On the Optimal Learning Rate Size for the Generalization Ability of Artificial Neural Networks in Forecasting TCP/IP Traffic Trends

Vusumuzi Moyo
Department of Computer science,
University of Fort Hare
P.O Box X1314, Alice, South Africa

Khulumani Sibanda
Department of Computer science,
University of Fort Hare
P.O Box X1314, Alice, South Africa

ABSTRACT

Artificial Neural Networks (ANNs) have attracted increasing attention from researchers in many fields. One area in which ANNs have featured prominently is in the forecasting of TCP/IP network traffic trends. Their ability to model almost any kind of function regardless of its degree of nonlinearity, positions them as good candidates for predicting self-similar time series such as TCP/IP traffic. Inspite of this, one of the most difficult and least understood tasks in the design of ANN models is the selection of the most appropriate size of the learning rate. Although some guidance in the form of heuristics is available for the choice of this parameter, none have been universally accepted. In this paper we empirically investigate various sizes of learning rates with the aim of determining the optimum learning rate size for generalization ability of an ANN trained on forecasting TCP/IP network traffic trends. MATLAB Version 7.4.0.287's Neural Network toolbox version 5.0.2 (R2007a) was used for our experiments. We found from the simulation experiments that, generally small learning rates produced consistent and better results, whereas large learning rates appeared to cause oscillations and inconsistent results. Depending on the difficulty of the problem at hand, it is advisable to set the learning rate to 0.1 for the standard Backpropagation algorithm and to either 0.1 or 0.2 if used in conjunction with the momentum term of 0.5 or 0.6. We advise minimal use of the momentum term as it greatly interferes with the training process of ANNs. While experimental results cannot cover all practical situations, our results do help to explain common behavior which does not agree with some theoretical expectations.

General Terms

Pattern Recognition, Machine Learning

Keywords

Generalization ability, Artificial Neural Networks and Learning rate size

1. INTRODUCTION

Artificial Neural Networks (ANNs) have been used in many fields for a variety of applications, and proven to be reliable. Inspired by biological systems, particularly the observation that biological learning systems are built of very complex webs of interconnected neurons, ANNs are able to learn and adapt from experience. They have demonstrated to be one of the most powerful tools in the domain of forecasting and analysis of various time series [1]. Time Series Forecasting (TSF) deals with the prediction of a chronologically ordered variable, and one of the most important application areas of TSF is in the domain of network engineering. As more applications vital to today's society migrate to TCP/IP networks it is essential to develop techniques that better understand and predict the behaviour of these systems.

TCP/IP network traffic forecasting is vital for the day to day running of large/medium scale organizations. By improving upon this task, network providers can optimize resources (e.g. adaptive congestion control and proactive network management), allowing an overall better Quality of Service (QoS). TCP/IP forecasting also helps to detect anomalies in the network. Security attacks like Denial-of-Service (DoS) or even an irregular amount of SPAM can be detected by comparing the real traffic with the values predicted by forecasting algorithms, resulting in economic gains from better resource management.

Literature from various authors has shown that unlike all other TSF methods, ANNs can approximate almost any function regardless of its degree of nonlinearity [2,3]. This positions them as good candidates for modeling non linear and self similar time series such as TCP/IP network traffic. Inspite of this huge advantage, ANNs are not completely absolved from any problems. One major issue that limits the applicability of ANN models in forecasting tasks is the selection of the optimal size of the learning rate. The learning rate also referred to as the step size parameter, determines how much the weights can change in response to an observed error on the training set. It is considered as a key parameter for a successful ANN application because it controls the size of each step toward the minimum of the objective function [4]. This has a profound influence on the generalization capabilities of the ANN [5]. Generalization is a measure that tells us how well the ANN performs on the actual problem once training is complete. Once the ANN can generalize well, it means that it is capable of dealing with new situations such as a new additional problem or a new point on the curve or surface.

Although individual studies have been conducted and some form of heuristics provided for the selection of the size of the learning rate, none have been universally accepted as the results are largely contradictory. Some researchers such as Richards (1991) [6] suggest that the larger the learning rate the better the ANN generalization whilst others such as Wilson and Martinez (2001) [7] are of the opposite view. In any case most of these studies have been conducted on synthetic datasets e.g. (Glass-Mackey time series) making the solutions thereof difficult to apply to real world problems. In fact, until a number of experiments have been done, it is unknown which size of the learning rate will provide optimum solutions. Hence new users of ANNs particularly in the forecasting of TCP/IP network traffic domain, usually blindly employ trial-and-error strategies to determine the optimal values for this parameter without any prior substantive guidelines. This results in the addition of more time to the already slow process of training an ANN.

