Spring 5-31-2017

Undersampling GA-SVM for network intrusion detection

Zhenyu He
New Jersey Institute of Technology

ABSTRACT

UNDERSAMPLING GA-SVM FOR NETWORK INTRUSION DETECTION

by
Zhenyu He

Network intrusion detection is one of the hottest issues in the world. An increasing number of researchers and engineers deal with this problem by using machine learning methods. However, how to improve the identification accuracy of all the attack classes remains unsolved since the dataset is an imbalanced one with high imbalance ratio. This thesis work intends to build a classifier to achieve high classification accuracy. It proposes an undersampling Genetic Algorithm-Support Vector Machine (GA-SVM) method to handle this problem. It applies an undersampling method in GA-SVM. To solve the multiclassification problem with a binary classifier, this work proposes to utilize the undersampling GA-SVM with several classic structures. After adjusting the parameter in genetic algorithm and undersampling ratio in each support vector machine, this work concludes that the proposed undersampling GA-SVM improves the performance of an intrusion detection system. Among its variants, the decision tree-based undersampling GA-SVM offers the best performance.

UNDERSAMPLING GA-SVM FOR NETWORK INTRUSION DETECTION

by
Zhenyu He

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Helen and John C. Hartmann
Department of Electrical and Computer Engineering

May 2017

APPROVAL PAGE

UNDERSAMPLING GA-SVM FOR NETWORK INTRUSION DETECTION

Zhenyu He

Dr. Mengchu Zhou, Thesis Advisor Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Xuan Liu, Committee Member Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Sisi Li, Co-Advisor Date
Assistant Professor of Computer Science Department of Mathematics and Computer Sciences, Mercy College

BIOGRAPHICAL SKETCH

Author: Zhenyu He

Degree: Master of Science

Date: May, 2017

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2017

- Bachelor of Science in information science and engineering,
  Southeast University Nanjing, P. R. China, 2011

Major: Electrical Engineering

Dedicated to my family, all inclusive, known and unknown –for giving birth to me at the first place and supporting me spiritually throughout my life

ACKNOWLEDGMENT

Foremost, I would like to express my deepest gratitude to my advisor, Dr. Mengchu Zhou, for excellent guidance, patience and bringing me to the academic research area. Professor Zhou served as my research advisor, but he was very influential in the academic path I have chosen and gave me many excellent suggestions in my thesis research.

Besides my advisor, I would like to thank Dr. Sisi Li as my thesis co-advisor and Dr Xuan Liu as my thesis committee member for giving me deep understanding of machine learning. Furthermore, my sincere thanks also go to my group members and my good friends for giving me many excellent suggestions and assisting me to complete this thesis.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF TABLES
(Continued)

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Network security is becoming increasingly important recently. Without effective security systems, our daily life would be in a state of chaos. Through the Internet, one can find personal information, reveal a commercial chain and impact homeland security. To prevent possible interferences of foreign countries on a nation's President election such as 2016 USA President election, many scientists, researchers and engineers are working on improving their nation's security systems. As the size, transformation speed and connectivity of the data dramatically increase every year, the current security methods turn to be less and less effective, which mainly focus on intrusion detection (ID), encryption, and firewall. To enhance the security of a system, we have to design an intrusion detection system (IDS). Its purpose is to classify a large variety of intrusion in real time with high accuracy. By testing the patterns of users' activity with audit records, an IDS aims to detect an intrusion.

1.1 Cost of Network Intrusion

People tend to hardly live without any network, specifically Internet. In a modern society, a network carries many different meanings for different people. Fundamentally it offers connections to people. Facebook and Twitter are two examples of using networks to connect various people. People are willing to post their personal information like their real name, birthday, and high school on the website.

However, it is undoubted that a network is not safe, and every minute, hundreds of people are under the attack through it with or without human users' awareness. One of the examples of network intrusion is computer viruses. With just a click of your email or downloading a file from Internet, you may be attacked by the viruses attached in the file. Those network intrusion activities are hard for human beings to recognize with their eyes and feelings and could cause great cost for the victims.

A recent survey shows that at least 16 million households are under the attack of a serious computer virus. The number of people with spyware problems could come up to 8 million. The money estimated for households to pay because of network intrusion could be about $4.5 billion in last two years in lost time, money or computer hardware.

In business, computer viruses could cost the company $55 billion per year even though they have spent about $8 billion per year on virus protection per year for all on-site computers and network equipment.

Even though most of times, we are under the protection of firewall in the computer, it is not enough. To improve the protection from the network intrusion, a real-time effective network intrusion detection system is highly demanded.

1.2 Intrusion Detection System

An intrusion detection system (IDS) [Deng, 2016] refers to a device or software application that protects a network or systems from dangerous activities. The detected intrusion would be either reported to an administrator or gathered by a security information and event management (SIEM) system. The purpose of IDS is to classify a large variety of intrusions in real time with high accuracy. By testing the patterns of users' activities with audit records, an IDS detects the intrusion.

Varying from antivirus software to hierarchical systems, there is a wide range of IDS. Based on the functionality, it is generally classified into two types: misuse detection system and anomaly-based intrusion detection system.

1.2.1 Misuse Detection System

A misuse detection system [Ibrahiml, 2013] is built upon the misuse detection that detects computer attacks from a network. Abnormal system behavior has to be defined first, and then all the other behaviors would be labeled as normal system behavior. Misuse IDS identifies the intrusion by making comparison between the already well defined patterns of attack with the user behavior. In other words, in such a system, any behavior not known before would be treated as being normal one. Clearly, it can only detect attacks for which a signature was previously created.

1.2.2 Anomaly-based Intrusion Detection System

Different from a misuse detection system, anomaly-based intrusion detection system [Ji, 2016] utilizes the reverse idea. It first defines normal system behavior and then all the other behaviors would be defined as abnormal. In other words, an anomaly IDS requires the storage of normal behavior and the system's proper judgment of the data. With the rapid development of machine learning technology, an increasing number of scientists tend to build the model of anomaly IDS. Although such built model tells the attack from the normal behavior, it is important to decrease the false positives.

1.2.3 Four Types of Attack

An IDS should be designed to separate the attack from the normal data but also classify the attack into different types, e.g., Probing, Figure, U2R, and R2L. In network security, a probing attack is defined as an attempt to get access to a computer through a weak point in a computer system. There are several types of probing attack: ip-sweep, port-sweep, nmap and satan.

The denial-of-service attack (DoS attack) is a cyber-attack. It tries to prevent a normal user from getting the network resource using the perpetrators such as pod and smurf.

The remote to local (R2L) attack is defined as unauthorized local access as the user of the machine. Its examples include imap and multihop.

User to root (U2R) attack is such attack that tries to get root access to the system. Its representative examples include rootkit and perl.

The traditional method for IDS is based on the definition of each possible attack and the corresponding background knowledge. Its example is like designing the IDS based on LAN. In this method, people can find the relation between attack and LAN based on the basic knowledge of LAN.

However, with the rapid development of high tech and network intrusion technology, the traditional method turns to be weak and need to be refined. One of the most popular methods is support vector machines (SVM). They are originally used to classify the data into binary classes. To make multi classification, there are several methods to solve the problem. To improve the parameter of an SVM algorithm, we apply genetic algorithm (GA) to improve the parameter of SVM. Since the data set of KDDCUP 99 contains imbalanced data, we improve the algorithm withthe help of anundersampling method.

Chapter 2 discusses a basic GA-SVM method for multiclassification. The concepts of ordinary GA-SVM with undersampling are introduced in Chapter 3. Chapter 4 presents the experimental result of GA-SVM with an undersampling method. Chapter 5 concludes this thesis with a discussion of its limitations and future research directions.

CHAPTER 2

LITERATURE REVIEW

2.1 Machine Learning

Traditionally, scientists and engineers built the IDS based on the concept of different types of attacks [Langthasa, 2015]. However, an increasing number of researchers are focusing on machine learning methods for IDS [Dong, 2016].

In recent years, machine learning turns to be one of the most heated fields in computer science. In 1959, Arthur Samuel gave the definition of machine learning that computers had the ability to learn without being explicitly programmed. In the artificial intelligence (AI) field, machine learning is evolved from the study of computational learning and pattern recognition. By constructing models and algorithms from training data (input data), a machine learning classifier predicts the testing data. In the field where the explicit programming algorithm does not perform so well, machine learning often provides good results.

Machine learning has a strong connection with computational statistics. With the help of data analytics, it turns to be a great method that can be used to build complex models and algorithms and then make prediction. These models i.e., neural networks, support vector machines, decision trees, and so on, allow data scientists, engineers and researchers to make repeatable and reliable results from the data.

Two important parts: pre-processing and classifier design contribute to the final performance of the IDS.

2.2 Preprocessing

Data pre-processing [Jaiswal, 2016] is a vital step in a data mining and machine learning process. Machine learning and data mining algorithms certainly follow the phrase "garbage in, garbage out". In other words, even an advanced algorithm is not able to build a good model by using training data with strong noise.

Some types of data like impossible data combinations (e.g., Sex: Male, Pregnant: Yes), out-of-range values (e.g., Weight: −120), and missing values may cause much trouble to the algorithm. These data without being carefully screened can produce misleading models. Thus, the quality and representation of data is the first step before training a dataset for a classifier.

The output of data pre-processing forms the final training dataset for the next step. Data cleansing, data editing, data reduction and data wrangling are normal methods that are used for data preprocessing.

Data cleansing, also called data cleaning, detects and corrects corrupt or inaccurate data from a dataset. After identifying incomplete, incorrect, inaccurate or irrelevant parts of the data, data cleansing modifies, replaces or even deletes the dirty data.

2.3 Principle Component Analysis

One of the most applied methods in data preprocessing is principle component analysis [Nskh, 2016][Han,2015].

Invented in 1901 by Karl Pearson, PCA is a statistical procedure. Analogue to the principal axis theorem in mechanics, mathematically PCA is defined as an orthogonal linear transformation that transforms the original data to a new coordinate one. A set of possible variables is transformed into principal components (principal modes of variation) which are a set of values of linearly uncorrelated variables. The number of original variables is often larger or equal to the number of principal components.

2.4 Sampling

In machine learning, a sampling method [Faiza, 2015] is widely applied in imbalanced problems. In an imbalanced dataset, the number of each class is not nearly equal. For a binary class dataset, the class with more instances is called a majority class and the other is called a minority class.

