Intrusion Detection Using PCASOM Neural Networks

Guisong Liu and Zhang Yi

Computational Intelligence Laboratory,
School of Computer Science and Engineering,
University of Electronic Science and Technology of China,
Chengdu 610054, P.R. China
{lgs, zhangyi}@uestc.edu.cn

Abstract. This paper proposes a method to detect network intrusions by using the PCASOM (principal components analysis and self-organizing map) neural networks. A modified unsupervised learning algorithm which is more suitable for intrusion detection is presented. Experiments are carried out to illustrate the performance of the proposed method by using DARPA 1998 evaluation data sets. It shows that the proposed method can cluster the network connections into proper clusters with high detection rate and relatively low false alarm rate.

1 Introduction

Building of intrusion detection systems (IDSs) has been widely studied since the early 1980's. The challenges faced by designers increase as the targeted systems become more diverse and complex [1]. Intrusion detection has been proven to be a very valuable and powerful approach for network management as well as network security. Generally, there are two fundamental approaches used in intrusion detection technology: misuse detection and anomaly detection [2]. The misuse detection uses a signature-based database of well known intrusions and use a pattern matching scheme to detect intrusions. The anomaly detection, on the other hand, tries to quantify the normal operation of the host, or the network as a whole, with various parameters and looks for anomalous values for those parameters in real-time [3].

Artificial neural networks (ANNs) provide the potential to identify and classify network activity based on limited, incomplete, and nonlinear data sources [1]. The goal in using ANNs for intrusion detection is to be able to generalize from incomplete data and to be able to classify online data as being normal or intrusive [4]. Principal components analysis (PCA) [5, 6] and Self Organizing Feature Map (SOFM) [7] are wildly used for intrusion detection. Ezequiel López-Rubio et al. [8] proposed a new self-organizing neural network model that performs principal components analysis named PCASOM. In this paper, we use the PCASOM neural networks to to build an intrusion detection model.

This paper is organized as follows. Some basic knowledge of PCASOM algorithm is described in Section 2. The proposed algorithm for intrusion detection is

J. Wang et al. (Eds.): ISNN 2006, LNCS 3973, pp. 240–245, 2006.

given in Section 3. In Section 4, some experiments are carried out to validate the performance of the proposed method. Finally, conclusion is given in Section 5.

2 The PCASOM

It is known that in SOM the neurons are trained to model the input space and the information are stored in the weight connections of the neurons. PCASOM uses convariance matrix to store the information [8]. For the purpose of locating the winning neuron given by the data sample, PCASOM uses principal components analysis (PCA) to give the distance measure by computing the projection errors among all the neurons.

The covariance matrix of the input sequence vector $\{x\}$ is defined by $R = E[(x - E(x))(x - E(x))']$ [9]. Suppose we have M input samples at time t, the covariance matrix and expectation can be given as

$$R(t) = \frac{1}{M-1}\sum_{i=1}^{M}(x_i - E(x))(x_i - E(x))', \tag{1}$$

$$e(t) = E(x) = \frac{1}{M}\sum_{i=1}^{M} x_i. \tag{2}$$

For online computing, the covariance matrix cannot be calculated in advance. An iterative method for computing $R(t)$ and $e(t)$ is proposed in [8]. Suppose one new input connection x_{i+1} is given at time instant $t+1$, then,

$$R'(t+1) = (x_{i+1} - E(x))(x_{i+1} - E(x))', \tag{3}$$

$$e'(t+1) = x_{i+1}. \tag{4}$$

It follows from (1),(2),(3) and (4) that,

$$R(t+1) = \frac{1}{M}\sum_{i=1}^{M+1}(x_i - E(x))(x_i - E(x))' = \frac{1}{M}((M-1)R(t) + R'(t+1)), \tag{5}$$

$$e(t+1) = \frac{1}{M+1}\sum_{i=1}^{M}(x_i + x_{i+1}) = \frac{1}{M+1}(Me(t) + e'(t+1)). \tag{6}$$

Define learning rate $\eta_r = \frac{1}{M}$ and $\eta_e = \frac{1}{M+1}$, the iterative equations can be given by:

$$R(t+1) = R(t) + \eta_r(R'(t+1) - R(t)), \tag{7}$$

$$e(t+1) = e(t) + \eta_e(e'(t+1) - e(t)). \tag{8}$$

Clearly, $M(t+1) = M(t) + 1$, so the learning rate η_r and η_e are approaching to zero as time goes on.

