Contents lists available at ScienceDirect

Applied Soft Computing

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

Data selection based on decision tree for SVM classification on large data sets

Jair Cervantes[a,*], Farid García Lamont[a], Asdrúbal López-Chau[b], Lisbeth Rodríguez Mazahua[c], J. Sergio Ruíz[a]

[a] CU UAEM Texcoco, Av. Jardín Zumpango s/n, Fracc. El Tejocote, Texcoco, Mexico
[b] CU UAEM Zumpango, Camino viejo a Jilotzingo continuación calle Rayón, Valle Hermoso, Zumpango, Estado de México CP. 55600, Mexico
[c] Division of Research and Postgraduate Studies, Instituto Tecnológico de Orizaba, Av. Oriente 9, 852. Col Emiliano Zapata, C.P. 9432, Orizaba, Veracruz, Mexico

ARTICLE INFO

Article history:
Received 27 February 2013
Received in revised form
21 December 2014
Accepted 18 August 2015
Available online 25 September 2015

Keywords:
SVM
Classification
Large data sets

ABSTRACT

Support Vector Machine (SVM) has important properties such as a strong mathematical background and a better generalization capability with respect to other classification methods. On the other hand, the major drawback of SVM occurs in its training phase, which is computationally expensive and highly dependent on the size of input data set. In this study, a new algorithm to speed up the training time of SVM is presented; this method selects a small and representative amount of data from data sets to improve training time of SVM. The novel method uses an induction tree to reduce the training data set for SVM, producing a very fast and high-accuracy algorithm. According to the results, the proposed algorithm produces results with similar accuracy and in a faster way than the current SVM implementations.

1. Introduction

SVM was introduced by Vapnik as a kernel based machine learning model for classification and regression task. The extraordinary generalization capability of SVM and its discriminative power have attracted the attention of data mining, pattern recognition and machine learning communities in the last years. SVM has been used as a powerful tool for solving practical binary classification problems [1–5] and regression [6,7]. However, it is well known that the major drawback of SVM occurs in its training phase [8–10]. This is because in order to train this classifier, it is necessary to solve a quadratic programming problem or QPP, which is a computationally expensive task. Solving the QPP becomes impractical when the data sets are huge because the amount of time and memory invested is between $O(n^2)$ and $O(n^3)$. In order to show how long the training time of a SVM is, Fig. 1 presents a comparative between training times using three popular methods whose source code is publicly available, these are: Sequential Minimal Optimization (SMO) [11], Library of Support Vector Machines (LibSVM) [10] and Simple Support Vector Machine (SSVM) [12]. The data set used in this example contains one million points and eight features; in this case, LibSVM significantly outperforms SMO and SSVM.

In this research, we propose a novel method to reduce the size of data sets based on a decision tree (DT). The latter has several interesting properties: they are tolerant to noise, their training is not costly, they are able to partition the input space into regions with low entropy, and in addition, they produce models that humans can interpret easily. By taking advantage of the ability of DTs to model class distributions with the use of partitions, new instances can be assigned to the same class of partition they belong to. Using this feature, we can detect critical instances that determine the decision boundaries of SVM. With proposed method, SVM can be enabled on large data sets. According to the experimental results there is some minor damage in some data sets. However, computational comparisons on benchmark data sets show that proposed method reduces the training time significantly in comparison with other state of the art proposals.

DTs have been used for data reduction in some previous works. In [9], each disjoint region (partition) discovered by a DT is used to train a SVM. That method is based on the well-known fact that, in general, the region (hyperplane defined by the solution) found on small data sets are less elaborated than the region obtained by the entire training set [13–15]. Small learning data sets reduce decision tree complexity simplifying the decision rules. A SVM is applied to each one of these regions, so that the computational cost is less expensive compared with training a SVM and with the whole data set.

* Corresponding author. Tel.: +52 55 6028 6207.
E-mail address: jcervantesc@uaemex.mx (J. Cervantes).

http://dx.doi.org/10.1016/j.asoc.2015.08.048
1568-4946/© 2015 Elsevier B.V. All rights reserved.

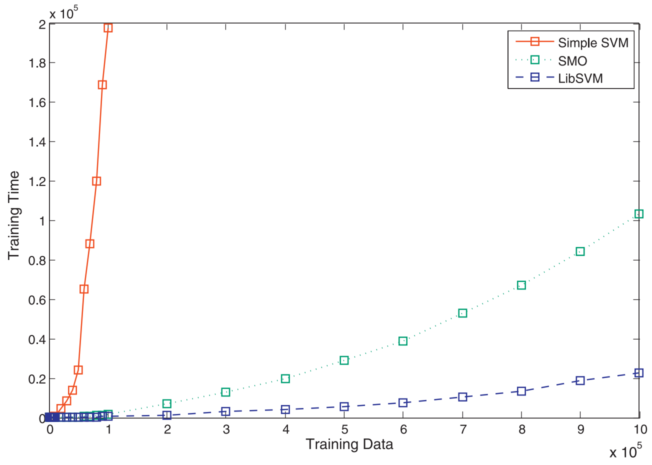

Fig. 1. Training times comparative of SVM implementations.

