ORIGINAL RESEARCH

Adaboost and SVM based cybercrime detection and prevention model

Hanif Mohaddes Deylami, Yashwant Prasad Singh

Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia

Correspondence: Hanif Mohaddes Deylami. Address: Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Selangor, Malaysia. Telephone: 006-017-871-7244. E-mail: hmdeilami@gmail.com.

Received: January 23, 2012 **Accepted:** June 6, 2012 **Published:** December 1, 2012
DOI: 10.5430/air.v1n2p117 **URL:** http://dx.doi.org/10.5430/air.v1n2p117

Abstract

This paper aims to propose cybercrime detection and prevention model by using Support Vector Machine (SVM) and AdaBoost algorithm in order to reduce data damaging due to running of malicious codes. The performance of this model will be evaluated on a Facebook dataset, which includes benign executable and malicious codes. The main objective of this paper is to find the effectiveness of different classifiers on the Facebook dataset for crime detection. Finally, we try to compare the classifier accuracy of SVM and AdaBoost by using Weka 3.7.4 software in order to choose the best model to classify the Facebook dataset with high accuracy.

Key words

Artificial Intelligence, Support Vector Machine, AdaBoost, AdaSVM, Cybercrime, Social Network

1 Introduction

Society has grown to rely on Internet services, where data and information are collected and stored in unprecedented volumes. The large and small enterprises collect data and information in various aspects such as: businesses, customers, human resources, products, and suppliers, which are opening the window of opportunities for malicious users and crooks. This paper presents cybercrime detection and prevention model by using Support Vector Machine (SVM) and Adaboost algorithm to detect and classify of malicious codes in Facebook dataset.

Boser, Guyon, and Vapnik in COLT-92 [1, 2] first introduced support Vector Machine (SVM) in 1992. Support vector machines (SVMs) are a set of related supervised learning methods deployed for regression and classification [3]. The fundamentals of Support Vector Machines (SVMs) have been enhanced by Vapnik [4] and became popular due to many promising features such as: better empirical performance. SVMs can be considered as techniques, which use the hypothesis space of linear separators in a high dimensional feature space, trained with a learning algorithm from optimization theory that makes a learning bias, derived from statistical learning theory. The SVM technique was developed to design separating hyperplanes for classification problems (see Figure 1) [5]. Boosting methods are used for solving the classification problems and give weak learners as an input and then try to make a strong learner as an output [6, 7]. AdaBoost, short form of adaptive boosting, is a machine-learning algorithm, introduced by Freund and Schapire [8]. It is a meta-algorithm, and can be deployed with many other learning algorithms to improve their performance.

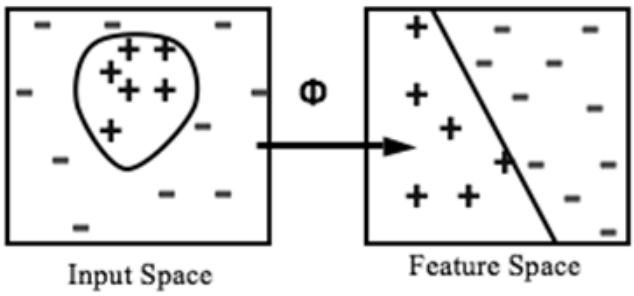

Figure 1. Separating Hyperplane in Feature Space [5]

The rest of the paper is organized as follows. In section 2 represents data mining and machine learning techniques. In section 3 include technical details about SVM algorithms. In section 4 elaborate about AdaBoost and AdaSVM and connection between these two methods. The experimental results are given in section 5, which presents the effectiveness of AdaSVM over SVM algorithms. Section 6 is follow by a conclusion and future works.

2 Data mining and machine learning techniques

2.1 Data mining

Data mining techniques are used to extract implicit information from a massive volume of unexplored, circumstantial, and potentially beneficial information from dataset. Data mining techniques include Statistical Techniques, Hierarchal Clustering, Decision Trees, Nearest Neighbor Classification, Neural Networks, and Rule Induction to extract hidden information from database with high performance, and reliability [9].

2.2 Machine learning

Machine learning is a field of Artificial Intelligent (AI), which automatically learns to predict based on experience and observation of data for making intelligent decisions. Machines learning tries to obtain knowledge by learning samples for developing and solving mining problems [10, 11]. The machine learning algorithms can be mainly categorized into many kinds, which elaborate just two kinds based on the paper goals [11, 12].

Supervised and unsupervised machine learning

Supervised learning methods for classification and regression are relatively new class of successful learning methods - they can represent linear/nonlinear functions and they have an efficient training algorithm derived from statistical learning theory by Vapnik and Chervonenkis [5]. Supervised learning is a way to design a classifier function, which analyzes and classifies the training data in order to predict output against any reliable input with high accuracy [13]. Unsupervised machine learning is a way to design a classifier function, which try to find solidarity of raw data without any training or external inputs. The unsupervised machine learning lets algorithms for deciding to which group of data is useful.

3 Technical details about Support Vector Machine (SVM) algorithms

The SVM method was developed to construct separating hyperplanes for classification problems [1]. SVM constructs functions (hyperplanes either in input space or in feature space) from a set of labeled training data. This hyperplanes try to split the positive samples from the negative samples. The split can be selected the largest distance from the hyperplanes to the nearest of positive and negative samples. SVM algorithms treat each sample in the matrix as a vector in a high

dimensional feature space, where the number of attributes identifies the dimensionality of the space. Then, the trained SVM can be used to make predictions about a test sample's membership in the class [5, 15, 16].

