Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Selection of Candidate Support Vectors in incremental SVM for network intrusion detection☆

Roshan Chitrakar[*], *Chuanhe Huang*

School of Computer, Wuhan University, Wuhan, Hubei, China

ARTICLE INFO

Article history:
Received 24 September 2013
Received in revised form
25 April 2014
Accepted 10 June 2014
Available online 19 June 2014

Keywords:
Incremental support vector machine
Karush–Kuhn–Tucker condition
Candidate Support Vector
Half-partition strategy
Network intrusion detection

ABSTRACT

In an Incremental Support Vector Machine classification, the data objects labelled as non-support vectors by the previous classification are re-used as training data in the next classification along with new data samples verified by Karush–Kuhn–Tucker (KKT) condition. This paper proposes Half-partition strategy of selecting and retaining non-support vectors of the current increment of classification – named as Candidate Support Vectors (CSV) – which are likely to become support vectors in the next increment of classification. This research work also designs an algorithm named the Candidate Support Vector based Incremental SVM (CSV-ISVM) algorithm that implements the proposed strategy and materializes the whole process of incremental SVM classification. This work also suggests modifications to the previously proposed concentric-ring method and reserved set strategy. Performance of the proposed method is evaluated with experiments and also by comparing it with other ISVM techniques. Experimental results and performance analyses show that the proposed algorithm CSV-ISVM is better than general ISVM classifications for real-time network intrusion detection.

1. Introduction

Network intrusion detection is also considered as a pattern recognition problem of classifying the network traffic patterns into two classes – normal and abnormal; according to the similarity between them. Nowadays, in the field of intrusion detection, Support Vector Machine (SVM) is becoming a popular classification tool based on statistical machine learning (Mohammad et al., 2011). There are two issues in machine learning – training of large-scale data sets and availability of a complete data set (Le and Nguyen, 2011; Du et al., 2009a,b). Computer's memory will not be enough and training time will be too long if training data set is very large. Next, when we capture data packets from a stream of a network, we cannot obtain the complete network information in the very first time and hence a continuous online learning is required for high learning precision with increasing number of samples. The challenge of incremental learning is to decide what and how much information from the previous learning should be selected for training in the

☆ This work is supported by the National Science Foundation of China (No. 61373040, No. 61173137), The Ph.D. Programs Foundation of Ministry of Education of China (20120141110073), Key Project of Natural Science Foundation of Hubei Province (No. 2010CDA004).

* Corresponding author.
E-mail addresses: roshanchi@gmail.com, roshanchi@whu.edu.cn (R. Chitrakar), huangch@whu.edu.cn (C. Huang).

http://dx.doi.org/10.1016/j.cose.2014.06.006
0167-4048/

next learning phase and how to deal with new data sets being added in that phase. So, the key of incremental learning is to cope with increasing data samples while retaining the information of original data samples in the meantime.

Most of the intrusion detection methods use non-incremental learning algorithms. With accumulation of new data samples, their training time increases continuously, and at the same time, they have difficulties in adjusting themselves with changing network environment. On the contrary, incremental learning has ability of rapidly learning from new samples and modifying their original model (Yi et al., 2011). Incremental learning methods can better meet requirements of real-time intrusion detection, and can also improve computational accuracy of real-time applications.

A simple Incremental Support Vector Machine (ISVM) algorithm acquires the support vectors by training the initial sample set. Both new data sets and the previous support vectors are merged to form a new sample set, and train them to produce new support vectors. The process is repeated till the final data set (Makili et al., 2013). Chances of new data samples becoming support vectors are tested using Karush–Kuhn–Tucker theory (Wang et al., 2006). Samples that violate the KKT conditions may change the previous support vector set, and hence they are added to previous data set. Samples that meet KKT conditions are discarded because they don't change the previous support vector set (Karasuyama and Takeuchi, 2010).

This paper proposes an improved and efficient learning approach of Incremental Support Vector Machine based on the idea of retaining the original and current data samples throughout the whole learning process. In this proposed approach, data points, that are not the support vectors in the current increment of classification but have chances of becoming support vectors in the next increment of classification, are selected and retained as the Candidate Support Vectors (CSVs) so that they can be combined with the new training data set that will be added in the next training. This approach also introduces Half-partition strategy as a part of the CSV selection method and devises an algorithm named CSV-ISVM using the strategy.

The rest of the paper is organized as follows. Section 2 presents some of the works related to this paper. In Section 3, CSV based incremental SVM is introduced and background knowledge for the approach (viz. KKT-conditions and SVM hyperplane rotations) is also described. Section 4 proposes modification to the concentric circle method and Section 5 explains the main work of this paper i.e. the Half-partition method. All the experiments of this research are illustrated in Section 6 along with necessary analyses. At last, Section 7 draws conclusion of the paper and suggests further work too.

2. Related work

Many research works have been done regarding incremental learning methods using SVM classification and, of course, various modifications have also been proposed. Most of the works seem to have suggested improvements on detection performance, accuracy etc.

A basic incremental SVM based approach called Incremental Batch Learning with SVM was suggested by Liu et al. (2004), in which only the support vectors were preserved for the next increment, while all other data samples were discarded. In fact, the discarded samples carry some amount of information about classification. With the addition of new data samples in the following increments of the learning process, these data samples may become support vectors or vice-versa. Therefore, in simple incremental SVM like this, the classification accuracy is seriously affected in the later increments.

WenJian and Wang explained that data samples lying near the hyperplane had greater possibilities of becoming support vectors after adding new training set (Wang, 2008). So, they proposed the Redundant Incremental Learning Algorithm that retained the redundant samples lying near the hyperplane and added them to the new data sets in the next increment to check if they become the support vectors.

Preserving support vectors in each increment of SVM classification increases the overall classification time. So, execution time reduction becomes primarily important. A typical solution to accelerate SVM training is to decompose the quadratic programming (QP) into a number of sub-problems so that the overall SVM training complexity can be reduced from $O(N^3)$ to $O(N^2)$ (Joachims, 1998; Platt, 1998). However, when the number of data points N is very large, the time complexity is still unsatisfactory and the training needs further improvement (Zhang et al., 2009).