A similar approach to proposed method was recently presented in [16]; it consists in reducing the number of instances to train a SVM. The central idea was to approximate the decision boundary of SVM by using DT, i.e., to capture the objects near the decision boundary; however, the strategy is different to the used in [16]; a SVM is trained twice, the first time with a few examples randomly selected. The hyperplane obtained before is used to select examples close to the Support Vectors of separating hyperplane detected by a DT. The SVM is trained again, but this time with those examples selected by the DT. In [17,18], also DT is used to select examples close to decision boundary, however, the authors does not take in consideration how to optimize the sigma parameter, and the use of the algorithm on imbalanced data sets.

The method presented in this research is very efficient with large data sets. The proposed approach, firstly, eliminates nonessential data, and then it recovers the data points that are near to the decision boundaries. The training of SVM is done on those remained "useful" data. The methodology presented in this study applies a decision tree in a unique, yet powerful and effective way to reduce the SVM training time. It is important because in many applications on the real world is necessary a reasonable trade-off between accuracy and training time.

The rest of the study is organized in 7 sections. In Section 2, an overview on the methods to train SVM is presented. In Section 3, we review some preliminaries of SVM and DT. Section 4 presents the proposed method in detail. The complete experimental study is carried out in Section 5. A thorough discussion is presented in Section 6. Finally, Section 7 summarizes the work and draws conclusions from it.

2. Review on methods for training support vector machines

According to the strategy used, the training methods for SVM can be categorized into data selection, decomposition, geometric, parallel implementations and heuristics. Their core ideas and the most representative algorithms are presented in this section.

Data selection methods for SVM intent to decrease the size of data sets by removing the instances that do not contribute to the definition of the optimal separating hyperplane. The latter depends completely on instances which are located closest to the separation boundary [19], and correspond to those whose Lagrange multipliers are greater than zero in the Karush–Kuhn–Tucker conditions (1). These instances are called support vectors (SVs). Generally, the number of SV is a small portion compared with the size of training sets.

$$\begin{aligned} \alpha_i = 0 \Rightarrow y_i(\langle \omega, x_i \rangle + b) \geq 1 \quad &\text{and} \quad \xi_i = 0 \\ 0 < \alpha_i < C \Rightarrow y_i(\langle \omega, x_i \rangle + b) = 1 \quad &\text{and} \quad \xi_i = 0 \\ alpha_i = C \Rightarrow y_i(\langle \omega, x_i \rangle + b) \leq 1 \quad &\text{and} \quad \xi_i \geq 0 \end{aligned} \tag{1}$$

Simple random sampling (SRS) is probably the most basic strategy to reduce the size of training sets. It consists in choosing a number of instances and then train a SVM with them. The works presented in [20,21] and [22] show that uniform random sampling is the optimal robust selection scheme in terms of several statistical criteria. However, although SRS is computationally cheap, the standard deviation of classification accuracy is large in most cases [22].

A more sophisticated form of this type of sampling consists in assigning to each instance a probability to be chosen. Once a number of instances is randomly selected, a SVM is trained with them. After this, the probabilities are updated, increasing those whose instances have been miss-classified [23–25]. This process is repeated several times.

Some data selection methods have been developed by computing the distance between the instances and the optimal hyperplane. Several metrics for measuring distance have been used in previous works: Euclidean [26,27], Mahalanobis [28] and Hausdorff [29]. Most of the current distance-based methods are inspired on two observations: (1) the instances closest to those ones with opposite label have high chances to be SV [29] and (2) instances far from hyperplane do not contribute to the definition of decision boundary [30]. A problem with naive implementations that require to compute all distances between objects is that this task has a temporal and a spatial complexity of $O(n^2)$.

The Condensed Nearest Neighbor (CNN) [31] choose instances near to class frontiers, reducing the size of training sets. However, CNN is not noise tolerant. Reduced Nearest Neighbor (RNN) [32], Selective Nearest Neighbor (SNN) [33] and Minimal Consistent Set (MCS) are methods based on CNN, and therefore, they have also problems with noisy data sets. RNN, SNN and MSC are more costly than CNN.

Neighborhood properties of SV have also been exploited to reduce size of training sets. Wang and Kwong [34] used neighborhood entropy, in [35] only the patterns in the overlap region around the decision boundary are selected. The method presented in [36] follows this trend but use fuzzy C-mean clustering to select samples on the boundaries of class distribution, whereas [29] uses hyper spheres.

