A Performance Comparison of Different Back Propagation Neural Networks Methods in Computer Network Intrusion Detection

Vu N.P. Dao [1]
dao1@llnl.gov

Rao Vemuri [1, 2]
rvemuri@ucdavis.edu

[1] Lawrence Livermore National Laboratory, 7000 East Ave., Livermore, CA 94551
[2] University of California, Davis, One Shields Ave., Davis, CA 95616

Abstract

This paper demonstrated that neural network (NN) techniques can be used in detecting intruders logging onto a computer network when computer users are profiled accurately. Next, the paper compares the performance of the five neural network methods in intrusion detection.

The NN techniques used are the gradient descent back propagation (BP), the gradient descent BP with momentum, the variable learning-rate gradient descent BP, the conjugate gradient BP, and the quasi-Newton method.

I. Introduction

In recent times, artificial intelligent (AI) techniques in general and neural network (NN) in particular have been used widely in solving problem like optimization, adaptive filtering, digital signal processing, and pattern recognition and classification. Indeed the discipline of neural networks is one of the earliest areas of AI that was introduced in the 1950s and now has gained momentum due to the many successes. Neural network techniques aim to construct useful "computers" to carry out useful computations while solving real-world problems of classification. It is this philosophy that has attracted much interest.

Information assurance is a field that deals with protecting information on computers or computer networks from being compromised. Intrusion detection is a major part of information assurance that deals with detecting unauthorized users from accessing the information on those computers. Current intrusion detection techniques can not detect new and novel attacks; instead the consensus solution seems to be those that detect known intrusion detection techniques. New intruding techniques are usually passed as authorized traffic.

The relevance of NN in intrusion detection becomes apparent when one views the intrusion detection problem as a pattern classification problem. By building profiles of authorized computer users [1], one can train the NN to classify the incoming computer traffic into authorized traffic or not authorized traffic (i.e. intrusion traffic).

This paper is organized as follows. Section II defines the intrusion detection problem. Next, a brief introduction to relevant aspects of neural networks is presented in section III. Section IV delves into the gradient descent BP method and discusses the gradient descent BP with momentum, the variable learning rate gradient descent with momentum, the conjugate gradient descent, and the quasi Newton methods. Section V discusses the test data used. Section VI presents the results of the

five back propagation methods in detecting intruders. Section VII summarizes the performance of the five BP methods. Section VIII offers a conclusion and points to future research.

II. Problem Definition

The task of intrusion detection is to construct a model that captures a set of user attributes and determine if that user's set of attributes belongs to the authorized user or those of the intruder. The input attribute set consists of the unique characteristics of the user [1] logging onto a computer network. The output set is of two types – authorized user and intruder.

Thus, the problem now has been reduced to a pattern recognition problem. Mathematically speaking, the problem can be stated as:

Given a data set S

$$S = \left\{ \left(\mathbf{x}_i, y_i \right) : \mathbf{x}_i \in R^p, y_i \in R, i = 1,..,m \right\}$$

Where:

$\boldsymbol{x}$ = input vector consisting a user's attributes

y = {authorized user, intruder}

We want to map the input set $\boldsymbol{x}$ to an output class y.

The solution to the above problem is to find a mapping function from the p-dimensional input space to the 1-dimensional output space [2].

From a modeling perspective, we seek a model that provides the best fit to training data and the best prediction on future data while minimizing model complexity. The resulting model would be employed to predict output values y' for future observed inputs $\boldsymbol{x}'$ where only the inputs $\boldsymbol{x}'$ would be available.

To accomplish the above objective, we need to come up with a model and train it prior to using that model for intrusion detection. During the training phase, for each input data vector $\boldsymbol{x}$, we already know the associated output y and the desired outpupt d. This desired output is compare with the actual network output y. That is,

d = desired output

y = actual output

Thus, the error of our model is:

$$e = d - y$$

In an ideal situation, *'e'* is equal to zero. Our objective is to minimize the error term *'e'* to improve our model. In all the five methods described in this paper, the network first undergoes training from a set of training data. From the knowledge learned during the training, the methods then classify new input with the minimum error possible.

III. Solving the Intrusion Detection Problem Using The Back Propagation Class of Neural Networks

The back propagation method is a technique used in training multilayer neural networks in a supervised manner. The back propagation method, also known as the error back propagation algorithm, is based on the error-correction learning rule [3]. It consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern is applied to the input nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the networks are all fixed. During the backward pass, the synaptic weights are all adjusted in accordance with an error-correction rule. The actual response of the network is subtracted from a desired response to produce an error signal. This error signal is then propagated backward through the network. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense. The weight adjustment is made according to the generalized delta rule [4] to minimize the error. An example of a multilayer perceptron with two hidden layers is shown in Figure 1.