A number of research works have also made contributions in decreasing execution time of incremental SVM classification. An incremental learning approach with SVM proposed by Yao et al. (2012) in classification of network data stream, an incremental learning method proposed by Sun and Guo (2012) in multi-model structure design, a fast incremental learning algorithm for SVM named the Active Set Iteration method (Tao, 2006) etc. are a few among such works.

Again, an incremental learning algorithm of SVM based on clustering was proposed by Du et al. (2009a,b) considering the fact that the boundary support vectors may change into support vectors after adding new samples. This work also used KKT theory in the learning process. The cluster centres obtained from clustering process were checked for the KKT conditions and the cluster centres violating the conditions were added to the support vector set for further training. The necessary and sufficient conditions violating the KKT were given by Wang, Zheng, Wu, and Zhang (Wang et al., 2006). They presented and proved that if there were any new samples contrary to the KKT conditions, then the non-support vectors of the original SVM would have the chance to become support vectors.

An incremental learning algorithm based on the Karush–Kuhn–Tucker (KKT) conditions was also proposed in the literature (Wen-hua & Jian, 2001). In this method, the whole learning process was divided into the initial process and the incremental processes. The optimal solution of the QP allowed each sample to satisfy the KKT conditions.

Though KKT based incremental SVM is better in detection rates, it needs longer training time because it has to do cross-judging and more training compared to simple incremental SVM. This was mentioned in the paper entitled Incremental SVM based on Reserved Set proposed by Yi et al. (2011). In order to further reduce the training time, they suggested a Concentric Circle method to be used in selecting samples to form a Reserved Set of support vectors that would be used to train in the next increment along with new data sets. It was explained in their work that the data samples lying in the ring region of two concentric circles had highest probabilities of becoming support vectors, and hence, were retained as the Reserved Set.

Besides retaining and preserving support vectors, there have been efforts of reducing support vector as well. In order to reduce the computational cost, Kobayashi and Otsu (2009) proposed a method to sequentially eliminate the redundant support vectors. Similarly, Habib et al. (2009) proposed various Support Vector Set Reduction algorithms in order to accelerate the classification process for a generic SVM-based Change Detection algorithm.

This paper proposes (1) an improvement to the Concentric Circle method, and (2) the Half-partition method of selecting support vectors in increment SVM classification, which mainly reduces classification time along with high detection and low false alarm rates.

3. Candidate Support Vectors based incremental SVM

The key of the proposed Incremental Support Vector Machine is the selection of Candidate Support Vectors. Here, two methods of selecting CSVs are presented: (1) Improved Concentric Circle method; and (2) Half-partition strategy.

Since the latter method is proven to be better, this work designs the CSV Selection algorithm using the Half-partition method. New data samples used for training in each increment are verified if they can become support vectors by using the decision function suggested by KKT-theory.

3.1. KKT conditions for ISVM

In SVM classification, a decision function is obtained by solving a quadratic programme (QP) (Wang et al., 2006). To get an optimal solution, the following Karush–Kuhn–Tucker (KKT) conditions must be satisfied:

$$\begin{cases} \alpha_i = 0 \Rightarrow & f(x_i) > 1 \text{ or } f(x_i) < -1 \\ 0 < \alpha_i < C \Rightarrow & f(x_i) = 1 \text{ or } f(x_i) = -1 \\ \alpha_i = C \Rightarrow & -1 < f(x_i) < 1 \end{cases} \quad (1)$$

where α_i is the Lagrange multiplier corresponding to the samples, and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots)$ is the optimal solution, if and only if every sample x_i meets these conditions.

Here, $f(x) = 0$ is the optimum separating hyperplane, $f(x) = \pm 1$ are the boundaries of the separating margin. Therefore, for a training sample x_i, if $\alpha_i = 0$, then it lies outside the boundaries; if $0 < \alpha_i < C$, then it lies on either of the boundaries; and if $\alpha_i = C$, then it lies inside the boundaries of the separating margin.

3.2. Rotation of the SV hyperplane

The idea of Half-partition method is arisen from the phenomenon of SV hyperplane rotations. So, the possible geometric rotations of the SV hyperplane are illustrated here. As seen from Fig. 1, the hyperplane can rotate either to clockwise or to anti-clockwise direction with respect to the current position of the hyperplane. No change in the rotation implies that the new samples satisfy the KKT conditions. To list the possible SV hyperplane rotations, we have the following:

(1) Towards the anti-clockwise direction,
(2) Towards the clockwise direction,
(3) No rotation

In Fig. 1, solid circles and solid squares represent the original samples, while the circles enclosed by hollow squares represent new incremental samples. The original hyperplane is $f(x) = 0$. Samples S_1 and S_2 are support vectors. Say, new samples i and j are introduced for incremental SVM learning purpose. Since they violate the KKT conditions, they lead to the rotations of the optimal hyperplane in clockwise and anti-clockwise directions respectively in the following iterations of the SVM learning process. Then, the non-support vectors {b, c, j} and {e, i} change to the support vectors.

If we consider new samples and original support vectors only, and discard original non-support vectors, some valuable information will be lost, which may result in obtaining a bad classifier.

The later learning increments will lead to oscillation if the initial samples are not enough. However, if all of non-support vectors are included in the learning process, then there will be more unwanted data instead of the real Candidate Support Vectors. Meanwhile, with new samples coming in continuously, the size of the training set eventually becomes too large. Therefore, a method to select CSVs is explained here to make ISVM more efficient.

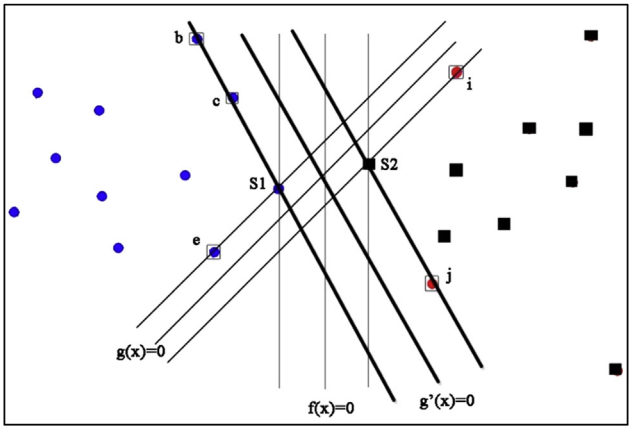

Fig. 1 – Hyperplane rotations due to new sample sets {*b*, *c*, *j*} & {*e*, *i*}.