The basis for decomposition methods lies in the fact that the training time can be reduced if only the active constraints of QPP are taken into account [37]. A similar idea to active sets methods for optimization is applied in decomposition methods. In the active set approach, two sets are used: the working set and the set of fixed variables. The optimization is made only on working set. For the case of SVM, the working set is usually composed of instances that violate the Karush–Kuhn–Tucker conditions. Apart of the proved convergence [38], a clear advantage of decomposition is that memory requirement is linear in the number of training examples; but on the other hand, because only a fraction of variables is being considered in each iteration, it is time consuming [39,40] if elements in active set are not carefully selected. One of the first decomposition methods was Chunking [30]. It consists in repetitively obtaining the maximum margin hyperplane from an amount of instances (called the chunk) and then forming a new chunk with the SV from the previous solution and some new instances. Probably the most famous decomposing algorithm is the SMO [11]. It considers the smallest size working set: only two training samples. LibSVM [10] is an algorithm based on SMO with the improvement of a more advanced mechanism of selection of working set by using the second order information method previously shown in [41]. The SVM^{light} [42] is another important state of the art decomposition method.

Variants of SVM speed up training time of SVM at expense of loosing accuracy [39]. These methods work by changing the original QPP formulation. Most of the variants methods conclude with a system of linear equations solved efficiently if the number of features is moderate, i.e., around 100. A representative method in this category is the least square SVM (LS-SVM) [43] which changes the original QPP by using a linear system of equations that can be solved explicitly or by using a conjugate gradient method. Other important methods are the PSVM (Proximal SVM) [44] and reduced SVM (RSVM) [45].

Parallel implementation of QPP is difficult because there is a strong dependence between data [46]. Most parallel methods for training SVM divide training set into independent subsets to train SVM in different processors, as in [46–48]. In [49], the kernel matrix of SVM is approximated by block diagonal matrices so that the original optimization problem can be decomposed into hundreds of sub problems, which are easy to solve in a parallel fashion. Other parallel implementations can be found in [50–54].

Geometric methods for SVM are based on that computing the optimal separating hyperplane is equivalent to find the closest pair of points belonging to convex hulls [19,55,56]. Recent advances on geometric methods can be found in [57–61].

Among all heuristic methods, the alpha seeding [62] consists in providing initial estimates of the α_i values for the starting of QPP. Alpha seeding seems to be a practical method to improve training time of SVM. Recently in [63] has been proposed an improvement of this method.

According to the reviewed literature, there are currently just few methods that combine DT for instance selection in a similar way to the presented in this research. In [64], the algorithm patterns by ordered projections (POP) is presented. It uses projections of instances on the axis of attributes to find the minimal number of elements to represent hyper-rectangles which contain instances of the same class (entropy zero). A disadvantage of POP is that reduction of the size of data sets is very low [65].

In [16], a method that approximates the decision boundary of SVM using a DT to speed up SVM in its testing phase is proposed. There are important differences with respect of our method: first, in [16], a SVM is used in some leaves of a DT. The idea is to reduce the number of test data points that require SVM's decision; and secondly, the other method is not focused in reducing the number of SV.

Recently, in [66], the combination of a DT and SVM was proposed. The underlying idea is to train a SVM first, and then use the predictions of the model obtained to modify the class of examples in the training set. A DT is afterward trained using the modified set. The SVM is used as a pre-processor for improving the performance of DT, when dealing with the problem of imbalance. The major drawback of this approach is its inability to deal with large data sets.

3. Preliminaries

This section presents some basic concepts of the SVM and decision trees. A more detailed explanation can be found in [67–69] respectively.

3.1. Support Vector Machines

Considering binary classification case, it is assumed that a training set is given as:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \tag{2}$$

i.e. $X = \{x_i, y_i\}_{i=1}^n$ where $x_i \in R^d$ and $y_i \in (+1, -1)$. The optimal separating hyperplane is given by

$$y = sign[w^T \varphi(x_i) + b] \tag{3}$$

which is obtained by solving the following QPP

$$\begin{aligned} &\min_{w,b} J(w) = \frac{1}{2} ww^T + c\sum_{i=1}^{n} \xi_i \\ &subject\ to:\ y_i[w^T\varphi(x_i) + b] \geq 1 - \xi_i \end{aligned} \tag{4}$$

where ξ_i are slack variables used to tolerate miss-classifications $\xi_i > 0, i = 1, 2, \ldots n, c > 0$. Eq. (4) is equivalent to the QPP 5 which is a dual problem with the Lagrange Multipliers $\alpha_i > 0$,

$$\begin{aligned} &\max_{\alpha_i} J(w) = -\frac{1}{2} \sum_{i=1,j=1}^{n} \alpha_i y_i \alpha_j y_j K\langle x_i \cdot x_j \rangle + \sum_{i=1}^{n} \alpha_i \\ &subject\ to:\ \sum_{i=1}^{n} \alpha_i y_i = 0, C \geq \alpha_i \geq 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{5}$$

