Intrusion Detection Using Self-Training Support Vector Machines

Prateek

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

Intrusion Detection Using Self-Training Support Vector Machines

Thesis submitted in
partial fulfillment of the requirements
for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Prateek

[Roll: 109CS0130]

under the guidance of

Dr. S. K. Jena

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India
May 2013

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

May, 2013

Certificate

This is to certify that the work in the thesis entitled ***Intrusion Detection Using Self-Trainining Support Vector Machines*** by ***Prateek*** is a record of an original research work carried out under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

To the best of my knowledge, the matter embodied in the thesis has not been submitted for any degree or academic award elsewhere.

Dr S. K. Jena
Professor
CSE Department of NIT Rourkela

Acknowledgment

I take this opportunity to express my profound gratitude and deep regards to my guide Dr. S. K. Jena for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I also take this opportunity to express a deep sense of gratitude to Sweta, my sister, for her support and motivation which helped me in completing this task through its various stages.

I am obliged to the faculty members of the Department of Computer Sceince & Engineering at NIT Rourkela, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment. In particular I'm thankful to Dr. B. K. Patra for helping me out with my various doubts in Data Mining.

Lastly, I thank Almighty, my parents and friends for their constant encouragement without which this assignment would not have been possible.

Prateek

Abstract

Intrusion is broadly defined as a successful attack on a network. The definition of attack itself is quite ambiguous and there exists various definitions of it. With the advent of Internet age and the tremendous increase in the computational resources available to an average user, the security risk of each and every computer has grown exponentially. Intrusion Detection System (IDS) is a software tool used to detect unauthorized access to a computer system or network. It is a dynamic monitoring entity that complements the static monitoring abilities of a firewall.

Data Mining techniques provide efficient methods for the development of IDS. The idea behind using data mining techniques is that they can automate the process of creating traffic models from some reference data and thereby eliminate the need of laborious manual intervention. Such systems are capable of detecting not only known attacks but also their variations.

Existing IDS technologies, on the basis of detection methodology are broadly classified as *Misuse or Signature Based Detection* and *Anomaly Detection* Based System. The idea behind misuse detection consists of comparing network traffic against a Model describing known intrusion. The anomaly detection method is based on the analysis of the profiles that represent normal traffic behavior.

Semi-Supervised systems for anomaly detection would reduce the demands of the training process by reducing the requirement of training labeled data. A Self Training Support Vector Machine based detection algorithm is presented in this thesis. In the past, Self-Training of SVM has been successfully used for reducing the size of labeled training set in other domains. A similar method was implemented and results of the simulation performed on the KDD Cup 99 dataset for intrusion detection show a reduction of upto 90% in the size of labeled training set required as compared to the supervised learning techniques.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Intrusion

Intrusion is generally defined as a successful attack on a network or system. In a technical report on the practice of Intrusion Detection[1], Julia et. al. have defined attack as *"An action conducted by one adversary, the intruder, against another adversary, the victim. The intruder carries out an attack with a specific objective in mind. From the perspective of an administrator responsible for maintaining a system, an attack is a set of one or more events that may have one or more security consequences. From the perspective of an intruder, an attack is a mechanism to fulfill an objective."*

By its very definition, an intrusion is a subjective phenomenon and its presence or absence can be perceived differently by different observers. An attacker would deem an attack to be successful if he is able to achieve the objectives with which the attack was initiated. From the viewpoint of the victim, an attack is considered successful if it has consequences for him. It is important to note that an attack, though successful from the victim's perspective may still be unsuccessful from the intruder's perspective. For the purpose of detection, usually the victim's perspective is considered.

Some common examples of intrusions at the network level would include Denial of Service (DoS) Attack, Packet Sniffing and Remote Login etc. Trojans and spywares are some of the mechanisms by which system level intrusions are achieved.

1.2 Intrusion Detection System

An **intrusion detection system** (IDS) is a software tool used to detect unauthorized access to a computer system or network. Ideally an intrusion detection system should be capable of detecting all types of malicious network traffic and computer usage. It is a dynamic entity that complements the static firewall. IDSs have been given the distinction of being dynamic entities by virtue of the fact that they take into account the present state of the system or network and can take actions accordingly. Consider the scenario of a guessing attack on login system. An IDS would be able to recognize the multiple failed attempts in a short span of time and would flag the activity as suspicious. However, the firewall would fail to do so as they are designed to work with a set of pre-configured rules.

Originally intrusion detection systems were tasked with the job of analyzing the network traffic or system activities and raise a flag in case of suspicious events. These systems were not capable of preventing the intrusion. Nowadays efforts are on to develop **Intrusion Detection and Prevention Systems** (IDPS). Apart from the detection module, these systems have a prevention system as well. The intrusion prevention system is supposed to take necessary actions required to prevent an intrusion detected by the detection system.

The advances in the field of social media have significantly contributed to lowering of the skills required for launching a successful attack. In addition to that, the variety and complexity of the systems used today also lead to enhanced and more sophisticated exploits. With our increased dependence on computers and more specifically on the Internet, intrusions present a very serious threat to the three goals of security i.e. confidentiality, integrity and availability. Hence more efficient and accurate intrusion detection systems have become the need of the hour.

