diacritical character since if only those diacritical characters present in the data sets were used the neural network's accuracy could possibly be compromised when unseen data is applied which contains diacritical characters which were not present in the original data set. In effect, the main reason why diacritical characters are discarded is to maintain simplicity.

As mentioned above, diacritical characters were discarded. This does however not mean that the character as a whole was completely ignored. Only the diacritical mark was removed, the base character is still preserved.

If one would just simply ignore the whole character, one could possibly ignore a fairly large amount of characters, in particular for a language such as Afrikaans where the diacritical characters are extremely common. If the distribution of these diacritical characters over the alphabet was a uniform distribution then one could consider

ignoring them. In Afrikaans however, diacritical marks are only applied to vowels, which is roughly one fifth of the whole set of input nodes. Thus, if the whole character was ignored, a particular set of characters would then be less prominent which could result in less accurate classification, particularly when taking into account that only character frequencies are considered and no syntactical attributes of the language.

All white space and non-alphabet characters were ignored. The casing of all characters was converted to lowers case, yet again done due to the fact that no syntactical attributes of the language was taken into consideration.

After pre-processing, the entire text should be presented as a single string with only lower case characters present.

## Calculating the frequencies

The method to calculate the frequency of the characters in the string is quite simple, as illustrated in the pseudo-code in Figure 1.1

```
Let A be the set of characters in the alphabet [a – z]
Let S be the set of characters in the data set [pre-processed string]
    ∀ c ∈ A
        occurrence(c) = 0
        ∀ c' ∈ S
            if c and c' are the same characters then
                occurrence(c)++
        frequency(c) = occurrence(c)/sizeOf(S)
```

*Figure 1.1 Pseudo code to calculate character frequencies*

The activation function used for a node in the neural network is the Sigmoid function. The sigmoid function:

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

Since the frequencies of characters will always be in the range [0.0-1.0] we need to scale the frequencies to the active domain of the Sigmoid function. An active domain is the domain of the function where meaningful changes in gradient take place. The active domain of the Sigmoid function is $[-\sqrt{3};\sqrt{3}]$. This can be quite clearly observed from Figure 1.2.

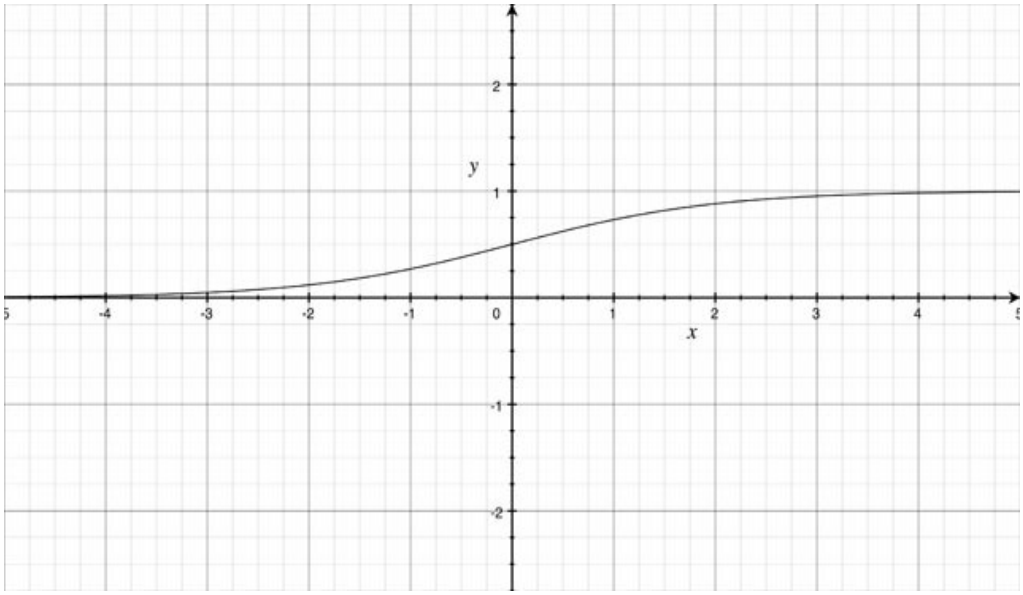

*Figure 1.2 The Sigmoid Function*

The "final frequencies", or otherwise stated, the actual input values to the neural network will then be determined by scaling with the following formula:

$$t_s = \frac{t_u - t_{u,\min}}{t_{u,\max} - t_{u,\min}}(t_{s,\max} - t_{s,\min}) + t_{s,\min}$$

Where,

- $t_s$ is the scaled frequency, used as input values
- $t_u$ is the unscaled frequency
- $t_{u,max}$ and $t_{u,min}$ are the lowest and highest frequencies calculated per character in the training example's data set respectively
- $t_{s,\max}$ and $t_{s,min}$ are the lower and upper bound of the active domain of the activation function. In this case $\sqrt{3}$ and $-\sqrt{3}$ respectively

# B.) Experimentation

## Some Terminology

When referring to Parameter Set N where N is an integer, one will abbreviate $P_N$, the configuration of the neural network is meant. $P_0$ is the initial configuration.

