P- ISSN 1991-8941 E-ISSN 2706-6703 Journal of University of Anbar for Pure Science (JUAPS) Open Access
2008,(2), (1) :94-105

ADAPTIVE LEARNING RATE VERSUS RESILIENT ACKPROPAGATION FOR NUMERAL RECOGNITION

Muntaser Abdul-Wahed Salman

University of Alanbar- College of Computer

A R T I C L E I N F O

Received: 1 / 4 /2008
Accepted: 24 / 4 /2008
Available online: 30/4/2008
DOI: 10.37652/juaps.2008.15445

Keywords:
ADAPTIVE ,
LEARNING RATE ,
BACKPROPAGATION ,
NUMERAL RECOGNITION.

A B S T R A C T

Two types of neural networks learning algorithms were created, trained, tested, and evaluated in an effort to find the appropriate neural network training method for use in numeral recognition problem. The purpose of this study was to compare the training speeds of two neural networks Backpropagation learning algorithms (Adaptive learning rate and Resilient) when exposed to ten number recognition data sets. Each algorithm was trained using ten data sets as a basic set (Boolean value), and a complex (noisy) set. The trials conducted indicated a significant difference between the two algorithms in the basic data set, with the Resilient training algorithm the neural network trained faster.The creation, training, and testing of each neural network was done using the MathWorks software package MATLAB which contains a "Neural Network Toolbox" that facilitates rapid creation, training, and testing of neural networks. MATLAB was chosen to use for learning algorithm development because this toolbox would save an enormous amount programming effort.

Introduction

It is often useful to have a machine perform pattern recognition. In particular, machines that can read symbols are very cost effective. A machine that reads banking checks can process many more checks than a human being in the same time. This kind of application saves time and money, and eliminates the requirement that a human perform such a repetitive task. Pattern recognition in neural networks is a very broad field, but a common use for neural networks is handwriting or numeral recognition. This pattern matching technique enables computers to identify and utilize human handwriting for numbers as well as characters [1, 3, 4].

Recognition of handwritten numerals is important because of its applicability to a number of problems, like postal code recognition and information extraction from fields of different forms. In the Indian context, there exists a need for development and/or evaluation of the existing techniques for recognition of numerals written in Indian scripts. Generic techniques cannot, in general, tackle problems associated with script specific peculiarities. In this paper, we present a neural network–based architecture for recognition of handwritten numerals. Although the architecture is generic, it has been found to be useful for recognition of handwritten numerals.

* Corresponding author at: University of Alanbar- College of Computer, Iraq.E-mail address: **muntaserabd1@yahoo.com**

An artificial neural network (ANN or NN for short) is an artificial intelligence closely modeled after a human brain. Such a neural network is composed of computer-programming objects called nodes [6]. These nodes closely correspond in both form and function to their organic counterparts, neurons. Individually, nodes are programmed to perform a simple mathematical function, or to process a small portion of data. A node has other components, called weights, which are an integral part of the neural network. Weights are variables applied to the data that each node outputs. By adjusting a weight on a node, the data output is changed, and the behavior of the neural network can be altered and controlled. By careful adjustment of weights, the network can learn. Networks learn their initial behavior by being exposed to training data. The network processes the data, and a controlling algorithm adjusts each weight to arrive at the correct or final answer(s) to the data. These algorithms or procedures are called learning algorithms.

Neural networks are often used for pattern recognition and classification. Their adaptability and learning capabilities make them excellent choices for tasks requiring comparison of data sets or extracting subtle patterns from complex data [7, 11]. The field of neural networks has a history of some five decades but has found solid application only in the past fifteen years [1, 6], and the field is still developing rapidly. Thus, it is distinctly different from the fields of control systems or optimization where the terminology, basic mathematics, and design procedures have been firmly established and applied for many years [6, 4]. This project was focused on numeral recognition in its most basic form, individual number recognition. The rationale for this project was to improve efficiency neural network numeral recognition. The study conducted a series of tests to determine which of two learning algorithms, Adaptive learning rate Backpropagation [2] or Resilient Backpropagation [10], trained a neural network faster. Ten sets of number where used to compare the algorithms, a basic Boolean value set, and noisy (which may be a handwritten) number set.

Learning Algorithms

Backpropagation was created by generalizing the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions[6]. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by you. Standard backpropagation is a gradient descent algorithm[1] in which the network weights are moved along the negative of the gradient of the performance function as shown in the following equation.

$$W(t+1) = W(t) - \alpha * \partial E/\partial W(t) \text{ - - - } (1)$$

Where α is the constant learning rate, and $\partial E/\partial W(t)$ is the derivative (slope) of the error E at time t (in epochs).

