SACJ 29(1) July 2017
Research Article

Adaptive SVM for Data Stream Classification

Isah A. Lawal[a], Salihu A. Abdulkarim[b]

[a] Department of Computer Engineering and Networks, AlJouf University, Saudi Arabia
[b] Department of Computer Science, Federal University Dutse, Nigeria

ABSTRACT
In this paper, we address the problem of learning an adaptive classifier for the classification of continuous streams of data. We present a solution based on incremental extensions of the Support Vector Machine (SVM) learning paradigm that updates an existing SVM whenever new training data are acquired. To ensure that the SVM effectiveness is guaranteed while exploiting the newly gathered data, we introduce an on-line model selection approach in the incremental learning process. We evaluated the proposed method on real world applications including on-line spam email filtering and human action classification from videos. Experimental results show the effectiveness and the potential of the proposed approach.

Keywords: incremental learning, support vector machine, spam filtering, human action classification

Categories: • **Machine Learning ~ Learning settings**

Email:
Isah A. Lawal ialawal@ju.edu.sa (CORRESPONDING),
Salihu A. Abdulkarim sakwami@fud.edu.ng

Article history:
Received: 3 September 2016
Accepted: 4 May 2017
Available online: 9 July 2017

1 INTRODUCTION

Over the last few years, many real-world applications that generate continuous streams of data have emerged (Nguyen, Woon, & Ng, 2015). For efficient interpretation of these streams of data, a timely and meaningful classification process is required. A classification process involves using a set of training data to learn a computational model (classifier) and then employing the developed model to classify a previously unseen stream of data (Dongre & Malik, 2014). Classical learning methods perform classification tasks off-line using a classifier trained on streams of data gathered in the past (Nguyen et al., 2015). However, several applications require on-line classification. For example, detecting abnormal human actions from camera-based real-time scene monitoring (Lu, Boukharouba, Boonært, Fleury, & Lecuche, 2014), and timely separation of legitimate emails from spam in an on-line spam email filtering (Vipin & Nizar, 2014).

However, performing on-line data classification is problematic for two major reasons (Fong, Luo, & Yap, 2013). First, data are generated in batches over time, it is difficult at one time to acquire sufficient training data that are representative of all the underlying classification problems in the application. Therefore, when the classifier is trained on the limited data only once within a given time span, the learned classifier will perform poorly when employed to classify a stream of data that

Lawal, I.A., and Abdulkarim, S.A. (2017). Adaptive SVM for Data Stream Classification. *South African Computer Journal 29*(1), 27–42. https://doi.org/10.18489/sacj.v29i1.414

SACJ is a publication of the South African Institute of Computer Scientists and Information Technologists. ISSN 1015-7999 (print) ISSN 2313-7835 (online).

are not well covered during the learning phase (Krempl et al., 2014). Second, the characteristics (e.g. underlying data distribution) of the data stream might change over time such that the classifier built on old data becomes inconsistent with new data, a condition commonly known as *concept drift* (Gama, Žliobaitė, Bifet, Pechenizkiy, & Bouchachia, 2014).

A possible strategy to handle these problems is to build the classifier on-line, learning incrementally from any new data generated during use. In this way, a temporally adaptive classifier that can accurately classify the evolving streams of data can be achieved. Recently, researchers have proposed techniques to facilitate learning a classifier in situations where all the training data are not available in advance. Cauwenberghs and Poggio (2001) proposed the first incremental algorithm for updating an existing SVM whenever new samples are acquired. The algorithm provides an efficient method for incorporating the information contained in new data into an existing classifier model in an on-line manner. It avoids re-estimating all the parameters of the model from scratch each time new data are available, rather it adapts the classifier to the changes imposed by the addition of the new information. Also, the algorithm incorporates an *unlearning* scheme which allows selective discarding of patterns considered less informative from the model without reducing the quality of the classifier. This is particularly important in handling concept drift issue because it allows the removal of obsolete patterns from an existing classifier during model revision. The incremental training allows incorporating additional training data into an existing SVM solution, which however, does not conclude the SVM learning process. The learning process has a training phase for selecting set of parameters and a model selection phase for tuning additional variables (called 'hyper-parameters') that finds classifier characterised by optimal performance in classifying previously unseen data (Shawe-Taylor & Sun, 2011). Unfortunately investigations in incremental SVM learning algorithms have largely focused on the training phase as opposed to the larger model selection.

This paper aim to build an adaptive SVM for data stream classification. We chose SVM algorithm for the following reasons: firstly, it provides a unique solution since its optimality problem is convex (no local minima) (Burges & Crisp, 1999), unlike other learning algorithms such as Artificial Neural Networks which have multiple solutions associated with local minima (Rocha, Cortez, & Neves, 2007) and therefore may not be accurate for the classification of the data stream. Moreover, with an appropriate kernel, SVM can work effectively even if the data are not linearly separable in the input space (Kapp, Sabourin, & Maupin, 2012). Lastly, SVM provides a compact representation of historical data in the form of *support vectors*, which allow the algorithm to be easily extended to fit into an incremental learning framework. We adopted the incremental training technique in order to train the SVM with continuous streams of labeled data. We then introduced incremental model selection for the SVM in order to guarantee the effectiveness of the classifier over time. We study the performance of our proposed method in terms of accuracy and training time on real-world applications including spam filtering and human action classification in videos.