4. The Improved Concentric Circle method

The decision function of SVM is determined by the support vectors. To determine which samples are most likely to be support vectors and thus to make the CSV set, this paper proposes Improved Concentric Circle method, which actually is a modification to the Concentric Circle method proposed by Yi et al. (2011). It explained that most of the marginal samples lie in the ring region between the two concentric circles, as shown in Fig. 2, and therefore, the Concentric Circle method retains the samples lying in the ring region. However, the process of constructing the ring i.e. determining radii R_i and R_o of the inner and the outer circle respectively needs separate experiments that keep on changing with the data samples. So, the proposed Improved Concentric Circle method suggests a fixed way of determining R_i and R_o.

Firstly, the centres of all classes are calculated as follows:

$$m_{ij} = \frac{1}{n_i}\sum_{k=1}^{n_i} x_{kj} \tag{2}$$

where $j = 1, 2, \ldots, d$; d is the dimension of the mean vector. n_i is the number of samples belonging to class i in sample set, and x_{kj} is the jth attribute of sample k. $i = 1, 2$, stands for the sample class.

Secondly, calculate the Euclidean distance R between two class centres. For the binary classification problem, it can be denoted as:

$$R = r(m_1, m_2) = \sqrt{\left(\sum_{k=1}^{d} |m_{1k} - m_{2k}|\right)} \tag{3}$$

where m_1 and m_2 are mean vectors, m_{1k} and m_{2k} are the kth dimensions of the mean vectors. The Euclidean distance between two samples is calculated by the following equation:

$$d_{ij} = \sqrt{\left(\sum_{k=1}^{m} |x_{ik} - x_{jk}|^2\right)} \tag{4}$$

4.1. Selection of radii R_i and R_o

Selection of R_i and R_o is vital and sensitive too. The method should be such that data points lying inside the inner circle have the least probabilities of being support vectors and those lying outside the outer circle do not have any. So, we take R_i and R_o as following measurements:

R_i = the distance between the class centre C and the nearest support vector S_1

R_o = the distance between the class centre C and the nearest point at the hyperplane h.

Here, R_i can be calculated using Equation (4). We know that the mid-point of two class centres, say H, lies on the hyperplane. If R is the distance between two class centres, then R is given by Equation (3). Once we know R, R_o may be calculated as $R_o = R/2$. But this paper recommends an alternate method of calculation as given in the following section.

4.2. Calculation of r

Suppose, r is the Euclidean distance from a point $\vec{x}$ to the decision boundary, as shown in Fig. 3, say at the point x'. We know that the shortest distance between a data point and a hyperplane is perpendicular to the plane (Manning et al., 2008), and hence, parallel to $\vec{w}$. A unit vector in this direction is $r\vec{w}/\vec{w}$. The dotted line in the diagram is then a translation of the vector $r\vec{w}/\vec{w}$. Let us label the point on the hyperplane closest to $\vec{x}$ as $\vec{x}'$. Then:

$$\vec{x}' = \vec{x} - yr\frac{\vec{w}}{|\vec{w}|} \tag{5}$$

where multiplying by y just changes the sign for the two cases of $\vec{x}$ being on either side of the decision surface. Moreover, $\vec{x}$ lies on the decision boundary and it satisfies $\vec{w}^T\vec{x}' + b = 0$. Hence:

$$\vec{w}^T\left(\vec{x} - yr\frac{\vec{w}}{|\vec{w}|}\right) \tag{6}$$

Solving for r gives:

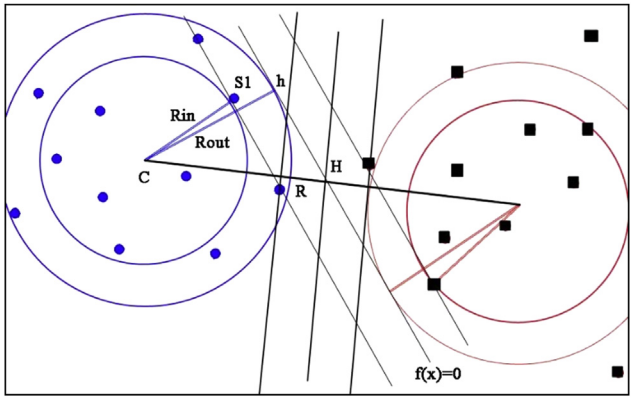

Fig. 2 – The Improved Concentric Circle method for selecting CSVs.

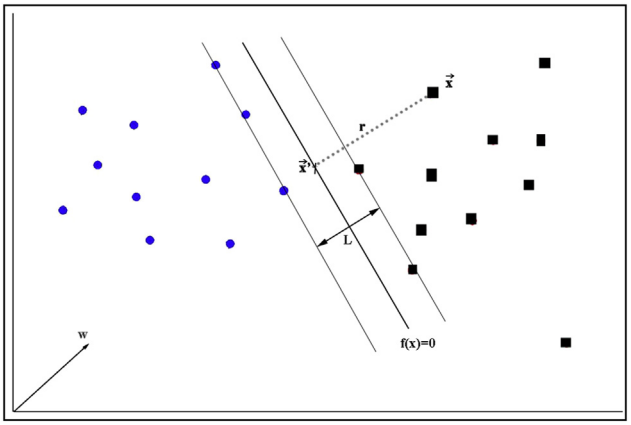

Fig. 3 – Distance (r) of a data point (x) from the hyperplane.

$$r = y\frac{\vec{w}^T\vec{x} + b}{|\vec{w}|} \quad (7)$$

The advantage of this calculation method is that we do not need to calculate distance between two class centres, R, which makes the calculation dependent only on the current class.