With $C > 0$, $\alpha_i = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$, $\alpha_i \geq 0$, $i = 1, 2, \ldots, n$, are the coefficients corresponding to x_i, all the x_i with nonzero α_i are called SV. The function K is the kernel which must satisfy the Mercer Condition [8]. The resulting optimal decision function is defined as

$$y_i = sign\left(\sum_{i=1}^{n} \alpha_i y_i K\langle x_i \cdot x_j \rangle + b\right) \tag{6}$$

where $x = [x_1, x_2, \ldots, x_n]$ is the input data, α_i and y_i are Lagrange multipliers. An unpreviously seen sample x can be classified using (6). There is a Lagrangian multiplier α for each training point. When the maximum margin of the hyperplane is founded, only the closed points of the hyperplane satisfy $\alpha > 0$. These points are called support vectors (SV), the other points satisfy $\alpha = 0$, so the solution vector is sparse. Where b is determined by Kuhn–Tucker conditions:

$$\begin{aligned} &\frac{\partial L}{\partial w} = 0, \quad w = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i) \\ &\frac{\partial L}{\partial b} = 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \\ &\frac{\partial L}{\partial \xi_i} = 0, \quad \alpha_i - c \geq 0 \\ &\alpha_i \{y_i[w^T\varphi(x_i) + b] \geq 1 - \xi_i\} = 0 \end{aligned} \tag{7}$$

3.2. Decision trees

Decision tree techniques have become one of the most popular tools for classification and regression. One of the attractiveness of decision trees is that they can extract knowledge by inferring human understandable rules from data. Typically, a decision tree algorithm begins with the entire set of data, splits the data into two or more subsets based on the values of attributes and then repeatedly splits each subset into finer subsets until the split size reaches an appropriate level. The entire modeling process can be represented in a tree structure and the model generated can be summarized as a set of "if-then" rules. Decision trees are easy to interpret, computationally inexpensive, and capable of work with noisy data.

We used the C4.5 algorithm [69] to construct the decision tree data filter. The selection of the best attribute node is based on the Gain ratio (S, A) where S is a set of records and A is an attribute. This

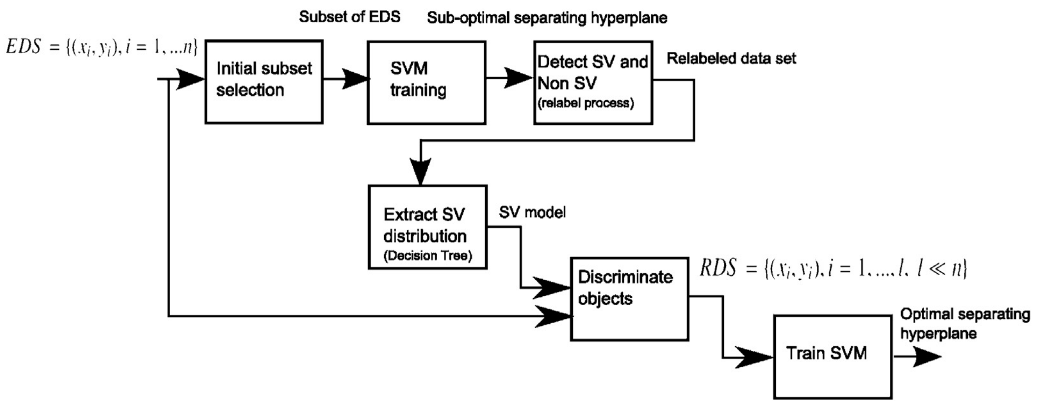
Subset of EDS
Sub-optimal separating hyperplane
$EDS = \{(x_i, y_i), i = 1, \ldots n\}$
Initial subset selection
SVM training
Detect SV and Non SV (relabel process)
Relabeled data set
Extract SV distribution (Decision Tree)
SV model
Discriminate objects
$RDS = \{(x_i, y_i), i = 1, \ldots, l, l \ll n\}$
Train SVM
Optimal separating hyperplane

Fig. 2. Block diagram of the proposed method.

gain defines the expected reduction in entropy due to sorting on A. It is calculated as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (8)$$

In general, if we are given a probability distribution $P = (p_1, p_2, \ldots, p_n)$ then the information conveyed by this distribution, which is called Entropy of P and it is given by

$$Entropy(P) = \sum_{i=1}^{n} - p_i \log_2 p_i$$

4. Proposed method

The decision boundaries discovered by SVM are determined by the Support Vectors, which are usually a small portion of training set. The SV correspond to the closest objects with opposite label for linearly separable case. In the general case, there is no way to exactly know a priori which objects in training set are the SV, however the knowledge on geometry of SVM can be exploited to pre-process a data-set and detect objects that are likely to be SV.

