IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 11, June 2016)

Intrusion Detection System based on Particle Swarm Optimized Neural Network

Priyanka Pawar
priyanka_k8546@yahoo.com

Damodar Tiwari
damodarptiwari21@gmail.com

Abstract –**This paper presents the performance of Neural Network for various values of number of clusters, based on experiments. The optimization of output is done using Particle Swarm Optimization (PSO) by selecting initial through PSO. Particle Swarm Optimization is used to optimize the output of our system, by appropriate selecting the input parameters through PSO. In this paper, an algorithm based on the Particle Swarm Optimization and Neural Network for analyzing program behaviour in intrusion detection is evaluated by experiments. Preliminary experiments with KDD cup'99 Data set show that the PSO optimized Neural Network can effectively detect intrusive attacks and achieves a low false positive rate.**

Keywords – **Intrusion Detection, KDD cup'99, Neural Network, PSO.**

I. INTRODUCTION

In the past two decades, networks have experienced tremendous growth that has speed up a shift in computing environments from centralized computer systems to network information systems. A large volume of valuable information such as personal profiles and credit card information is distributed and transferred through networks. Hence, network security has become more important than ever. However, given open and complex interconnected network systems, it is difficult to establish a secure networking environment. Intruders endanger system security by crashing services, changing critical data, and stealing important information.

Intrusion detection systems (IDSs) are designed to discover malicious activities that attempt to compromise the confidentiality, integrity and assurance of computer systems. Unlike a firewall that filters "bad" traffic, an IDS analyzes packets to detect malicious attack attempts.

Intrusion detection systems have become critical components in network security. Therefore, two factors need to be considered to ensure IDS effectively. First, the IDS should deliver reliable detection results. The detection method should be effective in discovering intrusions since poor detection performance ruins the trustworthiness of the IDS. Second, the IDS should be able to survive in hostile environments or even under attack. However, it is challenging for IDSs to maintain high detection accuracy. An IDS that uses attack signatures to detect intrusions cannot discover novel attacks. As the number of new intrusions increases, these IDSs are becoming incapable of protecting computers and applications. Therefore, a detection approach that is able to discover new attacks is necessary for building reliable IDSs.

Previous research work [1] uses back-propagation algorithm for learning and training the neural network, but there are two major disadvantages with back-propagation algorithm. First is that the initialization of the NN weights is a blind process hence it is not possible to find out globally optimized initial weights and there is a danger that the network output would run towards local optima hence the overall tendency of the network to find out a global solution is greatly affected. The second problem is that back-propagation algorithm is very slow in convergence and there is a possibility that network never converges. This problem of local optimum solution can be solved by optimizing the initial weights of neural network. For this we use particle swarm optimization (PSO) algorithm which is specialized for global searching. For this we first determine the number of inputs, layers and hidden neurons of the neural network and then we would use the back-propagation algorithm to train the networks using the weights optimized by PSO.

A computer system should provide confidentiality, integrity and assurance against denial of service. However, due to increased load and connectivity more and more system is subject to attack by intruders. These attempts try to exploit flaws in the operating system as well as in the application programs. In fact, it is not possible to build a complete secure system. It can have cryptographic methods but they have their own problems as passwords can easily be cracked, users can lose their passwords and entire crypto-system can be broken. Even a truly secure system is vulnerable to abuse by insiders who abuse their

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 11, June 2016)

privileges. Also, we need a balance between access control and user efficiency as stricter the mechanism, the lower the efficiency. So we need a system which is real time i.e. we would like to detect them as soon as possible and take appropriate action. This is what an intrusion detection system does. It is reactive rather than proactive. This dissertation tries to build a system which created clusters from its input data by labeling clusters as normal or anomalous data instances and finally used these clusters to classify unseen network data instances as either normal or anomalous. Both training and testing was done using different subset of KDD Cup 99 data which is very popular and widely used intrusion attack dataset.

