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 [-√3;√3]. This can be quite clearly observed from Figure 1.2.

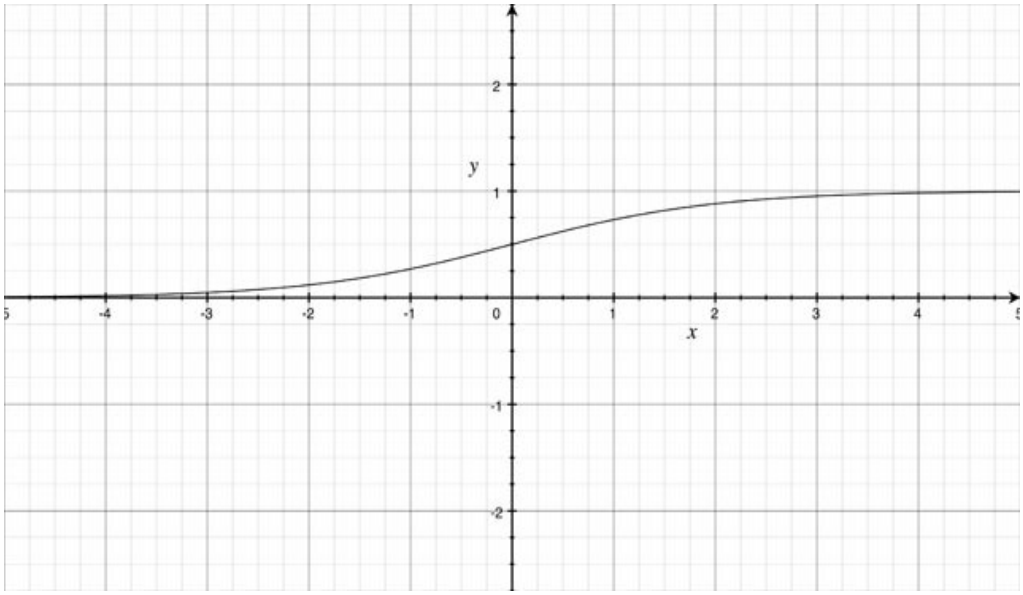

Figure 1.2 The Sigmoid Function

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 √3 and -√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

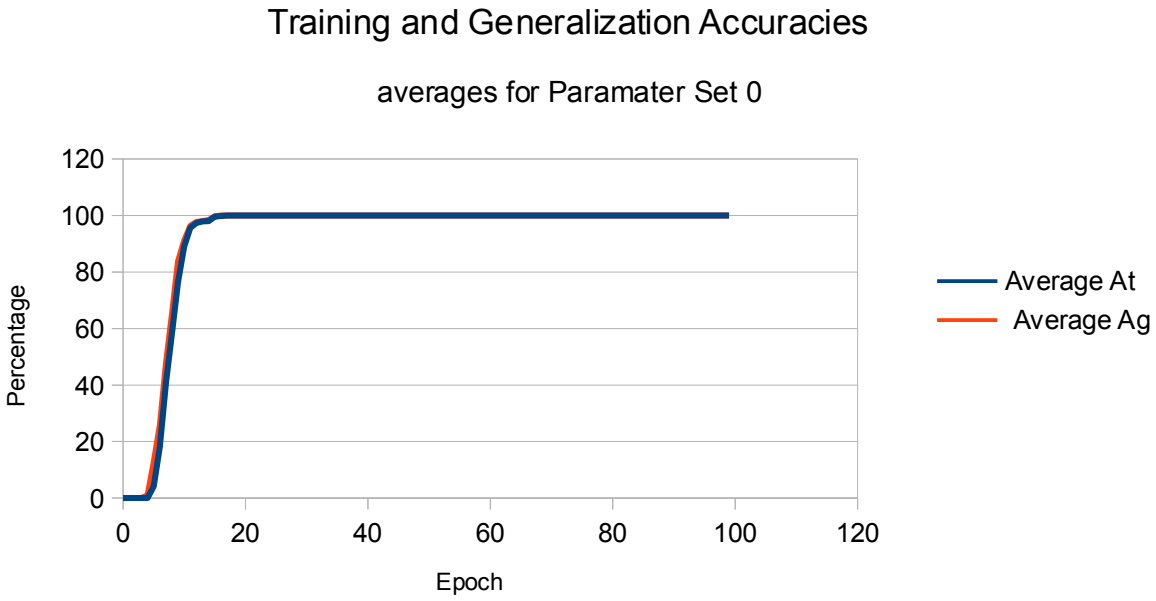

Figure 2.1 Parameter Set 0 results

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.