In this paper the effect of different sizes of learning rates on the generalization ability of ANNs is empirically investigated. Although the results presented in this paper are for a particular case study, they provide a valuable guide for engineers and scientists who are currently using, or intend to use ANNs.

2. ARTIFICIAL NEURAL NETWORKS

Haykin (1998) [8] defines ANNs as "physical systems which can acquire, store and utilize experimental knowledge". The basic unit of an ANN is a neuron. An artificial neuron acts in the same way as a biological neuron; each has a set of inputs and produces an output based on the inputs. A biological neuron produces an output by comparing the sum of each input to a threshold value. Based on that comparison it produces an output. In addition, it is able to differently weigh each input according to the priority of the input. The inputs and outputs of a biological neuron are called synapses and these synapses may act as inputs to other neurons or as outputs such as muscles. Thus it creates an interconnected network of neurons which combined produce an output based on a number of weights, sums and comparisons. One motivation for ANN systems is to capture this kind of highly parallel computation based on distributed representations.

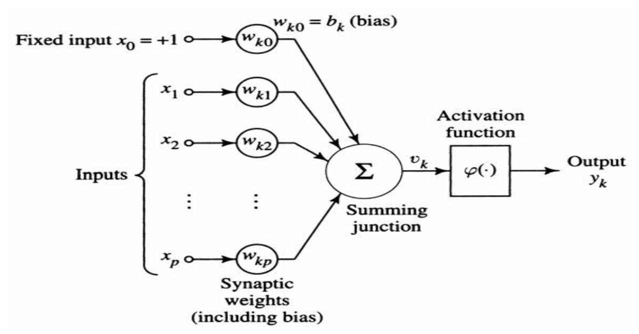

Fig 1: An artificial neuron (adapted from [8])

Fig 1 shows the typical structure of an artificial neuron, the inputs are denoted by $x_1, x_2 \dots x_p$ and weights are denoted by $w_{ko}, w_{k1}, w_{k2} \dots w_{kp}$. The neuron calculates the weighted sum w_k, x as:

$$w_k, x = \sum_{i=1}^{p} w_{ki} x_i \qquad (1)$$

The output of the neuron is governed by the activation function, which acts as a threshold. The output is given by:

$$y_k = f\left(\sum_{i=1}^{p} w_{ki} x_i + b_k\right) \qquad (2)$$

Where f *is* the activation function, (b_k) is the bias and y_k is the output signal.

Among the various types of ANN models, Multilayer perceptron (MLP) is the most extensively applied to a variety of problems. MLPs are formed by several neurons arranged in groups called layers. The most popular and the simplest MLP consist of three layers, an input layer, a hidden layer, and an output layer. The ANN thus has a simple interpretation as a form of input-output model, with the weights and thresholds (biases) being the free parameters of the model. The sliding time window approach is the most common MLP model for forecasting. It takes as inputs the time lags used to build a forecast and it is given by the overall formula:

$$X_{p,t} = w_{o,0} + \sum_{i=I+H}^{I+H} f \sum_{s=1}^{k} \sum_{r=1}^{w_s} X_{st-L_{sr}} w_{i,j} \qquad (3)$$

Where $w_{i,j}$ is the weight of the connection from node j to i (if j=0 then it is a bias connection), o denotes the output node and f is the Logistic sigmoidal activation function.

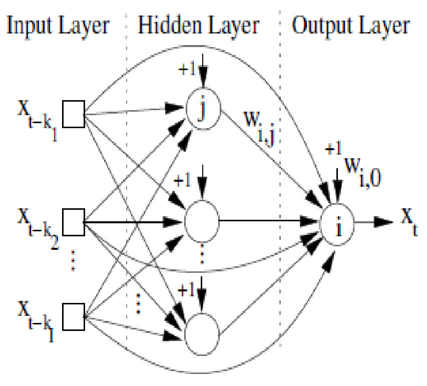

Fig 2: Sliding time window MLP (adapted from [8])

In the vast majority of papers that deal with the prediction and forecasting of TCP/IP traffic, Feedforward networks optimized with the aid of the Backpropagation (BP) algorithm have been used. According to Haykin (1998) [8], "*this is because BP is easy to implement and fast and efficient to operate*". The BP process is commenced by presenting the first example of the desired relationship to the network. The input signal flows through the network, producing an output signal. The output signal produced is then compared with the desired output signal and the errors propagated backwards in the network. In this work we have adopted the BP sliding time window approach for our ANN models.

