A Principal Components Analysis Neural Gas Algorithm for Anomalies Clustering

Xiufen Fang[1], Guisong Liu[2], Ting-Zhu Huang[1]
1. School of Applied Mathematics/Institute of Computational Science
2. School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan, 610054, P. R. China
xffang111@sina.com.cn, guisongliu@gmail.com, tingzhuhuang@126.com

Abstract: - Neural gas network is a single-layered soft competitive neural network, which can be applied to clustering analysis with fast convergent speed comparing to Self-organizing Map (SOM), K-means etc. Combining neural gas with principal component analysis, this paper proposes a new clustering method, namely principal components analysis neural gas (PCA-NG), and the online learning algorithm is also given. The soft competitive learning of PCA-NG is based on local principal subspace, which characterizes the profile of a certain cluster. We utilize the PCA-NG to the domain of intrusion detection. Some experiments are carried out to illustrate the performance of the proposed approach by using a synthetic Gaussian-distributed dataset and the KDD CUP 1999 Intrusion Detection Evaluation dataset.

Key-Words: - Intrusion Detection, Neural Gas Network, Principal Component Analysis, Cluster Analysis

1 Introduction

With the increase of network attacks in number and severity over the past few years, intrusion detection systems (IDSs) have became a necessary addition to the security infrastructure of most organizations. Intrusion detection technique is one of the most crucial methodologies in network security, which can guarantee the security in a certain degree of a system, a network or a user, behind the first line of defense (e.g., fire-wall). Bace et al. [1] gave a definition of intrusion detection systems as "software or hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems".

In a network-based IDS, by analyzing the network stream data, e.g. IP packets, a norm profile (normal activity profile) can be constructed from those normal behavior exhibited by either a user or a system. Any behavior with a certain degree deviation from the norm profile is determined as an intrusion, or an attack. This is called anomaly detection. Another intrusion detection technique is misuse detection, also called signature-based detection, which attempts to model attacks (abnormal connections in network-based IDS) on a system as specific patterns, then systematically scans the system for occurrences of these patterns [2]. Artificial neural networks (ANNs) resemble a powerful tool to separate clusters of feature vectors in a high-dimensional space. The goal of clustering is grouping given training data into classes of similar objects such that data points with similar semantical meaning are linked together. The most popular method to clustering analysis is the self-organizing feature map (SOM) proposed by Kohonen [3]. SOM is an unsupervised neural network which can transform an incoming signal pattern of arbitrary dimension into a one- or two-dimensional discrete map, and perform this transformation adaptively in a topologically ordered fashion [4]. SOM has been applied to the domain of intrusion detection for many years for different purposes [5-10]. But the limitations of SOM network will yield a big impact to the performance of intrusion detection, such as the static architecture and limited capabilities of representation of hierarchical relations of the input data [11].

As an alternative way for clustering, Martinetz et al. proposed the Neural Gas (NG) algorithm in 1993 [12], as a fast neural net-based clustering method, and it has been successfully applied to vector quantization, prediction and topology representation, etc. The name Neural Gas is just because the centers of the clusters move around in the data space similar to the *Brownian Movement* of gas molecules in a closed container. In [12], the authors gave an in-depth discussion of the *Gas-like* dynamics of the neurons and how this tends to result in a homogeneous distribution of the cluster centers over the input space. In [13], the Neural-Gas network was used to gas chromatographic patterns of Maillard reaction products. A robust clustering algorithm was presented within the growing neural

gas (GNG) framework in [14], which was insensitive to initialization, input sequence ordering and the presence of outliers. In [15], an extension of the neural gas to local principal component analysis was proposed and applied to the recognition of handwritten digits. In [16], based on the cost function of NG, a batch variant of NG was proposed which shown much faster convergence and which can be interpreted as an optimization of the cost function by the Newton method compared with SOM and k-means in a unified formulation.

In this paper, we combine the neural gas with principal component analysis, namely PCA-NG. The learning algorithm is different from Ref. [15], and we apply it to the domain of intrusion detection for clustering analysis. It is a novel method for intrusion detection, from the author's knowledge.

The rest of this paper is organized as follows. Some basic knowledge of Euclidean distance measure based neural gas algorithm is described in Section 2. In Section 3, the proposed principal component neural gas (PCA-NG) is given. In Section 4, two experiments are carried out; the first one is performed on a synthetic 2-D Gaussian-distributed dataset, and the other on the KDD CUP 1999 intrusion detection datasets. Finally, conclusion is given in Section 5.

2 Euclidean Distance Based Neural Gas Algorithm

Assume that $\{x(t)\}$ $t = 1, 2, \cdots l$, are n-dimensional stochastic input data, the mean vector e and the covariance matrix of $x(t)$ are defined by

$$e = \frac{1}{l}\sum_{t=1}^{l} x(t), \tag{1}$$

$$R = \frac{1}{l-1}\sum_{t=1}^{l}[(x(t)-e)(x(t)-e)^T]. \tag{2}$$

A traditional neural gas training algorithm with Euclidean distance measure can be stated as follows:
1) Initialize the neural gas network. Obtain the following inputs from the user:

– Number of predefined neurons (clusters), namely the number of Clusters c.