The proposed algorithm introduces a pre-processing step to quickly remove data that do not contribute to define the decision boundary of SVM, while preserving SV candidates.

In general the proposed method works as follows: it applies SVM on a small subset of original data-set in order to obtain a sketch of the optimal separating hyperplane, then it labels objects that are far from sketched hyperplane and objects that are close to it.

The method uses a decision tree to identify objects that have similar characteristics to the computed SV and then it removes the less important objects from the original data-set.

A deeper explanation of the process is given in the next subsections. Fig. 2 shows the proposed method.

4.1. Initial subset selection

In the first phase of the method, a small subset C is extracted from the Entire Data Set (EDS), subset C is used to compute a sketch of the optimal separating hyperplane.

In general, the separating hyperplane obtained from C is not the optimal. This happens because the random selection of examples from training data-set can discard some important objects that could be SV. Using just simple random sampling is not enough in most cases.

Figs. 3 and 4 exemplify this fact, the former shows a version of Checkerboard with 100,000 examples and the resulting decision boundaries found by SVM when using the entire data-set.

Training a SVM with this version of Checkerboard (or other one with similar size) takes long time, but a small subset dataset can be used to find a suboptimal separating hyperplane. Fig. 4 shows the decision boundaries computed by using a small subset set randomly chosen from the Checkerboard. It is clear that the classifier generalization is severally affected in this case. A key point in selection is to carefully chose a subset of examples from training set. This subset must be small and representative at the same time.

Commonly, the data sets contain similar number of examples with positive and with negative labels. Most classifiers including SVM can achieve good accuracy if their parameters are calibrated appropriately. In some cases, data sets are not balanced, but they contain most of the examples of one class (called majority class) and a few of the opposite class (minority class). These data sets are named imbalanced or skewed. In these cases, SVM

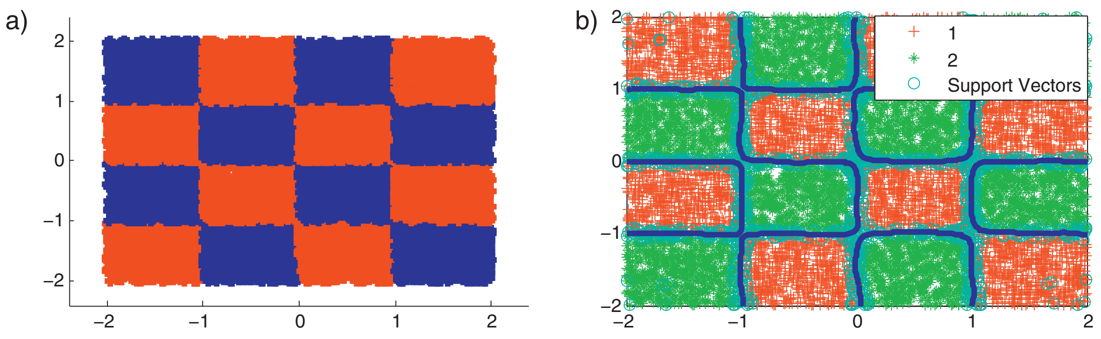

Fig. 3. Checkerboard data-set and decision boundaries founded by SVM.

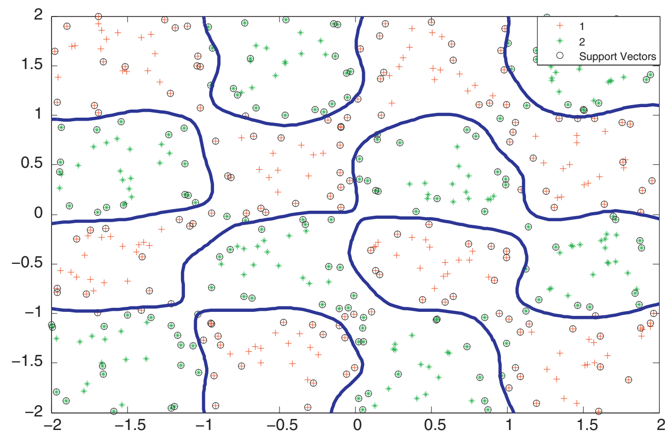

Fig. 4. Decision boundaries using simple random sampling.

does not perform well. Similar behavior occurs with other classifiers.

In order to face this problem, it is necessary to artificially balance the training set. A mechanism is built-in the proposed method to accomplish the balancing task. The result is an algorithm that implements a simple heuristic to select a small subset C from EDS taking into account the number of elements of each class.

The proposed method begins the selection of objects by computing the rate of positive and negative samples, this is achieved in linear time by using (9).

$$p \leftarrow \frac{\min\{N_+, N_-\}}{|EDS|} \tag{9}$$

with N_+, number of $x_i \in EDS$ s.t. $y_i = +1$; N_-, number of $x_i \in EDS$ s.t. $y_i = -1$; $|EDS| = N_+ + N_-$ and $N_+ > 0$, $N_- > 0$, i.e., at least one sample of each class.