1.3 Architecture of an IDS

An Intrusion detection system is considered to have the following components:

Data Acquisition Module This module is used in the data collection phase. In the

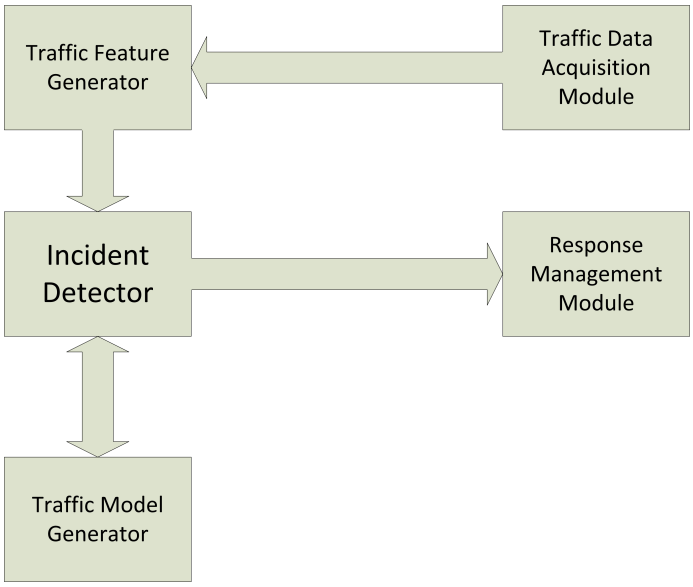

Figure 1.1: Architecture of a Network Intrusion Detection System

case of a Network Intrusion Detection System (NIDS), the source of the data can be the raw frames from the network or information from upper protocol layers such as the IP or UDP. In the case of host based detection system, source of data are the audit logs maintained by the operating system.

Feature Generator This module is responsible for extracting a set of selected features from the data acquired by the acquisition module. Features can be classified as low-level and high-level features. A low-level feature can be directly extracted from captured data whereas some deductions are required to be performed to extract the high-level features. Considering the example of a network based IDS, the source IP and destination IP of network packets would be the low level features whereas information such as number of failed login attempts would be classified as high level features. Sometimes features are categorized based on the source of data as well.

Incident Detector This is the core of an IDS. This is the module that processes the data generated by the Feature Generator and identifies intrusions. Intrusion detection methodologies are generally classified as misuse detection and anomaly detection. Misuse detection sytems have definitions of attacks and they match the input data against those definitions. Upon a successful match, the activity

is classified as intrusion. Anomaly detection systems are based on a definition of normal behaviour of a system. Any deviations from this normal profile lead to the classification of the corresponding activity as suspicious. Irrespective of the detection methodology, upon detection of an intrusion, an alert is generated and sent to the Response Management module.

Traffic Model Generator This module contains the reference data with which the Incident Detector compares the data acquired by the acquisition modules and processed by the feature generator. The source of data of the Traffic Model Generator could be non-automated(coming from human knowledge) or automated (coming from automated knowledge gathering process).

Response Management Upon receiving an alert from the incident detector, this module initiates actions in response to a possible intrusion.

A block diagram of the architecture of a Network Intrusion Detection is presented in fig 1.1. The architecture for a Host Based Intrusion Detection System would be similar.

1.4 Classification of IDS

Intrusion Detection systems are generally classified on the basis of detection methodology and source of data.

1.4.1 Detection Methodology based classification

Misuse or Signature Based Detection System Misuse detection based detection consists of comparing the traffic against a Model describing known intrusion events. This approach is quite effective at detecting known threats but its performance while detecting unknown threats is very poor. Pattern recognition, Implication rules and Data mining techniques are some of the most commonly used techniques for misuse detection.

Anomaly Detection Based System The anomaly detection method is based on the comparison of current traffic profiles against the profiles representing normal

traffic behavior. Initially an anomaly detector creates a baseline profile of the normal legitimate traffic activity. Thereafter, any new activity deviating from the normal profile is considered an anomaly. This detection methodology has the potential of detecting previously unknown attacks. However, currently the major problem with this system is the high false alarm rate. Statistical methods, machine learning and data mining techniques are among the most commonly used techniques for anomaly detection.

Stateful Protocol Analysis Based System This methodology is based on the assumption that IDS could know and trace the protocol states. Though SPA process seems similar to the Anomaly Detection methodology, they are basically different. SPA depends on vendor-developed generic profiles to specific protocols whereas, Anomaly Detection uses preloaded network or host specific profiles. Generally, the network protocol models in SPA are based on protocol standards from international standard organizations, e.g., IETF. SPA is also known as Specification- based Detection.

Hybrid Most existing IDSs use multiple methodologies to improve the accuracy of detection. For example, Signature Detection and Anomaly Detection are used as complementary methods as they provide a mixture of improved accuracy and ability to detect unknown attacks.

1.4.2 Data Source based classification

Network Based Intrusion Detection This class of IDS acquires its data from the raw frames from the network or information from upper protocol layers such as the IP or UDP. Analysis is then performed on the network logs and consequently the detection occurs at the network level.

Host Based Intrusion Detection In the case of host based detection system, source of data are the audit logs maintained by the operating system. System call logs and file system logs are the commonly used sources of data. This class of IDS detects intrusions occuring on a particular host device.

Methodology	**Pros**	**Cons**
Signature-Based	Simplest and an effective method to detect known attacks. Detailed contextual analysis.	Ineffective to detect unknown attacks, evasion attacks, and variants of known attacks. Little understanding of states and protocols. Hard to keep signatures/patterns up to date.
Anomaly-Based	Effective to detect new and unforeseen vulnerabilities. Less dependent on OS. Facilitate detections of privilege abuse.	Weak profile accuracy due to observed events being constantly changed. Unavailable during rebuilding of behavior profiles. Difficult to trigger alerts in right time.
Stateful Protocol Analysis	Know and trace the protocol states. Distinguish unexpected sequences of commands.	Protocol state tracing and examination is resource consuming. Distinguish unexpected sequences of commands. Might be incompatible to dedicated OSs or APs.