– Randomly initialized weigh vectors over the input space, $W = [w_1, w_2, \cdots w_c]$.

– Initial learning rate η_0 and final learning rate η_{end}, e.g., $\eta_0 = 0$ and $\eta_{end} = 0.001$.

– Number of training set N ,and the maximum training epoch Ep with $Ep_0 = 0$ and $Ep_{\max} = M$.

– Maximum number of iterations $t_{\max} = MN$, initial and final decay constants, λ_0 and λ_{end} (e.g.,10 and 0.001).

2) Input a sequential vector $x(t)$ at time instant t in m^{th} training epoch. The total training iteration step is

$$t_{iter} = Ep * N + t. \tag{3}$$

3) Calculate the distance (e.g., Euclidean distance) between $x(t)$ and w_i as

$$d_i = \|x(t) - w_i\|, i = 1, 2, \cdots, numClusters. \tag{4}$$

4) Calculate the neighbourhood ranking r_i (initial $r_i = 0, i = 1, 2, \cdots, c$) as follow (for $i = 1, 2, \cdots, numClusters$ and $j = i, \cdots, numClusters$):

$$r_i = r_i + \begin{cases} 1, d_i \geq d_j \\ 0, otherwise \end{cases} \tag{5}$$

The number r_i, associated with each w_i, denotes the order obtained due to the above sorting procedure.

5) Update the weight vectors w_i as

$$w_{i+1} = w_i + \eta(t) h_\lambda(r_i)(x(t) - w_i). \tag{6}$$

The neighbourhood function is,

$$h_\lambda(r_i) = e^{-\frac{r_i}{\lambda(t)}}, \tag{7}$$

where the learning rate $\eta(t)$ and decay constant $\lambda(t)$ are calculated as follows,

$$\eta(t) = \eta_0 (\frac{\eta_{end}}{\eta_0})^{\frac{t}{t\max}}, \tag{8}$$

$$\lambda(t) = \lambda_0 (\frac{\lambda_{end}}{\lambda_0})^{\frac{t}{t\max}}. \tag{9}$$

6) Increase t to t+1, repeat step 2 to step 6 until $t = t_{\max}$.

7) In the last training epoch, label the input $x(t)$ as one of the stabilized clusters according to the following criteria,

$$C_j = \arg\min_j \{r_j\}, j = 1, 2, \cdots c. \tag{10}$$

Note that the winning neuron C_j with minimum neighbourhood ranking r is the Best-Matching-Unit

(BMU) in the competitive process in a Euclidean sense.

The neural gas is a kind of single-layered soft competitive learning neural network with many advantages, including [12]: (1) faster convergence to low distortion errors, (2) lower distortion error than that of resulting from *k*-means clustering, maximum-entropy clustering and Kohonen's self-organizing map algorithm, (3) obeying a stochastic gradient descent on an explicit energy surface. Neural Gas (NG) constitutes a very robust clustering algorithm given Euclidean data which does not suffer from the problem of local minima like simple vector quantization, or topological restrictions like the self-organizing map [16].

3 Neural Gas Combined with Principal Components Analysis

In a traditional Euclidean-distance-based neural gas algorithm, the weight vector of one neuron represents the mean vector of one cluster in a high dimensional space. But it is not enough to use mean vector merely to match well for some cases of data distribution, such as Gaussian-based distributed data clusters. Principal component analysis (PCA) [17] can be used to tackle this problem in a subspace-decomposition sense [4]. PCA has been widely used in data compression and feature selection (extraction). Feature selection (extraction) refers to a process whereby a data space is transformed into a feature space, which has exactly the same (or reduced) dimension as the original data space. Suppose *R* in equation (2) is the covariance matrix of input vector $x(t)$. By sorting the eigenvalues λ_i of *R* in descending order, we obtain *m* eigenvectors, also called principal directions, denoted by $B^i = \{B^i_j \mid j = 1, 2, \cdots, m\}$ corresponding to those *m* largest eigenvalues (usually *m*<*n*) for feature extraction.

It is well known that the variance of the projections of the input data onto the principal direction is greater than that of any other directions. Giving a new input $x(t)$, we can project it onto the principal space that is composed of the first *m* principal directions.

As shown in Fig. 1, there are tow Gaussian-based clusters, namely C_1 and C_2, and each mean vector (the centroid of a cluster) is denoted by a small solid circle. The two lines with arrows crossed the data represent the principal directions of the two Gaussian-based clusters (only one principal direction selected in a 2-D plane). In a Euclidean sense, the separation line is vertical to the line which joints the centroids of the two clusters and passes through its midpoint. Obviously, there exist data samples in cluster C_1 are misclassified into C_2 by the Euclidean-metric-based separation line (the bold dashed line). But the separation line in a local principal subspace decomposition sense (the bold solid line) can match this Gaussian-based distribution data clusters well and can separate them distinctly (see Fig. 1).

