A General Framework for Adaptive Anomaly Detection with Evolving Connectionist Systems

Yihua Liao*, V. Rao Vemuri*† and Alejandro Pasos†

*Department of Computer Science

†Department of Applied Science
University of California, Davis
One Shields Ave, Davis
CA 95616, USA

Email: {yhliao, rvemuri, apasos}@ucdavis.edu

Abstract

A new adaptive anomaly detection framework, based on the use of unsupervised evolving connectionist systems, is proposed to address the issue of concept drift. It is designed to adapt to normal behavior changes while still recognizing anomalies. The evolving connectionist systems learn a subject's behavior in an online, adaptive fashion without *a priori* knowledge of the underlying data distributions. Experiments with the KDD Cup 1999 network data and the Windows NT user profiling data show that our adaptive anomaly detection systems, based on Fuzzy Adaptive Resonance Theory (ART) and Evolving Fuzzy Neural Networks (EFuNN), can significantly reduce the false alarm rate while the attack detection rate remains high.

Index Terms

Adaptive anomaly detection, concept drift, evolving connectionist systems, Fuzzy ART, EFuNN

A General Framework for Adaptive Anomaly Detection with Evolving Connectionist Systems

I. Introduction

Computer security vulnerabilities and flaws are being discovered every day. Given the rapid increase in connectivity and accessibility of computer systems in today's society, computer intrusions and security breaches are posing serious threats to national security as well as enterprise interests. As one of the two general approaches to intrusion detection, anomaly detection has been under intensive study for the last two decades [1]. Unlike the alternative approach, misuse detection, which generates an alarm when a known attack signature is matched, anomaly detection tries to identify activities that deviate from the normal behavior of the monitored system, network or users. Anomaly detection techniques hold great potential for detecting attempts to exploit new and unforeseen vulnerabilities, as well as "abuse of privileges" types of attacks by legitimate users, the so-called "insider threat".

Despite that a number of anomaly detection methodologies based on machine learning and statistical methods have been developed over the years (e.g., [2]), the success of anomaly detection has been limited. A major challenge is how to accurately model a subject's normal behavior while it changes over time in a continuous manner, a problem known as *concept drift*. In a practical environment, system and network activities as well as user behavior could change for bona fide reasons. For example, an employee of a large company may have to learn and run a completely different set of computer programs for a new assignment. As a result, his present computer usage logs would differ significantly from his previous profile, as if a masquerade attack had occurred. Modeling a subject's normal behavior in the presence of concept drift is a challenging task because the underlying data distribution is not known *a priori*, unexpected changes may happen at any time, and therefore the normal behavior may not be strictly predictable in the long term. The key to this difficult problem is adaptive learning. An effective anomaly detection system should be capable of adapting to normal behavior changes while still recognizing anomalous activities. Otherwise, large amount of false alarms would be generated if the model failed to change adaptively to accommodate the new patterns [3]. A seemingly obvious solution is to update the training corpus with each new batch of audit data and re-build the normal behavior model. However, it is not computationally feasible for most existing methods (e.g., [4] [5]) because they are expensive to generate a model and not suitable for incremental, adaptive learning. Moreover, selecting appropriate training instances without contaminating the normal behavior profile is a nontrivial issue.

In this paper, we present an adaptive anomaly detection framework that is applicable to host-based and network-based intrusion detection. Our framework employs unsupervised evolving connectionist systems to learn system, network or user behavior in an online, adaptive fashion without *a priori* knowledge of the underlying data distributions.

Adaptive learning and evolving connectionist systems are an active area of artificial intelligence research. Evolving connectionist systems are artificial neural networks that resemble the human cognitive information processing models. Due to their self-organizing and adaptive nature, they provide powerful tools for modeling evolving processes and knowledge discovery [6].

Our adaptive anomaly detection framework performs one-pass clustering of the input data stream that represents a monitored subject's behavior patterns. Each new incoming instance is assigned to one of the three states: *normal*, *uncertain* and *anomalous*. Two different alarm levels are defined to reduce the risk of false alarming. We evaluated our adaptive anomaly detection systems, based on the Fuzzy Adaptive Resonance Theory (Fuzzy ART) [7] and Evolving Fuzzy Neural Networks (EFuNN) [8], over two types of datasets, the KDD Cup 1999 network data [9] and Windows NT user profiling data. Our experiments show that both evolving connectionist systems are able to adapt to user or network normal behavior changes and at the same time detect anomalous activities. Compared to support vector machines (SVM) based static learning, our adaptive anomaly detection systems significantly reduced the false alarm rate.

The rest of this paper is organized as follows. In Section 2 we review some related work on adaptive anomaly detection. Section 3 presents our adaptive framework. Section 4 details our experiments with the KDD Cup

1999 network data and the Windows NT user profiling data. Section 5 contains further discussions. Finally, we summarize our conclusions and future work in Section 6.

II. Related Work

Among the few adaptive anomaly detection systems, NIDES [2] is probably the best known. It generates user profiles that are constantly aged by multiplying them by an exponential decay factors. This method of aging creates a moving time window for the profile data, so that the new behavior is only compared to the most recently observed behaviors that fall into the time window. One drawback of NIDES is that a user's recurring behavior can cause frequent and unnecessary updates of the profile, let alone its complex statistical model.

Teng et al. [10] used inductively generated sequential patterns to perform adaptive real time anomaly detection. Lane and Brodley [11] proposed an nearest neighbor classifiers based online learning scheme and examined the issues of incremental updating of system parameters and instance selection. Mixture models were employed in [12] and [13] to generate adaptive probabilistic models and detect anomalies within a dataset. Cannady [14] demonstrated the use of a reinforcement learning method that uses feedback from the protected system. Fan [15] used ensembles of classification models to adapt existing models in order to detect newly established patterns. Hossain and Bridges [16] proposed a fuzzy association rule mining architecture for adaptive anomaly detection.

Compared to previous statistical or rule-learning based adaptive anomaly detection systems, our framework does not require *a priori* knowledge of the underlying data distributions. Through the use of evolving connectionist systems, it provides efficient adaptation to new patterns in a dynamic environment. Unlike other neural networks that have been applied to intrusion detection (e.g., [17] [18]) as "black boxes", our evolving connectionist systems can provide knowledge (i.e., the weight vectors) to "explain" the learned normal behavior patterns.

