World Academy of Science, Engineering and Technology
International Journal of Humanities and Social Sciences
Vol:1, No:8, 2007

An Approach for Reducing the Computational Complexity of LAMSTAR Intrusion Detection System using Principal Component Analysis

V. Venkatachalam, and S. Selvan

Abstract**—**The security of computer networks plays a strategic role in modern computer systems. Intrusion Detection Systems (IDS) act as the ''second line of defense'' placed inside a protected network, looking for known or potential threats in network traffic and/or audit data recorded by hosts. We developed an Intrusion Detection System using LAMSTAR neural network to learn patterns of normal and intrusive activities, to classify observed system activities and compared the performance of LAMSTAR IDS with other classification techniques using 5 classes of KDDCup99 data. LAMSAR IDS gives better performance at the cost of high Computational complexity, Training time and Testing time, when compared to other classification techniques (Binary Tree classifier, RBF classifier, Gaussian Mixture classifier). we further reduced the Computational Complexity of LAMSTAR IDS by reducing the dimension of the data using principal component analysis which in turn reduces the training and testing time with almost the same performance.

Keywords**—**Binary Tree Classifier, Gaussian Mixture, Intrusion Detection System, LAMSTAR, Radial Basis Function.

I. Introduction

COMPUTER security has become a critical issue with the rapid development of business and other transaction systems over the internet. Intrusion detection is to detect intrusive activities while they are acting on computer network systems. There are two major intrusion detection techniques: misuse detection and anomaly detection [1]. Misuse detection discovers attacks based on the patterns extracted from known intrusions. Anomaly detection identifies attacks based on the deviations from the established profiles of normal activities. Activities that exceed thresholds of the deviations are detected as attacks. Misuse detection has low false positive rate, but cannot detect new types of attacks. Anomaly detection can detect unknown attacks, under a basic assumption that attacks deviate from normal behavior.

We developed an Intrusion Detection System using LAMSTAR neural network to learn patterns of normal and intrusive activities, to classify observed system activities and compared the performance of LAMSTAR IDS with other classification techniques using 5 classes of KDDCup99 data. LAMSAR IDS gives better performance at the cost of high Computational complexity, Training time and Testing time, when compared to other classification techniques (Binary Tree classifier, RBF classifier, Gaussian Mixture classifier). we further reduced the computational complexity of LAMSTAR IDS by reducing the dimension of the data using principal component analysis which in turn reduces the, training and testing time with almost the same performance.

This paper is organized as follows: Section 2 gives some theoretic background about LAMSTAR Neural Network. Section 3 presents the details about KDDCup99 dataset used for testing and training the Intrusion Detection system, and the cost matrix used to calculate cost per example. Section 4 gives details about the principal component analysis. In Section 5&6 we discuss the dimension reduction of data and performance of various algorithms in classifying the data. Section 7 summarizes the obtained results with comparison and discussions. The paper is finally concluded in section 8 with the most essential points.

Manuscript received November 22, 2006.

V. Venkatachalam is with the Erode Sengunthar Engineering College, Thudupathi, Erode-638057, Tamilnadu, India (phone: +91-04294 232704, 98420 20719; e-mail:vv@erode-sengunthar.ac.in).

S. Selvan is with the PSG College of Technology, Coimbatore, Tamilnadu, India.

II. LAMSTAR

A Large Scale Memory Storage and Retrieval (LAMSTAR) network is proposed in [2],[3] by combining SOM modules and statistical decision tools. It was specifically developed for application to problems involving very large memory that relates to many different categories (attributes) where some data is exact while the other is fuzzy and where for a given problem some categories might be totally missing [2]. Large Scale Memory Storage and Retrieval (LAMSTAR) network research, which targets large-scale memory storage and retrieval problems. This model attempts to imitate, in a gross manner, processes of the human central nervous system (CNS) concerning storage and retrieval of patterns, impressions, and sensed observations including processes of forgetting and recollection. It attempts to achieve this without contradicting findings from physiological and psychological observations, at least in an input/output manner. Furthermore, it attempts to do so in a computationally efficient manner using tools of neural networks, especially Self-Organizing-Map based (SOM) network modules, combined with statistical decision tools. Its design was guided by trying to find a mechanistic neural network-based model for very general storage and retrieval processes involved. This general

approach is related to Minsky's idea that the human brain consists of many agents, and a knowledge link is formed among them whenever the human memorizes an experience. When the knowledge link is subsequently activated, it reactivates the mental agents needed to recreate a mental state similar to the original. The LAMSTAR network employs this general philosophy of linkages between a large number of physically separate modules that represent concepts, such as time, location, patterns, etc., in an explicit algorithmic network.

The LAMSTAR network has been successfully applied in fields of medicine (diagnosis)[4]-[6], engineering (automotive fault detection) and multimedia information systems[7]. Whereas the LAMSTAR design addresses large-scale memory retrieval problems, we use LAMSTAR concepts to processes of storage and retrieval, interpolation and extrapolation of input data, and the use of reward-based correlation-links between modules to detect intrusions. In this modified LAMSTAR network, each Kohonen SOM module represents a class of sub-patterns. The model assumes that the input patterns have been separated into sub-patterns before entering the SOM module. The network is thus organized to assign each neuron to a class of neurons (i.e., one SOM module) that best corresponds to the input sub-pattern. This SOM configuration yields very rapid matching with good error tolerance, and is capable of generalization. Arrays of correlation links (C-links) connect the modules using coefficients determined by the statistical correlations between the various patterns considered. A coordinated activation of neurons between the various modules allows the network to recreate (interpolate) complex patterns and make associations.