3.1 Binary Support Vector Machine

The SVM originally developed for the binary classification problem, has a nice geometric interpretation of discriminating one class from another by constructing a hyperplane with the maximum margin [14, 15].

3.1.1 Maximum margin hyperplanes

The linear separable class includes many kinds of boundaries with different margins. The decision margins have to be as far from the samples of both classes. Figure 2 represents a plot of a dataset containing samples that belongs to two different classes, represented as plus and mines. We can find a hyperplane such that all positive samples reside on the one side of hyperplane and all of negative samples reside on the other side of the linearly separable dataset. The maximum margin can be established by putting parallel decision boundaries with the closest distance from positive and negative samples, and then reside the hyperplane away from of these samples. Figure 2 presents two decision boundaries B_1 and B_2. Both of these decision boundaries can separate the training samples into their respective classes without any misclassification errors. Each decision boundary B_i is related to a pair of hyperplanes, denoted as b_{i1} and b_{i2}, respectively. The b_{i1} is established by moving a parallel hyperplane away from the decision boundary until it obtains the closest negative samples (-), while b_{i2} is established by moving a hyperplane until it obtains the closest positive samples (+). The distance between these two hyperplanes in Figure 2 is named as the margin of the classifier. As it appears in Figure 2 the margin for B_1 is larger than for B_2, so B_1 considered being the maximum margin hyperplane of the training instances. This is the simplest kind of SVM (Called a Linear SVM) [17-19].

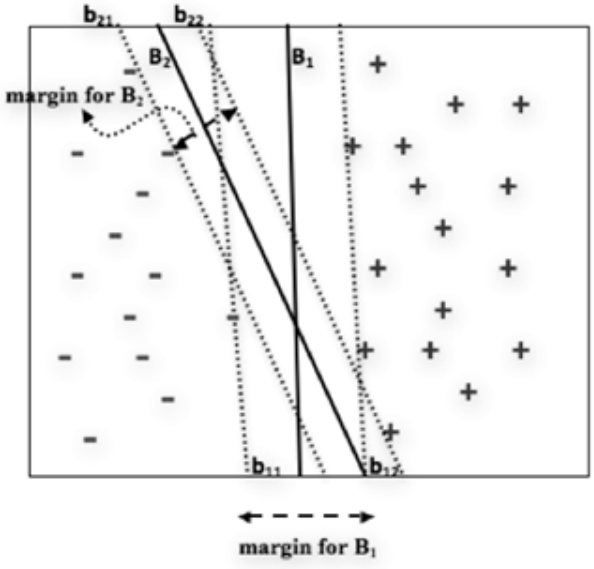

Figure 2. Margin of Decision Boundaries [17]

3.1.2 Linear separable SVM classifier

A linear SVM is a classifier that looks for a hyperplane with maximum margin classifier (see Figure 3). The classification algorithm for solving linear problem can demonstrate as a below:

The input $X=(x_1, \ldots,x_n)$ with $f(X) \geq 0$ is assigned to the positive class, otherwise it belongs to the negative class. We consider the case where $f(X)$ is a linear function of $x \in X$, and then it can be written as a following function. The decision boundary of a linear classifier can be written as an equation (1):

$$f(X) = <w.x> + b = \sum_{i=1}^{n} w_i x_i + b \quad (1)$$

Where $(w, b) \in R^n \times R$ are the parameter that control the function and the decision rule are given by $\text{sgn}(f(x))$, where we will use the convention that $\text{sgn}(0) = 0$. The learning methodology implies that these parameters must be learned from the dataset [16].

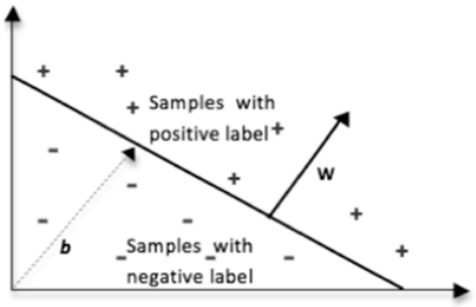

Figure 3. Separating hyperplane in two dimensional training set [16]

3.1.3 Hard-Margin SVM classifier

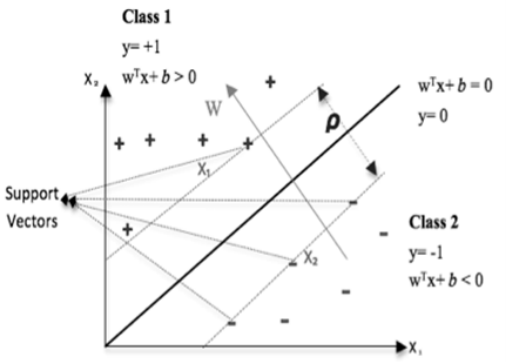

Figure 4. Linear separating hyperplanes in SVMs

Hard-Margin Support Vector Machine (HM-SVM) is a method that uses in linear separated support vector machine. Figure 4 represents two classes in N-dimensional training inputs x_i (i=1, 2, …, N), which $y_i = +1$ belongs to class 1 and $y_i = -1$ belongs to class 2 (see Figure 4). We assume these dataset is linearly separable, and then the decision function can be represented by equation (2).

