STUDY OF MODIFIED TRAINING ALGORITHM FOR OPTIMIZED CONVERGENCE SPEED OF NEURAL NETWORK

By

KANG MIEW HOW

A Dissertation submitted for partial fulfillment of the requirement for the degree of Master of Science (Microelectronic Engineering)

JUNE 2016

ACKNOWLEDGEMENT

I would like to thank everyone who contributed to the successful completion of this research. I would like to express my gratitude to my supervisor, Dr. Nor Muzlifah Mahyuddin for her invaluable advice, guidance and enormous patience throughout the development of the research. Thanks for all the guidance and constructive comments throughout this research time.

Besides, I would like to thanks to my degree lecturer, Mr Lau Kean Hong for sharing the basic artificial neural network (ANN) C++ codes to me as a starting point for the research.

Last but not least, I would also like thanks to my loving family, friends and colleagues who had helped and given me encouragement throughout this difficult time. Thanks for everyone for all the support and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial neural network
BP	Back-propagation
MSE	Mean square error
SGD	stochastic gradient descent
TTA	Total Training Accuracy
TGA	Total Generalization Accuracy
TVA	Total Validation Accuracy
ALR	Adaptation Learning Rate Algorithm
ALRS	Automatic Learning Rate Selection

ABSTRACT

Artificial neural network (ANN) are widely used as an engineering approach to mimic the human brain activities. It has applied in different aspects such as pattern recognition, alphabet or digit classification, handwriting recognition, speech recognition, fingerprint identification, data mining, robots and etc. Back-propagation is the most common artificial neural training algorithm, however it is suffering with the slow convergence rate issue. A study to improve the slow convergence rate of a neural network without sacrificing the accuracy of the network are carried out. In this research, a hand-written character recognition model are implemented in C++ programming with ability to classify digits 0, 1, 2, and 3. This model are built up with 64-40-4 neural network where input data are 8 x 8 dimension image and output are classified to 4 digits which are 0, 1, 2 and 3. Two modified algorithms are proposed in this research, which are mixture of the momentum algorithm with different learning rate algorithms. First proposed algorithm is the combination of momentum algorithm with adaptive learning rate (ALR) algorithm, and second proposed algorithm is the combination of momentum algorithm with automatic learning rate selection (ALRS) algorithm. Convergence rate are showed 18% improvement in the algorithm mixture with ALR algorithm, however there is no significant improvement of timing for algorithm mixture with ALRS algorithm.

ABSTRAK

Rangkaian neural buatan (RNB) adalah cara kejuruteraan yang digunakan secara meluas untuk meniru aktiviti otak manusia. Cara ini telah berjaya dilaksanakan di dalam banyak bidang yang berbeza seperti pengelasan digit, pengecaman tulisan tangan, pengiktirafan ucapan, pengenalan cap jari, perlombongan data, robot dan lain-lain. Algoritma perambatan balik merupakan algoritma yang paling terkenal digunakan di dalam RNB, tetapi algoritma ini menderita dengan isu kadar penumpuan yang perlahan. Kajian untuk meningkatkan kadar penumpuan yang perlahan ini tanpa mengorbankan ketepatan sistem telah dijalankan. Dalam penyelidikan ini, model pengecaman digit tulisan tangan telah dilaksanakan dalam C++ dengan keupayaan untuk mengecam digit 0, 1, 2 dan 3. Model ini dibina dengan 64-40-4 RNB dengan data input 8 x 8 dimensi dan 4 output iaitu digit 0, 1, 2 dan 3. Dua algorithma yang diubahsuai telah dicadangkan dalam penyelidikan ini iaitu campuran algorithma momentum dengan algorithma kadar pembelajaran yang berbeza. Algorithma pertama yang dicadangkan adalah gabungan algorithma momentum dengan algorithma kadar pembelajaran adaptif (ALR). Algorithma kedua yang dicadangkan adalah gabungan algorithma momentum dengan algorihtma autumatik pemilihan kadar pembelajaran (ALRS). Algorithma pertama menunjukkan 18% pengingkatan untuk melanjutkan kadar penumpuan pembelajaran RNB. Bagaimanapun, tiada pengingkatan yang ketara bagi algorithma kedua.