3. METHODOLOGY

In our approach for the study we used experimental method which is a proven method for testing and exploring cause and effect relationships. The benefit of using this method is that it allows the control of variables thereby enabling the isolation of a particular variable to observe the effects due to that variable alone. In this case our interest was on the effects of the size of the learning rate on ANN generalization. The software used for the purposes of this study is Matlab Version 7.4.0.287 (R2007a). Matlab is an application software and programming language with interfaces to Java, C/C++ and FORTRAN. In this study, Matlab provides an environment for creating programs with built-in functions for performance metrics and forecasting using its Neural Networks toolbox Version 5.0.2 (R2007a). The computer used to conduct this study is an Intel(R) Core(TM) 2CPU6300@1.86GHz. The data was collected from the South African Tertiary Institutions Network (TENET) website (www.TENET.ac.za). We analysed network traffic data which comprised inbound traffic in (bits/ sec) from the University of Fort Hare VC Alice

Boardroom 101 – Fa 0/1 router. The data spanned from the 1st of March 2010 from 02:00 hours to the 21st of September 2013 02:00 hours in daily intervals, equating to 700 observations. As in all practical applications the data suffered from several deficiencies that needed to be remedied before use for ANN training. Preprocessing was done which included Linear interpolation to fill in missing values, which amounted to 7 such observations. Matlab Neural Network toolbox has a built-in function, *mapminmax* which scales the data down before training so that it has 0 mean and unity standard deviation and then scales it up again after training, so as to produce outputs with 0 mean and unity standard deviation. The data was partitioned into training and testing sets. 547 samples were allocated to the train set whilst 182 were allocated to the test set.

To investigate the effect of the learning rate on the generalization ability of ANNs in predicting TCP/IP network traffic trends, various layered, fully connected ANNs with a single input neuron, Logistic sigmoid activation function in the hidden layers and a Linear output neuron were examined. We carried out investigations on both single and double hidden layer ANNs. A supervising script was written to compute the ANN inputs and targets. On visual inspection of the time series a sliding time window of size 150 was arbitrarily chosen. Training was stopped after 1000 epochs and the generalization performance of the ANNs tested by presenting the unknown test set to the ANNs and calculating the Root Mean Squared Error (RMSE) between the actual and predicted values. RMSE is a dimensionless value calculated to compare ANN performance. The RMSE on the test set (MSE_{te}) was calculated using the following equation:

$$RMSE_{te} = \sum_{p=1}^{P_{te}} (d_p - o_p)^2 \qquad \textbf{(4)}$$

where d_p is the desired output for each input pattern and o_p is the actual output produced by the ANN. In order to minimize the random effect of the initial weights on results, for each experiment conducted, 4 training runs were made and the results averaged. We also ensured that all other variables that could potentially affect the quality of results remain constant. Hence throughout the duration of our investigations the size of the training set was fixed at 547 samples and of the test set at 182 samples, weights were randomly initialised in the range of [-0.5, 0.5], the BP (trainlm) training rule was the weight updating algorithm and the momentum was fixed at 0.2 (unless stated otherwise).

4. THE EXPERIMENTS

The first set of experiments involved several single hidden layer ANNs trained on various learning rates on separate occasions. The ANNs selected for examination had 20, 40, 60 and 80 hidden neurons. These were selected based on stability times during preliminary investigations. The learning rates were varied according to 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1. The choice of the learning rates was purely done on an arbitrary basis. Fig 3 shows the results of those experiments.