There are many variations on the Backpropagation, gradient descent model. Two of these are the Adaptive learning rate Backpropagation algorithm [2], referred to as gda, and the Resilient Backpropagation [10], known as Rprop.

Adaptive learning rate Backpropagation algorithm:-

With standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate [9]. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface.

The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process [9]. An adaptive learning rate will attempt to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface. An adaptive learning rate requires some changes in the training procedure used by traingda (function used in Matlab) [5].

First, the initial network output and error are calculated. At each epoch new weights and biases are calculated using the current learning rate. New outputs and errors are then calculated. If the new error exceeds the old error by more than a predefined ratio max_perf_inc, the new weights and biases are discarded. In addition, the learning rate is decreased (typically by multiplying by lr_dec). Otherwise, the new weights, etc., are kept. If the new error is less than the old error, the learning rate is increased (typically by multiplying by lr_inc). This can be shown in the following equation:

$$\alpha(t+1)=\begin{cases}\alpha(t)*lr\_dec & if\ E(t)>E(t-1)\\ \alpha(t)*lr\_inc & if\ E(t)<E(t-1)\\ \alpha(t) & if\ E(t)=E(t-1)\end{cases} \quad \text{-- (2)}$$

Resilient Backpropagation (Rprop) training algorithm:-

The purpose of the resilient backpropagation (Rprop) training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives[8,9,10]. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. The size of the weight change is determined by a separate update value. The update value for each weight and bias is increased by a factor delt_inc whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations. The update value is decreased by a factor delt_dec whenever the derivative with respect that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same. Whenever the weights are oscillating the weight change will be reduced. If the weight continues to change in the same direction for several iterations, then the magnitude of the weight change will be increased. A complete description of the Rprop algorithm is given in [10]. This can be shown in the following equation:

$$\Delta W(t+1)=\begin{cases}-\mu & if\ \partial E(t)/\partial W(t)>0\\ +\mu & if\ \partial E(t)/\partial W(t)<0\\ 0 & else\end{cases} \quad \text{---(3)}$$

μ is the weight step, calculated by multiplying the derivative of the current slope and the previous slope as described above.

3. Numeral Recognition Procedure

A network is to be designed and trained to recognize the 10 numbers (from 0 to 9). An imaging system that digitizes each number centered in the

system's field of vision is available. The result is that each number is represented as a 5 by 7 grid of Boolean values. White has an input value of 0, black a value of 1. Each number is a 5x7 matrix. As shown in fig.(1)
However, the imaging system is not perfect and the numbers may suffer from noise. Perfect classification of ideal input vectors is required and reasonably accurate classification of noisy vectors.

The ten 35-element input vectors are defined in the function file recog as a matrix of input vectors called number. The target vectors are also defined in this file with variable called targets. Each target vector is a 10-element vector with a 1 in the position of the number it represents, and 0's everywhere else. For example, the number 0 is to be represented by a 1 in the first element (as 0 is the first number of the numbers), and 0's in elements two through ten.

Initialization

In our problem the neural network needs 35 inputs and 10 neurons in its output layer to identify the numbers. The network is a two-layer log-sigmoid/log-sigmoid network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output Boolean values. The hidden (first) layer has 10 neurons. This number was picked by guesswork and experience. If the network has trouble learning, then neurons can be added to this layer.

The network receives the 35 Boolean values as a 35-element input vector. It is then required to identify the number by responding with a 10-element output vector. The 10 elements of the output vector each represent a number. To operate correctly, the network should respond with a 1 in the position of the number being presented to the network. All other values in the output vector should be 0. In addition, the network should be able to handle noise. In practice, the network does not receive a perfect Boolean vector as input. Specifically, the network should make as few mistakes as possible when classifying vectors with noise of mean 0 and standard deviation of 0.2 or less. The two-layer network is created with newff and shown in Fig. (2).

```
S1 = 10;
[R,Q] = size(number);
[S2,Q] = size(targets);
P = number;
net=newff(minmax(P),[S1 S2],{'logsig'
'logsig'},'traingda');
```

Training

To create a network that can handle noisy input vectors it is best to train the network on both ideal and noisy vectors. To do this, the following conditions should be considered:-

The network is first trained on ideal vectors until it has a low sum-squared error.