Our approach also falls into the category of unsupervised anomaly detection [19]–[21] as it does not require the knowledge of data labels. However, our algorithms assign each instance into a cluster in an online, adaptive mode. No distinction between training and testing has to be made. Therefore the period of system initialization during which all behaviors are assumed normal is not necessary.

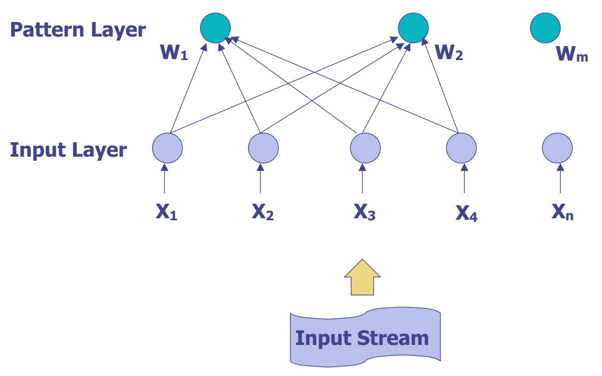
III. Adaptive Anomaly Detection Framework

In addressing the problem of adaptive anomaly detection two fundamental questions arise: (a) How to generate a model or profile that can concisely describe a subject's normal behavior, and more importantly, can it be updated efficiently to accommodate new behavior patterns? (b) How to select instances to update the model without introducing noise and incorporating abnormal patterns as normal? Our adaptive anomaly detection framework addresses these issues through the use of online unsupervised learning methods, under the assumption that normal instances cluster together in the input space, whereas the anomalous activities correspond to outliers that lie in sparse regions of the input space. Our framework is general in that the underlying clustering method can be any online unsupervised evolving connectionist system and it can be used for different types of audit data. Without loss of generality, we assume the audit data that is continuously fed into the adaptive anomaly detection system has been transformed into a stream of input vectors after pre-processing, where the input features describe the monitored subject's behavior.

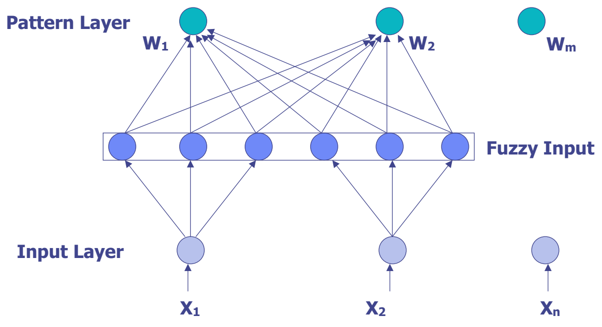
The evolving connectionist systems are designed for modeling evolving processes. They operate continuously in time and adapt their structure and functionality through a continuous interaction with the environment [6]. They are stable enough to retain patterns learned from previously observed data while being flexible enough to learn new patterns from new incoming data. They can learn in unsupervised, supervised or reinforcement learning modes. The online unsupervised evolving connectionist systems provide one-pass clustering of an input data stream, where there is no predefined number of different clusters that the data belong to.

A simplified diagram of an evolving connectionist system for online unsupervised learning is given in Figure 1(a) (some systems such as EFuNN may have an additional fuzzy input layer, shown in Figure 1(b), which represents the fuzzy quantization of the original inputs with the use of membership functions [22]). A typical unsupervised evolving connectionist system consists of two layers of nodes: an *input* layer that reads the input vectors into the system continuously, and a *pattern* layer (or cluster layer) representing previously learned patterns. Each pattern node corresponds to a cluster in the input space. Each cluster, in turn, is represented by a weight vector. Then the subject's normal behavior profile is conveniently described as a set of weight vectors that represent the clustering of the previous audit data.

A distance measure has to be defined to measure the mismatch between a new instance (i.e., a new input



(a)



(b)

Fig. 1. (a) A simplified diagram of an evolving connectionist system for unsupervised learning. The system has n input nodes and m pattern nodes. There is a connection from each input node to every pattern node. Some connections are not shown in the figure. (b) An evolving connectionist system that has an additional fuzzy input layer. The task of the fuzzy input nodes is to transfer the input values into membership degrees.

vector) and existing patterns. Based on the distance measure, the system either assigns an input vector to one of the existing patterns and updates the pattern weight vector to accommodate the new input, or otherwise creates a new pattern node for the input. The details of clustering vary with different evolving connectionist systems.

In order to reduce the risk of false alarms (classifying normal instances as abnormal), we define three states of behavior patterns (i.e., the pattern nodes of the evolving connectionist system): *normal*, *uncertain* and *anomalous*. Accordingly, each instance is labeled as either *normal*, *uncertain* or *anomalous*. In addition, the alarm is differentiated into two levels: *Level 1 alarm* and *Level 2 alarm*, representing different degrees of anomaly. As illustrated in Figure 2, a new instance is assigned to one of the existing normal patterns and labeled *normal* if the similarity between the input vector and the normal pattern is above a threshold (the *vigilance* parameter). Otherwise, it is *uncertain*. The *uncertain* instance is either assigned to one of the existing *uncertain* patterns if it is close enough to that *uncertain* pattern, or becomes the only member of a new *uncertain* pattern. A *Level 1 alarm* is triggered whenever a new *uncertain* pattern is created as the new instance is different from all the learned patterns and thus deserves special attention. At this point, some preliminary security measures need to be taken. However, one can not draw a final conclusion yet. The new instance can be truly anomalous or merely the beginning of a new normal behavior pattern, which will be determined by the subsequent instances. After the processing of a certain number (the N_{watch} parameter) of the subsequent instances in the same manner, if the number of members of an *uncertain* pattern reaches a threshold value (the Min_{count} parameter), the *uncertain* pattern becomes a *normal* pattern and the labels of all its members are changed from *uncertain* to *normal*. This indicates that a new behavior pattern has been developed and incorporated into the subject's normal behavior profile as enough instances have shown the same pattern. On the other hand, after N_{watch} subsequent instances, any *uncertain* pattern with less than Min_{count} members will be destroyed and all its members are labeled *anomalous*. This will make sure that anomalous patterns, corresponding to the sparse regions in the input space, will not be included into the normal profile. A *Level 2 alarm* is issued when an instance is labeled *anomalous* and further response actions are expected.

