Incremental Learning with Support Vector Data Description

Weiyi Xie, Stefan Uhlmann, Serkan Kiranyaz, and Moncef Gabbouj
Signal Processing Department
Tampere University of Technology
Tampere, Finland
{weiyi.xie, stefan.uhlmann, serkan.kiranyaz, moncef.gabbouj} @tut.fi

***Abstract*— Due to the simplicity and firm mathematical foundation, Support Vector Machines (SVMs) have been intensively used to solve classification problems. However, training SVMs on real world large-scale databases is computationally costly and sometimes infeasible when the dataset size is massive and non-stationary. In this paper, we propose an incremental learning approach that greatly reduces the time consumption and memory usage for training SVMs. The proposed method is fully dynamic, which stores only a small fraction of previous training examples whereas the rest can be discarded. It can further handle unseen labels in new training batches. The classification experiments show that the proposed method achieves the same level of classification accuracy as batch learning while the computational cost is significantly reduced, and it can outperform other incremental SVM approaches for the new class problem.**

Keywords—Support Vector Machines; Large-scale; Incremental Learning; Classification

I. INTRODUCTION

Support Vector Machine (SVM) [1] has attracted a great deal of research attention due to its firm mathematical foundation and simplicity. In the most basic terms, SVM training is to construct an optimal hyper plane to separate two classes with maximal margin, resulting in solving a convex quadratic optimization problem (QP) in the size of the training set. Solving such QP thus can be intractable for a massive dataset. Researchers have attempted to solve this problem with incremental learning methods where a large dataset is divided into small trunks. Then these trunks are learned in incremental steps. Incremental learning can also handle the case when meaningful examples cannot be gathered *before* the learning process e.g. stream data. So far, existing SVM incremental learning approaches still remain problematic when a new batch of training examples contains unseen class labels and capability of discarding previous learned examples is required.

According to [2], an incremental learning method can be categorized into example-incremental, class-incremental, and attributes-incremental. At early stages of example-incremental learning studies with SVMs, [3] reserves only Support Vectors from previous training steps and discards the rest. When new training data is available, reserved Support Vectors are simply fused with the new training data. This method is referred to as SV-incremental learning in the literature [5], which is based on the fact that training a SVM over Support Vectors results in the same decision plane as training it over the entire training set. SV-incremental learning compares well to batch learning in terms of time consumption and classification accuracy [3], while it may cause large deviations from constructing the final true classifier if distribution properties of training examples vary dramatically during incremental learning , known as so called "concept drift" [4]. The issue becomes more evident in class-incremental learning scenario where new training data contains unseen labels. In this case, the problem of SV-incremental learning is that the Support Vectors suffice to describe the decision plane but not to describe the distribution of the underlying training examples [5] appropriately. Therefore, in essence, the problem is how to find a sufficient representation for the original training data.

Clustering based methods are firstly introduced to address this problem. CB-SVM [6] recursively forms a hierarchical clustering tree and then selects the centroids of the clusters as the representatives. Tseng and Chen [7] assumed that the examples residing on the boundaries of the clusters are critical data, thus only these examples are used for training. Support Cluster Machine (SCM) [8] shows that after clustering the original training set clusters can be trained as examples based on a the probability product kernel. However, clustering models can be also problematic, due to their parameter dependency or their convergence highly depends on data distribution and dimensionality. Besides clustering, Core Vector Machine (CVM) [11] forms a core set by solving the Minimal Enclosing Ball (MEB) problem. As an extension, the authors further developed a simplified version of CVM, which solves the MEB problem with a given radius [12]. After finding a core set for SVM training, QP can be directly solved using that core set instead of using the entire training data set. However, this method has the same drawback as SV-incremental learning as the core set in CVM is sufficient to find an approximate solution of the SVM QP problem but it does not suffice to represent the whole training set.

