

A Novel Incremental Covariance-Guided One-Class Support Vector Machine

Takoua Kefi^{1(✉)}, Riadh Ksantini^{1,2}, Mohamed Bécha Kaâniche¹,
and Adel Bouhoula¹

¹ Sécurité Numérique, Higher School of Communication of Tunis, Tunis, Tunisia
{takoua.kefi,medbecha.kaaniche,adel.bouhoula}@supcom.tn

² University of Windsor, 401, Sunset Avenue, Windsor, ON, Canada
ksantini@uwindsor.ca

Abstract. Covariance-guided One-Class Support Vector Machine (COSVM) is a very competitive kernel classifier, as it emphasizes the low variance projectional directions of the training data, which results in high accuracy. However, COSVM training involves solving a constrained convex optimization problem, which requires large memory and enormous amount of training time, especially for large scale datasets. Moreover, it has difficulties in classifying sequentially obtained data. For these reasons, this paper introduces an incremental COSVM method by controlling the possible changes of support vectors after the addition of new data points. The control procedure is based on the relationship between the Karush-Kuhn-Tucker conditions of COSVM and the distribution of the training set. Comparative experiments have been carried out to show the effectiveness of our proposed method, both in terms of execution time and classification accuracy. Incremental COSVM results in better classification performance when compared to canonical COSVM and contemporary incremental one-class classifiers.

Keywords: One-class classification · Incremental learning · Support Vector Machine · Covariance

1 Introduction

One-Class Classification is considered as one of the most challenging areas of machine learning. It has gained a lot of attention and it can be found in many practical applications such as medical analysis [1], face recognition [2], authorship verification [3].

To solve one-class classification problems, several methods have been proposed and different concrete models have been constructed. However, the key limitation of the existing categories of one-class classification methods is that none of them consider the full scale of information available. In boundary-based methods, like the One-Class Support Vector Machine (OSVM) [4] or Support Vector Data Description (SVDD) [5], only boundary data points are considered to build the model, and the overall class is not completely considered.

Besides, unlike multi-class classification problems, the low variance directions of the target class distribution are crucial for one-class classification. In [6], it has been shown that projecting the data in the high variance directions (like PCA) will result in higher error (bias), while retaining the low variance directions will lower the total error. As a solution, Naimul Mefraz Khan et al. proposed in [7] to put more emphasis on the low variance directions while keeping the basic formulation of OSVM untouched, so that we still have a convex optimization problem with a unique global solution, that can be reached easily using numerical methods. Covariance Guided One-Class Support Vector Machine (COSVM) is a powerful kernel method for one-class classification, inspired from the Support Vector Machine (SVM), where the covariance matrix is incorporated into the dual optimization problem of OSVM. The covariance matrix is estimated in the kernel space. Concerning its classification performance, success of COSVM has been shown when compared to SVDD and OSVM. However, there are still some difficulties associated with COSVM application in real case problems, where data are sequentially obtained and learning has to be done from the first data. Besides, COSVM requires large memory and enormous amount of training time, especially for large dataset.

Implementations for the existing One-Class Classification methods assume that all the data are provided in advance, and learning process is carried out in the same step. Hence, these techniques are referred to as batch learning. Because of this limitation, batch techniques show a serious performance degradation in real-word applications when data are not available from the very beginning. For such situation, a new learning strategy is required. Opposed to batch learning, incremental learning is more effective when dealing with non-stationary or very large amount of data. Thus, it finds its application in a great variety of situations such as visual tracking [8], software project estimation [9], brain computer interfacing [10].

It has been defined in [11] with 4 criteria:

1. it should be able to learn additional information from new data
2. it should not require access to the original data
3. it should preserve previously acquired knowledge and use it to update an existing classifier.
4. it should be able accommodate new outliers and target samples.

Several learning algorithms have been studied and modified to incremental procedures, able to learn through time. Cauwenberghs and Poggio [12] proposed an online learning algorithm of Support Vector Machine (SVM). Their algorithm changes the coefficient of original Support Vectors (SV), and retains the Karuch-Kuhn-Tucker (KKT) conditions on all previously training data as a new sample acquired. Their approach have been extended by Laskov et al. [13] to OSVM. However, the performance evaluation was only based on multi-class SVM. From their side, Manuel Davy et al. introduced in [14] an online SVM for abnormal events detection. They proposed a strategy to perform abnormality detection over various signals by extracting relevant features from the considered signal and detecting novelty, using an incremental procedure. Incremental SVDD proposed

in [15] is also based on the control of the variation of the KKT conditions as new samples are added. An other approach to improve the classification performance is introduced in [16]. Incremental Weighted One-Class Support Vector Machine (WOCSVM) is an extension of incremental OSVM. The proposed algorithm aims to assign weights to each object of the training set, then it controls its influence on the shape of the decision boundary.

All the proposed Incremental One-Class SVM inherit the problem of classic SVM method which uses only boundary points to build a model, regardless of the spread of the remaining data. Also, none of them emphasizes the low variance direction, which results in performance degradation. Therefore, in this paper we try to solve mainly this problem by using an incremental COSVM (iCOSVM) approach. In fact, iCOSVM has the advantage of incrementally emphasizing the low variance direction to improve classification performance, which is not the case for classical incremental one-class models. Our preposition aims to take advantages from the accuracy of COSVM procedure and we prove that it is a good candidate for learning in non-stationary environments.