Table 1.1: Comparison of Intrusion Detection Methodologies

1.5 Literature Review

Automatic Network Intrusion Detection has been an area of active research for more than the last 20 years. In a survey paper by Catania et. al. [2], the evolution of this field of research and the issues with the existing systems have been discussed. The first Network Intrusion Detection Systems (NIDS) were misuse detection based system like P-BEST and SNORT. However since these systems rely deeply on human activity for traffic model acquistion process, they could not scale with the ever increasing variations of attacks. Data Mining was applied to some misuse based systems to reduce the demand of human intervention. Various anomaly detection techniques have been applied to this problem domain. Porras and Valdes presented a fairly successful Statistical-Based approach and various Machine Learning techniques have also been applied to this problem. Application of SVM and ANN for intrusion detection was propsed by Chen et. al [3] and Eskin et. al [4] presented an unsupervised technique based on hierarchical clustering. A detailed taxonomy and extensive comparison of various existing methods have been presented in a comprehensive review of Intrusion Detection Systems, Liao et. al. [5].

Apart from the issues related to the requirement of high level of human interaction, other problems with Intrusion Detection Systems have been discussed by Catania et. al. [2]. Lack of model adjustment information, proper traffic feature identification, lack of resource consumption information and lack of public network traffic data-sets have been mentioned as some of the important issues. Patcha et. al [6] have given a review of open problems in anomaly detection based IDS. High computation complexity, noise in audit data, high false positive rate, lack of recent standard data-set, inability of IDS to defend itself from attacks, precise definition of normal behaviour and inability of IDS to analyze encrypted packets have been cited as the prominent problems with these systems..

1.6 Motivation

As discussed earlier, with the recent advances in the field of software exploits and the lowering of skills required for launching a successful attack, the problem of detecting intrusions, effectively and accurately, is becoming more and more challenging. This is severely compounded by the fact that misuse detection based system cannot suffice to meet the present needs because the number of zero-day exploits is on the rise and the problem with most anomaly detection systems is that of high false alarm rate. Further to this, both misuse and anomaly based systems require a significant amount of labeled data for the development of the traffic models used by the incident detector. Labeling of data is extremely difficult, time consuming and costly. The extensive manual intervention required in the process makes it really slow and consequently the existing systems have not been able to scale according to the increasing demands of the networks. Hence the need for an anomaly based detection system which would significantly reduce the requirements of labeled data has been felt.

Data Mining is the process of automatically discovering useful information in large data repositories [7]. It includes methods like Classification, Clustering, Anomaly Detection and Association Analysis and it can help in automating the process of finding novel and useful patterns that might otherwise remain unknown. These techniques also provide capabilities to predict the outcome of future observations. Considering these traits of the data mining techniques, it was felt that application of data mining to the problem of intrusion detection would be a suitable course of research to tackle the current issues with the problem domain.

1.7 Objective and Scope of Work

The research was carried out with the following objectives:

1. To study the performance of various existing data mining based intrusion detection systems and compare their accuracy and efficiency.

2. To put efforts in the direction of development of a novel intrusion detection

systems which may overcome some of the drawbacks of the existing systems.

For the purpose of this research, network based detection systems have been considered. However, the same could be applied to the problem of host based detection systems with minor modifications. The current effort was concentrated on the analysis and development of only the Traffic Model Generator and Incident Detector. The other components of IDS, such as the Traffic Data Acquisition Module or the Response Management Module were not considered. This was done to concentrate on the core features of the intrusion detection process.

1.8 Outline of Thesis

The thesis consist of three chapters following this chapter:

Chapter 2: Proposed Work

A semi-supervised data mining based solution is discussed in this chapter. Along with the the proposed algorithm, the preliminaries required are also discussed briefly.

Chapter 3: Simulation and Results

The details of the simulation and the results obtained through them are discussed here. Major inferences obtained are also outlined in the chapter.

Chapter 4: Conclusion and Scope of Future Work

This chapter discusses the outcome of the research work, the significance of the proposed solution and the scope for further improvements in the proposed methodology.

Chapter 2

Proposed Work

2.1 Problem Formulation

Under the supervised learning paradigm, the problem of intrusion detection can be modeled as a classification problem. This approach consists of first obtaining labeled traffic data and then training a classifier to discern between the normal traffic and intrusions. Each record belonging to the training set consists of a certain number of traffic features, such as the protocol type, service requested, size of payload etc. Each of these records has a label indicating the class of traffic (normal/ intrusion) they belong to. The requirement of labeled data for the training of the classifier can be significantly reduced by the application of semi-supervised learning techniques.

The anomaly detection approach for intrusion detection is generally based on the following assumptions

- Records belonging to normal traffic and intrusion are inherently different in nature and hence can be separated by a suitable classifier.
- Records contained in the training set belong mostly to normal traffic data, with the number of records pertaining to intrusions being comparatively small.

2.2 Support Vector Machines

Support Vector Machines (SVM) are a classification technique given by Boser et. al. (1992) [8]. Based on the concept of **optimal margin classifiers**, this classification