In this paper, instead of finding MEB as in [11], we aim at finding the boundary of the training dataset in the SVM kernel space. The training examples lying on the boundary can be then considered as a sufficient representation of training dataset as training linear classifiers such as SVM largely relies on the boundary training examples. Geometrical methods for finding exact boundaries such as Convex-hull [9] or Concaved-Hull [10] are not efficient for high-dimensional data. In an extreme case, some kernel functions such as Radial Basis Function Kernel (RBF) map input features into an infinite-dimensional

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.669

space, where extracting exact boundaries via geometrical methods becomes infeasible. Therefore, instead of finding the exact boundary, this paper proposes a method that finds a sufficient representation of it. Firstly, we partition training examples into subsets and each corresponds to one class label. Then we use the Support Vector Data Description [13, 14] to find the Minimal Enclosing Ball (MEB) for each subset. The examples lying on the found MEB, called the marginal vectors in this paper generate a preliminary representation of boundary geometry of the subset. Meanwhile, the proposed method trains binary SVM classifiers on subset pairs in *One-Versus-One* manner, and the generated support vectors are accumulated according to its class label. The accumulated support vectors achieve a dynamically-updating description of boundary geometry of the training subset along with more and more new classes involved in incremental process. Thus, we combine the accumulated support vectors with the marginal vectors to be the representation of training data boundary and this combination is further used to present previous training data in proposed incremental learning approach. The experiments carried out on benchmark data sets confirm the effectiveness of proposed approach on handling both example-incremental and class-incremental learning cases in terms of training memory, time consumption, and classification accuracy.

The rest of the paper is organized as follows: Section II details the existing techniques corresponding to the proposed method. Section III presents the proposed incremental learning approach for multi-class SVMs. In Section IV, we present our test results on real world databases. Section V concludes the paper and suggests topics for future research.

II. Related Techniques

A. Support Vector Machine basics

Training SVM is to find a hyper plane that separates the labeled training examples with maximum margin. Given labelled training examples $\{x_i,y_i\}$, $i = 1,\ldots,n$, $y_i \in \{-1,1\}$, $x \in R^d$, SVM training is formulated as solving a quadratic optimization problem:

$$\min_{w,b,\xi_i} \quad \frac{1}{2}\| w \|^2 + C\sum_{i=1}^{n} \xi_i \tag{1}$$

$$s.t \quad y_i(w \cdot x_i) + b \geq 1 - \xi_i, \xi_i \geq 0\,, \tag{2}$$

where ξ_i is the slack variable and C controls the trade-off between the slack variable penalty and the margin size. w is the normal vector to the hyper plane and the offset of the hyper plane from the origin is determined by $b/\|w\|$. Examples, which turn the inequality (2) to equality, are called Support Vectors (SVs). This method can be extended to a nonlinear case by transforming the input features into a high-dimensional space ϕ: $x_i \rightarrow \phi(x_i)$ via nonlinear kernel mapping:

$$K(x_1, x_2) = \phi(x_1)^T \cdot \phi(x_2)\,.$$

The primal problem (1) can be transformed into a dual problem by introducing Lagrange multipliers α_i, which can be written in kernel form:

$$\max_{a} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{3}$$

$$s.t. 0 \leq \alpha_i \leq C, \forall i, \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4}$$

The optimal solution of SVM primal problem is presented in the form of a weighted sum of SVs $w = \sum_{x_i \in SVs} \alpha_i y_i \phi(x_i)$.

B. Multiclass Support Vector Machine

SVM is inherently designed to solve binary classification problems. For multiclass problems, the most commonly used technique is "divide and conquer" in which a single multiclass problem is divided into binary pairs and then a SVM is trained for each pair. Such techniques are *One-versus-One* and *One-versus-Rest*. *One-versus-One* constructs one classifier per pair of classes, resulting in *N (N-1)/2* binary classifiers where *N* is the number of classes. *One-versus-Rest* fits one classifier per class (*N* binary classifiers to be trained) and for each classifier the class is fitted against all the other classes. There are comparative studies [15, 16], performing comparative evaluations between *One-versus-One* and *One-versus-Rest*. They demonstrated that the performance of these two methods is highly dependent on the database used and, therefore, it is hard to say which one outperforms the other in general. Henceforth, in this paper, we use *One-versus-One* considering it is usually faster than *One-versus-Rest* when the number of classes is high.