The main tunable parameters of an adaptive anomaly detection system are summarized as follows:

- *Vigilance* ρ. This threshold controls the degree of mismatch between new instances and existing patterns that the system can tolerate.
- *Learning rate* β. It determines how fast the system should adapt to a new instance when it is assigned to a pattern.
- N_{watch}. It is the period that the system will wait before making a decision on a newly created *uncertain* pattern.
- Min_{count}, the minimum number of members that an *uncertain* pattern should have in order to be recognized as a *normal* pattern.

Our framework does not require *a priori* knowledge of the number of input features. When a new input

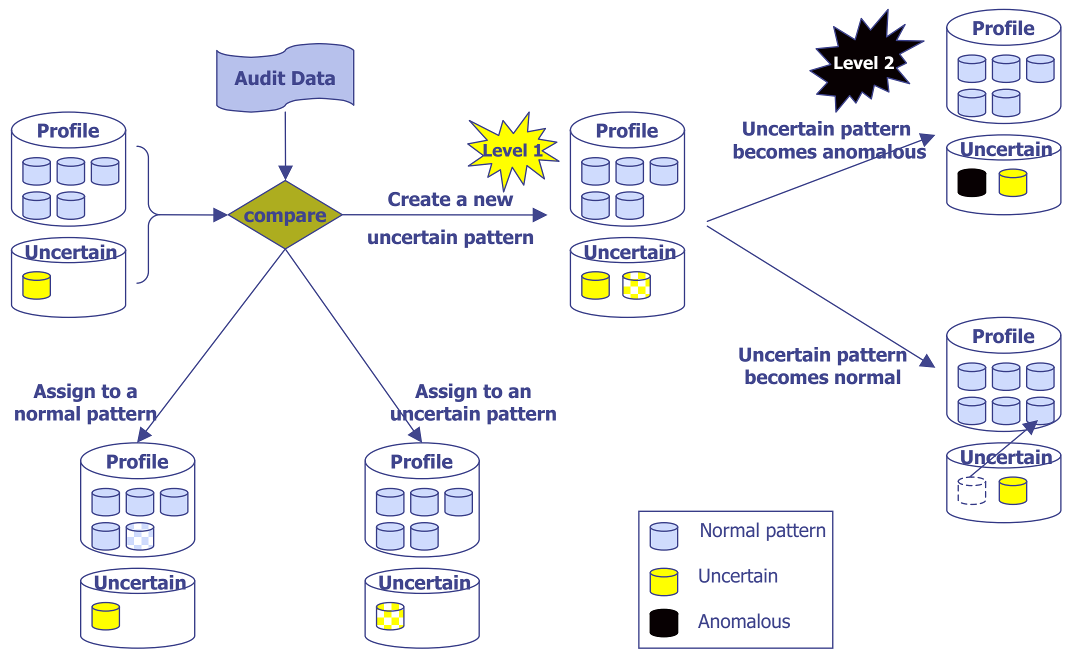


Fig. 2. Adaptive anomaly detection framework.

feature is presented, the system simply adds a new input node to the input layer and connections from this newly created input node to the existing pattern nodes. This can be very important when the features that describe a subject's behavior grow over time and can't be foreseen in a dynamic environment. Similarly, accommodation of a new pattern is efficiently realized by creating a new pattern node and adding connections from input nodes to this new pattern node. The rest of the structure remains the same.

With the framework, the learned normal profile is expressed as a set of weight vectors representing the coordinates of the cluster centers in the input space. These weight vectors can be interpreted as a knowledge presentation that can be used to describe the subject's behavior patterns, and thus they can facilitate understanding of the subject's behavior. The weight vectors are stored in the long term memory of the connectionist systems. Since new instances are compared to all previously learned patterns, recurring activities would be recognized easily.

While the underlying clustering method of the adaptive anomaly detection framework can be any unsupervised evolving connectionist system, Fuzzy ART and the unsupervised learning version of EFuNN are adapted for anomaly detection in this paper. Both of them are conceptually simple and computationally fast. Furthermore, they cope well with fuzzy data, and the fuzzy distance measures help to smooth the abrupt separation of normality and abnormality of a subject's behavior. The details on Fuzzy ART and EFuNN are presented in the Appendix.

IV. Experiments

In this section we describe some experiments. The emphasis of the experiments is on the understanding of how Fuzzy ART and EFuNN based adaptive anomaly detection systems work in practice. One objective of our experiments is to observe the influence of variability of the tunable parameters on the performance of an anomaly detection system. Another objective of the experiments is to compare SVM based static learning and evolving connectionist system based adaptive learning.

A. *Static learning via support vector machines*

Support Vector Machine is a relatively new and powerful learning method pioneered by Vapnik [23]. It is based on the so-called *structural risk minimization* principle, which minimizes an upper bound on the generalization error. The method performs a mapping from the input space to a higher-dimensional feature space through the use of a kernel function. It separates the data in the feature space by means of a maximum margin hyperplane. Schölkopf et al. [24] proposed a method of adapting the SVM paradigm to the one-class problem. The origin of the coordinate system, after transforming the feature via a kernel, is treated as the only member of the second class. Training a SVM is equivalent to solving a linearly constrained quadratic programming problem.

In our experiments, we used SVM to demonstrate the weakness of static learning and the importance of adaptive learning. SVM was employed to learn a model (i.e., support vectors) that fits the training dataset. The model was then tested on the testing dataset without any update (thus it is static learning). SVM is optimal when the data are independent and identically distributed (i.i.d.). If there was concept drift between the training dataset and the testing dataset, SVM would generate classification errors. Adaptive learning can adapt to concept changes incrementally and learn new patterns when new testing instances are presented to the learning system. Therefore the classification accuracy is improved.

In our research, we used LIBSVM (version 2.35) [25], an integrated tool for SVM classification and regression.