2 SVM FOR DATA CLASSIFICATION

SVM (Cortes & Vapnik, 1995) is an effective machine learning approach for solving classification problems. An SVM is train to find within a given labeled training input data, sets of points known

https://doi.org/10.18489/sacj.v29i1.414

as the support vectors. These support vectors define some separating hyper planes (also known as the decision boundaries) that partition the labeled training data into a pre-defined number of classes (Cortes & Vapnik, 1995). Model selection in SVM, on the other hand, involves searching for the optimal SVM hyper-parameters that allows us to build the SVM characterised by optimal performance (i.e. low misclassification error) on previously unseen data (Kapp et al., 2012). For linearly separable training data, for example, the only hyper-parameter is C, the regularization parameter. However, when dealing with non-linearly separable training data, at least one additional variable is introduced into the set of hyper-parameters: this could be the width of the Gaussian γ or the order of the polynomial p of the specific kernel function used (Kapp et al., 2012). The subsequent discussion describes the proposed SVM learning process for the classification of data stream.

2.1 Incremental training of SVM

Given a stream of labeled training data $D^t = \left\{(\boldsymbol{x}_1^t, y_1^t), \ldots, (\boldsymbol{x}_n^t, y_n^t)\right\}$, where $\boldsymbol{x}_i^t \in \Re^d$ is the i^{th} feature vector representing a data sample at time t, $y_i^t = \pm 1$ is the sample class label. At $t = 1$, we aim to exploit D^1 to learn the initial SVM f^1 that can generalize well with previously unseen data and then incrementally update the classifier when new labeled training data are acquired at $t > 1$. The f^1 is defined as:

$$f^1(\boldsymbol{x}^1) = sign\left[\boldsymbol{w^1}^T \boldsymbol{x}^1 + b^1\right] \tag{1}$$

where $b^1 \in \Re$ is the bias and $\boldsymbol{w^1} \in \Re^d$ are the weights at $t = 1$. f^1 is a linear SVM with b^1 and w^1 values that give the optimal separation of the $\boldsymbol{x}^1$ in D^1. The values of b^1 and w^1 are obtained by solving the following Convex Constrained Quadratic Programming (CCQP) problem (Cortes & Vapnik, 1995):

$$\begin{aligned} \min_{\boldsymbol{w^1}, b^1, \xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ & y_i^1\left[(\boldsymbol{w^1}^T \boldsymbol{x}_i^1) + b^t\right] \geq 1 - \xi_i \quad \forall i \in \{1, \ldots, n\} \\ & \xi_i \geq 0 \quad \forall i \in \{1, \ldots, n\}, \end{aligned} \tag{2}$$

where ξ_i are slack variables for penalising misclassification errors and C is a regularization parameter (Cortes & Vapnik, 1995). The problem of Eq. 2 can be solved more easily in its Wolfe dual formulation by using the Lagrange multipliers. Two sets of multipliers are used, $\alpha^1 \in \Re^n$ and $\mu^1 \in \Re^n$, one for each of the two constraints in Eq. 2, respectively. We transform Eq. 2 to

$$\begin{aligned} L(\boldsymbol{w^1}, b^1, \xi) &= \frac{1}{2}\left\|\boldsymbol{w^1}\right\|^2 + C\sum_{i=1}^{n}\xi_i \\ &+ \sum_{i=1}^{n}\alpha_i^1(y_i^1\left[(\boldsymbol{w^1}^T \boldsymbol{x}_i^1) + b^t\right] - 1 + \xi_i) + \sum_{i=1}^{n}\mu_i^t\xi_i. \end{aligned} \tag{3}$$

https://doi.org/10.18489/sacj.v29i1.414

By taking the partial derivatives of L with respect to w^1, b^1 and ξ_i, we obtain the Karush-Kuhn-Tucker (KKT) optimality conditions for the Wolfe dual formulation as (Cortes & Vapnik, 1995)

$$\frac{\partial L}{\partial w_i^1} = 0 \rightarrow w_i^1 = \sum_{i=1}^{n} \alpha_i^1 y_i^1 \boldsymbol{x}_i^1, \quad \forall i = 1, \ldots, n \tag{4}$$

$$\frac{\partial L}{\partial b^1} = 0 \rightarrow \sum_{i=1}^{n} \alpha_i^1 y_i^1 = 0 \tag{5}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \rightarrow C - \alpha_i^1 - \mu_i^t = 0, \quad \forall i = 1, \ldots, n \tag{6}$$

$$\alpha_i^1 (y_i^1 \left[(\boldsymbol{w}^T \boldsymbol{x}_i^1) + b \right] - 1 + \xi_i^1) = 0, \quad \forall i = 1, \ldots, n \tag{7}$$

$$\mu_i^t \xi_i = 0, \quad \forall i = 1, \ldots, n \tag{8}$$

$$(C - \alpha_i^1) \xi_i = 0, \quad \forall i = 1, \ldots, n \tag{9}$$

$$\alpha_i^1 \mu_i^1 \xi_i = 0, \quad \forall i = 1, \ldots, n, \tag{10}$$

which allows us to substitute the w^1, b^1 and ξ_i in Eq. 2 in order to obtain the final Wolfe dual formulation as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i^1 \alpha_j^1 Q_{ij}^1 - \sum_{i=1}^{n} \alpha_i^1 \\ & 0 \leq \alpha_i^1 \leq C \quad \forall i \in \{1, \ldots, n\} \\ & \sum_{i=1}^{n} y_i^1 \alpha_i^1 = 0, \end{aligned} \tag{11}$$

where $Q_{ij} = y_i^1 y_j^1 K(\boldsymbol{x}_i^1, \boldsymbol{x}_j^1)$ and $K(.,.)$ is a Mercer's kernel function, which allows non-linear mappings of the training data (Cortes & Vapnik, 1995). We obtain the solution of Eq. 11 i.e. $\alpha_i^1 \ \forall i$ by using the efficient CCQP solvers already developed in the literature (Hsieh, Si, & Dhillon, 2014). Finally, we estimate the SVM by using the α_i^1 as

$$f^1(\boldsymbol{x}^1) = sign \left[\sum_{i=1}^{n} y_i^1 \alpha_i^1 K(\boldsymbol{x}^t, \boldsymbol{x}_i^1) + b^1 \right]. \tag{12}$$

From Eq. 12 we can partition D^1 into three sets: the set S_v^1 of margin support vectors with $y_i^1 f^1(\boldsymbol{x}_i^1) = 1$ and $\alpha_i^1 \in = [0, C]$; the set B_v^1 of bounded support vectors with $y_i^1 f^1(\boldsymbol{x}_i^1) < 1$ and $\alpha_i^1 = C$; the set I_v^1 of correctly classified training samples, with $y_i^1 f^1(\boldsymbol{x}_i^1) > 1$ and $\alpha_i^1 = 0$.