The rest of the paper is organized as follows. Section 2 reviews the canonical COSVM method since it is the basis of our proposed method. In Sect. 3, we present in details the mathematical derivation of iCOSVM and we describe the incremental algorithm. Section 4 presents our experimental studies and comparison with canonical COSVM and other incremental one-class classifiers. Finally, Sect. 5 contains some concluding remarks and perspectives.

2 The COSVM Method

Mathematically, OSVM tries to find the hyperplane that separates the training data from the origin with maximum margin. It can be modeled by the following dual problem, formulated using Lagrange multipliers.

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T \mathbf{K} \alpha + b \left(1 - \sum_{i=1}^N \alpha_i \right). \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu N} = C, \quad \sum_{i=1}^N \alpha_i = 1. \end{aligned} \quad (1)$$

Here, $\nu \in (0, 1]$ is a key parameter that controls the fraction of outliers and that of support vectors, C is the penalty weight punishing the misclassified training examples, $\mathbf{K}(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle, \forall i, j \in \{1, 2, \dots, N\}$ is the kernel matrix for the training data, and α are the Lagrange multipliers to be determined.

The covariance matrix is then plugged in the dual problem and a parameter $\eta \in [0, 1]$ is introduced to control the contribution of the kernel matrix \mathbf{K} and the covariance matrix to the objective function. The COSVM optimization problem can be written as follows:

$$\min_{\alpha} W(\alpha, b) = \frac{1}{2} \alpha^T (\eta \mathbf{K} + (1 - \eta) \Delta) \alpha - b \left(1 - \sum_{i=1}^N \alpha_i \right). \quad (2)$$

$$s.t. \quad 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^N \alpha_i = 1,$$

where $\Delta = \mathbf{K} (I - \mathbf{1}_N) \mathbf{K}^T$. The control parameter η can take values from 0 to 1.

3 The Incremental COSVM Method

The key of our method is to construct a solution recursively, by adding one point at a time [12], and retain the Karush-Kuhn-Tucker Conditions on all previously acquired data.

3.1 Karush-Kuhn-Tucker Conditions

Both the kernel matrix \mathbf{K} and the covariance matrix Δ are positive definite [17]. Therefore, the proposed method still results in a convex optimization problem. Thus, the solution to this optimization problem will have one global optimum solution and can be solved efficiently using a mathematical method. Karush-Kuhn-Tucker (KKT) conditions [18] are among the most important theoretical optimization methods.

First, let's note

$$\Gamma = (\eta \mathbf{K} + (1 - \eta) \Delta).$$

The slopes g_i of the cost function W in equation (2) are expressed using the KKT conditions as:

$$g_i = \frac{\partial W}{\partial \alpha} = \sum_j \Gamma_{i,j} \alpha_j - b \begin{cases} \geq 0; & \alpha_i = 0 \\ = 0; & 0 < \alpha_i < C \\ \leq 0; & \alpha_i = C \end{cases} \quad (3)$$

$$\frac{\partial W}{\partial b} = 1 - \sum \alpha = 0. \quad (4)$$

According to the KKT conditions above, the target training data can be divided into three categories, shown in Figs. 1, 2 and 3:

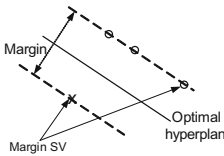


Fig. 1. Subset \mathcal{S} . $g_i = 0$ and $0 < \alpha_i < C$

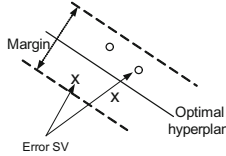


Fig. 2. Subset \mathcal{E} . $g_i < 0$ and $\alpha_i = C$

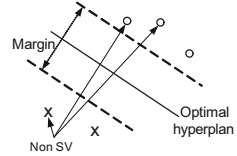


Fig. 3. Subset \mathcal{O} . $g_i > 0$ and $\alpha_i = 0$

1. Margin or unbounded Support Vectors are training points $\mathcal{S} = \{i/0 < \alpha_i < C\}$,
2. Error or bounded Support Vectors $\mathcal{E} = \{i/\alpha_i = C\}$,
3. Non Support Vectors $\mathcal{O} = \{i/\alpha_i = 0\}$.

The KKT conditions have to be maintained for all trained data before a new data x_c is added and preserved after the new data is trained. Hence, the change of Lagrange multipliers $\Delta\alpha$ is determined to hold the KKT.