C. SVL-incremental learning

In this section, we describe SVL-incremental learning proposed by Stefan Rüping in [5] as a special case of SV-incremental learning method. This method emphasizes the importance of previous obtained Support Vectors as new training examples might drive the decision plane to their own distribution. To achieve this goal, a weighting factor L can be applied on the cost factor of previous Support Vectors in SVM optimization problem so as to make them more "costly". We shall call this approach as the SVL in the remainder of the paper, which can be formulated by adjusting Eq. (1) as,

$$\min_{w,b,\xi_i} \quad \frac{1}{2}\| w \|^2 + C\Big(\sum_{\xi_i \in NB} \xi_i + L \sum_{\xi_i \in SVs} \xi_i\Big)\,, \tag{5}$$

where *NB* indicates a set of new batch training examples and L is the weight given to cost factor of previous Support Vectors. Generally, L can be assigned as the ratio between the number of Support Vectors and number of training examples.

D. Support Vector Data Description

Support Vector Data Description (SVDD) is to find an optimal radius R for a hyper sphere $S(c,R)$ containing all n training examples (i.e., the MEB problem) where c is the sphere center. Given a kernel K with corresponding feature map ϕ, SVDD is to solve:

$$\min_{R,c,\xi_i} : R^2 + C\sum_{i=1}^{n} \xi_i \tag{6}$$

$$s.t \quad \left\|\phi(x_i) - c\right\|^2 \leq R^2 + \xi_i\,. \tag{7}$$

The corresponding dual form:

$$\max_{a} \quad \sum_{i=1}^{n} \alpha_i K(x_i, x_i) - \sum_{i,j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \qquad (8)$$

$$s.t \quad \sum_{i=1}^{n} a_i y_i = 0, 0 \le a_i \le C \cdot \qquad (9)$$

The optimal solution of this QP programing can be written as:

$$c = \sum_{i=1}^{n} \alpha_i \varphi(x_i), R = \sqrt{\alpha' diag(K) - \alpha' K \alpha}$$

where $\alpha=[\alpha_1,...,\alpha_n]'$ is a vector of the Lagrange multipliers and K is the kernel matrix $[K(x_i,x_j)]$. Training examples with corresponding $\alpha>0$ shall be called Marginal Vectors (MVs).

III. Proposed SVM Incremental Learning

In this section, we firstly present the overview of our incremental learning approach as shown in Fig.1. It consists of two major modules: *Training block* and *Fusion*, both of which are further described within the following subsections.

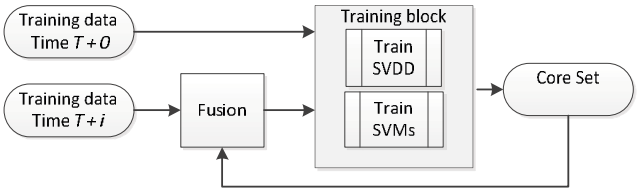

Fig.1 Overview of the proposed incremental learning.