II. PROPOSED METHOD

In this research work, Particle Swarm Optimized Neural Network approach determines optimum number of clusters in analyzed data.

Block Diagram

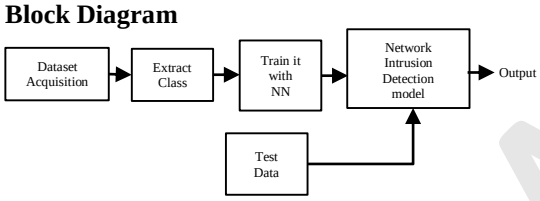

Figure 1: Block diagram for proposed approach using Neural Network only

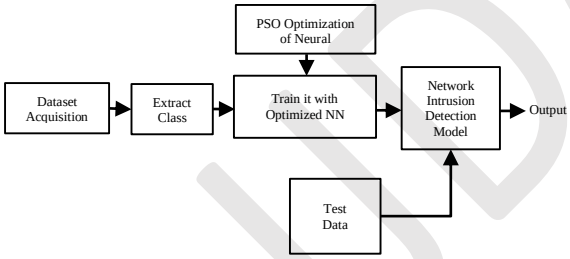

Figure 2: Block diagram for proposed approach using PSO optimized Neural Network

Figure 1 shows the basic block diagram of Neural Network based intrusion detection system and Figure 2 shows the block diagram for the proposed research work. It includes following phases.

Data Acquisition

In the very first phase, the test data is taken form KDD Cup'99 dataset. Since KDD'99 [2] has been the most wildly used data set for the evaluation of anomaly detection methods. This data set is prepared by Stolfo et al. [3] and is built based on the data captured in DARPA'98 IDS evaluation program [4]. DARPA'98 is about 4 gigabytes of compressed raw (binary) TCP dump data of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with about 100 bytes. The two weeks of test data have around 2 million connection records. KDD training dataset consists of approximately 4,900,000 single connection vectors each of which contains 41 features and is labelled as either normal or an attack, with exactly one specific attack type. The simulated attacks fall in one of the following four categories:

1. **Denial of Service Attack (DoS):** It is an attack in which the attacker makes some computing or memory resource too busy or too full to handle legitimate requests, or denies legitimate users access to a machine.
2. **User to Root Attack (U2R):** It is a class of exploit in which the attacker starts out with access to a normal user account on the system (perhaps gained by sniffing passwords, a dictionary attack, or social engineering) and is able to exploit some vulnerability to gain root access to the system.
3. **Remote to Local Attack (R2L):** It occurs when an attacker who has the ability to send packets to a machine over a network but who does not have an account on that machine exploits some vulnerability to gain local access as a user of that machine.
4. **Probing Attack:** It is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls.

The datasets contain a total number of 24 training attack types, with an additional 14 types in the test data only. KDD'99 features can be classified into five groups:

1. **Basic Features:** This category encapsulates all the attributes that can be extracted from a TCP/IP connection. Most of these features leading to an implicit delay in detection.
2. **Traffic Features:** This category includes features that are computed with respect to a window interval and is divided into two groups.
3. **"Same host" Features:** It examine only the connections in the past 2 seconds that have the same destination host as the current connection, and calculate statistics related to protocol behaviour, service, etc.
4. "**Same service" Features:** It examines only the connections in the past 2 seconds that have the same service as the current connection.

   The two aforementioned types of "traffic" features are called time-based. However, there are several slow probing attacks that scan the hosts (or ports) using a much larger time interval than 2 seconds, for

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 11, June 2016)

example, one in every minute. As a result, these attacks do not produce intrusion patterns with a time window of 2 seconds. To solve this problem, the "same host" and "same service" features are re-calculated but based on the connection window of 100 connections rather than a time window of 2 seconds. These features are called connection-based traffic features.