$$D(x) = w.x + w_0 \quad (2)$$

$$w^T x_i + w_0 \begin{cases} > 0 \text{ , if } y_i = +1 \\ < 0 \text{ , if } y_i = -1 \end{cases}$$

While the training data is non-linearly separable, then the equation (2) is not appropriate to classify dataset. Therefore, this equation is changed to the equation (3).

$$D(x) = w.x + w_0 \tag{3}$$

$$w^T x_i + w_0 \begin{cases} \geq +1 \text{ , if } y_i = +1 \\ \leq -1 \text{ , if } y_i = -1 \end{cases}$$

Here, +1 and -1 on the right-hand sides of the inequalities can be considered as any constant a (a>0) and -a, respectively. But by dividing both sides of the inequalities by a, then the equation (3) is equal equation (4):

$$y_i (w^T x_i + w_0) \geqslant 1 \qquad \text{for } i=1, 2, \ldots, N \tag{4}$$

The hyperplane forms a separating hyperplane that separates X_i (i=1, 2, …, N)

$$D(x) = w^T x + w_0 = c \quad \text{for } -1 < c < +1 \tag{5}$$

While c=0, the separating hyperplane located in the middle of the two hyperplanes and when the c=+1 or c=-1 then the distance between the separating hyperplane and the training instances nearest to the hyperplane. The generalization ability depends on the location of the separating hyperplane, and the hyperplane with the maximum margin is selecting as an optimal separating hyperplane.

3.1.4 Soft margin SVM classifier

We assume that the training data are linearly separable in Hard Margin SVM. When the data cannot be separated linearly, the Hard Margin support vector machine won't be practical, so we should map data from an input space to feature space to support nonlinear classification problems. In this section we extend the SVM so that it is applicable to an inseparable case (see Figure 5). To allow inseparability, we consider the nonnegative slack variables $\xi_i (\geqslant 0)$ into the equation (6):

$$y_i (w^T x_i + b) \geqslant 1 \text{ for } (x_i, y_i), \; i = 1, 2, \ldots, N \tag{6}$$

$$\Phi(w) = w^T w + C \sum \xi_i \tag{7}$$

and for (x_i, y_i), i = 1, 2, …, N

$$y_i (w^T x_i + w_0) \geqslant 1 - \xi_i \tag{8}$$

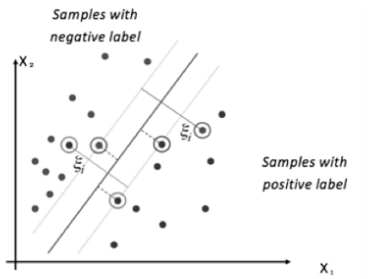

Figure 5. Slack variables for linearly non-separable data [20]

By the slack variables (ξ_i), possible solutions always exist. For the training data X_i, if $0 < \xi_i < +1$ (see Figure 5) the data will not have the maximum margin but are still correctly classified. If $\xi_i \geqslant +1$ (see Figure 5), the data are misclassified by the optimal hyperplane. To achieve the optimal hyperplane in which the number of training data that do not have the maximum margin is minimum, we need to minimize

$$Q(w) = \sum_{i=1}^{N} \Theta(\xi_i) \tag{9}$$

Where

$$\Theta(\xi_i) = \begin{cases} 1, \text{ for } \xi_i > 0 \\ 0, \text{ for } \xi_i = 0 \end{cases} \tag{10}$$

3.1.5 Mapping to a high-dimensional space

Support vector machine uses n-dimensional data point's x_i (i=1, 2, …, N) to change non-linear function in complex low dimension to linear in the simple high dimension in feature space for handling classification problems. SVMs algorithm can be done this approach by using kernel maps functions [12, 21]. The four commonly used families of the kernels are:

- Linear Kernel: $K(x, y) = x^T y + c$ (11)
- Polynomial Kernel: $K(x, y) = (x^T y + c)^d$ (12)
- Radial Basis Function Kernel: $K(x, y) = \exp(- \| x - y \|^2 / (2\sigma^2))$ (13)
- Puk Kernel: $K(x_i, x_j) = 1 / [1 + ((2 \sqrt{} \| x_i - x_j \|^2 \sqrt{} 2^{1/w}-1) / \sigma)^2]^w$ (14)

3.2 Sequential Minimal Optimization (SMO) algorithm

Sequential Minimal Optimization (SMO) is an algorithm, which is used for solving large quadratic programming (QP) optimization problems, widely deployed for the training of support vector machines [14]. SMO breaks these large QP problems into a series of smallest possible QP problems, and then these small QP problems are solved analytically, which prevents using a time-consuming numerical QP optimization as inner loop. SMO is fastest for linear SVMs and sparse datasets, due to SMO's computational time is dominated by SVM evolution.