3.2 Adiabatic Increments

To maintain the equilibrium of the KKT conditions expressed in Eqs. (3) and (4), we express them differentially:

$$\Delta g_i = \Gamma_{i,c}\alpha_c + \sum_j \Gamma_{i,j}\alpha_j - \Delta b, \quad (5)$$

$$\Delta\alpha_c + \sum_{j \in \mathcal{S}} \alpha_j = 0. \quad (6)$$

The two equations above can be written as:

$$\begin{bmatrix} \Delta g_c \\ \Delta g_s \\ \Delta g_r \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & \Gamma_{c,s} \\ 1 & \Gamma_{s,s} \\ 1 & \Gamma_{r,s} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -\Delta b \\ \Delta\alpha_s \end{bmatrix} + \Delta\alpha_c \begin{bmatrix} \Gamma_{c,c} \\ \Gamma_{s,c} \\ \Gamma_{r,c} \\ 1 \end{bmatrix}. \quad (7)$$

Since $\Delta g_i = 0$ when $i \in \mathcal{S}$ (it remains zero), lines 2 and 4 of the system (7) can be re-written as:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1} & \Gamma_{s,s} \end{bmatrix} \begin{bmatrix} -\Delta b \\ \Delta\alpha_s \end{bmatrix} + \Delta\alpha_c \begin{bmatrix} \mathbf{1} \\ \Gamma_{s,c} \end{bmatrix}. \quad (8)$$

Thus, we can express the dependence of $\Delta\alpha_i$, $i \in \mathcal{S}$ and $\Delta g_i = 0$, $i \notin \mathcal{S}$ on $\Delta\alpha_c$ as the following:

$$\begin{bmatrix} -\Delta b \\ \Delta\alpha_s \end{bmatrix} = -\mathbf{R} \begin{bmatrix} 1 \\ \Gamma_{s,c} \end{bmatrix} \Delta\alpha_c, \quad (9)$$

with

$$\mathbf{R} = \begin{bmatrix} 0 & \mathbf{1} \\ \mathbf{1} & \Gamma_{s,s} \end{bmatrix}^{-1}.$$

Here, $\Gamma_{s,s}$ is the kernel matrix whose entries are support vectors, and $\Gamma_{s,c}$ is a vector of kernels between the margin support vectors and the new candidate vector x_c .

The Eq. (9) gives the following:

$$\begin{bmatrix} -\Delta b \\ \Delta\alpha_s \end{bmatrix} = \beta \Delta\alpha_c,$$

where

$$\beta = -\mathbf{R} \begin{bmatrix} 1 \\ \Gamma_{s,c} \end{bmatrix}. \quad (10)$$

In equilibrium,

$$\begin{cases} \Delta b = -\beta_b \Delta \alpha_c, \\ \Delta \alpha_j = \beta_j \Delta \alpha_c, j \in \mathcal{S} \end{cases} \quad (11)$$

and $\beta_j = 0$ for all j outside the subset \mathcal{S} .

Substituting Eq. (11) into lines 1 and 3 of the system (7) leads to the desired relation between Δg_i and $\Delta \alpha_c$:

$$\Delta g_i = \gamma_i \Delta \alpha_c, i \in \{1 \dots n\} \cup \{c\} \quad (12)$$

where we define

$$\begin{cases} \gamma_i = \Gamma_{i,c} + \sum_{j \in \mathcal{S}} \Gamma_{i,j} \beta_j, & i \notin \mathcal{S} \\ \gamma_i = 0, & i \in \mathcal{S} \end{cases} \quad (13)$$

3.3 Vectors Entering and Leaving a Subset

During the incremental procedure, a new example x_c can be added to the previous training set, and depending on the value of the calculated parameters g_c and α_c , the x_c is recognized as a support vector, an error vector or a data vector. If x_c is classified as a support vector, the set \mathcal{S} , as well as the classification boundaries and margins should be updated. Since the Margin Support Vectors are our first concern in a classification process, it is worth to focus on the changes in the subset \mathcal{S} . Besides, we can see from the Eqs. (10), (11), (12) and (13) of the previous section, that only \mathbf{R} matrix needs to be computed to obtain all updated parameters. Let us consider a vector x_k entering to the subset \mathcal{S} . Using the Woodbury formula [19], \mathbf{R} expands as:

$$\tilde{\mathbf{R}} = \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\gamma_c} \begin{bmatrix} \beta \\ 1 \end{bmatrix} \begin{bmatrix} \beta \\ 1 \end{bmatrix}^T. \quad (14)$$

When x_k leaves \mathcal{S} , and using the same formula, \mathbf{R} contracts as:

$$\tilde{\mathbf{R}} = \mathbf{R}_{\bar{k},\bar{k}} - \mathbf{R}_{k,k} \mathbf{R}_{\bar{k},k} \mathbf{R}_{k,\bar{k}}. \quad (15)$$

3.4 The Impact of the Tradeoff Parameter η

The contribution of our kernel matrix \mathbf{K} and the covariance matrix Δ is controlled using the parameter η . Figures 4, 5 and 6 present three different cases showing the impact of the covariance matrix on the direction of the separating hyperplane in the kernel space optimality. In Fig. 4, the optimal decision hyperplane is on the same direction as the high variance direction. Hence, the low variance direction will not improve the separating direction. That is why the

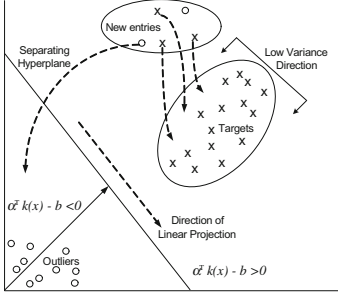


Fig. 4. Case 1: schematic depiction of the decision hyperplane for iCOSVM when the optimal control parameter value is $\eta = 1$. The optimal linear projection is along the direction of high variance.