5. **Content Features:** Unlike most of the DoS and Probing attacks, the R2L and U2R attacks don't have any intrusion frequent sequential patterns. This is because the DoS and Probing attacks involve many connections to some host(s) in a very short period of time; however the R2L and U2R attacks are embedded in the data portions of the packets, and normally involves only a single connection. To detect these kinds of attacks, we need some features to be able to look for suspicious behavior in the data portion, e.g., number of failed login attempts. These features are called content features.

Extract Class

There were a total of 24 attack types in the data set. The simulated attacks fell in exactly one of the four categories, User to Root; Remote to Local; Denial of Service; and Probe.

- **Denial of Service (dos):** Attacker tries to prevent legitimate users from using a service.
- **Remote to Local (r2l):** Attacker does not have an account on the victim machine, hence tries to gain access.
- **User to Root (u2r):** Attacker has local access to the victim machine and tries to gain super user privileges.
- **Probe:** Attacker tries to gain information about the target host.

Out of these, only five classes of test data is taken for training and testing purpose out of which three come under DOS attack and one comes under probe attack. Each class is labelled as one specific kind of attack. Table 1 shows the extracted class of attacks.

Table 1: DOS Attacks

S. No	Attack	Category
1	Smurf	DOS
2	Neptune	DOS
3	Back	DOS
4	Normal	Normal
5	IPsweep	Probe

1. **Smurf Attack:** The Smurf Attack is a way of generating significant computer network traffic on a victim network. This is a type of denial-of-service attack that floods a system via spoofed broadcast ping messages. This attack relies on a perpetrator sending a large amount of ICMP (Internet Control Message Protocol) echo request (ping) traffic to IP broadcast addresses, all of which have a spoofed source IP address of the intended victim.
2. **Neptune Attack:** It is a denial of service attack to which every TCP/IP implementation is vulnerable (to some degree). Each half-open TCP connection made to a machine causes the 'TCPD' server to add a record to the data structure that stores information describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections.
3. **Back Attack:** It involves sending forged requests of some type to a very large number of computers that will reply to the requests. Using Internet protocol spoofing, the source address is set to that of the targeted victim, which means all the replies will go to (and flood) the target.
4. **Normal:** Data which does not contains any attack.
5. **IPsweep:** The IPsweep and Portsweep, as their names suggest, sweep through IP addresses and port numbers fora victim network and host respectively looking for open ports that could potentially be used later in an attack.

PSO Optimization of Neural Network

Particle Swarm Optimization (PSO) is an intelligent algorithm to fix the discrete optimization problem. Thus the algorithm should be improved because the optimization of the parameters in the back propagation neural network is continuous optimization problem. Assuming there is following continuous optimization problem:

$$y = \min f(x), X = (x_1, x_2, \ldots x_d) \qquad (1)$$

Particle Swarm Algorithm

1. Begin
2. Factor settings and swarm initialization
3. Evaluation
4. g = 1
5. While (the stopping criterion is not met) do
6. for each particle

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 11, June 2016)

7. Update velocity
8. revise place and localized best place
9. Evaluation
10. End For
11. Update leader (global best particle)
12. g + +
13. End While
14. End

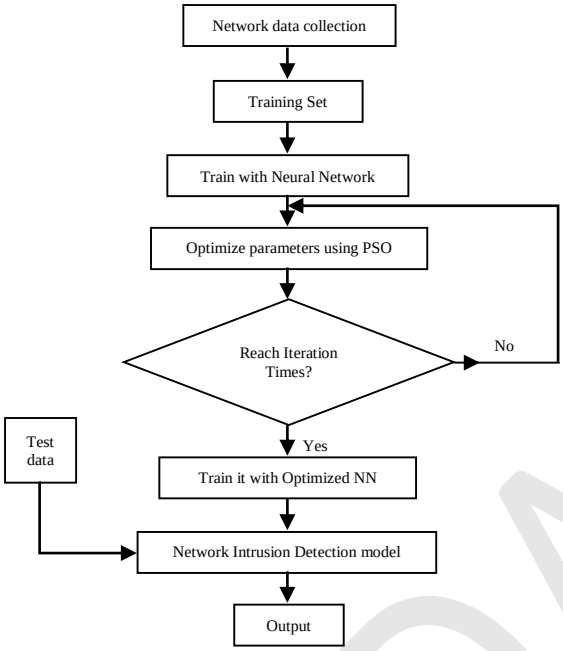

Figure 3: Flow diagram for PSO-BP Neural Network based network intrusion detection model

The PSO procedure has various phases consist of Initialization, Evaluation, Update Velocity and Update Position.