CHAPTER 1

INTRODUCTION

1.0 Background

Artificial neural network (ANN) is an engineering approach used to imitate the human brain's activities. ANN can be trained to solve a lot of complexity problems such as pattern recognition (Bishop, 1995), alphabet or digit classification, handwriting recognition, speech recognition, series prediction (Casas, 2012), fingerprint identification (Werner, et al., 2016), data mining, robots and etc. Typically, ANN is the network which uses to estimate or approximate functions.

An artificial neural network can be distinguished by three portions. They are the network architectures (the connection structure between the neurons), the training algorithms (weight-setting method of a network) and also the activation functions used for the network. Different combination of these three portions can be used to solve different aspects of tasks. Back-propagation algorithm is the common training algorithm used to train a neural network nowadays. It is popular due to easy implementation and able to provide a high accuracy output results (Casas, 2012). However it also has some significant limitations such as slower convergence rate and not guaranteed to find a global minimum but only a local minimum (Pan, et al., 2013).

To overcome these shortcomings, there are many different training algorithms published by researchers to improve the performance of the neural network. In this research, analyzing and comparing the existing algorithms are carried out and then a modified algorithm which optimize the network convergence speed are proposed.

In Section 1.1, problem statements of this research are described. Section 1.2 lists down the objective of the research and Section 1.3 scopes down the research to a specific area. Lastly, Section 1.4 states the thesis outline.

1.1 Problem Statements

ANNs can solve different tasks or problems by learning from data. To obtain a precise and accuracy outcome for a complex task, ANN will require to learn from a huge number of input data. For example, ANN is used to perform a character recognition. In this case, ANN will require to learn from a set of known input data first before it can start to identify an unknown character. The requirement of larger training dataset causes the network training period becomes very long, and the same time lead to an industry cost waste.

ANNs require a training or learning process before it can start performing a task. However, most of the training algorithms are suffering of a common issue, which is the slow convergence rate. Nowadays, back-propagation algorithm is one of the common training algorithm applied to the ANN model. Even though it able to provide a

higher accuracy outcome, but it is also suffering the long training period issue due to the slow convergence rate of the algorithm. With a need of larger training dataset to get a higher accuracy results, the BP slow convergence rate issue become more and more urgent to resolve. This resolution are needed to speed up the whole training process and reducing the cost waste of industry.

In this research, aim is to find a better training algorithm which able to speed up the convergence rate, and at the same time maintaining the output accuracy of the network. A proposed algorithm will use the BP algorithm as a basis and modified it to enhance the convergence rate of the BP algorithm.

1.2 The Research Objectives

The aim of the research project is to improve the slow convergence rate of a neural network without sacrifice the accuracy of neural network. Hence, the objectives are split into several supporting points as follows:

1. To propose a training algorithm that may optimize the convergence rate of neural network.
2. To implement the proposed algorithm into a character recognition model using C++ language and evaluate the performance.

1.3 Scope of the Research

The research will investigate on the existing training algorithms and then propose a better or more suitable training algorithm to speed up the back-propagation algorithm convergence rate without sacrificing the accuracy of the neural network. Design and simulate the ANN models with C++ programming will be carried out. Performance evaluation of the proposed algorithm will be done as well.

A character recognition tasks is chosen for this research and is implemented in C++ programming. This ANN model is developed to identify character 0 to 3. Back-propagation algorithm will be used as a reference point to compare with the modified training algorithm in performance evaluation process. Training speed improvement without sacrificing the accuracy of the new neural network is the main focus of this research.

1.4 Thesis Outline

This thesis was organized into five chapters as follows:

Chapter 1 states the research background and also the problem aimed to be solve in this research. Objective and the scope of the research are highlighted here as well.

Chapter 2 discusses the literature review on the existing training algorithms. The advantages and disadvantages of the training algorithms and a comparison among different training algorithms are discussed.

Chapter 3 presents the design methodology and implementation of the research. The details of design flow, data collection and details of algorithm implementation are elaborated in this chapter as well.

Chapter 4 covers the results of the design flow implementation. The results will be analyzed using table or graph and the comparison between training algorithms will be discussed.

Chapter 5 outlines the conclusions and future works. Conclusion for the overall research findings and the results of the proposed training algorithms and future works for this neural network model will be discussed.