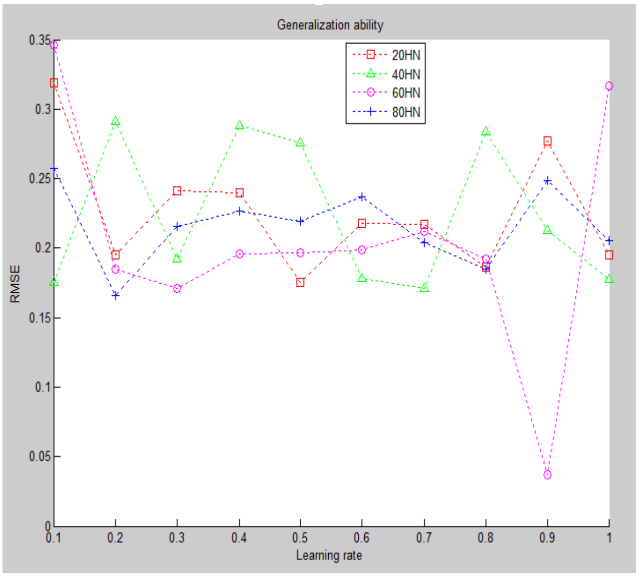

Fig 3: Generalization errors (RMSE) for the different learning rates at various network architectures.

In the second set of experiments, we trained a single hidden layer ANN of network architecture of 60 hidden neurons, at learning rates of 0.2, 0.4, 0.6, and 0.8 for a different number of epochs in order to ascertain how the impact of learning rate on generalization ability varies with an increase in number of training iterations. The choice of the values of the learning rates was motivated by a similar case study by Attoh-Okine (1999) [4] who utilised the same values of the learning rates for his experiments. The reason a 60 hidden neuron ANN was selected for investigation was because it exhibited better generalization performance than other network architectures during preliminary experiments conducted prior to these investigations. Fig 4 shows the generalization errors from the experiments.

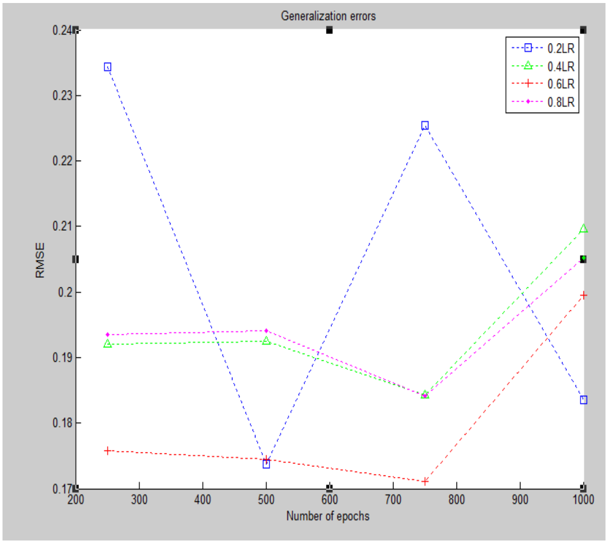

Fig 4: Generalization errors (RMSE) for the different learning rates at various training iterations.

In the third set of experiments, we assessed the performance of two hidden layer architecture. After conducting several trial runs on various ANN architectures from preliminary experiments a two hidden layer ANN of architecture (5, 35) i.e. 5 first hidden layer neurons and 35 second hidden layer neurons, was chosen as the bases for conducting these investigations. The bases for arriving at the chosen architecture was the fact that the (5, 35) network architecture exhibited better generalization performance amongst all the two hidden layer architectures examined in the preliminary experiments. We trained the ANN on two different learning rates of 0.1 and 0.01 on separate runs. The results of the generalization performance are shown in Fig 5.

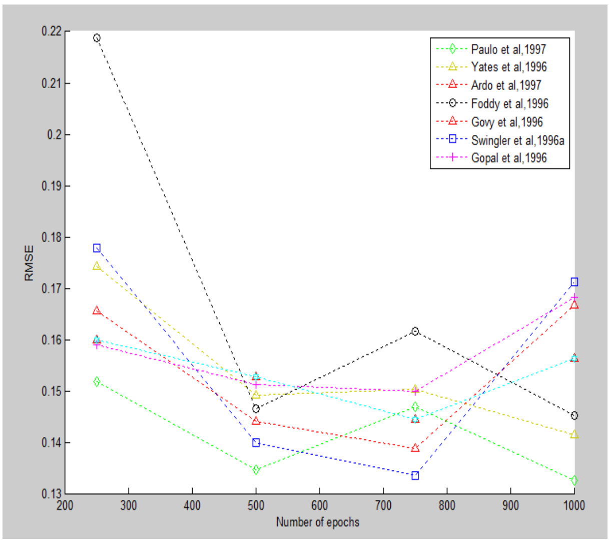

Fig 0: Generalization errors (RMSE) for different learning rates at various training iterations for (5, 35) ANN.