At time $t+1$ when a new training data $(\boldsymbol{x}_k^{t+1}, y_k^{t+1})$ is received, we want to update the SVM (Eq. 12) without having to re-build it from scratch. We employed an incremental training approach (Cauwenberghs & Poggio, 2001) which allows us to efficiently update the values of α_i^t and b^t in Eq. 12. Firstly, $z = y_k^{t+1} f^t(\boldsymbol{x}_k^{t+1})$ is computed in order to check whether $\boldsymbol{x}_k^{t+1}$ is correctly classified, i.e. if $\boldsymbol{x}_k^{t+1}$ belongs to the set I_v (i.e. z > 1). We initialize incremental training only when $(\boldsymbol{x}_k^{t+1}, y_k^{t+1})$

has the potential to become a support vector, i.e. z ≤ 1. In this case we set the α_k^{t+1} to 0 and perturb the SVM by gradually increasing the value of α_k^{t+1} to its largest possible value ($\alpha_k^{t+1} > 0$) until the optimal SVM solution is obtained without violating the KKT optimality conditions Cauwenberghs and Poggio, 2001. During the perturbation of the SVM the α_i^t $\forall i \in S_v^t$, and the bias b^t are also adjusted in order to maintain the KKT conditions. Alg. 1 summarizes the steps for the incremental SVM training.

Algorithm 1 Incremental training of SVM (Cauwenberghs & Poggio, 2001)

Input: $(\boldsymbol{x}_k^t, y_k^t)$: labeled training sample at time t, f^{t-1}: SVM at $t-1$
Output: f^t : updated SVM
Definitions: α_i^t: coefficient of the i^{th} sample at t, S_v^t: margin support vector set, B_v^t: Bounded support vector set, I_v^t: correctly classified training data set, Q: kernel matrix

1: *Begin:*
2: Compute $z = y_k^t f^{t-1}(\boldsymbol{x}_k^t)$,
3: If $z > 1$, go to step 9, because $(\boldsymbol{x}_k^t, y_k^t)$ cannot improve the classifier
4: Initialization: $\alpha_k^t \leftarrow 0$
5: Compute Q_{ik} for $\forall$ i$\in S_v^t$
6: If $z \leq 1$,
7: Increment α_k^t to its largest value while ensuring the KKT optimality conditions are satisfied
8: Check if one of the following conditions occurs:
I. If $y_k^t f^t(\boldsymbol{x}_k^t) = 1$, then $S_v^t \leftarrow (\boldsymbol{x}_k^t, y_k^t)$
II. Else if $\alpha_k^t = C$: then $B_v^t \leftarrow (\boldsymbol{x}_k^t, y_k^t)$
III. Else if $(\boldsymbol{x}_i^t, y_i^t)$, i $\in S_v^t$, become part of B_v^t or I_v, adjust α_i^t, i $\in S_v^t$, and b^t accordingly to keep the KKT conditions satisfied
9: Return f^t
10: *End*

2.2 Incremental model selection for SVM

During the incremental SVM update, we aim to revise both the regularization parameter C and the function f^t by exploiting newly gathered labeled training data. In the incremental SVM training framework, two approaches for model selection have been identified, namely: No Model Selection (NO-MS) (Zheng, Shen, Fan, & Zhao, 2013) and Complete Model Selection (C-MS) (Wang, 2008). In the NO-MS method typically the D^t collected at $t = 1$ is used to perform the model selection and to learn the initial SVM. The model selection is done using for example k-fold Cross Validation (k-CV), Leave-One-Out (LOO) or Bootstraps (Duarte & Wainer, 2017), and the optimal hyper-parameters are kept constant throughout the subsequent incremental learning processes. Thus when new sets of training data are gathered at time $t > 1$, only the set of support vectors of the classifier is revised accordingly, using incremental training method (Zheng et al., 2013). As the hyper-parameters of the SVM are not updated in this process, the NO-MS SVM performance may degrade over time. A more desirable approach is the C-MS which updates both the hyper-parameters and the support vectors of

the SVM whenever a new set of training data is available. This method is the most accurate, but it is computationally expensive as it requires performing a complete training and model selection process from scratch every time new sets of training data are collected.