CHAPTER 2

LITERATURE REVIEW

2.0 Introduction

Main focus of this research will be the training algorithms of a neural network. In this chapter, different kind of training algorithms are reviewed and discussed. Section 2.1 will brief about the background of training algorithms and how it works in the ANN model. Section 2.2 will discuss about the back-propagation (BP) algorithm. It is the common algorithm used to train an ANN nowadays. The advantages and disadvantages of back-propagation algorithm will be highlighted in this section. Next, Section 2.3 to Section 2.5 will focus on three different algorithms which are developed to speed up the existing BP algorithm. Lastly, Section 2.4 will compare all the algorithms and propose a modified algorithm to enhance the convergence rate of ANN. Then, a short summary will be concluded in the end of this chapter.

2.1 Background of Training Algorithms

Training algorithms is also known as weight-setting method. It is used to determine how to adjust the weight of each neuron node in the network. There are two

major training algorithms can be found, which are supervised learning and unsupervised learning.

Supervised learning incorporates an external teacher. In the supervised training, a set of data sample with the output aim of the function are provided. Network goes through a training process and the weight of each neuron node are adjusted to reduce the error gradient found. These adjusted weights will help the network to move towards to the desired outcome. A set of optimal sample provide to the network during the training process will allow the algorithm to correctly determine the class for unseen instances.

Back-propagation algorithm is one of the typical supervised learning which commonly used to train a feed-forward multilayer networks. Back-propagation will propagate the error information from the output back towards the input nodes. Then, weight of each node will be adjusted to reduce the errors and move the network to the desired direction. The whole training process will walk through a set of optimal sample to train the network to generalize from the training data to any unseen situation.

Another type of training algorithm is unsupervised learning. In this learning, there are set of unlabeled data provide to the network. Since the provided data are unlabeled, there is no error signal to evaluate the potential solution. The concept of unsupervised training is all fired nodes will link together and they are assigned to the same output units. It is self-organized network to find a similar property among the input vectors and linked the similar input vectors to a same output units. This kind of

unsupervised learning can be implemented in automatic target recognition (Chen, 2011) and etc.

For character recognition issues, a supervised training implementation is be chosen. It is due to the output target can be easily provided together with the input training data for the network to learn. Hence, this research will focus on supervised training algorithm.

2.2 Back-propagation algorithm

Back-propagation algorithm, is also known as the generalized delta rule, is proposed by D.E. Rumelhart and J.L. McClelland to train the multilayer neural network. It used to be a simple gradient descent method to compute the networks and minimize the total square error of the output (Rumelhart, et al, 1986). To train a network using back-propagation algorithm, there are three main stages which are the feed-forward of the input training pattern, the back-propagation of the associated error and the adjustment of the weights (Fausett, 1994).

For simplicity, the algorithm can be explained by using a multilayer neural network with only one hidden layer. At first, a set of weight value are initialized to a small random value. This set of initial weight value are used by all the nodes of the neural networks to compute output value, including inputs nodes, hidden nodes and outputs nodes. After initialization, the feed-forward process is carried on. A training

pattern is sent to the input nodes of the neural network with a desired output value given. Each input unit will then broadcasted the signal to the units in the hidden layer. The input signal will be modulated by the weight for that particular node. Each hidden node will then sum its weighted input signal as well as its weighted bias value received. Next, the summed signal will pass through an activation function to compute its output activation value and the computed value will be sent to the units in output layer. The output node processes the signal as the hidden node and produces an output value.

After getting an output value, the back-propagation of the associated error is start executed. The actual output value get from earlier stage has compared to the given desired output value for the particular input pattern and produced as error function. This error function, ε is calculated by subtracted the desired output value to the actual output value. The total mean square error is calculated using Equation 2.1 where E is total means square error, and ε is the error function.

$$E = \frac{1}{2}\varepsilon^2 \tag{2.1}$$

To minimize this total square error, a gradient descent method is proposed. This gradient descent method uses to set the weight change equal to the derivative of total mean square error for that particular weight. All the weight changes in the network are then be calculated.

After weight changes counts, the last stage which is the adjustment of the weights is carried on. A learning rate is used in this stage to determine the weight changes extent in a training time. A small value of learning rate is suggested to avoid oscillating occur and cause the convergence towards the target output failed. The weight changes are then modulated by the learning rate and then add back to the original weights to improve the performance of network. After adjusting the weights, a new set of training pattern is sent to the input nodes and same process are repeated again. Back-propagation training complete after pass through all the input patterns with the weight modifications done. Figure 2.1 demonstrates of the back-propagation algorithm in a diagram form.