5. The Half-partition strategy

In Fig. 2, there are data points even outside the ring-region. These data points are not being considered as the Candidate Support Vector by the Concentric Circle methods. Unless we prove they are the real outliers (i.e. do not belong to the same class) or do not contribute in classification process, they cannot be excluded from the CSV set. So, this research work considers removing the outer circle (then the ring no longer exists as shown in Fig. 4) so as to maximize the CSV set. By so doing, all non-support vectors lying outside the inner circle (having radius R_i) and also the outliers belong to the greater CSV set, thus providing them the equal opportunity to become a CSV.

Furthermore, looking at the possible rotations of the SV hyperplanes in Fig. 1, it is seen that the outer half of the entire data space covered by each class is not touched by the rotated hyperplane i.e. $g(x) = 0$ and $g'(x) = 0$ respectively, which means that outer halves of the classes do not contain data points that could be selected as CSVs for the next increment. And, this situation is very true in case of two-class classification. For a binary class network intrusion detection problem, the outer-partitions of the both classes can be avoided for CSV selection. Hence, this paper proposes the Half-partition method as illustrated in Fig. 4, which considers data points lying only at the inner half partitions of two classes.

If a non-support vector, say a data point x in Fig. 4, satisfies the following condition, then it will be considered as a Candidate Support Vector:

$$d \geq R_i \quad \text{AND} \quad r \leq R_o \quad (8)$$

where d is the distance between x and the class centre C, and r is the distance between x and the hyperplane.

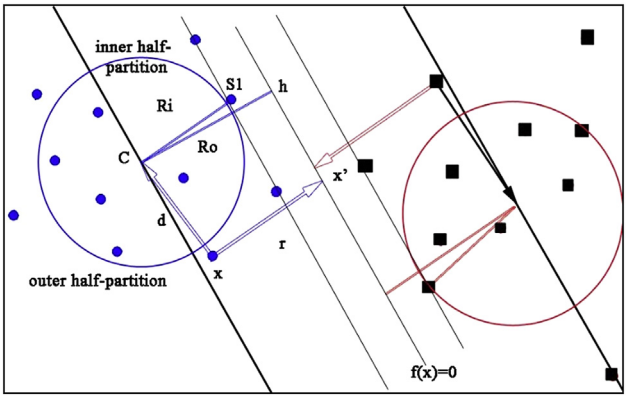

Fig. 4 – The Half-partition strategy and weight calculation of CSVs.

5.1. Calculation of weights of CSV

Probability of being a CSV depends on two parameters:

(1) d – the Euclidean distance between Candidate Support Vector, x in Fig. 3, and the class centre C; and
(2) r – the distance between the Candidate Support Vector x and the SVM hyperplane, x'. Here, d and r can be calculated using Equations (4) and (7) respectively.

The data samples which lie farther from the hyperplane has less probability of becoming support vectors. Similarly, the data points lying inside the circle also have less probability of being support vectors. So, data points lying outside the circle and nearer to the hyperplane have maximum probability.

The greater the value of d, the higher x has probability to be a support vector; and, the smaller the value of r, the higher the probability again. Therefore, $d \geq R_i$ and $r \leq L$ are the two criteria for formulating the equation for weight calculation. Here, L is the distance between the support vector S_1 and the hyperplane, which is given by $1/\vec{w}$ (We know that $2/\vec{w}$ is the margin of the decision boundary).

For weight calculation, the following expression is proposed:

$$W_C = \{d/(d + R_i) + L/(L + r)\} \quad (9)$$

where, W_C is the total weighted distance of the Candidate Support Vector. For the Support Vector nearest from the class centre C, $d = R_i$ and $r = L$, therefore, $d/(R_i + d) = 0.5$ and $L/(L + r) = 0.5$ and $W_C = 1$. So, all the non-support vectors which yield the value of W_C close to 1 (one) are selected as CSVs.

5.2. Threshold assignment

From the previous section, we learnt that threshold T might be defined as $W_C - 1$ such that $T = 0$ for a typical support vector. Now, the equation for the threshold T may be expressed as

$$T = W_C - 1 = \{d/(d + R_i) + L/(L + r)\} - 1 \quad (10)$$

Depending upon the distribution of data points and nature of occurring new data samples, the threshold T may be set variably. For instance, we can take $T = \pm 0.25$ could be a decision factor to consider a data point that yields the weighted value W_C either equal to 1.25 or 0.75.

5.3. The CSV selection and the CSV-ISVM algorithms

The first-ever CSV Selection algorithm triggers as soon as the first SVM learning is finished. Using the CSV Selection algorithm, described in Algorithm 1, CSVs are selected and retained for the next iteration of the SVM classification. During the CSV selection process, the weights of CSVs are also preserved along with the CSV sets. These CSV weights may also be determined by a threshold value.

ALGORITHM 1. CSV Selection

Input: Sample set X_0
Output: The CSV set
//Calculate the mean vector of each class//
For every sample x_i (i =1,2, ..., l) in X_0
  If x_i belongs to the normal class
    For k =1, ..., d
      $m_1(k)= m_1(k)+ x_i(k)$;
    EndFor
    $n_1 = n_1 +1$;
  Else //x_i belongs to attack class//
    For k =1, ..., d
      $m_2(k)= m_2(k)+ x_i(k)$;
    EndFor
    $n_2 = n_2 +1$;
  EndIf
EndFor
For k =1, ..., d
  $m_1(k)= m_1(k)/n_1$;
  $m_2(k)= m_2(k)/n_2$;
EndFor
Calculate the radius R_i using Equation 3;
//Select samples to construct the CSV set//
For every sample x_i (i =1,2, ..., l)
  Calculate the distance d from x_i to the mean vector of the class that x_i belongs to;
  Calculate the distance r from x_i to the hyperplane at h;
  If $d \geq R_i$ and $r \leq R_o$
    Add x_i into the CSV set;
  EndIf
EndFor

Every time a new incremental sample appears, simple incremental SVM checks whether all the samples meet the KKT conditions. If there is none, it will add these samples to the support vectors to form a new training set. Meanwhile, samples from the CSV set are selected according to their weights, and combined them with the samples from the new sample set that violate the KKT conditions; and finally added them to original support vectors to form a new training set.