We propose a new approach, referred to as Incremental Learning Model Selection (IL-MS), where MS is not completely neglected in the incremental learning, as in NO-MS, but a whole re-learning is avoided, unlike to C-MS. We start by identifying the best C and classifier f^t at the $t = 1$ step with a k-CV procedure. The k-CV procedure is enumerated as follows:

1. Re-sample the n training samples in D^t in a random order
2. Divide the re-sampled D^t into k folds (k chunks of approximately $\frac{n}{k}$ samples each)
3. Define a regularization parameter set $\{C_1, \ldots, C_m\}$
4. For i $= 1, \ldots, m$
5. For j $= 1, \ldots, k$
6. Train a classifier f_j^t using C_i and the samples that do not belong to fold j (i.e. k-CV training set)
7. Evaluate the trained f_j^t with all the samples in fold j (i.e. k-CV validation set)
8. Compute the number of training samples, e_j, in fold j that were misclassified by f_j^t
9. Estimate the total misclassification error E_i for each C_i as

$$E_i = \frac{\sum_{j=1}^{k} e_j}{n} \tag{13}$$

10. Select the $C_*^t = C_i$ that give the lowest E_i
11. Learn the final classifier f^t using C_*^t and $\bar{D}^t$.

We keep the k-CV training set and validation set, and also all the classifiers $f_{(1,\ldots,k)}^t$ trained while applying k-CV for future use. Thus, when new streams of training data $\left\{(\boldsymbol{x}_1^{t+1}, y_1^{t+1}), \ldots, (\boldsymbol{x}_i^{t+1}, y_i^{t+1})\right\}$ are acquired at time $t+1$, we modify C_*^t and the classifier f^t: it is reasonable to assume that, since the time step for which the new streams of data are collected is not large (i.e. it is not comparable to t), the regularization parameter will not vary too much from the previous best value. Thus, we define a neighborhood set centered around C_*^t as $\left[\frac{C_*^t}{\epsilon}, \epsilon C_*^t\right]$, where $\epsilon > 1$. We update both the k-CV training set and validation set and, accordingly, the $f_{1,\ldots,k}^{t+1}$ by using a regularization parameter optimization technique (Diehl & Cauwenberghs, 2003) for every value of $C \in \left[\frac{C_*^t}{\epsilon}, \epsilon C_*^t\right]$. We thus identify the best regularization parameter configuration C_*^{t+1}. Finally, the f^{t+1} is updated according to C_*^{t+1} and the new streams of training data collected using Alg. 1. The supplementary computational burden with respect to NO-MS is limited: in fact, we have to update $k+1$ classifiers instead of 1 and we have to perform a k-CV at each updating step, but limited to a restricted neighborhood set.

https://doi.org/10.18489/sacj.v29i1.414

3 EXPERIMENTS AND EVALUATION

We assess the effectiveness of the proposed IL-MS SVM by applying it to spam Email Classification (SEC) and Human Action Classification (HAC) in video. We chose the SEC because it represents a data stream problem with a gradual concept drift issue where the characteristic of spam email changes over time and the spam filter (classifier) needs to be continuously revised in order to ensure its effectiveness. We chose the HAC application as a potential on-line data classification problem whereby training data (i.e. stream of human action video clips) are acquired over time. Therefore the human actions in the videos need to be classified on the fly and the classifier also needs to be incrementally revised with any new action video recorded during use in order to maintain its effectiveness. We compare the results of the proposed IL-MS SVM with that of NO-MS SVM and C-MS SVM based on their error rate in classifying previously unseen email and human action videos. We also compare the time taken to incorporate the information contained in a newly gathered email stream and also the time taken to incorporate the information contained in a newly obtained human action videos. In the following sections we discuss the datasets, the experimental setup and analysis of the results.

3.1 Dataset

3.1.1 Spam Email Dataset

The spam email dataset (Katakis, Tsoumakas, & Vlahavas, 2010) consists of two classes: legitimate emails and spam. There are 9324 emails in the dataset which are arranged in a chronological order as a collection of streams of data over time. The spam ratio is approximately 20%. In order to be useful for learning a classifier, the emails are represented using the boolean bag-of-words model, where the (frequency of) occurrence of words in the email is used as feature attributes. The cardinality of a feature vector representing each email is 40,000.

3.1.2 KTH Action Video Dataset

The KTH action video dataset (Schüldt, Laptev, & Caputo, 2004) contains 6 types (classes) of actions which include boxing, hand clapping, hand waving, jogging, running and walking performed by 25 persons in 4 different scenarios (i.e. outdoor, indoor, variation in scale and changes in clothing). Each video clip is 25fps and contains only one person performing a single action. There is a total of 600 video clips for all combinations of 25 individuals, 6 actions, and 4 scenarios. This dataset, unlike the spam Email dataset, represents a multi-class data stream classification problem.We organised the video clips in sets of 24 videos per set, such that each set contains clips of the same actions that are, however, performed differently. We use the toolbox developed by Dollar, Rabaud, Cottrell, and Belongie (2005) to detect interest points and extract spatio-temporal feature vectors from each video clip to represent the human action. Figure 1 shows some examples of the KTH dataset.

https://doi.org/10.18489/sacj.v29i1.414

Figure 1: Some frames from the KTH action dataset, from top left to right: boxing, hand waving, running, clapping, jogging, and walking

3.2 Experimental setup

For the SEC experiment, we use an initial set of feature vectors representing 1324 emails to learn the first IL-MS SVM, and then we used the Interleaved Test-Then-Train evaluation method to perform incremental learning by adding feature vectors of 320 emails per iteration. Because the dimension of the feature vector for each email is far greater than the number of emails in the dataset, we develop a linear SVM to separate the legitimate email from spam. We search for the C in the range $[10^{-5}, 10^{3}]$ by using 50 points equally spaced in logarithmic scale, while we choose $k = 4$ for the k-CV procedure. We also set $\epsilon = 10$ in IL-MS.

