Improving Performance of Support Vector Machine for Intrusion Detection using Discretization

Yogita Bhavsar[1], Kalyani Waghmare[2]

[1]*Post Graduate Student, Pune Institute of Computer Technology, Pune, Maharashtra, India*

[2] *Assistant Professor, Pune Institute of Computer Technology, Pune, Maharashtra, India*

Abstract

Intrusion Detection System plays important role in network security as it detects various types of attacks in network. An Intrusion Detection is an important in assuring security of network and its different recourses. Intrusion detection attempts to detect computer attacks by examining various data records observed in processes on the network. Here, we are going to propose Intrusion Detection System using data mining technique: Support Vector Machine (SVM).Here, classification will be done by using SVM and dataset used for effectiveness verification purpose is NSL-KDD Cup'99 dataset .The SVM is one of the most successful classification algorithms in the data mining area, but its limitation is its long training time. Discretization technique is applied before applying classification to improve the performance of SVM. Discretization is the process of converting numerical attributes into nominal ones. The experiments carried out shows that proper discretization of dataset and proper selection of SVM kernel function can increase the performance of SVM in terms of high accuracy of attack detection and reduce long training time of SVM to few seconds. Thus our proposed approach improves SVM classification process.

1. Introduction

As network-based computer systems have increasingly important roles in modern society, they have become the targets of intruders. Therefore, we need to protect our systems. The protection of a computer system is compromised when an intrusion takes place. An intrusion can be defined as "any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource". Intrusion prevention techniques have been used to protect computer systems as a first line of defence. Intrusion prevention alone is not sufficient because as systems become more complex, there are always exploitable weaknesses in the systems.

Intrusion detection is therefore needed to give security to computer systems. The elements central to intrusion detection are: ***resources*** to be protected in a target system; models that characterize the "normal" or "legitimate" behaviour of these resources; ***techniques*** that compare the actual system activities with the established models, and identify those that are "intrusive".

2. Related Work

In 1980, the concept of intrusion detection began with Anderson's seminal paper [1]; he introduced a threat classification model that develops a security monitoring surveillance system based on detecting anomalies in user behaviour. In 2003, Kaining Lu Zehua Chen Zhigang Jin Jichang Guo, [4] has presented one collaborate IDS module to make a real-time detection and block intrusions before occurrences based on HIDS using sequences of system call anomaly detection. In 2009, Chunhua Gu and Xueqin Zhang,[6] proposed a system using rough set for attribution reduction and support vector machine for intrusion detection classification .In 2009, Yong-Xiang Xia Zhi-Cai Shi and Zhi-Hua Hu ,[5] proposed a method of detecting intrusion using incremental SVM based on key feature selection. Again in the same year, Rung-Ching Chen, Kai-Fan Cheng and Chia-Fen Hsieh,[7] used RST (Rough Set Theory) and SVM (Support Vector Machine) to detect intrusions. First, RST is used to preprocess the data and reduce the dimensions. Next, the features were selected by RST will be sent to SVM model to learn and test respectively.

In 2010, Heba F. Eid [8] effectively introduced intrusion detection system by using Principal Component Analysis (PCA) with Support Vector Machines (SVMs) as an approach to select the optimum feature subset. In 2011, Shingo Mabu, Member, *IEEE*, [9] has described a novel fuzzy class-association rule mining method based on genetic network programming (GNP) for detecting network intrusions. Again in the same year, Carol J Fung and Jie Zhang, [10] have proposed Dirichlet-based trust management to measure the level of trust among IDSes according to their mutual experience.

Recently in 2012, [11] has described an adaptive network intrusion detection system which uses a two stage architecture. In the first stage a probabilistic classifier is used to detect potential anomalies in the traffic. In the second stage a HMM based traffic model is used to narrow down the potential attack IP addresses. Again in 2012, V. Jaiganesh, [15] proposed Kernelized Support Vector Machine with Levenberg-Marquardt (LM) Learning. Again In 2012, Gholam Reza Zargar, Tania Baghaie, [13] proposed a category-based selection of effective parameters for intrusion detection using Principal Components Analysis (PCA).