A. LAMSTAR IDS Design

A modified LAMSTAR network used for intrusion detection is as shown in Fig. 1. The model reads in KDDCup 99 data sends it first to the feature extraction module which extracts 41 features of the data and sends it to preprocessing module. The preprocessing module converts the 41 features into a standardized numeric representation. Normalization block reads the preprocessed data and normalizes the data into a format required by the SOM's. The normalized input pattern was split into sub patterns (basic features 9, content features 13, traffic 9, and others 10) [8]. Each sub pattern is given to one SOM module. This SOM configuration yields very rapid matching with good error tolerance, and is capable of generalization.

Between SOM modules, connections are established using correlation links. The correlation links distribute information between various modules. The training data contains 22 attack patterns and normal patterns. The SOM modules are trained using this pattern. The coordinated activation of neurons between the various modules allows the network to detect intrusions.

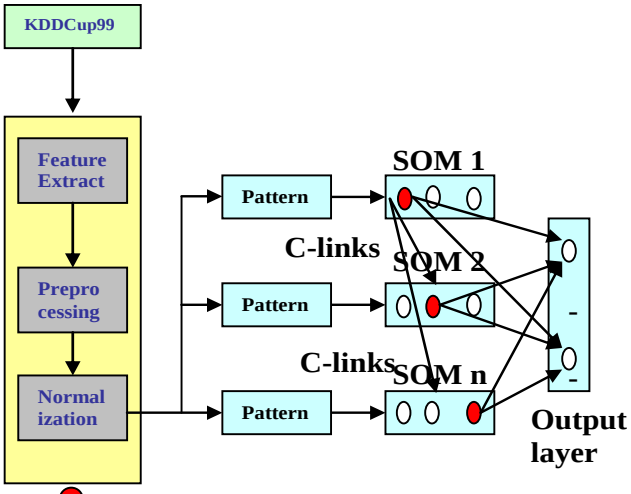

Fig. 1 Modified LAMSTAR architecture

The input pattern is stored as a real vector x given by:

$$X = \left[x^{1T}, \ldots x^{iT}, \ldots x^{mT} \right]^T \quad (1)$$

To store data concerning the *i'th* category of the input pattern, each sub-pattern $\mathbf{x}_i$ is channeled to the corresponding *i'th* SOM module. A winning neuron is determined for each input based on the similarity between the input vector $\mathbf{x}_i$ and weight vectors $\mathbf{w}_i$ (stored information). For a sub-pattern $\mathbf{x}_i$, the winning neuron is determined by the minimum Euclidean distance between $\mathbf{x}_i$ and $\mathbf{w}_i$:

$$\| x^i - w^i_{winner} \| = \min \| x^i - w^i_k \| \; \forall \quad k \quad (2)$$

where x^i - input vector in i'th SOM module

winner -index of the winning neuron

w^i_{winner} - winner weight vector in ith SOM module

k - a number of neurons(stored patterns)in ith SOM module

||-|| - Vector Euclidean distance

$$\| x - w \| = \sum_{i=1}^{i=n} \left(w_i - x^i \right)^2$$

where n - dimension of sub vectors x and w

The SOM module is a Winner-Take-All [9] network where only the neuron with the highest correlation between its input vector and its correspondence weight vector will have a non-zero output. The Winner-Take-All feature also involves lateral inhibition such that each neuron has a single positive feedback onto itself and negative feedback connections to all other units.

$$o^i_j = \begin{cases} 1 & for \; \| x^i - w^i_{winner} \| < \| x^i - w^i_j \| \\ 0 & Otherwise \end{cases} \quad \forall \; winner \neq j \quad (3)$$

where

O_j^i - output of neuron j in ith SOM module

W_{winner}^i - winning weight vector in ith SOM module

winner - index of winning neuron in ith SOM module

The neuron with the smallest error determined is declared the *winner* and its weights $\mathbf{W_{winner}}$ are adjusted using the Hebbian learning law, which leads to an approximate solution:

$$W_{winner}^i(t+1) = W_{winner}^i(t) + \alpha(X^i(t) - W_{winner}^i(t) \quad (4)$$

α -Learning rate a slowly decreasing function of time, initial weights are assumed with random values. The learning rate is updated by, α (t+1) = 0.5 α (t).