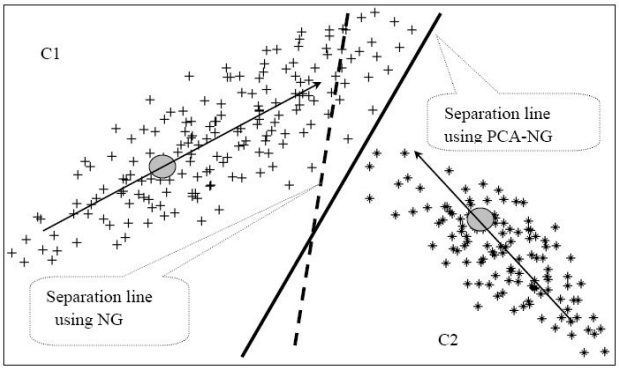

Fig.1 Different separation lines representation by using NG and PCA-NG.

Therefore, combining neural gas algorithm with PCA, the belongingness of an input vector $x(t)$ to neuron *i* can be denoted by a new parameter set as follows:

$$C_i \leftarrow \{r_i, e_i, \{B^i_j \mid j = 1, 2, \cdots, m\}\}, \quad (11)$$

where e_i is the mean vector of cluster *i* at time instant *t*, and $B^i_j, j = 1, 2, \cdots, m$ are basis vectors of the local principal subspace (m is the number of principal directions used in a local cluster, usually m is less than n.).

Correspondingly, the projection of $x(t)$ on the local principal subspace, is written as

$$p_i = \sum_{h=1}^{K} B_h^{iT}(x(t) - e_i) B_h^i \ , \quad (12)$$

and the distance calculation between $x(t)$ and neuron *i is* as follows,

$$d_i = \left\| x(t) - e_i - p_i \right\|. \quad (13)$$

This can be illustrated by Fig. 2 in a 2-D plane, where only one principal direction is selected, B^i_1. The projection p_i is the approximation of x_i and d_i is the distance error. Different from $\| x(t) - e_i \|$ in the Euclidean distance sense, the d_i calculated

by different neurons in PCA-NG network is the criteria to determine which clusters $x(t)$ belongs to. It can be written as,

$$C = \arg\min_i \left(\| x(t) - e_i - \sum_{h=1}^{K} B_h^{iT}(x(t) - e_i) B_h^i \| \right) \quad (14)$$

By sorting the distance $d_i, i = 1, 2, \cdots, numclusters$, different neighbor ranking r_i is calculated using equation (5). The parameters sets updating of the neurons include mean vector e_i and the basis vectors of the local principal subspace B_j^i.

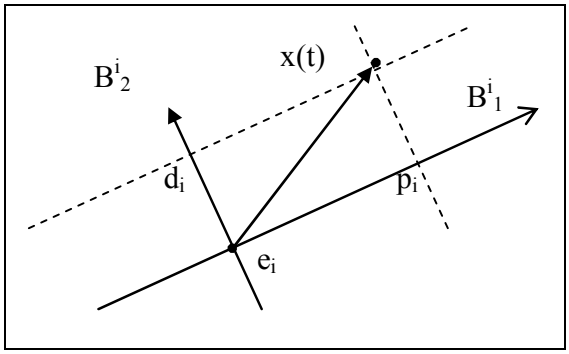

Fig.2 Illustration of local principal subspace, projection of the input, distance error of the input in a 2-D plane.

PCA has two main properties. First, it finds the uncorrelated directions of maximum variance in the data space, and second, it provides the optimal linear projection in the least square sense. There exist many online PCA methods [18-20]. In [15], an extension of neural gas to local PCA was implemented by using RRLSA [2], a neural method for principal component analysis based on the recursive lease-squares method. In our PCA-NG algorithm, an online estimation of covariance matrix is performed to store the information of the basis vectors of the principal subspace. Hence, the updating of mean vector e_i and covariance matrix R_i can be stated as follows,

$$e_{i+1} = e_i + \eta(t) h_\lambda(r_i)[x(t) - e_i], \quad (15)$$

$$R_{i+1} = R_i + \eta_t h_\lambda(r_i)[(x(t) - e_i)(x(t) - e_i)^T - R_i] \quad (16)$$

By using the eigen-decomposition of covariance matrix R, the principal eigenvectors can be obtained. Generally, in a high-dimensional input space, the determination of the number of principal basis vectors, denoted by K is based on the following criteria,

$$K = \min_i (i \mid \sum_{j=1}^{i} \lambda_j \geq \alpha \sum_{j=1}^{n} \lambda_j). \quad (17)$$

Where λ_i represents the corresponding eigenvalues (in a descending order) of covariance matrix R, and α is the proportion factor usually more than 90%.

4 Experiments

4.1 Synthetic Data

We construct several 2-D synthetic datasets to test our PCA-NG algorithm. The datasets include five clusters and each cluster contains 40 data samples. Every cluster is apparently Gaussian-based distribution. The initialization of the PCA-NG algorithm is as follows,

(1) Neuron number is 5.