3. Data Set Collection

To verify the effectiveness and the feasibility of the proposed IDS system, we have used NSL-KDD dataset. It is a new version of KDDcup99 dataset. NSL-KDD dataset has some advantages over KDDcup99 dataset. It has solved some of the inherent problems of the KDDcup99 [21], which is considered as standard benchmark for intrusion detection evaluation [17]. The training dataset of NSL-KDD consist of approximately 4,900,000 single connection vectors each of which contains 41 features and is labelled as either normal or attack type. Use of NSL-KDD dataset for classification gives an accurate evaluation of different learning techniques. It is good to use NSL-KDD dataset for experiment purpose as it consists of reasonable numbers of instances both in the training set and testing set.

4. Data Set Pre-processing

Pre-processing of original NSL-KDD dataset is necessary to make it as a suitable input for SVM. Data set pre-processing can be achieved by applying: Data Set Transformation and Data Set Discretization

4.1 Data Set Transformation

The training dataset of NSL-KDD consist of approximately 4,900,000 single connection instances. Each connection instance contains 42 features including attacks or normal. From these labelled connection instances, we need to transform the nominal features to numeric values so as to make it suitable input for classification using SVM. For this transformation, we will use table 2. Also, we have to assign numeric value to the last feature in the connection instance which is target class. For doing this, we have assigned a target class 'zero' for 'normal connection' and a 'one' for any deviation from that (i.e. if that is an attack) as per transformation table 2.In this step, some useless data will be filtered and modified. For example, some text items need to be converted into numeric values. Every instance in the dataset has 42 features or attributes including target class. An example of original NSL-KDD data set record is shown in figure 1.

0	tcp	ftp_data	SF	491	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	2	2	0	0
	0	0	1	0	0
	150	25	0.17	0.03	
	0.17	0	0	0	
	0.05	0	normal		
0	udp	other	SF	146	0
	0	0	0	0	0
	0	0	0	0	0
	0	0	0	0	0
	0	13	1	0	0
	0	0	0.08	0.15	0
	255	1	0	0.6	
	0.88	0	0	0	0
	0	normal			

Figure 1: The Original NSL KDD cup'99 dataset

There are several nominal values like http, tcp, SF in the dataset. Hence we have to transform these nominal values to numeric values in advance. For example, the service type of "tcp" is mapped to 2,"udp" is mapped to 3,"icmp" is mapped to 4 and we will follow Table 1 to transform the nominal values of dataset features into the numeric values. After transformation, the original NSL- KDD cup'99 dataset will become as shown in Figure 2.

0,2,32,14,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,150,25,0.17,0.03,0.17,0,0,0,0.05,0,0
0,3,16,14,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,1,0,0,0,0,0.08,0.15,0,255,1,0,0.6,0.88,0,0,0,0,0,0

Figure 2: NSL-KDD Cup'99 Dataset after Transformation

Table 1: Transformation Table

Type	Attribute Name	Numeric Value Assigned
Attack or Normal	Normal	0
Attack or Normal	Attack	1
Protocol type	TCP	2
	UDP	3
	ICMP	4
Flags	OTH	5
	REJ	6
	RSTO	7
	RSTOS0	8
	RSTR	9
	S0	10
	S1	11
	S2	12
	S3	13
	SF	14
	SH	15

4.2 Discretization

Many knowledge discovery in database (KDD) algorithms developed in machine learning area focus on only data sets with nominal attributes. On the other hand, it is very common that real world datasets contain numerical attributes. Those algorithms cannot be applied to real world classification tasks unless numerical attributes are discretized first.

Discretization is a process that transforms quantitative data into qualitative data. Quantitative data are commonly involved in data mining applications. However, many learning algorithms are designed primarily to handle qualitative data. Even for algorithms that can directly deal with quantitative data, learning is often less efficient and less effective. Hence discretization must be used in data mining for achieving efficiency.

Discretization is the process of converting numerical attributes into nominal ones. This conversion is done by partitioning numerical attribute domains into intervals. Discretization usually increases the efficiency of classification algorithms because reduced data require fewer computational operations that may discretize data during the learning process. Discretized data also require less space than the original data, thus discretization can save a lot of storage space of the datasets. Moreover, discretized data are more general and meaningful and induced rules are easier to interpret. Therefore, discretization is an important step in data mining, especially for the classification problems. Dataset discretization technique is used for continuous feature selection of intrusion detection and to create some homogeneity between values, which have different data types. Here, we have used range discretization technique for this purpose.