Two thresholds τ_b,τ_u are defined to give an indication to the method on how should be considered the grade of imbalance of a data-set.

$$\tau_b, \tau_u \in (0, 1) s.t. \tau_b > \tau_u. \tag{10}$$

The thresholds (10) are used to indicate when the data-set is considered with small imbalance (13), moderate imbalance (12) or high imbalance (11).

$$p_h < \tau_u \tag{11}$$

$$\tau_u \le p_m < \tau_b \tag{12}$$

$$\tau_b \le p_s \le 0.5 \tag{13}$$

Fig. 5a shows a hypothetical example for p values of a data-set containing 1000 examples. Instances in minority class with less than 1% have a big imbalance as shown in Fig. 5b. If we have two data sets, with different size of minority class defined by $p = 0.1$ and $p = 0.5$, then the imbalance is given by 9:1 and 1:1 respectively as shown in Fig. 5b.

Algorithm 1. Initial Data Selection

```
Data
EDS: Entire Data Set
$\tau_u$, $\tau_b$: According to (10)
Output
$EDS_r$: A subset of instances ($EDS_r \subset EDS$)
Begin algorithm
  Compute $p$ using (9)
  If EDS is balanced then //see Eq. (13)
    Apply SRS to create $EDS_r$
  else if EDS is semi balanced then //see Eq. (12)
    Select samples from majority and minority classes considering the
      probabilities
    as $p$ and $1 - p$ respectively.
  else //The data set is imbalanced, see Eq. (11)
    Create $EDS_r$ taking 100% of minority class and selecting a number of
      instances of majority class.
  end if
  return $EDS_r$
End algorithm
```

According to the data-set, the user can modify the thresholds. For the experiments, the following values were used:

$$\tau_b = 0.25 \quad and \quad \tau_u = 0.1$$

The number of instances selected from EDS using these thresholds values is therefore as explained in the following:

Criterion I. If the amounts of both positive and negative labels in X are similar, i.e., the proportion between the number of positive and the number of negative labels is lesser or equal to 2:1, then the algorithm uses a simple random sampling (SRS) where each element of the data-set has an equal probability of selection.
Criterion II. If the proportion between positive and negative labels is greater than 2:1 and lesser than 9:1, then the algorithm uses an inverse probability proportional to size sampling, e.g., if there are 90% data points with negative labels and 10% data points with positive labels, the random selection algorithm selects 90% of the small data-set with positive labels.
Criterion III. Finally, if the proportion between data points with positive and negative labels is bigger than 9:1, then the algorithm uses all data points that have label with fewer rates.

This random selection approach has the greatest impact on determining the suboptimal separating hyperplane. This simple heuristic ensures that the subset obtained is balanced, regardless if the original one is skewed or not.

Recent studies show that the imbalance of data-sets considerably affects the performance of most classifiers [70], because in general they are designed to reduce the global mean error regardless classes distribution and therefore the decision boundaries are biased to the majority class in the training phase. This method of selection can improve the accuracy for a given sample size by concentrating sampling on few elements in this stage, but it has the greatest impact on determining a hyperplane.

Formally, our training data-set EDS consists of n pairs (x_1, y_1), $(x_2, y_2), \ldots, (x_n, y_n)$, with $x_i \in R^d$, and $y_i \in \{-1, 1\}$. After the applying the heuristic selection the obtained data set is $C = \{x_i, y_i\}_{i=1}^{l}$, with $l \ll n$.

Algorithm 1 shows the implementation of the proposed heuristic.

4.2. Detecting objects close to decision boundaries

After the reduced subset set C has been created from EDS, a sketch of optimal separating hyperplane is computed using data-set C and SVM. This hyperplane becomes important because it is

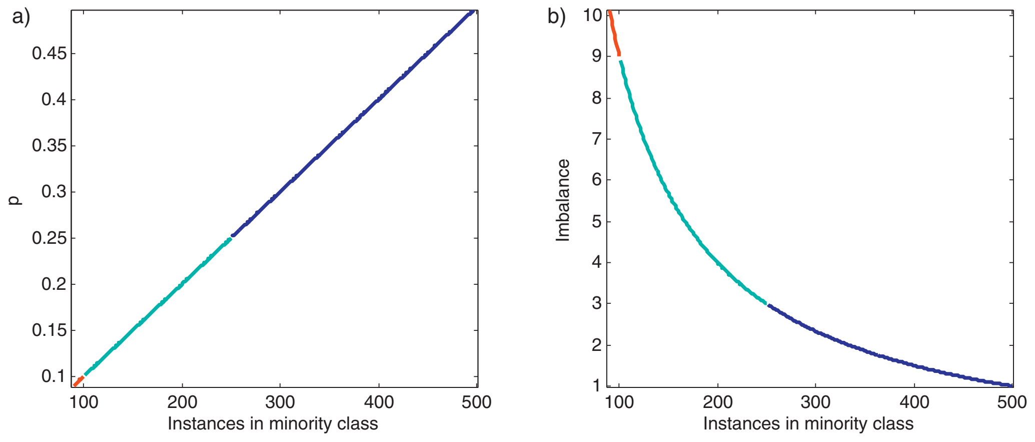

Fig. 5. Thresholds to different ratio of imbalance in data-sets.

used to model the pattern distribution of SVs and non-SVs in order to reduce the original training set *EDS*.