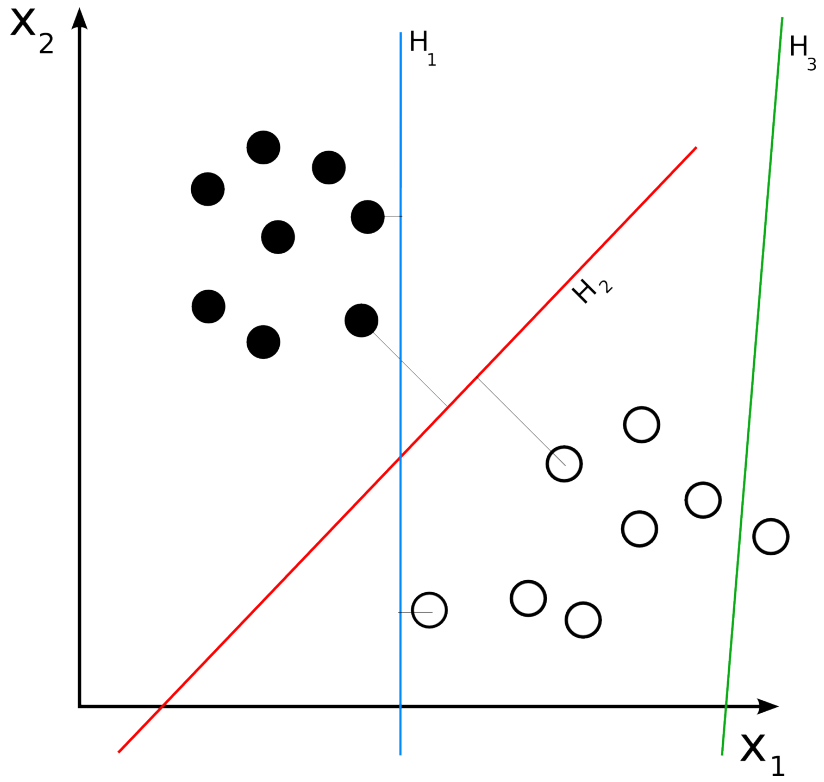

Figure 2.1: Infinite Decision Hyperplanes for a Binary Classification Problem

H1, H2 and H3 are three of the infinite possible decision hyperplanes. The hyperplane H3 (green) doesn't separate the two classes and is not suitable for use in classification. The hyperplane H1 (blue) does separate the two classes but with a small margin and H2 (red) separates the two classes with the maximum margin.
Source - http://en.wikipedia.org/wiki/Support_Vector_Machines

method gives a very high accuracy rate for a large number of problem domains and is highly suited for high-dimensional data.

2.2.1 Maximal Margin Hyperplanes

For the purpose of illustration, lets consider a data set that is linearly separable. Given a set of labeled training data, we can find a hyperplane such that it completely separates points belonging to the two classes. This is called the decision boundry. An infinite number of such decision boundaries are possible (fig 2.1). Decision Boundry margin refers to the shortest distance between the closest points on the either side of the half plane (fig 2.2). It is evident by intuition and has been mathematically proven[8] that the decision hyperplane with the maximal margin provides better generalization error. **Support Vectors** refers to training samples lying on the margins of

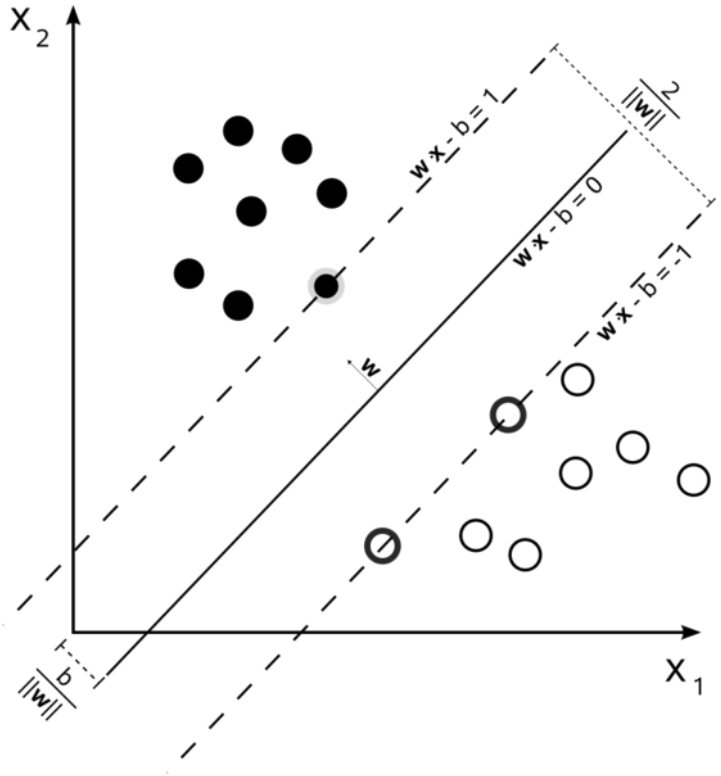

Figure 2.2: Optimal Margin Classifier for Binary Classification Problem

The figure shows the maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.
Source - http://en.wikipedia.org/wiki/Support_Vector_Machines

the decision plane and the processs of training the SVM involves finding these support vectors.

2.2.2 Linear SVM for Separable Case

Consider a binary classification problem consisting of N examples in the training data-set. Each example is denoted by a record $(\mathbf{x_i}, y_i)(i = 1, 2, ..., N)$ where $\mathbf{x_i} = (x_i1, x_i2, ..., x_id)^T$ represents the set of attributes for the $\mathbf{i}^{th}$ example. Let $y_i \in -1, 1$ be the class labels for the two classes.

The decision boundary for the classifier would be written as

$$\mathbf{w}.\mathbf{x} + b = 0 \tag{2.1}$$

Here $\mathbf{w}, \mathbf{b}$ are the parameters of the SVM and the training process is concerned with

finding the values for these parameters.

Considering two points $\mathbf{x_a}, \mathbf{x_b}$ located on the decision boundary. We have

$$\mathbf{w}.\mathbf{x_a} + b = 0, \tag{2.2}$$

$$\mathbf{w}.\mathbf{x_b} + b = 0 \tag{2.3}$$

Subtracting the two equations we get

$$\mathbf{w}.(\mathbf{x_a} - \mathbf{x_b}) = 0, \tag{2.4}$$

For a point $\mathbf{x_b}$ above the decision boundary, we have

$$\mathbf{w}.\mathbf{x_b} + b = k, k > 0 \tag{2.5}$$

For a point $\mathbf{x_w}$ above the decision boundary, we have

$$\mathbf{w}.\mathbf{x_w} + b = k, k < 0 \tag{2.6}$$

Accordingly we have,

$$y = 1 \quad if \quad \mathbf{w}.\mathbf{z} + b > 0$$

$$-1 \quad if \quad \mathbf{w}.\mathbf{z} + b < 0$$

Considering two hyperplanes b_{i1} and b_{i2}, such that they pass through the points closest to the decision margin on each side of it, we have

$$b_{i1} : \mathbf{w}.\mathbf{x} + b = 1 \tag{2.7}$$

$$b_{i2} : \mathbf{w}.\mathbf{x} + b = -1 \tag{2.8}$$

It can be seen that the margin of the classifier would be given as

$$d = \frac{2}{\mathbf{w}} \tag{2.9}$$

The problem of training a SVM is that of optimizing the above equation, which translates to the determination of the model parameters $\mathbf{w}$ and b based on the training

examples. This problem is one of a convex optimization problems and is solved for the Dual formulation using Lagranges Multiplier Method.