(2) The mean vectors are chosen randomly.

(3) Only one subspace principal component is chosen with B_i randomly selected.

(4) Learning rates η_0 and η_{end} is 0.8 and 0.008, respectively.

(5) The initial and final decay constants, λ_0 and λ_{end}, are 10 and 0.01.

(6) The training epoch is 50.

We input all the data samples to the PCA-NG network randomly. Figure 3 shows the updating process of the parameter sets of the five neurons. Different mean vectors and principal directions are denoted with different symbols and arrow lines, respectively. The first sub-figure shows the initialization of the parameter sets and the other five sub-figures demonstrate the updating process in different training epoch, namely 1, 3, 5, 10, 50, respectively. In order to test the robustness of the PCA-NG algorithm, we perform the experiments 20 times with different initialization and different random input sequence of the training data, and Fig. 2 shows the average performance state. It shows that the mean vectors e_i (I from one to five) of every neuron converge to the centroid of every data cluster, and B_i^1 (one principal direction in a 2-D plane) converge to the principal eigenvectors of each Gaussian-distribution dataset at the same time.

As for the convergence speed of the PCA-NG, figure 4 shows the error decreasing process in different training epoch. Only mean vector errors of the five neurons (distances calculated by Euclidean norm) are illustrated in the figure. Obviously, each of the five neurons converges to its stable state after only 10 epoch training.

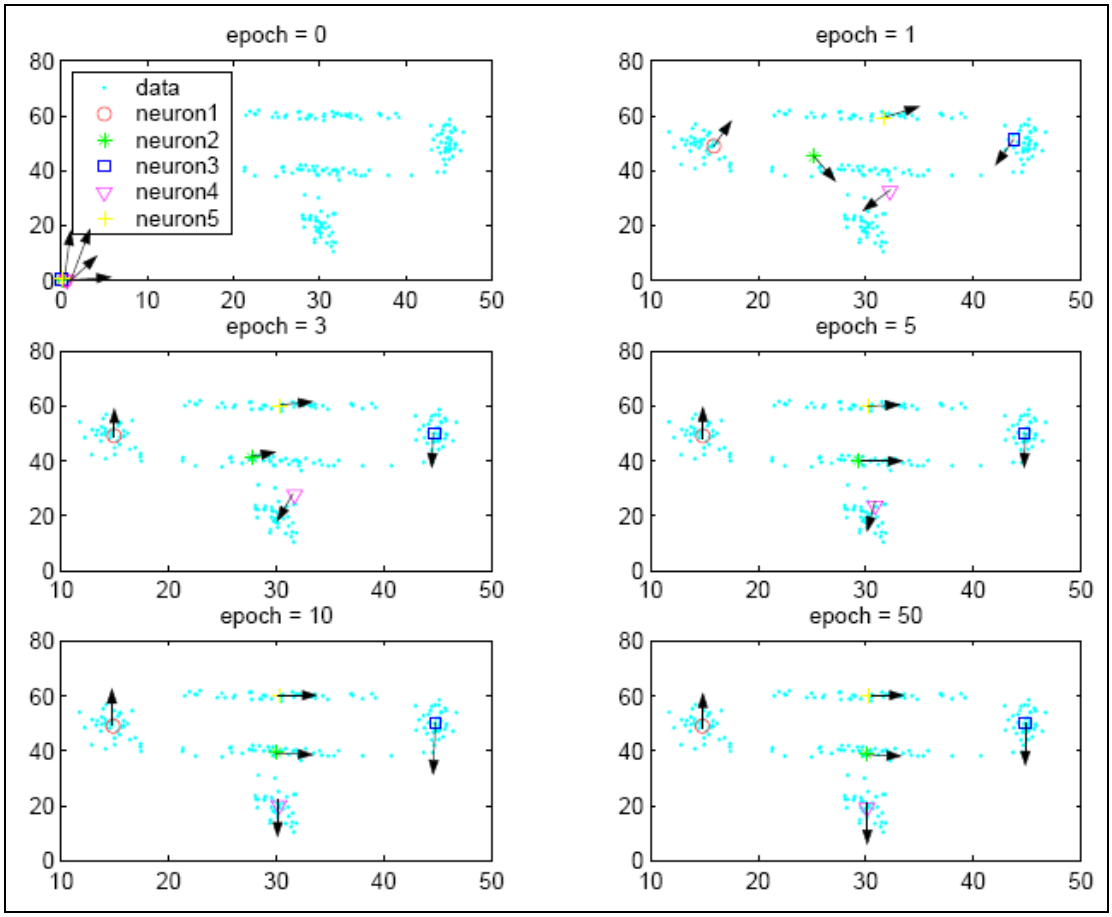

Fig.3 PCA-NG cluster analysis using synthetic 2-D Gaussian-distributed datasets. Six sub-figures with different titles represent different network states in different training epoch.

.