Then, the network is trained on 10 sets of ideal and noisy vectors. The network is trained on two copies of the noise-free number at the same time as it is trained on noisy vectors. The two copies of the noise-free number are used to maintain the network's ability to classify ideal input vectors.

Unfortunately, after the training described above the network may have learned to classify some difficult noisy vectors at the expense of properly classifying a noise-free vector. Therefore, the network is again trained on just ideal vectors.This ensures that the network responds perfectly when presented with an ideal number.

Training is done using two neural networks Backpropagation learning algorithms, adaptive learning rate with the function traingda and Resilient Backpropagation with the function trainrp respectively.

A study was conducted to compare the convergence or learning speed of two different weight adjustment algorithms (gda and Rprop) in feed-forward neural networks. Ten different training data sets where created, one basic set made up of binary number bitmaps, and a second complex set made up of human handwriting grayscale bitmaps. The basic set was used to train a network using gda, and a network using Rprop. This was repeated 10 times for each algorithm, to eliminate the possible confounding variable of random weight setting. The results from these 20 trials were compared to each other. The complex set was also used to train a gda network, as well as a Rprop network. Again, to eliminate harmful initial weight settings, each simulation was run 10 times. The complex training set results were compared to each other. In total, 40 trials were run (10 for each training set and algorithm combination).

The first training set was made up of ten 5x7 pixel standard Binary number bitmaps (see fig.1). The second training was made up of four set of 5x7pixel bitmaps, two of standard binary bitmaps as well as two of noisy sets (see fig.3 a & b). The networks were simulated using Matlab.

The basic networks (5x7 number set) had 35 input nodes (each corresponding to a bit in the 5x7 matrix), 10 nodes in the one hidden layer, and ten output nodes (parallel to the number of patterns in the set). The exact parameters of both algorithms used are considered to be standard as shown in Table 1.

There are six training parameters associated with traingda and trainrp algorithms: (epochs, show, goal, time, min_grad and max_fail)

The training status is displayed for every show iterations of the algorithm. (If show is set to NaN, then the training status is never displayed.) The other parameters determine when the training stops. The training stops if the number of iterations exceeds epochs, if the performance function drops below goal, if the magnitude of the gradient is less than min_grad, or if the training time is longer than time seconds. max_fail, is associated with the early stopping technique.

In traingda algorithm the learning rate lr is multiplied times the negative of the gradient to determine the changes to the weights and biases. This change has been described in section 2.2. If the new error exceeds the old error by more than a predefined ratio, max_perf_inc, the new weights and biases are discarded. In addition, the learning rate is decreased by multiplying by lr_dec. Otherwise, the new weights, etc., are kept. If the new error is less than the old error, the learning rate is increased by multiplying by lr_inc.

In trainrp algorithm the size of the weight change is determined by a separate update value. The update value for each weight and bias is increased by a factor delt_inc whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations.

The update value is decreased by a factor delt_dec whenever the derivative with respect to that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same. delta0 and deltamax are the initial step size and the maximum step size, respectively. The performance of Rprop is not very sensitive to the settings of the training parameters [4].

Training without Noise

The network is initially trained without noise for a maximum of 1000 epochs or until the network sum-squared error falls beneath 0.001.

```
P = number;
T = targets;
net.performFcn = 'sse';
net.trainParam.goal = 0.001;
net.trainParam.show = 50;
net.trainParam.epochs = 1000;
[net,tr] = train(net,P,T);
```

Training with Noise

To obtain a network not sensitive to noise, we trained with two ideal copies and two noisy copies of the vectors in number. The target vectors consist of four copies of the vectors in target. The noisy vectors have noise of mean 0.1 and 0.2 added to them as shown in fig.(3 a & b) below. This forces the neuron to learn how to properly identify noisy numbers, while requiring that it can still respond well to ideal vectors.

To train with noise, the maximum number of epochs is reduced to 250 and the error goal is increased to 0.1, reflecting that higher error is expected because more vectors (including some with noise), are being presented.

```
netn = net;
netn.trainParam.goal = 0.1;
netn.trainParam.epochs = 250;
T = [targets targets targets targets];
for pass = 1:10
P = [number, number, ...
(number + randn(R,Q)*0.1), ...
(number + randn(R,Q)*0.2)];
[netn,tr] = train(netn,P,T);
end
```

Once the network is trained with noise, it makes sense to train it without noise once more to ensure that ideal input vectors are always classified correctly. Therefore, the network is again trained with code identical to the previous section.