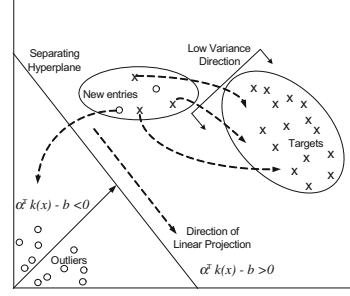


Fig. 5. Case 2: Schematic depiction of the decision hyperplane for iCOSVM when the optimal control parameter value is $\eta = 0$. The optimal linear projection is along the direction of low variance.

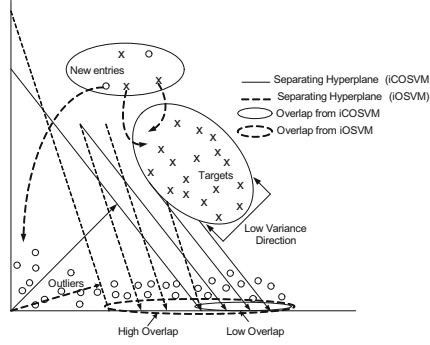


Fig. 6. General Case: schematic depiction of the decision hyperplane for iCOSVM when the optimal parameter value lies in between 0 and 1 ($0 < \eta < 1$). The linear projection direction for iOSVM (depicted by dotted arrows) results in higher overlap between the example target and hypothetical outlier data (circled by dotted boundary) than the iCOSVM projection direction (depicted by solid arrows and the overlap circled by solid boundary).

value of η should be set to 1 in order to eliminate the covariance matrix term. On the other hand, in Fig. 5, the directions of the optimal decision hyperplane and the low variance are parallel. Therefore the incremental OSVM (iOSVM) term (kernel matrix) is ignored by setting η to 0. However, in real world cases, it is very rare that the optimal decision hyperplane has the same direction as the low or high variance. For this reason, the value of η needs to be tuned so that we have less overlap between the linear projections of the target data and the outlier data. As Fig. 6 shows, by using optimal η value, iCOSVM can reduce the huge overlap caused by iOSVM projection.

3.5 The Incremental Algorithm

Our implementation of incremental Covariance-guided One-Class SVM is presented as pseudo-code in Algorithm 1.

Algorithm 1. Incremental Covariance-guided One-Class SVM algorithm

1. **Initialization**
 2. Compute \mathbf{R} , and use it to compute β and γ according to Eqs. (10) and (13)
 3. Set α_c and $\Delta\alpha_c$ to 0
 4. Compute g_c using Eq. (3)
 5. **While** $g_c < 0$ and $\alpha_c < C$ **do**
 6. **if** $g_c = 0$ then x_c is a margin support vector. Add c to \mathcal{S} and equilibrium is reached. Set $\alpha_c = \Delta\alpha_c$ and update $(\alpha_i)_{i=1\dots n}$. Update \mathbf{R} , and b .
 7. **if** $g_c < 0$ then x_c is an error support vector. Add c to \mathcal{E} and equilibrium is reached. Set $\alpha_c = C$, update $(\alpha_i)_{i=1\dots n}$ and b .
 8. **if** a support vector reaches its upper bound, x_k becomes a non-support vector. Remove k from \mathcal{S} and add it to \mathcal{O} . Update \mathbf{R} , $(\alpha_i)_{i=1\dots n}$ and b .
 9. **if** a support vector reaches its lower bound, x_k becomes an error support vector. Remove k from \mathcal{S} and add it to \mathcal{E} . Update \mathbf{R} , $(\alpha_i)_{i=1\dots n}$ and b .
 10. **if** g_i becomes 0, x_k becomes a support vector. Add k to \mathcal{S} Update \mathbf{R} , $(\alpha_i)_{i=1\dots n}$ and b .
-

If the equilibrium is not reached, parameters are sequentially moved until the equilibrium is met. We aim to determine the largest possible increment $\Delta\alpha_c$ so that the decomposition of the set remains intact, while accounting for the movement of some data from set to another during the update process. This is the idea of adiabatic increments [12].

4 Experimental Results

In this section, we present detailed experimental analysis and results for our proposed method, performed on artificially synthesized dataset and real world datasets. We have evaluated the performance of our method with two different experiment sets. In the first one, we compared the accuracy and time results with non-incremental COSVM, to tease out the advantage of our incremental model over batch learning model. In the second experiment set, we compare the iCOSVM performance against the performance of contemporary incremental one-class classifiers, to show the advantage of incrementally projecting data in low variance directions. For the implementation, we used Tax’s data description toolbox [20] in Matlab. First, we provided an analysis of the effect of tuning the key control parameter η . This analysis will lead us to decide how to optimize the value of η for a particular dataset.