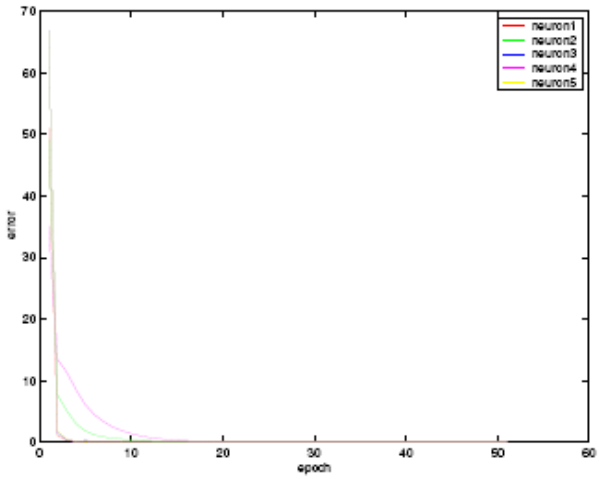

Fig 4. Convergence speed illustration in cluster analysis using PCA-NG on synthetic 2-d Gaussian-distributed datasets.

4.2 KDD CUP 1999 Intrusion Detection Evaluation Datasets

4.2.1 About the Datasets

The DARPA 1998 and 1999 Intrusion Detection Evaluations consist of comprehensive technical evaluations of research intrusion detection systems [22]. The 1998 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. Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records. The 1999 KDD intrusion detection contest uses a version of this dataset. The

KDD Cup 1999 dataset, which is a subversion of DARPA project, includes *good* normal connections and *bad* intrusion connections. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only. All the attacks fall in four main categories, such as DoS, R2L, U2R and Probe [23], which are stated as follows:

(1) DoS: denial-of-service, e.g. syn flood,
(2) R2L: unauthorized access from a remote machine, e.g. guessing password,
(3) U2R: unauthorized access to local super user (root) privileges, e.g., various ``buffer overflow'' attacks,
(4) Probing: surveillance and other probing, e.g., port scanning.

4.2.2 Data Preprocessing

Every network connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection includes forty-one feature values. The detailed description of the available features and intrusion instances can be found in [24]. The attributes in each connection of the KDD datasets has some forms, namely continuous, discrete and symbolic with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence preprocessing was required before pattern classification models could be built. For a symbolic type attribute, we first order them with a sequence number from 0 to n-1, where n is the specific class number of the attribute. Then we linearly map them to [0, 1]. For the discrete type attributes, e.g., land, with value 0 or 1, they do not require any preprocessing. The results of preprocessing of some attributes in our experiments are shown in Table 1.

4.2.3 How to Label the Clusters

The trained PCA-NG can be used as a detector or a classifier after each cluster is labeled. A simple scheme is used to label a trained cluster. Assume there is Tn testing observations which are categorized for k classes. n_i denotes the number of observations belonging to testing class i. Input one class (e.g., the label is l_i) of observations to the trained PCA-NG, denote the number of observations tested for one cluster j as $\overline{n}_j$. The label l_i will be assigned to the cluster which obtains the maximum testing observations, this is stated as,

$$l_i \to \max_i \{\overline{n}_i\}, \tag{18}$$

and the detection rate of l_i is

$$DetectionRate = \frac{\overline{n}_i}{n_i}. \tag{19}$$

If $\overline{n}_i$ is equal to n_i, we call the cluster l_i a homogeneous cluster with 100% detection rate. The training, labeling and testing processes for intrusive connections clustering using PCA-NG Algorithm are illustrated by Fig. 5.

Feature Name	**Data Type**	**Range**	**Preprocessing**
protocol-type	symbolic	3	0~2→ [0,1]
service	symbolic	70	0~69→ [0,1]
flag	Symbolic	11	0~11→ [0,1]
duration	continuous	[0,58329]	[0,1]
wrong-fragment	continuous	[0,3]	[0,1]
urgent	continuous	[0,14]	[0,1]
hot	continuous	[0,101]	[0,1]
num-failed-logins	continuous	[0,5]	[0,1]
num-compromised	continuous	[0,9]	[0,1]
su-attempted	continuous	[0,2]	[0,1]
num-root	continuous	[0,7468]	[0,1]
num-file creation	continuous	[0,100]	[0,1]
num-shells	continuous	[0,5]	[0,1]
num-access-files	continuous	[0,9]	[0,1]
count	continuous	[0,511]	[0,1]
srv-count	continuous	[0,511]	[0,1]
dst-host-count	continuous	[0,255]	[0,1]
dst-host-srv-count	continuous	[0,255]	[0,1]

Table 1: Preprocessing of some attributes from the raw datasets of each network connections.

4.2.4 Clustering Results Using PCA-NG

The training data are chosen from a 10% subset (*kddcup.data_10_percent.gz*) randomly and the testing data are from the labeled dataset (*corrected.gz*). We merge the testing and labeling two processes into single testing process. The first four columns of Table 2 show the data preparation for training and testing. Remind all the individual attack except for *normal* type connections belongs to the four main categories, such as DoS, Probe, U2R and R2L.