<u>The effect of the parameters on the performance of the Neural Network</u>

Momentum:

For the general case, a relatively large momentum value is the norm. This is quite a bold statement to make without any evidence. Keeping the parameters of P_0 constant, we define parameter sets P_{M0} to P_{Mn} in which the momentum value is adjusted in order to experimentally determine the effect thereof on the performance and convergence rate of the neural network.

One starts at a momentum value of 0.0 incrementing the value up to 1.0 with increments of 0.1 to determine the larger range of momentum values that have a meaningful effect.

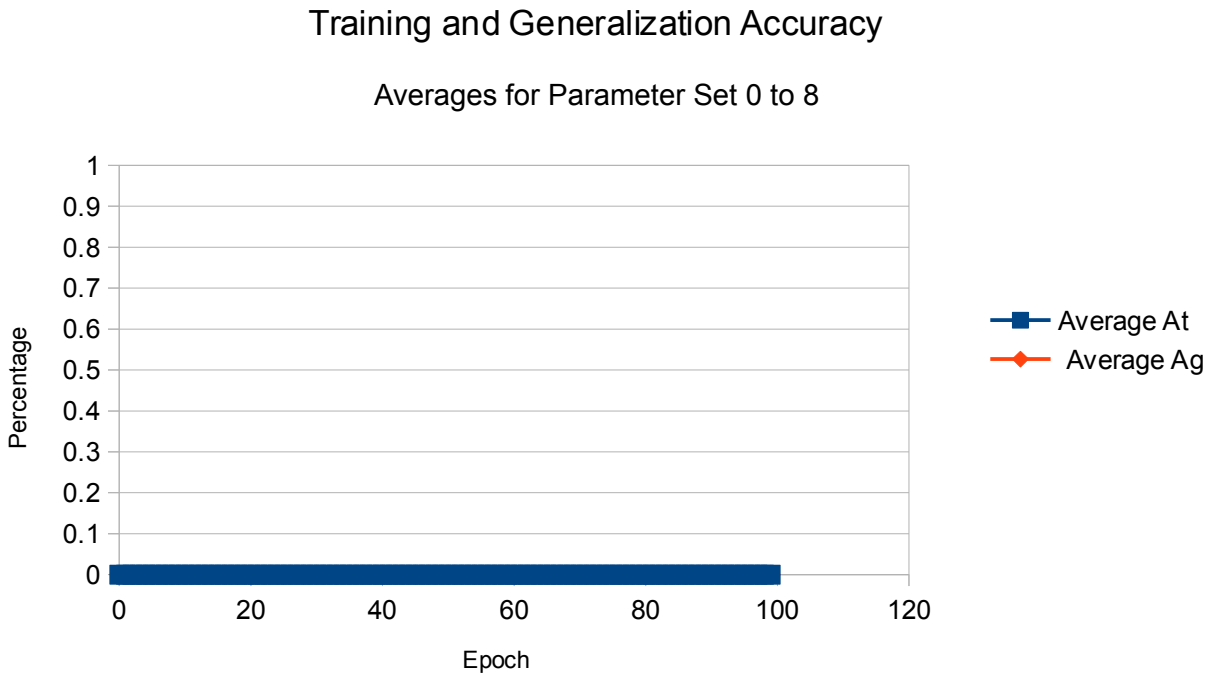

Figure 2.2 P_{M0} to P_{M8} results with momentum values 0.0 to 0.8

The previous statement claiming momentum values to be high quite clearly holds, as can be seen by Figure 2.2. Lower momentum values causes the neural network, in this case to not classify correctly at all. This phenomenon is due to the fact that oscillation occurs during the back-propagation phase. Oscillation is extremely prominent when the error surface has a very narrow minimum area, as is the case here.

Regard Figure 2.3 and Figure 2.4. With regards to Figure 2.2 it is apparent that the neural network starts to converge and classify somewhat better as the momentum value is incremented. One can thus state that the "ideal" momentum value should be somewhere in the range 0.9 to 0.999... as this can be considered the "active range".