4 Technical details about AdaBoost and AdaSVM

4.1 AdaBoost

AdaBoost, short form of Adaptive Boosting, is a machine-learning algorithm, introduced by Freund and Schapire in 1999 [22, 23]. It is a meta-algorithm, and able to deploy with many other learning algorithms for improving performance. The algorithm takes a training set (x_1, y_1), …, (x_m, y_m) as input where each x_i belongs to input space X, and each label y_i is in some label set Y. AdaBoost algorithm supports a distribution or set of weights over the training set. Initially, all weights equally set, but on each round, the weights of incorrectly classified samples are increased then the weak learner focuses on these samples [23]. AdaBoost originally ability to minimize the error, and maximize the margin with respect to features.

Algorithm: AdaBoost

Input: N training samples $(x_1, …, x_N)$ with $x_i \in X$, corresponding labels $(y_1, …, y_N)$ with $y_i \in \{0, 1\}$, and initial distribution of weights $D_1(i)$ over the examples. For t = 1, …, T:

1. Train a weak classifier $h_t : X \in \{0, 1\}$ using distribution D_t

2. Calculate the error of

   h_t: $\varepsilon_t = \sum_{i=1}^{N} D_t(i)\, I(y_i \neq h_t(x_i))$

3. Set $\alpha_t = -1/2 \log(\varepsilon_t / (1-\varepsilon_t))$

4. Set $D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i h_t(x_i)) / Z_t$,

   where $Z_t = 2\sqrt{\varepsilon_t (1-\varepsilon_t)}$ is a normalization factor.

   Output the strong classifier $H(x) = \text{sign}(f(x))$, where $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) / \sum_{t=1}^{T} \alpha_t$

This algorithm doing repeatedly in a series of cycle, which assumed the input, is a set of training samples with labels. The constant α as a component learner algorithm and the number of cycles is represented by T. Then, AdaBoost at the cycle t provides training samples with distribution D_t to component learn. In response, the component learns trains a classifier h_t. The distribution D_t is updated after each cycle according to the prediction results on the training samples. AdaBoost more considerate on the samples with higher weights, which seem to be harder for component learn. This process continues for T cycles, and finally AdaBoost linearly combines all the component classifiers into a single final hypothesis f, as well as the greater weights are given to component classifiers with lower training errors. The significant theoretical feature of AdaBoost is that component classifiers need to be only slightly better than random [25].

4.2 AdaSVM (AdaBoostSVM)

When applying the Boosting method to strong component classifiers, these component classifiers must be appropriately weakened in order to benefit from Boosting [25]. Hence, if PolynomialSVM is used as a component classifier for AdaBoost, a relatively large "d" value, which corresponds to a PolynomialSVM with relatively weak learning ability, is preferred. In the proposed algorithm, without loss of generality, the re-weighting technique is used to reflect the weights of training samples in the training process. Proposed AdaBoostSVM methods can be described as follows: Firstly, a large value is set to "d", which corresponds to a PolynomialSVM classifier with very weak learning ability. Then, PolynomialSVM with this "d" is used in as many cycles as possible as long as more than half accuracy can be obtained. Otherwise, this "d" value is decreased slightly to increase the learning capability of the PolynomialSVM to help it achieving more than half accuracy. Since we only decrease the "d" value slightly, this prevents the new PolynomialSVM from being too strong for the current weighted training samples, and thus moderately accurate PolynomialSVM classifiers can be secured. The reason why moderately accurate PolynomialSVM component classifiers are favored lies in that these classifiers often have a larger diversity than those component classifiers, which are very accurate. These larger diversities may lead to a better generalization performance of AdaBoost. This process continues until the "d" is decreased to the given minimal "d" value.

Algorithm: AdaSVM

Input: N training samples $(x_1, \ldots, x_N)$ with $x \in X$, corresponding labels $(y_1, \ldots, y_N)$ with $y_i \in \{0, 1\}$, the initial d, d_{ini}; the minimal d, d_{min}; the step of d, d_{step}, an initial distribution of weights $D_1(i)$ over the examples for all i=1, …,N.

Do While ($d > d_{min}$)

1. Use PolynomialSVM to train a component classifier $h_t: X \rightarrow \{0, 1\}$ using distribution D_t

2. Calculate the training error of h_t: $\varepsilon_t = \sum_{i=1}^{N} D_t(i)\, I(y_i \neq h_t(x_i))$

3. If $\varepsilon_t > 0.5$, decrease d value by d_{step} and go to 1

4. Set $a_t = -1/2 \log (\varepsilon_t / (1-\varepsilon_t))$

5. Update the set $D_{t+1}(i) = D_t (i) \exp (-a_t y_i h_t (x_i)) / Z_t$,

   where $Z_t = 2 \sqrt{\varepsilon_t (1- \varepsilon_t)}$ is a normalization factor

   The strong classifiers H(x)=sign (f (x)), where $f(x)= \sum_{t=1}^{T} a_t h_t(x) \ / \ \sum_{t=1}^{T} a_t$

4.3 Connection between SVMs and Boosting

It is a common folklore statement that Boosting and SVMs are "essentially the same" except for the way they measure the margin or the way they optimize their weight vector: SVMs use the ℓ_2-norm and Boosting employs an ℓ_1-norm. Gunnar and Scholkopf [24] try to compare between Boosting and SVMs by using two different strategies in high or even infinite dimensional spaces. SVM by using kernel trick try to use ℓ_2-norm in order to compute scalar products in feature space. Boosting by using ℓ_1-norm or another sparseness including regularization functional try to rely on the fact that there are only a few hypotheses necessary to express the solution during each iteration. Boosting relies on the fact that there are only a few hypotheses necessary to express the solution, which tries to find during each iteration. Basically, Boosting considers only the most salient dimensions in the feature space spanned by the hypotheses and can be efficient. Also on the level of the mathematics programs we can see the relations between Boosting and SVMs: The equations (16) and (17) clearly similar for p=1 [24, 25].