The adjustment in the LAMSTAR SOM module is weighted according to a pre-assigned Gaussian hat neighborhood function $\Delta(winner,j)$.

$$W_j^i(t+1) = W_j^i(t) + \Delta(winner, j).\alpha(X^i(t) - W_j^i(t)) \quad (5)$$

Where $W_j^i(t+1)$ - new weight of the neighbour neuron j from winning neuron

$\Delta(winner, j)$ – Neighborhood define as gaussian hat

B. Training Phase

The training of the SOM modules are done as described below SOM modules are trained with sub-patterns derived from the KDDCup99 data. Given an input pattern **x** and for each $\mathbf{x}_i$ sub-pattern to be stored, the network inspects all weight vectors **w** in the *i'th* SOM module. If any previously stored pattern matches the input sub-pattern within a preset tolerance (*error* ε), the system updates the proper weights or creates a new pattern in the SOM module. The choice of ε 's value depend on the size of the cluster. The following expression is used to calculate the value of ε:

$$\varepsilon = \underset{x\varepsilon.cli}{MAX}.\ dist(x, c_i) / 10 \quad (6)$$

where c_i is the cluster $_i$ center and cli is the cluster i. It stores the input sub-pattern $\mathbf{x}_i$ as a new pattern, $\mathbf{x} = \mathbf{w}_j$, where index *j* is the first unused k_j neuron in *i'th* SOM module. If there are no more 'free' neurons, the system will fail, which means either the preset tolerance has to be increased to include more patterns in the same cluster of already stored patterns, or more neurons have to be added on the *i'th* SOM module.

Correlation links C-links among SOM modules are created as follows. Individual neurons represent only a limited portion of the information input. Sub-patterns are stored in SOM's and the correlation links between these sub-patterns are established in such a way that the information's are distributed between neurons in various SOM modules and correlation links. Even if one neuron fails only a little information is lost since the information is spread among SOM's and correlation links. Correlation-link coefficient values C-link are determined by evaluation distance minimization to determine winning neurons, where a win activates a count-up element associated with each neuron and with its respective input-side link. During training sessions, the values of C-links are modified according to the following simple rule (reward)

$C_{k,j}^{i,l}$ (new)= $C_{k,j}^{i,l}$ (old) - $\beta_{reward}($ $C_{k,j}^{i,l}$ (old)- C Max) , for $C_{k,j}^{i,l}$ (old) $\neq 0$, 1 otherwise (7)

$C_{k,j}^{i,l}$ - correlation link between k'th neuron in i'th SOM module and l' th neuron in j'th SOM module

β_{reward} - reward coefficient ,initially value is assumed with random values . β_{reward}(t+1) = .5 β_{reward}(t)

To keep link-weights within a reasonable range, whenever the highest of all weights reaches a certain threshold all link-weights to that SOM are uniformly reduced by the same proportion, for example 50%. Additionally, link-weights are never reduced to zero or the connection between the two neurons will be lost permanently. If the correlation link between two sub-patterns already exists, namely, $C_{k,l}^{i,j} > 0$ (a result from previous training), the formula of equation 6 updates (increases) the analyzed C-link. If there are no correlations ($C_{k,l}^{i,j} = 0$), the system creates new C-link with initial value $C_{k,l}^{i,j} = 1$.

C. Detection Phase

The sub-patterns from input pattern is selected and the correlations with stored sub-patterns in each SOM module is examined. For example, one *i'th* SOM module could have previously stored source IP address, and will correlate any given input *i'th* sub-pattern and determine if there is a match or not. The Intruder packet is detected by means of its C-links. Once all the winning neurons are determined, the system obtains all correlation-links coefficient values among all SOM modules. The output SOM layer (Fig. 1), with which all C-links are inter-connected, will determine whether the input pattern is an intruder packet or a normal packet.

III. Data Set & Cost Matrix

A. Data Set

The KDDCup99 intrusion detection datasets are based on the 1999 DARPA initiative, which provides designers of intrusion detection systems (IDS) with a benchmark on which to evaluate different methodologies .We trained and tested our system using KDDCup99 dataset [8],[10] which covers 22 attack types in the training data which are classified into 5 classes : Denial of Service (**DoS**) attacks:deny legitimate

requests to a system, e.g.syn flood, User-to-Root (**U2R**) attacks: unauthorized access to local super user(root) privileges, e.g.various buffer overflow attacks, Remote-to-Local (**R2L**) attacks: unauthorized access from a remote machine, e.g. guessing password, and **Probing**: surveillance and other probing, e.g. port scanning. The 1999 Defense Advanced Research Projects Agency (**DARPA**) Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. Table I gives the details of KDDCup99 data.

TABLE I
KDDCUP99 TRAINING AND TESTING DATA

Dataset Label	DOS	PROBE	U2R	R2L	Total Attack	Total Normal
Training data	391458	4107	52	1126	494020	97277
Testing data	229853	4166	228	16189	311029	60593

B. Feature Extractions and Preprocessing

The input data to the neural network must be in the range [0 1] or [-1 1]. Hence preprocessing and normalization of data is required. The KDDCup99 format data is preprocessed. Each record in KDDCup99 format has 41 features, each of which is in one of the continuous, discrete and symbolic form, with significantly varying ranges. Based on the type of neural nets, the input data may have different forms and so needs different preprocessing. Some neural nets only accept binary input and some can also accept continuous-valued data. In Preprocessor, after extracting KDDCup99 features from each record, each feature is converted from text or symbolic form into numerical form. For converting symbols into numerical form, an integer code is assigned to each symbol. For instance, in the case of protocol_type feature, 0 is assigned to *tcp*, 1 to *udp*, and 2 to the *icmp* symbol. Attack names were first mapped to one of the five classes, 0 for Normal, 1 for Probe, 2 for DoS, 3 for U2R, and 4 for R2L.

