Chapter from the book *Machine Learning*
Downloaded from: http://www.intechopen.com/books/machine_learning

Machine Learning for Sequential Behavior Modeling and Prediction

Xin Xu
Institute of Automation, National University of Defense Technology, Changsha, 410073, China

1. Introduction

In the information era, as computer networks and related applications become more and more popular, security problems are more and more serious in global information infrastructure. It was reported that in the past two years, large amounts of network attacks and computer viruses caused great damages to global economy and the potential threats to the global information infrastructure have increased a lot. To defend various cyber attacks and computer viruses, lots of computer security techniques have been studied, which include cryptography, firewalls and intrusion detection, etc. As an important computer security technique, intrusion detection [1,2] has been considered to be more promising for defending complex computer attacks than other techniques such as cryptography, firewalls, etc. The aim of intrusion detection is to find cyber attacks or non-permitted deviations of the characteristic properties in a computer system or monitored networks. Thus, one of the central problems for intrusion detection systems (IDSs) is to build effective behavior models or patterns to distinguish normal behaviors from abnormal behaviors by observing collected audit data. To solve this problem, earlier IDSs usually rely on security experts to analyze the audit data and construct intrusion detection rules manually [2]. However, since the amount of audit data, including network data, process execution traces and user command data, etc., increases vary fast, it becomes a time-consuming, tedious and even impossible work for human experts to analyze dynamic, huge volumes of audit data and extract attack signatures or detection rules. Furthermore, detection rules constructed by human experts are usually based on fixed features or signatures of existing attacks, so it will be very difficult for these rules to detect deformed or even completely new attacks.

According to the differences in the monitored data, IDSs can be mainly classified into two categories, i.e., network-based intrusion detection and host-based intrusion detection. Network-based intrusion detection observes data from network packets and extracts various features from them, which usually include connection features, traffic features, and content features. A systematic discussion on feature representation in network-based intrusion detection can be found in [3]. For host-based intrusion detection, various observation data from the corresponding operation systems are collected, which mainly include system call data and shell command data [4], etc. Despite of having different observation data, both host-based and network-based intrusion detection need to improve the detection accuracy for large volumes and variability of normal and attack behaviors. Aiming at this problem,

Source: Machine Learning, Book edited by: Abdelhamid Mellouk and Abdennacer Chebira,
ISBN 978-3-902613-56-1, pp. 422, February 2009, I-Tech, Vienna, Austria

lots of research work has been devoted to develop intrusion detection systems (IDSs) using various artificial intelligence (AI) methods and tools [3-5]. Thus, the motivations for applying AI techniques in IDSs are due to large amounts of dynamic behaviors and the lack of *a priori* knowledge for unknown attacks. How to establish appropriate behavior models has been a central problem in the development of IDSs since the distinctions between normal behaviors and computer attacks are usually very vague. In earlier research on IDSs, it was very popular to separately construct behavior models either for normal usages or attacks. To model intrusion behaviors alone is called misuse detection and anomaly detection refers to establish profiles of normal usages. In misuse detection, behavior patterns or models of known attacks are constructed and alarms are raised when the patterns of observation data match the attack models. On the other hand, anomaly detection only models the patterns of normal behaviors and detects any possible attacks as deviations from the normal behavior model. Until now, although there have been many advances in misuse detection and anomaly detection, some significant challenges still exist to meet the requirements of defending computer systems from attacks with increasing complexity, intelligence, and variability. For misuse detection, the inability of detecting new attacks is its inevitable weakness and it is very hard to improve the performance of pure misuse detection systems for the sake of increasing amounts of novel attacks. Although anomaly detection has the ability of detecting new attacks, it usually suffers from high rates of false alarms since it is very difficult to obtain a complete model of normal behaviors.

To solve the above problems in IDSs, machine learning and data mining methods for intrusion detection have received a lot of research interests in recent years [4-10]. One motivation for applying machine learning and data mining techniques in IDSs is to construct and optimize detection models automatically, which will eliminate the tedious work of human experts for data analysis and model building in earlier IDSs. To detect novel attacks, several adaptive anomaly detection methods were proposed by employing data mining methods based on statistics [7], or clustering techniques [10]. Recently, there have been several efforts in designing anomaly detection algorithms using supervised learning algorithms, such as neural networks [8], support vector machines [11], etc. In addition to supervised or inductive learning methods for misuse and anomaly detection, another approach to adaptive intrusion detection is to use unsupervised learning methods. Unlike supervised learning methods, where detection models are constructed by careful labeling of normal behaviors, unsupervised anomaly detection tries to detect anomalous behaviors with very little a priori knowledge about the training data. However, as studied in [12], the performance of pure unsupervised anomaly detection approaches is usually unsatisfactory, e.g., it was demonstrated in [12] that supervised learning methods significantly outperform the unsupervised ones if the test data contains no unknown attacks.

Despite of many advances that have been achieved, existing IDSs still have some difficulties in improving their performance to meet the needs of detecting increasing types of attacks in high-speed networks. One difficulty is to improve detection abilities for complex or new attacks without increasing false alarms. Since misuse IDSs employ signatures of known attacks, it is hard for them to detect deformed attacks, notwithstanding completely new attacks. On the other hand, although anomaly detection can detect new types of attacks by constructing a model of normal behaviors, the false alarm rates in anomaly-based IDSs are usually high. How to increase the detecting ability while maintaining low false alarms is still an open problem of IDS research.

In addition to the ability of realizing automatic model construction for misuse detection and anomaly detection, another promising application of machine learning methods in intrusion detection is to build dynamic behavior modeling frameworks which can combine the advantages of misuse detection and anomaly detection while eliminate the weakness of both. Many previous results on misuse detection and anomaly detection were usually based on static behavior modeling, i.e., normal behaviors or attack behaviors were modeled as static feature patterns and the intrusion detection problem was transformed to a pattern matching or classification procedure. However, dynamic behavior modeling is different from static behavior modeling approaches in two aspects. One aspect is that the relationships between temporal features are explicitly modeled in dynamic modeling approaches while static modeling only considers time independent features. The other aspect is that probabilistic frameworks are usually employed in dynamic behavior models while most static models make use of deterministic decision functions. Furthermore, many complex attacks are composed of multiple stages of behaviors, for example, a remote-to-local (R2L) attack commonly performs probe attacks to find target computers with vulnerabilities at first, and later realizes various buffer overflow attacks by utilizing the vulnerabilities in the target host computers. Therefore, sequential modeling approaches will be more beneficial to precisely describe the properties of complex multi-stage attacks. In [4], dynamic behavior modeling and static behavior modeling approaches were discussed and compared in detail, where a Hidden Markov Model was proposed to establish dynamic behavior models of audit data in host computers including system call data and shell command data. It was demonstrated in [4] that dynamic behavior modeling is more suitable for sequential data patterns such as system call data of host computers. However, the main difficulty for applying HMMs in real-time IDS applications is that the computational costs of HMM training and testing increase very fast with the number of states and the length of observation traces.