The active parameter sets P_{A0} to P_{An} are now investigated, incrementing the momentum value by 0.01 for the range 0.9 to 1.0. Since the network converges much earlier than 100 epochs, only 30 epochs will now be allowed.

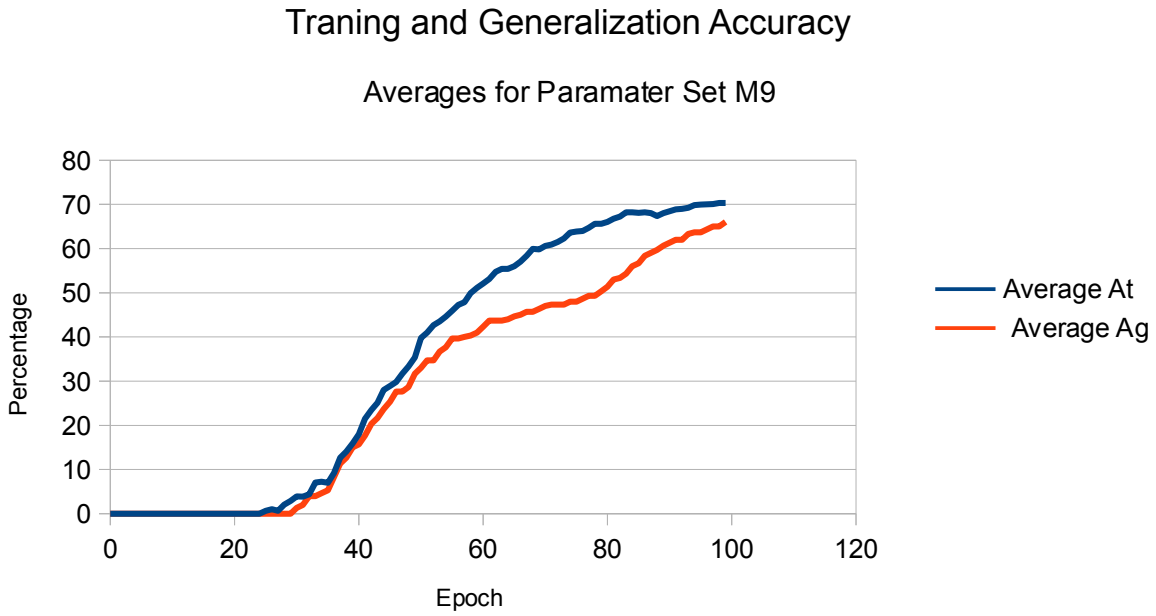

Figure 2.3 A momentum value of 0.9 being used

NOTE: When implementing a neural network for practical use one would add a stopping condition for training such that training stops when the network has fully converged. This stopping condition was omitted in favour of a better visual representation.