Two features spanned over a very large integer range, namely src_bytes [0, 1.3 billion] and dst_bytes [0, 1.3 billion]. Logarithmic scaling (with base 10) was applied to these features to reduce the range to [0.0, 9.14]. All other features were boolean, in the range [0.0, 1.0]. Hence scaling was not necessary for these attributes.

C. Normalizations

For normalizing feature values, a statistical analysis is performed on the values of each feature based on the existing data from KDDCup99 dataset and then acceptable maximum value for each feature is determined. According to the maximum values and the following simple formula, normalization of feature values in the range [0,1] is calculated.

If (f > MaxF) Nf=1; Otherwise Nf = (f / MaxF) (8)

F: Feature f: Feature value MaxF: Maximum acceptable value for F Nf: Normalized or scaled value of F

Another simple way to normalize the data is to use SOM toolbox of the MATLAB software. In this paper the following MATLAB commands were used to normalize the data.

```
sD=som_read_data('KDDCup99.data')
sD=som_normalize(sD,'var',1:4)
sD=som_normalize(sD,'log',5:6)
sD=som_normalize(sD,'var',7:41)
sD=som_normalize(sD,'var',1:41)
```

D. Cost Matrix

A cost matrix (C) is defined by associating classes as labels for the rows and columns of a square matrix: in the current context for the KDDCup99 dataset, there are five classes, {Normal, Probe, DoS, U2R, R2L}, and therefore the matrix has dimensions of 5×5. An entry at row i and column j, C(i,j), represents the non-negative cost of misclassifying a pattern belonging to class i into class j. Cost matrix values employed for the KDDCup99 dataset are defined elsewhere in [11]. These values were also used for evaluating results of the KDDCup99 competition. The magnitude of these values was directly proportional to the impact on the computing platform under attack if a test record was placed in a wrong category. A confusion matrix (CM) is similarly defined in that row and column labels are class names: a 5×5 matrix for the KDDCup99 dataset. An entry at row i and column j, CM(i,j), represents the number of misclassified patterns, which originally belong to class i yet mistakenly identified as a member of class j. Given the cost matrix as predefined in [11] and the confusion matrix obtained subsequent to an empirical testing process, cost per example (CPE) was calculated using the formula,

$$\text{CPE}=\frac{1}{N}\sum_{i=1}^{5}\sum_{j=1}^{5}CM(i,j)*C(i,j) \qquad (9)$$

where CM corresponds to confusion matrix, C corresponds to the cost matrix, and N represents the number of patterns tested. A lower value for the cost per example indicates a better classifier model. Comparing performances of classifiers for a given attack category is implemented through the probability of detection along with the false alarm rate, which are widely accepted as standard measures. Table II shows the cost matrix used for scoring entries.

TABLE II
THE COST MATRIX USED FOR SCORING ENTRIES

	Normal	Probe	DOS	U2R	R2L
Normal	0	1	2	2	2
Probe	1	0	2	2	2
DOS	2	1	0	2	2
U2R	3	2	2	0	2
R2L	4	2	2	2	0

In the confusion matrices above, columns correspond to predicted categories, while rows correspond to actual categories.

The software tool LNKnet, which is a publicly available pattern classification software package [12], was used to simulate pattern recognition and machine learning models. The LAMSTAR was simulated using JNNS [13] software tool.

E. Standard Metrics for Evaluations of Intrusions(Attacks)

We evaluated the performance of various IDS systems based on the **Detection Rate**: detecting normal traffic from attack and recognizing the known attack type **False Alarm Rate**: mis-detecting attack [14]. Table III shows the standard metrics for evaluation of Intrusions.

$$\text{Detection rate: } \frac{\text{Number of samples classified correctly}}{\text{Number of samples used for training}}$$

$$\text{False Alarm Rate: } \frac{\text{False Positives}}{\text{Total number of normal connections}}$$

TABLE III
STANDARD METRICS FOR EVALUATIONS OF INTRUSIONS (ATTACKS)

CONFUSION MATRIX (STANDARD METRICS)		PREDICTED CONNECTION LABEL	
		NORMAL	Intrusions (Attacks)
Actual Connection label	Normal	True Negative(TN)	False Alarm(FP)
	Intrusions (Attacks)	False Negative(FN)	Correctly detected Attacks (TP)

IV. PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) [15]-[17] is one of the most widely used dimensionality reduction techniques for data analysis and compression. It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimensions, PCA is a powerful tool for analyzing data. Once patterns in the data are found data can be compressed by reducing the number of dimensions without loss of information.

Given the KDDCup99 data, each data has 41 features represented by $x1_1\ x1_2 \ldots x1_{41}$, $x2_1\ x2_2 \ldots x2_{41}$ and so on. The data set can be represented by a matrix Xnxm.