$$y_n (w.\Phi (x_n)) \geqslant p, \quad n = 1, \ldots, N \tag{16}$$

$$\max, w \in F, p \in R, \|W\|_p = 1$$

$$y_n \sum_{j=1}^{J} w_j h_j (x_n) \geqslant p , \quad n = 1, \ldots, N \tag{17}$$

$$\max w \in R_J + p \in R, \|W\|_1 = 1$$

To make this explicit, note that any hypothesis set H implies a mapping Φ by

$$\Phi : x \rightarrow [h_1(x), \ldots, h_J(x)]^T \tag{18}$$

and therefore also a kernel $K(x, x') = \sum_{j=1}^{J} hj(x)\, hj(x')$. Thus, any hypothesis set H spans a feature space F. Furthermore, for any feature space F, which is spanned by some mapping Φ, the corresponding hypothesis set H can be constructed by $h_j = P_j \Phi$, where P_j denotes the projection onto the j-th dimension in feature space [24].

5 Experiments on SVM and AdaBoost to classify Facebook dataset

This section presents the experimental results by using SVM and AdaBoost algorithm to classify Facebook dataset with high percentage of accuracy. SVM and AdaBoost algorithms are chosen suitable parameters (C and Number of Iterations, respectively) to optimize problems in order to improve the percentage of classification accuracy. After that, we are trying to compare these algorithms to find a high percentage of accuracy.

 ISSN 1925-4040 E-ISSN 1925-4059

5.1 Dataset characteristics and system requirements

In this section, we provide experimental results on Facebook-Dataset from Max Plank institute for software system (http://socialnetworks.mpi-sws.org/data-wosn2009.html). The Facebook dataset includes 2700 samples. By using the test option as a cross-validation equal ten folds, then the dataset divided into 2430 (Training data = (2700/10)*9) samples as a training data and 270 samples (Test data=2700-2430) as a test data (see Table 1). The dataset includes two classes (Class "a": Non-Cybercrime, Class "b": Cybercrime) and consists of the following six attributes as listed in Table 2. Each line of dataset contains two unknown user identifiers (The second user posts on the first user's Facebook wall) and the other columns are number of frequent messages, frequent post, frequent business, and frequent application, respectively. All classification methods are implemented in Weka 3.7.4. To compare the CPU time and accuracies of mentioned algorithms we run Weka on Mac OS X version 10.6.8 running on a Laptop with system configuration Intel Core 2 Duo processor (2.4GHz) with 4GB 1067 MHz DDR3 of RAM.

Table 1. Information of Facebook dataset

Dataset Samples	#Sample	#Class	#Attribute
Training Samples	2430	2	6
Testing Samples	270	2	6

Table 2. Facebook dataset include attributes

#	Attribute	Nominal	Numerical
x1	FirstUser	-	
x2	SecondUser	-	
x3	Freq_Message	-	
x4	Freq_Post	-	
x5	Freq_Business	-	
x6	Freq_Application	-	

5.2 Comparison on Facebook dataset

Although SVM has achieved great success in many areas, such as handwriting recognition [13], text classification [26] and image retrieval [27], when handling the Facebook classification problems, its performance drops significantly. In this section, we will show the performance of our proposed AdaBoostSVM on the Facebook classification problems and compare it with other state-of-the-art algorithms specifically designed to solve these problems.

5.2.1 Review of current algorithms dealing with imbalanced problems

In the case of binary classification, imbalanced classification means that the number of negative instances is much larger than that of positive ones, or vice versa, such as imbalanced document categorization [28], imbalanced clustering for microarray data [29], and detecting credit card fraud [30]. A common method to handle imbalanced problems is to rebalance them artificially by under-sampling [31] (Ignoring instances from the majority class) or over-sampling [32] (Replicating instances from the minority class) or combination of both under-sampling and over-sampling [33]. The popular type of algorithms focuses on biasing the SVM to deal with the imbalanced problems. Several different ways are used. The different penalty constants are used for different classes to control the balance between false positive instances and false negative instances [34]. Cristianini et al. [35] use kernel alignment to adjust the kernel matrix to fit the training samples. Wu and Chang [36] realize this by kernel boundary alignment. In Guo and Viktor [37], Boosting combined with data generation is used to solve imbalanced problems. Another similar work (Yan et al. [38]) uses SVM ensemble to predict rare classes in scene classification.