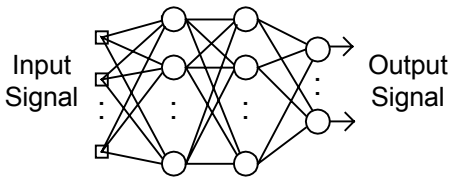

Figure 1: Multilayer Perceptron with Two Hidden Layers

Two commonly used neuron activation functions for the neuron in Figure 1 are sigmoidal

and tansig functions. Both functions are continuously differentiable everywhere and typically have the following mathematical form:

$$\text{Sigmoidal}: \quad f(x) = \frac{1}{1+\exp(-ax)}, \quad a > 0$$

$$\text{Tansig}: \quad f(x) = a\tanh(bx), \quad a \;\&\; b > 0$$

IV. The Different Methods of Back Propagation Neural Networks

Following is a description of each of the BP methods used in this investigation.

Gradient Descent BP (GD)

This method updates the network weights and biases in the direction of the performance function that decreases most rapidly, i.e. the negative of the gradient. The new weight vector $\mathbf{w}_{k+1}$ is adjusted according to:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha\ \mathbf{g}_k$$

The parameter α is the learning rate and $\mathbf{g}_k$ is the gradient of the error with respect to the weight vector. The negative sign indicates that the new weight vector $\mathbf{w}_{k+1}$ is moving in a direction opposite to that of the gradient.

Gradient Descent BP with Momentum (GDM)

Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum. With momentum a network can slide through such a minimum [5].

Momentum can be added to BP method learning by making weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the gradient descent BP rule. The magnitude of the effect that the last weight change is allowed to have is mediated by a momentum constant, μ, which can be any number between *0* and *1*. When the momentum constant is *0* a weight change is based solely on the gradient. When the momentum constant is *1* the new weight change is set to equal the last weight change and the gradient is simply ignored. The new weight vector $\mathbf{w}_{k+1}$ is adjusted as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha\ \mathbf{g}_k + \mu\ \mathbf{w}_{k-1}$$

Variable Learning Rate BP with Momentum (GDX)

The learning rate parameter is used to determine how fast the BP method converges to the minimum solution. The larger the learning rate, the bigger the step and the faster the convergence. However, if the learning rate is made too large the algorithm will become unstable. On the other hand, if the learning rate is set to too small, the algorithm will take a long time to converge. To speed up the convergence time, the variable learning rate gradient descent BP utilizes larger learning rate α when the neural network model is far from the solution and smaller learning rate α when the neural net is near the solution. The new weight vector $\mathbf{w}_{k+1}$ is adjusted the same as in the gradient descent with momentum above but with a varying α_k. Typically, the new weight vector $\mathbf{w}_{k+1}$ is defined as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_{k+1}\mathbf{g}_k + \mu\ \mathbf{w}_{k-1}$$

$$\alpha_{k+1} = \beta\,\alpha_k$$

$$\beta = \begin{cases} 0.7 & \text{if new error} > 1.04\,(\text{old error}) \\ 1.05 & \text{if new error} < 1.04\,(\text{old error}) \end{cases}$$

Conjugate Gradient BP (CGP)

The basic BP algorithm adjusts the weights in the steepest descent direction. This is the direction in which the performance function is decreasing most rapidly. Although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. In the conjugate gradient algorithms the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size which will minimize the performance function along that line. The conjugate gradient used here is proposed by Polak and Ribiere [6]. The search direction at each iteration is determined by updating the weight vector as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha \mathbf{p}_k$$

where : $\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}$

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}$$

and $\Delta \mathbf{g}_{k-1}^T = \mathbf{g}_k^T - \mathbf{g}_{k-1}^T$

Quasi-Newton BP (BFGS)

Newton's method is an alternative to the conjugate gradient methods for fast optimization. Newton's method often converges faster than conjugate gradient methods. The weight update for the Newton's method is:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{A}_k^{-1} \mathbf{g}_k$$

A_k is the Hessian matrix of the performance index at the current values of the weights and biases. When A_k is large, it is complex and time consuming to compute $\mathbf{w}_{k+1}$. Fortunately, there is a class of algorithms based on the works of Broyden, Fletcher, Goldfarb, and Shanno (BFGS) [4,7] that are based on Newton's method but which don't require intensive calculation. This new class of method is called quasi-Newton method. The new weight $\mathbf{w}_{k+1}$ is computed as a function of the gradient and the current weight $\mathbf{w}_k$.