The average observation is defined as

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{10}$$

The deviation from the average is defined as

$$\Phi_i = x_i - \mu \tag{11}$$

The sample covariance matrix of the data set is defined as

C

$$C = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T = \frac{1}{n}\sum_{i=1}^{n}\Phi_i\Phi_i^T = \frac{1}{n}A\,A^T \tag{12}$$

Eigen values and Eigen vectors of the sample covariance matrix C are usually computed by the Singular Value Decomposition. Suppose ($\lambda_{1,}$ u_1), ($\lambda_{2,}$ u_2)…. ($\lambda_{m,}$ u_m) are m eigenvalue-eigenvector pairs of the sample covariance matrix C. The k eigenvectors having the largest eigenvalues are selected. The dimensionality of the subspace k can be determined by

$$\frac{\sum_{i=1}^{k}\lambda i}{\sum_{i=1}^{m}\lambda i} \geq \alpha \tag{13}$$

where α is the ratio of the variation in the subspace to the total variation in the original space. A mxk matrix U is formed whose columns consists of the k eigenvectors. The representation of the data by principal components consist of projecting the data onto the k-dimensional subspace according to the following rules

$$Y_i = U^T(x_i - \mu) = U^T\Phi i \tag{14}$$

V. DATA REDUCTION AND PERFORMANCE OF VARIOUS ALGORITHMS

Principal component analysis is performed on the KDDCup 99 training and test data using Lnknet simulator and best 13 features were selected. The features with best eigen values are as shown in Table IV.

TABLE IV
BEST 13 FEATURES SELECTED AFTER PRINCIPAL COMPONENT ANALYSIS

S.no	Feature	Description
0	Duration	Continuous
1	Flag	Symbolic
2	src_bytes	Continuous
3	dst_bytes	Continuous
4	Land	Symbolic
5	wrong_fragment	Continuous
6	Urgent	Continuous
7	num_failed_logins	Continuous
8	logged_in	Continuous
9	dst_host_serror_rate	Continuous
10	dst_host_srv_serror_rate	Continuous
11	dst_host_rerror_rate	Continuous
12	dst_host_srv_rerror_rate	Continuous

VI. Intrusion Detection Performance of Various Algorithms using all Features and Reduced Features

A. Gaussian Mixture

The Gaussian Mixture classifier [18] can perform better than a Gaussian classifier when classifier distributions are not unimodal Gaussian. Different simulations were performed by changing various parameters like, each class has its own Gaussian mixture, all classes share a single set of tied Gaussian mixtures, diagonal covariance, full matrices covariance, separate variance for each Gaussian. The simulation result with parameters, each class has its own Gaussian mixture and diagonal covariance gives the better cost per example 0.2796 when 41 features were used and .2776 when 13 features were used. Table V shows the Confusion Matrix obtained for Gaussian mixture IDS for all features. Table VI shows the results for reduced features.

TABLE V
CONFUSION MATRIX FOR GAUSSIAN MIXTURE IDS (41 FEATURES)
CPE = .2796

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	59969	423	190	5	6	98.97
Probe	195	3876	95	0	0	93.03
DOS	18015	9002	202822	9	5	88.24
U2R	145	25	1	52	5	22.8
R2l	13950	650	5	30	1554	9.6
%correct	64.99	27.73	99.85	54.16	98.98	

TABLE VI
CONFUSION MATRIX FOR GAUSSIAN MIXTURE IDS (13 FEATURES)
CPE = .2776

Predicted / ACTUAL	NORMAL	PROBE	DOS	U2R	R2L	% correct
Normal	60019	383	180	5	6	99.05
Probe	195	3876	95	0	0	93.03
DOS	17945	8972	202922	9	5	88.28
U2R	155	25	1	42	5	18.42
R2l	13825	625	5	30	1704	10.5
%correct	65.13	27.92	99.86	48.83	99.06	

The top left entry in the confusion matrix shows that 59969 of the actual "normal" test examples were predicted to be normal by this entry. The last column indicates that in total 98.97% of the "normal" examples were recognized correctly. The bottom row shows that 64.99% of test examples said to be normal were indeed "normal" in reality. From the last column, we can obtain the average detect rate of 62.52%. The false positive rate for Normal class is 100-64.99 =35.01 %.

B. Radial Basis Function

Radial Basis Function classifiers [19] calculate discriminant functions using local Gaussian functions. A total of six simulations for 41 features and six simulations for 13 features were performed using the RBF algorithm. Each simulation used initial clusters created using K-means algorithm: there were 8,16,32,40,64 and 75 clusters each in different output classes. Weights are trained using least-square matrix inversion to minimize the squared error of the output sums given the basis function outputs for the training patterns. During training and testing variance are increased to provide good coverage of the data .For each simulation using the RBF, cost per example for the test dataset were calculated. The model with 64 clusters performed best with the cost per example equal to .3801 when 41 features were used and .3805 when 13 features were used. Table VII shows the Confusion Matrix obtained for Radial Basis IDS for all features. Table VIII shows the results for reduced features.

TABLE VII
CONFUSION MATRIX FOR RBF IDS(41 FEATURES)
CPE = .3801

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	% correct
Normal	60030	263	290	8	2	99.07
Probe	350	3804	8	3	1	91.31
DOS	37050	20163	172620	15	5	75.10
U2R	190	20	2	16	0	7.01
R2l	7282	7800	200	0	907	5.6
%correct	57.22	11.87	99.71	38.09	99.1	

TABLE VIII
CONFUSION MATRIX FOR RBF IDS (13 FEATURES)
CPE = .3805

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	59940	313	330	8	2	98.92
Probe	450	3704	8	3	1	88.91
DOS	36975	20116	172742	15	5	75.15
U2R	196	20	2	10	0	4.38
R2l	7294	7820	200	0	875	5.4
%correct	57.16	11.58	99.68	27.77	99.09	

C. Binary Tree

The binary decision tree classifier [20] trains and tests very quickly. It can also be used to identify the input features which are most important for classification because feature selection is part of the tree-building process. Two different training options were used 1. Expand tree until there are no errors. 2. Stop Expansion Early. Two different testing options were used: 1. Full tree for testing, 2. Maximum number of nodes during testing. The simulation with the parameters Expand tree until there are no errors for training and Full tree for testing gives the best cost per example .1841 when 41 features were used and .1837 when 13 features were used. Table IX shows the Confusion Matrix obtained for Binary IDS for all features. Table X shows the results for reduced features.