Due to the loss function which tries to optimize such quantities as mean square error or error rate, a conventional algorithm often performs to be biased to the majority class. In the worst case, the algorithm treats the minority class as the outlier of the majority class and ignores the former completely. The algorithm simply treats the entire sample from the dataset as the majority class and then builds a classifier.

The sampling methods not only improve the training step for the later classification process but also improve the accuracy to identify a minority instance.

Oversampling and undersampling are two sampling methods used to adjust the distribution of an original class in a dataset. In an imbalanced dataset, people try to build a balanced dataset with an undersampling or oversampling method.

Oversampling and undersampling are opposite ways to the dataset with the goal to achieve the similar majority and minority size. They both use a bias to change the distribution of one class which is needed. Oversampling method randomly repeats the minority instances to enlarge their population. Undersampling method randomly decreases the majority instances to undersample it. Figures 2.1-2.2 show the ideas of oversampling and undersampling.

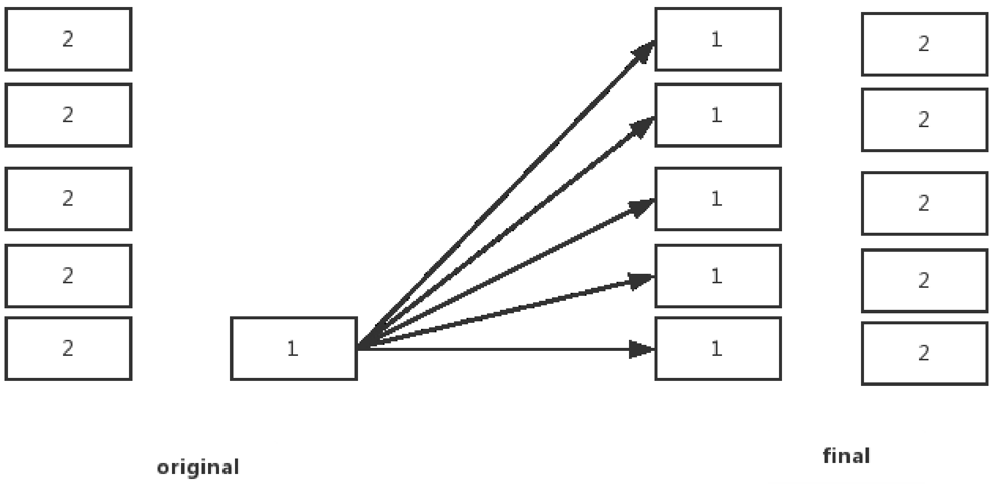

Figure 2.1 Model of oversampling.

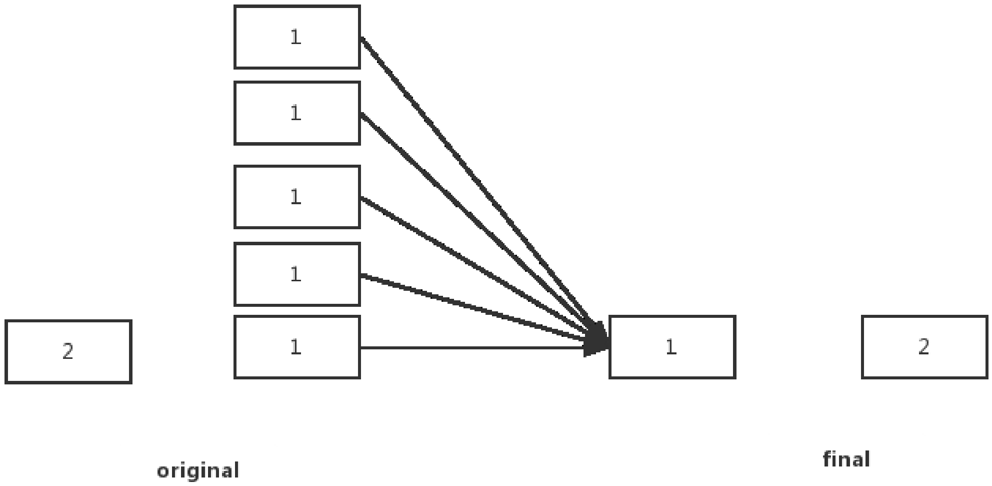

Figure 2.2 Model of undersampling.

Some data analysts prefer the oversampling method because the oversampling results in more data in training step while the undersampling removes data. However, the oversampling may results in an overfitting problem for a classifier and also requires more computational time for the learning step.

Apart from the preprocessing step, some researchers focus on the construction of classifiers to improve the performance of IDS.

2.5 SVM

Support vector machine [Yang, 2016][NSKH,2016][Khan,2007][Zhong,2010] and neural networks[Alrawashdeh,2016] are the most popular methods in classifier design.

Support Vector Machine (SVM) is a strong algorithm in machine learning for both classification and regression. Proposed by Vapnik and Chervonenkis based on statistical learning theory in 1963, SVM turns to be one of the best binary classification methods for data classification problems. Figure 2.3 shows the structure of SVM.

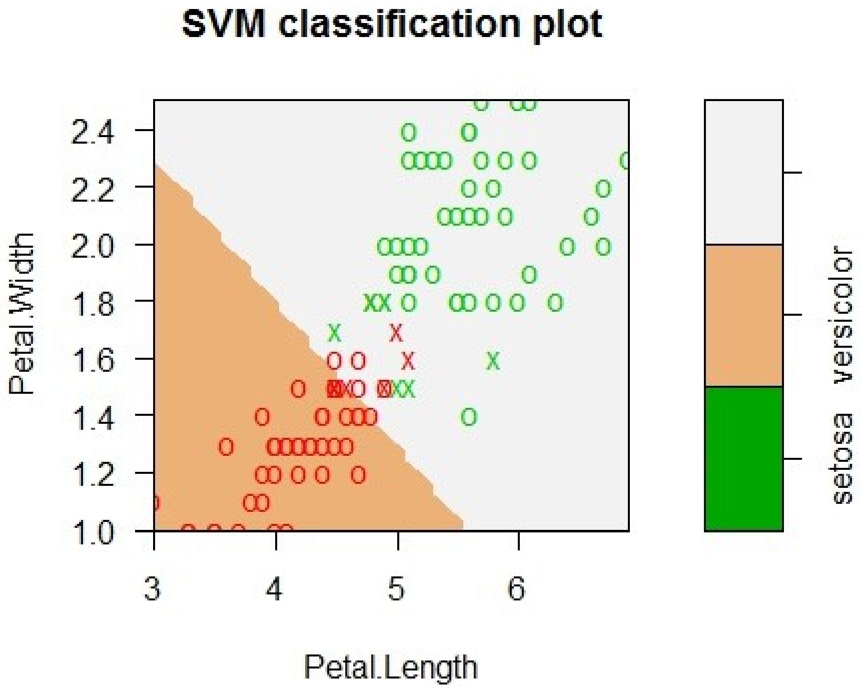

Figure2.3 The model of SVM.

SVM is usually a good learning method that suits for a small size training dataset. The distance of a hyperplane to the nearest of the positive and negative data point is defined as margin. The basic concept of a binary classification SVM is to separate the positive and negative instances with a hyperplane.

Given a set of training data in a binary classification problem: $\{(x_1,y_1),\ldots,(x_i,y_i)\}$, where $x_i \in \mathrm{R}^D$ is a set of feature vectors of the *i*-th sample in the training dataset and y $\in\{+1,-1\}$ is the label to which x_i is classified. The hyperplane satisfies the function:

$$\vec{w}\cdot\vec{x}-b=0 \tag{2.1}$$

where $\vec{w}$ is the vector of the hyperplane.

If the dataset can be linearly separated, it means that the dataset can be separated by two parallel hyperlanes. The hyperplanes satisfy the function:

$$\vec{w}\cdot\vec{x}-b=1 \tag{2.2}$$

And

$$\vec{w}\cdot\vec{x}-b=\text{-}1 \tag{2.3}$$

In geometry, we can write the distance between the two hyperplanes as $\frac{2}{\|\vec{\mathrm{w}}\|}$. To achieve the maximum distance, $\|\vec{\mathrm{w}}\|$ should be minimized. $\vec{w}$ and b are two important factors to solve the problem, $\vec{x} \mapsto \quad (\vec{\ } \cdot \vec{x} - b)$.

In some nonlinear classification problems, the kernel is one important method to reduce the complexity.

A kernel function simplifies the steps by calculating the inner products between the images of all pairs of data in the feature space instead of calculating the coordinates of the data in that space. With the kernel function, kernel method designs its own feature space from the original high dimensional or even infinite feature space.

The kernel function can be defined as

$$\mathrm{k}\left(\vec{x}_i,\vec{x}_j\right)=\varphi\left(\vec{x}_i\right)\cdot\varphi\left(\vec{x}_j\right) \tag{2.4}$$

Where $\varphi\left(\vec{x}_i\right)$ is the transformed data point corresponding to the linear classification rule from the nonlinear classification.

2.6 Decision Tree

In the machine learning, a decision tree is one of the predictive modeling methods [Cataltepe, 2016]. By applying a decision tree as a predictive model that maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). A decision tree model includes node, branches and leaf nodes.

Each internal node stands for a "test" on an attribute (e.g. Sex Male or female). Each branch represents the output of a node, and each leaf node stands for a class label after the decision tree runs the whole algorithm. The classification rules of a decision tree are the paths from the root to the leaves.

The decision tree model is easy to understand but the calculation could be complex for a dataset where the values are uncertain. The basic decision tree model is shown in Figure 2.4.