In the case of the HAC, we use the feature vectors from a set of 100 video clips to learn the initial IL-MS SVM, and then we used the Interleaved Test-Then-Train evaluation method to perform incremental learning by adding feature vectors extracted from 24 video clips per iteration. Because the feature vectors are not linearly separable in the input space, we employ a non-linear SVM with a Gaussian kernel for the classification. The initial optimal SVM hyper-parameter set C, γ is found using an exhaustive grid search in the range $[10^{-4}, 10^{3}]$ by using 50 points equally spaced in logarithmic scale, while we fix $k = 4$ for the k-CV procedure. We also use different values of ϵ from $[10^{1}, \ldots, 10^{3}]$ in IL-MS in order to study the effect of varying ϵ on the quality of our results (both in terms of misclassification rate and training time). Given that action classification is a multi-class problem, we study some schemes used to extend the SVM to allow multi-class classification. In the literature, two popular approaches have been reported (Doğan, Glasmachers, & Igel, 2016). The first approach, known as One-Vs-One (OVO), involves training one SVM per pair of classes, which can be formulated

Table 1: Specifications of the PC used for conducting our experiments

Component	Description
Processor	Intel Core i5 @ 2.50GHz
Memory	6GB RAM
Operating system	Microsoft Windows 7

Table 2: Training times for the different SVMs on the spam email dataset. The NO-MS SVM has the lowest training time followed by IL-MS SVM, while C-MS SVM has worst training time.

Method	Training Time (seconds)
NO–MS	0.90 ± 0.35
C–MS	45.97 ± 1.51
IL–MS	4.36 ± 0.45

by distinguishing, in turn, one class from each of the other classes. Let m be the number of classes; the total number of trained SVMs equals $m(m-1)/2$. Alternatively, it is possible to learn m SVMs, where each classifier learns to separate features of a class from the features belonging to the other classes: this approach is known as One-Vs-All (OVA). We adopt the OVA method since it requires less computation and its accuracy is comparable with OVO (Doğan et al., 2016). Our proposed approach was implemented in Matlab and all the experiments were conducted on a PC with the specifications shown in Table 1.

3.3 Discussion of results

3.3.1 Spam Email Classification

Figure 2 shows the error rate on the evaluation set. The proposed IL-MS (Figure 2 (blue)) and the C-MS (Figure 2 (green)) SVMs produce better classification results with an average percentage classification error of 4.75 ± 1.49 and 4.67 ± 1.48, respectively. The non-adaptive NO-MS SVM (Figure 2 (red)), on the other hand, performed poorly with an average percentage classification error of 7.89 ± 1.04 (see Figure 3). Thanks to the introduced model selection, the IL-MS SVM is able to maintain it effectiveness (low error rate) over time. Moreover, as shown in Table 2, the training time for the IL-MS SVM is 4.36 ± 0.45 seconds, while that of C-MS SVM is 45.97 ± 1.51 seconds. The training time for IL-MS SVM (thanks to the incorporated incremental technique) is less than that of C-MS SVM but higher than NO-MS SVM which does not do model selection. These results clearly show that the proposed IL-MS SVM represents a good trade-off between accuracy and training time. While the accuracy is comparable to the one of C-MS SVM, the training time is more than one order of magnitude smaller than the time needed by C-MS SVM and acceptable for on-line SEC.

https://doi.org/10.18489/sacj.v29i1.414

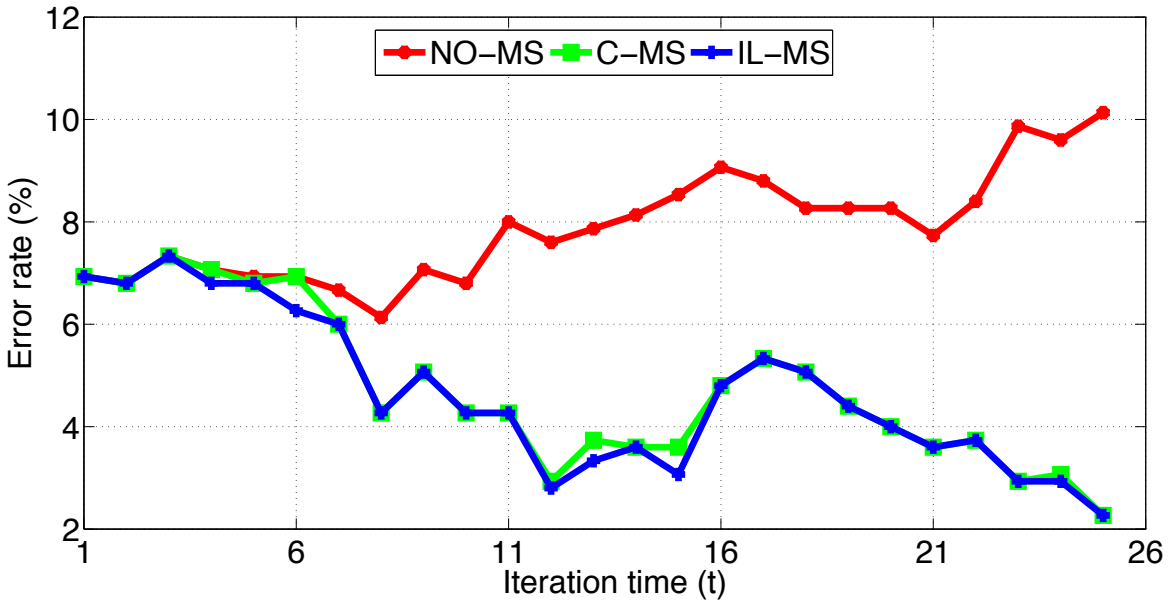

Figure 2: Comparison of misclassification error rates on the spam email dataset. The plots show that the C-MS (green) and IL-MS (blue) SVMs with low error rate outperform the NO-MS SVM classifier (red).