ALGORITHM 2. CSV-ISVM

Input: Sample set X_0, New incremental sample set X_1
Output: Updated CSV set and the classifier SVM
Get classifier SVM_0 and support vector set SV_0 by training on X_0;
Calculate class centres X_{0+} and X_{0-} using Equation 2;
Compute the distance between samples and class centres using Equation 4;
Construct CSV set N_0 by selecting samples that satisfy Equation 8;
Get weights Θ_i and thresholds T_i using Equation 9 and Equation 10 respectively;
Take new sample set X_1;
For every sample x_i in X_1
  If x_i violates the KKT conditions of SVM_0;
    Add x_i to X_{1s};
  Else
    Add x_i to X_{1d};
  EndIf
EndFor
If X_{1s} is not null //Select samples according to their thresholds//
  For every sample x_i in X_{1s}
    If $T \geq$ given threshold value
      Add sample x_i to the set NSV_0;
    EndIf
  EndFor
  Set $N_1 = SV_0 \cup NSV_0 \cup X_{1s}$;
  Select samples in N_1 to construct N_1';
  Obtain classifier SVM_1 *and* SV_1 by training on X_0;
  Set SVM_1 as the output;
  Update the CSV set and also weights Θ_i and thresholds T_i;
Else
  Set SVM_0 as the output;
EndIf

After incremental learning, the CSV set and the weights of the samples in it are updated. The entire process of incremental learning is described in Algorithm 2.

6. The experiment

6.1. Experimental data

Mostly two benchmark data sets are used for performance evaluation of any experiment related to intrusion detection. They are KDD Cup 1999 data set (UCI, 1999) and Kyoto 2006+ data set (Kyoto+, 2009). In this work, Kyoto 2006+ data set is used as data sets for the experiments. After examination of the data – specially the attack ratio – and preliminary experiments, the data sets of the days 2007 Nov. 1, 2 and 3 are chosen for the final experiments.

6.2. Data pre-processing

Data pre-processing techniques like sampling and filtering are applied for easy and smooth operation of the experiments. As suggested by Chitrakar and Huang (2012a,b), data samples are extracted such that each set contains 1% of attacks that includes unknown attacks too. Attributes of the data sets are converted to appropriate types and normalized according to the need of the SVM kernel. Samples with both known and unknown attacks are treated as a single attack class and labelled as −1. Thus, the entire selected data can be categorized into {−1, 1} where 1 represents normal class. This makes almost any type of SVM classification able to use the data and the classification process easier and simpler.

6.3. The experimental detail

The experimental data thus obtained from Kyoto 2006+ data set is randomly divided into two non-overlapping subsets – training and test set. Then, 10 (ten) separate data sets from the training subset are extracted as the incremental training sets, each set containing 1000 samples having both normal and abnormal data. Similarly, 10 (ten) data sets are extracted from the test data set too.

Using each data sample, experiments are carried out for Simple-ISVM, KKT-ISVM, RS-ISVM and CSV-ISVM one by one. In all four methods, the RBF kernel is used because SVM produces better performance with RBF (Bhavsar et al., 2013).

The number of true positive, true negative, false positive and false negative values of all four methods are recorded and used for performance evaluation.

6.4. Performance evaluation and analysis

The performance evaluation of the experiment is carried out in terms of accuracy (A), detection rate (DR) and false alarm rate (FAR) by using the following equations:

$$A = (TP + TN)/(TP + TN + FP + FN) \quad (11)$$

$$DR = (TP)/(TP + FP) \quad (12)$$

$$FAR = (FP)/(FP + TN) \quad (13)$$

where, TP, TN, FP and FN stand for True Positive (attack detected as attack) and True Negative (normal detected as normal), False Positive (normal detected as attack) and False Negative (attack detected as normal) respectively.

Evaluation of CSV-ISVM is done by comparing it with Simple-ISVM, the KKT-ISVM and RS-ISVM, on detection rate and false alarm rate. Evaluation has also been done for the training and testing time of all four methods.

Table 1 shows that the values of DRs and FARs in case of Simple-ISVM increase in one increment whereas decrease in another increment, e.g. the DR has increased to 86.826% in the 2nd iteration but decreased to 81.435% in the 3rd iteration; and it is similar in case of FAR too. This implies that they depend heavily upon new data sets of each iteration. KKT-ISVM produces higher DR values (and lower FAR values) in the last couple of increments compared to the initial few increments, but produces increasing and decreasing values over middle increments, e.g. in iterations 5, 6 and 7, DRs are 81.941%, 80.69% and 86.518% respectively and FARs are 4.518%, 5.153% and 4.327% respectively. It is seen from Figs. 5 and 6 that KKT-ISVM compared to Simple-ISVM has a better growing trend of DR and a better falling trend of FAR, in general, but still has inconsistencies in some increments.

Again from Table 1, it is seen that both RS-ISVM and CSV-ISVM produce continuously increasing DRs and decreasing FARs and also have better rates compared to Simple-ISVM and

Table 1 – Comparison of DR and FAR.

Iteration count	Simple-ISVM		KKT-ISVM		RS-ISVM		CSV-ISVM	
	DR (%)	FAR (%)	DR (%)	FAR (%)	DR (%)	FAR (%)	DR (%)	FAR (%)
1	79.646	4.535	79.646	4.535	79.646	4.535	79.646	4.535
2	86.826	7.245	77.451	4.449	84.113	4.099	84.724	4.025
3	81.435	6.169	82.608	5.494	85.069	4.059	85.576	3.915
4	76.871	6.076	81.829	5.501	85.917	3.948	86.082	3.802
5	77.513	6.393	81.941	4.518	86.407	3.805	86.814	3.54
6	77.474	6.403	80.69	5.153	87.352	3.476	87.238	3.33
7	75.618	6.552	86.518	4.327	87.54	3.372	88.082	3.027
8	71.882	3.216	84.221	4.141	87.968	3.225	88.095	3.012
9	80.606	6.762	87.419	4.162	89.144	3.124	89.643	2.52
10	79.576	5.699	87.706	4.16	89.817	3.015	90.147	2.314