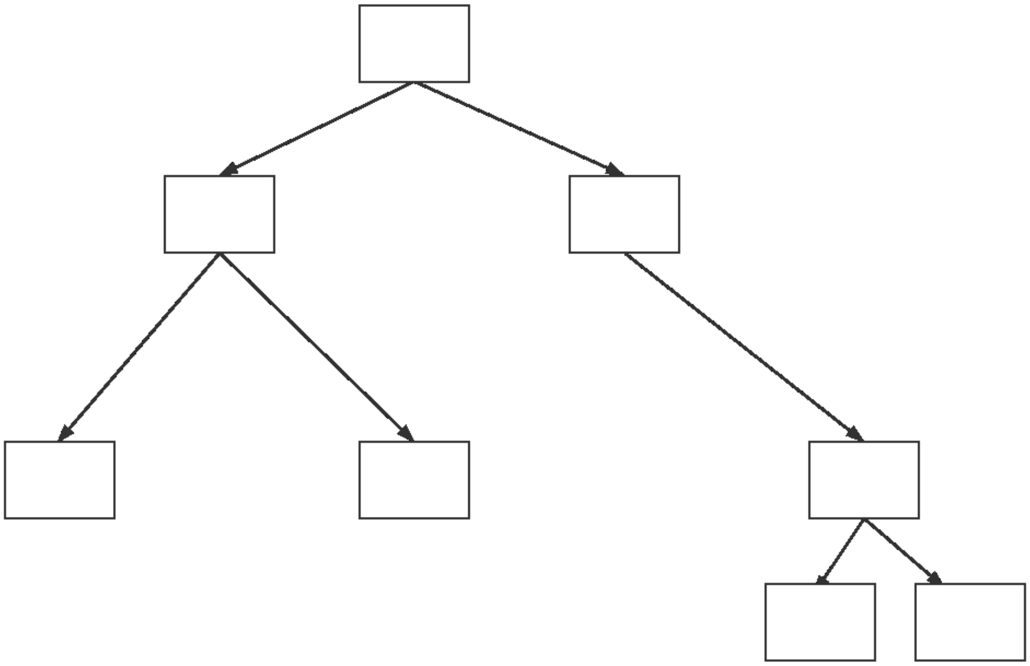

Figure 2.4 The model of a decision tree.

2.7 Decision Tree-based SVM

Decision tree-based SVM is one of the most popular and efficient methods to solve the multiclassification problem [Mulay, 2010]. In Mulay's paper, he concludes the advantage of decision tree-based SVM. SVM is originally designed for binary classification. To solve the multiclassification problem more effectively, scientists and engineers combine SVM and a decision tree to form a decision tree-based SVM, which decreases the computational time. The combined algorithm separates a dataset into two subsets from the root to the leaf until every subset consists of only one class. The classification performance is strongly influenced by the tree's construction. The structure of decision tree-based SVM is shown in Figure 2.5.

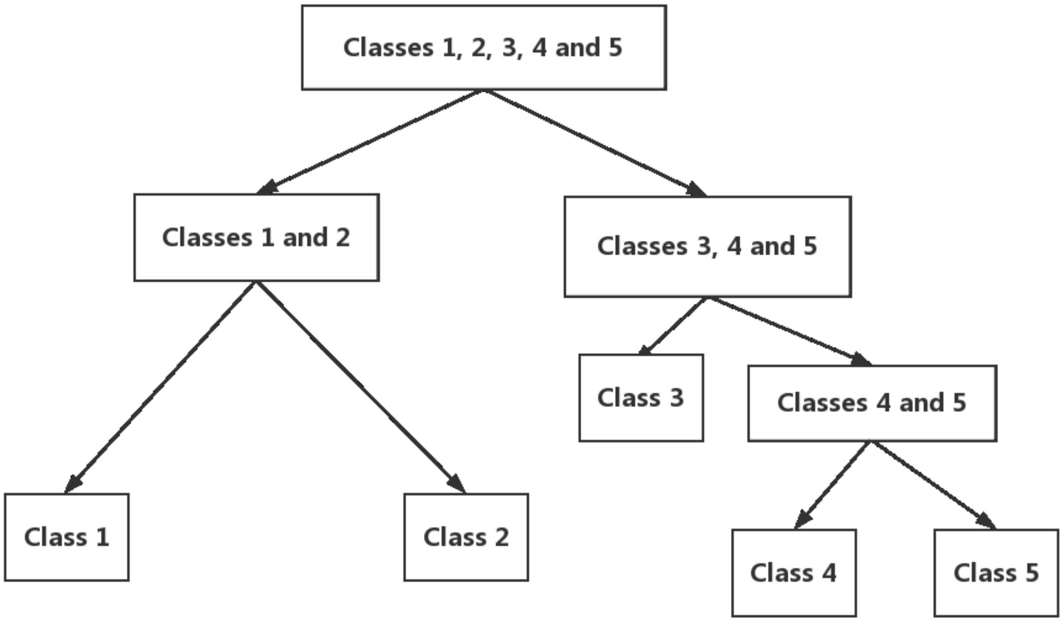

Figure 2.5 The model of decision tree-based SVM.

For example, assume that we have a five class multiclassification problem. We first set the node to separate the dataset into two subclasses: classes 1 and 2 and classes 3, 4 and 5 with a binary classifier (SVM). We repeat the step until there is only one class left in each subset.

2.8 DAGSVM

There are other solutions to solve the multiclassification problems. Directed Acyclic Graph Support Vector Machine (DAGSVM) is also one of the most common methods [Hao,2015]. DAGSVM requires n(n-1)/2 classifiers, where n is the number of classes. DAGSVM uses DAG with binary classifiers to determine the result. The DAGSVM is shown in Figure 2.6.

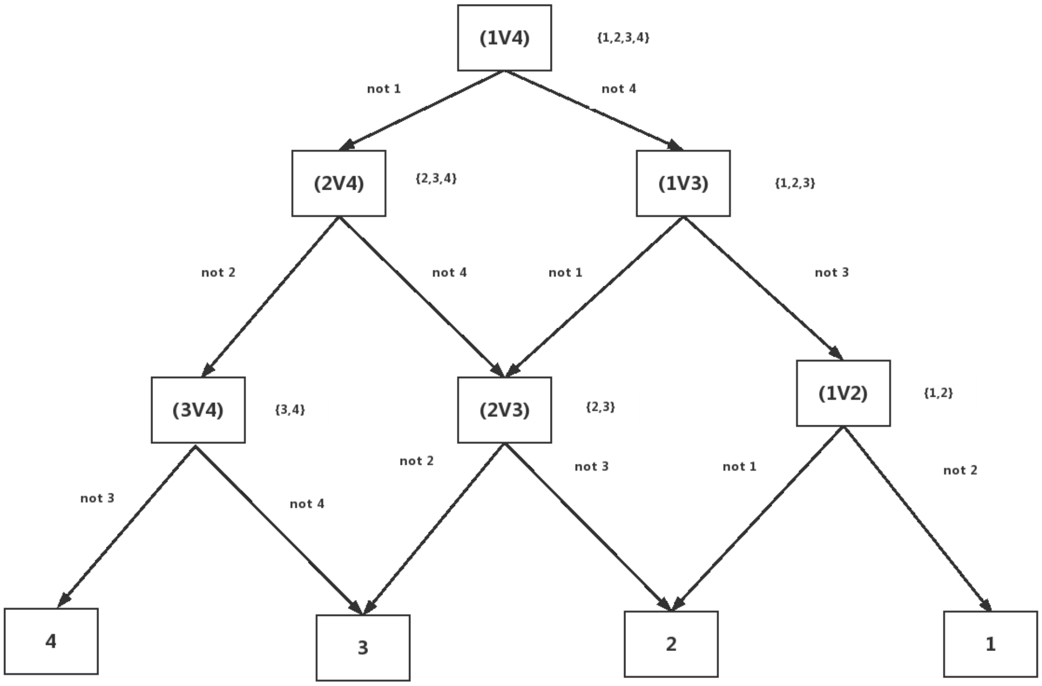

Figure 2.6 The model of DAGSVM.

Assume that we are classifying a testing data in class 4. For the first layer, we are tested whether it belongs to class 1 or 4. Thus we take the left branch to the node 2v4. Then the data is tested to see whether it belongs to class 2 or class 4. We still take the left branch. We stop the process until we touch the bottom layer.

2.9 Genetic Algorithm

In machine learning, a genetic algorithm (GA) is a common method to solve an optimization problem. Beginning with a set of randomly generated individuals (representing solutions), called a generation, GA evolves and finds a better set of the solutions. Each individual represents a feasible solution. Commonly, an individual called chromosome is represented with a string of 0's and 1's. The evolution of the individuals relies on the bio-inspired operators e.g. selection, crossover and mutation.

2.9.1 Selection

In GA, selection is a stage where individuals are chosen from a population for the later process. All individuals have their fitness values through the fitness function. After sorting them, those individuals with better performance are selected into the next step (crossover) to create the next candidates.

2.9.2 Crossover

The Crossover step plays a role as a genetic operator to change the programming of chromosomes between the two generations in GA. This step is analogous to biological crossover. The chromosomes from the parent solution are spliced at crossover points and the spliced parts are combined for the child solutions. This strep ensures those good characteristics from one individual's chromosomes can perform a better result after combining with others.

2.9.3 Mutation

Analogous to biological mutation, the mutation step maintains the genetic diversity of the chromosome from one generation to the next in GA. It may change one or more genes or even the entire gene values of the chromosome. For example, if the last bit in the chromosome of every individual in a generation happens to be a 1, the last bit of the new chromosome created through crossover will also have 1 without a mutation. The mutation step changes the current value of the chromosome to a different one. For binary string chromosome, this step changes the value of several selected bits from 0 to 1 or vice versa.

The limitation of GA is that it is almost not available to run the algorithm for the global optimization. This work intends to use GA to help decide the proper parameters in our SVM classifiers.

2.10 GA-SVM

To optimize the parameters of SVM [Yang, 2016], GA is applied to improve the performance of SVM for classification problems. The parameters in the SVM play an important role in the final model. Without an optimization method, people tend to set those parameters by their experience. In GA-SVM, the individuals in GA are the parameters in SVM. The fitness function is based on the performance of SVM with that individual after the training step. After several generations, the individuals are often trained as a good set of parameters for SVM.

We later show that GA-SVM is a great method for classification problems. But it requires more computational time than other methods since both SVM and GA are time consuming.

2.11 Neural Networks

Invented in 1940s by Warren McCulloch and Walter Pitts, artificial neural networks (ANNs) also called neural network models are a computational model applied in machine learning and deep learning. Analogous to the behavior of human brain’s axons, neural networks are based on a large number of simple neural units (artificial neurons). Each neural unit is connected with many other neural ones, and links can enhance or inhibit the activation state of adjoining neural units. The summation function is applied to each individual. To enhance the accuracy, a threshold function is set on each connection and on the unit itself. The function ensures the signal under the limit before transferring to other neurons.

Generally, ANNs typically consist of multiple layers. The first layer offers the input neurons to store the input data. Via synapses, the data is sent to the next layer. The outcome of the neurons in that layer after its neuron's network function is sent to the next layer via synapses. The last layer of neural units shows the final result. Figure 2.7 shows the structure of neural networks.