Results

The purpose of this study was to compare the training speeds of two neural network learning algorithms (gda and Rprop), when exposed to ten numeral recognition data sets. Based on previous studies, it was hypothesized in this project that if the learning algorithm used is resilient (Rprop), then it will have a quicker convergence time (fewer training cycles, or epochs) than gda, when both exposed to the same numeral recognition training data.

Data collection for this study was done by collecting a log file of all outputs from the neural network simulator (Matlab NN Toolbox). There were ten trials of each algorithm (Rprop and gda) for each data set (basic and noisy). The algorithms were compared against each other for speed of training. The log files contained measurements of the Sum Squared Error (SSE) versus the number of training epochs. These total training time values were considered the dependent variable in this study. These values where assembled into two tables, one for the basic data set in the appendix Table (A.1), and one for the complex data set Table (A.2).

For the basic data set, there was a significant difference between the two algorithms, indicating that Rprop trained faster than gda as clear from Table (A.1). This difference is also well indicated by the means of the basic data set training times, with Rprop's mean being

66.3 epochs, and gda's being 473.8 epochs, a difference of approximately seven times. A second test was conducted on the noisy data set, to compare the algorithms in a different situation. Also Table (A.2) indicating that there was significant difference between the training times of the two algorithms. This was also indicated fairly well by the means of the two training times, with Rprop's being 47.2 epochs, and gda's at 114, a difference of approximately two and a half times.

The Rprop algorithm trained faster than the gda algorithm for the basic data set as well as the complex data. This would indicate that for true number recognition problem Rprop is a better choice for training its neural networks.

There were no problems encountered in this study. Possible confounding variables were the small size of the training set and the small number of networks trained (only ten per algorithm, per data set). Further exploration into this topic is certainly warranted.

System Performance

The reliability of the neural network pattern recognition system is measured by testing the network with hundreds of input vectors with varying quantities of noise. This paper tests the network at various noise levels, and then graphs the percentage of network errors versus noise (see Appendix B). Noise with a mean of 0 and a standard deviation from 0 to 0.5 is added to input vectors. At each noise level, 100 presentations of different noisy versions of each number are made and the network's output is calculated. The output is then passed through the competitive transfer function so that only one of the 10 outputs (representing the perfect number of the ten numbers), has a value of 1. The number of erroneous classifications is then added and percentages are obtained as shown in the Appendix C graph.

The solid line on the graph shows the reliability for the network trained with and without noise. The reliability of the same network when it had only been trained without noise is shown with a dashed line. Thus, training the network on noisy input vectors greatly reduces its errors when it has to classify noisy vectors. The network trained with gda Algorithm did not make any errors for vectors with noise of mean from 0.0 to 0.2. When noise of mean 0.25 was added to the vectors both networks began making errors. While the network trained with Rprop Algorithm did not make any errors for vectors with noise of mean from 0.0 to 0.05. When noise of mean 0.1 was added to the vectors both networks began making errors. This means that Rprop Algorithm has more error in training noisy inputs.

If a higher accuracy is needed, the network can be trained for a longer time or retrained with more neurons in its hidden layer. Also, the resolution of the input vectors can be increased to a 10-by-14 grid. Finally, the network could be trained on input vectors with greater amounts of noise if greater reliability were needed for higher levels of noise.To test the system, a number with noise can be created and presented to the network.

```
noisy9 = number(:,10)+randn(35,1) * 0.2;
plotchar(noisy9);
A2 = sim(net,noisy9);A2 = compet(A2);
answer = find(compet(A2) = = 1);
plotchar(number(:,answer));
```

Here is the noisy number and the number the network picked (correctly) as shown in Fig.(4).

The network is trained to output a 1 in the correct position of the output vector and to fill the rest of the output vector with 0's. However, noisy input vectors may result in the network not creating perfect 1's and 0's. After the network is trained the output is passed through the competitive transfer function compet. This makes sure that the output corresponding to the number most like the noisy input vector takes on a value of 1, and all others have a value of 0. The result of this post-processing is the output that is actually used.

Conclusions

Comparison of the results of the study indicated that:-

The original research hypothesis that Rprop would perform faster in all cases was proven correct. However, since Rprop performed much better than gda during the basic trials, it can be inferred that the hypothesis is supported for our problems.

To eliminate the possible confounding variables in this study, the number of trials could be increased and the size of the data sets also enlarged.

Other back-propagation based algorithms could be comparatively tested fairly easily, utilizing the same data sets and similar network structures.

This problem demonstrates how a simple pattern recognition system can be designed. Note that the training process did not consist of a single call to a training function. Instead, the network was trained several times on various input vectors. In this case, training a network on different sets of noisy vectors forced the network to learn how to deal with noise, a common problem in the real world.