Finally, in order to make fair conclusions additional experiments were conducted using different combinations of learning rates and momentum values so as to assess whether the effect of different learning rates is the same regardless of momentum. These 2 parameters have been suggested to be quite closely related in literature [4]. We trained as in the previous case, the same ANN of network architecture (5, 35) using different heuristics of learning rates and momentum combinations provided in literature. The generalization errors are shown in Fig 6.

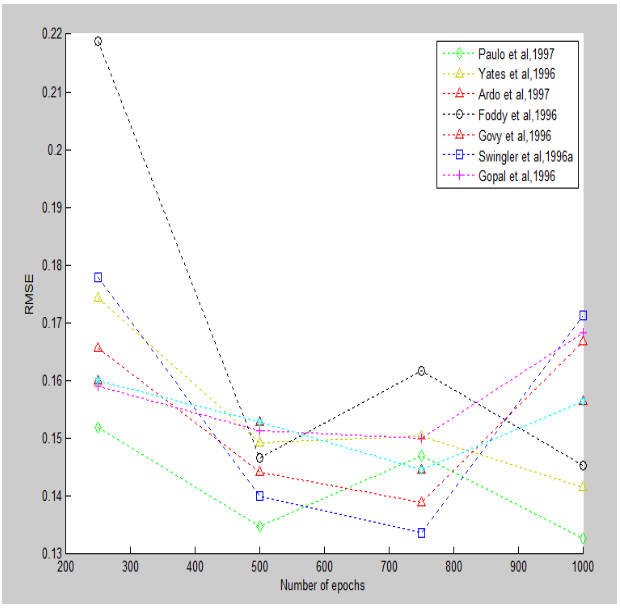

Fig 6: Generalization errors (RMSE) for different learning rates and momentum combinations at various training iterations for (5, 35) ANN.

5. RESULTS AND DISCUSSIONS

In this section we discuss the empirical results on experiments regarding the relationship between ANN generalizations and learning rate. We begin our analysis by examining the results in Fig 3, which depict the generalization errors of various single hidden layer ANNs trained on various learning rates. From Fig 3, it would appear that indeed selecting the optimum learning rate for a given problem is a complex task. From the results, we note that the best performing ANN is architecture of 60 hidden neurons, trained on a learning rate of 0.9. We note that for many of the ANNs examined the lowest generalization errors occurred at lower learning rates mostly between 0.2 to 0.6, although it is difficult to pinpoint a single universal value which we can safely conclude to be effective in all cases. However, in many cases a learning rate of 0.6 seems to give the most reasonable generalization errors. Our results indicate that the smaller ANNs performed badly in these experiments, especially the 20 hidden neuron and 40 hidden neuron architectures. These two architectures also displayed the most erratic behaviour, whether trained on smaller or larger learning rates, indicating the intrinsic relationship between learning rate and network architecture, emphasizing the sensitivity of smaller ANNs to learning rate. Generally small learning rates produced consistent and better results, whereas large learning rates appeared to cause oscillations and inconsistent results.

We then trained a 60 hidden neuron ANN at different learning rates of 0.2, 0.4, 0.6, and 0.8 for a different number of epochs in order to ascertain how the impact of learning rates on generalization ability varies with an increase in the number of training iterations. The results of that endeavour are shown in Fig 4. From Fig 4 note that the best generalization performance is attained when the ANN is trained on a learning rate of 0.6. It is evident from the results that the generalization ability of the ANN increases as the size of the learning rate increases, however this is only true up to a certain threshold, beyond which any further increment in learning rate results in adverse effects. This is true judging by the generalization errors incurred in Fig 4. From Fig 4, note that the ANN performs the worst and is mostly erratic at a learning rate of 0.2, when the learning rate is increased to 0.4, the generalization ability of the ANN dramatically improves, at a learning rate of 0.6, the ANN is at its best in terms of generalization ability, however at a learning rate of 0.8, instead of following a similar antecedent, the generalization ability of the ANN decreases.

It is also quite interesting to note that for all the (learning rate-number of iterations) combinations the ANN performs the best in terms of generalization ability at 750 epochs. Additionally, training time was fast for learning rates of 0.2 and 0.4, and reasonably fast for rates of 0.6 and 0.8. Note that larger learning rates take many epochs to reach their maximum generalization ability, which is ultimately poor anyway. The small learning rates take longer to reach the same accuracy, but yield no further improvement in accuracy.