## Initial Solution

In order to investigate the effect of the parameters to the neural network, a starting solution is required. This solution's parameters were obtained by making the following assumptions for the general case:

- The momentum value should be high
- The learning rate value should be low

By making these assumptions, a range of values were tested between 0.5 and 1.0 for the momentum value and a range of values between 0.0 and 0.5 for the learning rate

with increments of 0.01. The number of hidden nodes were also tested ranging from 1 to 52.

A particular parameter set was deemed superior to another in the case that the sum of the training accuracy $A_T$ and the generalization accuracy $A_G$ for that set was higher than that of the highest sum thus far. Once a sum of 200.0 was reached, the search stopped and the solution was chosen as the initial solution for the experimentation phase. Due to the stochastic nature of the initialised weights in the neural network this parameter set is however not a very good indication of a "good" configuration since one would normally run the configuration multiple times in order to obtain some sense of superiority. For the purpose of obtaining "some" relatively decent starting configuration, this method is sufficient.

The initial parameter set $P_0$ is thus:

- Momentum: 0.98
- Learning Rate: 0.07
- Hidden Nodes: 1

Running this configuration 50 times, and using the averages of the training accuracy $A_T$ and the generalization accuracy $A_G$ for each epoch for 100 epochs yields the results as shown in Figure 2.1

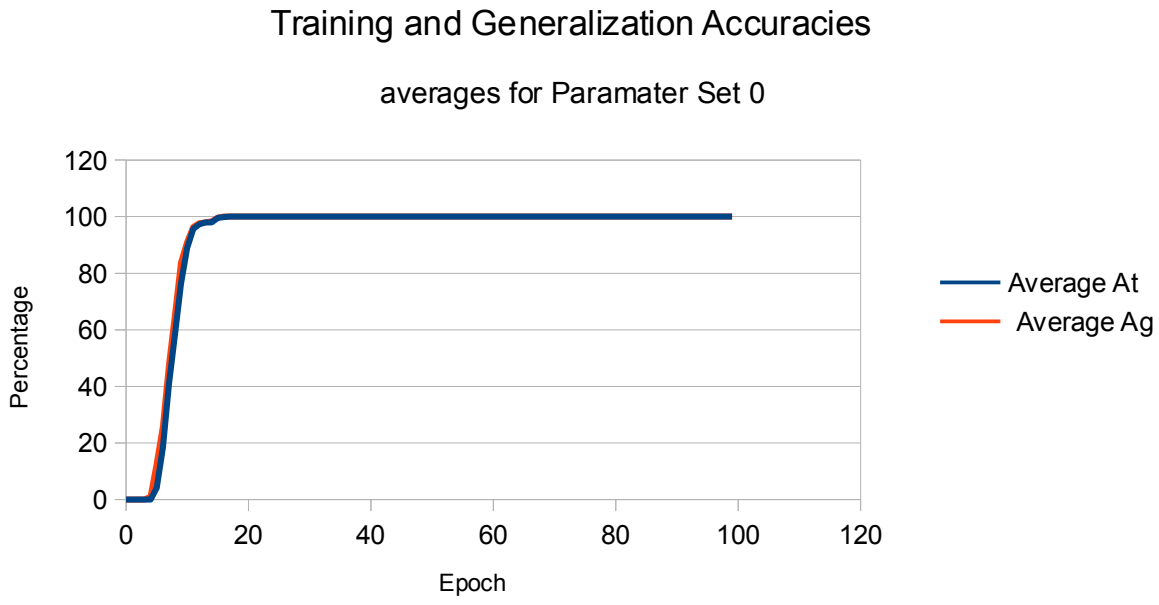


*Figure 2.1 Paramater Set 0 results*

From Figure 2.1 it can clearly be observed that the network converges quite early (epoch number 17). Due to the simplicity of the problem, this behaviour is as expected. For further experimental testing, the maximum number of epochs will be reduced to 25.

**The effect of the Momentum value on the neural network**
**The effect of the Learning Rate value on the neural network**
**The effect of the number of hidden nodes on the neural network**