In this Chapter, some recently developed machine learning techniques for sequential behavior modeling and prediction are studied, where adaptive intrusion detection in computer systems is used as the application case. At first, a general framework for applying machine learning to computer intrusion detection is analyzed. Then, reinforcement learning algorithms based on Markov reward models as well as previous approaches using Hidden Markov Models (HMMs) are studied for sequential behavior modeling and prediction in adaptive intrusion detection. At last, the performance of different methods are evaluated and compared.

2. A general framework of ML applications in intrusion detection

In [9], based on a comprehensive analysis for the current research challenges in intrusion detection, a framework for adaptive intrusion detection using machine learning techniques was presented, which is shown in Fig.1. The framework is composed of three main parts. The first one is for data acquisition and feature extraction. Data acquisition is realized by a data sensing module that observes network flow data or process execution trajectories from network or host computers. After pre-processing of the raw data, a feature extraction module is used to convert the raw data into feature vectors that can be processed by machine learning algorithms and an extraction model based on unsupervised learning can be employed to extract more useful features or reduce the dimensionality of feature vectors. This process for automated feature extraction is a component of the machine learning part in the framework. In the machine learning part, audit data for training are stored in databases and they can be dynamically updated by human analysts or by machine learning algorithms. The third part in the framework depicted in Fig.1 is for real-time detection, which is to make use of the detection models as well as the extracted feature vectors to determine whether an observed pattern or a sequence of patterns is normal or abnormal.

To automatically construct detection models from the audit data, various machine learning methods can be applied, which include unsupervised learning, supervised learning and reinforcement learning. In addition, there are three perspectives of research challenges for intrusion detection, which include feature extraction, classifier construction and sequential behavior prediction. Although various hybrid approaches may be employed, it was illustrated that these three perspectives of research challenges are mainly suitable for machine learning methods using unsupervised, supervised and reinforcement learning algorithms, respectively. In contrast, in the previous adaptive IDS framework in [13], feature selection and classifier construction of IDSs were mainly tackled by traditional association data mining methods such as the *Apriori* algorithm.

2.1 Feature extraction

As illustrated in Fig.1, feature extraction is the basis for high-performance intrusion detection using data mining since the detection models have to be optimized based on the selection of feature spaces. If the features are improperly selected, the ultimate performance of detection models will be influenced a lot. This problem has been studied during the early work of W.K. Lee and his research results lead to the benchmark dataset KDD99 [13-14], where a 41-dimensional feature vector was constructed for each network connection. The feature extraction method in KDD99 made use of various data mining techniques to identify some of the important features for detecting anomalous connections. The features employed in KDD99 can serve as the basis of further feature extraction.

In KDD99, there are 494,021 records in the 10% training data set and the number of records in the testing data set is about five million, with a 10 percent testing subset of 311028 records. The data set contains a total of 22 different attack types. There are 41 features for each connection record that have either discrete values or continuous values. The 41-dimensional feature can be divided into three groups. The first group of features is called basic or intrinsic features of a network connection, which include the duration, prototype, service, number of bytes from source IP addresses or from destination IP addresses, and some flags in TCP connections. The second group of features in KDD99 is composed of the content features of network connections and the third group is composed of the statistical features that are computed either by a time window or a window of certain kind of connections.

The feature extraction method in the KDD99 dataset has been widely used as a standard feature construction method for network-based intrusion detection. However, in the later work of other researchers, it was found that the 41-dimensional features are not the best ones for intrusion detection and the performance of IDSs may be further improved by studying new feature extraction or dimension reduction methods [11]. In [11], a dimension reduction method based on principal component analysis (PCA) was developed so that the classification speed of IDSs can be improved a lot without much loss of detection precision.

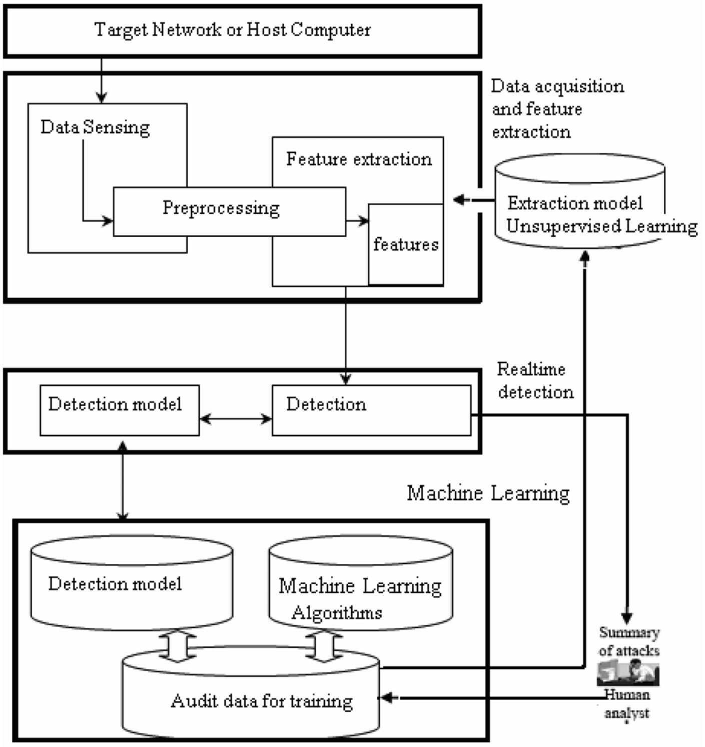


Fig. 1. A framework for adaptive IDSs based on machine learning

2.2 Classifier construction

After performing feature extraction of network flow data, every network connection record can be denoted by a numerical feature vector and a class label can be assigned to the record, i.e.,

$$\{(\vec{x}_i, y_i)\}, \quad i = 1,2,...,N \qquad y_i \in \{1,2,...,m\}$$

For the extracted features of audit data such as KDD99, when labels were assigned to each data record, the classifier construction problem can be solved by applying various supervised learning algorithms such as neural networks, decision trees, etc. However, the classification precision of most existing methods needs to be improved further since it is very difficult to detect lots of new attacks by only training on limited audit data. Using anomaly detection strategy can detect novel attacks but the false alarm rate is usually very high since to model normal patterns very well is also hard. Thus, the classifier construction in IDSs remains another technical challenge for intrusion detection based on machine learning.

2.3 Sequential behavior prediction

As discussed above, host-based IDSs are different from network-based IDSs in that the observed trajectories of processes or user shell commands in a host computer are sequential patterns. For example, if we use system call traces as audit data, a trajectory of system calls can be modeled as a state transition sequence of short sequences. In the following Fig. 2, it is shown that every state is a short sequence of length 3 and different system call traces can form different state transitions, where *a*, *b*, and *c* are symbols for system calls in a host computer.