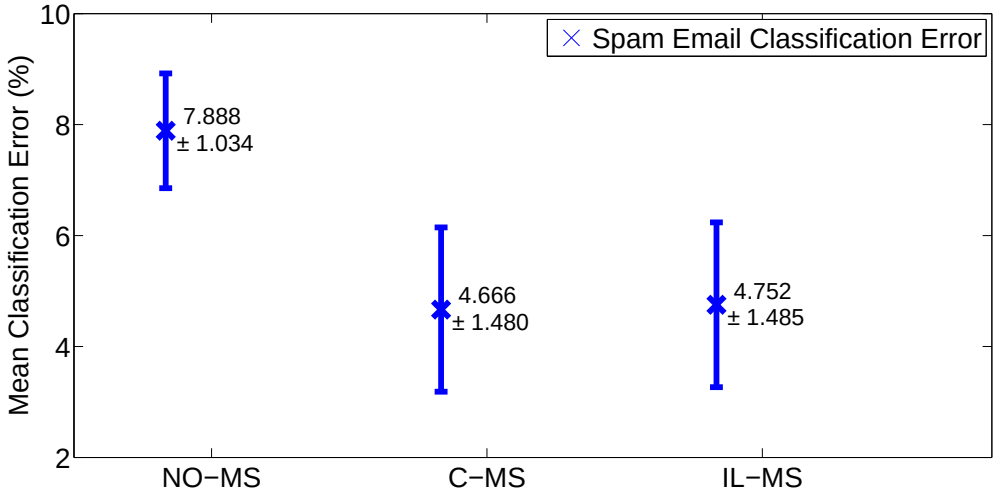

Figure 3: Comparison of the mean classification error on the spam email dataset.

https://doi.org/10.18489/sacj.v29i1.414

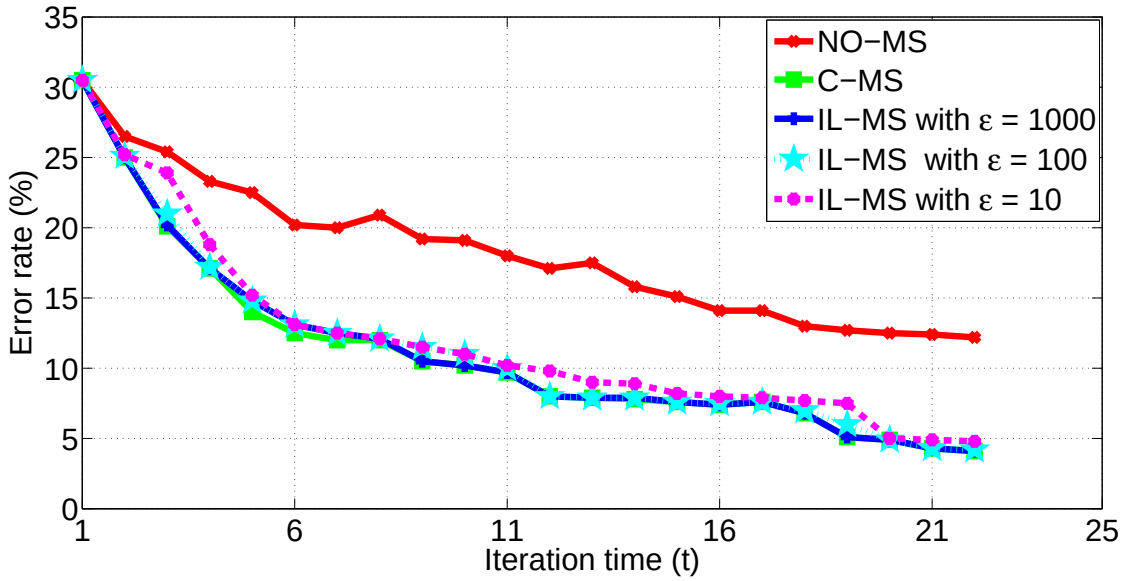

Figure 4: Comparison of error rates obtained by the SVMs trained using the IL-MS, C-MS and NO-MS learning methods on the KTH action videos over a period of time t. The C-MS SVM (green) and IL-MS SVM (blue, cyan and purple) outperform (i.e. lower error rates) the NO-MS SVM (red), because, unlike the NO-MS SVM, the C-MS SVM and IL-MS SVM maintain their effectiveness by performing model selection every time new sets of training data are exploited.

3.3.2 Human Action Classification

Figure 4 shows the result of the three SVMs on the KTH action dataset. The error rate plots for all the three SVMs indicate a decreasing error trend; this is because as more information from the video streams are learned incrementally the SVMs are able to discriminate among the different action classes present in the video. Moreover, it can be seen from Figure 5, that overall, the C-MS SVM and IL-MS SVM produce lower error with an average classification error of 11.24 ± 6.63 and 11.14 ± 6.06, respectively, while the NO-MS SVM achieved 18.28 ± 4.96. This improvement is due to the fact that the IL-MS C-MS methods are able to re-tune the SVM hyper-parameter configuration that guarantee accurate classification, as more training data are being exploited. Moreover, as shown in Table 3, the training time for the IL-MS SVM is 2.55 ± 0.09, while that of C-MS SVM is 28.03 ± 0.21. The training time for IL-MS SVM is less than that of C-MS SVM but higher than NO-MS SVM which does not do model selection.