We initialize the unit number as m in the model. For the unit j at time instant t, based on the theory of principal components analysis and subspace

decomposition [9], the K main components directions of the input vector x_i (network connection data) along which x_i has the largest deviation can be obtained (This means every unit has K neurons.The weights of the neurons represent the K principal components directions.). Those orthogonal vectors $B = \{b_i | i = 1, \ldots, K\}$ consist the feature space of the input network connections. The input vector error on the unit j can be easily defined by projecting it on the basis vector B_j as follows [8]:

$$\hat{x}_i^j(t) = \left\| x_i(t) - e^j(t) - \sum_{h=1}^{K} \left(b_h^{iT}(x_i(t) - e^j(t)) b_h^i \right) \right\|. \tag{9}$$

The winning unit has the minimum value $\hat{x}_i^j(t)$. For simplicity we assume the degree of neighborhood of the winning unit as one. The winning unit C can be obtained through competition process as follows:

$$C = \arg\min_j \left\{ \hat{x}_i^j(t) \right\}, j = 1, \cdots, m. \tag{10}$$

For every unit j, two small constants μ_r and μ_e are defined. If at time t, the new input x_{i+1} satisfies

$$||R'(t+1) - R(t)|| \leq \mu_r, \tag{11}$$

$$||e'(t+1) - e(t)|| \leq \mu_e, \tag{12}$$

we can ignore the influence of the new input to the winning unit and need not update the weights of the neurons in the unit. The training process of that unit will be stopped.

3 Algorithm Description

The proposed algorithm can be described as follows:

1. Preparing training and testing data. Decide the unit number m according to particular applications. Set the initial principal components number K.
2. For every unit j, initialize $R^j(0)$ and $e^j(0)$. $R^j(0)$ is a random symmetric nonnegative matrix by setting the main diagonal element near to one and all the others near to zero. Initial the vector $e^j(0)$ by using small random values near to zero [8].
3. Training process: at time t, input the training data. For every unit j, according to (3) and (4), compute $R'(t+1)$ and $e'(t+1)$.
4. Competition and updating: according to (9)and(10), find the winning unit j, then update $R^j(t+1)$ and $e^j(t+1)$ according to (7) and (8).
5. For the winning unit j, if (11) and (12) are satisfied, stop training process. Otherwise go to step 3. Until all the units satisfy (11) and (12), stop the whole training process.

4 Experiments

We use KDD Cup 1999 Data in our experiments [10]. It is a subversion of DARPA project [11]. The available database is made up of a large number of network connections related to normal and malicious traffic. Each connection includes forty-one feature values. We discard protocol-type, service and flag three pure symbolic features. Then we obtain the dataset with thirty-eight feature values.

We select $5,760$ ($1,740$ normal and $4,020$ intrusions) labeled connections randomly to test our IDS model. Besides normal, the other five popular type of attacks are included (such as smurf,neptune,back,guess-password and satan). The labeled value (data type) in the connections is discarded in our training process but just for experimental result evaluation. Gopi K. Kuchimanchi et al [5] use traditional PCA to reduce the dimensionality from forty-one to nineteen. We select the principal components number as twenty and the unit number as ten. The experimental results are shown in Table 1.

Table 1. The clustering results of PCASOM (20 PCs, 10 units)

clusters	1	2	3	4	5	6	7	8	9	10	detection
type	–	smurf	–	–	gpwd	–	neptune	normal	–	back	rate(%)
normal	3	26	0	0	1	1	32	1676	1	0	96.32
neptune	0	0	0	0	0	0	992	0	0	28	97.26
smurf	0	1000	0	0	0	0	0	0	0	0	100
gpwd	3	9	0	13	693	1	0	181	0	0	77
back	0	0	0	0	0	0	0	1	0	799	99.87
satan	0	1	4	0	0	0	295	0	0	0	0
accuracy(%)	–	96.53	–	–	99.86	–	75.21	90.20	–	96.61	–

Obviously, the selection of learning rate in the training process is very important for every unsupervised algorithm. In our experiments, the learning rate is defined as follows:

$$\eta_r^j(t) = \eta_e^j(t) = \frac{1}{1 + WinTimes(t,j)/p}.$$

For the winning unit j at time t, the winning times is recorded in $WinTimes(t,j)$. The parameter p is an integer type constant. The learning rate can be adjusted through varying p. In the above experiment, we selected p as 8 to obtain the best performance.

Table 1 shows that all the training data are clearly clustered into five clusters which have been labeled with type name using our label machine. The last row is named accuracy. For example,cluster two (smurf, 96.53%) means our model has detected $1,036$ connections as smurf, but only $1,000$ connections are real smurf (accuracy $= 1,000/1,036 * 100\%$). We can calculate total intrusion detection rate (IDR, as the view of two types, normal or intrusions), average detection rate (ADR) and false alarm rate (FAR), where

$$IDR = 1 - (1 + 181)/4020 * 100\% = 95.47\%,$$
$$ADR = (1676 + 992 + 1000 + 693 + 799)/5760 * 100\% = 89.58\%,$$
$$FAR = (1 - 1676/1740) * 100\% = 3.68\%.$$

The second step of the experiment is to test the performance of this model in different unit setup. We select the unit number as 6, 8, 10, 12, 15 respectively. Table 2 shows the results under different scenarios.