References:

1. Ben, K. and Patrick S. (1996). An Introduction to Neural Networks. Eighth Edition. November 1996.
2. Fahlman, S. (1988). An Empirical Study of Learning Speed in Back-Propagation Networks. Carnegie Mellon: CMU-CS-88-162
3. Firas, H. (2000). Handwritten Numeral Recognition Using Neural Networks. EE368, Stanford University. 27 May, 2000. *Available at http://scien.stanford.edu/class/ee368/projects2000/project/node1.html.*
4. Goss, N. J. (2000). Resilient Backpropagation versus Quickprop for Character Recognition in Neural Networks.
5. Howard, D. and Mark, B. (2002). Neural Network Toolbox for use with Matlab. 'User's Guide Version 4'. July 2002.
6. Jacek, M. Zurada (1996). Introduction to Artificial Neural Systems.
7. Klimis, S. (2000). Hand Gesture Recognition Using Neural Networks
8. Riedmiller, M. (1994). Rprop - Description and Implementation Details Technical Report. University of Karlsruhe: *W-76128 Karlsruhe*.
9. Riedmiller, M. (1994). Advanced Supervised Learning in Multi-layer Perceptrons From Backpropagation to Adaptive Learning Algorithms. University of Karlsruhe: *W-76128 Karlsruhe*.
10. Riedmiller, M. and H. Braun 1993. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, 1993.
11. Sang-W. M. and Seong-G. K. (2002). Pattern Recognition with Block-based Neural Networks. *0-7803-7278-6/2002 ©2002 IEEE*.

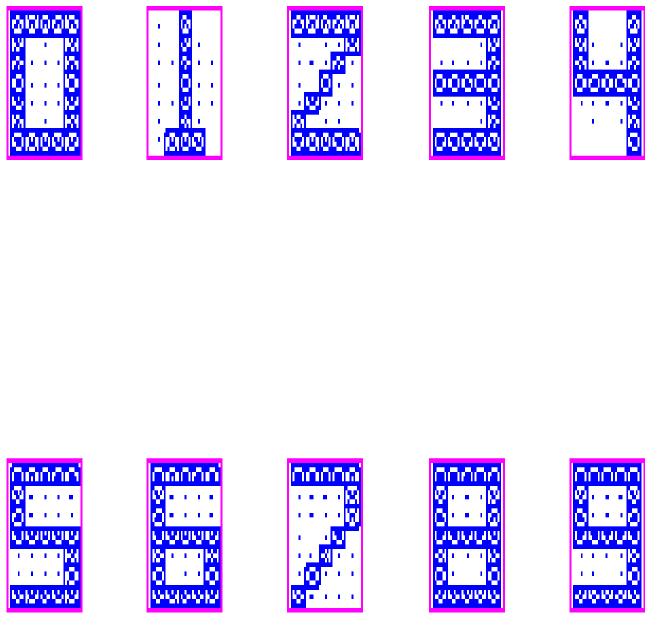
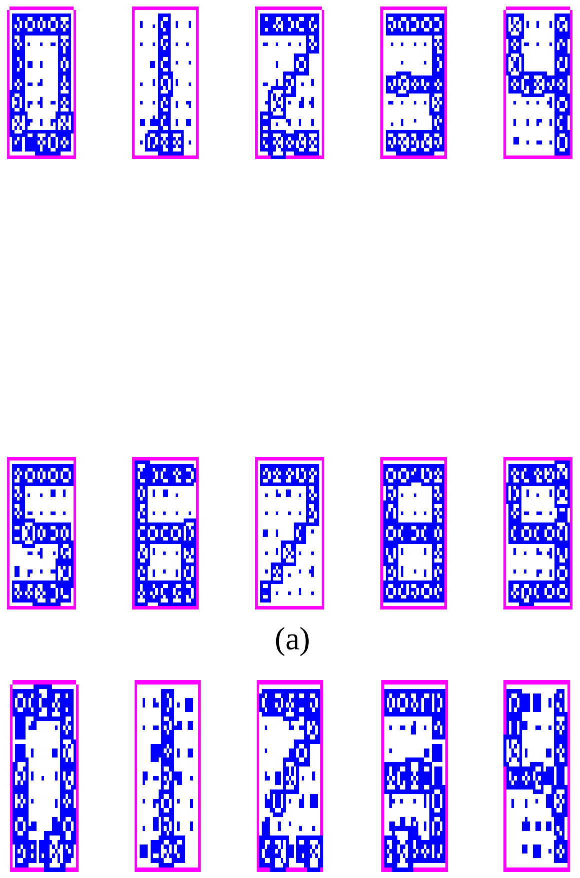