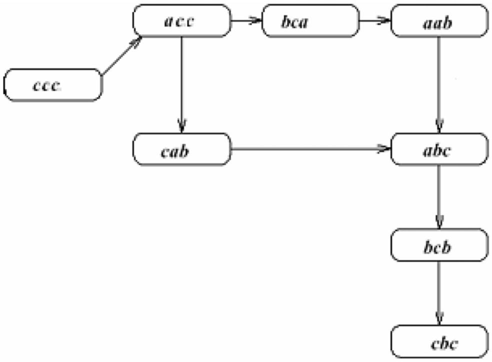


Fig. 2. A sequential state transition model for host-based IDSs

Therefore, the host-based intrusion detection problem can be considered as a sequential prediction problem since it is hard to determine a single short sequence of system calls to be normal and normal and there are intrinsic temporal relationships between sequences. Although we can still transform the above problem to a static classification problem by mapping the whole trace of a process to a feature vector [15], it has been shown that dynamic behavior modeling methods, such as Hidden Markov Models (HMMs) [4], are more suitable for this kind of intrusion detection problem. In the following, a host-based intrusion detection method will be studied based on reinforcement learning, where a Markov reward model is established for sequential pattern prediction and temporal difference (TD) algorithms [16] are used to realize high-precision prediction without many computational costs. At first, the popular HMMs for sequential behavior modeling will be introduced in the next section.

3. Hidden Markov Models (HMMs) for sequential behavior modeling

Due to the large volumes of audit data, to establish and modify detection models manually by human experts becomes more and more impractical. Therefore, machine learning and data mining methods have been widely considered as important techniques for adaptive intrusion detection, i.e., to construct and optimize detection models automatically. Previous work using supervised learning mainly focused on static behavior modeling methods based on pre-processed training data with class labels. However, training data labeling is one of the most important and difficult tasks since it is hard to extract signatures precisely even for known attacks and there are still increasing amounts of unknown attacks. In most of the previous works using static behavior modeling and supervised learning algorithms, every single sample of the training data was either labeled as normal or abnormal. However, the distinctions between normal and abnormal behaviors are usually very vague and improper labeling may limit or worsen the detection performance of supervised learning methods.

More importantly, for complex multi-stage attacks, it is very difficult or even impossible for static behavior models based on supervised learning to describe precisely the temporal relationships between sequential patterns. The above problems become the main reasons leading to the unsatisfactory performance of previous supervised learning approaches to adaptive IDSs, especially for complex sequential data. The recent works on applying HMMs [4] and other sequence learning methods [17] have been focused on dynamic behavior modeling for IDSs, which tried to explicitly estimate the probabilistic transition model of sequential patterns. For the purpose of comparisons, in the following, a brief introduction on HMM-based methods for intrusion detection will be given.

As a popular sequential modeling approach, HMMs have been widely studied and applied in lots of areas such as speech recognition [18], protein structure prediction, etc. A discrete state, discrete time, first order hidden Markov model describes a stochastic, memory-less process. A full HMM can be specified as a tuple: λ = (N, M, A, B, π), where N is the number of states, M is the number of observable symbols, A is the state transition probability matrix which satisfies the Markov property:

$$a_{ij} = P(q_{t+1} = j|q_t = i) = P(q_{t+1} = j|q_t = i, q_{t-1}, \dots, q_0) \tag{1}$$

B is the observation probability distribution

$$b_j(k) = P(o_t = k|q_t = j)\ ,\ i \leqslant k \leqslant M \tag{2}$$

and π is the initial state distribution. The initial state distribution π satisfies:

$$\pi_i = P(x_0 = i) \tag{3}$$

$$0 \le \pi_i \le 1 \tag{4}$$

$$\sum_{i=1}^{N} \pi_i = 1 \tag{5}$$

For discrete state HMMs, we can let $Q = \{q_1, q_2, \dots, q_M\}$ denote the set of all states, $O = \{O_1, O_2, \dots, O_N\}$ denote the set of all observation symbols. A typical trace of HMMs is shown in the following Fig.3, where O_i (i=1,2,…,T) are observation symbols and q_i (i=1,2,…,T) are the corresponding states.

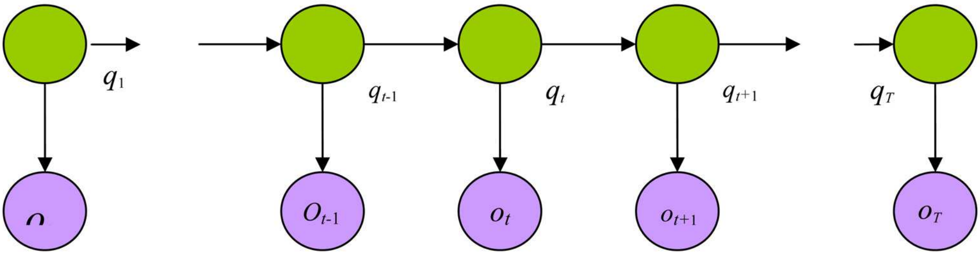


Fig. 3. An HMM model

In practice, there might be *a priori* reasons to assign certain values to each of the initial state probabilities. For example, in some applications, one typically expects HMMs to start in a particular state. Thus, one can assign probability one to that state and zero to others.

For HMMs, there are two important algorithms to compute the data likelihood when the model of an HMM is given. One algorithm is the Forward-Backward algorithm which calculates the incomplete data likelihood and the other is the Viterbi algorithm which calculates the complete data likelihood. Implicitly, both Forward-Backward and Viterbi find the most likely sequence of states, although differently defined. For detailed discussion on the two algorithms, please refer to [8].

Another important problem in HMMs is the model learning problem which is to estimate the model parameters when the model is unknown and only observation data can be obtained. The model learning problem is essential for HMMs to be applied in intrusion detection since a detection model must be constructed only by training data samples. For model learning in HMMs, the Expectation-Maximization (EM) algorithm is the most popular one which finds maximum a posteriori or maximum likelihood parameter estimate from incomplete data. The Baum-Welch algorithm is a particular form of EM for maximum likelihood parameter estimation in HMMs. For a detailed discussion on HMMs, the readers may refer to [18].

In intrusion detection based on HMMs, the Baum-Welch algorithm can be used to establish dynamic behavior models of normal data and after the learning process is completed, attack behaviors can be identified as deviations from the normal behavior models.

4. Reinforcement learning for sequential behavior prediction

4.1 Intrusion detection using Markov reward model and temporal-difference learning

In HMM-based dynamic behavior modeling for intrusion detection, the probabilistic transition model of the IDS problem is explicitly estimated, which is computationally expensive when the number of states and the length of traces increase. In this Section, an alternative approach to adaptive intrusion detection will be presented. In the alternative approach, Markov state transition models are also employed but have an additional evaluative reward function, which is used to indicate the possibility of anomaly. Therefore, the intrusion detection problem can be tackled by learning prediction of value functions of a Markov reward process, which have been widely studied in the reinforcement learning community. To explain the principle of the RL-based approach to intrusion detection, the sequential behavior modeling problem in host-based IDSs using sequences of system calls is discussed in the following.