Initialization

The initialization phase is used to determine the position of the m particles. The random initialization is one of the most popular methods for this job. There is no assurance that a randomly created particle be a better answer and this will make the initialization more attractive.

A good initialization algorithm makes the optimization algorithm more efficient and reliable. For initialization, initial information or knowledge of the problem can help the algorithm to converge in less iterations.

Update Velocity and Position

In each iteration, each particle updates its velocity and position according to its heretofore best position, its current velocity and some information of its neighbours. Equation (2) is used for updating the velocity:

$$V_i^{(k+1)} = w * V_i^k + C_1 * rand_1 * \left(pbest_i^k - S_i^k\right) + C_2 * rand_2 * \left(gbest^k - S_i^k\right) \quad (2)$$

Where V_i^k is the velocity of i^{th} particle vector at k^{th} iteration;

w is the weighting function;

$C_1$and C_2 are the positive weighting factors;

$rand_1$ and $rand_2$are the random numbers between 0 and 1;

S_i^k is the current position of i^{th} particle vector $h(n)$ at k^{th} iteration;

$pbest_i^k$is the personal best of the i^{th} particle at the k^{th} iteration;

$gbest^k$is the group best of the group at the k^{th} iteration.

The searching point in the solution space may be modified by the following equation:

$$S_i^{(k+1)} = S_i^k + V_i^{(k+1)} \quad (3)$$

The first term of Equation (2) is the previous velocity of the particle vector. The second and third terms are used to change the velocity of the particle vector. Without the second and third terms, the particle vector will keep on ‘‘flying’’ in the same direction until it hits the boundary. Namely, it corresponds to a kind of inertia represented by the inertia constant, w and tries to explore new areas.

Training with Optimized Neural Network

In previous phase, neural network is optimized using particle swarm optimization then the optimized NN is used to train extracted class data using back propagation algorithm.

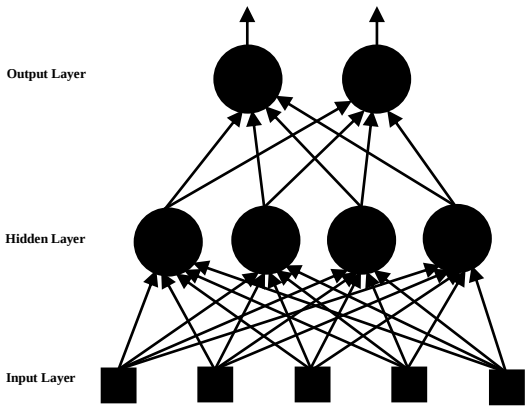

Figure 4: Structure for back propagation neural network

Objective Function

Back propagation neural network is a type of multi-layer feed forward network in which each layer is connected by transfer functions and can fulfil arbitrary nonlinear mapping. It is widely applied in stock price, petroleum price, economic time

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 11, June 2016)

sequence, network flow and other nonlinear areas and attained satisfactory performance. The structure of back propagation neural network is shown in Figure 3.