After the SVM classification stage, the hyperplane obtained with subset *C* is given by (14).

$$\sum_{k \in V_{small}} y_k \alpha^*_{small} K(x_k, x) + b^*_{small} = 0 \tag{14}$$

where V_{small} are the SVs, i.e. the data points that contain the most important features in the small data-set.

SVs are the most important data points of all input data-set and generally constitute a small fraction of the training examples. If most non-SVs can be removed from the entire data-set, then the SVM training can be accelerated drastically.

It is necessary to compute a general model of distribution of SVs in *C*, to detect SVs in *EDS* and remove objects that are non-SVs from it. The algorithm takes the SVs of *C* and labels them with an extra class E^{+1}, and non-SVs as outliers and labels them with an extra class E^{-1}. The outliers are observations that do not follow the statistical distribution of the bulk of data, in this case, outliers are objects far from suboptimal separating hyperplane. These points are not important when obtaining the optimal classification hyperplane.

Summarizing, the small data-set is given by

$$C = \{x_i, y_i\}_{i=1}^{l}$$

with $l \ll n$ the SVs in the small data-set are given by

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}, \quad 1, 2, \ldots, k,$$

where $k \in V_{small}$ and non-SVs are given by

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}, \quad 1, 2, \ldots, m \notin V_{small}$$

It is clear that $\{x_l, y_l\} = \{(x_k, y_k) \cup (x_m, y_m)\}$, consequently, data-set $\{x_i, y_i\}_{i=1}^{k} \in V_{small}$ is labeled with a extra class E^{+1} and $\{x_i, y_i\}_{i=1}^{m} \notin V_{small}$ with E^{-1}.

4.3. Modeling the distribution of support vectors

Once all the SV and outliers of *C* have been identified in the previous stage, a decision tree is induced in order to model the distribution of SV. Fig. 6 shows the SVs and the decision tree obtained from SVs and non-SVs. Decision tree is obtained from support vectors and non-support vectors. The function of DT is to reduce the entire data set, obtaining a small data set with the most important data. In Fig. 6 SVC represent the Support Vectors Candidates and N-SV the non-SV.

A decision tree is used here in the model to detect and retain all data that are close to the objects $(x_k, y_k) \in V_{small}$ and to remove data that are similar to outliers $\{x_i, y_i\} \notin V_{small}$.

The new distribution class is expressed by:

$$(x_1, y_{E_1}), (x_1, y_{E_2}), \ldots, (x_n, y_{E_n}) \tag{15}$$

i.e., training data-set $\{x_i, y_{E_i}\}_{i=1}^{q}$ where $q = k + m$, $x_i \in R^d$, and $y_{E_i} \in (+1)$ represent the SV obtained in the first stage of SVM and the pairs $x_i \in R^d$, $y_{E_i} \in (-1)$ represent the non-SV obtained in the first stage of SVM. Where $y_{E_i} \in (+1)$ constitute data points that contain the most important features and $y_{E_i} \in (-1)$ constitute the data points far from hyperplane (outliers). From these data points (SVs and non-SVs) obtained in the first stage of SVM is trained the decision tree.

To find a decision function that separate SVs from non-SVs, a decision tree is induced. In this way, objects close from hyperplane and data points far from hyperplane are distinguished. The reduced data-set RDS can be expressed as

$$RDS = \{x_{pi}, y_{pi}\} \cup \{\{x_i, y_i\}_{i=1}^{k} \in V_{small}\}$$

where x_{pi} are the data close from the decision hyperplane obtained in the first stage, which have a positive label, and V_{small} are the support vector from small data-set.

The obtained subsets contain objects that have similar distribution of SV in *EDS*, i.e., the entire data-set *EDS* is reduced into a small data subset with the most important data objects. It is clear that the quality of the reduced data points obtained in this phase benefits directly, in accuracy and training time, the proposed algorithm.

Different from algorithm presented in [16], where the authors used a $\delta - region$ to control the influence of the method on the training set, in this study the parameter used to recover the most important objects is the parameter σ of the RBF kernel.