For host-based intrusion detection, the audit data are usually obtained by collecting the execution trajectories of processes or user commands in a host computer. As discussed in [19], host-based IDSs can be realized by observing sequences of system calls, which are related to the operating systems in the host computer. The execution trajectories of different processes form different traces of system calls. Each trace is defined as the list of system calls issued by a single process from the beginning of its execution to the end. If a state at a time step is defined as *m* successive system calls and a sliding window with length l is defined, the traces of system calls can be transformed to a state transition sequences and different traces correspond to different state transition sequences. For example, if we select a sequence of 4 system calls as one state and the sliding length between sequences is 1, the state transitions corresponding to a short trace *tr*={ open, read, mmap, mmap, open, read, mmap} are:

State *S*1: *open, read, mmap, mmap*

State *S*2: *read, mmap, mmap, open*

State S3: mmap, mmap, open, read

State *S*4: *mmap, open, read, mmap*

Then the state transition sequence of the above trace *tr* is:

$$S1 \to S2 \to S3 \to S4$$

As studied and verified in [4], dynamic behavior models for sequential pattern prediction are superior to static models when temporal relationships between feature patterns need to be described accurately. Different from the previous work in [4], where an HMM-based dynamic behavior modeling approach was studied, the following dynamic behavior modeling method for intrusion detection is based on learning prediction using Markov reward models. The method is focused on a learning prediction approach, which has been popularly studied in RL research [21-22], by introducing a Markov reward model of the IDS problem so that high accuracy and low computational costs can both be guaranteed [20]. Firstly, the Markov reward model for the IDS problem is introduced as follows.

Markov reward processes are popular stochastic models for sequential modeling and decision making. A Markov reward process can be denoted as a tuple {*S*, *R*, *P*}, where *S* is the state space, *R* is the reward function, *P* is the state transition probability. Let $\{x_t \mid t=0,1,2,\ldots;\ x_t \in S\}$ denote a trajectory generated by a Markov reward process. For each state transition from x_t to x_{t+1}, a scalar reward r_t is defined. The state transition probabilities satisfy the following Markov property:

$$P\{x_{t+1} | x_t, x_{t-1}, \ldots, x_1, x_0\} = P\{x_{t+1} | x_t\} \tag{6}$$

The reward function of the Markov reward plays an important role for dynamic behavior modeling in intrusion detection problems. As described in the following Fig.2, in a Markov reward model for intrusion detection based on system calls, each state is defined as a short sequence of successive system calls and after each state transition, a scalar reward r_t is given to indicate whether there is a possibility to be normal or attack behaviors. The design of the reward function can make use of available *a priori* information so that the anomaly probability of a whole state trajectory can be estimated based on the accumulated reward function. In one extreme case, we can indicate every state to be normal or abnormal with high confidence and the immediate reward of each state is designed as

$$r_t = \begin{cases} -1, & \textit{for normal state} \\ 1, & \textit{for abnormal state} \end{cases}, \qquad \text{for } t=1,2,\ldots T \tag{7}$$

The above extreme case is identical to transform the dynamic behavior modeling problem to a static pattern classification problem since we have class labels for every possible states, where the reward becomes a class label for every state. However, in fact, due to the sequential properties of system call data and the vague distinctions between normal traces and abnormal traces, it is usually not appropriate or even impossible to tell whether an intermediate state to be normal or abnormal definitely. Moreover, even if it is reasonable to assign precise class labels to every states, it is also very hard to obtain precise class labels for large amounts of audit data. Therefore, it is more reasonable to develop dynamic behavior modeling approaches which not only incorporate the temporal properties of state transitions but also need little *a priori* knowledge for class labeling. An extreme case toward this direction is to provide evaluative signals to a whole state transition trajectory, i.e., only a whole state trajectory is indicated to be normal or abnormal while the intermediate states are not definitely labeled. For example, in the following Fig.4, the reward at the terminal state r_T can be precisely given as:

$$r_T = \begin{cases} -1, & \textit{for normal trace} \\ 1, & \textit{for abnormal trace} \end{cases} \tag{8}$$

For intermediate states $s_1,\ldots, s_{T-1}$, a zero reward can be given to each state when there is no *a priori* knowledge about the anomaly of the states. However, in more general cases, the intermediate rewards can be designed based on available prior knowledge on some features or signatures of known attacks.

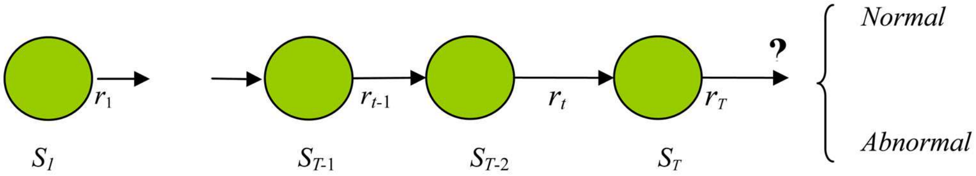


Fig. 4. A Markov reward process for intrusion detection

According to the above Markov reward process model, the detection of attack behaviors can be tackled by the sequential prediction of expected total rewards of a state in a trajectory since the reward signals, especially the terminal reward at the end of the trajectory provide information about whether the trajectory is normal or abnormal. Therefore, the intrusion detection problem becomes a value function prediction problem of a Markov reward process, which has been popularly studied by many researchers in the framework of reinforcement learning [21-24]. Among the learning prediction methods studied in RL, temporal difference learning (TD) is one of the most important one and in the following discussions, we will focus on the TD learning prediction algorithm for intrusion detection. Firstly, some basic definitions on value functions and dynamic programming are given as follows.

In order to predict the expected total rewards received after a state trajectory starting from a state x, the value function of state x is defined as follows:

$$V(x) = E\{\sum_{t=0}^{\infty} \gamma^t r_t | x_0 = x\} \tag{9}$$

where $x \in S$, $0 < \gamma \leq 1$ is the discount factor, r_t is the reward received after state transition $x_t \rightarrow x_{t+1}$ and $E\{.\}$ is the expectation over the state transition probabilities.

According to the theory of dynamic programming, the above value function satisfies the following Bellman equation.

$$V(s_t) = R_t + \gamma E[V(s_{t+1})] \tag{10}$$

where R_t is the expected reward received after state transition $x_t \rightarrow x_{t+1}$.