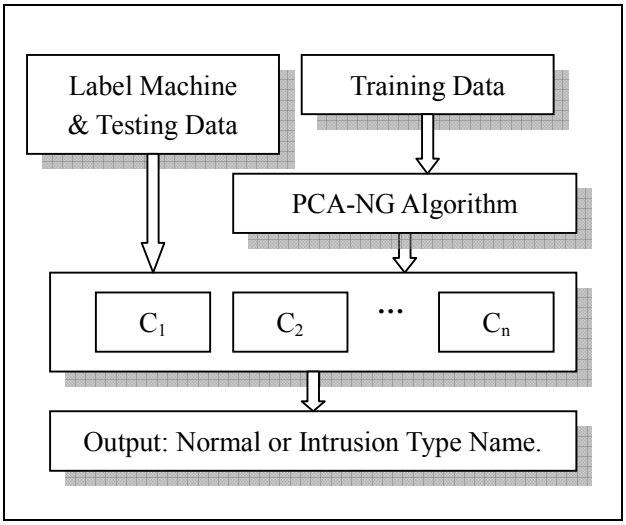

Fig 5. Training, testing and labeling processes for intrusion detecting using PCA-NG Algorithm.

In the training stage, we select the neuron number of PCA-NG model as 14 and hope to cluster all the training dataset to proper number of clusters. The PCA-NG initializations of all the other parameters are same to the first experiments before (on Synthetic Datasets). We input the training data samples into PCA-NG model in random order with 10 training epochs, and then test it in same manner with testing dataset. The performance calculated using equation (19) is listed in the last column of Table 2.

Obviously, the number of training data will influence the performance of PCA-NG. Generally, a larger amount of training data for one cluster will obtain better detection rate (see Table 2, e.g., more than 94% detection rate achieved for normal, back, smurf, neptune, satan, etc.). On the other hand, the detection rate is lower for that with less training data, e.g., rootkit, buffer-overflow, etc.

Furthermore, we set the neuron number as 5 and expect to cluster all the training dataset to five main categories, namely Normal, DoS, Probe, U2R and R2L. At the same time, from Table 2, we can compute the average detection rate of the five main categories. We denote the average detection rate of one main category j as ADR^j, and DR_i are the detection rate of one individual attack belongs to the main category j, n_i is its testing data number. Hence, the average detection rate of one main category can be calculated as follows,

$$ADR^j = \frac{\sum DR_i n_i}{\sum n_i}, j = 1, 2, \cdots, 5 . \qquad (20)$$

Attack	Category (#Train/#Test)	Detection Rate
normal	Normal (2500/3200)	97.2%
back	DoS (1200/890)	98.9%
smurf	DoS (3488/5680)	95.2%
neptune	DoS (4500/3500)	94.3%
teardrop	DoS (748/112)	80.4%
ipsweep	Probe (1200/140)	78.6%
nmap	Probe (123/26)	96.2%
portsweep	Probe (880/220)	96.4%
satan	Probe (1023/1500)	93.8%
rootkit	U2R (10/12)	66.7%
buffer-overflow	U2R (30/22)	63.6%
guess-passwd	R2L (1210/255)	82.4%
imap	R2L (119/200)	70.0%
waerzclient	R2L (350/55)	60.0%

Table 2: Data preparation from KDD CUP 1999 for training and testing of PCA-NG are listed in the first four columns; the last column is detection rate using clustering analysis. (DR means Detection Rate.)

Table 3 shows the performance comparison of PCA-NG model by using different neurons number on the five main categories. Clearly, the detection rates are higher by using 14 neurons than that by using 5 neurons. The reason is that more neurons can represent the input connections more accurately. The individual attacks, e.g., back and neptune, belong to the category DoS, but they are not necessarily close to the cluster DoS completely. On the other hand, there are many other type connections falling into the big cluster, and this will result in misclassifications and lower detection rate.

Table 3 also shows that PCA-NG can detect 97.2% normal connections; the 2.8% of them are misclassified as intrusions; it means the false positive rate of PCA-NG for intrusion detection is only 2.8%. It is known that the false positive rate is more important than detection rate in an actual intrusion detection system.

4.2.5 Related work and Comparisons with Other Approaches

In the intrusion detection field, the widely used clustering methods include K-means, SOM and its variations etc.

Ref. [10] designed an Anomalous Network-Traffic Detection with Self Organizing Maps (ANDSOM) module to detect anomalous network traffic based on the Self-Organizing Map algorithm. Each network connection is characterized by six parameters and specified as a six-dimensional vector. The ANDSOM module creates a Self-Organizing Map (SOM) having a two-dimensional lattice of neurons for each network service. During the training phase, normal network traffic is fed to the ANDSOM module, and the neurons in the SOM are trained to capture its characteristic patterns. During real-time operation, a network connection is fed to its respective SOM, and a "winner" is selected by finding the neuron that is closest in distance to it. The network connection is then classified as an intrusion if this distance is more than a pre-set threshold.