B. Cost function

To facilitate performance comparison among different methods, we used the cost function:

$$Cost = (1 - hit\,rate) + \gamma * false\,positive\,rate,$$

where the hit rate is the rate of detected intrusions (attacks or masquerades), the false positive rate is the probability that a normal instance is classified as *anomalous*, and the parameter γ represents the cost difference between a false alarm and a miss. Here we set the γ value to 6, which was used in [3]. The lower cost, the better performance an intrusion detection system has.

C. Network intrusion detection

We conducted a series of experiments on a subset of the dataset KDD Cup 1999 [9] prepared for network intrusion detection. Many methods have been tested with this popular dataset for supervised intrusion detection. The data labels were usually used for training the learning systems. Our evolving connectionist systems, however, do not rely on the data labels. They build network connection patterns incrementally in an online unsupervised learning mode.

The 1999 KDD Cup network traffic data are connection-based. Each data record, described by 7 symbolic attributes and 34 continuous attributes, corresponds to a TCP/IP connection between two IP addresses. In addition, a label is provided indicating whether the record is normal or it belongs one of the four attack types (*Probe, DoS, U2R* and *R2L*). The symbolic attributes that have two possible values (e.g., *logged_in*) were represented by a binary entry with the value of 0 or 1. For symbolic attributes that have more than two possible categorical values, we used multiple entries to encode them in the vector representation, one entry for each possible value. The entry corresponding to the category value has a value of 1 while the other entries are set to 0. The attribute *service* has 41 types, and we further classified them into {*http, smtp, ftp, ftp_data, others*} to reduce the vector dimensions. The resulting feature vectors have a total of 57 dimensions.

Since different continuous attributes were measured on very different scales, the effect of some attributes might be completely dwarfed by others that have larger scales. Therefore we scaled the attributes to the range of $[0, 1]$ by calculating:

$$X_i = \frac{v_i - min(v_i)}{max(v_i) - min(v_i)},$$

where v_i is the actual value of attribute i, and the maximum and minimum are taken over the whole dataset. However, we are aware that this scaling technique would not work if the maximum and minimum values are not known *a priori*.

We formed a subset of the original dataset consisting of 97277 normal connections and 9199 attacks by randomly sampling. We then conducted two experiments with this subset. The first experiment (*Exp. 1*) was designed to test our evolving connectionist systems. In the data stream of *Exp. 1*, the attack examples randomly drawn from the 9199 attacks were inserted into the 97277 normal examples with a 1% probability. Fuzzy ART and EFuNN were employed to model the network connections on the fly from an empty set of normal patterns and detect the intrusions in the data stream. For the second experiment (*Exp. 2*), the training dataset and testing dataset were formed to compare the performance between static learning and adaptive learning. The first 40% of the 97277 normal examples were used for training, and the rest for testing. The testing dataset also included attacks interspersed into the normal examples with the probability of 1%. The model learned from the training examples was applied to the testing dataset. The model remained unchanged during the testing process for static learning, while it was updated continuously for adaptive learning methods. Table I lists the numbers of normal and attack examples in *Exp. 1* and *Exp. 2*.

TABLE I
NUMBERS OF NORMAL AND ATTACK EXAMPLES IN *Exp. 1* AND *Exp. 2*.

Exp. 1		*Exp. 2*		
normal	attacks	training normal	testing normal	attacks
97277	998	38910	58367	580

1) Effectiveness of varying vigilance: The *vigilance* parameter ρ controls the degree of mismatch between new instances and previously learned patterns. The greater the value of *vigilance*, the more similar the instances ought to be in order to be assigned to a pattern. We studied the effect of varying ρ while keeping the values of other parameters fixed. Table II presents the results when ρ's value was varied from 0.9 to 0.99 with the data stream of *Exp. 1*. The *learning rate* parameter β was set to 0.1, N_{watch} was 8 and Min_{count} was 4. The false positive rate was calculated as the percentage of normal instances that were labeled *anomalous* out of the 97277 normal examples. Similarly, the hit rate was the percentage of detected attacks (i.e., labeled *anomalous*) out of the 998 attacks.

The results show that the false positive rate increases monotonically as the vigilance threshold is raised. This is due to the fact that more normal instances are classified as *uncertain* and then *anomalous* when the value of ρ increases. Meanwhile the hit rate oscillates at lower ρ values, and then approaches to 100% as ρ is raised nearer to 1.0. The cost of Fuzzy ART reaches the lowest value at $\rho = 0.93$ with a false positive rate of 2.35% and hit rate of 86.3%. For EFuNN, the lowest cost is obtained at $\rho = 0.96$ while the hit rate is 90% and the false positive rate is as low as 1.97%.

2) Effectiveness of varying learning rate: The *learning rate* parameter β determines how fast the system should adapt to new instances in order to accommodate them. A higher value of β places more weight to the new instance when it is assigned to a pattern and less weight to existing members of the pattern. We evaluated the performance of Fuzzy ART and EFuNN with the *Exp. 1* data stream by widely varying the *learning rate*. The results are described in Table III. The *vigilance* parameter was set to 0.93 for Fuzzy ART and 0.96 for EFuNN respectively since they provided the lowest cost when the effectiveness of varying vigilance was studied. N_{watch} was set to 8 and Min_{count} was 4.

It is interesting to note that for the *Exp. 1* dataset, $\beta = 0.1$ appears to be the best choice for both Fuzzy ART and EFuNN in terms of the cost. Higher β values provide relatively stable false positive rates and hit rates. For Fuzzy ART, lower β values ($\beta = 0.01$ or 0.001) causes much lower false positive rates as well as lower hit rates. For EFuNN, however, the false positive rate gets even higher at lower β values while the hit rate declines slightly.

3) Effectiveness of varying N_{watch} and Min_{count}: N_{watch} and Min_{count} are two other important parameters for an adaptive anomaly detection system. N_{watch} represents the delay the system will experience before it evaluates a newly created *uncertain* pattern. If it is too long, there is a risk that an anomalous instance can not be handled in a timely manner. If it's too short, large amount of false alarms may be generated. Min_{count} is the minimum number of members that an *uncertain* pattern ought to have before it is changed to *normal*. We empirically studied the effect of varying N_{watch} and Min_{count} on the performance of Fuzzy ART and EFuNN. Different values of N_{watch} and Min_{count} and the corresponding results are described in Table IV. The *vigilance* parameter was set to 0.93 for Fuzzy ART and 0.96 for EFuNN, and the *learning rate* was 0.1 for both of them.