V. The Test Data

In our previous work [1], we showed that users in the UNIX OS environment could be profiled via four attributes – command, host, time, and execution time. In that work, we discovered two important results. First, each computer user has a unique UNIX command set, works a regular schedule, logs onto a regular host, and his commands running time do not vary much. Secondly, user profile drift occurs over time [1,8].

For simplicity in testing the back propagation methods, we decided to generate a user profile data file without profile drift. The generated data used here was organized into two parts. The first part is for training the BP methods. Each training input data came with a desired output. Here, ninety percent (90%) of the input data was generated as authorized traffic and ten percent (10%) as intrusion traffic. The second part is for testing the performance of the five BP methods in intrusion detection. In this part, we generated ninety eight percent (98%) of the traffic to be authorized traffic and two percent (2%) of the traffic to be intrusion traffic.

In both parts of the training and testing data section, several bursts of intrusion data are inserted into the authorized data stream. Each of the generated input data file has *7000* samples. The first *5000* samples are the training data, and the next *2000* samples are the test data. We denoted class *positive one* as authorized traffic and class *negative one* as intrusion traffic. Figure 2 illustrates the construct of the generated data files.

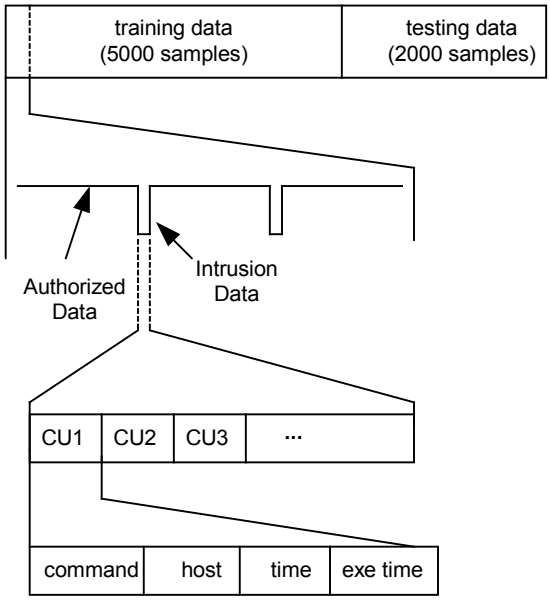

Figure 2: Construction of The Generated Data

In Figure 2, an input sample is defined as set of data being fed into the neural network at one time. A command unit (CU) is defined as a set of four elements – the UNIX command, the login host, the time of login, and the execution time of the command. From these two definitions, we know that one sample can have many CUs.

Our objective is to test the neural networks for detecting intrusion traffic with the fewest number of intrusion samples. With that objective, we generated three files for testing – File1, File2, and File3. File1 consists of 5 CUs in each of the input sample. Likewise, File2 consists of 6 CUs in an input sample, and File3 consists of 7 CUs in an

input sample. Since each CU has 4 elements, the total element for each input file is summarized in Table 1.

	File1	File2	File3
Train Data	100,000	120,000	140,000
Test Data	40,000	48,000	56,000
Total	140,000	168,000	196,000

Table 1: Total Element of the Three Test Files

VI. Performance Comparison

The five BP methods that we used are the gradient descent (GD), the gradient descent with momentum (GDM), the variable learning rate gradient descent with momentum (GDX), the Polak-Ribiere [6] conjugate gradient descent (CGP), and the quasi-Newton (BFGS).

The following notations specify the characteristics of each BP methods. Topology specifies the neural network architecture. In all cases a three-layer neural networks is used. For instance a topology of {20,10,1} indicates a 20-input, 10 hidden neurons, and one output architecture neural network. The parameters α, μ, indicate the learning rate and the momentum constants respectively of the BP, β indicates the change in learning rate for the GDX method. As defined earlier:

$$\beta = \begin{cases} 0.7 & \text{if new error} > 1.04\,(\text{old error}) \\ 1.05 & \text{if new error} < 1.04\,(\text{old error}) \end{cases}$$

The mean square error (MSE) is the condition to terminate training of all the BP methods. MSE is originally set at *exp(-5)*, however, these BP methods can also stop once the training exceeds the number of epoch set. *Nbr Epoch* is set at *1000* originally. The number of computation that each method required to run is specified in *Nbr Flops* (number of floating operations). The error percentage indicates the error in classifying the traffic into intrusion or authorized traffic. The results are summarized in Tables 2, 3, and 4.