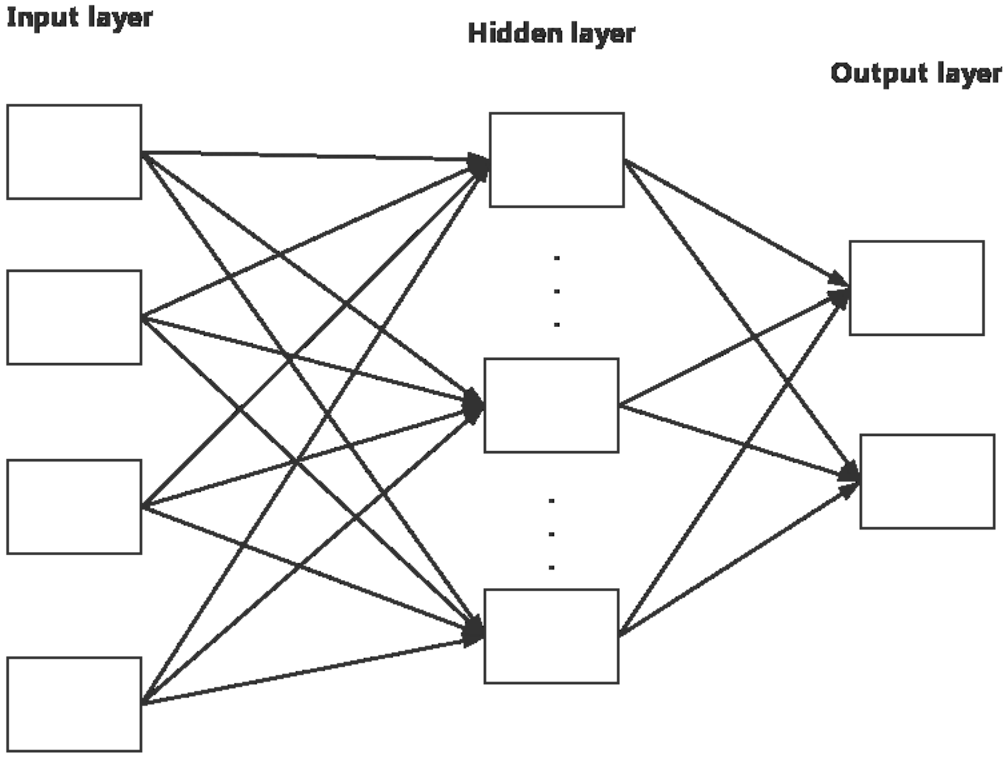

Figure 2.7 The model of neural networks.

2.12 Evaluation

2.12.1 True Positive Rate

There are several metrics people generally adopt to test the behavior of the models in machine learning: accuracy rate, true positive rate (TPR), false positive rate (FPR) and so on. In a binary classification problem, the result can be separated into four types as shown in Table 2.1.

In a binary classification problem, the result can be separated into four types:

True positive (TP): true positive means that the prediction is positive and the original condition is also positive.

False positive (FP): false positive means that the prediction is positive while the original condition is negative.

True negative (TN): true negative means that the prediction is negative and the original condition is also negative.

False negative (FN): false negative means that the prediction is negative while the original condition is positive.

Table 2.1 Confusion Matrix

Total population	prediction positive	prediction negative
Condition positive	True Positive (TP)	False Negative (FN)
Condition negative	False Positive (FP)	True Negative (TN)

In a binary classification problem, the positive rate and true negative rate are statistical measures of the result.

The TPR, also called recall, tests the proportion of positives that are correctly identified. The TPR is calculated as the ratio between the positive datasets correctly classified as positive and the total number of the actual positive datasets.

$$TPR = \frac{TP}{TP + FN} \tag{2.5}$$

True negative rate (TNR) measures the proportion of negatives that are correctly identified. It is calculated as the ratio of the number of negative instances correctly classified as negative ones to the total number of the actual negative instances:

$$TNR = \frac{TN}{FP + TN} \tag{2.6}$$

False positive rate (FPR), is defined as the ratio between negative data sets wrongly classified as negative and the total number of the actual negative datasets:

$$\mathrm{FP}R = \frac{\mathrm{FP}}{FP + TN} \tag{2.7}$$

The accuracy tests the performance of the model defined as the ratio between all correctly classified data sets and the number of all the datasets:

$$ACCURACY = \frac{\text{All corrctly classified datasets}}{\text{All datasets}} \tag{2.8}$$

2.12.2 Cross-validation

Cross-validation is a method that is used to estimate how the result of a model in machine learning generalizes to an independent dataset. When researchers or engineers want to assess what is the accuracy of the model, they would apply a cross-validation method in the settings in the prediction. Especially when the dataset is not large enough, cross-validation is a good method to solve the problem.

2.12.3 K-fold Cross-validation

K-fold cross-validation is one of the most common methods in cross-validation. In k-fold cross-validation, the researcher randomly separates the original dataset into k equal sized subsets. Among all the k subsets, a single subset is chosen as the validation data to test the model and the remaining k − 1 subsets are treated as the training data. The process of k-fold validation requires each of the k subsets used exactly once as the testing data so the cross-validation process is then required to be run k times. The k result from all k-fold subsets is averaged to produce the final estimation.

2.13 Result of Previous Work

The Tables 2.2-2.3 show the results of classifications with a random selection and a pure SVM. Since the dataset is an imbalanced dataset and usually the data in class U2R is the most imbalanced class. So the pure SVM method is weak in classifying U2R, R2L and Probe [Khan, 2007].

Table 2.2 The Output of Random Selection [Khan, 2007]

	Normal	DoS	U2R	R2L	Probe	Accuracy	FP	FN
Normal	47137	86	2	6	682	98	5	2
DoS	303	8491	0	0	12926	39	1	61
U2R	13	0	4	0	0	23	100	76
R2L	1955	2	1	368	0	15	2	84
Probe	148	1	0	0	1120	88	92	12

Table 2.3 The Output of Pure SVM [Khan, 2007]

	Normal	DoS	U2R	R2L	Probe	Accuracy	FP	FN
Normal	47180	138	4	7	584	98	7	2
DOS	1510	18437	0	0	1773	84	1	15
U2R	16	0	0	1	0	0	100	100
R2L	1888	0	8	431	1	18	2	81
Probe	144	3	0	0	1122	88	68	12

Deep learning is one of the most popular methods in the multiclassification problem. Tables 2.4-2.5 show the results of deep learning with different hidden layer. Since the dataset is imbalanced, the classifier performs weak in the Probe and U2R with the high accuracy. Also, in a deep learning method, more hidden layers do not have a strong connection with better performance [Alrawashdeh, 2016].

Table 2.4 The Output of a 4 Hidden Layer Deep Learning [Alrawashdeh, 2016]

Attack class	Number of Attack	Accurately classified	Accuracy
Normal	60593	60284	99.49%
DoS	229854	229040	99.65%
Probe	4166	591	14.19%
R2L	16347	14590	89.25%
U2R	70	5	7.14%
Total	311029	304510	97.904%

Table2.5 The Output of 2 Hidden Layer Deep Learning [Alrawashdeh, 2016]

Attack class	Number of Attack	Accurately classified	Accuracy
Normal	60593	60270	99.47%
DoS	229854	228984	99.62%
Probe	4166	654	15.7%
R2L	16347	14590	89.25%
U2R	70	16	22.89%
Total	311029	304524	97.91%

To improve the performance of the classifier, some people proposed the improved SVM method: LSSVM to classify the data. Its performance is shown in Table 2.6. [Zhong, 2010]

Table 2.6 The Output of Comparison of Results among LS-SVM, SVM and BPNN [Zhong, 2010]

Algorithm	Detection accuracy%
LS-SVM	93.67
SVM	89.0
BPNN	80.33

The traditional IDS is designed based on the definition of attacks and its influence on the LAN, TCP or others. Table 2.7 shows the performance of traditional methods [Langthasa, 2015].

Table 2.7 The Output of Intrusion Detection Based on LAN [Langthasa, 2015]

	DOS	PROBE	R2L	U2R	Normal
Number of Attack	1307	736	571	19	708
Accurately classified	1255	598	512	12	614
Accuracy (%)	95.8	81.25	89.66	63.15	86.72

Some researches combine the preprocessing methods and classification methods to improve the performance of the IDS. Figure 2.8 shows the result of PCA SVM with different kernels. Table 2.8 shows the performance of ACO with preprocessing KNN algorithm [NSKH, 2016].

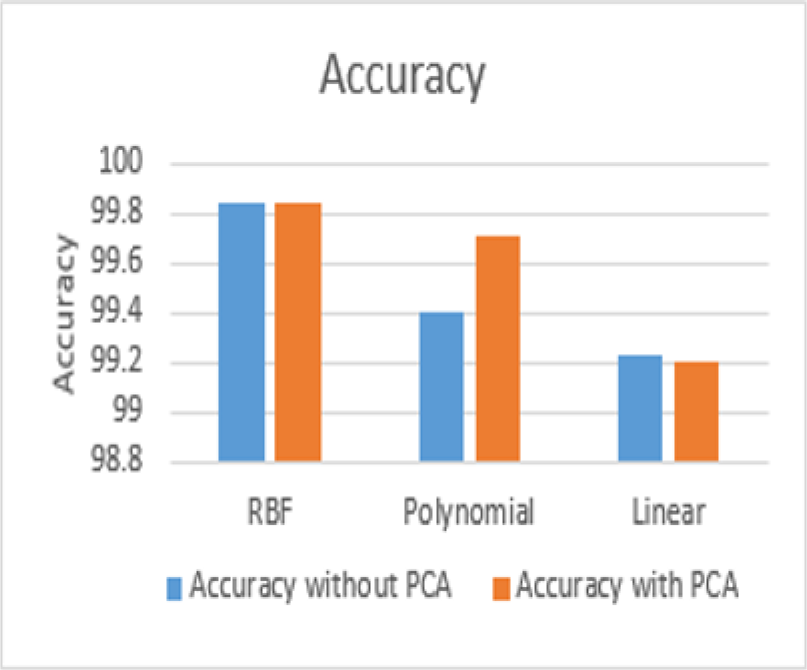

Figure2.8 The output of PCA SVM with different kernel [NSKH, 2016].