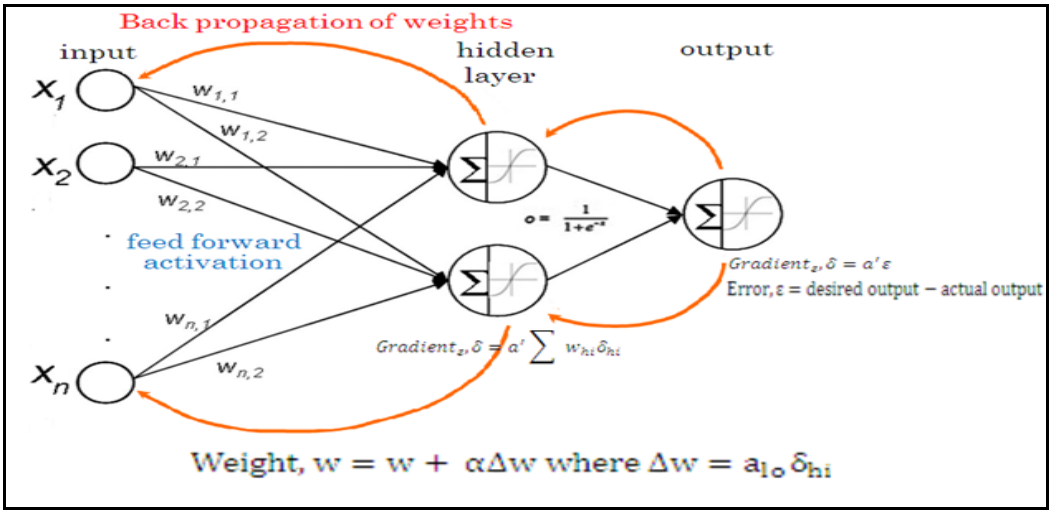

Figure 2.1 Back-propagation algorithm demonstration (Fausett, 1994)

Back-propagation (BP) algorithm is found to be one of the best training method to build up a multilayer neural network and it help to resolve the non-linear

problem which is impossible to be done in single-layer neural network. The gradient descent method proposed in BP algorithm ensures the convergence of the output value are adjusted towards the desired target. This helps to provide high accuracy output to the users to resolve their problem.

However, there are a few number of drawbacks found in the back-propagation as well. Slow convergence rate make the network training time long before the networks can really be used. Besides that, the gradient descent method will converge the output towards a local minimum value and end up the output prediction go wrong and cannot be recovered (Pan, et. al., 2013). Slower convergence rate and longer training times are the main shortcoming when back-propagation algorithm are compared to other competing techniques.

To overcome these shortcomings of back-propagation algorithm, many different training algorithms are proposed to improve the training speed performance in different usage model. Three algorithms are chosen to compare and discuss in the following sections, which are momentum algorithm (Pan, et al, 2013), adaptation learning rate algorithm (Chao, et al, 2009) and automatic learning rate selection algorithm (Guenter, et al., 2013). All of them are suggested by researchers to speed up the neural network and discusses in Section 2.3, 2.4 and 2.5 respectively.

2.3 Momentum algorithm

In the conventional back-propagation algorithm, weight modification only based on the error in the gradient. The initial weight and learning rate chosen are quite small, normally range in 0.01-0.1, to avoid network oscillating. It end up causes the slow convergence rate and very long training time for a neural network. The network are taking plenty of time to slowly adjust the weight value to reach the final minimum point. The conventional algorithm even though provides higher accuracy compared to other network algorithm but it comes with a very time-consuming training process.

For momentum algorithm, it not only depends on the error in the gradient to update the weights in each nodes, it also depends on the variation tendency of the error curve with a tolerance of tiny change in the neural network (Pan, et al, 2013). It help to speed up the training by adding a proportion of the previous weight changes to the current weight changes. Besides, for momentum algorithm only, there are many modified algorithm proposed as well. Different formula for the changes are proposed but they are conceptual similar.

2.3.1 Determination of Momentum algorithm

Two momentum algorithms are discussed in this section which having similar end results for the neural networks which is speed up the training time. These two

algorithms are additive momentum algorithm and adaptive momentum algorithm. The formula on weight changes for each momentum algorithm are discussed as below.