The basic learning process of the back propagation neural network algorithm is as follows:

1. Initialize the connection weights w_{ij}, v_{jt} and threshold $\theta_j$in the back propagation neural network.
2. Input the first learning sample couples to the back propagation neural network.
3. Compute the input $u_j$of each neural unit and the output $h_j$in the hidden layer. The equation is:

$$u_j = \sum_{i=1}^{n} w_{ij} x_i - \theta_j \quad (4)$$

$$h_j = f(u_j) = \frac{1}{1+\exp(-u_j)} \quad (5)$$

4. Compute the input l_t of each neural unit and the output $y_t$in the output layer. The equation is:

$$l_t = \sum v_{jt} h_j - \gamma_t \quad (6)$$

$$y_t = \frac{1}{1+\exp(-l_t)} \quad (7)$$

5. Compute the weights error $\delta_t$which is connected to the neural unit t in the output layer.

$$\delta_t = (c_t - y_t) y_t (1 - y_t) \quad (8)$$

In the equation (8), c_t represents the expectation of the sample.

6. Compute the weights error $\delta_j$which is connected to the neural unit j in the hidden layer.

$$\delta_j = \sum_{t=1}^{q} \delta_t v_{jt} h_j (1 - h_j) \quad (9)$$

7. Update the connection weights v_{jt} and threshold γ_t in the back propagation neural network.

$$v_{jt}(N+1) = v_{jt}(N) + \alpha \delta_t h_j \quad (10)$$

$$\gamma_t(N+1) = \gamma_t(N) + \beta\delta \quad (11)$$

8. Update the connection weights w_{jt} and threshold $\theta_j$in the back propagation neural network.

$$w_{jt}(N+1) = w_{jt}(N) + \alpha \delta_j x_i \quad (12)$$

$$\theta_j(N+1) = \theta_j(N) + \beta \delta_j \quad (13)$$

9. Input the next learning sample and go to the step 3 until all of the samples are trained.
10. Back propagation neural network go to a new round of learning. If it meets the equation (14), the training of the back propagation network can be ended.

$$\left|\sum_{k=1}^{z} E_k\right| \le \varepsilon \quad (14)$$

In the equation (14), ε represents the accuracy requirement of back propagation neural network, E_k represents the mean square error and the definition are as follows:

$$E_k = \frac{1}{2}\sum_{t=1}^{q}(c_t - y_t)^2 \quad (15)$$

Before training the back propagation neural network, proper connection weights w_{ij} and v_{jt} of the back propagation neural network should be chosen. Normally the initialization is randomly which can cause the convergence is slow and the defect of local optimal solutions.

Pseudo Code

The back propagation algorithm for a 3-layer network (only one hidden layer) is as follows:

initialize the weights in the network (often small random values)
do
  for *each image i in the training set of*
    database O = neural-network-output(network, i)
    T = desired output for i
    calculate error (T - O) at the output units;
    calculate δ_h for all weights from hidden layer to output layer;
    calculate δ_i for all weights from input layer to hidden layer;
    update the weights to minimize error in the network; ***until*** *some stopping criterion satisfied*
return *the network*

III. Simulation And Results

The performance of proposed algorithms has been studied by means of MATLAB simulation.

Figure 5: Confusion Matrix plot for Intrusion classifier scheme using Neural Network

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 11, June 2016)