Table 2.8 The Output of KNN-ACO [Jaiswal, 2016]

Algorithm	Accuracy(%)
KNN-ACO	94.17
SVM	83.09

Among many IDS designs, FARCHD-FITIMB shows the best performance with high accuracy in each class while most of the algorithm is weak in the minority class U2R.

Table 2.9 The Output of FARCHD-FITIMB [Gaffer, 2012]

	Normal	DOS	PROBE	R2L	U2R	Recall(%)
Normal	95849	1356	24	43	6	98.5
DOS	2353	388681	73	351	0	99.3
PRB	1	3	4102	1	0	99.8
R2L	7	9	4	1098	8	97.5
U2R	1	0	0	0	51	98.1
Precision(%)	97.5	99.6	97.6	73.5	78.4	

CHAPTER 3

Methodology

3.1 Undersampling GA-SVM

The problem in IDS turns to be an imbalanced multiclassification problem. For a better performance, we choose a support vector machine (SVM) as the basic classifier for the problem. For all the SVM, we choose the same kernel, i.e., radial basis function (RBF). We use genetic algorithm (GA) to optimize the parameters: cost and gamma in SVM, thereby leading to a classifier called GA-SVM. The SVM is originally a binary classifier. In this project, we try two different structures: decision tree and directed acyclic graph (DAG) which are both consists of a set of SVMs to solve a multiclassification problem.

The biggest weakness of GA-SVM is its low efficiency. To solve this problem, we use an undersampling method. It not only reduces the number of instances in a majority class, thus improving the efficiency, but also deals with the imbalanced problem. The undersampling process is only added to the binary classifier GA-SVM when the classifier is trained with an imbalanced dataset. The process of the undersampling GA-SVM is shown in Figure 3.1

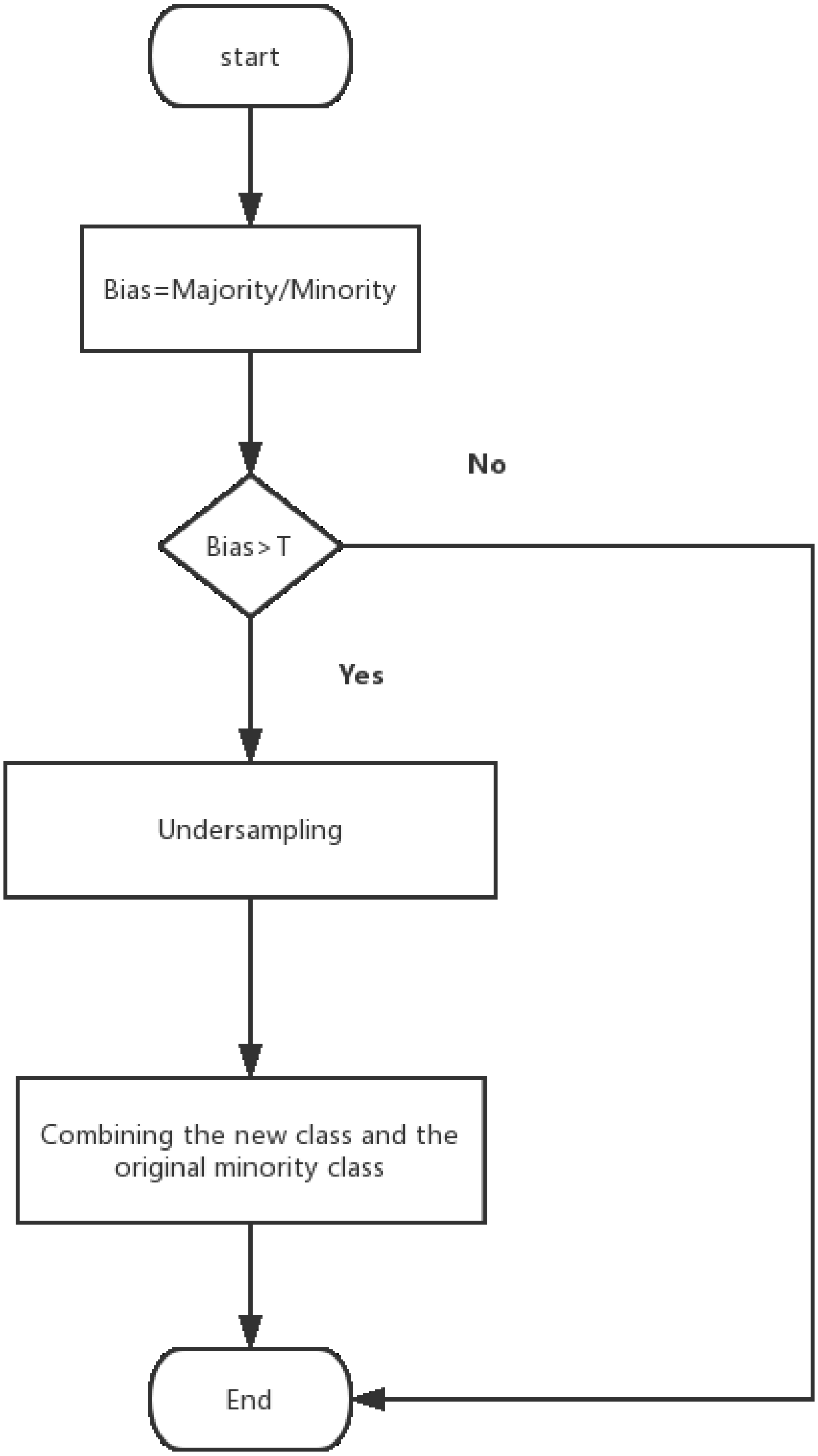

Figure 3.1 The process of undersampling.

In the process, Majority and Minority represent the number of instances in majority class and minority classes, respectively Bias stands for the bias ratio between the majority class and minority class which is Majority/Minority. *T* is the threshold we set by ourselves to evaluate whether we need to do the undersampling. If the bias is larger than the threshold, then we randomly undersample the majority class with the undersampling ratio we have set. After undersampling the majority class, we combine the new class with the minority class to form a new training dataset.

For example, assume that Majority = 10000 and Minority =10. *T*=10 and Bias=1000.

In this process, Bias > T. Hence, we randomly undersample the majority class with the undersampling ratio 1000 to form a new class with instance 10000/1000=10. After undersampling, we combine the new class and original minority class together, as a new training dataset with 10+10=20 instances.

The value of undersampling ratio *m* should be set appropriately. The value Too low value of *m* would result in low true positive rate of the minority class since it is still an imbalanced problem after undersampling with a low ratio.

3.2 Decision Tree-Based Undersampling GA-SVM

The different set of nodes in a decision tree strongly influence the performance of the decision tree, so in our works, we choose two different ways to build the nodes and make the comparison which are based on a class and based on the distance between two class patterns and the number of each class patterns.

For the first method, we set the class as a node in a decision tree. In other words, we separate the class with undersampling GA-SVM one against the rest. We continue separating each class until we meet the leaf implying there is only one class left. The process is shown in the Figure 3.2

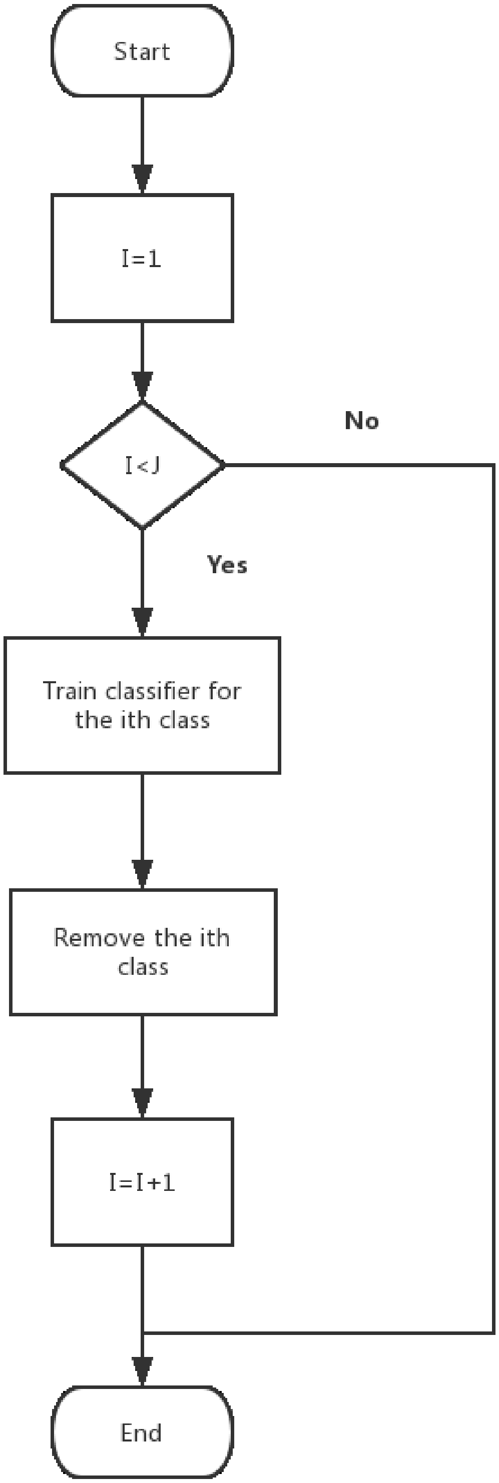

Figure 3.2 The process of decision tree.

Assume that we have a multiclassification problem with J classes. We set the first class as the first node to separate it from the others. We then remove the first class from the training dataset. We repeat that step until there is only one original class left.

The sequence of class in the nodes for classification is important to determine the performance of the classifier. We take a five classes' multicalssification problem for example and give the graph in Figure 3.3 to show the steps.