We now turn our discussions to the results shown in Fig 5 which show the generalization performance of a 2 hidden layer ANN of network architecture (5, 35). The results in Fig 5 indicate that for the task of forecasting a TCP/IP network traffic time series, a 2 hidden layer ANN is more sensitive to smaller leaning rates than larger ones. The generalization errors show that the ANN has significantly better generalization ability when trained on a learning rate of 0.01 than 0.1, this is more pronounced at an epoch size of 500. Although a learning rate of 0.1 outperforms the 0.01 learning rate between 750 to 800 epochs, we cannot not read much into this as it is largely short-lived

To conclude with the discussions we examine the results of different combinations of learning rates and momentum heuristics given by various authors. The reason for such was to ascertain whether the effect of different learning rates on generalization ability is the same regardless of momentum. The experiments were conducted on a (5, 35) network architecture. The results of this assessment are given in Fig 6. Several conclusions can be deduced from these results regarding the effect of different learning rates and momentum combinations on the generalization ability of the ANNs.

From the results, for the case study considered we note that the 0.1-0.9 (learning rate-momentum) combination given by Foody et al. (1996) [9] fails to produce satisfactory generalization ability. We note extensive oscillatory behaviour in terms of generalization errors at those values of the learning rate and momentum, this as stated by Kavzoglu (1999) [10] could be attributed to the fact that the use of large momentum term increases the effect of oscillations by extending the steps taken in faulty direction or perhaps the ANN could have been stuck in a local minima resulting from the large momentum term.

It can be seen from Fig 6 that the lowest generalization errors are produced by small learning rate and momentum combinations, such as 0.25-0.2 of Swinger (1996) [11] and 0.1-0.3 of Ardö et al. (1997) [12]. Another important observation is that the addition of the momentum term to the training considerably slowed down the learning process.

Although the selection of an appropriate combination of the learning rate and momentum is a mammoth task, it appears that a very small learning rate, roughly 0.1 and a moderately high momentum term between 0.2–0.4 provided neo-optimal solutions for forecasting TCP/IP traffic trends.

Now, to answer the question: *How does the use of a momentum term affect the way in which an ANN responds to the learning rate*? From Fig 6, we see that the generalization errors obtained in those tests where various combinations of learning rate and momentum values were tried out, were in essence not that significantly different from the generalization errors obtained when a momentum value of 0.2 was used as in the previous experiments. However, the behaviour of the ANNs was less controlled with increasing momentum, as a result of the larger steps taken in weight space. In fact, when a momentum value of 0.8 was used in conjunction with a learning rate of 0.2 and when a momentum value of 0.9 was used in conjunction with learning rates of 0.1, the steps taken in weight space were too large and divergent behaviour occurred during training. Discussions on the sizes of training epochs as a function of the network architecture and the resultant effect upon the learning rate and momentum needs further insight so that a more generalized result can be proposed. It is also important to note that during these experiments, the ANNs did not show any signs of overfitting.

6. CONCLUSIONS AND FUTURE WORK

The experimental results regarding the relationship between the learning rate and network generalization are discussed in section 5. We note that for a single hidden layer network, a learning rate of 0.6 gave the most reasonable generalization errors, particularly at an epoch size of 750, and a 2 hidden layer ANN was more sensitive to larger learning rates than smaller ones. We also noted that smaller learning rates decreased the training time quite significantly.

With regards to the impact the momentum term has on the relationship between the learning rate and generalization ability of ANNs, we discovered that a very small learning rate, roughly of about 0.1 and a moderate momentum between 0.2–0.4 provided neo-optimal solutions for the task considered. However, the behaviour of the ANNs became less controlled with increasing momentum, as a result of the larger steps taken in weight space. We conclude that the degree of effects of learning rate and momentum on the model differ as stated by Wilson and Martinez (2001) [7]. The learning rate is more powerful than momentum, as when a large learning rate and small momentum are achieved, the result is more precise than the opposite.

For researchers in this domain seeking simply for the learning rate that produces the fastest convergence should probably settle on a learning rate of 0.4. However, doing so would mean sacrificing generalization ability, which could be more efficiently achieved by using a larger learning rate, therefore trade-off between these two variables is an ardent necessity. Unfortunately for these experiments we did not require the training process to converge, rather, the training process is used to perform a direct search of a model with superior generalization performance.