The aim of RL is to approximate the optimal or near-optimal policies from its experiences without knowing the parameters of this process. To estimate the optimal policy of an MDP, RL algorithms usually predict the value functions by observing data from state transitions and rewards. Thus, value function prediction of Markov reward models becomes a central problem in RL since optimal policies or optimal value functions can be obtained based on the estimation of value functions. However, in RL, learning prediction is more difficult that in supervised learning. As pointed out by Sutton [22], the prediction problems in supervised learning are single-step prediction problems while learning prediction in reinforcement learning belongs to multi-step prediction, which is to predict outcomes that depend on a future sequence of decisions.

Until now, temporal difference learning or TD learning has been considered as one of the most efficient approaches to value function prediction without any *a priori* model information about Markov reward processes. Different from supervised learning for sequential prediction such as Monte Carlo estimation methods, TD learning is to update the estimations based on the differences between two temporally successive estimations, which constitutes the main ideas of a popular class of TD learning algorithms called TD(λ) [22]. In TD(λ), there are two basic mechanisms which are the temporal difference and the eligibility trace, respectively. Temporal differences are defined as the differences between two successive estimations and have the following form

$$\delta_t = r_t + \gamma \widetilde{V}_t(x_{t+1}) - \widetilde{V}_t(x_t) \tag{11}$$

where x_{t+1} is the successive state of x_t, $\tilde{V}$ (x) denotes the estimate of value function $V(x)$ and r_t is the reward received after the state transition from x_t to x_{t+1}.

As discussed in [22], the eligibility trace can be viewed as an algebraic trick to improve learning efficiency without recording all the data of a multi-step prediction process. This trick is originated from the idea of using a truncated reward sum of Markov reward processes. In TD learning with eligibility traces, an n-step truncated return is defined as

$$R_t^n = r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma^n \widetilde{V}_t(s_{t+n}) \tag{12}$$

For an absorbing Markov reward process whose length is T, the weighted average of truncated returns is

$$R_t^\lambda = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^n + \lambda^{T-t-1} R_T \tag{13}$$

where $0 \le \lambda \le 1$ is a decaying factor and

$$R_T = r_t + \gamma r_{t+1} + \ldots + \gamma^T r_T \tag{14}$$

R_T is the Monte-Carlo return at the terminal state. In each step of TD(λ), the update rule of value function estimation is determined by the weighted average of truncated returns defined above, i.e.,

$$\Delta\tilde{V}_t(s_i) = \alpha_t(R_t^{\lambda} - \tilde{V}_t(s_i)) \tag{15}$$

where α_t is a learning factor.

The update equation (25) can be used only after the whole trajectory of the Markov reward process is observed. To realize incremental or online learning, eligibility traces are defined for each state as follows:

$$z_{t+1}(s_i) = \begin{cases} \gamma\lambda\, z_t(s_i) + 1, & \text{if } s_i = s_t \\ \gamma\lambda\, z_t(s_i), & \text{if } s_i \neq s_t \end{cases} \tag{16}$$

The online TD(λ) update rule with eligibility traces is

$$\tilde{V}_{t+1}(s_i) = \tilde{V}_t(s_i) + \alpha_t \delta_t z_{t+1}(s_i) \tag{17}$$

where δ_t is the temporal difference at time step t, which is defined in (21) and $z_0(s)=0$ for all s.

Based on the above TD learning prediction principle, the intrusion detection problem can be solved by a model learning process and an online detection process. In the model learning process, the value functions are estimated based on the online TD(λ) update rules and in the detection process, the estimated value functions are used to determine whether a sequence of states belongs to a normal trajectory or an abnormal trajectory. For the reward function defined in (18), when an appropriate threshold μ is selected, the detection rules of the IDS can be designed as follows:

If $V(x) > \mu$, then raise alarms for attacks,

Else there are no alarms.

Since the state space of a Markov reward process is usually large or infinite in practice, function approximators such as neural networks are commonly used to approximate the value function. Among the existing TD learning prediction methods, TD(λ) algorithms with linear function approximators are the most popular and well-studied ones, which can be called linear TD(λ) algorithms.

In linear TD(λ), consider a general linear function approximator with a fixed basis function vector

$$\phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_n(x))^T \tag{18}$$

The estimated value function can be denoted as

$$\tilde{V}_t(x) = \phi^T(x) W_t \tag{19}$$

where $W_t = (w_1, w_2, \ldots, w_n)^T$ is the weight vector.

The corresponding incremental weight update rule is

$$W_{t+1} = W_t + \alpha_t (r_t + \gamma \phi^T (x_{t+1}) W_t - \phi^T (x_t) W_t) \vec{z}_{t+1} \tag{20}$$

where the eligibility trace vector $\vec{z}_t(s) = (z_{1t}(s), z_{2t}(s), ..., z_{nt}(s))^T$ is defined as

$$\vec{z}_{t+1} = \gamma\lambda \vec{z}_t + \phi(x_t) \tag{21}$$

In [19], the above linear TD(λ) algorithm is proved to converge with probability 1 under certain assumptions and the limit of convergence W^* is also derived, which satisfies the following equation

$$E_0[A(X_t)]W^* - E_0[b(X_t)] = 0 \tag{22}$$

where $X_t = (x_t, x_{t+1}, z_{t+1})$ (t=1,2,…) form a Markov process, $E_0[\cdot]$ stands for the expectation with respect to the unique invariant distribution of $\{X_t\}$, and $A(X_t)$, $b(X_t)$ are defined as

$$A(X_t) = \vec{z}_t (\phi^T (x_t) - \gamma \phi^T (x_{t+1})) \tag{23}$$

$$b(X_t) = \vec{z}_t r_t \tag{24}$$

Then, based on a set of observation data $\{(x_t, r_t)\}$ (t=1,2,…,T), a least-squares solution to the above problem can be obtained as [24]:

$$W = (\sum_{t=1}^{T} A(X_t))^{-1} \sum_{t=1}^{T} b(X_t) \tag{25}$$

4.2 Kernel-based RL for sequential behavior learning

After introducing the above Markov reward model, the intrusion detection problem using system call traces can be solved by a class of reinforcement learning algorithms called temporal-difference (TD) learning. The aim of TD learning is to predict the state value functions of a Markov reward process by updating the value function estimations based on the differences between temporally successive predictions rather than using errors between the real values and the predicted ones. And it has been verified that TD learning is more efficient than supervised learning in multi-step prediction problems [22].

Until now, TD learning algorithms with linear function approximators have been widely studied in the literature [23-24]. In [24], a linear TD learning algorithm was applied to host-based intrusion detection using sequences of system calls and very promising results have been obtained. Nevertheless, the approximation ability of linear function approximators is limited and the performance of linear TD learning is greatly influenced by the selection of linear basis functions. In the following, a sparse kernel-based LS-TD(λ) algorithm will be presented for value function prediction in host-based IDSs [25]. The sparse kernel-based LS-TD algorithm was recently developed in [26] and it was demonstrated that by realizing least-squares TD learning in a kernel-induced high-dimensional feature space, nonlinear value function estimation can be implicitly implemented by a linear form of computation with high approximation accuracy. Therefore, by making use of the kernel-based LS-TD learning algorithm, the predictions of anomaly probabilities for intrusion detection will have higher precision and it will be more beneficial to realize high-performance IDSs based on dynamic behavior modeling.