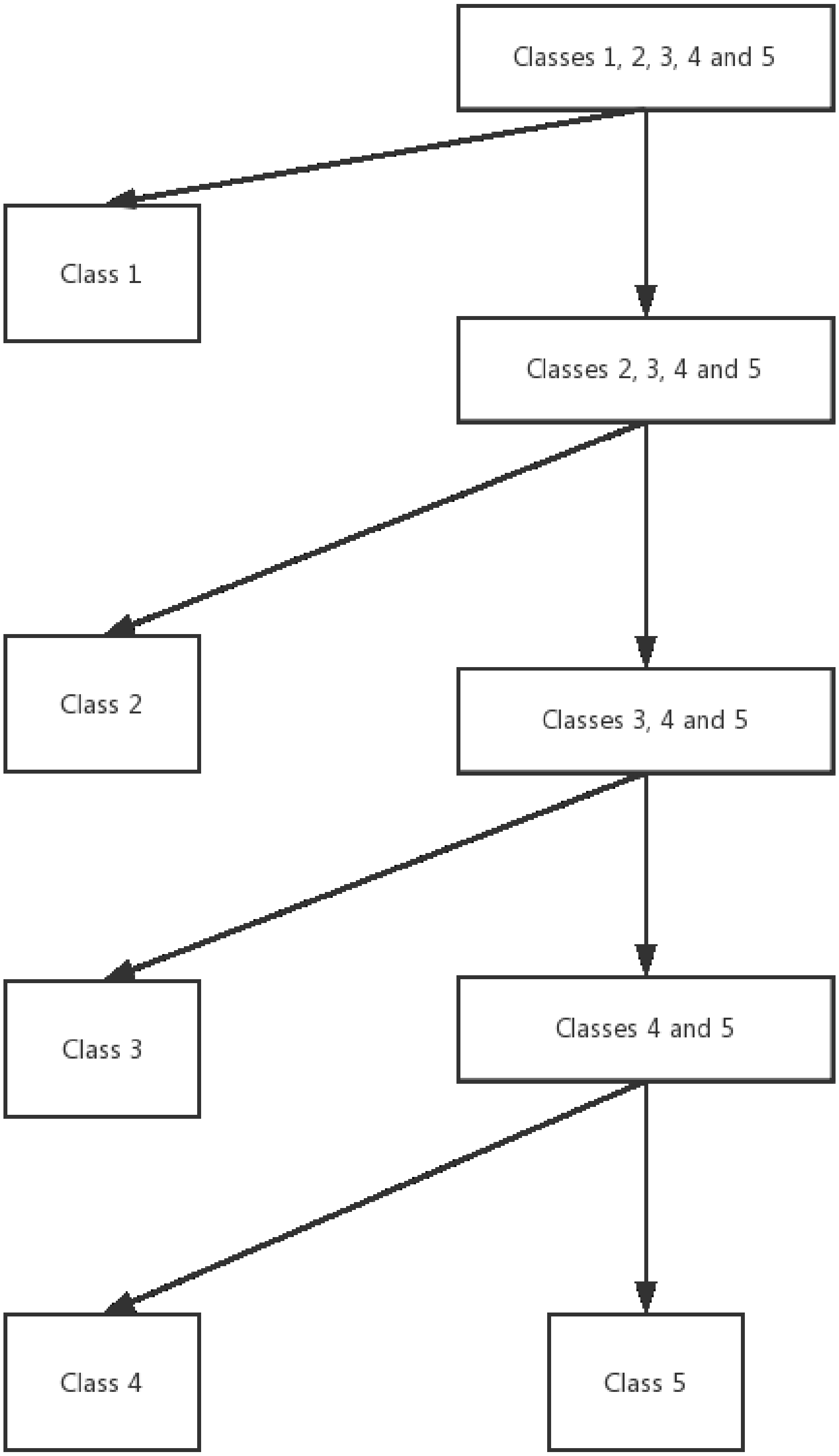

Figure 3.3 The decision tree-based SVM which nodes are based on classes.

We build the second type of tree-based on the distance between two classes and the number of instances in each class. [Mulay, 2010]

Assume that the number of instances in the *i*th class is x$_i$, the center point c$_i$ of the *i*th class is the mean value of all the points in the ith class. It can be calculated as follows:

$$c_i = \frac{\sum_{s=1}^{n_i} x_s^i}{n_i} \tag{3.1}$$

The distance between each two classes *i* and *j* is calculated by the Euclidian distance between their center points

$$E_{ij} = \left\| c_i - c_j \right\| \tag{3.2}$$

The following distance measures the distribution difference between the two class instances given as 3.3:

$$d_{ij} = \frac{Ed_{ij}}{\gamma_i + \gamma_j} \tag{3.3}$$

Where

$$\gamma_i = \frac{\sum_{s=1}^{n_i} \left\| x_s^i - c_i \right\|}{n_i} \tag{3.4}$$

The steps of how to set the node to separate the original class into two subsets is shown in Figure 3.4:

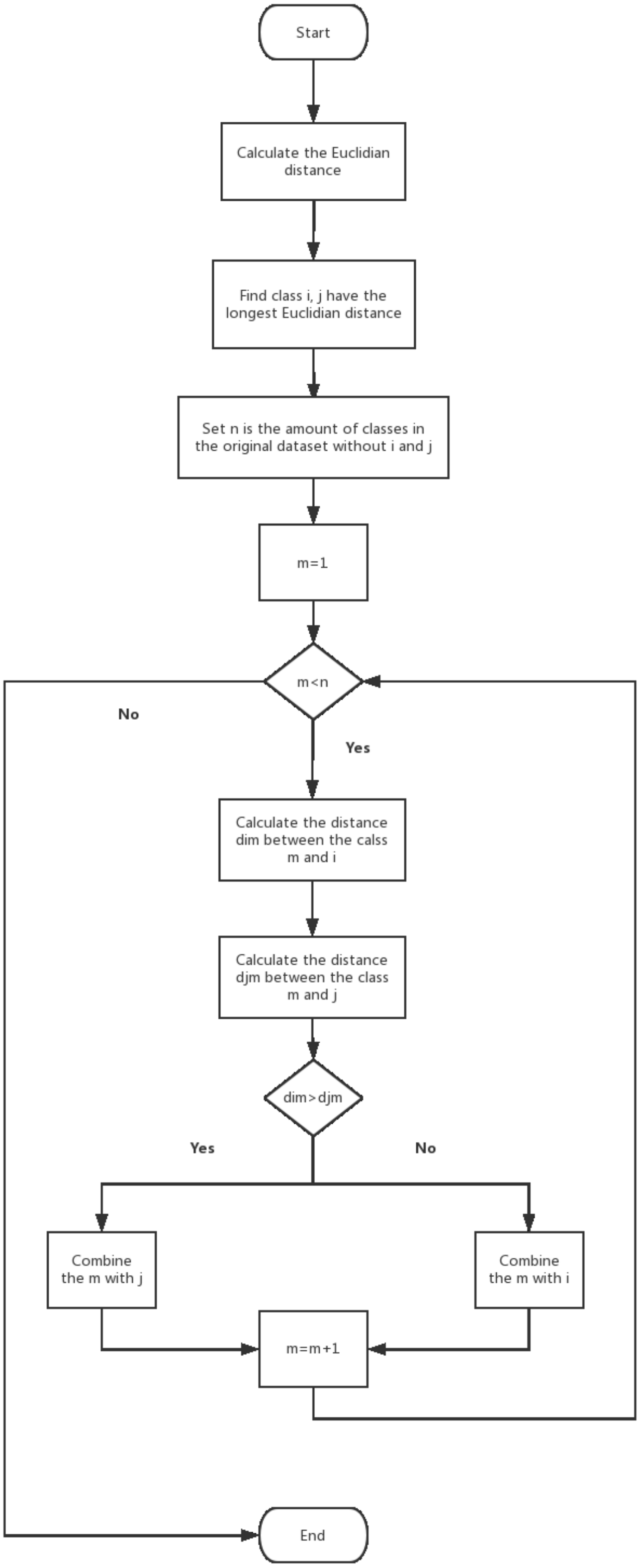
Start
Calculate the Euclidian distance
Find class i, j have the longest Euclidian distance
Set n is the amount of classes in the original dataset without i and j
m=1
m<n
No
Yes
Calculate the distance dim between the calss m and i
Calculate the distance djm between the class m and j
dim>djm
Yes
No
Combine the m with j
Combine the m with i
m=m+1
End

Figure 3.4 A process how to separate the original dataset into two classes.

In this process, we first calculate the Euclidian distance in Equation (3.2) between each pair of classes. We then find the largest distance between classes *i* and *j* and then separate the two classes. Then we pick one class and calculate the distance d_i and d_j in (3.3) with class *i* and *j*. If d_i is larger than d_j, we combine the class with class *j*. Otherwise we combine the class with class *i*. We repeat the step until all the classes has been separated into two classes.

For the whole algorithm, we separate the classes until there is only one class left.

For example, assume that we have a five classes multiclassification problem. We first calculate the Euclidian distance between each pair of classes and find that classes 1 and 4 have the largest distance. We then calculated the distance d_{12} between class 2 and 1 d_{42} between 2 and 4. We find that d_{12} is smaller than d_{24}. Thus we combine classes 1 and 2 together. We then repeat the same step with classes 3 and 5. We find that $d_{13}<d_{43}$ and $d_{15}>d_{45}$. Hence, we separate the dataset into two subsets classes 1 2 and 3 and classed 4 and 5.

The Figure 3.5 shows the structure of a decision tree-based undersampling GA-SVM for five classes.

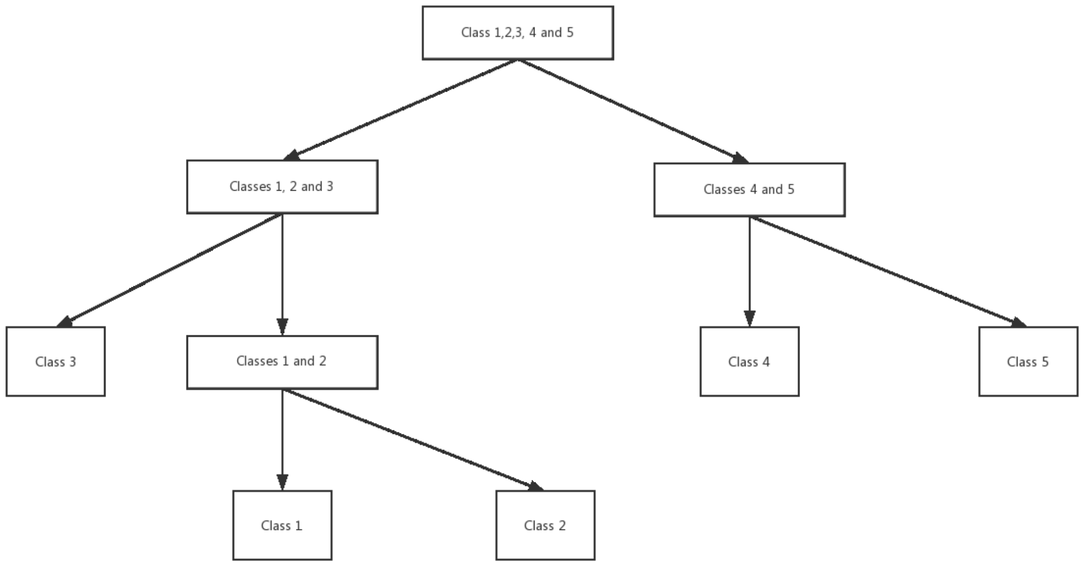

Figure 3.5 The decision tree-based SVM which nodes are based on the distance between classes.