A. Training block

There are two parallel sub-processes in this block. The sub-process, *Train SVMs* is designed to train binary SVM classifiers, from which we intend to find most informative examples to represent the boundaries of the previous data. Each binary classifier provides an optimal decision plane that separates two classes and a set of Support Vectors (SVs) for each class. Due to their intrinsic property, the SVs from each class of a binary problem actually define the boundary between this class and the other class examples with a maximum margin. This boundary can also be used to describe the boundary of the training examples. Meanwhile, based on the decision plane, we find the farthest examples from decision plane for each class as they can also be considered lying on the boundary of training examples. Searching for boundary examples are organized class wise so as to generate a set of Accumulated Support Vectors (ASVs) for each class. The detailed process to form sets of ASVs is described in Table 1. The second process in the *Training block* is *Train SVDD*, which solves the SVDD problem and further creates a Marginal Vectors (MVs) set per class. SVDD is trained with the same SVM parameters as used for training SVMs. The output of the Training block is a unique set that merges MVs with ASVs while removing duplicated examples. The unique set which we call the *Core set* in the remainder of this paper, is merged with the previous Core set, whenever exists, to ensure that examples in previous Core set are not discarded during the incremental learning.

TABLE 1 ACCUMULATING SUPPORT VECTORS

Algorithm: Accumulating Support Vectors

For every class i in training dataset ,let $ASVs_i = \emptyset$
  For every class $j \neq i$ in training dataset, DO:
    I. Train binary SVM classifier $P(i,j)$. $SV_{i,j}$ is a set of Support Vectors obtained after training $P(i,j)$.
    II. Add samples in $SV_{i,j}$ belonging to class i into $ASVs_i$
    III. According to decision plane of $P(i,j)$, find the farthest examples belonging to class i from decision plane. Add them into $ASVs_i$

B. Fusion for the Incremental Learning

The *Fusion* is to provide an input training set for *Training block* by simply fusing previous Core set, if exists, with new batch of training data. To improve training efficiency, The *Training block* will firstly examine the training examples in fused dataset if training is need according to their class labels. Training a binary SVM classifier $p(i,j)$ is not required when the new training batch does not contain the examples with class labels i or j. In this scenario, the Support Vectors obtained from training $p(i,j)$ at previous step are directly merged into new Core set. Similarly, training SVDD on a class that cannot be found in new batch is also unnecessary and previously obtained marginal vectors are merged into new Core set.

Besides training efficiency, The *Training block* shall manage to avoid the learning imbalance, e.g. if the size of the previous Core set is relatively small comparing to the size of the new batch training data, after fusing, SVMs and SVDD would learn mostly on the new batch. The main reason is both models are robust against outliers while in this case, the examples from previous Core set are seen as outliers. To overcome this issue, we shall give the previous examples higher impact than new batch of examples on training. This can be done in a similar way as for the SVL by adjusting weights for the cost factor. In detail, for training SVMs, we slightly modify Eq.(5) into:

$$\min_{w,b} \quad \frac{1}{2} \| w \|^2 + C\left(\sum_{i \in NB} \xi_i + L \sum_{i \in CoreSet} \xi_i\right) \cdot \qquad (10)$$

For training SVDD, we reformulate the primal problem of SVDD in Eq.(6) to:

$$\min_{R,c} : R^2 + C\left(\sum_{i \in NB} \xi_i + L \sum_{i \in Coreset} \xi_i\right) \qquad (11)$$

The weighting factor L can be calculated similarly to the weighted SVM approach, for instance as the percentage of the number of examples from the previous Core Set in previous training data set.

C. Interpretation of Marginal Vectors and Accumulated Support Vectors

SVDD aims to find a sphere-shaped decision boundary that separates training examples from one class from other class examples. Training SVDD generates the Marginal Vectors

(MVs) that are the examples "supporting" the sphere-shaped boundary. In this sense, MVs will be able to describe the boundaries of a given training set if training set forms a convex shape in the kernel space. Otherwise, MVs might be disadvantageous to represent the real boundaries. We illustrate this drawback of MVs in Fig.2, over a 2D toy example containing two classes, and the positive class does not form a convex set. Its MVs lose the boundaries on the concave part while binary SVMs decision hyper plane describe it significantly better. As a result, the Support Vectors (SVs) of positive class are located on the concave boundary. Thus, accumulating SVs could detect the inner sphere boundaries, which otherwise cannot be found by SVDD. We assume that with constantly new labels involving in incremental learning process, Accumulated Support Vectors alone in the end shall be able to describe the boundaries for a given class if the training examples with different labels could cover all regions surrounding that class in the projected space.