4.1 Optimising the Value of η

Cross validation can not be used to optimize the value of η . Therefore, a stopping criterion is considered to find the optimum value. We use a pre-defined lowest fraction of outliers allowed (f_{OL}) as a stopping criterion. For new datasets, we set η to 1, and we decrease its value, while observing the fraction of outliers. When it hits (f_{OL}), we stop and use the current value of η for the considered dataset. We have to mention that there is no conflict between (f_{OL}) and the OSVM parameter ν , and they can be set independently to fit the purpose of the dataset to be trained on. There is no strict conditions on how to choose the value of ν , it can be set to any value from 0 to 1 [21]. For our additional parameter (f_{OL}), it is set to any value between 0 to ν .

Table 1. Description of datasets.

| Dataset name | Number of targets | Number of outliers | Number of features |
|--------------------|-------------------|--------------------|--------------------|
| Biomedical | 67 | 127 | 5 |
| Heart disease | 160 | 137 | 13 |
| Liver (diseased) | 145 | 200 | 6 |
| Liver (healthy) | 200 | 145 | 6 |
| Diabetes (absent) | 268 | 500 | 8 |
| Diabetes (present) | 500 | 268 | 8 |
| Arrhythmia-1 | 237 | 183 | 278 |
| Arrhythmia-2 | 36 | 384 | 278 |
| Chromosome-1 | 392 | 751 | 30 |
| Chromosome-2 | 447 | 696 | 30 |
| Chromosome-3 | 492 | 651 | 30 |
| Chromosome-4 | 536 | 598 | 30 |

4.2 Datasets Used

We have used both artificially generated datasets and real world datasets in our experiments to tease out the effectiveness of our proposed method in different scenarios. For the experiments on artificially generated data, we have created a number of 2D two class data drawn from two different set of distributions: (1) Gaussian distribution with different covariance matrices. (2) Banana-shaped distribution with different variances. For each distribution, two datasets were generated, the first one with low overlap, and the second with high overlap. Each class of each dataset was used as a target class and outliers in turns, such that we evaluate the performance on 8 datasets (2 distributions \times 2 classes \times 2 overlaps). Figure 7 presents the plots of the generated datasets.

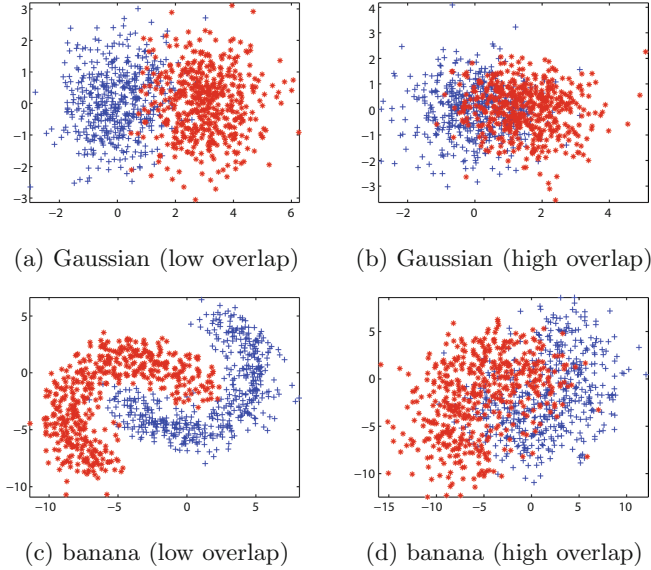


Fig. 7. Artificial datasets used for comparison. The two shapes denote two different classes generated from a pre-defined distribution. Each class was used as target and outlier in turns.

For the real world case, we focused on medical datasets as this domain is one of the key fields where one class classification is applied [1]. A detailed description of the used datasets can be found in Table 1. These datasets are collected from the UCI machine learning repository [22] and picked carefully, so that we have a variety of sizes and dimensions, and we can, then, test the robustness of our iCOSVM. As these datasets are originally two-class or multi-class, we used one of them as a target class and the other ones are kept outliers.

4.3 Experimental Protocol

To make sure that our results are not coincidental or overoptimistic, we used a cross-validation process [23]. The considered dataset was randomly split into 10 subsets of equal size. To build a model, one of the 10 subsets was removed, and the rest was used as the training data. The previously removed subset was added to the outliers and this whole set was used for testing. Finally the 10 accuracy estimates are averaged to provide the accuracy over all the models of a dataset. This guarantees that the achieved results were not a coincidence. Moreover, to measure the performance of one class classifier, the Receiver Operating Characteristic (ROC) curves [24] are usually used. The ROC curve presents a powerful measurement of the performance of studied classifier. It does not depend on the number of training or testing data points neither on the number of outliers, it only depends on rates of correct and incorrect target detection. To evaluate the

methods, we have also used the Area Under Curve (AUC) [25] produced by the ROC curves, and we presented them in the results.

4.4 Classifiers

The iCOSVM was evaluated with the comparison of its performance against the following classifiers' performance:

- COSVM: Since our incremental approach is built upon COSVM, this classifier has been described in details in Sect. 2.
- iOSVM: This method tries to find, recursively, the maximum margin hyper-plane that separates targets from outliers.
- iSVDD: This method gives the sphere boundary description of the target data points with minimum volume.