3.3 Directed Acyclic Graph-based Undersampling GA-SVM

Another method for multiclassification of SVM is based on directed acyclic graph. In this algorithm, we replace the original classifier SVM in each node with the undersampling GA-SVM.

A directed acyclic graph refers to a finite directed graph without directed cycles.Superior to other traditional SVM applied in the multiclassification problem; the DAG helps the classifier improve the test efficiency.

We take a 4 class multiclassification problem as an example in Figure 3.6. The process of classification is shown in it. Assume that we are classifying a testing data in class 4.

For the first layer, we are tested whether it belongs to class 1 or 4. Thus we take the left branch to the node 2v4. Then the data is tested to see whether it belongs to class 2 or class 4. We still take the left branch. We stop the process until we touch the bottom layer.

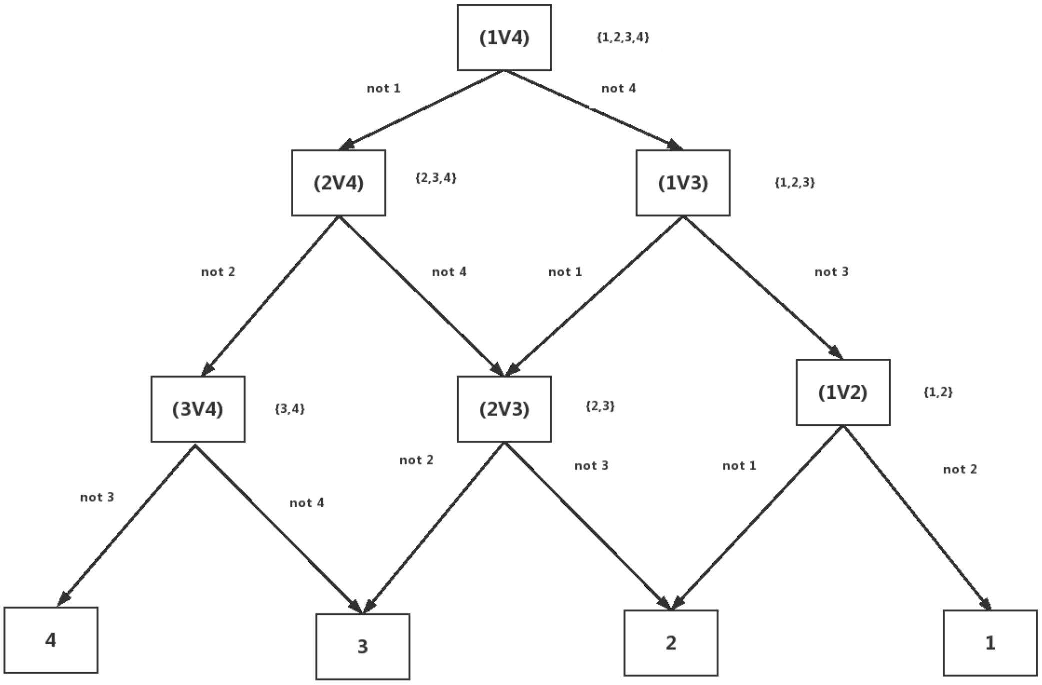

Figure 3.6 DAGSVM.

The sequences of the classifiers on each layer and the performance of the binary classification influence the result greatly. If the first layer misleads the result, the final result would immediately turn to be wrong.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Dataset

In the experiment, we take the dataset 10% corrected KDD CUP 99 which is data set used for The Third International Knowledge Discovery and Data Mining Tools Competition to test the performance of each classifier. The 10% corrected KDD CUP 99 dataset is a large collection of attack and normal instances. The dataset is the most popular dataset to evaluate the performance of the IDS. It contains 41 features, 32 of which are continuous and the rest are discrete. It has five classes: DoS, Normal, U2R, R2L, and Probe in the dataset. Their respective number of instances of the dataset is shown in Table 4.1.

Table 4.1 The 10% Corrected KDD CUP 99 Data

10% corrected KDD CUP 99 data					
Normal	DoS	Probe	U2R	R2L	Total
97298	391458	4107	52	1126	494021

In our experiment, we take 80% of the data as a training dataset and the rest 20% as a testing one. The quantities of training and testing dataset are shown in Table 4.2 - 4.3.

Table 4.2 The Quantities of Training Data

Training data					
Normal	Attack				Total
	DoS	Probe	U2R	R2L	
77698	313290	3278	42	908	395216

Table 4.3 The Quantities of Testing Data

Testing data					
Normal	Attack				Total
	DoS	Probe	U2R	R2L	
19580	78168	829	10	218	98805

In our experiment, we separate this intrusion detection problem into binary classification and multiclassification problems. We first test the influence on the number of generations in GA and undersampling ratio in GA-SVM in a binary classification problem.

Then we test decision tree-based undersampling GA-SVM and DAG-based undersampling GA-SVM in a multi classification problem.

4.2 Performance of GA-SVM

In this experiment, we choose the number of generations from range 10 to 40, i.e., 10, 20, 30, 40 in GA. The number of particles equals 50. The purpose of our GA-SVM is to make the binary classification between normal and the attack classes (DoS, U2R, Probe, and R2L). For SVM, RBF is chosen as the kernel function. The result is shown in Table 4.4.

Table4.4 True Positive Rate (TPR) with Different Generation

Generations	10	20	30	40
TPR (%)	95.8	96.3	98.7	98.9

From the Table 4.4, we conclude that GA-SVM vields higher TPR with the number of generations. The TPR improvement is influenced less and less with more and more generations.

4.3 Performance of Undersampling GA-SVM

In this experiment, we compare the TPR performance between undersampling GA-SVM and GA-SVM. Since the number of instances in U2R is too small, so we choose the second smallest class R2L as the minority class, and other classes as the majority class. Then we make the binary classification between R2L and all the others (normal, DoS, U2R and

Probe). The bias ratio between R2L and all the others is 1:499. We set the undersampling ratio as 1:200 which means that we randomly pick up one instance among 200 instances in the majority class. The comparison result is shown in Table 4.5.

Table 4.5 True Positive Rate (TPR) of GA-SVM vs. Undersampling GA-SVM

Method	GA-SVM	Undersampling GA-SVM
TPR(%)	91.2	98.5

The result in Table 4.5 shows that the undersampling method improves the performance of TPR in an imbalanced problem. Further, we want to find how the undersampling ratio influences the performance of TPR.

The undersampling ratio is set at1:1, 1:50, 1:100, 1:200 and 1:500. The result is shown in Table 4.6.

Table 4.6 The TPR of different undersampling ratio

Undersampling ratio	1:1	1:50	1:100	1:200	1:500
TPR (%)	91.2	96.7	98.5	98.5	98.5

From the result, we conclude that the undersampling ratio influences the performance of TPR. When the undersampling ratio is larger than some threshold, the TPR remains the same value. Moreover, traditionally, we prefer the ratio between majority and minority classes equals 1:1 after undersampling. However, from the experiment, we conclude that the numbers of instances between the majority and minority classes after undersampling are not required to be equal since 1:100 and smaller ratios have achieved the best result. Specifically, when selecting the undersampling ratio equals 1:100, we can calculatethe imbalanced ratio that equals 1:50.This is against the conventional wisdom.

4.4 Performance of Decision Tree-Based Undersampling GA-SVM

We test the dataset with our proposed decision tree-based undersampling GA-SVM. The first type of nodes is based on the class. Since the SVM is a binary classifier, so when we classify the five classes we just need to set four nodes.

Table 4.7 shows the accuracy among five classes. The undersampling ratio in the algorithm equals1:1000 in U2R to others, 1:200 in R2L to others, 1:10 in probe to others. The kernel function in our SVM is RBF. We compare the results between the method with and without undersampling with the node sequence U2R, R2L, Probe and Normal according to the increasing number of instances in each class. We also compare the result with the reverse node sequence: DoS, Normal, Probe and R2L. In this problem, we test the accuracy to identify each class.

Table 4.7 The Accuracy of Decision Tree-based Undersampling GA-SVM with Nodes Sequence U2R, R2L, Probe and Normal

Class	Original	Predicted	Accuracy (%)
U2R	10	6	60.00
R2L	218	208	95.41
Probe	829	781	94.21
Normal	19580	19502	99.60
DoS	78168	78082	99.89
Total	98805	98579	99.77

Table 4.8 The Accuracy of Decision Tree-based GA-SVM with Nodes Sequence U2R, R2L, Probe, and Normal

Class	Original	Predicted	Accuracy (%)
U2R	10	3	30.00
R2L	218	206	94.49
Probe	829	763	92.03
Normal	19580	19521	99.70
DoS	78168	77988	99.77
Total	98805	98481	99.67

Table 4.9 The Accuracy of Decision Tree-based Undersampling GA-SVM with Nodes Sequence DoS, Normal, Probe and R2L

Class	Original	Predicted	Accuracy(%)
U2R	10	3	30.00
R2L	218	207	94.95
Probe	829	764	92.15
Normal	19580	19566	99.93
DoS	78168	78144	99.97
Total	98805	98684	99.88

Comparing the result between decision tree-based GA-SVM with and without undersampling in Tables 4.7-4.8, weconclude that the undersampling method for GA-SVM can improve the accuracy of the minority classes U2R, R2L and Probe and one of twomajority classes DoS.Comparing the result from decision tree-based undersampling GA-SVM in Tables 4.7 and 4.9, we find that the accuracy increases in the majority classes DoS and Normal, but decreases in the minority classes.

Secondly we test the decision tree-based undersampling GA-SVM based on distance. According to the algorithm, we find that the distance between classes Normal and U2R is the largest. We separate the combination of class U2R, R2L and Probe and combination of normal and DoS with undersampling ratio 1:40. Then we find that probe has the largest distance with U2R, so we separate Probe from the subset.