TABLE IX
CONFUSION MATRIX FOR BINARY TREE CLASSIFIER IDS
CPE=.1841

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	58430	1475	678	7	3	96.43
Probe	419	3247	492	6	2	77.94
DOS	7159	995	221694	4	1	96.45
U2R	97	99	1	31	0	13.59
R2l	8063	1000	7054	0	72	0.44
%correct	78.78	47.63	96.42	64.58	92.30	

TABLE X
CONFUSION MATRIX FOR BINARY TREE CLASSIFIER IDS
CPE=.1837

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	58683	1300	600	7	3	96.84
Probe	564	3102	492	6	2	74.45
DOS	7174	1003	221671	4	1	96.44
U2R	97	99	2	29	1	12.71
R2l	8063	1000	7054	0	72	0.44
%correct	78.68	47.69	96.45	64.04	91.13	

D. LAMSTAR

Using the LAMSTAR [21, 4, 5] algorithm, different clusters were specified and generated for each output class. Simulations were run having 2,4,8,16,32,40,64 clusters. Clusters were trained until the average squared error difference between two epochs was less than 1%. The cost per example for 41 features is .1027 and for reduced features is .1030. Table XI shows the Confusion Matrix obtained for Lamstar IDS for all features. Table XII shows the results for reduced features.

TABLE XI
CONFUSION MATRIX FOR LAMSTAR IDS
CPE=.1027

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	60411	140	37	4	1	99.69
Probe	56	4103	6	0	1	98.48
DOS	1603	186	228060	3	1	99.21
U2R	99	54	8	66	1	28.94
R2L	7519	985	1015	0	6670	41.20
%correct	86.68	75.03	99.53	90.04	99.94	

TABLE XII
CONFUSION MATRIX FOR LAMSTAR IDS
CPE=.1030

Predicted / Actual	Normal	Probe	DOS	U2R	R2L	%correct
Normal	60411	140	37	4	1	99.69
Probe	42	4118	5	0	1	98.84
DOS	1688	146	228015	3	1	99.20
U2R	99	54	5	69	1	30.26
R2L	7519	985	1020	0	6665	41.16
%correct	86.68	75.03	99.53	90.04	99.94	

VII. EXPERIMENTAL RESULTS AND COMPARISONS

Best performing instances of all classifiers, developed through the KDDCup99 data set [22]. For a given classifier, its detection rate, false alarm rate, Training time and Testing time, performance on a specific attack category for full features and reduced features using principal component analysis were recorded. Simulation results are presented in Table XIII and XIV. Detection rate, false alarm rate, training time and testing time are indicated for each classifier and each attack category. From the results it can be seen that the training time and testing time significantly reduces when reduced features were used keeping the detection rate and false alarm rate almost the same. Performance of LAMSTAR IDS for all the five categories of data is significant compared to other classifiers. The comparison charts are shown in Fig. 2 to Fig. 9.

TABLE XIII
COMPARISON OF DETECTION RATE, FALSE ALARM RATE, TRAINING TIME AND TESTING TIME OF VARIOUS CLASSIFIERS (41 FEATURES)

	Cost Per Example		Normal	Probe	DOS	U2R	R2L
Gmix	0.2796	DR	98.97	93.03	88.24	22.8	9.6
		FAR	35.01	72.27	0.15	45.84	1.02
		Training Time	40s	15s	60s	5s	10
		Testing Time	28s	10s	45s	5s	12s
RBF	0.3801	DR	99.07	91.31	75.10	7.01	5.6
		FAR	42.78	88.13	0.29	61.91	0.88
		Training Time	41s	14s	55s	5s	11s
		Testing Time	31s	10s	40s	5s	9s
BINARYTree	0.1841	DR	96.43	77.94	96.45	13.59	0.44
		FAR	21.22	52.37	3.58	35.42	7.70
		Training Time	39s	14s	53s	6s	11s
		Testing Time	30s	12s	29s	6s	9s
LAMSTAR	0.1027	DR	99.69	98.48	99.21	28.94	41.20
		FAR	13.32	24.97	0.47	9.96	0.06
		Training Time	47s	16s	60s	6s	15s
		Testing Time	28s	13s	28s	5s	9s

TABLE XIV
COMPARISON OF DETECTION RATE, FALSE ALARM RATE, TRAINING TIME AND TESTING TIME OF VARIOUS CLASSIFIERS (13 FEATURES)