The incremental classifiers, iOSVM, iSVDD and iCOSVM were implemented with the help of DDtools [20]. For the implementation of COSVM, the SVM-KM toolbox was used [26]. The radial basis kernel was used for kernelization. This kernel is calculated as $\mathcal{K}(x_i, x_j) = e^{-\|x_i - x_j\|^2 / \sigma}$. It is proved to be robust and flexible [27]. Here, σ represents the positive “width” parameter. For η value optimization, the value of σ was set to 1. But, when comparing with other methods σ is optimized first. The parameter ν for COSVM, iOSVM and iCOSVM, also called *fraction of rejection* in the case of iSVDD was set to 0.2.

While optimizing η , the lowest threshold for the fraction of outliers (f_{OL}) was set to 0.1 (see Sect. 4.1). However, it is too difficult, and even not possible to define optimal values for the parameters f_{OL} and ν in real cases, where data points are unknown in the beginning of the classification process. Therefore, we have set both of the two parameters to 0.2.

4.5 Results and Discussion

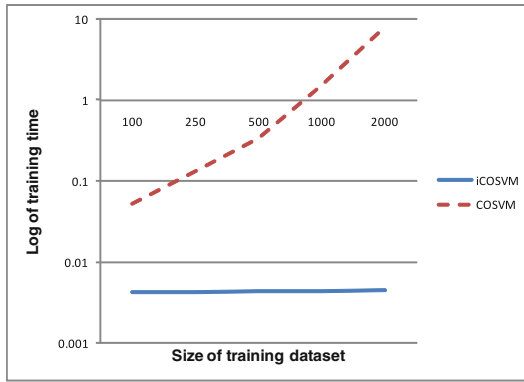
To test the effectiveness of our proposed algorithm, we started by comparing iCOSVM with canonical COSVM on artificially generated datasets.

As we can see in Table 2, iCOSVM provides better results in terms of AUC values, on all datasets, by averaging over 10 different models. Figure 8 shows the average training time per model for artificial datasets of different sizes. The training speed of our algorithm is faster than the COSVM, mainly on large data sets, and presents insignificant variation as the size of the dataset increases. It has been shown in a number of recent studies [28] that incremental learning algorithms outperform batch learning algorithms in both speed and accuracy, because they provide cleaner solution.

In fact, the complexity for solving the convex optimization problem of COSVM is $O(N^3)$, where N is the number of training data points. Whereas, a key to efficiency of the iCOSVM algorithm lies in identifying performance bottlenecks associated with inverting matrices to solve the convex optimization problem. These operations were eliminated thanks to the introduction of

Table 2. Average AUC of COSVM and iCOSVM for the 8 artificial datasets. Each dataset has 1000 data points(best method in **bold**).

| Dataset | COSVM | iCOSVM |
|-------------------------|-------|--------------|
| Gauss. (low overlap)-1 | 98.33 | 98.45 |
| Gauss. (low overlap)-2 | 98.28 | 98.52 |
| Gauss. (high overlap)-1 | 81.47 | 84.19 |
| Gauss. (high overlap)-2 | 87.14 | 87.74 |
| Banana (low overlap)-1 | 98.46 | 98.88 |
| Banana (low overlap)-2 | 98.33 | 99.26 |
| Banana (high overlap)-1 | 85.73 | 86.43 |
| Banana (high overlap)-2 | 84.88 | 84.97 |

**Fig. 8.** Log of training times (per model) in *seconds* for COSVM and iCOSVM for the experiments on the artificial datasets of different sizes.

the Woodbery formula for the re-computation of the gradient, β and γ . This involves matrix-vector multiplications and recursive updates of the matrix \mathbf{R} , whose dimension is equal to the support vectors number N_s . The running time needed for an update of the matrix \mathbf{R} is quadratic in the number of support vectors, which is much better than explicit inversion. Thus, in incremental learning, the complexity is $O(N_s^2)$, where $N_s \leq N$.

Tables 3 and 4 contain the average AUC for the incremental classifiers on the artificial and real datasets, respectively. As we can see, iCOSVM provides better results on all datasets. Specially in case of the biomedical and chromosome datasets, iCOSVM performs significantly better when compared to other methods. It is not surprising that iSVDD gives almost the worst accuracy values, as SVM and its derivatives are constructed to give the better separation [29].

We notice that iCOSVM outperforms the other classifiers as η values are in the neighborhood of 0.7, which puts more emphasize on the kernel matrix and fine-tune the contribution of the covariance matrix.