For additive momentum algorithm, the additive value proportional to the value of the previous weights and thresholds change and generates the new change based on the back propagation algorithm. The adjusted weight function, $\Delta\omega$ is represented in Equation 2.2.

$$\Delta\omega(k) = mc \times \Delta\omega(k-1) + lr \times mc \times \frac{d\varepsilon}{d\omega} \tag{2.2}$$

where $\Delta\omega$ = changes of node weight, ω = original node weight, k = training times, mc = momentum factor, lr = learning rate and ε = error function.

In this algorithm, the suggested momentum factor value is 0.95. The learning rate is a constant with small value. With this additive momentum algorithm, $\Delta\omega$ is adjusting in every iteration based on the previous weight changes with guidance of momentum factor. This helps the weight movement in the same direction on successive steps (Pan, et al, 2013).

Another type of momentum algorithm is the adaptive momentum techniques. The proposed algorithm suggests that each weight has its own momentum factor and it is adaptive at every iteration (Yu, et al, 2002). The adjusted weight function is represent in Equation 2.3.

$$\Delta\omega(k) = lr \times a_{lo} \times \delta_{hi} + mc \times \Delta\omega(k-1) \quad (2.3)$$

where $\Delta\omega$ = changes of node weight, ω = original node weight, k = training time, lr = learning rate, a_{lo} = activation output of lower layer, δ_{hi} = error gradient of higher level, and mc = momentum factor.

Equation 2.3 is not only applied on weights change among layer, but also applied to the bias change among each node. Same equation can be applied to both weights and bias. By using the adaptive momentum algorithm, it helps to enlarge the weight movement in the same direction changes, whereas it moves only a small step for the different direction changes. This helps to optimize the training pattern where similar desired output gets faster movement and to shorten the whole training process to get to a convergence point.

2.3.2 Advantages and Disadvantages of the momentum algorithm

The algorithm named as momentum due to the techniques proposed encourages the movement in same direction on successive steps (Nii, 1999). Both momentum algorithms are helping to speed up the movement if the previous step and current step are in the same direction, whereas it will only slight change if two consecutive steps are in the opposite direction. Besides, a lot of research has also found out that momentum algorithm helps to reduce the oscillations during the training time (Pan, et al, 2013). Just by adding the momentum factor to each weight changes, it helps

to smooth out the descent path and also prevents the extreme changes in the gradient due to local anomalies (Yu, et al, 2002). Compared these momentum algorithms with conventional back-propagation algorithm, the momentum algorithms able to speed up the convergence rate and at the same time reducing the oscillation or over-training behavior in the learning process. It helps to optimize the original back-propagation algorithm with a simple adjustment in the formula.

However, these two momentum algorithms have their own disadvantages as well. For additive momentum, the momentum factor suggested to be 0.95, but this parameter value can only be found through experiments. It might taking extra time to find an optimal value for this new adding parameter. Different usage model might have different momentum factor value. The suggested value in the paper might not valid in different aspects.

For adaptive momentum algorithm, ranges of momentum factors are identified that yield stable performance for a given network architecture, but different value of momentum factors work differently. The momentum factor has dependency on the initial learning rate value set. Different initial value will require different momentum factor value to get an optimal network. Hence, to get an optimized momentum factor, some small experiments needs to be carried out before the network's training can be executed.

2.4 Adaptation Learning Rate Algorithm

Conventional back-propagation algorithm has a constant learning rate value. The learning rate experiments are carried out by many researchers and found out only tiny value of learning rate should be used to avoid network oscillation. If learning rate is too big, networks might have large swing value on every iteration move. It will end up taking a lot of the training period to settle down the convergence value and sometimes it will just converge to a wrong local minimum value. In contrast, if the learning rate is smaller, the learning process is more stable but the learning speed will take longer time to converge to a value. Besides, the fix learning rate also need to go through some experiments to obtain an optimize value which balance between the learning rate and the learning stability. It might require a very time-consuming parameter setting process.

Adaptation learning rate (ALR) algorithm is proposed by Yang Chao and Xu Ruzzhi in year 2009 which help to increase the convergence rate but not causing the networks to oscillate and swing away from the ideal path. Besides, adaptation learning rate algorithm has found to be more robust on convergence the value than using a fix learning rate value (Jing, 2012).

2.4.1 Determination of Adaptation learning rate algorithm

The new formula proposed in adaptation learning rate is tried to link the output mean square error in the conventional back-propagation algorithm to the learning rate.