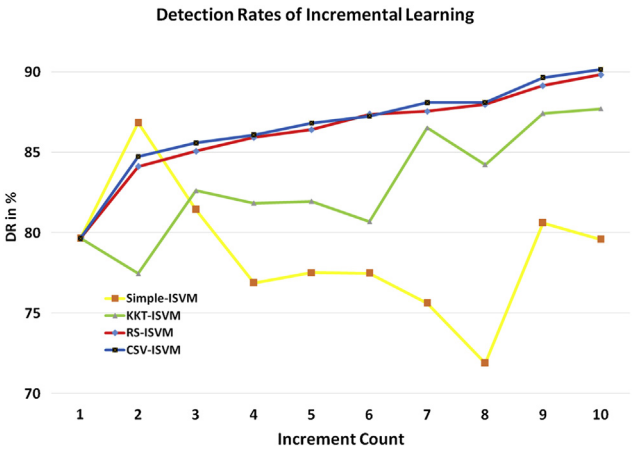

Fig. 5 – Comparison of DR.

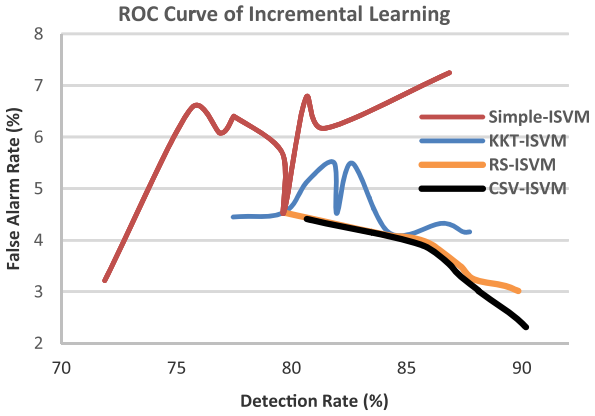

Fig. 7 – Comparison of ROC curves.

KKT-ISVM. The very first iteration has the same result for all four methods because SVs are trained along with new data sets only from the second iteration onwards; and the final result is obtained only in the 10th iteration. Here, RS-ISVM has 89.817% and 3.015% of final DR and FAR respectively, whereas CSV-ISVM shows 90.147% and 2.314% of final DR and final FAR respectively.

As seen from Figs. 5 and 6, both RS-ISVM and CSV-ISVM show growing DR lines and falling FAR lines as increments go on. Moreover, comparing RS-ISVM and CSV-ISVM methods, DR and FAR of CSV-ISVM have seem to have better rates and more perfect decreasing trend than RS-ISVM. This clearly indicates that the idea of retaining support vectors as prior-knowledge for incremental learning yields better results.

The ROC curves of DR vs. FAR are shown in Fig. 7, where we can see smoother and more declining ROC curves produced by CSV-ISVM compared to the other ISVM methods. The shape of ROC curve of the Simple-ISVM is not smooth because this method discards initial data samples (including potential support vectors) and hence leads to instability (Yi et al., 2011). KKT-ISVM, though has improved ROC curve compared to simple ISVM, does not yet maintain the ratio DR/FAR all the time. The figure also illustrates that the CSV-ISVM yields the highest possible DR while maintaining the lowest FAR.

When compared specially with RS-ISVM, the CSV-ISVM seems to have maintained producing lesser FAR rates when DR exceeds about 85% throughout the subsequent increments as illustrated by further down-warded curve of CSV-ISVM as shown in Fig. 8. This indicates that even with the new data sample sets which keep coming in each new increments, the CSV-ISVM still produces increasing DR and decreasing FAR because the Half-partition strategy prevents the learning method from retaining potentially unwanted data points.

Table 2 compares time taken by all four methods in training and testing data samples in each increment. The amount of time taken for training data sets in the initial increment is almost same for all methods i.e. 1.8xx seconds, whereas that for testing has been reduced in CSV-ISVM to 7.76 s from 9.4 s in Simple-ISVM by saving 1.64 s. Similarly, in the last increment, 39.766 s of training time and 39.881 s of testing time have been saved by CSV-ISVM. Unlike other methods, KKT-ISVM, though has better DRs and FARs compared to Simple-ISVM, takes longer learning time,

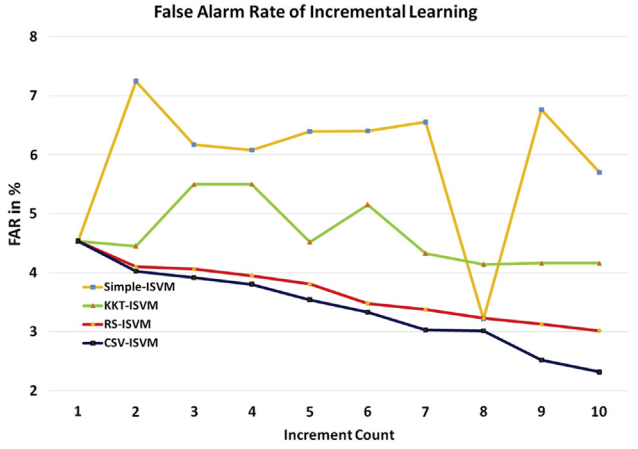

Fig. 6 – Comparison of FAR.

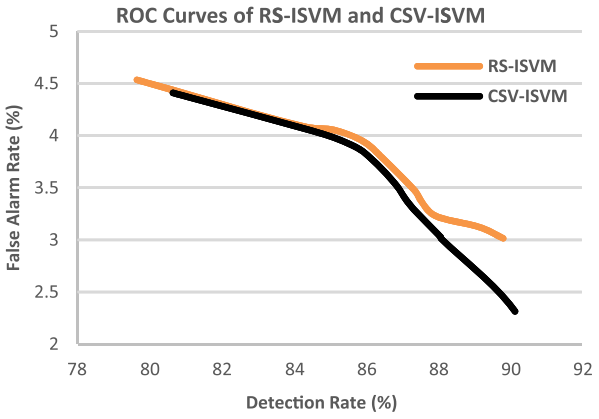

Fig. 8 – RS-ISVM and CSV-ISVM compared.