In the kernel-based LS-TD learning method [26], the same solution to the following LS-TD problem was considered:

$$E_0[\vec{z}_t(\phi^T(x_t)-\gamma\phi^T(x_{t+1}))]W^* - E_0[\vec{z}_t r_t] = 0 \tag{26}$$

where the corresponding value functions are estimated by

$$V(x_t) = \phi^T(x_t)W^*$$

Using the average value of observations as the estimation of expectation $E_0[\cdot]$, equation (26) can be expressed as follows:

$$\sum_{i=1}^{N}[\vec{z}(s_i)(\phi^T(s_i)-\gamma\phi^T(s_{i+1})]\,W = \sum_{i=1}^{N}\vec{z}(s_i)r_i \tag{27}$$

Based on the idea of kernel methods, a high-dimensional nonlinear feature mapping can be constructed by selecting a Mercer kernel function $k(x_1, x_2)$ in a reproducing kernel Hilbert space (RKHS). In the following, the nonlinear feature mapping based on the kernel function $k(.,.)$ is also denoted by $\phi(s)$ and according to the Mercer Theorem [27], the inner product of two feature vectors is computed by

$$k(x_i, x_j) = \phi^T(x_i)\phi(x_j) \tag{28}$$

Due to the properties of RKHS [27], the weight vector W can be represented by the weighted sum of the state feature vectors:

$$W = \Phi_N\alpha = \sum_{t=1}^{N}\ \phi(s(x_t))\alpha_i \tag{29}$$

where x_i (i = 1,2,..., N) are the observed states, N is the total number of states and $\alpha = [\alpha_1, \alpha_2, ..., \alpha_N]^T$ are the corresponding coefficients, and the matrix notation of the feature vectors is denoted as

$$\Phi_N = (\phi(s_1), \phi(s_2), ..., \phi(s_N)) \tag{30}$$

For a state sequence x_i (i = 1, 2,..., N) , let the corresponding kernel matrix K be denoted as $K=(k_{ij})_{N\times N}$, where $k_{ij}=k(x_i, x_j)$.

$$\bar{Z}_N H_N K\alpha = \bar{Z}_N R_N \tag{31}$$

By substituting (28), (29) and (30) into (27), and multiplying the two sides of (27) with Φ_N^T we can get

$$\overline{Z}_N = [\overline{z}_1, \overline{z}_2, ..., \overline{z}_{N-1}] = [k_{1N}, k_{2N} + \gamma\lambda\overline{z}_1, ..., k_{(N-1)N} + \gamma\lambda\overline{z}_{N-2}] \tag{32}$$

$$R_N = [r_1, r_2, ..., r_{N-1}]^T \tag{33}$$

$$H_N = \begin{bmatrix} 1 & \beta_1\gamma & & & \\ & 1 & \beta_2\gamma & & \\ & & \cdots & & \\ & & & 1 & \beta_{N-1}\gamma \end{bmatrix}_{(N-1)\times N} \tag{34}$$

In (34), the values of β_i (i=1,2,…,N-1) are determined by the following rule: when state x_{i-1} is not an absorbing state, β_i is equal to -1, otherwise, β_i is set to zero.

As discussed in [26], by using the techniques of generalized inverse matrix in [28], the kernel-based LS-TD solution to (26) is as follows:

$$\alpha = (H_N K)^+ \overline{Z}_N^+ \overline{Z}_N R_N \tag{35}$$

where $(.)^+$ denotes the generalized inverse of a matrix.

One problem remained for the above kernel-based LS-TD learning algorithm is that the dimension of the kernel-based LS-TD solution is equal to the number of state transition samples, which will cause huge computational costs when the number of observation data is large. To make the above algorithm be practical, one key problem is to decrease the dimension of kernel matrix K as well as the dimensional of α. The problem has been studied in [29] by employing an approximately linear dependence (ALD) analysis method [30] for the sparsification of kernel matrix K.

The main idea of ALD-based sparcification is to represent the feature vectors of the original data samples by an approximately linearly independent subset of feature vectors, which is to compute the following optimization problem

$$\delta_t = \min_{\vec{a}} \left\| \sum_j \vec{a}_j \phi(x_j) - \phi(x_t) \right\|^2 \tag{36}$$

During the sparsification procedure, a data dictionary is incrementally constructed and every new data sample x_t is tested by compute the solution δ_t of (36). Only if δ_t is greater than a predefined threshold, the tested data sample x_t will be added to the dictionary. For detailed discussion of the sparsification process, please refer to [29] and [30]. After the sparsification procedure, a data dictionary D_N with reduced number of feature vectors will be obtained and the approximated state value function can be represented as:

$$\widetilde{V}(x) = \sum_{j=1}^{n(D_N)} \hat{\alpha}_j \hat{k}(x_j, x) \tag{37}$$

where $n(D_N)$ is the size of the dictionary.

When the above learning and sparcification process is completed, a value function model of the IDS problem can be obtained. And the accumulated anomaly probability of a state sequence $S_n=\{x_1, x_2, \ldots x_n\}$ can be computed as

$$P(s) = \frac{1}{n}\sum_{i=1}^{n}\widetilde{V}(x_i) \tag{38}$$

By selecting an appropriate threshold μ, the detection output of the adaptive IDS can be simply determined as follows:

$$If \quad P(S_n) > \mu \text{, then raise alarms}$$

4.3 Performance evaluations

Generally speaking, previous works on machine learning methods for adaptive intrusion detection can be mainly classified into four categories, i.e., supervised learning methods, unsupervised learning methods, semi-supervised methods and statistical modeling methods. Compared with the supervised learning methods in intrusion detection, the proposed model does not require precise labeling of every observed feature, which is a difficult task and may usually lead to the poor performance of supervised methods, especially for complex sequential attacks. For unsupervised learning algorithms in intrusion detection, e.g., SOM, clustering, due to the lack of prior information, the performance of IDSs can not be optimized adequately [12].

The proposed RL-based dynamic behavior modeling approach for intrusion detection estimates the anomaly probability of states based on the learning prediction of state value functions. Therefore, it can be applied to detect complex attack behaviors with complex sequential patterns. The computational complexity of TD learning algorithms is linear with respect to the number k of state features and the length m of traces, i.e., it has time complexity of $O(km)$, which is lower than the training algorithm for HMMs, which runs in time $O(nm^2)$, where n is the number of states in the HMM and m is the size of the trace. Furthermore, since TD learning prediction methods using function approximators are commonly used, the number k of state features can become much smaller than n and the computational efficiency will be further improved.