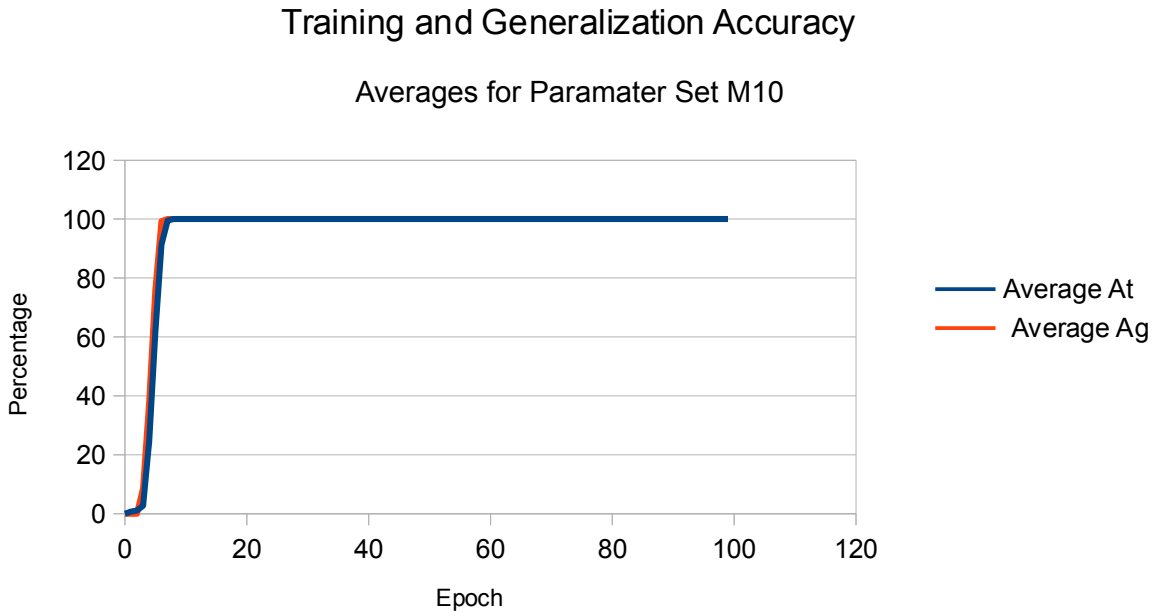

Figure 2.4 A momentum value of 0.9999... being used

Figure 2.5 shows when the network converges based on the momentum value used in $P_{AMn.}$ As can be seen from this figure P_{AM8} reaches an average of 100% training and generalization accuracy. P_{AM9} and P_{AM10} learn faster but obtain a perfect classification record later than $P_{AM8.}$ With the goal in mind of finding some "good configuration" for the neural network, $P_1 = P_{AM8.}$ P_1 will now be used in stead of P_0 in further experimentation. The sharp reader may notice that $P_0 = P_{1.}$ A lucky guess was indeed made based on incomplete data.

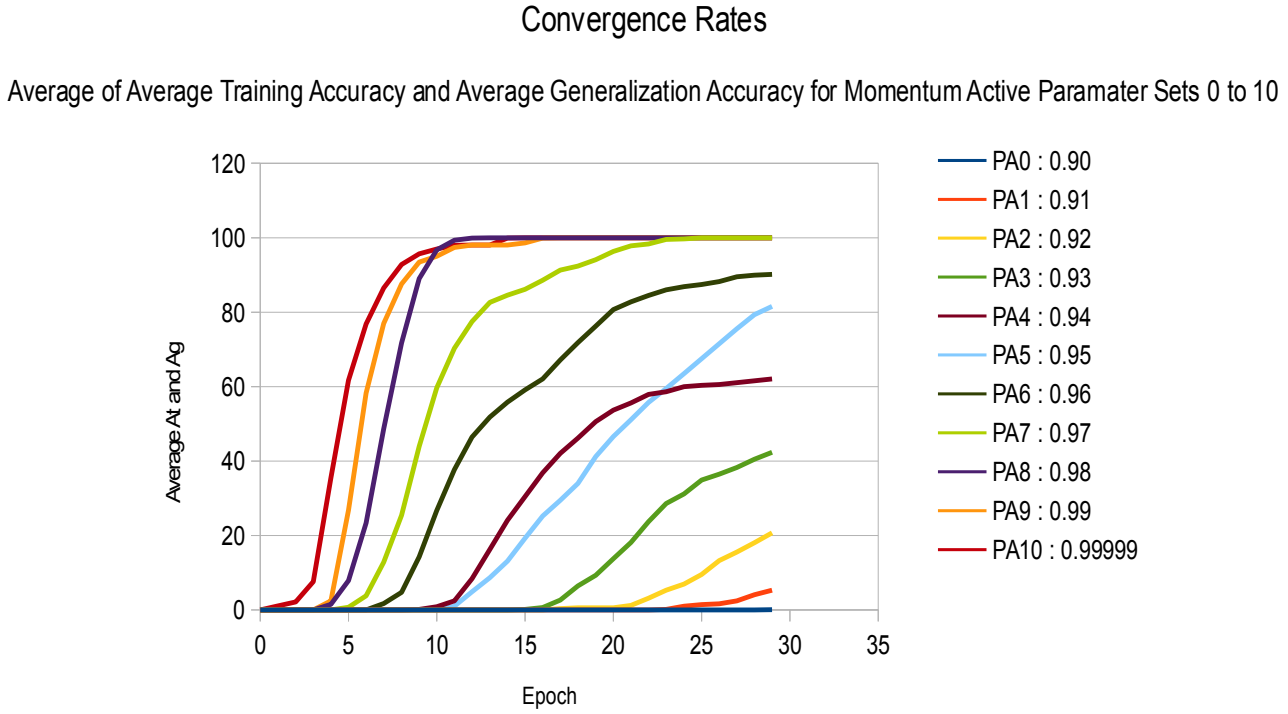

Figure 2.5 Averages to indicate converge rate of changing momentum values

Next, the effect of the learning rate is investigated.