5.2.2 Generalization performance on Facebook dataset

Firstly, we compare our AdaBoostSVM with the standard SVM [13] on the Facebook dataset. In table 1 and 2, the general characteristics of this dataset is given including the number of attributes, the number of positive instances and the number of negative instances. The commonly used sensitivity and specificity are taken to measure the performance of each algorithm on the imbalanced dataset. They are defined as

$$\text{Sensitivity} = \#\text{true\_positive} / (\#\text{true\_positive} + \#\text{false\_negative'}) \tag{19}$$

$$\text{Specificity} = \#\text{true\_negative} / (\#\text{true\_negative} + \#\text{false\_positive'}) \tag{20}$$

Several researchers (Kubat and Matwin [31], Wu and Chang [36], Akbani et al. [39]) have used the g-means metric to evaluate the algorithm performance on imbalanced problems because g-means metric combines both the sensitivity and specificity by taking their geometric mean. Based on sensitivity and specificity, the g-means metric in Kubat and Matwin [31] is calculated to evaluate these two algorithms on the imbalanced dataset. It is defined as follows:

$$g = \surd\ (\text{sensitivity} * \text{specificity}) \tag{21}$$

The g-means metric values of the two algorithms on the Facebook imbalanced dataset are shown in Table 3. From this table, it can be found that proposed AdaBoostSVM performs best between the two algorithms in general. The success of the proposed algorithm lies in its Boosting mechanism forcing part of PolynomialSVM component classifiers to focus on the misclassified instances in the minority class, which can prevent the minority class from being wrongly recognized as a noise of the majority class and classified into it. Hence, the AdaBoostSVM achieves better generalization performance on the Facebook dataset. Note that the g-means metric of SVM and AdaSVM on Facebook dataset with C equals 25, 55, 80 and number of iterations equals 20, 25 is 0, respectively. These are because SVM and AdaSVM predicted all the instances into the majority class.

Table 3. G-means metric results on the Facebook imbalanced dataset

Classifier	C	#Iterations	d*	g-means
SVM	10	-	1.0	0.885
	25	-	1.0	0
	40	-	1.0	0.989
	55	-	1.0	0
	80	-	1.0	0
AdaSVM	-	5	1.0	0.956
	-	10	1.0	0.997
	-	15	1.0	0.993
	-	20	1.0	0
	-	25	1.0	0

* d: Degree of Polynomial

5.3 Experiments on Facebook dataset

We try to classify the Facebook dataset by different algorithms such as: SVM and AdaBoost in order to find the high percentage of classification accuracy. Firstly, we deployed SMO (Sequential Minimal Optimization) as a kind of SVM algorithms with non-linear dataset and obtained associated results as shown in Table 4. It could correctly classify 98.5556 percent of all samples so from entire samples (2700) of the training set 2661 samples correctly classified and 39 samples incorrectly classified. The number of support vectors and the time is taken for building model according to train data is 104 and 0.28 seconds, respectively. Here we have two classes, and therefore our confusion matrix is 2×2. The number of correctly classified samples is the sum of diagonals in the matrix; all others are incorrectly classified (class “a” gets misclassified as “b” exactly zero samples and class “b” gets misclassified as “a” is 39 samples). The True Positive (TP)

rate is equivalent to Recall. In the confusion matrix, this is the diagonal element divided by the sum over the relevant row, for class "a" equivalent 2661/ (2661+0) = 1 and for class "b" equivalent 0/ (0+39) = 0. The False Positive (FP) rate in the matrix is the column sum of class a minus the diagonal element, divided by the row sums of the class b. Thus, for class "a" equivalent 39/ (39+0) = 1 and for class "b" equivalent 0/ (0+2661) = 0. The Precision in the matrix is the diagonal element divided by the sum over the relevant column. Therefore, the Precision of class "a" is 2661/ (2661+39) = 0.98 and for class "b" is 0/ (0+0) = 0.

Table 4. SVM by Using Different Value of C

Classifier	C	d	Accuracy (%)	#Incorrectly Classify Instance	#Correctly Classify Instance	#Support Vectors	Computation Time (s)
SVM	10	1.0	98.5556	39	2661	104	0.28
	25	1.0	98.5556	39	2661	121	0.29
	40	1.0	98.5556	39	2661	136	0.30
	55	1.0	98.5556	39	2661	150	0.32
	80	1.0	98.5556	39	2661	192	0.38
	100	1.0	98.5556	39	2661	213	0.41
	110	1.0	98.5556	39	2661	218	0.42

Table 4 represents the SVM by using different value of C in order to find high accuracy of classifying. Based on this table, the SVM with C equal ten (C=10) is better than other values of C for classifying the Facebook dataset. The correctly classification samples with constant C=10, is 2661 and the percentage of accuracy is 98.5556. Besides this, the number of support vectors is 104, as well as the time taken to build a model is 0.28 seconds.

Secondly, we try to find high percentage of classification accuracy by using AdaSVM method with different value of Number Iteration. AdaSVM consists of AdaBoost, with PolynomialSVM as a learner. Table 5 represents the AdaSVM method by using different numbers of iterations. Therefore, by considering Table 5, the best percentage of accuracy is AdaSVM with NumIteration equal 15. The accuracy and time taken to build a model is 98.7037% and 0.83 seconds, respectively.