2.2.3 Linear SVM for Non Separable Case

To adapt the formulation of the decision boundary presented for the separable case, we need to adopt the **soft margin**[7] approach. A slack variable ξ is introduced as the penalty for deviating from the hard decision boundary. It the estimate of the error for a particular training example. The modified formulation is given as:

$$\mathbf{w}.\mathbf{x_i} + b \geq 1 - \xi_i \quad if \quad y_i = 1,$$

$$\mathbf{w}.\mathbf{x_i} + b \leq -1 + \xi_i \quad if \quad y_i = -1, \tag{2.10}$$

where $\forall i : \xi_i > 0$

Considering the change in the formulation, the modified objective function is given as:

$$f(\mathbf{w} = \frac{\|\mathbf{w}^2\|}{2} + C(\sum_{i=1}^{N} \xi_i)^k \tag{2.11}$$

where C and k are user defined parameters. If we want to emphasize on the firm boundary, we need to set the value of C to be small and if we want to optimize the residual error, we set the value of C to be big. For most cases, the value of the parameter k is assumed to be 1.

2.2.4 Non-Linear SVM and Kernel Functions

Cases where the decision boundary is non-linear require the data in the orginial space $\mathbf{x}$ to be transformed to a new feature space $\phi(\mathbf{x})$. This transformation is brought about by the transformation function ϕ which is chosen so that the decision boundary in the transformed space is a linear one.

In most cases the determination of the actual transform function is difficult and is not required. A manipulation called the **Kernel Trick**[7] is applied to compute the similarities in the transformed space using the attributes in the original feature

space.

2.3 Self-Training: A Semi-Supervised Learning Technique

Traditionally machine learning has had two types of tasks i.e *supervised learning* and *unsupervised learning*[9]. Supervised learning methods require a set of labeled examples, called the training set, over which the algorithm trains by adjusting its parameters. Artificial Neural Networks, K-Means classifiers and Bayesian Belief Networks are some of the examples of supervised learning methods.Unsupervised learning methods attempt to find the inherent structure in the data, without the use of any previously labeled data. Methods such as the various clustering algorithms and outlier detection algorithms fall under the class of unsupervised learning methods.

Semi-Supervised learning is an amalgamation of the two previously discussed learning methodologies. In this paradigm, training process involves the use of unlabeled data along with some labeled examples. Self-Training, also known as self-learning, self-labeling or decision-directed learning is a wrapper-algorithm that uses a supervised learning. Intially it starts labeling the unlabeled points according to the model learned with the help of the intial set of the labeled points. Thereafter a part of the unlabeled points is labeled using the current model and the using the labels of those points, retraining occurs and a new model is learned. This process is reapeated untill the required model accuracy is achieved or the algorithm converges.

2.4 Intrusion Detection Using Self-Training SVM

Self-Training of SVM has been used in the past for applications such as recognition of Transcription start sites[10], Pixel classification for Remote Sensing Imagery[11] and EEG-based brain computer interface speller system [12]. A similar algorithm is proposed for developing a Self-Training SVM for Intrusion Detection.

2.4.1 Algorithm

The formulation for a standard SVM for a binary classification problem is given as

$$min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \tag{2.12}$$

under the constraints

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, i = 1, ..., N,$$

where $\mathbf{x}_i \in \mathcal{R}^n$ is the feature vector for the i_{th} training example. $y_i \in -1, 1$ is the class label of $x_i, i = 1, ..., N, \quad C > 0$ is a regularization constant. The pseudo code for the Self-Training wrapper algorithm is given below:

Algorithm 1 Self-Training-SVM

Input: F_I, F_T and σ_0

F_I : The set of N_1 labeled training examples $\mathbf{x}_i, i = 1, ..., N$. Labels of the examples are $y_0(1), ..., y_0(N)$
F_T: The set of N_2 training examples for which the labels are unknown.
σ_0: The threshold for convergence

Output: A Trained SVM

Train a SVM using F_I and classify F_T using the model obtained
k=2
while TRUE **do**
    $F_N = F_I + F_T$ where labels of F_T are the ones predicted using the current model
    Train a new SVM using F_N and again classify F_T
    Evaluate objective function $f(\mathbf{w}^{(k)}, \xi(k)) = \frac{1}{2}\|\mathbf{w}^k\|^2 + C\sum_{i=1}^{N_1+N_2}\xi_i$
    if $f(\mathbf{w}^{(k)}, \xi(k)) - f(\mathbf{w}^{(k-1)}, \xi(k-1))) < \sigma_0$ **then**
        break
    end if
    k=k+1
end while

The last trained SVM is considered as the final classification model. The proof of convergence of the algorithm is given in Li et. al. [12]

Chapter 3

Simulation and Results

3.1 Data-Set

The KDD Cup 1999 Dataset[13] was used for the purpose of this simulation.In 1998 MIT Lincoln Labs had prepared a data set under the DARPA Intrusion Detection Evaluation Program[14]. The Third International Knowledge Discovery and Data Mining Tools Contest, which was held along with the The Fifth International Conference on Knowledge Discovery and Data Mining, used a version of the DARPA Intrusion Detection Data Set. The data set, generated from the raw TCP dump data had more than 40 features.