Learning Rate:

Generally speaking, the rate at which the neural network converges is proportional to the learning rate. As was done with momentum, one attempts to find the "active range" of the learning rate : there where the configuration results in an optimal or semi-optimal network performance.

Let P_{L0} to P_{Ln} be the parameter sets with constant parameter values as in P_1 but for a variable learning rate. The learning rate will first be investigated for 0.0 to 0.1 with increments of 0.1.

Epoch	PL0	PL1	PL2	PL3	PL4	PL5	PL6	PL7	PL8	PL9	PL10
0	0	0	0.1251	0	11.7919	11.0844	3.9584	19.9163	37.0842	32.0417	29.3747
1	0	0.2917	2.8335	16.6246	41.5001	51.1246	65.4163	82.5423	60.3754	53.3751	52
2	0	0.5417	30.5422	66.7918	85.3337	92.7087	97.7497	99.875	95.9998	83.0001	81.6249
3	0	0.9583	76.4999	92.0833	99.1667	99.4583	99.2917	75.3758	99.5417	89.1667	86.5001
4	0	8.9996	93.3334	95.7083	100	100	85.9585	58.6668	86.458	81.6664	74.1248
5	0	33.2493	98.0834	98.75	99.4167	100	67.4165	55.1666	74.1665	62.4161	64.1665
6	0	59.6657	99.875	100	94.1251	100	57.3748	55.5417	64.1247	51.9999	56.7499
7	0	82.6656	100	100	86.5834	99.125	53.5831	58.4583	55.875	50.7499	52.0834
8	0	92.4573	100	100	71.4166	99	53.8331	55.5419	54.0416	50.6666	49.5834
9	0	97.1244	100	100	59.9166	99	50.6666	51.7918	52.4999	49.5	49.1249
10	0	98.2914	100	100	59.2501	98.125	49.9582	48.8334	49.2082	48	48
11	0	99.8749	100	100	62.2501	97.0833	48.9583	49.0416	45.9166	46.2501	50.0418
12	0	99.9583	100	100	58.9165	95	49.125	46.0416	44.5833	42.3749	49.9165
13	0	99.9583	100	100	59.0832	94	48.6249	47.4583	47.25	42.5417	48.2082
14	0	100	100	100	64.2081	94.0417	49.7917	47.5832	47.625	40.3748	47
15	0	100	100	100	65.5416	91.4167	47.5834	47.9999	44.8334	37.2915	43.8336
16	0	100	100	100	62.2083	85.0834	48.7082	50.2499	45.4583	38.1251	42.7084
17	0	100	100	100	59.625	83.4582	47.4167	46.2085	46.5832	42.875	42.0833
18	0	100	100	100	62.3331	83.875	47.8749	46.6248	46.5	42.0415	42.5833
19	0	100	100	100	62.625	82.25	48.7915	46.5415	41.4167	36.2501	42.3751
20	0	100	100	99.2083	60.375	78.2083	50.708	44.9998	43.375	34.4166	41.4166
21	0	100	100	97.4583	61.9999	74	52.4165	48.4166	43.5416	34	40.9167
22	0	100	100	91.5831	59.0831	72.4586	48.3332	50.3334	42.5	36.5833	39.4583
23	0	100	100	82.5413	63.1249	71.75	47.1669	49.9998	43.6251	36.125	37.5833
24	0	100	100	76.5833	63.6668	65.9999	46.7499	48.75	40.0417	37	39.7083
25	0	100	100	74.4166	63.9582	62.6667	49.8749	47.875	38.0833	37.875	42.6667
26	0	100	100	72.7915	62.7081	63.0416	49.625	49.1665	39.625	32.9584	39.2084
27	0	100	100	72.3332	60.5417	63.9582	50.0415	51.0834	41	33.3333	37.8333
28	0	100	100	68.6665	62.2917	63.0833	47.4584	48.6249	41.4167	35.9166	35.4165
29	0	100	100	68.2083	60.7499	56.5	47.0417	47.8751	39.8333	35.0417	36.5001

Figure 2.6 Experimental Data showing the Average of the Average At and Ag per epoch for 50 runs each

In cases such as this, the human eye may be better off at recognising a pattern when looking at the raw data. Regard Figure 2.6 and Figure 2.7. Figure 2.7 is simply a graph representation of the data presented in Figure 2.6. When looking at Figure 2.6 regard the pattern made by the values 100.