Table 2 – Comparison of training and testing time.

Iteration count	Simple-ISVM		KKT-ISVM		RS-ISVM		CSV-ISVM	
	Train (s)	Test (s)	Train (s)	Test (s)	Train (s)	Test (s)	Train (s)	Test (s)
1	1.843	9.4	1.842	9.48	1.833	8.58	1.823	7.76
2	6.407	24.688	12.578	27.005	4.935	17.5255	3.463	10.363
3	9.375	34.27	26.667	39.973	7.3175	24.8565	5.26	15.443
4	15.155	39.87	50.733	49.427	12.4085	29.701	9.662	19.532
5	31.433	44.427	63.833	57.162	21.3675	33.581	11.302	22.735
6	30.078	58.412	81.15	64.245	21.8355	42.1495	13.593	25.887
7	37.787	70.157	119.4	74.272	26.0675	49.1935	14.348	28.23
8	55.157	79.948	138.667	83.36	36.316	55.7815	17.475	31.615
9	46.9	73.022	160.45	88.412	34.96	54.2845	23.02	35.547
10	67.023	79.978	168.75	99.912	47.14	60.0375	27.257	40.097

illustrated also by Figs. 9 and 10, which is obvious because it has to do cross-judging and more training. It is clear that the time taken by CSV-ISVM for both training and testing are reduced at least by half compared to Simple-ISVM and KKT-ISVM. Moreover, due to the Half-partition strategy, time taken both in training and testing are also reduced remarkably compared to Concentric Circle method of RS-ISVM.

The amounts of time taken by all methods are obviously increasing as increments go on, but RS-ISVM and CSV-ISVM takes less time to learn in each increment compared to other two methods.

In other words, as shown in Table 3, the time difference between two subsequent increments for both training and testing data sets in one method is remarkably different from the other. The average training and testing time difference in case of Simple-ISVM are calculated to be 7.242 s and 7.842 s respectively, whereas those in case of CSV-ISVM are 2.826 s and 3.593 s respectively. This time difference between two increments may be regarded as the time taken to handle the new data set added in the later increment. This means that CSV-ISVM also handles the new data sets most efficiently of all other ISVM methods.

All afore-mentioned experimental results and in-depth analyses show clearly that the proposed CSV-ISVM has surpassed the Simple-ISVM, the KKT-ISVM and the RS-ISVM too in terms of DR, FAR as well as in terms of training and testing time.

7. Conclusion and further work

An improved approach to Incremental Support Vector Machine classification, called CSV-ISVM, has been proposed and applied to the incremental learning to SVM based network intrusion detection. The approach has contributed in two ways. Firstly, it has suggested modifications to the previously proposed Concentric Circle method. Secondly, it has proposed a more efficient and faster Support Vector selection method named Half-partition strategy along with the algorithm. The experiment works and the analyses have shown that the proposed method has better detection rate and false alarm rate as well as acceptable amount of learning time and, therefore, can be used for network intrusion detection in real-time.

The Half-partition strategy actually discards almost half the data points lying farther than the class centre from the hyperplane. And, such strategy cannot be implemented in methods other than binary classification. Modifying the strategy in order to make it work with multi-class classification might be considered as a future work of this research.

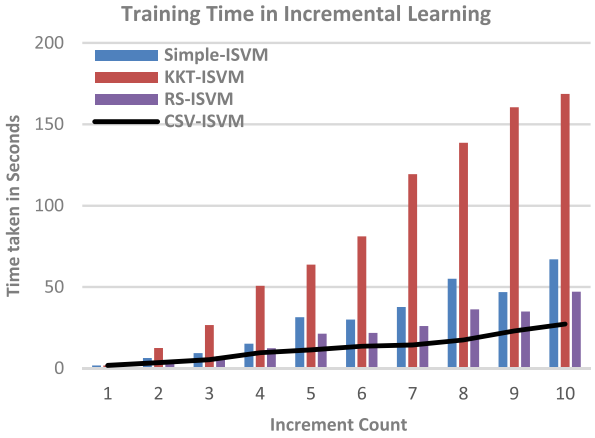

Fig. 9 – Comparison of training time.

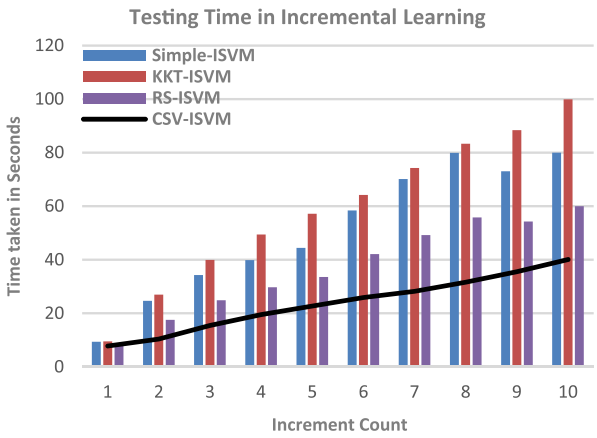

Fig. 10 – Comparison of testing time.

Table 3 – Comparison of time difference between two subsequent increments.

Iteration count	Simple-ISVM		KKT-ISVM		RS-ISVM		CSV-ISVM	
	Train (s)	Test (s)	Train (s)	Test (s)	Train (s)	Test (s)	Train (s)	Test (s)
1	*{This is the very first iteration, no incremental learning have been applied so far.}*							
2	4.564	15.288	10.736	17.525	3.102	8.9455	1.64	2.603
3	2.968	9.582	14.089	12.968	2.3825	7.331	1.797	5.08
4	5.78	5.6	24.066	9.454	5.091	4.8445	4.402	4.089
5	16.278	4.557	13.1	7.735	8.959	3.88	1.64	3.203
6	−1.355	13.985	17.317	7.083	0.468	8.5685	2.291	3.152
7	7.709	11.745	38.25	10.027	4.232	7.044	0.755	2.343
8	17.37	9.791	19.267	9.088	10.2485	6.588	3.127	3.385
9	−8.257	−6.926	21.783	5.052	−1.356	−1.497	5.545	3.932
10	20.123	6.956	8.3	11.5	12.18	5.753	4.237	4.55
Average	7.242222	7.842	18.54533	10.048	5.034111	5.7175	2.826	3.593

Appendix A. Supplementary data

Supplementary data related to this article can be found at http://dx.doi.org/10.1016/j.cose.2014.06.006.