5. Conversion of datasets to LibSVM format and Linear Scaling

Pre-processed datasets are converted to LibSVM format. For this process, first categorical features from both training and testing datasets are converted to numeric value and then we have to determine target classes for classification phase. Here, we have determined two target classes: class 'zero' for normal instance and class 'one' for attack or intrusion. Then we have to save target class and feature values of each instance in LibSVM format. LibSVM format is:

[Label][index1]: [value1] [index2]: [value 2]…

Where,

Label' is target classes of classification. Usually we put integers for the class value. Here, we have used [0, 1] target class. Where class '0' indicates 'normal' and class '1' indicates 'attack'.

'Index' is the ordered index. Usually continuous integer. 'Value' is the input data for training. Usually lots of real numbers. Input dataset to the problem we are trying to solve, involves 41 of 'features', so the input will be a set of these 41 features.

6. Linear Scaling

After this conversion, we have to perform linear scaling of LibSVM format datasets and store these scaled datasets for further use. Linear scaling of datasets is done to improve the performance of classification using SVM.

7. Support Vector Machine (SVM)

The Support Vector Machine is one of the most successful classification algorithms in the data mining area .The SVM uses a portion of the data to train the system. It finds several support vectors that represent the training data. These support vectors will form a SVM model. According to this model, the SVM will classify a given unknown dataset into target classes.

In the proposed system, we have constructed a SVM model for classification. While intrusion behaviors happen, SVM will detect the intrusion. A classification task involves training set and testing set which consist of instances. Each instance in the training set contains one "target value" (class labels: Normal or Attack) and 41 features. The goal of SVM is to produce a model which predicts target value of data instance in the testing set which consist of only features. To achieve this goal, we have used Radial Basis Function (RBF) kernel functions available with SVM.

8. Experimental Results and Discussions

We have carried out experiment using WEKA 3.7 [20] and LibSVM 1.5 [19] .Initially dataset transformation and normalization is done using MATLAB. Dataset pre-processing using discretization technique is done using WEKA and classification is done using LibSVM. Experimental results shown in Table 2 indicate that time required to build the SVM training model, before discretization is too much i.e. 3332.16 seconds which gets reduced to 77.01 seconds after discretization. Before discretization, the accuracy of SVM Classifier is 94.18 % which is increased to 99.99 % after discretization.

Our experimental results indicate that due to use of discretization at the time of dataset pre-processing and linear scaling of LibSVM format dataset has improved the performance of SVM. Graph of SVM model building time is shown in figure 3 and SVM accuracy graph is shown in figure 4.

Table 2: Time Required to Build Model and Accuracy of Classifier Before and After Discretization

	Before Discretization	**After Discretization**
Time (in seconds)	3332.16	77.01
Accuracy (in percentage)	94.1857	99.996