Table 5. AdaSVM Classification on Facebook Dataset

Classifier	#Iterations	d	Accuracy (%)	#Incorrectly Classify Instance	#Correctly Classify Instance	Computation Time (s)
AdaSVM	5	1.0	98.6296	37	2663	0.61
	10	1.0	98.6667	36	2664	0.76
	15	1.0	98.7037	35	2665	0.83
	20	1.0	98.7037	35	2665	0.98
	25	1.0	98.7037	35	2665	1.08

Then, the result of two methods represents the effectiveness and robustness of AdaSVM by number iteration equal 15 is better than SVM by Polynomial kernel function on Facebook dataset (see Table 6). Besides this, Figure 6 represents detail of AdaSVM result, which shows the dispersion of samples as a Cybercrimes or Non-Cybercrimes on Facebook dataset. The samples near the conjunction of axis X and Y are Non-Cybercrimes and the samples is putting the end of axis X are Cybercrimes. The performance of AdaSVM depends to a great extent on the choice of classifier function and number of iterations in order to transform data from an input space to a higher dimensional feature space. Also, Figure 7 represents the threshold curve with class's value equals one. The axis X shows the false positive rate number and the axis Y shows the

number of true positive rate. This curve represents the process of increasing Non-Cybercrime samples in FP (False Positive Rate) and TP (True Positive Rate). Finally, Figure 8 represents threshold curve with class value is zero for AdaSVM and the number of iterations is 15.This curve shows the Cybercrime samples between two axes X and Y as a false positive rate (FP) and true positive rate (TP), respectively.

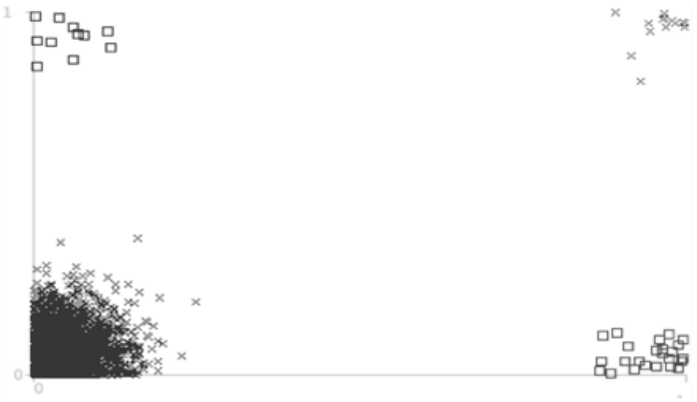

Figure 6. AdaSVM Dispersion of Classes

Table 6. Average Classification Accuracy Result of SVM and AdaSVM

Classifier	Accuracy (%)	#Incorrectly Classify Instance	#Correctly Classify Instance	Computation Time (s)
SVM	98.5556	39	2661	0.28
AdaSVM	98.7037	35	2665	0.83

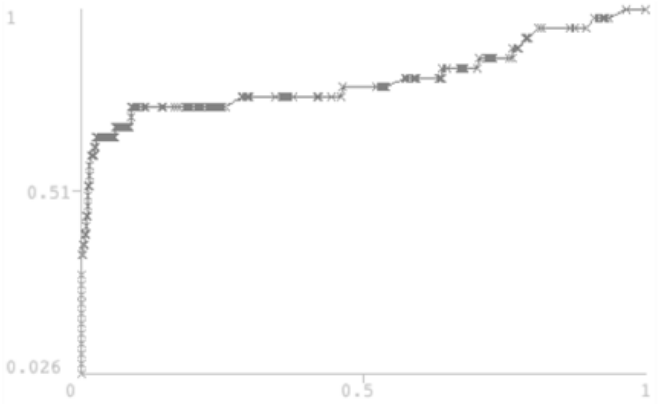

Figure 7. AdaSVM Threshold Curve with Classes Value =1.

X: False Positive Rate (Num) and Y: True Positive Rate (Num)

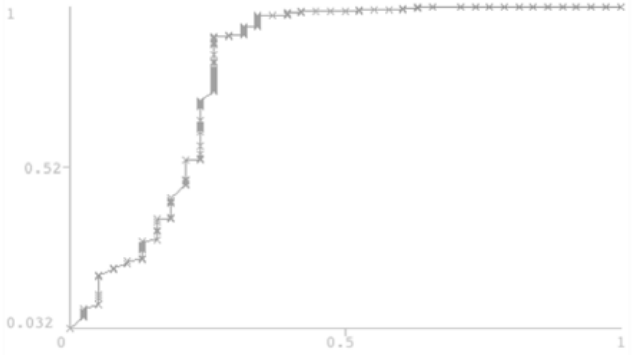

Figure 8. AdaSVM Threshold Curve with Classes Value =0.

X: False Positive Rate (Num) and Y: True Positive Rate (Num)

6 Conclusion & future works

Experimental results on Facebook dataset demonstrate that proposed AdaBoostSVM (AdaSVM) performs better than other approaches of using component classifiers such as: PolynomialSVM. Besides these, it is found that AdaBoostSVM demonstrates good performance on imbalanced classification problems, as well as improved version is further developed to deal with the accuracy/diversity dilemma in Boosting algorithms, giving rise to better generalization performance. Based on this result, we try to propose a new cybercrime detection and prevention model by using AdaSVM method and the Girvan-Newman algorithm to decrease community detection problem on the social network (Facebook dataset) in future works.