Figure 6 shows the confusion matrix depicting the classification results of the IL-MS SVM on the KTH action test set for $\epsilon = 10$. Rows of the matrix represent the actual action class and columns the predicted action class. The diagonal entries (in bold) show the ratio of the number of test samples correctly classified for a given class of action to the total number of test samples belonging to that class. The average classification accuracy of the IL-MS SVM improved from 79.1% to 91.6% after 10 incremental learning steps.

https://doi.org/10.18489/sacj.v29i1.414

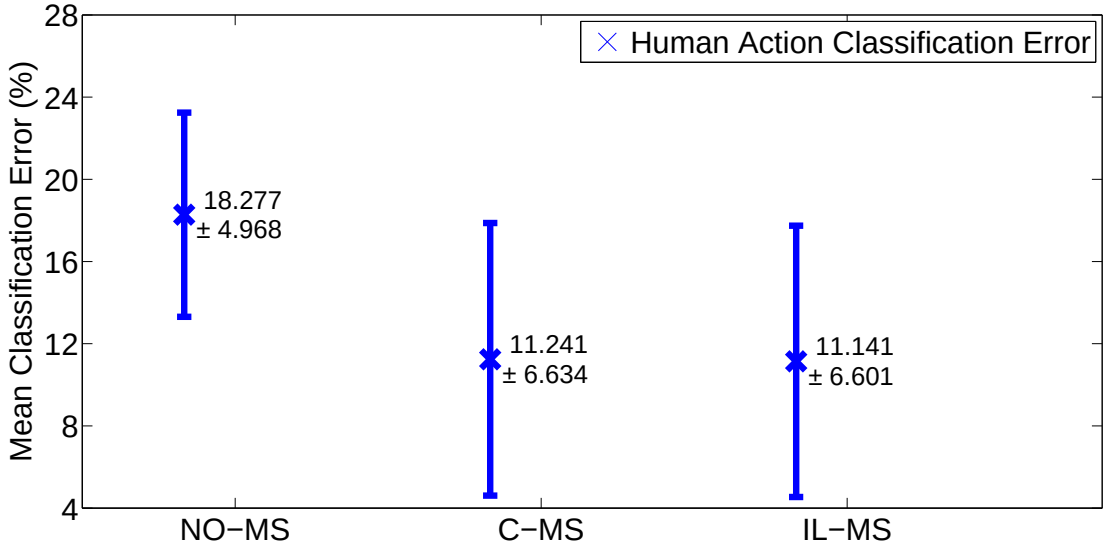

Figure 5: Comparison of the mean classification error on the KTH action dataset.

Table 3: Training times for the SVMs on the KTH dataset. The NO-MS SVM has the best training time followed by IL-MS SVM, while C-MS SVM has the worst time.

Method		**Training Time (seconds)**
NO–MS		0.23 ± 0.02
C–MS		28.03 ± 0.21
IL-MS	$\epsilon = 10$	2.55 ± 0.09
	$\epsilon = 100$	7.49 ± 0.27
	$\epsilon = 1000$	14.71 ± 0.31

https://doi.org/10.18489/sacj.v29i1.414

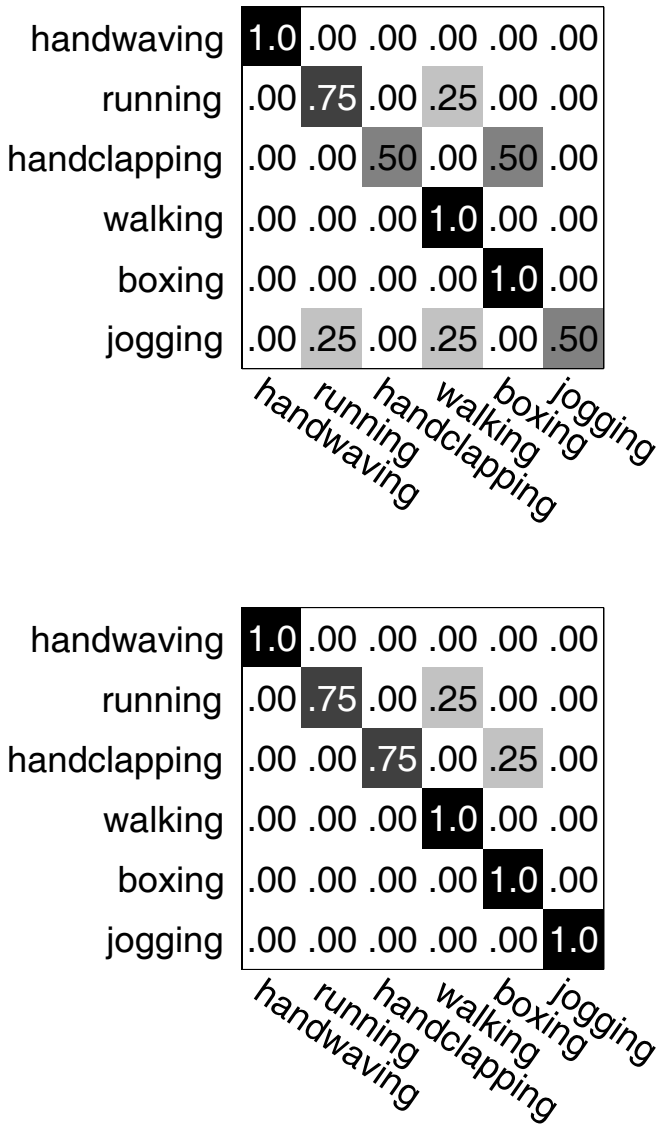

Figure 6: Confusion matrix of the classification results on the KTH test data using the IL-MS method with $\epsilon = 10$. The matrix on the top shows the classifier output at $t = 2$ while the matrix on the bottom gives the classifier output at $t = 12$. Rows of the matrix represent the actual action class and columns the predicted action class. The diagonal entries (in bold) show the ratio of the number of test samples correctly recognised for a given class of action to the total number of test samples belonging to that class.

https://doi.org/10.18489/sacj.v29i1.414

The matrix also indicates some misclassification mainly in the closely related actions such as running and jogging, etc. As there is no clear separation between these two action classes; identifying when running becomes slow enough to be classified as jogging is a subjective human process. Overall, the classification performance of the HAC improves significantly as more training video clips of human action become available and the information contained in them are incorporated into the SVM.