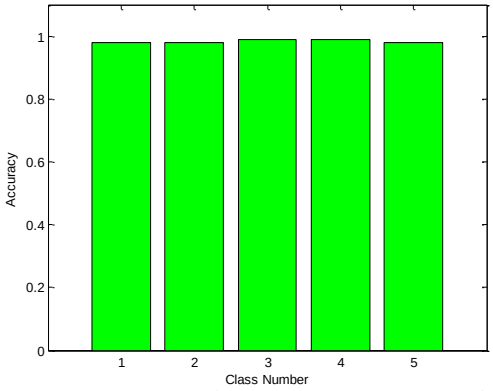

Figure 6: Accuracy graph for neural network based classifier

Table 1: Accuracy for neural network based classifier

Class Number	Accuracy
1	0.9795
2	0.9795
3	0.9899
4	0.9895
5	0.9800

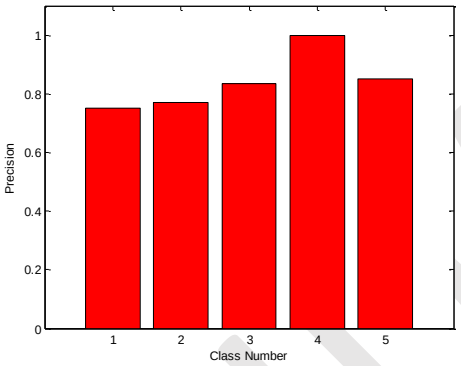

Figure 7: Precision graph for neural network based classifier

Table 2: Precision for neural network based classifier

Class Number	Precision
1	0.7500
2	0.7692
3	0.8333
4	1.0000
5	0.8500

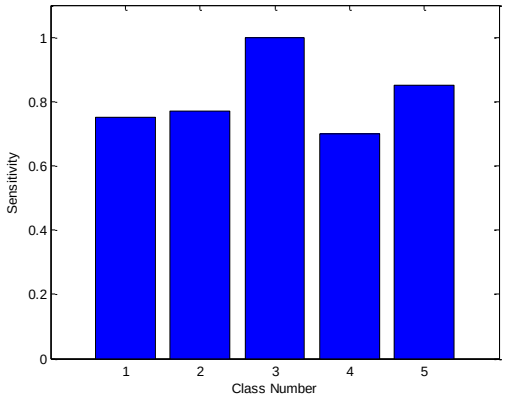

Figure 8: Sensitivity graph for neural network based classifier

Table 3: Sensitivity for neural network based classifier

Class Number	Sensitivity
1	0.7500
2	0.7692
3	1.0000
4	0.7000
5	0.8500

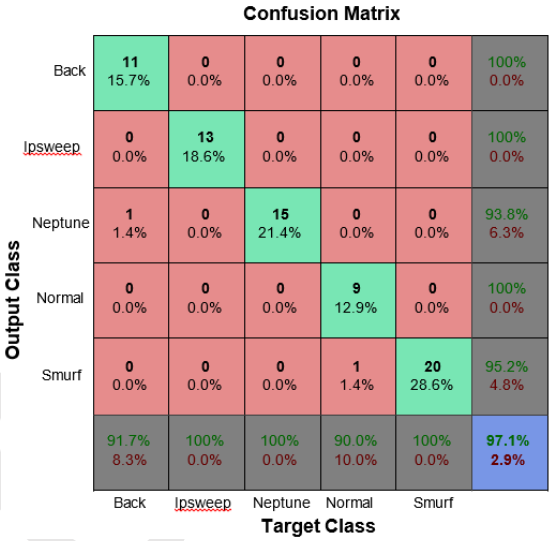

Figure 9: Confusion Matrix plot for Intrusion classifier scheme using PSO-Neural Network

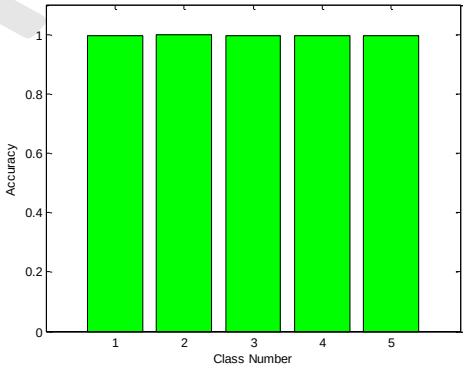

Figure 10: Accuracy graph for PSO-Neural Network based classifier

Table 4: Accuracy for PSO-Neural Network based classifier

Class Number	Accuracy
1	0.9966
2	1.0000
3	0.9966
4	0.9965
5	0.9967

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 4, Issue 11, June 2016)