These results illustrate the phenomenon called “overfitting”. The neural network trains, achieves a perfect or almost perfect (or at the very least, a good) ratio of accuracy, but then worsens since it keeps training. If one were to consider which value of the learning rate should be chosen, this may be quite a tricky choice to make.

Regard $P_{L7.}$ P_{L7} trains to a 99.875% accuracy within two epochs, but then overfits greatly. P_{L2} is chosen as the preferred learning rate since it doesn't overfit within our number of epochs and causes the network to converge fairly quickly.

Since the learning rate is directly proportional to how quickly the neural network converges, it may still be a viable choice to choose P_{L7} if the problem requires a fast rate of convergence.

Let $P_2 = P_{L7,}$ where P_2 will be used for the remainder of the experimentation.

Next, the effect of the number of hidden nodes is investigated.

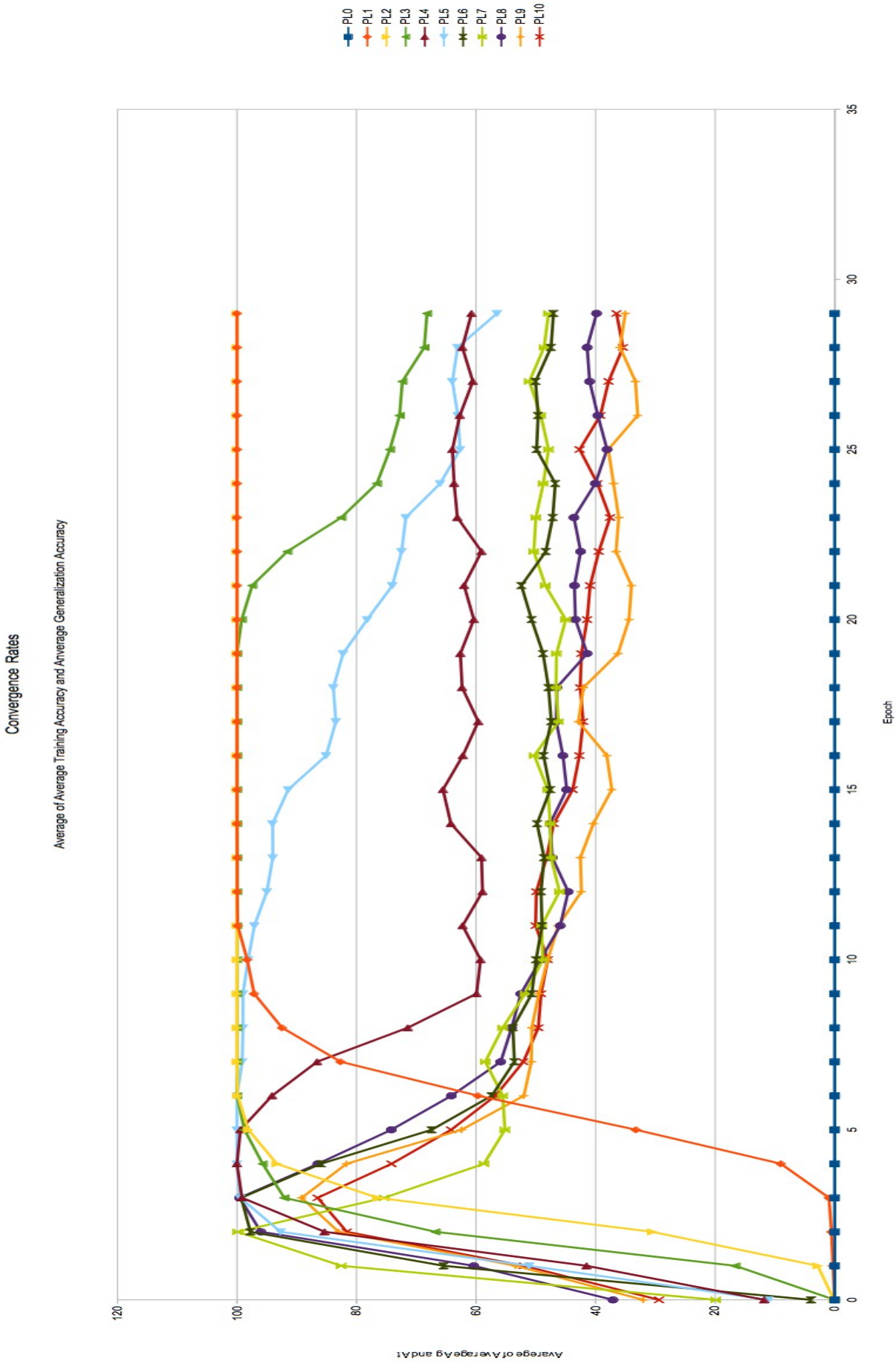

Figure 2.7 Visual representation of Figure 2.6

Hidden Nodes:

Let P_{H0} to P_{Hn} be the parameter sets with constant parameter values as in P_2 but for a variable number of hidden nodes. The number of hidden nodes will be investigated for the values 1 to 52, which is the number of input node with the sum and difference of that same number in order to establish a uniform range.

Due to the sheer amount of data generated by this approach, it was decided to instead choose the "optimal parameter set" P_{Hn} as the amount parameter set which converges first. The epoch at which P converges was recorded (overfitting ignored, since when a network converges normally a stopping condition will be applied).

The results are displayed in Figure 2.8.

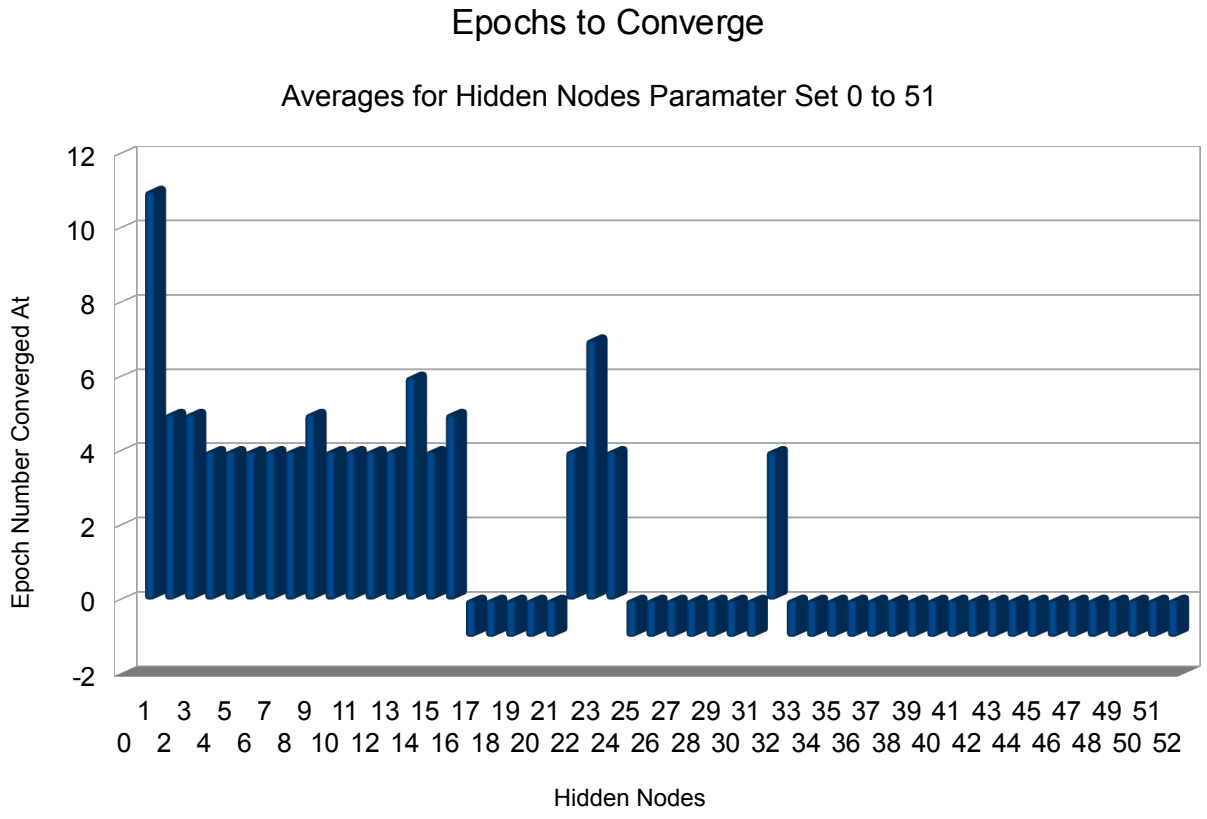

Figure 2.8 Convergence at Epoch for number of hidden nodes using P_2.