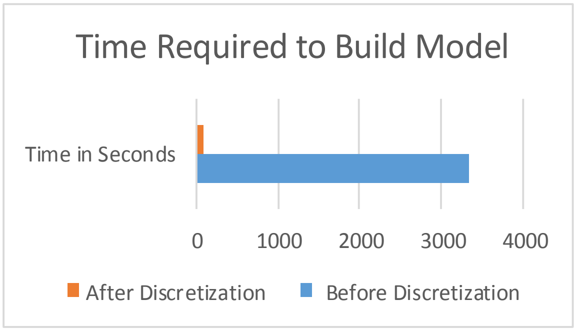

Figure 3: SVM Model Build Time Graph

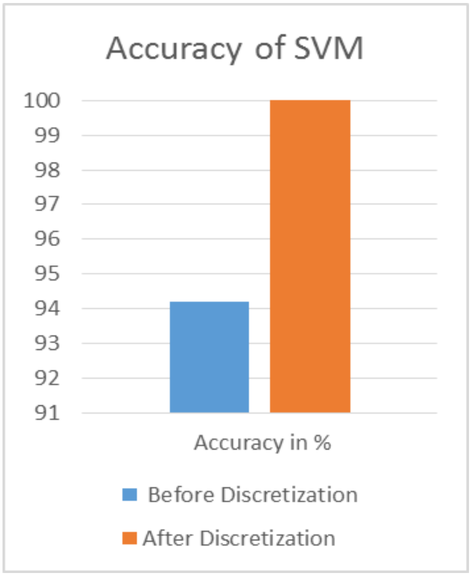

Figure 4: Accuracy of SVM Before and After Discretization

9. Conclusion:

We have applied discretization technique on transformed NSL- KDD cup'99 dataset before applying classification using SVM, in order to speed up the training process of SVM. Use of discretization technique reduces the training set and hence reduces the training time required for SVM to few seconds i.e. 77.01 seconds. Also use of RBF kernel function instead of any other kernel function of SVM improves the accuracy of SVM classifier**.**

10. References

[1] James P. Anderson, "Computer Security Threat Monitoring and Surveillance," Technical report, James P. Anderson Co., Fort Washington, Pennsylvania. April 1980.
[2] Tomas Abraham, "IDDM: INTRUSION Detection using Data Mining Techniques", Technical report DTSO electronics and surveillance research laboratory, Salisbury.
[3] Wenke Lee and Salvatore J. Stolfo, "A Framework for constructing features and models for intrusion detection systems", ACM transactions on Information and system security (TISSEC), vol.3, Issue 4, Nov 2000.
[4] Kaining Lu Zehua Chen Zhigang Jin Jichang Guo." An Adaptive Real-Time Intrusion Detection System Using Sequences Of System Call", CCECE 2003
[5] Yong-Xiang Xia, Zhi-Cai Shi, Zhi-Hua Hu,"An Incremental SVM for Intrusion Detection Based on Key Feature Selection" 2009 Third International Symposium on Intelligent Information Technology Application.
[6] Chunhua Gu and Xueqin Zhang," A Rough Set and SVM Based Intrusion Detection Classifier", Second International Workshop on Computer Science and Engineering, 2009.

[7] Rung-Ching Chen, Kai-Fan Cheng and Chia-Fen Hsieh ,"Using Rough Set And Support Vector Machine For Network Intrusion Detection" International Journal of Network Security & Its Applications (IJNSA), Vol 1, No 1, 2009
[8] Heba F. Eid, Ashraf Darwish, Aboul Ella Hassanien, and Ajith Abraham" Principle Components Analysis and Support Vector Machine" based Intrusion Detection System", IEEE 2010.
[9] Shingo Mabu, Ci Chen, Nannan Lu, Kaoru Shimada,and Kotaro Hirasawa," An Intrusion-Detection Model Based on Fuzzy Class-Association-Rule Mining Using Genetic Network Programming", IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications And Reviews, Vol. 41, No. 1, January 2011
[10]Carol J Fung and *Jie* Zhang," Dirichlet-Based Trust Management for Effective Collaborative Intrusion Detection Networks" ,IEEE Transactions On Network And Service Management, Vol. 8, No. 2, June 2011
[11]R Rangadurai Karthick, Vipul P. Hattiwale, Balaraman Ravindran," Adaptive Network Intrusion Detection System using a Hybrid Approach ", IEEE 2012
[12]Vincent F. Mancuso, Dev Minotra, Nicklaus Giacobe, Michael McNeese and Michael Tyworth " *idsNETS*: An Experimental Platform to Study Situation Awareness for Intrusion Detection Analysts", IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support, New Orleans, LA,2012
[13]Gholam Reza Zargar, Tania Baghaie, "Category-Based Intrusion Detection Using PCA", Journal of Information Security, 2012.
[14] Neethu *B*, "Classification of Intrusion Detection Dataset using machine learning Approaches", International Journal of Electronics and Computer Science Engineering, 2012.
[15] V. Jaiganesh, "Intrusion Detection Using Kernelized Support Vector Machine With Levenbergmarquardt Learning", International Journal of Engineering Science and Technology , 2012.
[16] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali, A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set", proceeding of IEEE symposium on computational Intelligence in security and defense application, 2009
[17] KDD Cup 1999. Available on http://kdd.ics.uci.edu/ Databases/kddcup99/kddcup99.html, October 2007 [18] R.O. Duda, P.E. Hart, and D. G. Stork, "Pattern Classification", Vol. 1, Wiley, 2002.

[19] Chih-Chung Chang and Chih-Jen Lin, 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

[20] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.