Fig. (1) Graphics of Basic Inputs Numeral


(a)

(b)

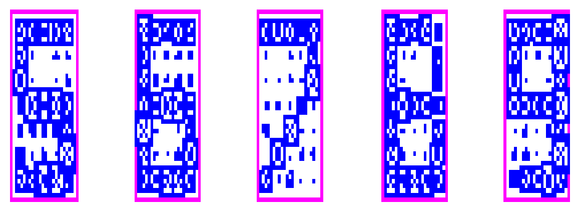


Fig.(3) Graphics of Noisy Data Numeral
(a) with 0.1 noise added (b) with 0.2 noise added

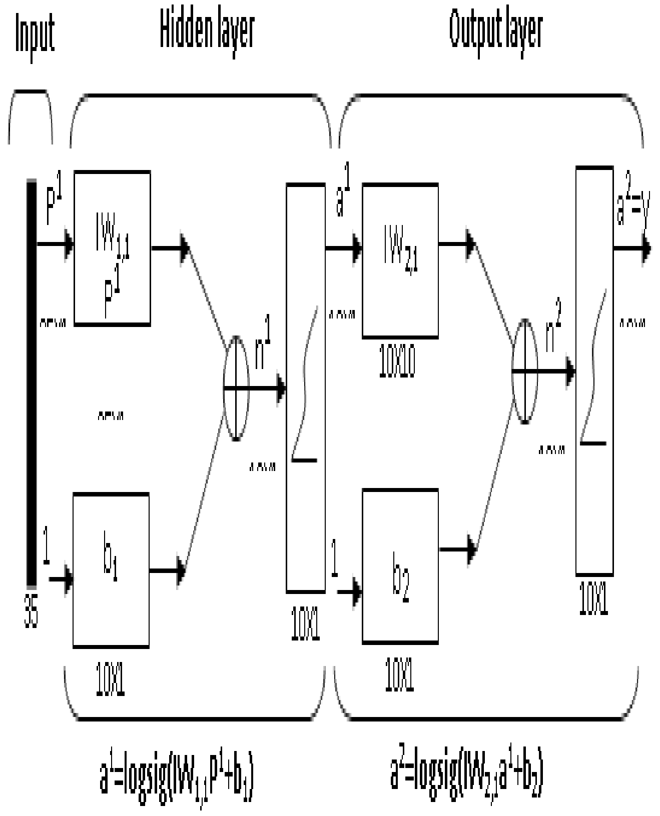
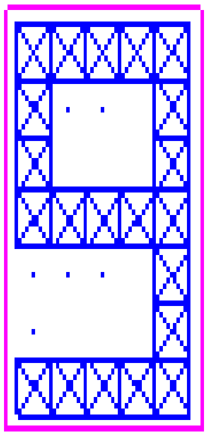


Fig.(2) Neural Network Structure for Our Problem


(a)

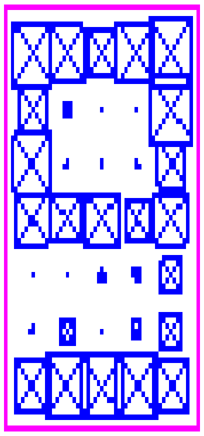

(b)

Fig.(4) Tested Number
(a) Correct number (b) Noisy number

Complex Data Set Total Training Time (epochs)	
Gda	**Rprop**
118	24
139	59
133	15
135	59
120	15
127	81
123	68
108	18
129	87
126	45
Mean=	
125.9	47.2

Table 1 – Algorithm Parameters

Parameters of Gda Algorithm	Parameters of Rprop Algorithm
epochs: 1000 show: 50 goal: 0.001 lr: 0.0100 lr_inc: 1.0500 lr_dec: 0.7000 max_fail: 5 max_perf_inc: 1.0400 min_grad: 1.0000e-006 time: Inf	epochs: 1000 show: 50 goal: 0.001 delta0: 0.0700 delt_inc: 1.2000 delt_dec: 0.5000 max_fail: 5 deltamax: 50 min_grad: 1.0000e-006 Time: Inf

Appendix A: Training Data Tables

Basic Data Set Total Training Time (epochs)	
Gda	**Rprop**
516	62
479	66
365	62
426	66
468	62
433	67
469	58
486	73
551	66
545	81
Mean=	
473.8	66.3