REFERENCES

Bhavsar Himani, Panchal Mahesh H, Patel Mittal. A comprehensive study on RBF kernel in SVM for multiclsss using OAA. Int J Comput Sci Manag Res 2013;2(5).

Chitrakar Roshan, Huang Chuanhe. Anomaly based intrusion detection using hybrid learning approach of combining k-medoids clustering and Naïve Bayes classification. In: Proceedings of the 8th international conference on wireless communication, networking and mobile computing; 2012.

Chitrakar Roshan, Huang Chuanhe. Anomaly detection using support vector machine classification with k-medoids clustering. In: Proceedings of the third Asian Himalayan international conference on internet (AH-ICI); 2012.

Du Hongle, Teng Shaohua, Fu Xiufen, Zhang Wei, Pu Yuanfang. A cooperative intrusion detection system based on improved parallel SVM. In: Proceedings of Joint conferences on pervasive computing (JCPC); 2009.

Du Hongle, Teng Shaohua, Yang Mei, Zhu Qingfang. Intrusion detection system based on improved SVM incremental learning. In: Proceedings of international conference on artificial intelligence and computational intelligence; 2009.

Habib Tarek, Inglada Jordi, Mercier Grégoire, Chanussot Jocelyn. Support vector reduction in SVM algorithm for abrupt change detection in remote sensing. Proc IEEE Geosci Remote Sens Lett 2009;6(3).

Joachims T. Making large-scale support vector machine learning practical. In: Smola AJ, Scholkopf B, Burges C, editors. Advances in kernel methods: support vector machines. Cambridge: MIT Press; 1998.

Karasuyama M, Takeuchi Ichiro. Multiple incremental decremental learning of support vector machines. IEEE Trans Neural Netw 2010;21(7).

Kobayashi Takumi, Otsu Nobuyuki. Efficient reduction of support vectors in kernel-based methods. In: Proceedings of IEEE international conference on image processing. ICIP; 2009. pp. 2077–80.

Kyoto+ Data. Traffic data from Kyoto University's Honeypots. Retrieved, http://www.takakura.com/Kyoto_data; 2009. on 16.10.12.

Le Giang, Nguyen Hoang. Machine learning with informative samples for large and imbalanced data sets [PhD thesis]. School of Electrical, Computer and Telecommunication Engineering, School of Wollongong; 2011.

Liu Yangguang, He Qinming, Chen Qi. Incremental batch learning with support vector machines. In: Proceedings of the 5th world congress on intelligent control and automation; 2004. pp. 1857–61.

Makili L, Vega J, Dormido-Canto S. Incremental support vector machines for fast reliable image recognition. Fusion Engineering and Design 2013;88:1170–3.

Manning Christopher D, Raghavan Prabhakar, Schütze Hinrich. Introduction to information retrieval. Cambridge University Press; 2008.

Mohammad Muamer N, Sulaiman Norrozila, Khalaf Emad T. A novel local network intrusion detection system based on support vector machine. J Comput Sci 2011;7(10):1560–4.

Platt J. Fast training of support vector machines using sequential minimal optimization. In: Smola AJ, et al., editors. Advances in kernel methods: support vector machines. Cambridge: MIT Press; 1998.

Sun Na, Guo Yanfeng. A modified incremental learning approach for data stream classification. In: Sixth international conference on internet computing for science and engineering; 2012.

Tao Liang. Fast incremental SVM learning algorithm based on active set iteration. Chin J Syst Simul 2006;18(11):3305–8.

UCI KDD. The third international knowledge discovery and data mining tools competition dataset KDD Cup 1999 data. Retrieved, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html; 1999. on October 16, 2012.

Wang Wen-Jian. A redundant incremental learning algorithm for SVM. In: Proceedings of the seventh international conference on machine learning and cybernetics; 2008. pp. 734–8.

Wang XD, Zheng CY, Wu CM, Zhang HD. New algorithm for SVM-based incremental learning. Comput Appl 2006;26(10):2440–3.

Wen-Hua Z, Jian M. An incremental learning algorithm for support vector machine and its application. Comput Integr Manuf Syst Spec Mag 2001;9:144–8.

Yao Yuan, Feng Lin, Jin Bo, Chen Feng. An incremental learning approach with support vector machine for network data stream classification problem. Proc Inf Technol J 2012;11:200–8.

Yi Yang, Wu Jiansheng, Xu Wei. Incremental SVM based on reserved set for network intrusion detection. Expert Syst Appl 2011:7698–707.

Zhang Ying, Wang Xizhao, Zhai Junhai. A fast support vector machine classification algorithm based on Karush–Kuhn–Tucker conditions. In: Rough sets, fuzzy sets, data mining and granular computing. Lecture notes in computer science, vol. 5908; 2009. pp. 382–9.

Roshan Chitrakar, born in 1967, is recently pursuing a Ph.D. in Information Security at Wuhan University, Hubei, China. He was the Asst. Professor and Head of Department, Co-ordinator at colleges affiliated to Pokhara University, Nepal. His main areas of interest are Computer Security, Data Mining and Expert Systems, Real-Time Systems, Databases and Software Engineering etc. He has published a couple of books and a few papers in national and international journals and conferences.

Huang Chuanhe, Ph.D., born in 1963, is currently a Professor and Ph.D. students' supervisor at the School of Computers of Wuhan University, Hubei, China. He is also the Chief of Computer Networks Research Unit. His main interests of research are Computer Networks (MANET,WSN, Wireless Mesh, VANET), Internet of Things, Network Security, Distributed Parallel Processing etc.