	Cost Per Example		Normal	Probe	DOS	U2R	R2L
GMIx	0.2776	DR	99.05	93.03	88.28	18.42	10.50
		FAR	34.87	72.08	0.14	51.17	0.94
		Training Time	30s	12s	45s	4s	9s
		Testing Time	23s	8s	36s	4s	10s
RBF	0.3805	DR	98.92	88.91	75.15	4.38	5.40
		FAR	42.84	88.42	0.32	72.23	0.91
		Training Time	31s	11s	42s	5s	9s
		Testing Time	25s	8s	33s	5s	8s
BINARYTREE	0.1837	DR	96.84	74.45	96.44	12.71	0.44
		FAR	21.32	52.31	3.55	35.96	8.87
		Training Time	29s	11s	40s	5s	9s
		Testing Time	23s	10s	22s	5s	8s
LAMSTAR	0.1030	DR	99.69	98.84	99.20	30.26	41.16
		FAR	13.32	24.97	0.47	9.96	0.06
		Training Time	36s	12s	46s	6s	12s
		Testing Time	23s	11s	22s	5s	8s

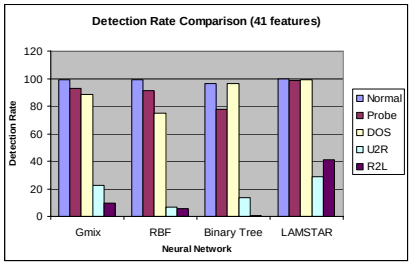

Fig. 2 Detection Rate Comparison (41 features)

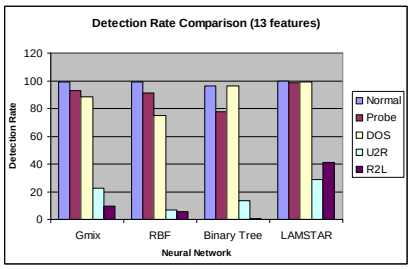

Fig. 3 Detection Rate Comparison (13 features)

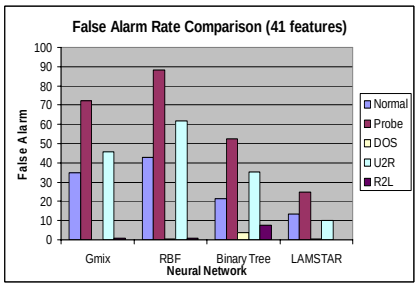

Fig. 4 False Alarm Rate Comparison (41 features)

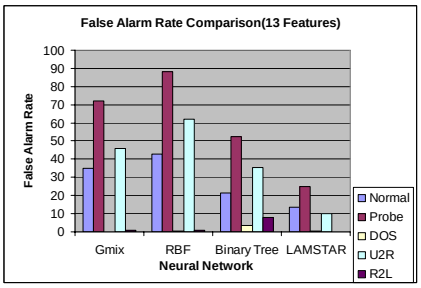

Fig. 5 False Alarm Rate Comparison (13 features)

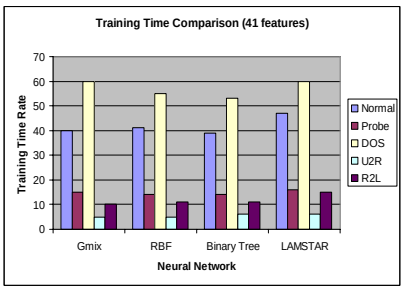

Fig. 6 Training Time Comparison (41 features)

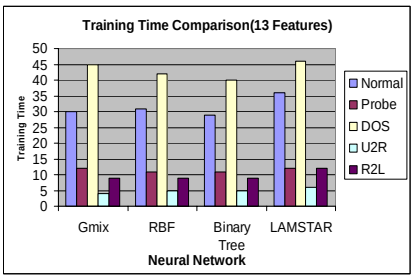

Fig. 7 Training Time Comparison (13 features)

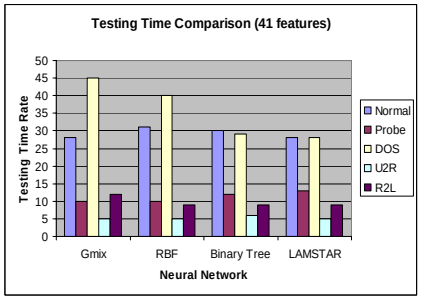

Fig. 8 Testing Time Comparison (41 features)

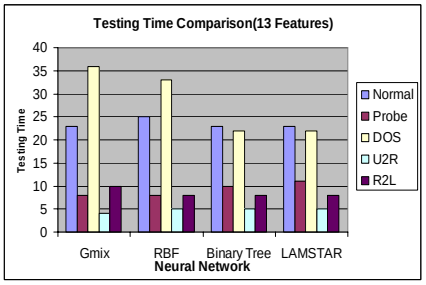

Fig. 9 Testing Time Comparison (13 features)

VIII. Conclusion

A novel approach for detecting network intrusions using LAMSTAR Neural Network is proposed in this paper. The performance of LAMSTAR IDS evaluated using KDDCup99 data and compared with three other classifiers. Simulation results demonstrated that all the algorithms performed well for NORMAL, DOS and PROBE classes except Binary Tree, which shows poor result for PROBE class. For the U2R and R2L class LAMSTAR gives a better performance than the other algorithms. The performance of LAMSTAR IDS is obtained at the cost of high training and testing time due to computational complexity. To reduce the computational complexity, training and testing time, Principal Component Analysis was applied to KDDCup99 data and 13 important features were selected out of 41 features in the KDDCup99 data. Experimental results with 13 features show significant reduction in training and testing time due to the reduction in computation, while keeping the detection rate and false alarm rate almost the same.