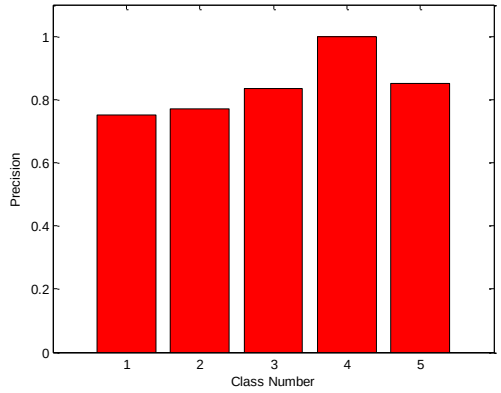

Figure 11: Precision graph for PSO-Neural Network based classifier

Table 5: Precision for PSO-Neural Network based classifier

Class Number	Precision
1	1.0000
2	1.0000
3	0.9375
4	1.0000
5	0.9524

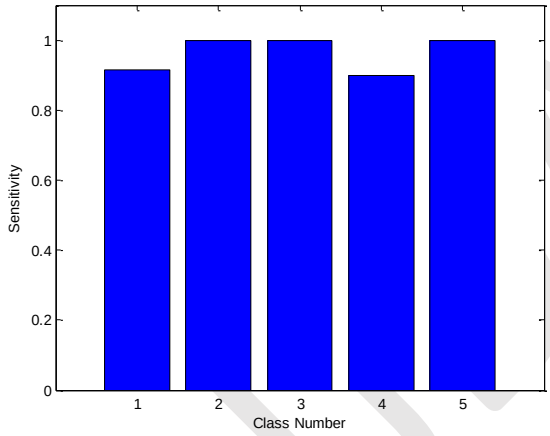

Figure 12: Sensitivity graph for PSO-Neural Network based classifier

Table 6: Sensitivity for PSO-Neural Network based classifier

Class Number	Sensitivity
1	0.9167
2	1.0000
3	1.0000
4	0.9000
5	1.0000

IV. CONCLUSION

The preliminary experiments with the 1999 KDD cup'99 Database have shown that this approach is able to effectively detect intrusive program behaviour. With the frequency-weighting method where each entry is equal to the number of occurrences of a system call during the TCP (Transmission Control Protocol) communication. Neural Network training process easily added to the training data set without changing the weights of the existing training samples. Particle Swarm Optimization is used to optimize the output of our system, by appropriate selecting the input parameters through PSO.

REFERENCE

[1] Chen Yan, "Intelligent Intrusion Detection based on Soft Computing", Seventh International Conference on Measuring Technology and Mechatronics Automation 2015.

[2] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," ACM Transactions on Information and System Security, Volume 3, No. 4, pp. 262–294, 2000.

[3] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost based modeling for fraud and intrusion detection: Results from the jam project," discex, Volume 02, p. 1130, 2000.

[4] R. P. Lippmann, D. J. Fried, J. W. Haines, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation," Volume 02, pp.1012, 2000.

[5] Venkata Suneetha Takkellapati, G.V.S.N.R.V Prasad, "Network Intrusion Detectionsystem based on Feature Selection and Triangle area Support Vector Machine", International Journal of Engineering Trends and Technology- Volume 03, Issue 4, 2012.

[6] Esh Narayan, Pankaj Singh and Gaurav Kumar Tak, "Intrusion Detection System Using Fuzzy C-Means Clustering with Unsupervised Learning via EM Algorithms" VSRD-IJCSIT, Volume 2 (6), 502-510, 2012.

[7] Deepika Dave, Prof. Vineet Richhariya, "Intrusion detection with KNN classification and DS- theory", IRACST Volume 2, No.2, April 2012.

[8] P.S.Prabhu, "Network Intrusion Detection Using Enhanced Adaboost Algorithm", International Journal of Communications and Engineering Volume 3, No.3, Issue: 02 March 2012.

[9] Dalila BOUGHACI, Mohamed Lamine HERKAT, Mohamed Amine LAZZAZI,"A Specific Fuzzy Genetic Algorithm for Intrusion Detection", ICCIT, 2012.

[10] R. Shanmugavadivu, Dr.N.Nagarajan, "Network Intrusion Detection System Using Fuzzy Logic" IJCSE Volume 2 No. 1, 2011.