The results show that $N_{watch} = 8$ and $Min_{count} = 4$ is a better choice than others for Fuzzy ART as it provides the lowest cost. Similarly, $N_{watch} = 4$ and $Min_{count} = 2$ gives the best performance for EFuNN. The hit rate of EFuNN is higher and more stable than that of Fuzzy ART as the values of N_{watch} and Min_{count} change. It indicates that given the distance measure of EFuNN, the attacks are more distinguishable among the normal instances.

4) Static Learning vs adaptive learning: We compared Fuzzy ART and EFuNN with SVM using the *Exp. 2* datasets. During the training process, Fuzzy ART and EFuNN assumed every pattern was normal and no instances was discarded. During the testing process, however, the task of Fuzzy ART and EFuNN became twofold: evolving their structure to accommodate new patterns and detecting anomalous instances. For simplicity, we set N_{watch} to 8 and Min_{count} to 4. We then varied the *vigilance* parameter's value from 0.9 to 0.99, and the *learning rate*'s value from 0.01 to 0.9. The parameter settings that provide the lowest cost for Fuzzy ART and EFuNN are shown in Table V.

The SVM model learned from the one-class training dataset was applied to the testing dataset. Common types of kernel functions used in SVM include linear, radial basis and polynomial functions. In our experiments, we found the radial basis kernel performed better than other kernel functions for one-class learning. The parameter ν [24], which controls the number of support vectors and errors, was determined by cross validation with the training data sets.

Table V compares the performance of SVM, Fuzzy ART and EFuNN. SVM was able to detect 90% of the attacks in the testing dataset. However, the flase positive rate was as high as 12.4%, which indicates the presence of concept drift between the training dataset and the testing dataset. Compared to SVM, Fuzzy ART and EFuNN generated significantly less false alarms. Fuzzy ART was the best in terms of hit rate, whereas EFuNN

TABLE II

THE PERFORMANCE (FALSE POSITIVE RATE, HIT RATE AND COST) OF FUZZY ART AND EFUNN WITH THE EXP. 1 DATA STREAM. RESULTS ILLUSTRATE THE IMPACT OF VARYING ρ ON THEIR PERFORMANCE.

	Fuzzy ART			EFuNN		
ρ	false positive rate	hit rate	cost	false positive rate	hit rate	cost
0.90	1.82%	79.8%	0.311	0.259%	33.4%	0.682
0.91	2.07%	73.6%	0.389	0.340%	37.2%	0.649
0.92	2.06%	66.3%	0.460	0.421%	39.3%	0.632
0.93	**2.35%**	**86.3%**	**0.278**	0.573%	66.4%	0.370
0.94	2.31%	66.9%	0.469	0.823%	57.4%	0.475
0.95	3.13%	66.6%	0.521	1.29%	74.9%	0.328
0.96	3.33%	64.4%	0.556	**1.97%**	**90.0%**	**0.218**
0.97	4.42%	89.7%	0.369	3.30%	91.7%	0.281
0.98	5.81%	93.2%	0.417	6.99%	98.7%	0.433
0.99	8.84%	98.1%	0.549	18.3%	99.6%	1.10

TABLE III

THE PERFORMANCE OF FUZZY ART AND EFUNN WITH THE EXP. 1 DATA STREAM. RESULTS ILLUSTRATE THE IMPACT OF VARYING β ON THEIR PERFORMANCE.

	Fuzzy ART			EFuNN		
β	false positive rate	hit rate	cost	false positive rate	hit rate	cost
0.001	0.256%	24.9%	0.766	2.34%	76.4%	0.377
0.01	0.675%	54.7%	0.493	2.20%	88.3%	0.249
0.1	**2.35%**	**86.3%**	**0.278**	**1.97%**	**90.0%**	**0.218**
0.3	3.17%	68.6%	0.504	1.69%	77.3%	0.329
0.5	3.44%	71.0%	0.496	1.64%	72.7%	0.371
0.7	3.58%	70.1%	0.513	1.60%	77.4%	0.322
0.9	3.53%	79.0%	0.369	1.47%	76.1%	0.327
1.0	3.23%	67.8%	0.515	1.55%	74.9%	0.343

TABLE IV

THE PERFORMANCE OF FUZZY ART AND EFUNN WITH THE EXP. 1 DATA STREAM. RESULTS ILLUSTRATE THE IMPACT OF VARYING N_{watch} AND Min_{count} ON THEIR PERFORMANCE.

		Fuzzy ART			EFuNN		
N_{watch}	Min_{count}	false positive rate	hit rate	cost	false positive rate	hit rate	cost
4	2	1.71%	74.2%	0.360	**1.53%**	**88.9%**	**0.203**
8	4	**2.35%**	**86.3%**	**0.278**	1.97%	90.0%	0.218
12	4	1.77%	77.1%	0.335	1.65%	89.4%	0.205
12	6	4.03%	70.0%	0.542	3.66%	92.9%	0.291
12	8	6.04%	95.4%	0.408	5.04%	95.7%	0.345
16	6	2.97%	61.1%	0.567	2.95%	92.9%	0.248
16	8	4.95%	72.0%	0.577	4.19%	94.3%	0.309
16	10	6.77%	84.2%	0.564	6.41%	96.3%	0.422

gave the lowest cost.

D. Masquerade detection with the user profiling data

1) dataset descriptions: We obtained a set of Windows NT user profiling data from an NSA officer. The data was collected for 20 users on 21 different hosts in a real-world government agency environment (a single user might have worked on multiple hosts). During the raw data collection, a tool was developed to query the Windows NT process table periodically (2 to 3 times per second) and collect all the process information of each user's login session. Processes that were not related to user identification were filtered out during the pre-processing. The processes that correspond to the windows the user activated are of special interest to us because they represent the programs the user was

TABLE V
THE PERFORMANCE OF SVM, FUZZY ART AND EFUNN IN *Exp. 2*.