Yang Chao and Xu Ruzhi (Chao, et al., 2009) are also proposed the learning rate should change on each iteration run, and suitable value of learning rate should be used in different iteration period to maximize the learning process.

In this new algorithm, the training process is similar to the back-propagation algorithm. Maintaining the same feed-forward input networks and same activation function, but changing the learning rate throughout the whole learning period. It can be done by replacing the fixed learning rate by an adaptation learning rate. New technique formula are expressed in Equation 2.4.

$$lr = lr_o \exp(h \times E) \tag{2.4}$$

where lr = learning rate for next iteration, lr_0 = initial learning rate, h = amplifying factor and E = mean square error on one iteration run.

By modifying the equation, initial rate value can be kept in small value, and the learning rate will amplify if the mean square error is found to be a large value. If mean square error is small, network will just learn slowly with almost the same as initial learning rate. In facts, mean square error (MSE) has larger value in the beginning and after weight adjusting through many iterations, MSE will become smaller and smaller. With smaller MSE value, it means the networks are approaching its converging value. At this point, learning rate should keep small enough to avoid over-train the networks which might cause the system oscillating and not able to converge to the correct static point. This proposed expression is then found to be able to accelerate the learning speed

at the beginning of training process and perform a small step improvement towards the target at the end of training. It will help to move the output to a convergence point faster and minimize the impact of causing the oscillating networks.

2.4.2 Advantages and Disadvantages of the adaptation learning rate algorithm

There is a significant advantages in this adaptation learning rate algorithm where it able to performs a big step movement in the beginning of the training process towards the correct direction and slightly smaller and smaller steps on the sub-sequential iteration until the end of the training process. This algorithm makes only a tiny movement which equal to the initial learning rate value when the training process closer to the end.

The concept are successfully implemented in the new formula where the learning rate are exponentially proportional to the MSE value. Initially, the calculated MSE value is large, then the learning rate is enlarged by the exponential of MSE value to move a larger step toward the desired output. As the learning proceeds, the MSE error value is getting smaller and smaller and hence the amplified rate of the learning rate is reduced exponentially. Network move slower and slower toward the target, which is a good trend to reduce the over-train issue and avoid swinging from the desired output value. At the end of the training process, the error value is almost close to zero where exponential of 0 is equal to 1, therefore the adaptive learning rate turns back to its initial

rate. This help to descend the error value faster and in the same time maintaining the stable learning progress.

Another advantages of this adaptation algorithm is the implementation is very easy. Same back-propagation method are used by just updating the fix learning value to the adaptive learning value. This can be easily added on to other modifying algorithms to get a win-win solution.

In this proposed formula, there is another important parameter which is *h*, the amplifying factor. From the paper (Chao, et al, 2009), this amplifying factor might need to be handle carefully, else the oscillating system will happen due to the wrong amplifying factor setting. However, the amplifying factor ranges or any suggestion value are not found in the research paper. This will be another new parameter to define through an experiment process.

2.5 Automatic Learning Rate Selection algorithm

Automatic learning rate selection is another algorithm proposed to make the learning rates adaptively through the learning process, but it is not just modifying the existing back-propagation weight change formula as Section 2.4. The main idea of this algorithm is similar to Section 2.4 where trying to create an adaptive learning rate throughout the learning process, however the concept and implementation of these two algorithm are totally different.

This automatic learning rate selection (ALRS) algorithm is proposed to use the concept of stochastic gradient descent (SGD) algorithm rather than the typical gradient descent algorithm in back-propagation (Guenter, et al., 2013). This automatic learning rate selection is a type of an empirical search algorithm. For typical empirical search algorithm, a full range search will require to process the training data with different learning rate at least 10 times to then getting the optimize value to use in each iteration run (Schaul, et. al., 2012). However, with this typical way, it incurs an additional computation cost. In year 2013, Brain Guenter is then proposed a novel method which is similar to the typical empirical search but simplify it to save the computation cost and timing. This algorithm are suitable for non-stationary problems.

2.5.1 Determination Automatic learning rate selection algorithm

The algorithm for this approach are unlike other empirical methods, it determines the learning rate before every iteration run or epoch by using a small sample of the training set. This might requires extra additional computation to get the learning rate value, but the value get will be more precise to the change of the networks.