Table 4.10 The Accuracy of Decision tree-based Undersampling GA-SVM

Class	Original	Predicted	Accuracy(%)
U2R	10	2	20.00
R2L	218	210	96.33
Probe	829	796	96.01
Normal	19580	19503	99.61
DoS	78168	78166	99.99
Total	98805	98677	99.87

We find that this tree structure based on the distance between two class instances and the number of instances in each class achieves the best accuracy for DoS, Normal, Probe and R2L. The accuracy of U2R is low. It may result from the extreme imbalanced ratio in the dataset.

4.5 Performance of DAGSVM

In this experiment we test a DAGSVM model based on different undersampling ratio to build two sets of 10 binary undersampling GA-SVM classifier groups. In Tables 4.11-4.14, we use 1 to 5 to represent the classes: DoS, Normal, R2L, U2R and Probe, respectively. Classifier 1v2 means that we train the binary classifier between class DoS and Normal. For this classifier, we pick the required classes of data from the training and testing datasets to form a new dataset. The undersampling ratio for each binary classifiers is shown in Table 4.11.

Table 4.11 Undersampling Ratio for the Binary Classifier

Classifier	Undersampling ratio
1v2	4:1
1v3	100:1
1v4	700:1
1v5	30:1
2v3	11:1
2v4	100:1
2v5	12:1
3v4	1:1
3v5	1:1
4v5	1:23

Table 4.12 The TPR of 10 Undersampling Binary GA-SVM

Classifier	acuracy(%)
1v2	99.86
1v3	90.83
1v4	10
1v5	95.66
2v3	89.45
2v4	20
2v5	77.44
3v4	100
3v5	90.37
4v5	10

Table 4.13 The TNR of 10 Undersampling Binary GA-SVM

Classifier	Accuracy(%)
1v2	99.97
1v3	100
1v4	100
1v5	99.98
2v3	99.98
2v4	99.99
2v5	99.95
3v4	87.6
3v5	100
4v5	100

The result of DAGSVM is shown in Table 4.14.we build the DAGSVM based on the accuracy =(TPR+TNR)/2.We then randomly change the nodes in DAGSVM and obtainthe result in Table 4.15.

Table 4.14 The Result of DAGSVM Set Based on Accuracy

Attack Class	Accuracy (%)
U2R	10
R2L	89.44
Probe	95.29
Normal	96.76
DoS	99.84

Table 4.15 The Result of DAGSVM set Randomly

Attack Class	Accuracy(%)
U2R	10
R2L	89.44
Probe	76.23
Normal	98.93
DoS	89.44

From the result, we conclude that the accuracy decreases for Normal, but increases for Probe and DoS, if we build the model based on the accuracy. Thus we conclude that the accuracy of each binary classifier does not influence greatly the classification performance. We then change the set of 10 binary classifiers with higher TPR in each SVM and build DAGSVM with the high accuracy.

Table 4.16 the TPR of 10 Undersampling Binary GA-SVM with Higher TPR

Classifier	Accuracy (%)
1v2	99.86
1v3	90.83
1v4	100
1v5	95.66
2v3	89.45
2v4	20
2v5	77.44
3v4	90
3v5	90.37
4v5	90

Table 4.17 The TNR of 10 Undersampling Binary GA-SVM with Higher TPR

Classifier	Percentage(%)
1v2	99.97
1v3	100
1v4	89.04
1v5	99.98
2v3	99.98
2v4	99.99
2v5	99.95
3v4	93.42
3v5	100
4v5	80.27

Table 4.18 Result of DAGSVM with 10 SVMs

Class	Accuracy(%)
U2R	10
R2L	89.44
Probe	95.17
Normal	95.76
DoS	99.84

From Table 4.18 we find that the accuracy of DAGSVM is not influenced by TPR in the binary classifier. Even though we increase the accuracy of each binary classifier, the accuracy for the minority class does not change greatly.

4.6 Summary

We have proposed an undersampling GA-SVM method to solve a multiclassification problem with decision tree structure and DAG. The decision tree based GA-SVM obtains better performance than DAGSVM. The accuracy for classes R2L, Probe, Normal and DoS is the highest, when we choose the nodes based on the distance between two classes and the number of instances in each class. The class U2R cannot be classified well perhaps because the number of instances in U2R is extremely small.

CHAPTER 5

CONCLUSION

5.1 Summary of Contribution of This Thesis

This thesis proposes undersampling GA-SVM methods to solve a multiclassification problem with two different structures: decision tree and DAG. They can be used to deal with imbalanced multiclassification problems in a network intrusion detection system, This thesis makes the following contributions.

(1) Making literature review about intrusion detection systems, support vector machines, decision tree and genetic algorithm.

An intrusion detection system is highly demanded in the current society. To design an efficient and effective system, we apply machine learning to solve this multiclassification problem. An effective classifier method can accurately identify given a dataset. A support vector machine is a classifier that performs well in binary classification. A decision tree is a classifier that can be built up by using a different splitting criterion. Its advantage is the ease of detecting which feature is used as a splitting node. Genetic algorithm can help to find the optimal value while an undersampling method is often used to solve an imbalanced problem.

(2)Proposing two methods for multiclassification: decision tree and DAG

For the decision tree method, we select its nodes by classes and the distance between two class patterns and the number of each class pattern.

(3) Comparing with two decision tree methods and a DAG method and analyzing their experimental results.

Decision tree and DAG methods are tested through the 10% corrected KDD CUP 99. According to the experimental results, the decision tree-based GA-SVM performs better than DAGSVM. The accuracy to classify classes R2L, Probe, Normal and DoS is the highest when we choose the nodes based on the distance between two class patterns and the number of each class pattern. The class U2R does not perform well perhaps because the number of U2R instance in the dataset is extremely small.

5.2 Limitations and Future Work

There are some limitations in our methods. A decision tree method performs better with higher accuracy for class R2l, Probe, Normal and DoS. However, the accuracy of U2R remains low. Also, the decision tree is influenced by the selection of nodes. It is hard to say the proposed methods are the best especially after preprocessing. The dataset was more than 10 years ago, so some of the information in the dataset may be out of date in the current situation.

As future work, one of them is to add a preprocessing step to improve the accuracy. Another direction is to find better a way to build the decision tree. For example, we will try KNN method to determine the nodes.

REFERENCES

Yang Li, Binxing FANG, Li GUO, et al. "A Network Anomaly Detection Method Based on Transduction Scheme." *Journal of Software*, vol.18, no.10, ,pp.2595-2604 (October, 2007).

Xian Jia, Peiyu Liu, GONG Wei. "Research of intrusion forensics based on improved attribute Weighted Naive Bayes[J]. " *Computer Engineering and Applications*, 49(7): 81-84. (2003)

Jun Guo, Norikazu Takahashi, Wenxin Hu. (2008). “An Improved Algorithm for Multiclass Text Categorization with Support Vector Machine” *2008 International Symposium on Computational Intelligence and Design* (2008*)*

Ymahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. “A detailed analysis of KDD CUP’99 data set.” *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. (2009)

Snehal A. Mulay , P.R. Devale , G.V. Garje . “Intrusion Detection System using Support Vector Machine and Decision Tree.” *International Journal of Computer Applications* (0975 – 8887) Volume 3 – No.3 (June, 2010)

David S. Moore and George P. McCabe. “inroduction to the practice of statistics(5th edition).” *W.H. Freeman & Company*. ISBN 0-7167-6282-X. (February 2005)

Zimek, A, Schubert, E.;Kriegel, H.-P. “A survey on unsupervised outlier detection in high-dimensional numerical data.” *Statistical Analysis and Data Mining.* 5 (5): 363–387. (2012).

Salma Mahgoub Gaffer; Moawia Elfaki Yahia;Khaled Ragab. “Genetic fuzzy system for intrusion detection: Analysis of improving of multiclass classification accuracy using KDDCup-99 imbalance dataset.” *2012 12th International Conference on Hybrid Intelligent Systems* (HIS), (2012)

Shengnan Hao;Jing Hu;Songyin Liu;Tiecheng Song;Jinghong Guo;Shidong Liu “Network traffic classification based on improved DAG-SVM. ” *2015 International Conference on Communications, Management and Telecommunications (ComManTel)* Pages:256-261， (2015)

Laheebm. Ibrahiml, Dujant. Basheerl, Mahmod S. Mahmod. “A Comparison Study For Intrusion Database (KDD99, NSL-KDD) Based On Self Organization Map

(SOM).” *Artificial Neural Network Journal of Engineering Science and Technology*, Vol. 8(1). (February 2013)

Sakchi Jaiswal; Khushboo Saxena; Amit Mishra; Shiv K. Sahu “A KNN-ACO approach for intrusion detection using KDDCUP'99 dataset.” *2016 3rd International Conference on Computing for Sustainable Global Development* (INDIACom) Pages: 628 – 633 (2016)

QiuweiYang, HongjuanFu,TingZhuAn . “Optimization Method for Parameters of SVM in Network Intrusion Detection System.” *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)* Pages:136-142, DOI:10.1109/DCOSS. (2016.4.8)

Praneeth Nskh; M Naveen Varma; Roshan Ramakrishna Naik. “Principle component analysis based intrusion detection system using support vector machine.” *2016 IEEE International Conference on Recent Trends in Electronics*, (2016)

Biswajit Langthasa;Bikash Acharya; Satyajit Sarmah. “Information &Communication Technology(RTEICT) Classification of network traffic in LAN.” *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)* Pages:92-99,(2015).

Khaled Alrawashdeh, Carla Purdy "Toward an Online Anomaly Intrusion Detection System Based on Deep Learning." 2016 *15th IEEE International Conference on Machine Learning and Applications* (2016)

Chao Deng; Haiye Qiao “Network security intrusion detection system based on incremental improved convolutional neural network model” *2016 International Conference on Communication and Electronics Systems (ICCES)* Pages: 1 - 5, DOI: 10.1109/CESYS.2016.7889881 (2016)

Chen, X., Wang, M., and Zhang, H.. "The use of classification trees for bioinformatics." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1, no. 1 (2011): 55-63. (2011)

Zehra Cataltepe; Umit Ekmekci; Tanju Cataltepe; Ismail Kelebek, “Online feature selected semi-supervised decision trees for network intrusion detection.” *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium* Pages: 1085 - 1088, DOI: 10.1109/NOMS.2016.7502965 (2016)