Gradient Descent (GD)	
Topology: {20,10,1}	Nbr Epoch = 1000
α = 0.01	Nbr Flops = 5 GFlops
MSE =4.5 exp(-5)	Error Percentage = 0
GD with Momentum (GDM)	
Topology: {20,10,1}	Nbr Epoch = 1000
α = 0.05, μ = 0.75	Nbr Flops = 5 GFlops
MSE = 60	Error Percentage = 100
Variable Learning Rate GD with Momentum (GDX)	
Topology: {20,10,1}	Epoch = 1000
α = 0.05, μ = 0.75, β = 0.7 & 1.05	Nbr Flops = 8.4 GFlops
MSE = 4.5 exp(-4)	Error Percentage = 0
Conjugate Gradient Descent (CGP)	
Topology: {20,6,1}	Nbr Epoch = 35
	Nbr Flops = 0.23 GFlops
MSE = exp(-5)	Error Percentage = 0
Quasi-Newton (BFGS)	
Topology: {20,6,1}	Nbr Epoch = 20
	Nbr Flops = 0.2 GFlops
MSE = 1.2exp(-6)	Error Percentage = 0

Table 2: Result when Input is File1

Gradient Descent (GD)	
Topology: {24,10,1}	Nbr Epoch = 1000
α = 0.01	Nbr Flops = 5.8 GFlops
MSE = 1.1 exp(-4)	Error Percentage = 0
GD with Momentum (GDM)	
Topology: {24,10,1}	Nbr Epoch = 1000
α = 0.05, μ = 0.75	Nbr Flops = 5.8 GFlops
MSE = 2.28 exp(-4)	Error Percentage = 0
Variable Learning Rate GD with Momentum (GDX)	
Topology: {24,10,1}	Epoch = 1000
α = 0.05, μ = 0.75, β = 0.7 & 1.05	Nbr Flops = 1.5 GFlops
MSE = 5.1 exp(-4)	Error Percentage = 0
Conjugate Gradient Descent (CGP)	
Topology: {24,6,1}	Nbr Epoch = 1000
	Nbr Flops = 1.5GFlops
MSE = exp(-5)	Error Percentage = 0
Quasi-Newton (BFGS)	
Topology: {24,6,1}	Nbr Epoch = 45
	Nbr Flops = 0.6 GFlops
MSE = exp(-5)	Error Percentage = 0

Table 3: Result when Input is File2

Gradient Descent (GD)	
Topology: {28,16,1}	Nbr Epoch = 1000
$\alpha = 0.01$	Nbr Flops = 21 GFlops
MSE = 2.3 exp(-5)	Error Percentage = 0
GD with Momentum (GDM)	
Topology: {28,18,1}	Nbr Epoch = 1000
$\alpha = 0.05$, $\mu = 0.75$	Nbr Flops = 15 GFlops
MSE = 2.3 exp(-4)	Error Percentage = 0
Variable Learning Rate GD with Momentum (GDX)	
Topology: {28,12,1}	Nbr Epoch = 1000
$\alpha = 0.05$, $\mu = 0.75$, $\beta = 0.7$ & 1.05	Nbr Flops = 7.5 GFlops
MSE = exp(-3)	Error Percentage = 0
Conjugate Gradient Descent (CGP)	
Topology: {28,10,1}	Nbr Epoch = 45
	Nbr Flops = 0.6 GFlops
MSE = 1.9exp(-3)	Error Percentage = 0
Quasi-Newton (BFGS)	
Topology: {28,8,1}	Nbr Epoch = 30
	Nbr Flops = 0.6 GFlops
MSE = exp(-5)	Error Percentage = 0

Table 4: Result when Input is File3

Table 2 summarized the result obtained when the input was File1. Table 3 summarized the result obtained when the input was File2. Table 4 summarized the result obtained when File3 was used as input. In achieving the results in Tables 2 – 4, we made the following discoveries:

First, the gradient descent with momentum method was not as good in detecting the intrusion traffic from the authorized traffic as the gradient descent method. We varied the learning rate α to value in the range [0.05, 0.25] and momentum constant μ to value in the range [0.7, 0.95]. When File1 was used as input, the gradient descent method with momentum was not able to classify the intrusion traffic from the authorized traffic at all. On the other hand, the gradient descent method was able to classify the intrusion traffic from the authorized traffic. When File2 and File3 were used as inputs, the gradient descent with momentum methods were able to classify intrusion traffic from authorized traffic but with higher *MSE* values.