Table 3. Average AUC of each method for the 12 artificial datasets (best method in **bold**, second best *emphasized*).

| Experiment | iSVDD | iOSVM | iCOSVM |
|-------------------------|-------|-------------|-------------|
| Gauss. (low overlap)-1 | 90.7 | <i>95.0</i> | 95.2 |
| Gauss. (low overlap)-2 | 92.2 | <i>95.2</i> | 96.0 |
| Gauss. (high overlap)-1 | 71.7 | <i>73.8</i> | 75.2 |
| Gauss. (high overlap)-2 | 69.8 | <i>73.3</i> | 76.3 |
| Banana (low overlap)-1 | 95.2 | <i>97.5</i> | 97.8 |
| Banana (low overlap)-2 | 92.2 | <i>97.3</i> | 97.5 |
| Banana (high overlap)-1 | 74.8 | <i>83.2</i> | 83.7 |
| Banana (high overlap)-2 | 74.0 | <i>81.0</i> | 82.8 |

Table 4. Average AUC of each method for the 12 real-world datasets (best method in **bold**, second best *emphasized*).

| Experiment | iSVDD | iOSVM | iCOSVM |
|--------------------|-------------|-------------|-------------|
| Biomedical | 28.4 | <i>77.7</i> | 82.9 |
| heart disease | 49.4 | <i>60.9</i> | 61.9 |
| Liver (diseased) | 54.8 | <i>69.1</i> | 69.6 |
| Liver (healthy) | 52.5 | <i>67.3</i> | 68.7 |
| Diabetes (present) | 95.2 | <i>97.5</i> | 97.8 |
| Diabetes (normal) | 92.2 | <i>97.3</i> | 97.5 |
| arrhythmia-1 | 74.8 | <i>83.2</i> | 83.7 |
| arrhythmia-2 | 74.0 | <i>81.0</i> | 82.8 |
| Chromosome-1 | 48.0 | <i>65.2</i> | 78.2 |
| Chromosome-2 | 48.2 | <i>63.2</i> | 73.4 |
| Chromosome-3 | <i>47.4</i> | 46.6 | 55.5 |
| Chromosome-4 | 47.9 | <i>58.6</i> | 70.8 |

Since the process of computing the covariance matrix is done as a pre-processing and re-used during all training phase, in terms of training complexity, iCOSVM does not have additional overhead on top of the original iOSVM. Table 5 shows the average training times per model for both the artificial and the real-world datasets. As we expect, iCOSVM performs almost as fast as iOSVM, while providing better classification accuracy.

Also, we present some individual graphical results for the dataset models by plotting the actual ROC curves for a real world dataset. Figure 9 shows the ROC curves of the three incremental classifiers for four models of the chromosome dataset. The rule-of-thumb to judge the performance of a classifier from a ROC

Table 5. Average training times (per model) in *seconds* for iOSVM and iCOSVM for the experiments on the artificial and real-world datasets. Average training times (per model) in *seconds* for iOSVM and iCOSVM for the experiments on the artificial and real-world datasets.

| Experiment | iOSVM | iCOSVM |
|---------------------|--------|--------|
| Artificial datasets | 0.0047 | 0.0046 |
| Real-world datasets | 0.0044 | 0.0043 |

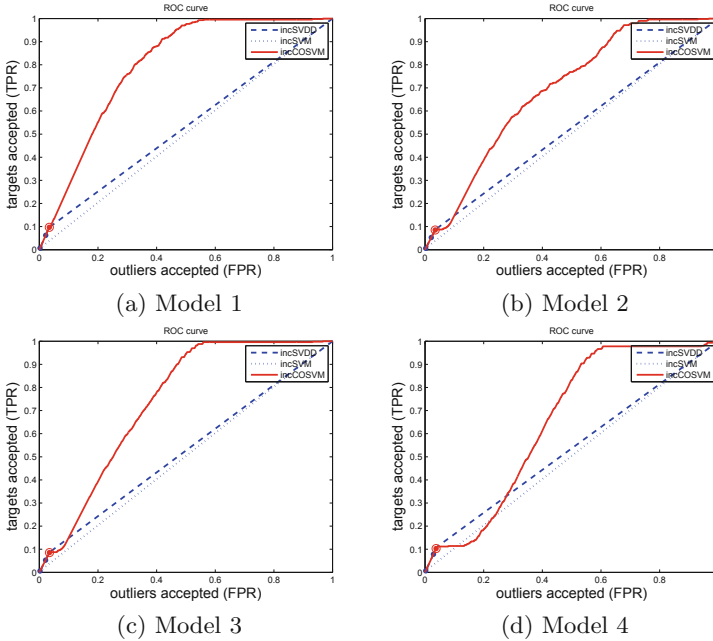


Fig. 9. ROC curves for the three incremental classifiers applied on Chromosome dataset

curve is “The best classification has the largest area under curve”. We can clearly see from the Fig. 9 that iCOSVM indeed leads to better ROC curves.

5 Conclusion

In this paper, we have proposed an incremental Covariance-guided One-Class Support Vector Machine (iCOSVM) classification approach. iCOSVM improves upon the incremental One-Class Support Vector Machine method by the incorporation of the covariance matrix into the optimization problem. The new introduced term emphasized the projection in the directions of low variances of the training datasets. The contribution of both Kernel and covariance matrices are controlled via a parameter that was tuned efficiently for optimum performance. iCOSVM takes advantages from the high accuracy of the canonical

Covariance-guided One-Class Support Vector Machine (COSVM). We have presented detailed experiments on several artificial and real-world datasets, where we compared our method against contemporary batch and incremental learning methods. Results have shown the superiority of the method. Future works will consist in validating these results on strong applications such as face recognition, anomaly detection, etc.