Appendix B: Error Performance Graph

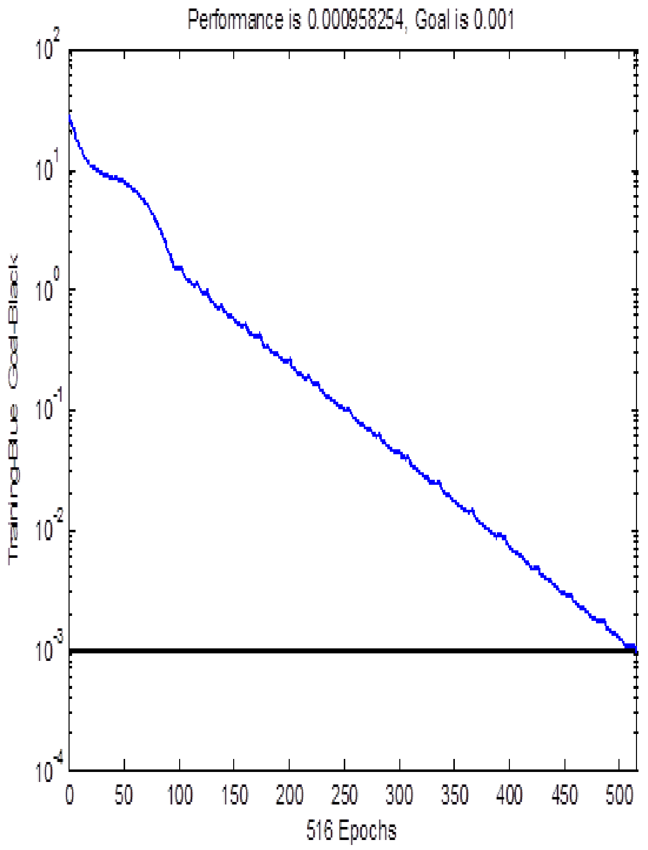


Fig. (B.1) Error Performance of gda training Algorithm

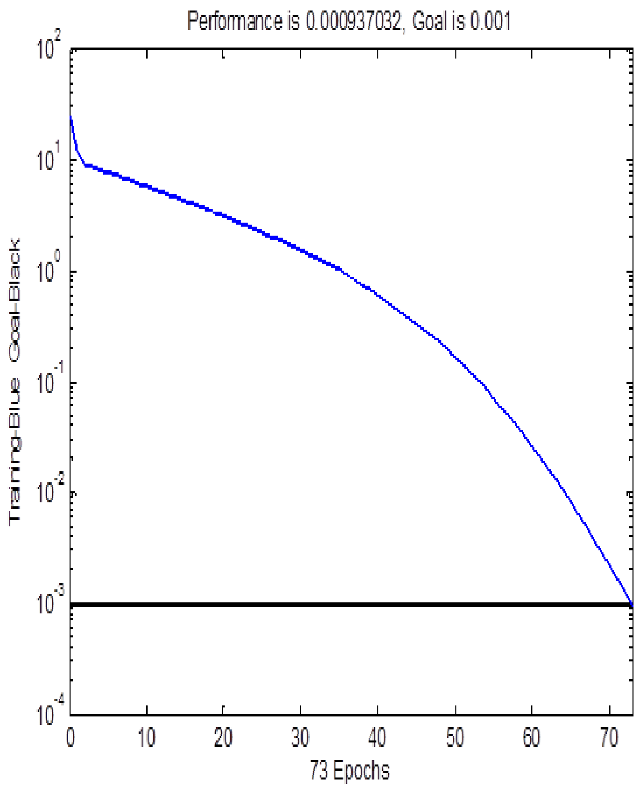


Fig. (B.2) Error Performance of Rprop training Algorithm

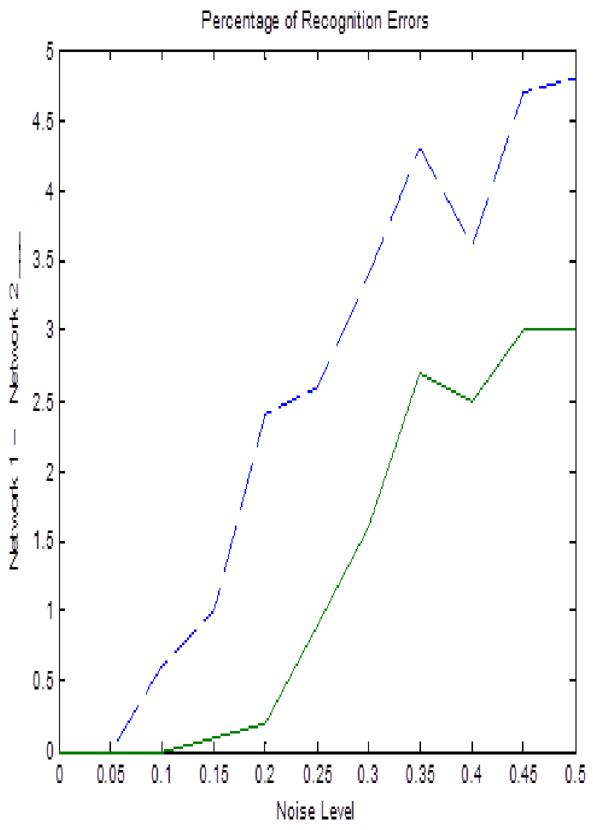


Fig. (C.2) Percentage Error for Rprop training Algorithm

Appendix C: Percentage Error Graph

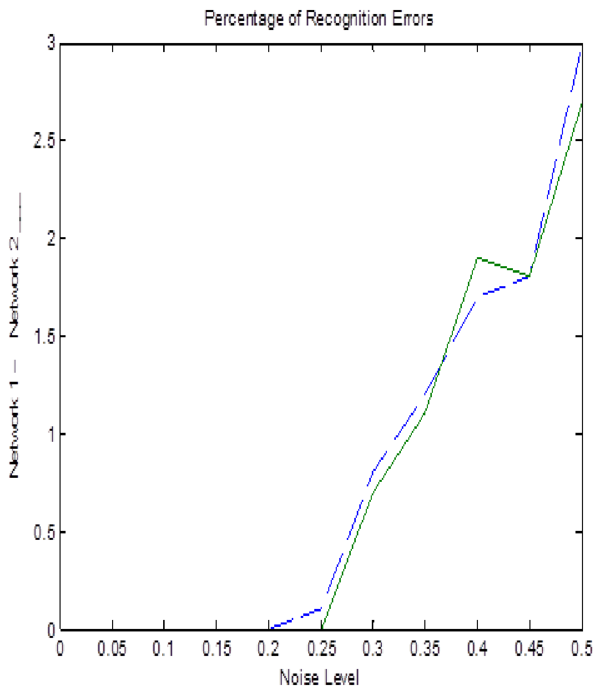


Fig. (C.1) Percentage Error for gda training Algorithm

P- ISSN 1991-8941 E-ISSN 2706-6703 Journal of University of Anbar for Pure Science (JUAPS) Open Access
2008,(2), (1) :94-105

معدل سرعة تدريب متكيف بالمقارنة مع الارتداد العكسي المرن للشبكة العصبية لتمييز الأعداد

منتصر عبدالواحد سلمان

Email: muntaserabd1@yahoo.com

الخلاصة:

نوعين من طرق تدريب الشبكات العصبية تم استخدامهم، تدريبهم، فحصهم وتقييمهم في محاولة لايجاد طريقة تدريب شبكة عصبية مناسبة لمشكلة تمييز الأرقام العشرية. الغرض من هذا البحث مقارنة سرعة تدريب خوارزميات الشبكات العصبية ذات الارتداد العكسي التي تستخدم معدل سرعة تدريب متكيف Adaptive learning rate مع تلك التي تستخدم معدل سرعة تدريب مرن Resilient عند تدريب شبكة عصبية لتمييز الأعداد العشرة numeral recognition للأرقام العربية. كل خوارزمية تم تدريبها باستخدام عشرة مجاميع من الأرقام العشرية كمجموعة أساسية (تمثيل ثنائي) وكذلك مجموعة معقدة (مشوشة). الدراسة المشار إليها أثبتت وجود فرق واضح بين الطريقتين بالمجموعة الأساسية فضلا عن المشوشة حيث إن تدريب الشبكات العصبية لتمييز الأعداد باستخدام معدل سرعة تدريب مرن Resilient أسرع لهذه المشكلة (تمييز الأعداد).

إنشاء ، تدريب وفحص خوارزميات تدريب الشبكات العصبية تم باستخدام مجموعه برامج MathWorks الماتلاب MATLAB حيث يحتوي على صندوق أدوات الشبكات العصبية الذي سهل من عملية إنشاء وتدريب وفحص الشبكات العصبية واختصار بوقت وكلفة برمجة طرق التدريب لهذه الشبكات العصبية.