Table 2. The performance comparison using different unit number setup (20 PCs)

unit number	6	8	10	12	15
main clusters	5	5	5	7	7
$IDR(\%)$	95.57	95.52	95.47	94.60	95.52
$ADR(\%)$	84.97	82.69	89.58	87.80	84.30
$FAR(\%)$	3.68	23.68	3.68	2.70	17.70

In PCASOM model, the different unit setup will impact the performance as other clustering algorithms do. But Table 2 shows that the ADR and IDR vary little and maintain a high performance value under different scenarios. The main cause of FAR rising is that this model classifies the normal connections into two or three clusters. To solve the problem in actual intrusion detection system, we can use a anomaly-based classifier to separate the intrusions from normal first, then this model can be used to categorize those filtered attack data to avoid relative high false alarm rate.

Generally, supervised learning methods significantly outperform the unsupervised ones if the testing data contain no unknown attacks. But the performance of unsupervised learning is not impacted by unknown attacks. The following experiments demonstrate this standpoint. Some data of new type intrusion are added(700 new intrusive connections like *teardrop*) into the testing data, which never appears in the previous training. The model can group them as a new cluster (but not all other types of intrusion like this). The detection rate is 94.286% (660 of 700). The result shows that our model is adapted to novel intrusion detection with high accuracy.

In our experiments with different unit setup, the satan type intrusion cannot be separated from neptune. Maybe these tow type connections are too similar to separate by using this model. Hence, the distance measure methods deserve a more detail analysis. Another good way is using hybrid neural network [12]. These are all our future works.

5 Conclusions

The PCASOM is used in this paper to study the network-based intrusion detection. This model can cluster the input data as SOM with competition process based on principal components analysis. Unlike supervised learning algorithm,

this model has the ability to learn from unlabeled training data. It's helpful for data preparation, because it is difficult to label the network connections as normal or not in the real network environment.

The simulations shown that this model is adapted to both anomaly detection and misuse detection. Different unit number setup will impact the performance of ADR and IDR little. It can achieve high detection rate and relative low false alarm rate.

References

1. Cannady, J.: Artificial Neural Networks for Misuse Detection. Proceedings, National Information Systems Security Conference (NISSC'98), Arlington VA (1998) 443-456
2. Anderson, D., Frivold, T., Valdes, A.: Next-generation Intrusion Detection Expert System(NIDES): A Summary. SRI International Technical Report, SRI-CSL-95-07
3. Ramadas, M., Ostermann, S., Tjaden, B.: Detecting Anomalous Network Traffic with Self-organizing Maps. Lecture Notes in Computer Science, Vol. 2820. Springer-Verlag GmbH (2003) 36-54
4. Ghosh, A., Schwartzbard, A.: A Study in Using Neural Networks for Anomaly and Misuse Detection. In Proceedings of the Eighth USENIX Security Symposium (1999) 141-151
5. Kuchimanchi, G.K., Phoha, V.V., Balagami, K.S., Gaddam, S.R.: Dimension Reduction Using Feature Extraction Methods for Real-time Misuse Detection Systems. Proceedings of the 2004 IEEE Workshop on Information Assurance and Security (2004) 195-202
6. Labib, K., Vemuri, V. R.: Detecting and Visualizing Denial-of-Service and Network Probe Attacks Using Principal Component Analysis. Third Conference on Security and Network Architectures, La Londe, France (2004)
7. Lei, J.Z., Ghorbani, A.: Network Intrusion Detection Using an Improved Competitive Learning Neural Network. Second Annual Conference on Communication Networks and Services Research (2004) 190-197
8. Rubio, E.L., Prez, J. M., Antonio, J., Ruiz, G.: A Principal Components Analysis Self-organizing Map. Neural Networks 17(2) (2004) 261-270
9. Haykin, S.: Neural Networks: A Comprehensive Foundation. 2nd edn. Tsinghua University Press, Beijing (2001)
10. KDD Cup 1999 Data: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
11. DARPA Intrusion Detection Evaluation Project:http://www.ll.mit.edu/LST/ideval/
12. Pan, Z.S., Chen, S.C., Hu, G.B., Zhang, D.Q.: Hybrid Neural Network and C4.5 for Misuse Detection. Proceedings of the Second International Conference on Machine Learning and Cybernetics (2003) 2463-2467