From Figure 2.8 we can clearly see that for this specific problem a low amount of hidden nodes are to be used optimally. This could be due to the simplicity of the problem. A negative epoch in Figure 2.8 shows that the network did not converge within the maximum amount of epochs.

To choose the "optimal" number of hidden nodes, the smallest number of hidden nodes with the lowest epoch number converged at is chosen. Thus a hidden node number of 4 is decided on.

Finally the optimal parameter set $P_{Optimal}$ can be defined as:

- Momentum: 0.98
- Learning Rate: 0.2
- Hidden Nodes: 4

Conclusion

In conclusion, we now compare our $P_{optimal}$ which was experimentally determined with our initial solution P_0 in order to establish if truly a "better" configuration was obtained.

Taking the data from Figure 2.1 and showing only the first 10 epochs, as well as the data from P_{H4} which is equal to $P_{optimal}$ Figure 2.9 is thus constructed.

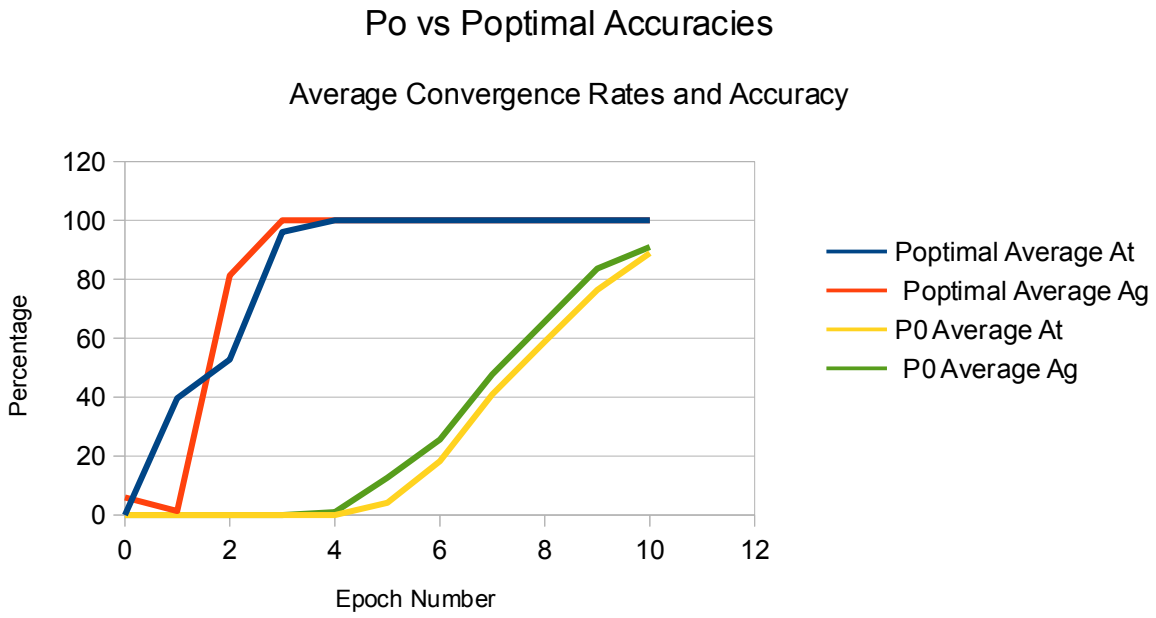

Figure 2.9 Comparison between P_0 and $P_{optimal}$

It is quite clear that the convergence rate and accuracy of the network significantly improved experimentally by using $P_{optimal}$ instead of P_0.

DISCLAIMER: It is however possible that the solution obtained here is a local optima due to the stochastic nature of the original starting solution. In order to

optimise the network better in terms of P, one would perhaps consider utilising an optimisation algorithm such as a simple Hill Climber or Genetic Algorithm.