Second, the number of samples used as inputs affected the performance of the classification of the data. Of the three input files: File1, File2, and File3, the output values for the intrusion and authorized traffics are listed in Table 5.

Output	File1	File2	File3
Intrusion Data	-1 ± 0.1	-1 ± 0.02	-1 ± 0.05
Authorized Data	1 ± 0.1	1 ± 0.02	1 ± 0.05

Table 5: Output Values of the Two Classes

Third, the gradient descent, the gradient descent with momentum and the variable rate gradient descent with momentum method could not converge to a *MSE = exp(-5)*. These three methods converged to a *MSE = exp(-3)*. Nevertheless, they were able to classify the intrusion traffic from the authorized traffic.

Fourth, the conjugate gradient descent and the quasi Newton BP methods have the best performance. These two methods required the least amount of computation measurement in the number of epochs for convergence. For the same input file, these two methods required simpler NN topology. For instance a NN topology of {20,6,1} and {24,6,1} were used for File1 and File2 inputs, and a NN topology of {28,8,1} and {28,10,1} were needed for File3 input. The quasi Newton method had a slightly better performance than the conjugate gradient descent.

Fifth, the input sample that yielded the best performance for all five methods contained 6 CUs (i.e. when the input is File2).

Finally, the number of neurons used at the hidden layer depended on the number of CUs in the input samples. In these cases, the NN topology of {24,10,1} yielded the best results for the gradient descent, the gradient descent with momentum and the variable learning rate gradient descent methods. Similarly the NN topology of {24,6,1} yielded the best results for the conjugate gradient descent and the quasi Newton methods.

VII. Summary

In this paper, we defined the problem of computer network intrusion detection as a classification problem. We then applied five different BP neural network methods to solve this problem.

For the purpose of assuring that the BP methods here are capable of detecting intruders, we generated test data without profile drift to test these BP techniques. We demonstrated that all five BP methods were able to detect intruders logging onto a computer network.

We listed the performance of the five BP methods in detecting intrusion traffic in Tables 2, 3, and 4. We found that the conjugate gradient descent and the quasi-Newton BP methods yielded the best performance in classifying intrusion traffic from the authorized traffic.

As summarized in Table 5, when the input data files were File1 and File3 (i.e. input sample consists

of 5 or 7 CUs), we did not get good results compare to the case when the input is File2. We think that when the input is File1, too little information about the user is presented at the input. In this case, our neural networks methods were not able to fit the training data to generalize the test data. Likewise when the input is File3, too much information is presented at the input. In this case, the neural networks methods fitted the data too closely and a condition of overfitting might have occurred. When the input is File2, the information content in the input sample is just right. The neural networks methods were able to generalize the training to classify the intrusion traffic apart from the authorized traffic.

VIII. Conclusion and Future Work

This paper showed that neural network techniques could be used in intrusion detection of controlled input data files. With the generated data, the conjugate gradient descent BP and the quasi Newton BP can detect the intruders in real time. These two methods only required an average of 6 CUs (command inputs). The performance of the BP methods depend on the neural networks topology. Thus with a given input data file, one should experiment with different topology to get the best performance.

In future work we plan to accomplish these next three action items. First, we need to take into account the profile drift of the users. Second, there are other BP techniques (besides the BP methods discussed in this paper) that we could use in solving problems in intrusion detection. Third, besides the BP methods, there also are other neural networks methods like the radial basis function (RBF) that we can implement to detect the intruders. These methods will be studied next.

IX. References

[1] V. Dao, R. Vemuri, S. Templeton, "Profiling Users in the UNIX OS Environment", International Computer Science Conventions Conference, Dec. 2000.

[2] M. Shin, C. Park, A Radial Basis Function Approach to Pattern Recognition and Its Applications, ETRI Journal, Vol. 22, No. 2, June 2000.

[3] J. Principe, N. Euliano, W. Lefebvre, *Neural and Adaptive System – Fundamentals Through Simulations*, Wiley, 2000.

[4] S. Haykin, *Neural Networks – A Comprehensive Foundation*, 2nd Edition, Prentice Hall, 2000.

[5] ——, *Neural Network Toolbox*, Version 3, The Math Works Inc., 1998.

[6] E. Dennis, R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[7] M. Hagen, H. Demuth, and M. Beale, Neural Network Design, Boston, MA., PWS Publishing, 1996.

[8] T. Lane, C. Brodley, "Temporal Sequence Learning and Data Reduction for Anomaly Detection", http://mow.ecn.purdue.edu/~terran/facts/research.html, 1998.