References

[1] A.K.Ghosh, A.Schwartzbard, "Study in Using Neural Networks for Anomaly and Misuse Detection", in Proc. *8th USENIX Security Symposium*, pp 131-142, August 1999, Washington, D.C.

[2] Abirami Muralidharan, J.Patrick Rousche, "Decoding of auditory cortex signals with a LAMSTAR neural network", *Neurological Research*, Volume 27, pp. 4-10, January 2005.

[3] D.Graupe and H. Kordylewski, "A Large Memory Storage and Retrieval Neural Network for Adaptive Retrieval and Diagnosis", *International Journal of Software Engineering and Knowledge Engineering*, volume 8, pp.115-138, 1998.

[4] D.Graupe, "Principles of Artificial Neural Networks", pp. 191-222, *World Scientific Publishing Co. Pte. Ltd*., Singapore, 1997.

[5] H. Kordylewski, "A Large Memory Storage and Retrieval Neural Network for Medical and Engineering Diagnosis/Fault Detection", *Doctor of Philosophy's Thesis*, University of Illinois at Chicago, TK-99999-K629, 1998.

[6] D.Graupe and H. Kordylewski, "A large scale memory (LAMSTAR) neural network for medical diagnosis", in Proc. *19th Annual International Conference of the IEEE*, Volume 3, Issue 30, Oct-2 Nov 1997 Page(s):1332 – 1335.

[7] S.K.Chang, D.Graupe, K.Hasegawa, H.Kordylewski, "An Active Multimedia Information System for Information Retrieval, Discovery and Fusion", *International Journal of Software Engineering and Knowledge Engineering*, volume 8, pp. 139-160, 1998.

[8] http://kdd.ics.uci.edu//databases/kddcup99/kddcup99.html

[9] Teuvo Kohonen , "The Self Organizing Map", in Proc. *IEEE*, Volume 78, No. 9, pp 1464 – 1480, September 1990.

[10] Srilatha Chebrolu, Ajith Abraham, Johnson P.Thomas, "Feature deduction and ensemble design of intrusion detection systems", *Elsevier Journal of Computers & Security*" Vol. 24/4, pp. 295-307, 2005.

[11] Itzhak Levin, KDD-99 Classifier Learning Contest LLSoft's Results Overview, "*SIGKDD Explorations. Copyright* 2000 *ACM SIGKDD"*, Vol. 1, Issue 2, pp. 67 -75, January 2000.

[12] www.ll.mit.edu/SST/lnknet/

[13] www-ra.informatik.uni-tuebingen.de/ software/ JavaNNS/ welcome_e. html.

[14] Dae-Ki Kang, "Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation", in Proc. *6th IEEE Workshop on Information Assurance and Security* United States Military Academy, West Point, NY, 2005.

[15] D. Nguyen, A. Das, G. Memik, and A. Choudhary , "Reconfigurable Architecture for Network Intrusion Detection Using Principal Component Analysis" In Proc. *ACM/SIGDA 14th international symposium on Field programmable gate arrays* , pp. 235 – 235, 2006.

[16] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier", In Proc. *IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM'03)*, pp 172–179, Nov. 2003.

[17] I. T. Jolliffe, "Principal Component Analysis", *Springer Verlag*, New York, NY, third edition, July 2002.

[18] Jing Gao, Haibin Cheng, Pang Ming Tan, "A Novel Framework for Incorporating Labeled Examples into Anomaly Detection", in Proc. of the *Siam Conference on Data Mining*, April 2006.

[19] Dima Novikov, Roman V. Yampolskiy, Leon Reznik, "Anomaly Detection Based Intrusion Detection" in Proc. of the *Third IEEE International Conference on Information Technology: New Generations (ITNG'06)*, pp. 420-425, 2005.

[20] Richard Lippmann, **"**Passive Operating System Identification From TCP/IP Packet Headers" in Proc. of the *Workshop on Data Mining for Computer Security (DMSEC)*, Lincoln Laboratory ,Massachusetts, 2003.

[21] Liberios Vokorokos, Anton Baley, Martin Chovenac, "Intrusion detection system using self organizing map", *Acta Electrotechnica et Informatica* , Vol. 6 No.1, pp.1-6, 2006.

[22] Chaker Katar, "Combining Multiple Techniques for Intrusion Detection", *International Journal of Computer Science and Network Security*, Vol. 6 No.2B, February 2006.

Dr. S. Selvan is Professor and Head, Department of Information Technology at PSG College of Technology, Coimbatore, India. He has 27 years of teaching experience. He has published more than 60 papers in international and national journals and conference proceedings. His areas of research include digital image processing, soft computing, digital signal processing and computer networking.

V. Venkatachalam received the B.E. degree in Electronics and Communication from Bharathiyar University and M.S. degree in software systems from Birla Institute of Technology. He received M. Tech. Degree in Computer Science from National Institute of Technology. His Research interest includes Network Security and Pattern recognition. He is currently pursuing his PhD degree in Network Security. Presently working as Head of the Dept. CSE in Erode Sengunthar Engineering College, Erode.