References

1. Gardner, A.B., Krieger, A.M., Vachtsevanos, G., Litt, B.: One-class novelty detection for seizure analysis from intracranial EEG. *J. Mach. Learn. Res.* **7**, 1025–1044 (2006)
2. Zeng, Z., Fu, Y., Roisman, G.I., Wen, Z., Hu, Y., Huang, T.S.: Spontaneous emotional facial expression detection. *J. Multimedia* **1**, 1–8 (2006)
3. Koppel, M., Schler, J.: Authorship verification as a one-class classification problem. In: 21st International Conference on Machine Learning, pp. 62–68. ACM, New York (2004)
4. Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., Williamson, R.: Estimating the support of a high-dimensional distribution. *Neural Comput.* **13**, 1443–1471 (2001)
5. Tax, D.M.J.: One-class classification. Concept-learning in the absence of counter-examples. Ph.D. Thesis, Technical University of Delft, June 2001
6. Tax, D.M.J., Müller, K.R.: Feature extraction for one-class classification. In: Artificial Neural Networks and Neural Information Processing, pp. 342–349. IEEE Press (2003)
7. Khan, N.M., Ksantini, R., Ahmad, I.S., Guan, L.: Covariance-guided one-class support vector machine. *Pattern Recognit.* **47**, 2165–2177 (2014)
8. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. *Int. J. Comput. Vis.* **77**, 125–141 (2007)
9. Giraud-Carrier, C.: A note on the utility of incremental learning. *AI Commun.* **13**, 215–223 (2000)
10. Blankertz, B., Dornhege, G., Schäfer, C., Krepki, R., Kohlmorgen, J., Müller, K.R., Kunzmann, V., Losch, F., Curio, G.: Boosting bit rates and error detection for the classification of fast-paced motor commands based on single-trial EEG analysis. *IEEE Trans. Neural Syst. Rehabil. Eng.* **11**, 127–131 (2003)
11. Polikar, R., Upda, L., Atish, S., Upda, S., Honavar, V.: Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Syst. Man Cybern.* **31**, 497–508 (2002)
12. Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: Neural Information Processing Systems, pp. 409–415. NIPS (2000)
13. Laskov, P., Gehl, C., Krüger, S.: Incremental support vector learning: analysis, implementation and applications. *J. Mach. Learn. Res.* **7**, 1909–1936 (2006)
14. Davy, M., Desorby, F., Gretton, A., Doncarli, C.: An online support vector machine for abnormal events detection. *Sig. Process.* **86**, 2009–2025 (2005)
15. Hua, X., Ding, S.: Incremental learning algorithm for support vector data description. *J. Softw.* **6**, 1166–1173 (2011)
16. Krawczyk, B., Woźniak, M.: Incremental weighted one-class classifier for mining stationary data streams. *J. Comput. Sci.* **9**, 19–25 (2015)

17. Horn, R.A., Charles, R.: *Matrix Analysis*. Cambridge University Press (1990)
18. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: The Fritz John and Karush-Kuhn-Tucker optimality conditions. In: *Nonlinear Programming: Theory and Algorithms*. Wiley, Singapore (2006)
19. Hager, W.W.: Updating the inverse of a matrix. *Soc. Ind. Appl. Math.* **31**, 221–239 (1989)
20. Tax, D.M.J.: *DDtools, the Data Description Toolbox for Matlab*, version 2.1.2 (2015)
21. Parra, L., Deco, L., Miesbach, S.: Statistical independence and novelty detection with information preserving nonlinear maps. *Neural Comput.* **8**, 260–269 (1996)
22. Lichman, M.: *Machine Learning Repository*. University of California, Iverine, School of Information and computer Sciences (2013)
23. Alippi, C., Roveri, M.: Virtual k-fold cross validation: an effective method for accuracy assessment. In: *International Joint Conference on Neural Networks*, pp. 1–6. IEEE Conference Publications (2010)
24. Fawcett, T.: An introduction to ROC analysis. *Pattern Recogn. Lett.* **27**, 861–874 (2006)
25. Hanley, J., McNeil, B.J.: A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology* **148**, 839–843 (1983)
26. Canu, S., Grandvalet, Y., Guigue, V., Rakotomamonjy, A.: *SVM and Kernel Methods Matlab Toolbox*. Perception Systmes et Information, INSA de Rouen, Rouen, France (2005)
27. Cristianini, N., Shawe-Taylor, J.: *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge (2000)
28. Lorenzo, C.Y., Guan, B.R., Lu, L., Liang, Y.: Incremental and decremental affinity propagation for semisupervised clustering in multispectral images. *IEEE Trans. Geosci. Remote Sens.* **51**, 1666–1679 (2013)
29. Wei, H., Li, J.: Credit scoring based on eigencredits and SVDD. In: *Applied Informatics and Communication*, pp. 32–40 (2011)

Machine Learning and Knowledge Discovery in
Databases

European Conference, ECML PKDD 2016, Riva del
Garda, Italy, September 19-23, 2016, Proceedings, Part
II

Frasconi, P.; Landwehr, N.; Manco, G.; Vreeken, J. (Eds.)
2016, XXVIII, 825 p. 205 illus., Softcover
ISBN: 978-3-319-46226-4