3.1.1 Features

The features broadly belonged to the following three classes:

- **Basic Connection Features** Some of these features were basic features of the individual TCP connections, e.g. duration of the connection and type of protocol (udp, tcp etc.).
- **Content Features** Content features which were determined using domain knowledge. Examples of content features include number of failed login attempts, login status etc.
- **High Level Traffic Features** Some of the features were high level traffic features computed using a two-second time. Examples include the number of

connections to the same host as the current connection in the past two seconds window, and the percentage of connections to the same service.

3.1.2 Attacks

The training set contained 24 known attacks whereas the testing set contained an additional set of 13 novel attacks. Additionally the probability distributiion of the test data was different from that of the training data. This was done to make the simulation more realistic. The attacks simulated fall under the following four categories:

- Denial of Service (DoS)
- Unauthorized access from a remote machine (R2L)
- Unauthorized access to local superuser privileges (U2R)
- Probing

3.2 A LIBSVM Based Implementation

The procedure for simulating the Self- Training SVM can be divided into two phases- Data Set Generation and Self-Training.

3.2.1 Data-Set Generation

During this phase, two sets of data sets are extracted from the KDD Cup '99 Training Set which consists of over 4 lakh records. The first set F_I is a set of labeled records and is used to train the initial SVM. The second set, F_T is the set of unlabeled records and is used to retrain the SVM model during the iterations of the algorithm 1. All the 41 features of KDD Cup '99 were used in the simulation.

For the purpose of this simulation, the size of F_I was taken to be much smaller than that of F_T so that the efficiency of the proposed scheme in reducing the requirement of labeled data may be properly tested.

The KDD Cup '99 Test set consisting of over 3 lakh records was used as the independent test set.

Additionally, the original data sets were scaled and converted to the libsvm format by using the data mining software Weka [15].

3.2.2 Self-Training

The wrapper code based on the algorithm given in 1 called the respective LIBSVM routines for SVM model training and class prediction. LIBSVM[16] developed by Chang et. al. is a library for Support Vector Machines and can be easliy integrated with C or JAVA codes. Its binaries can be called from virtually any language capable of executing a system call.

A RBF Kernel $\exp(-\gamma * |u-v|^2)$ was used for training a cost based SVM and the parameters for training (c and γ) can be determined either by a grid search or by the model selection algorithm as given in Li. et. al. [12].

A detailed illustration of the simulation process is given in the fig: 3.1

3.3 Results

The simulation was run with various sizes of the labeled and unlabeled set, where the maximum ratio between the labeled and unlabeled set was maintained to be 1:10. This ratio was decided on an empirical observation of results obtained by Li et. al. [12].

It was observed that the minimum size of labeled training set required for effective Self-Training was around 500 records. For labeled sets having very few examples, e.g 50-60, the overall accuracy of detection either did not change or in some cases it got reduced from its orginial value. This may be explained by considering the fact that in case of limited labeled points in the original case, the decision boundary obtained may not be accurate and upon use of the model on the unlabeled set, the points belonging to the set may be classified incorrectly. This may further lead to a reduction in the