In this method, small set of training data (around 5% of the full training data) are selected for the determination of the learning rate value. During the epoch, the model parameters are randomly initialized, then the algorithm will search for the best learning rate which should use for next epoch based on the small set of random samples. For subsequent epochs, same small set training go through again, algorithms will search

for a biggest learning rate which can see the improvement in the average training criterion. This calculation of learning rate using a small set random training data will run on 2-5% of the whole epoch size to speed up the training rate at the beginning process. The formula used for this algorithm are expressed in C++ programming code shown in Figure 2.2.

Algorithm 2 Learning Rate Search

function λ=SEARCHLEARNINGRATE(N, n, e_{t-1})
Input: N=total number of samples in an epoch; n=number of samples used to estimate learning rate; e_{t-1}=training criterion from previous epoch
Output: λ=optimal learning rate
if $t \leq 0$ **then**
λ=Grid search the best λ to minimize e^s_λ
else
Compute e^s_0 with learning rate 0 by running SGD over n samples
$r = sqrt(n/N)$
$e_{base} = (1 - r) \times e^s_0 + r \times e_{t-1}$
λ=Grid search for the largest λ so that $e^s_\lambda \leq e_{base}$
end ifreturn λ
end function

Figure 2.2 Pseudo code for the Learning Rate Search algorithm (Guenter, et al., 2013)

2.5.2 Advantages and Disadvantages of the automatic learning rate algorithm

The advantages of the algorithm is the learning rate get computed using small set of training data rather than full set of training data to optimize the best learning rate. In conventional BP algorithm, an initial learning rate found through the trial-and-error method can also optimize to small set of training data rather than using the full set of training data to optimize the parameter. Besides, this pre-set of learning rate value with

consideration of previous error value will help to set better learning rate through the training process. 2-5% of the full epoch size will be useful to converge the network to the desired output faster, after the 5% epoch time, the initial learning rate will be used to slowly move the target closer and closer to the desired output (Guenter, et al, 2013). It is the same concepts as adaptation learning rate algorithm which accelerate the speed of training in the beginning, and then slightly move towards the desired output at the end. However, adaptation learning rate does not go through the evaluation process before every epoch run to recalculate the optimal learning rate. Automatic learning rate selection can help to correct the over-train or wrong direction move through the pre-set learning rate process. This is the advantage that does not found in momentum algorithm and also adaptation learning rate algorithm.

However, the best learning rate chosen will very subjective to the random set data choose as a pre-setting process. This might cause a risk where the networks converge to only local minimal point, and lead to an inferior final model. A good training set are selected for learning rate setting data will bring to a good trends of networks whereas bad training set might lead to a bad model (Nogueira, et. al., 2016). Besides, Figure 2.2 has provided the implementation of the algorithm in the published paper, but the important key point of how the grid search calculation should be done is not stated clearly in the paper, only concept of the algorithm are provided. This will require a conceptual understanding to implement the algorithm in C++ programming.

2.6 Summary

Back-propagation and three other advance algorithms which improving the BP convergence rate are discussed. All three algorithms are found to resolve the shortcoming of the conventional BP algorithm which is convergence speed issue. Different method has slightly different concepts and different pros and cons are found and discussed in Section 2.3 to Section 2.5.

In summary, momentum algorithm can help to speed up the network when there are the same direction movement but only small step movement for opposite direction. Adaptation algorithm able to accelerate the learning speed at the beginning and then slow down the moving step when closer and closer to the desired target. Similar idea found at automatic learning rate selection algorithm where finding the best initial learning rate at the beginning 5% epoch run, and once network start converging, a fix learning rate has applied to the system. All these three algorithm are able to speed up the convergence rate and at the same time avoid the oscillation which might cause the whole network broken.

Theoretically, both momentum factor and learning rate value can be used on the BP algorithms to make it achieve a faster convergence rate. This two algorithms are not offending each other and they can survive together to improve BP algorithms. The conventional BP algorithm can be updated with momentum factor and adjustable learning rate added, since the algorithms reviewed are all build on top of BP basis.

Hence, a mixture of these two type of algorithms are proposed and executed in this research. Mixture of momentum algorithm with two different adjustable learning rate algorithm are proposed which are momentum plus ALR algorithm and momentum plus ALRS algorithm. The methodology used in this research and the way to carry out these proposed algorithms are discussed in Chapter 3.