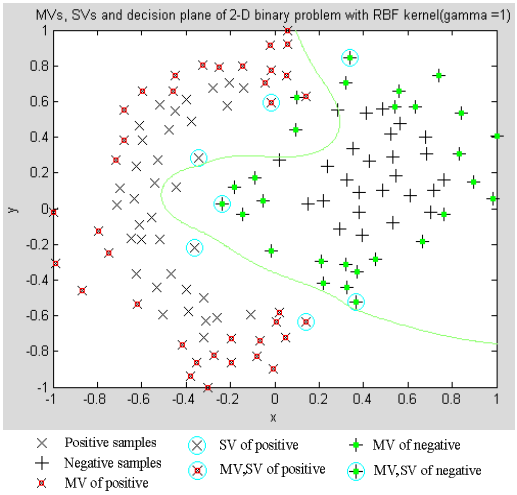

Fig.2 an example where Support Vectors compensate the deficiency of Marginal Vectors for describing boundaries.

IV. Experimental Results

A. Test Setups

We perform our incremental learning experiments on a PC with 3.0 GHz Intel Pentium CPU with 2GB RAM employing the software package LIBSVM (v3.10.) [17]. Three well-known multiclass benchmark databases *mnist*, *pendigits,* and *letter* are chosen to evaluate the classification performance of proposed incremental learning approach. For brevity, we select the RBF kernel function for training SVMs and SVDD. The SVM parameters C and γ are set to the one that produces the highest cross-validation (10-fold) classification accuracy from a uniform grid search. Table 2 shows database attributes and the chosen SVM parameters. Before presenting the experimental results, we introduce two important measurements: shrinking rate (SR) indicates the ratio of the number of examples stored after training to size of entire training dataset. Classification error (CE) denotes the classification error calculated on the test set. As a comparative baseline, batch learning results are presented in Table 3. The SR in batch learning results shows that SVs usually compose only a fraction of the entire training dataset. We can now compare the proposed method against the SVL-incremental learning in terms of classification accuracy, training time and memory consumption; and we shall further investigate the effect of presenting previous training data with its Marginal Vectors (MVs) set in the incremental learning. We shall call it MVL-incremental learning, where MVs are preserved to represent the data from the previous steps and previous MVs have a higher impact on forming decision hyper plane than new training data in further steps. To evaluate the performance of proposed method, we consider two test scenarios: example-incremental learning and class-incremental learning.

B. Example-incremental learning experiments

In this section, we investigate the case when the new batch of training examples only contains the known class labels. For justifying generalization capability of the proposed method, databases are partitioned in three different ways: the number of training examples per class is increasing gradually, the number of training examples per class is decreasing gradually, and the size of training set per class stays constant during the incremental learning. To achieve this, we apply three test settings such as "40%-30%-20%-10%", "10%-20%-30%-40%", "20%-20%-20%-20%-20%" where the percentages present the fraction of the training examples per step for each class. These settings are notated as *dec*, *inc*, and *even* in the test results, respectively. Results from this test are shown in Table 4 based on the average of 5 trials. Note that the statistics of the classification error (CE) are presented in the form of the mean + standard deviation.

TABLE 2 DATABASES ATTRIBUTES AND CORRESPONDING SVM PARAMETERS

	# of Class	# of data (train/test)	# of features	training parameter
mnist	10	60000 / 10000	778	$\gamma = 0.031$ $C = 64$
pendigits	10	7494 / 3498	16	$\gamma=1$ $C=16$
letter	26	15000 / 5000	16	$\gamma=0.125$ $C=16$

TABLE 3 TEST RESULTS OF BATCH LEARNING

	CE	training time(s)	SR
mnist	0.0143	1748.69	0.2753
pendigits	0.0214	4.503	0.2141
letter	0.0636	11.815	0.433