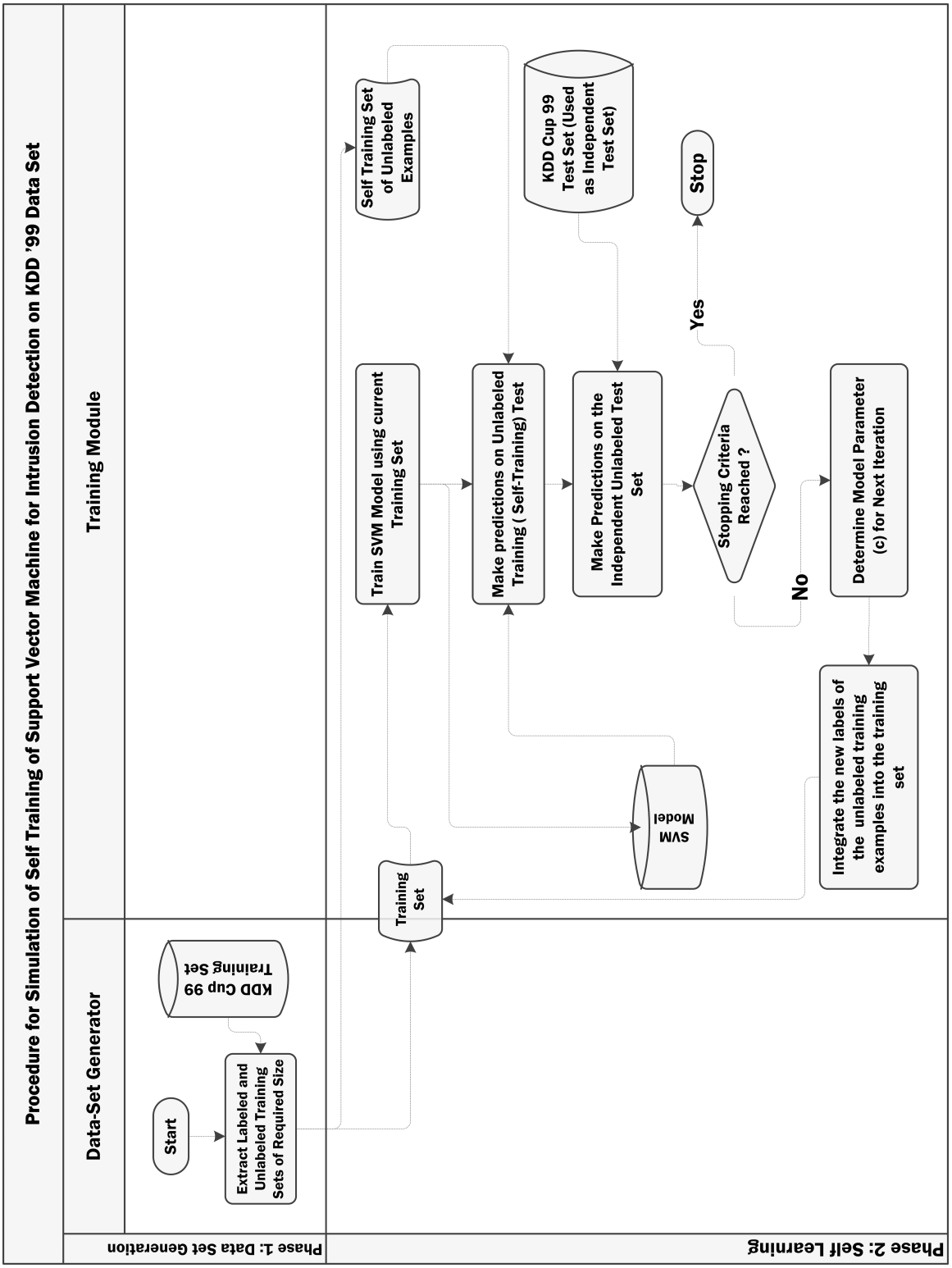

Figure 3.1: Procedure for Simulation of Intrusion Detection on the KDD '99 Data Set Using Self Training SVM

overall accuracy of detection.

Results obtained for a labeled set of 500 records with an unlabeled set of 5000 records is presented in figure 3.2. Results for another simulation with a labeled set of 5000 records and unlabeled set 25000 records is given in figure 3.3.

It can be inferred from the results that Self-Training process as given in algorithm 1 converges and for the given examples, it converges pretty quickly (after around 6 iterations in both the cases).

The degree of improvement in the detection accuracy with the iterations of the Self-Training algorithm depends on the size of the labeled and unlabeled training set. This result can be inferred from the fact that after 6 iterations, the change in the detection accuracy for the simulation with 5000 labeled records set is almost double that of the simulation with 500 labeled records set. This observation is also reaffirmed by the fact that for very small labeled training sets, there was virtually no positive improvement in the detection accuracy.

The results also show that the the overall accuracy is most sensitive to the size of the labeled set. In case of the simulation with 500 labeled records, the final detection accuracy was around 75.5% whereas for the simulation with 5000 labeled records, it was found to be around 86%.

Finally the results validate the hypothesis that Self-Training can be used for reduction of the labeled training set size in the domain of Intrusion Detection as well. A reduction of upto 90% has been achieved in the number of labeled training examples required. A comparison of the performance of Standard SVM and Self-Training SVM has been given in figure 3.4.

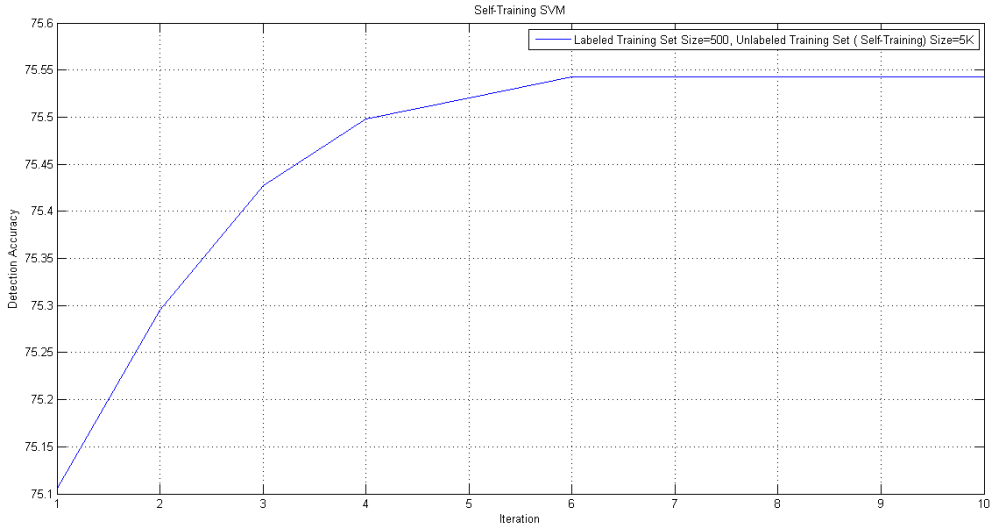

Figure 3.2: Self Training SVM with a Labeled Training Set of Size 500 and Unlabeled Training Set (Self-Training Set) of Size 5K

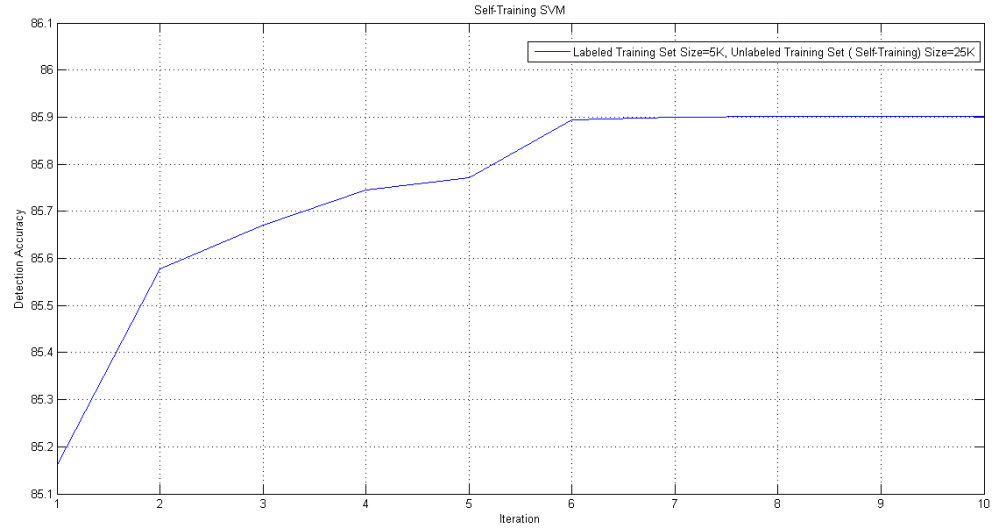

Figure 3.3: Self Training SVM with a Labeled Training Set of Size 5K and Unlabeled Training Set (Self-Training Set) of Size 25 K