4 CONCLUSION

In this paper, we present a procedure for exploiting continuous streams of labeled data in order to develop an adaptive SVM for the classification of data streams. Specifically, we proposed an incremental learning-model selection (IL-MS) method for SVM, where we introduce the idea of incremental k-fold cross validation to allow the incremental tuning of the hyper-parameters of the SVM in order to guarantee the effectiveness of the SVM while exploiting newly acquired streams of training data. We evaluated the IL-MS approach quantitatively (in terms of misclassification error and training time) by applying it to the problem of on-line spam email filtering and human action classification in video streams. We compared the results of the proposed IL-MS method with two other baseline learning approaches for SVMs i.e. NO-MS and C-MS. The experimental results show that the proposed IL-MS SVM achieved a comparable performance with C-MS SVM in terms of classification accuracy, and also with NO-MS SVM in terms of training time. Thus we conclude that the IL-MS SVM represents an effective trade-off between the computationally infeasible C-MS SVM and the non-adaptive NO-MS SVM for on-line classification of data streams.

ACKNOWLEDGEMENTS

This work was completed in part with the computing resources provided by the AlJouf University, Saudi Arabia and Federal University Dutse, Nigeria.

References

Burges, C. J. C. & Crisp, D. J. (1999, November). Uniqueness of the SVM solution. In *Proc. of the Conference on Advances in Neural Information Processing Systems* (pp. 223–229).

Cauwenberghs, G. & Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Proc. of the International Conference on Advances in Neural Information Processing Systems* (pp. 409–415).

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Diehl, C. P. & Cauwenberghs, G. (2003). SVM incremental learning, adaptation and optimization. In *Proc. of the International Joint Conference on Neural Networks* (pp. 2685–2690). https://doi.org/10.1109/ijcnn.2003.1223991

Doğan, Ü., Glasmachers, T., & Igel, C. (2016). A unified view on multi-class support vector classification. *Journal of Machine Learning Research*, *17*(45), 1–32.

Dollar, P., Rabaud, V., Cottrell, G., & Belongie, S. (2005). Behavior recognition via sparse spatio-temporal features. In *Proc. of the Joint International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance* (pp. 65–72). https://doi.org/10.1109/vspets.2005.1570899

Dongre, P. B. & Malik, L. G. (2014, February). A review on real time data stream classification and adapting to various concept drift scenarios. In *Proc. of the International Advance Computing Conference* (pp. 533–537). https://doi.org/10.1109/iadcc.2014.6779381

Duarte, E. & Wainer, J. (2017, March). Empirical comparison of cross-validation and internal metrics for tuning SVM hyperparameters. *Pattern Recognition Letters*, *88*(1), 6–11. https://doi.org/10.1016/j.patrec.2017.01.007

Fong, S., Luo, Z., & Yap, B. W. (2013, August). Incremental learning algorithms for fast classification in data stream. In *Proc. of the International Symposium on Computational and Business Intelligence* (pp. 186–190). https://doi.org/10.1109/iscbi.2013.45

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), 1–37. https://doi.org/10.1145/2523813

Hsieh, C.-J., Si, S., & Dhillon, I. S. (2014). A divide-and-conquer solver for kernel support vector machines. In *Proc. of the International Conference on International Conference on Machine Learning* (pp. 566–574). Beijing, China.

Kapp, M. N., Sabourin, R., & Maupin, P. (2012). A dynamic model selection strategy for support vector machine classifiers. *Applied Soft Computing*, *12*(8), 2550–2565. https://doi.org/10.1016/j.asoc.2012.04.001

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems*, *22*(3), 371–391. https://doi.org/10.1007/s10115-009-0206-2

Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., … Stefanowski, J. (2014, September). Open challenges for data stream mining research. *ACM SIGKDD Explorations Newsletter - Special issue on big data*, *16*(1), 1–10.

Lu, Y., Boukharouba, K., Boonært, J., Fleury, A., & Lecuche, S. (2014). Application of an incremental SVM algorithm for on-line human recognition from video surveillance using texture and color features. *Neurocomputing*, *126*, 132–140. https://doi.org/10.1016/j.neucom.2012.08.071

Nguyen, H.-L., Woon, Y.-K., & Ng, W.-K. (2015). A survey on data stream clustering and classification. *Knowledge and Information Systems*, *45*(3), 535–569. https://doi.org/10.1007/s10115-014-0808-1

Rocha, M., Cortez, P., & Neves, J. (2007). Evolution of neural networks for classification and regression. *Neurocomputing*, *70*(16–18), 2809–2816. https://doi.org/10.1016/j.neucom.2006.05.023

Schüldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: a local svm approach. In *Proc. of the International Conference on Pattern Recognition* (pp. 32–36). https://doi.org/10.1109/icpr.2004.1334462

Shawe-Taylor, J. & Sun, S. (2011). A review of optimization methodologies in support vector machines. *Neurocomputing*, *74*(17), 3609–3618. https://doi.org/10.1016/j.neucom.2011.06.026

Vipin, N. S. & Nizar, M. A. (2014, December). A proposal for efficient on-line spam filtering. In *Proc. of the International Conference on Computational Systems and Communications* (pp. 323–327). https://doi.org/10.1109/compsc.2014.7032671

Wang, G. (2008, September). A survey on training algorithms for support vector machine classifiers. In *Proc. of the International Conference on Networked Computing and Advanced Information Management* (pp. 123–128). https://doi.org/10.1109/ncm.2008.103

Zheng, J., Shen, F., Fan, H., & Zhao, J. (2013). An online incremental learning support vector machine for large-scale data. *Neural Computing and Applications*, *22*(5), 1023–1035. https://doi.org/10.1007/s00521-011-0793-1