	SVM	Fuzzy ART	EFuNN
ρ		0.93	0.96
β		0.2	0.01
false positive rate	12.4%	2.98%	0.884%
hit rate	90.7%	94.0%	85.0%
cost	0.836	0.239	0.203

running. The accumulated CPU time was calculated for each of these window-associated processes from a user's login to logout, which reflects the workload the user performed during this login session. For processes that have the same process name, their CPU times were added together. Then a CPU time vector can be formed for each login session, where the value of each entry is the percentage of CPU time consumed by a unique process during this login session. There are 105 processes contained in this dataset, for example, *netscape*, *explorer*, *outlook*, *msoffice* and so on, while each individual user has his or her own process "vocabulary".

In addition to the CPU information, we also included the login time in the input vectors. A user's login time was categorized as *early morning* (before 7 AM), *morning* (between 7 AM and 12 PM), *afternoon* (between 12 PM and 6 PM) or *evening* (later than 6 PM). Since the login time has four possible values, we have four entries in the input vectors corresponding to the login time. Therefore each login session is encoded as a vector that contains CPU time percentages consumed by the user as well as four additional entries that represent the user's login time.

We selected the 7 users that have the most login sessions to serve as our masquerade targets. We then used the remaining 13 users as masqueraders and inserted their data into the data of the 7 users. Two experiments were conducted with the dataset, similar to the ones with the KDD Cup 1999 dataset. The first experiment (*Exp. 3*) was designed to test our evolving connectionist systems. The masquerade examples randomly drawn from the 13 users were embedded into the 7 users' login sessions with a 3% probability. In the second experiment (*Exp. 4*), in order to compare the performance of SVM, Fuzzy ART and EFuNN, the 7 normal users' login sessions were split into two parts. The first half of the login sessions were used for training the learning systems, and the remaining for testing. Then the masquerade examples were inserted in the testing datasets with a probability of 6%. The values of the masquerade probability were chosen so that there was at least one masquerade example in each user's testing dataset for both experiments. Table VI shows the numbers of the 7 users' login sessions and the corresponding masquerade examples for each experiment. The user IDs are user identification numbers inherited from the original dataset. We built learning models for each of the 7 individual users to see if they could identify the masquerade examples hiding in their testing datasets.

2) Results: For Fuzzy ART and EFuNN, when a testing instance is labeled *anomalous* and a *Level 2 alarm* is generated, it is either a true positive (i.e., a hit) if the instance represents a masquerade example, or a false positive if the instance is the user's own login session. We varied *vigilance* ρ's value from 0.89 to 0.99 and *learning rate* β's values from 0.1 to 1.0, respectively. The choice of N_{watch} and Min_{count} depends on the profiled individual user. For simplicity, we set N_{watch} to 3 and Min_{count} to 2 for all users, which were found to be an acceptable compromise. For each of the 7 users, we report the parameter settings (ρ and β) that give the lowest cost in Table VII for *Exp. 3* and Table VIII for *Exp. 4*.

In *Exp. 3*, both Fuzzy ART and EFuNN were able to model user behavior starting from an empty set of normal patterns and still recognize the majority of the masquerade instances, while the false positive rate was under 5%. EFuNN performed slightly better than Fuzzy ART because EFuNN provided higher hit rate and lower false positive rate and thus lower overall cost.

Table VIII shows the performance comparison of static learning with SVM and adaptive learning with Fuzzy ART and EFuNN in *Exp. 4*. The parameter ν of SVM was again determined by cross validation with the training data sets. SVM generated 42 false alarms (15.7% false positive rate) due to the concept drift between the training dataset and the testing dataset. The false positive rates of Fuzzy ART and EFuNN were significantly less because they were able to adapt to user behavior changes incrementally, while the hit rates were comparable to that of SVM. Therefore the overall cost of Fuzzy ART and EFuNN was largely reduced.

V. DISCUSSION

Our approach assumes that the number of normal instances vastly outnumbers the number of anomalies, and the anomalous activities appear as outliers in the data. This approach would miss the attacks or masquerades if the underlying assumptions do not hold. For example, some DoS attacks would not be identified by our adaptive anomaly detection systems. Nevertheless, our anomaly detection framework can be easily extended

TABLE VI
NUMBER OF LOGIN SESSIONS AND THE MASQUERADE EXAMPLES.

	Exp. 3		*Exp. 4*		
UserID	self sessions	masquerades	training	testing self sessions	masquerades
1	184	7	92	92	7
2	54	1	27	27	2
4	45	2	22	23	2
6	55	2	27	28	3
7	50	2	25	25	2
14	58	2	29	29	3
19	87	6	43	44	6
Total	527	22	259	268	25

TABLE VII
THE PERFORMANCE OF FUZZY ART AND EFUNN IN *Exp. 3*. $N_{watch} = 3$, $Min_{count} = 2$.

	Fuzzy ART				EFuNN			
UserID	ρ	β	false positives	hits	ρ	β	false positives	hits
1	0.95	0.4	5	3	0.94	0.2	7	5
2	0.92	0.8	6	1	0.91	0.8	5	1
4	0.90	0.4	1	1	0.89	0.2	2	2
6	0.91	0.4	1	2	0.90	0.4	0	2
7	0.90	0.6	4	1	0.90	0.2	0	1
14	0.93	0.8	7	1	0.90	0.4	2	1
19	0.91	1.0	0	6	0.89	1.0	2	6
Total			24	15			18	18
Overall	false positive rate $= 24/527 = 4.55\%$ hit rate $= 15/22 = 68.2\%$ Cost = 0.591				false positive rate $= 18/527 = 3.42\%$ hit rate $= 18/22 = 81.8\%$ Cost = 0.387			

TABLE VIII
THE PERFORMANCE OF SVM, FUZZY ART AND EFUNN IN *Exp. 4*.