We also advise, depending on the difficulty of the problem at hand to set the learning rate to 0.1 for the standard Backpropagation algorithm and to either 0.1 or 0.2 if used in conjunction with the momentum term of 0.5 or 0.6,its important not to set the momentum term too large as it would cause the ANN to be greatly unstable. If possible we advise minimal use of the momentum term as it greatly interferes with the training process of ANNs.

As with almost any area of research, progress leads toward more questions. Based on the research carried out in this study, our results suggest considerable potential for future work. We plan in extending our investigations to new self-similar and chaotic time series and to other ANN models and learning parameters. In addition, more testing is needed to evaluate the applicability of our guidelines to other datasets to be able to make claims about their robustness and to validate the effectiveness of the conclusions reached in this research.

In order to improve and extend the investigations reported in this work, in addition to constant learning rates, the use of adaptive learning rate strategies could be examined and their effects on ANN generalization ability compared to those produced by their counterparts. Another issue which we can possibly look at is using a variable momentum value. A variable momentum value is currently being researched and its impact upon the generalization ability is not exactly known to date.

As this study was limited to Feed-forward ANN learning problems with the Backpropagation learning algorithm, it could be also beneficial to investigate the effects of the size of the learning rate on the performance and generalization ability of other ANN models, including Self Organizing Maps (SOM) and Learning Vector Quantization (LVQ), with the aim of deriving some general conclusions that can be used to construct some guidelines for users in the design of these particular network models.

7. ACKNOWLEDGMENTS

This work is based on the research undertaken within the TELKOM Coe in ICTD supported in part by Telkom SA, Tellabs, SAAB Grintek Technologies, Eastell, Khula Holdings, THRIP, GMRDC and National Research Foundation of South Africa (UID: 86108). The opinions, findings and conclusions or recommendations expressed here are those of the authors and none of the above sponsors accepts liability whatsoever in this regard.

8. REFERENCES

[1] S. Chabaa, "Identification and Prediction of Internet Traffic Using Artificial Neural Networks," *J. Intell. Learn. Syst. Appl.*, vol. 02, no. 03, pp. 147–155, 2010.

[2] R. Aamodt, "Using Artificial Neural Networks To Forecast Financial Time Series," Norwegian university of science and technology, 2010.

[3] H. Tong, C. Li, J. He, and Y. Chen, "Internet Traffic Prediction by W-Boost : Classification and Regression ," *Neural Comput.*, vol. 2, no. 973, pp. 397–402, 2005.

[4] Attoh-Okine, N.O., 1999. Analysis of learning rate and momentum term in backpropagation neural network algorithm trained to predict pavement performance. *Advances in Engineering Software*, 30(4), pp.291–302. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0965997898000714.

[5] Chen, C.J. & Miikkulainen, R., 2001. Creating Melodies with Evolving Recurrent Neural Networks. *In Proceedings of the 2001 International Joint Conference on Neural Networks*, 2(2), pp.20–60.

[6] E. Richards, "Generalization in Neural Networks,Experiments in Speech Recognition," University of Colarado, 1991.

[7] Wilson, D.R. & Martinez, T.R., 2001. The need for small learning rates on large problems. *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, 1, pp.115–119. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=939002

[8] S. Haykin, *Neural Networks: A comprehensive foundation*, Second. Pearson, 1999, pp. 2–3

[9] Foody, G., Lucas, R. & Curran, M., 1996. Estimation of the areal extent of land cover classes that only occur at a sub-pixel level. *Canadian Journal of Remote Sensing*, 22(4), pp.428–432.

[10] Kavzoglu, T., 1999. Determining Optimum Structure for Artificial Neural Networks. In *In proceddings of the 25th Annual Technical Conference and Exhibition of the Remote Sensing Society*. Cardiff, UK, pp. 675–682

[11] Swinger, K., 1996. Financial prediction. *Journal of Neural Computing and Applications*, 4(4), pp.192–197.

[12] Ardö, J., Pilesjö, P. & Skidmore, A., 1997. Neural networks, multitemporal Landsat Thematic Mapper data and topographic data to classify forest damages in the Czech Republic. *Canadian Journal of Remote Sensing*, 23(2), pp.217–229.