Type	Detection Rates	
	#neurons (5)	#neurons (14)
Normal	91.5%	97.2%
DOS	88.6%	95.1%
Probe	85.9%	93.0%
U2R	58.4%	64.7%
R2L	40.8%	75.1%

Table 3: The performance of PCA-NG on main categories using different neurons number

Sarasamma et al. used a multilevel hierarchical Kohonen network to detect anomalous events in network data [8]. A cost effective hierarchical extension of the simple Kohonen network was used to detect maximal number of attack types at low false positive rates. They also evaluated the effects of various feature subsets on the detection rate and false-positive rate. Another motivation of the work was to achieve a high-order nonlinear classifier model to create clusters that model the intersection of hyper cylinders in a computationally efficient way. Using a three-level hierarchical Kohonen net, they achieved detection rates between 90.94% and 93.46% at false-positive rates between 2.19% and 3.99% for three feature combinations. Their work show that when the attack types were limited to Neptune, satan, and portsweep, they achieved a 99.63% detection rate at a 0.34% false-positive rate. However, detection rates for attack types such as buffer-overflow, guess-passwd, and xsnoop in KDD CUP 1999 data were poor.

In Ref. [9], the authors create hyper ellipsoidal clusters of maximum intra-cluster similarity and minimum intercluster similarity to more accurately classify data points of a highly intertwining nature, as seen in the KDD CUP 1999 data sets. They addressed this problem by accretively building hyper ellipsoidal clusters at a slightly higher cost than that in [8]. They were able to get detection rates of 77.27%, 95.83%, and 100%, respectively, for the attack types buffer-overflow, guess-passwd, and xsnoop. The achieved overall detection rates is between 91.55% and 91.71% at false-positive rates between 2.68% and 4.84% when used for the entire attack range of the KDD Cup 1999 data.

K-means clustering algorithm [25] positions K centers in the pattern space such that the total squared error distance between each training pattern and the nearest center is minimized. Using the K-means clustering algorithm, different clusters were specified and generated for each output class. In Ref. [25], simulations were run having 2, 4, 8, 16, 32, 40, 64, 75, 90, 110, 128, and 256 clusters. Each simulation had equal number of clusters for each attack class. For number of clusters (K) that are not integer powers of 2, after generating P clusters (P being an integer power of 2) where P>K, the cluster centers having minimum variance among its patterns were removed one at a time until the clusters were reduced to K. An epoch consisted of presenting all training patterns in an output class for which centers are being generated. Clusters were trained until the average squared error difference between two epochs was less than 1%. During splitting, the centers were disturbed by $\pm 1\%$ of the standard deviation in each cluster so that new clusters are formed. The model that achieved the lowest cost per example value (0.2389) had 16 clusters in each class.

In this paper, the proposed PCA-NG algorithm, as an alternative clustering analysis way, is applied for intrusion detection. Table 4 shows the

comparative results using the published results. The comparative approaches include HSOM [8], ESOM [9], and K-means [25]. It is fair for the comparison because all of these clustering methods are based on the KDD CUP 1999 data sets. But these experiments are based on different feature subsets or different sampled network connections. We choose the best performance of these methods for the comparison.

As Table 4 shows, the best methods for the five main categories are highlighted with gray background color. For DoS detection, all the methods get higher detection rate. For Probe detection, PCA-NG and ESOM are better. HSOM is the best for detection U2R and the PCA-NG is the best for detection R2L. The results show that our method PCA-NG outperforms the other listed clustering approaches, in average detection rates sense and especially in U2R and R2L detection. It seems that PCA-NG is very promising in the domain of intrusion detection.

Type	HSOM [8]	ESOM [9]	K-means [25]	PCA-NG
Normal	-	7.3%	-	97.2%
DOS	97.2%	97.1%	97.3%	95.1%
Probe	88.2%	97.5%	87.6%	93.0%
U2R	71.2%	52.9%	29.8%	64.7%
R2L	20.9%	1.7%	6.4%	75.1%

Table 4: The performance comparison of the five main categories data clustering on KDD CUP 1999 datasets using different clustering methods.

5 Conclusion

In this paper, we propose a new clustering method, namely principal components neural gas (PCA-NG), and its online learning algorithm is also given. The self-organizing process of neural gas network is held in the input space, while that of Kohonen's SOM is held in the out space. Combined with local principal component analysis, the PCA-NG is more suitable for Gaussian-distributed data clustering. Another contribution of this paper is that we first utilize neural gas network to the domain of intrusion detection. We give a simple labeling scheme; furthermore, the trained and labeled PCA-NG can perform as a detector or a classifier for intrusion detection. We first carry out an experiment to test the PCA-NG model on a synthetic dataset. Then we perform PCA-NG on KDD CUP 1999 intrusion detection evaluation datasets, and the results of the experiments and comparisons demonstrate the effectiveness of the methods.