	SVM		Fuzzy ART				EFuNN			
UserID	false positives	hits	ρ	β	false positives	hits	ρ	β	false positives	hits
1	11	6	0.92	1.0	6	7	0.96	0.1	2	6
2	7	1	0.91	1.0	0	1	0.90	1.0	0	1
4	3	2	0.91	0.6	0	2	0.91	0.8	0	1
6	3	2	0.89	0.8	1	2	0.94	0.6	1	3
7	8	2	0.89	0.6	1	1	0.90	0.6	1	1
14	6	1	0.91	1.0	2	1	0.91	0.4	3	1
19	4	6	0.91	0.4	0	6	0.92	0.2	0	6
Total	42	20			10	20			7	19
Overall	false positive rate $= 42/268 = 15.7\%$ hit rate $= 20/25 = 80.0\%$ Cost = 1.14		false positive rate $= 10/268 = 3.73\%$ hit rate $= 20/25 = 80.0\%$ Cost = 0.424				false positive rate $= 7/268 = 2.61\%$ hit rate $= 19/25 = 76.0\%$ Cost = 0.397			

to incorporate signature detection. Previously learned patterns can be labeled in such a way that certain patterns may generate an alert no matter how frequently they are observed, while other patterns do not trigger an alarm even if they are rarely seen [21].

with our adaptive anomaly detection framework, it is possible that one can deliberately cover his malicious activities by slowly changing his behavior patterns without triggering a *level2 alarm*. However, a *level1 alarm* is issued whenever a new pattern is being formed. It is then the security officer's responsibility to identify the user's intent in order to distinguish malicious from non-malicious anomalies, which is beyond the scope of this paper.

VI. Conclusion

This paper has presented a new adaptive anomaly detection framework through the use of evolving connectionist systems. A subject's normal behavior is learned in the online unsupervised mode. The performance of two adaptive anomaly detection systems, based on Fuzzy ART and EFuNN, was empirically tested with the KDD Cup 1999 network data and the user profiling data. The experiments have shown that our adaptive anomaly detection systems are able to adapt to user or network behavior changes while still recognizing anomalous activities. Compared to the SVM based static learning, the adaptive anomaly detection methods can significantly reduce the false alarms.

In order to make an adaptive anomaly detection system scalable, it might be necessary to prune or aggregate pattern nodes as the system evolves, which is a significant issue for our future work. Other issues of our future work include exploring automated determination of the parameters and comparing more evolving connectionist systems, such as evolving self-organizing maps.

Appendix

A. Fuzzy ART

Fuzzy ART [7] is a member of the Adaptive Resonance Theory (ART) neural network family [26]. It incorporates computations from fuzzy set theory [22] into the ART 1 neural network. It is capable of fast stable unsupervised category learning and pattern recognition in response to arbitrary input sequences.

Fuzzy ART clusters input vectors into patterns based on two separate distance criteria, *match* and *choice*. For input vector X and pattern j, the *match* function is defined by

$$S_j(X) = \frac{|X \wedge W_j|}{|X|},$$

where W_j is the weight vector associated with pattern j. Here, the fuzzy AND operator $\wedge$ is defined by

$$(X \wedge Y)_i \equiv min(x_i, y_i),$$

and the norm $|\cdot|$ is defined by

$$|X| \equiv \sum_i |x_i|.$$

The *choice* function is defined by

$$T_j(X) = \frac{|X \wedge W_j|}{\alpha + |W_j|},$$

where α is a small constant.

For each input vector X, Fuzzy ART assigns it to the pattern j that maximize $T_j(X)$ while satisfying $S_j(X) \geq \rho$, where ρ is the *vigilance* parameter, $0 \leq \rho \leq 1$. The weight vector W_j is then updated according to the equation

$$W_j^{(new)} = \beta(X \wedge W_j^{(old)}) + (1-\beta)W_j^{(old)},$$

where β is the *learning rate* parameter, $0 \leq \beta \leq 1$. If no such pattern can be found, a new pattern node is created. This procedure is illustrated in Figure 3.

In order to avoid the pattern proliferation problem, Fuzzy ART uses a *complement coding* technique to normalize the inputs. The complement of vector X, denoted by X^c, is defined by

$$(X^c)_i \equiv 1 - x_i.$$

For an n-dimensional original input X, the complemented coded input X' to the Fuzzy ART system is the $2n$-dimensional vector

$$X' = (X, X^c) \equiv (x_1, x_2, ..., x_n, x_1^c, x_2^c, ..., x_n^c).$$

B. EFuNN

EFuNN is one of the evolving connectionist systems developed by Kasabov [8] that is capable of modeling evolving processes through incremental, online learning. It has been successfully applied to bio-informatics, speech and image recognition [6]. The original EFuNN has a five-layer structure. Here we only use its first three layers for unsupervised learning (Figure 1(b)).

The fuzzy input layer transfers the original input values into membership degrees with a membership function attached to the fuzzy input nodes. The membership function can be triangular, Gaussian, and so on. The number and the type of the membership function can be dynamically modified during the evolving process. In this research we used the triangular membership function.

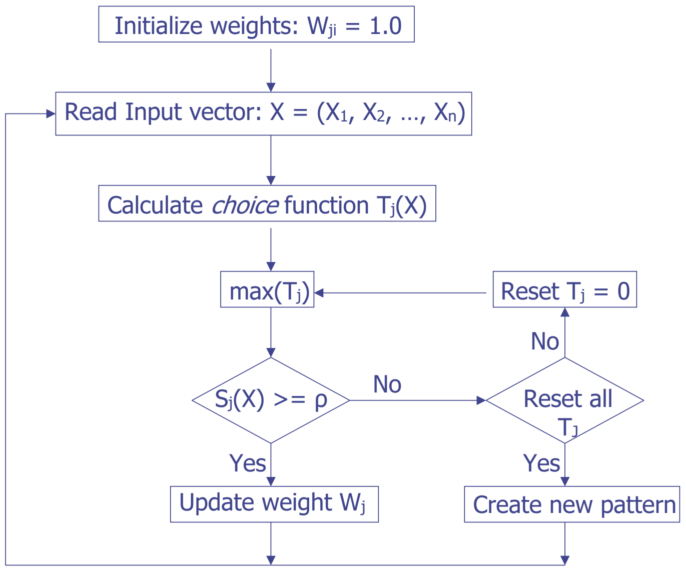

Initialize weights: W_ji = 1.0
Read Input vector: X = (X1, X2, ..., Xn)
Calculate *choice* function Tj(X)
max(Tj)
Reset Tj = 0
No
Sj(X) >= ρ
No
Reset all Tj
Yes
Yes
Update weight Wj
Create new pattern

Fig. 3. Flow graph representation of the Fuzzy ART algorithm.