The results show that both SVL-incremental learning and the proposed method achieve a similar classification performance level in general. This is expected since SVL has shown to perform well in such setup and the proposed method provides additional information on top of SVs. Note further that they also achieve equal classification accuracies on the batch learning presented in Table 3 with much lower computational cost, i.e., training time and memory cost. The proposed method theoretically requires more computational time and memory than SVL for the obvious reason that training

both SVDD and SVMs would be more costly than training only SVMs. however, by training SVDD and SVMs in parallel. The training time consumption of the proposed method is drastically reduced to be a similar level with SVL and MVL. The larger memory requirement of the proposed method is still needed as it intends to find a proper presentation of previous training set while the others are not designed for it. For MVL incremental learning, as we showed in Chapter III section D, Marginal Vectors alone might fail to present the real boundaries of a given training dataset which is not convex, resulting in the degraded accuracy of MVL on the dataset, *letter*.

C. *Class-incremental learning experiments*

For class-incremental learning, databases are partitioned according to their different class labels beforehand. In this setup, the new batch of training data at each step contains examples from only unknown classes. Similar to example-incremental learning, we conduct the following three different test cases to add new classes at each step: the growing number of new classes (*inc*), the decreasing number of new classes (*dec*), and the number of new classes in new batch stay the same (*even*). In detail, for *10* classes datasets *mnist* and *pendigits*, the three test cases are: *inc*(2,3,5), *dec*(5,3,2), and *even*(2,1,1,1,1,1,1,1,1) where the number in parentheses denotes the number of new classes at each step. For database

TABLE 4 CLASSIFICATION PERFORMANCE OF EXAMPLE-INCREMENTAL LEARNING AMONG PROPOSED METHOD, SVL AND MVL. SHRINK RATE (SR) INDICATES THE RATIO OF THE NUMBER OF EXAMPLES STORED AFTER TRAINING TO SIZE OF ENTIRE TRAINING DATASET. CLASSIFICATION ERROR (CE) DENOTES THE CLASSIFICATION ERROR CALCULATED ON THE TEST SET. BEST CES ARE HIGHLIGHTED WITH BOLD FIGURE.

	inc			dec			even		
	CE	Time(s)	SR	CE	Time(s)	SR	CE	Time(s)	SR
mnist									
Proposed	**0.0142 +00013**	1286.92	0.4322	**0.0138 +0.0014**	1252.32	0.4573	**0.0142 +0.0017**	1227.5	0.4892
SVL	0.0158 +0.0021	1142.9	0.2457	0.0200 +0.0012	1117.7	0.2491	0.0152 +0.0036	1116.2	0.2506
MVL	0.0257 +0.0033	1262.12	0.322	0.0290 +0.0102	1294.7	0.313	0.0233 +0.0033	1345.7	0.3313
pendigit									
Proposed	**0.0214 +0.0004**	4.1420	0.4555	**0.0214 +0.0009**	6.0000	0.4415	0.0214 +0.0011	5.8280	0.4239
SVL	0.0215 +0.0002	4.8440	0.2064	0.0215 +0.0002	4.8440	0.2064	**0.0211 +0.0002**	4.7180	0.2057
MVL	0.0215 +0.0002	4.0000	0.4426	0.0228 +0.0005	5.8900	0.4279	0.0214 +0.0007	5.5150	0.4089
letter									
Proposed	0.0594 +0.0004	9.5170	0.6064	**0.0562 +0.0004**	9.8730	0.6068	**0.0574 +0.0012**	9.8950	0.5839
SVL	**0.0584 +0.0005**	8.3750	0.4036	0.0562 +0.0008	9.5100	0.3734	0.0578 +0.0006	9.9200	0.3835
MVL	0.0770 +0.0015	9.5930	0.3940	0.0770 +0.0021	9.5930	0.3940	0.0734 +0.0025	8.4230	0.3528