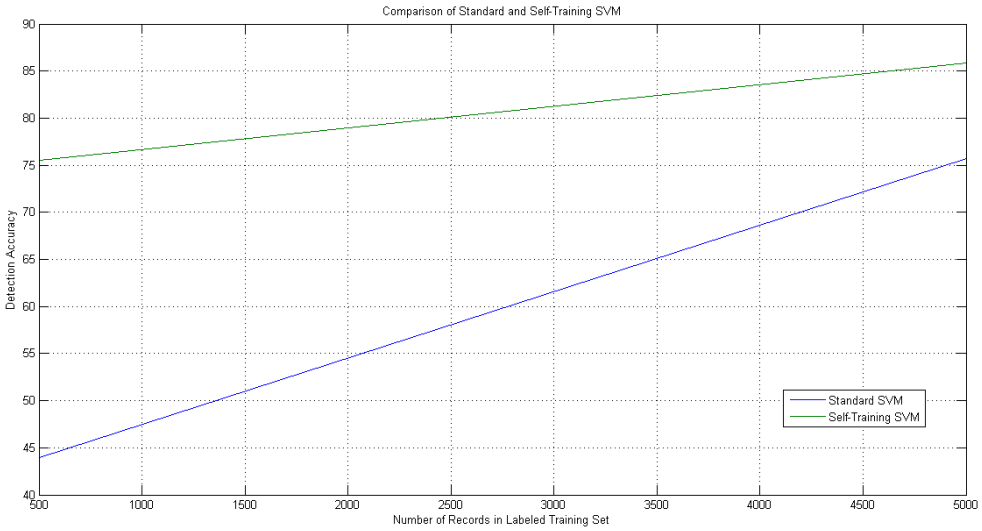

Figure 3.4: Comparison of Standard SVM and Self-Training SVM

Chapter 4

Conclusions and Future Work

A new method for Intrusion Detection under the Semi-Supervised Learning paradigm has been presented and evaluated in this thesis. The correctness of the algorithm and its effectiveness for the Intrusion Detection Problem domain has been verified by simulation on the standard KDD Cup 1999 dataset. Further, the given algorithm achieves good results in reduction of requirement of labeled training data. In the simulations run for the purpose of this thesis, a reduction of upto 90% was achieved. This value may vary from case of case, depending upon the compositions of the labeled training set.

The work presented in this thesis may be extended to the case of host based intrusion detection. The performance of this method may also be compared with that of other supervised learning approaches. Additionally the application of Self-Training scheme to other classification techniques used in intrusion detection such as the Bayesian Belief Network can be worked upon.

Bibliography

[1] Julia Allen, Alan Christie, William Fithen John McHugh, Jed Pickel, and Ed Stoner. State of the practice of intrusion detection technologies. Technical report, Carnegie Mellon University, 2001.

[2] Carlos A. Catania and Carlos Garcia Garino. Automatic network intrusion detection - current techniques and open issues. *Computers and Electrical Engineering*, 2012.

[3] Wun-Hwa Chen, Sheng-Hsun Hsu, and Hwang-Pin Shen. Application of svm and ann for intrusion detection. *Computers and Operations Research*, 2005.

[4] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leoniod Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection detecting intrusions in unlabeled data. *Advances in information security*, 2002.

[5] Hung-Jen Liao, Kuang-Yuan Tung, Chun-Hung Richard Lin, and Ying-Chih Lin. Intrusion detection system - a comprehensive review. *Journal of Network and Computer Applications*, 2013.

[6] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques- existing solutions and latest technological trends. *Computer Networks*, 2007.

[7] Pang-Ning Tan, Vipin Kumar, and Michael Steinbach. *Introduction to Data Mining*. Pearson, 2006.

[8] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. A training algorithm for maximal margin classifiers. In *The proceedings of the Fifth Annual Workshop of Computational Learning Theory*, pages 144–152. ACM, 1992.

[9] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien, editors. *Semi-Supervised Learning*, chapter Introduction to Semi-Supervised Learning. MIT Press, 2006.

[10] Jun Cai Huang, Feng Bi Wang, Huan Zhang Mao, and Ming Tian Zhou. A self-training semi-supervised support vector machine method for recognizing transcription start sites. *International Conference on Apperceiving Computing and Intelligence Analysis (ICACIA)*, 2010.

[11] Ujjwal Maulik and Debasis Chakraborty. A self-trained ensemble with semisupervised svm - an application to pixel classification of remote sensing imagery. *Pattern Recognition Letters*, 2011.

[12] Yuanqing Li, Cuntai Guan, Huiqi Li, and Zhengyang Chin. A self-training semi-supervised svm algorithm and its application in an eeg-based brain computer interface speller system. *Pattern Recognition Letters*, 2008.

[13] Kdd cup 99 data set, 1999. Data Set available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[14] Darpa intrusion detection evaluation, 1998. Data Set available at http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/1998data.html.

[15] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11, 2009. Software available at http://www.cs.waikato.ac.nz/ml/weka.

[16] Chang, Chih-Chung, Lin, and Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.