Unlike Fuzzy ART, EFuNN groups input vectors into patterns based on one distance measure only, the local normalized fuzzy distance between a fuzzy input vector X_f and a weight vector W_j associated with pattern j, which is defined by

$$D(X_f, W_j) = \frac{|X_f - W_j|}{|X_f + W_j|},$$

where $|\cdot|$ denotes the same vector norm defined in Fuzzy ART. The local normalized fuzzy distance between any two fuzzy membership vectors is within the range of $[0, 1]$.

The rest of the clustering algorithm of EFuNN is very similar to that of Fuzzy ART. When a new input vector X is presented, EFuNN calculates the corresponding fuzzy input vector X_f and evaluates the normalized fuzzy distance between X_f and the existing pattern weight vectors. The activation of the pattern node layer A is then calculated. The activation of a single pattern node j is defined by

$$A(j) = f(D(X_f, W_j)),$$

where f can be a simple linear function, for example, $A(j) = 1 - D(X_f, W_j)$. EFuNN finds the closest pattern node j to the fuzzy input vector that has the highest activation value $A(j)$. If $A(j) \geq \rho$, where ρ is the *vigilance* parameter (the original EFuNN paper named it *sensitivity threshold* [8]), the new input is assigned to the jth pattern and the weight vector W_j is updated according to the following vector operation:

$$W_j^{(new)} = W_j^{(old)} + \beta(X_f - W_j^{(old)}),$$

where β is the *learning rate*. Otherwise, a new pattern node is created to accommodate the current instance X.

The parameters ρ and β can be static, or they can be self-adjustable while the structure of EFuNN evolves. They can hold the same values for all the patterns, or they can be pattern-specific so that the pattern node that has more instance members will change less when it accommodates a new instance. In our early implementation, all the pattern nodes share the same static ρ and β values.

Acknowledgment

The authors wish to thank Dr. Tom Goldring of NSA for providing us the Windows NT user profiling data set. This work is supported in part by the AFOSR grant F49620-01-1-0327 to the Center for Digital Security of the University of California, Davis.

References

[1] J. McHugh, "Intrusion and intrusion detection," *International Journal of Information Security*, no. 1, pp. 14–35, 2001.
[2] T. F. Lunt, "Detecting intruders in computer systems," in *Proceedings of Conference on Auditing and Computer Technology*, 1993. [Online]. Available: http://www.sdl.sri.com/nides/index5.html
[3] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in *Proceedings of International Conference on Dependable Systems & Networks*, Washington DC, June 2002, pp. 219–228.
[4] W. Lee, S. Stolfo, and P. Chan, "Learning patterns from unix process execution traces for intrusion detection," in *Proceedings of the AAAI97 workshop on AI methods in Fraud and risk management*, July 1997, pp. 50–56.

[5] C. Warrender, S. Forrest, and B. Pearlmutter, “Detecting intrusions using system calls: Alternative data models,” in *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 1999, pp. 133–145.

[6] N. Kasabov, *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligence Machines*. Springer, 2002.

[7] G. A. Carpenter, S. Grossberg, and D. B. Rosen, “Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system,” *Neural Networks*, vol. 4, pp. 759–771, 1991.

[8] N. Kasabov, “Evolving fuzzy neural networks for supervised/unsupervised on-line, knowledge-based learning,” *IEEE Trans. on Man, Machine and Cybernetics–Part B: Cybernetics*, vol. 31, no. 6, pp. 902–918, Dec. 2001.

[9] (1999) The third international knowledge discovery and data mining tools competition (kdd cup 1999) data. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[10] H. S. Teng, K. Chen, and S. C. Lu, “Adaptive real-time anomaly detection using inductively generated sequential patterns,” in *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 1990, pp. 278–284.

[11] T. Lane and C. Brodley, “Approaches to online learning and conceptual drift for user identification in computer security,” ECE and the COAST Laboratory, Purdue University, Tech. Rep. Coast TR 98-12, 1998.

[12] E. Eskin, M. Miller, Z.-D. Zhong, G. Yi, W.-A. Lee, and S. Stolfo, “Adaptive model generation for intrusion detection systems,” in *Proceedings of the ACMCCS Workshop on Intrusion Detection and Prevention*, Athens, Greece, 2000.

[13] K. Yamanishi, J. Takeuchi, and G. Williams, “On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms,” in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, Aug. 2000, pp. 320–324.

[14] J. Cannady, “Next generation intrusion detection: Autonomous reinforcement learning of network attacks,” in *Proceedings of the 23rd National Information Systems Security Conference*, 2000.

[15] W. Fan, “Cost-sensitive, scalable and adaptive learning using ensemble-based methods,” Ph.D. dissertation, Columbia University, Feb. 2001.

[16] M. Hossain and S. M. Bridges, “A framework for an adaptive intrusion detection system with data mining,” in *Proceedings of the 13th Annual Canadian Information Technology Security Symposium*, Ottawa, Canada, June 2001.

[17] H. Debar, M. Becker, and D. Siboni, “A neural network component for an intrusion detection system,” in *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 1992, pp. 240–250.

[18] A. K. Ghosh and A. Schwartzbard, “A study in using neural networks for anomaly and misuse detection,” in *Proceedings of the Eighth USENIX Security Symposium*, 1999.

[19] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, “A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data,” in *Applications of Data Mining in Computer Security*, D. Barbara and S. Jajodia, Eds. Kluwer, 2002.

[20] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, “A comparative study of anomaly detection schemes in network intrusion detection,” in *Proceedings of the 3rd SIAM International Conference on Data Mining*, San Francisco, California, 2003.

[21] A. Valdes, “Detecting novel scans through pattern anomaly detection,” in *Proceedings of DARPA Information Survivability Conference and Exposition*, Washington, DC, Apr. 2003, pp. 140–151.

[22] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A computational approach to learning and machine intelligence*. Prentice Hall, 1997.

[23] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[24] B. Scholkopf, J. C. Platt, J. Schawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[25] C.-C. Chang and C.-J. Lin. (2001) LIBSVM: a library for support vector machines. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[26] G. A. Carpenter and S. Grossberg, Eds., *Pattern Recognition by Self-Organizing Neural Networks*. Cambridge, MA: MIT Press, 1991.