TABLE 5 CLASSIFICATION PERFORMANCE OF CLASS-INCREMENTAL LEARNING AMONG PROPOSED METHOD, SVL AND MVL.

	inc			dec			even		
	CE	Time(s)	SR	CE	Time(s)	SR	CE	Time(s)	SR
mnist									
Proposed	**0.0146 +00163**	932.03	0.5126	**0.0142 +0.005**	1041.2	0.5314	**0.0148 +0.0002**	1183.5	0.5124
SVL	0.0234 +0.0021	823.00	0.2457	0.0155 +0.0012	956.53	0.249	0.0297 +0.0036	1021.6	0.2506
MVL	0.0157 +0.0033	916.87	0.5126	0.0153 +0.0102	1029.9	0.5314	0.0167 +0.0002	1175.9	0.3313
pendigits									
Proposed	**0.0211 +0.0003**	2.6924	0.5196	**0.0210 +0.0000**	3.0108	0.5204	**0.0212 +0.0012**	3.2406	0.5178
SVL	0.0228 +0.0002	2.7960	0.1948	0.0231 +0.0002	2.5630	0.1978	0.0231 +0.0002	2.0000	0.1860
MVL	0.0214 +0.0002	2.5770	0.5138	0.0214 +0.0005	2.8440	0.5138	0.0214 +0.0007	2.4680	0.5138
letter									
Proposed	**0.0684 +0.0004**	3.8404	0.6315	**0.0633 +0.0014**	4.2823	0.6426	**0.0692 +0.0013**	5.1560	0.61147
SVL	0.1352 +0.0005	3.6090	0.3332	0.0710 +0.0018	3.1560	0.3581	0.1688 +0.0018	3.6890	0.2949
MVL	0.0816 +0.0011	3.8460	0.5000	0.0810 +0.0021	4.0630	0.5000	0.0904 +0.0012	4.3440	0.5000

letter containing *26* classes, we apply three cases as: *inc* (9,6,5,4,2) , *dec* (2,4,5,6,9), and *even* (2,1,1,…,1). All these classes are selected randomly into training process. Table 5 presents the mean and standard deviation of the CE on the test set, training time, and shrinking rate (SR) based on 5 trials.

The results show that in *pendigits* and *mnist*, three methods perform on the same level of the batch learning. However, they all require significantly less computational complexity in terms of training time and memory than batch learning. SVL method even has a slightly lower shrinking rate for *pendigits,* which means that at each step only fewer than 20% of entire training examples are involved in training. However, the significant accuracy drop of SVL on *letter* reveals that, using reserved previous SVs is not enough for incrementally learning new labels. The reason is that the large numbers of examples are discarded in each step of SVL learning, as they are redundant to update the decision planes. However, the discarded examples might indeed be very informative to describe the current distribution of the training data, which may be the potential SVs for training a classifier between the existing and new classes. Furthermore, there is a large deviation among the results of SVL incremental learning over three test cases. In the *dec* case, the large amount of classes involved at the beginning increases the chance to collect more informative examples as SVs, which may maintain the accuracy. On the contrary, in the *inc* case, fewer SVs are accumulated for the classes during the initial step, and this presents limited information in terms of distribution of their previous training set. Yet they are used as a substitute of the previous training set to train unknown classes, resulting in a noticeable accuracy drop. The root cause of the problem in the SVL methods is the lack of proper representation regarding the previous training set, while the proposed method and MVL aim to overcome this problem; and thus they both present similar classification accuracies among the different test cases. Comparing with each other, the proposed method outperforms MVL showing that the accumulated SVs are providing valuable information for overcoming the deficiency of presenting the real boundaries when the dataset is not convex in the kernel-projected space. In terms of memory consumption and training time, due to the parallelization, the proposed method achieves comparable performance to the other two, even twice faster than batch learning on *mnist*, while it spends more memory than SVL but evidently the extra memory consumption is necessary for a proper representation of the previous training data.