Nevertheless, there are some problems in the PCA-NG method, such as the limitations of its static architecture, the influence on performance and convergent speed of its parameter initializations, the hierarchical netowrk architecture for effectiveness, etc. We will take them into account in our future work.

Acknowledgement

This work was supported by 973 Program (2007CB311002) and the Specialized Research Fund for the Doctoral Program of Chinese Universities (20070614001) and the Sichuan Province Technology R & D Program (2009GZ0004).

References:

[1] Bace and G. Rebecca, Intrusion Detection, Macmillan Technical Publishing, 2000.

[2] A. Ghosh and A. Schwartzbard, A study in using neural networks for anomaly and misuse detection, In Proceedings of the Eighth USENIX Security Symposium, (1999)141-151.

[3] T. Kohonen, Self-Organizing Maps, Berlin, Germany: Springer, 1995.

[4] S. Haykin, Neural Networks, A Comprehensive Foundation, Second Edition, Prentice-Hall, Inc, (1999).

[5] K. Fox, R. Henning and J. Reed, A neural network approach toward intrusion detection, in Proc. 13th Nat. Computer Security Conf., Washington, DC, 1990.

[6] J. Cannady and J. Mahaffey, The application of artificial intelligence to misuse detection, in Proc. 1st Recent Advances in Intrusion Detection (RAID) Conf., Louvain-la-Neuve, Belgium, 1998.

[7] B. Rhodes, J. Mahaffey and J. Cannady, Multiple self-organizing maps for intrusion detection, in Proc. 23rd Nat. Information Systems Security Conf., Baltimore, MD, Oct. 2000.

[8] S. T. Sarasamma, Q. A. Zhu and J. Huff, Hierarchical Kohonenen Net for Anomaly Detection in Network Security, IEEE Transactions on System, Man, and Cybernetics-Part B, vol. 35, no. 2, pp 302-312, Apr. 2005.

[9] S. T. Sarasamma and Q. A. Zhu, Min-Max Hyperellipsoidal Clustering for Anomaly Detection in Network Security, IEEE

Transactions on System, Man, and Cybernetics-Part B, vol. 36, no. 4, pp 887-901, Aug. 2006.

[10] M. Ramadas, S. Ostermann and B. Tjaden, Detecting Anomalous Network Traffic with Self-organizing Maps, RAID 2003, LNCS 2820, pp 36-54, Springer-Verlag Berlin Heidelberg, 2003.

[11] A. Rauber, D. Merkl and M. Dittenbach, The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data, IEEE Trans. Neural Networks, vol. 13, no. 6, Nov. 2002.

[12] M. Martinetz, S. Berkovich and K. Schulten, Neural-gas network for vector quantization and its application to time series prediction, IEEE Trans. Neural Networks 4(1993)558-569.

[13] F. Questier, Q. Guo, B. Walczak, D.L. Massart, C. Boucon and S.D. Jong, The neural-gas network for classifying analytical data, Chemometrics and Intelligent Laboratory Systems. 61(1-2)(2002)105-121.

[14] A. K. Qin and P. N. Suganthan, Robust growing neural gas algorithm with application in cluster analysis, Neural Networks, 17(2004)1135-1148

[15] R. Möller and H. Hoffmman, An extension of neural gas to local PCA, Neurocomputing, 62(2004)305-326.

[16] M. Cottrell, B. Hammer, A. Hasenfu and T. Villmann, Batch and median neural gas, Neural Networks, 19(2006)762-771.

[17] E. Oja, Principal components, minor components, and linear neural networks. Neural Networks, vol. 5, pp 927-935.

[18] P. F. Baldi, K. Hornik, Learning in linear neural networks: A survey, IEEE Trans. Neural Networks, vol. 6, pp. 837-858, July 1995.

[19] K. I. Diamantaras and S. Y. Kung, Principal component neural networks: Theory and applications, in Adaptive and Learning Systems for Signal Processing, Communications, and Control. New York: Wiley, 1996.

[20] A. Weingessel and K. Hornik, Local PCA algorithms, IEEE Trans. Neural Networks, vol. 11, NO. 6, pp.1242-1250, Nov. 2000.

[21] S. Ouyang, Z. Bao and G. Liao, Robust recursive least squares algorithm for principal component analysis, IEEE Trans. Neural Networks, 11(1) (2000)215-221.

[22] R. Lippmann, J. Haines, D. Fried, J. Korba and K. Das, The 1999 DARPA off-line intrusion detection evaluation, Computer Networks, 34, 2000, 579-595.

[23] KDD Cup 1999 Data, http://kdd.ics.uci.edu/ databases/ kddcup99, Reference data: Aug. 2009.

[24] W. Lee and S. Stolfo, A framework for constructing features and models for intrusion detection systems, In: ACM Transactions on Information and System Security, 3(2001)227-261.

[25] M. Sabhnani and G. Serpen, Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context, in Proc. Int. Conf. Machine Learning Models, Technologies and Applications, Las Vegas, NV, Jun. 2003, pp.209-215.