For the RL-based approach, the most related methods are based on Markov chain modeling or Hidden Markov models (HMMs), which are anomaly detection techniques that aim to establish the probabilistic structure model of the normal data sequences explicitly. However, the Markov reward model and the TD prediction method are based on hybrid modeling strategy where the intrusion data can be combined with normal data to train the detection model. Moreover, the RL-based method only implicitly constructs the probabilistic model and the detection of anomalies is based on the estimated value functions. In [31], the robustness of Markov chain modeling techniques was studied and it was shown that when explicitly estimating the probabilistic structure of the Markov chain model for normal data, the detection accuracy was very sensitive to the noise of data, i.e., when the intrusion data were mixed with normal data, the performance of the Markov chain model would become worse. Nevertheless, in our approach, the detection accuracy is not influenced by the mixing of normal and abnormal data due to the hybrid modeling strategy.

To compare the performance between the previous HMM-based approach and the RL-based approach, experiments on host-based intrusion detection using system calls were conducted. In the experiments, two types of data sets were used, which include system call traces from the "live" lpr and the Sendmail programs. Table 1 shows some of the details of the data, which include two kinds of attack data and corresponding normal data. All of these data sets are publicly available at the website of University of New Mexico [32].

In the data sets, each trace is a sequence of system calls generated by a single process from the beginning of its execution to the end. Since the traces were generated by different programs under different environments, the number of system calls per trace varies widely. In the MIT environment, lpr was traced by running the program on 77 different hosts, each running SunOS, for two weeks, to obtain traces of a total of 2766 print jobs. For detailed discussion of the properties of the data sets, please refer to [32-33].

The two types of system call traces were divided into two parts. One part is for model training and threshold determination and the other part is for performance evaluation. Table 1 shows the numbers of normal and attack traces for training and testing. As can be seen in the table, the numbers of testing traces are usually larger than those of testing traces.

		lpr	*sendmail*
Training & Threshold Selection	Normal Trace Number	10	13
	Attack Trace Number	20	5
Testing	Normal Trace Number	2703	67
	Attack Trace Number	1001	7
Total system call number		1023950	223733

Table 1. Experimental data for host-based IDS

During the threshold determination process, the same data sets were used as the training process, i.e., the training data sets and the data sets for threshold determination are the same. For performance testing, the data sets are different from those in model training and their sizes are usually larger than the training data. In the testing stage, two criterions for performance evaluations were used, which are the detection rate Dr and the false alarm or false positive rate Fp, and they are computed as follows:

$$Dr = \frac{n_d}{n_a} \tag{36}$$

$$Fp = \frac{N_a}{N} \tag{37}$$

where n_d is the number of abnormal traces that have been correctly identified by the detection model and n_a is the total number of abnormal traces, N_a is the number of normal states that have been incorrectly identified as anomaly by the detection model, and N is the total number of normal states. In the computation of false alarm rates, we use the same ideas discussed in [4], where every possible false alarms during a long state traces are all counted and the total sum of false alarms is divided by the number of all states in traces. Therefore, the false positives were measured differently from the detection rates or true positives. To detect an intrusion, it is only required that the anomaly probabilities exceed a preset threshold at some point during the intrusion. However, making a single decision as to whether a normal trace is abnormal or not is not sufficient, especially for very long traces. For example, if a program runs for several days or more, each time that it is flagged as anomalous must be counted separately. As pointed out in [17], the simplest way to measure this is to count all the individual decisions. Then, the false-positive rate is selected as the percentage of decisions in which normal data were detected as anomalous.

In the experiments, the TD learning prediction method was applied to the above data sets. Every state in the Markov reward model has a system-call sequence length of 6, which has been widely employed in previous works. The reward function is defined by (18). A linear function approximator, which is a polynomial function of the observation states and has a dimension of 24, was used as the value function approximator. To compare the performance of TD learning prediction and previous approaches, the experimental results in [4], where HMM-based dynamic behavior modeling methods were applied to the same data sets, are also shown in the following Table 2.

	TD Prediction		HMM [9]	
	Detection rate	False alarm rate	Detection rate	False alarm rate
lpr	100%	0.00749	100%	3e-4
sendmail	100%	0.002951	61.5%	0.05
			84.6%	0.10
			92.3%	0.20

Table 2. Performance comparisons between TD and HMM methods

To compare the performance between the kernel LS-TD approach with the linear LS-TD [16] and the HMM-based approach [4], experiments on host-based intrusion detection using system calls were conducted. In the experiments, the data set of system call traces generated from the *Sendmail* program was used. The system call traces were divided into two parts.

One part is for model training and threshold determination and the other part is for performance evaluation. The normal trace numbers for training and testing are 13 and 67, respectively. The numbers of attack traces used for training and testing are 5 and 7. The total number of system calls in the data set is 223733. During the threshold determination process, the same traces were used as the training process. The testing data are different from those in model training and their sizes are usually larger than the training data.

In the learning prediction experiments for intrusion detection, the kernel LS-TD algorithm and previous linear TD(λ) algorithms, i.e., LS-TD(λ), are all implemented for the learning prediction task. In the kernel-based LS-TD algorithm, a radius basis function (RBF) kernel is selected and its width parameter is set to 0.8 in all the experiments. A threshold parameter δ=0.001 is selected for the sparsification procedure of the kernel-based LS-TD learning algorithm. The LS-TD(λ) algorithm uses a linear function approximator, which is a polynomial function of the observation states and has a dimension of 24.

	Kernel LS-TD		LS-TD[11]		HMM [7]	
	Dr	*Fp*	*Dr*	*Fp*	*Dr*	*Fp*
sendmail	1.00	0.00016	1.00	0.0029	0.615	0.05*
					0.846	0.10*
					0.923	0.20*

* The false alarm rates were only computed for trace numbers, not for single state

Table 3. Performance comparisons between different methods

The experimental results are shown in Table 3. It can be seen from the results that both of the two RL methods, i.e., the kernel LS-TD and linear LS-TD, have 100% detection rates and the kernel-based LS-TD approach has better performance in false alarm rates than the linear LS-TD method. The main reason is due to the learning prediction accuracy of kernel-based LS-TD for value function estimation. It is also illustrated that the two TD learning prediction methods have much better performance than the previous HMM-based method. Therefore, the applications of kernel-based reinforcement learning methods, which are based on the Markov reward model, will be very promising to realize dynamic behavior modeling and prediction for complex multi-stage attacks so that the performance of IDSs can be efficiently optimized.