V. CONCLUSIONS

In this paper, we propose an incremental SVM learning method to deal with large real world databases. The experimental results approve that the proposed method can efficiently adapt to both example-incremental learning and class-incremental learning cases. The results further show that the proposed method is able to reduce time and memory consumption of SVM by around 50% than batch learning without a noticeable degradation on the classification accuracy. In contrast to SVL-incremental learning, the proposed method overcomes the inadequacy and instability of SVL method on class-incremental learning cases, and we have further shown that Accumulated Support Vectors (ASVs) and Marginal Vectors (MVs) put together can form a fine representation of the previous training data. The inferior classification performance of MVL confirms that MVs might generate incomplete boundaries when the dataset in projected space is not convex, which is solved by introducing ASVs into the proposed method. Moreover, The proposed approach may improve classification accuracy comparing corresponding results with the batch learning since weighting on previous data might help to alleviate the influence of outliers on new training batch. We shall pursue further investigations to find an even better representation of previous training data with lower cost. This will be the topic of our future research.

REFERENCES

[1] V.N. Vapnik, The Nature of Statistical Learning Theory. Springer-Verlag, 1995.

[2] Z.H. Zhou and Z.Q. Chen, "Hybrid Decision Tree," Knowledge-Based Systems, pp. 515-528, 2002.

[3] N. A. Syed, H. Liu, and K. K. Sung, "Incremental Learning with Support Vector Machines" in Proc. the Int.Joint.Conf.Artijicial Intelligence(IJCAI), 1999.

[4] R. Klinkenberg and T. Joachims, "Detecting concept drift with supportvector machines, " in Proc. 17th Int.Conf on Machine Learning(ICML), pp 487–494, 2000 .

[5] S. Ruping, "Incremental Learning with Support Vector Machines," in Proc. IEEE Int Conf. On Data Mining (ICDM), pp. 641-642, 2001.

[6] H. Yu, J. Yang and J. Han. "Classifying Large Data Sets Using SVMs with Hierarchical Clusters" in Proc, 9th ACM SIGKDD international Conference on Knowledge discovery and data mining,pp.306–315, 2003.

[7] S. Sun, C. Tseng, Y. Chen, S. Chuang and H. Fu. "Cluster-based Support Vector Machines in Text-Independent Speaker identification" in Proc. the Int'l Joint Conf.on Neural Network(IJCNN), 2004.

[8] B. Li, M. Chi, J. Fan, and X. Xue, "Support cluster machine,"in Proc. 24th Int. Conf on Maching Learning(ICML).,pp. 505–512, 2007.

[9] B. Valentina, "Survey of algorithms for the convex hull problem," Department of Computer Science, Oregon State University, 1999.

[10] A. Moreira and M.Y. Santos, "Concave hull:a k-nearest neighbors approach for the computation of the region occupied by a set of points," in Int Conf on Computer Graphics Theory and Applications(GRAPP), pp. 61–68, 2007.

[11] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines:Fast SVM training on very large data sets," *In Journal of Machine Learning Research*, pp. 363-392. 2005.

[12] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Simpler core vector machines with enclosing balls," in Proc. 24th Int Conf on Machine Learning(ICML), pp. 911–918, 2007.

[13] D. M. Tax and R. P. Duin, "Support vector domain description," *Pattern Recognition Letter*, vol. 20, nos. 11–13, pp. 1191–1199, 1999.

[14] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine Learning*, pp. 45–66, 2004.

[15] E. Allwein, R.E. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *Journal of Machine Learning Research,* pp. 113–141, 2000.

[16] C.W. Hsu and C.J. Lin, "A Comparison of Methods for MultiClass Support Vector Machines," IEEE Trans. Neural Networks, pp. 415-425, 2002.

[17] C.C. Chang and C.J. Lin, "LIBSVM : A library for support vector machines", ACM Transactions on Intelligent Systems and Technology, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm,2011