References

[1] V.N.Vapnik. The Nature of Statistical Learning Theory. 1995.
[2] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. 1998. http://dx.doi.org/10.1023/A:1009715923555
[3] Chan A K, XU P. Support vector machine for multi-class signal classification with unbalanced samples. International Joint Conference on Neural Networks; 2003: 1116-1119.
[4] Vapnik V N. The Nature of Statistical Learning Theory. New York: Springer; 2000.
[5] V Vapnik, B Boser, I Guyon. A training algorithm for optimal margin classifiers. Conference on Computational Learning Theory 15th. 1992: 144-152.
[6] Michael Kearns. Thoughts on hypothesis boosting. Unpublished manuscript. 1988.
[7] Rob Schapire. Strength of Weak Learnability. Machine Learning. 1990; 5: 197-227. http://dx.doi.org/10.1023/A:1022648800760
[8] Yoav Freund, Robert E. Schapire. Game theory, on-line prediction and boosting. In Proceedings of the Ninth Annual Conference on Computational Learning Theory. 1996: 325-332. http://dx.doi.org/10.1145/238061.238163
[9] Tzu-Yen, Chin-Hsiung, Chu-Cheng. Virus Prevention Model Based on Static Analysis and Data Mining Methods. 2009.
[10] Genton Marc G. Classes of Kernels for Machine Learning: A Statistics Perspective. 2001.
[11] Taiwo Oladipupo Ayodele. Type of Machine Learning Algorithm. 2010.
[12] César Souza. Kernel Function for Machine Learning Application. 2010.
[13] Vapnik V. Statistical Learning Theory. New York: John Wiley; 1998.
[14] Platt C J. Fast training of support vector machines using sequential minimal optimization Source. In Advances in kernel method: support vector learning. 1999: 185-208.
[15] Abe S. Support Vector Machines for Pattern Classification. London: Springer-Verlag; 2006.

[16] Nello Cristianini N, Shawe Taylor, J. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. 2000.
[17] Genton Marc G. Classes of Kernels for Machine Learning: A Statistics Perspective. 2001.
[18] Chen Lin. Working set selection using second order information for training SVM. 2005.
[19] Hsu C W, Chang C C, Lin C J. A Practical Guide to Support Vector Classification. 2010.
[20] Christopher J C Burges. A Tutorial on Support Vector Machines for Pattern Recognition. 1998.
[21] Klaus-Robert Müller, and et al. An Introduction to Kernel-Based Learning Algorithms. 2001.
[22] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences. 1997; 55(1): 119-139. http://dx.doi.org/10.1006/jcss.1997.1504
[23] Freund Y, Schapire R E, A Short Introduction to Boosting Introduction to AdaBoost, Journal of Japanese Society for Artificial Intelligence. 1999; 14(5): 771-780.
[24] SVM and Boosting: One Class. Gunnar Ratsch, Bernhard Scholkopf, Sebastian Mika, Klaus-Robert Muller. November 9, 2000.
[25] T G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning. 2000; 40(2): 139-157. http://dx.doi.org/10.1023/A:1007607513941
[26] Joachims. Text categorization with support vector machines: learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning. 1998: 137-142.
[27] Tong S, Koller D. Support vector machine active learning with applications to text classification. Journal of Machine Learning Research. 2001; 2: 45-66.
[28] Del Castillo M D, Serrano J I. A multistrategy approach for digital text categorization from imbalanced documents. In: ACM SIGKDD Explorations: Special Issue on Learning from Imbalanced Datasets. 2004: 39-70.
[29] Pearson R, Goney G, Shwaber J. Imbalanced clustering for microarray time-series. In: ICML'2003 Workshop on Learning from Imbalanced Data Sets (II). 2003.
[30] Fawcett T, Provost F. Adaptive fraud detection. Data Mining and Knowledge Discovery. 1997; 1(3): 291-316. http://dx.doi.org/10.1023/A:1009700419189
[31] Kubat M, Matwin S. Addressing the curse of imbalanced training sets: one-sided selection. In: Proceedings of the 14th International Conference on Machine Learning. 1997: 179-186.
[32] Chawla N V, Bowyer K W, Hall L O, Kegelmeyer W P. Smote: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research. 2002; 16: 321-357.
[33] Ling C X, Li C. Data mining for direct marketing problems and solutions. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining. 1998.
[34] Veropoulos K, Campbell C, Cristianini N. Controlling the sensitivity of support vector machines. In: Proceedings of the International Joint Conference on Artificial Intelligent. 1999: 55-60.
[35] Cristianini N, Shawe-Taylor J, Elisseeff A, Kandola J S. On kernel-target alignment. In: Advances in Neural Information Processing Systems. 2001: 367-373.
[36] Wu G, Chang EY. Kba: kernel boundary alignment considering imbalanced data distribution. IEEE Transactions on Knowledge and Data Engineering. 2005; 17 (6): 786-795. http://dx.doi.org/10.1109/TKDE.2005.95
[37] Guo H, Viktor H L. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. In: ACM SIGKDD Explorations: Special Issue on Learning from Imbalanced Datasets. 2004: 30-39.
[38] Yan R, Liu Y, Jin R, Hauptmann A. On predicting rare class with SVM ensemble in scene classification. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal. 2003; April 2003: III-21-4.
[39] Akbani R, Kwek S, Japkowicz N. Applying support vector machines to imbalanced datasets. In: Proceedings of the 15th European Conference on Machine Learning; 2004: 39-50.