5. Conclusions

Although in recent years, there are many research works on applying machine learning and statistical modeling methods to intrusion detection problems, the sequential modeling problem in intelligent intrusion detection has not been well solved yet. In this Chapter, the TD learning prediction method is introduced to construct detection models and improve the performance of IDSs only by simplified labeling schemes using evaluative signals or feedbacks for sequential training data. It is illustrated that compared with previous anomaly detection approaches using machine learning, the TD learning and prediction method can obtain comparable or even better detection accuracies for complex sequential attacks. More importantly, the proposed TD learning and prediction approach provides an efficient anomaly detection technique with simplified labeling procedure and reduced computational complexity. Future work may need to be focused on the extension of the proposed method to more general intrusion detection systems with real-time applications.

6. References

[1] D. Denning: An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2) (1987) 222-232

[2] M. M. Sebring, E. Shellhouse, M. E. Hanna, and R. Alan Whitehurst. Expert systems in intrusion detection: A case study. In *Proceedings of the 11th National Computer Security Conference*, Baltimore, Maryland, October, (1988) 74-81

[3] W. K. Lee, Stolfo, S., and Mok, K.: Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, 14(6), (2000) 533 – 567

[4] D.Y. Yeung, Y.X. Ding, Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36 (2003) 229 – 243

[5] A. K. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. in *Proceedings of the 8th USENIX Security Symposium*, (1999).

[6] H.Shah, J.Undercoffer and A.Joshi: Fuzzy clustering for intrusion detection. In: *Proceedings of the 12th IEEE International Conference on Fuzzy Systems*. (2003) 1274-1278

[7] D. Barbara, N. Wu, S. Jajodia, Detecting novel network intrusions using Bayes estimators, *First SIAM Conference on Data Mining*, Chicago, IL, (2001).

[8] J. Ryan, M-J. Lin, R. Miikkulainen, Intrusion detection with neural networks, *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, AAAI Press, (1997) 72-77.

[9] X. Xu. Adaptive Intrusion Detection Based on Machine Learning: Feature Extraction, Classifier Construction and Sequential Pattern Prediction. *International Journal of Web Services Practices*, Vol.2, No.1-2 (2006), pp. 49-58

[10] M. Mahoney, P.Chan: Learning nonstationary models of normal network traffic for detecting novel attacks. In: *Proceedings of 8th International Conference on Knowledge Discovery and Data Mining*, (2002) 376-385

[11] X. Xu, X. N. Wang, Adaptive network intrusion detection method based on PCA and support vector machines . *Lecture Notes in Artificial Intelligence*, ADMA 2005, LNAI 3584, (2005) 696 – 703.

[12] P. Laskov, P. Düssel, C. Schäfer, K. Rieck, Learning intrusion detection: supervised or unsupervised? Proc. ICIAP 2005, September, *Lecture Notes in Computer Science* , LNCS 3617 (2005) 50-57

[13] W.K. Lee, S.J.Stolfo: A data mining framework for building intrusion detection model. In: Gong L., Reiter M.K. (eds.): *Proceedings of the IEEE Symposium on Security and Privacy*. Oakland, CA: IEEE Computer Society Press (1999) 120~132

[14] http://www.kdnuggets.com/datasets/kddcup.html

[15] Y. H. Liao, V. Rao Vemuri, Using text categorization techniques for intrusion detection, *Proceedings of the 11th USENIX Security Symposium*, August, (2002) 51-59.

[16] X.Xu, Intrusion Detection Based on Dynamic Behavior Modeling: Reinforcement Learning versus Hidden Markov Models, *International Journal of Computational Intelligence Theory and Practice*, 2(1), (2007) 57-66

[17] T. Lane, C. Brodley, Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3) (1999) 295–331

[18] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2): 257-286, 1986.

[19] S. Hofmeyr et al., Intrusion detection using sequences of systems call, *Journal of Computer Security*, 6 (1998) 151-180

[20] X.Xu, A Reinforcement Learning Approach for Host-Based Intrusion Detection Using Sequences of System Calls. *Lecture Notes in Computer Science*, LNCS 3644, pp. 995 – 1003

[21] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, vol. 4, (1996) 237--285.

[22] R. Sutton, Learning to predict by the method of temporal differences. *Machine Learning*, 3(1), (1988) 9-44

[23] X. Xu, H. G. He, D. W. Hu: Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research*, vol.16, (2002) 259-292

[24] J. A. Boyan, Technical Update: Least-squares temporal difference learning. *Machine Learning*, 49, (2002) 233-246

[25] X. Xu, Yirong Luo, A Kernel-Based Reinforcement Learning Approach to Dynamic Behavior Modeling of Intrusion Detection, In : D. Liu et al. (Eds.): ISNN 2007, *Lecture Notes in Computer Science*, LNCS 4491, Part I, (2007) 459–468

[26] X. Xu, et al., Kernel Least-Squares Temporal Difference Learning, *International Journal of Information Technology*,11(9), (2005) 54-63

[27] Schölkopf, B., Smola, A.: *Learning with Kernels*. Cambridge, MA: MIT Press (2002)

[28] Nashed, M. Z., ed.: *Generalized Inverses and Applications*. Academic Press, New York, (1976)

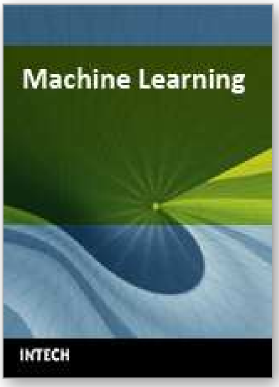
[29] Xu, X.: A Sparse Kernel-Based Least-Squares Temporal Difference Algorithm for Reinforcement Learning. In: Proceedings of International Conference on Intelligent Computing. 2006, *Lecture Notes in Computer Science*, LNCS 4221 (2006) 47-56

[30] Engel, Y., Mannor, S., Meir, R.: The Kernel Recursive Least-Squares Algorithm. *IEEE Transactions on Signal Processing*, 52 (8) (2004) 2275-2285

[31] N. Ye, Y. Zhang, and C. M. Borror. Robustness of the Markov-Chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53(1), (2004) 116-123.

[32] http://www.cs.unm.edu/~immsec/data/

[33] C. Warrender, S. Forrest, B. Pearlmutter. Detecting intrusions using system calls: alternative data models. in the *1999 IEEE Symposium on Security and Privacy*, May 9-12, (1999)



Machine Learning

Edited by Abdelhamid Mellouk and Abdennacer Chebira

ISBN 978-953-7619-56-1

Hard cover, 450 pages

Publisher InTech

Published online 01, January, 2009

Published in print edition January, 2009

Machine Learning can be defined in various ways related to a scientific domain concerned with the design and development of theoretical and implementation tools that allow building systems with some Human Like intelligent behavior. Machine learning addresses more specifically the ability to improve automatically through experience.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Xin Xu (2009). Machine Learning for Sequential Behavior Modeling and Prediction, Machine Learning, Abdelhamid Mellouk and Abdennacer Chebira (Ed.), ISBN: 978-953-7619-56-1, InTech, Available from: http://www.intechopen.com/books/machine_learning/machine_learning_for_sequential_behavior_modeling_and_prediction

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821