Fig. 7 shows the data points recovered with different values of σ used in the first stage of SVM. Fig. 7a shows the recovered data set with $\sigma = 0.2$, in Fig. 7b are shown data points recovered with $\sigma = 0.5$ and in Fig. 7c are shown data points recovered with $\sigma = 0.8$.

Is clear that, the training time rely on the proper choice of σ parameter, which is selected in this research by grid search and cross validation. In the special case of the checkerboard data-set, the training time was reduced and the optimal hyperplane was obtained by using the smallest data-set. However, in some cases it is not the best solution and is necessary to find the best value of σ that reduces the training time without affecting the classifier accuracy.

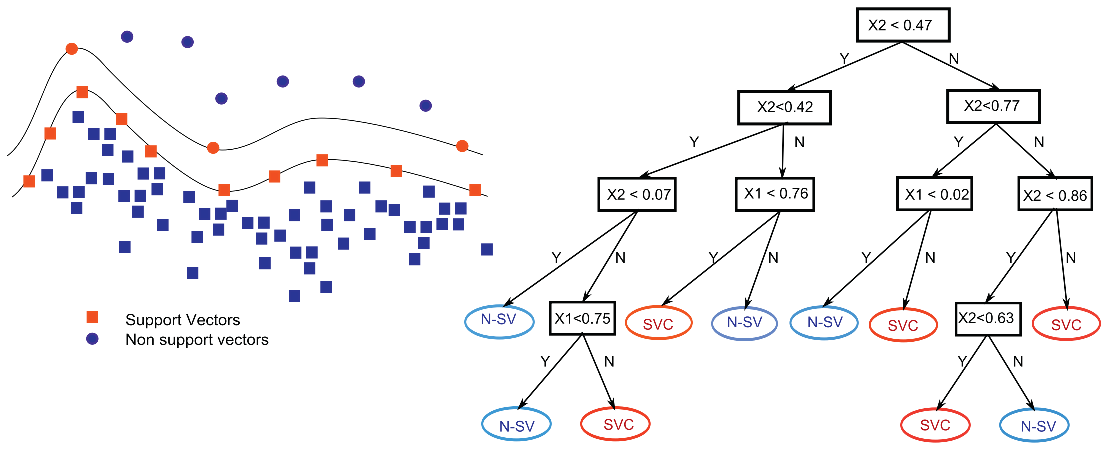

Fig. 6. Building a DT from SV and non-SV obtained in first stage of SVM.

4.4. *Training the Support Vector Machine*

The SVM is trained using the small subset selected in the previous state of the method. The complete method is shown in the block diagram of Fig. 2.

5. Experimental results and comparison

In order to show the effectiveness of the proposed method, it was tested with public available data-sets, in addition with two synthetic ones which were created by ourselves. The performance of the proposed method was compared with LibSVM, a state of the art training method for training SVM.

Organization of the experimental results is as follows: in the first part the characteristics of data-sets are presented. The second part of this section shows the performance of the proposed algorithm in terms of generalization capability and training times. A discussion on the choice of parameter σ is shown. The value of this parameter affects the classification accuracy.

5.1. *Data sets*

In order to demonstrate the performance of the proposed method, several experiments on benchmark problems were conducted. The following data-sets are used in the experiments: Checkerboard and UCI Adult data sets from UCI Machine learning repository (available at http://archive.ics.uci.edu/ml/); IJCNN1, Covtype Binary and Magic data sets from LibSVM (publicly available at http://www.csie.ntu.edu.tw/ cjlin/libsvm); ARN (www.pubmedcentral.nih.gov), and Synthetic 1 and Synthetic 2 which were created for the experiments.

The main characteristics of each data-set (training size, number of features and class of each data-set) are shown in Table 1. In the experiments, training and testing sets were randomly created by selecting the 80% and 20% of objects respectively.

The two synthetic data sets have two classes of purposes for testing binary classification. The Synthetic 1 data-set is linearly separable whereas Synthetic 2 data-set is linearly inseparable.

5.2. *Classification accuracy and training times*

The methods were implemented in Matlab. The experiments are carried out on several data-sets using SMO [11] in Support Vector Machine and the results were compared with the ones obtained using LibSVM [10]. The LibSVM method was selected because it is the most representative in the area of SVM, and it has demonstrated good performance in practical applications.

Training a SVM involves to choose some parameters. Such parameters have an important effect on the classifier's performance. In all the experiments, except in Synthetic 1, radial basis function (RBF) was used as kernel.

$$(K(x_i - x_j) = \exp(\|x_i - x_j\|/\sigma), \sigma > 0) \quad (16)$$

Cross validation and grid search was used to find parameters in (16) and also for computing the regularization parameter of SVM. We use model selection to get the optimal parameters. The hyper-parameter space is explored by a two-dimensional grid with $\sigma = [10^{-2}, 10^{-1}, 10^0, 10^1]$ and the regularization parameter $C = [10^1,$

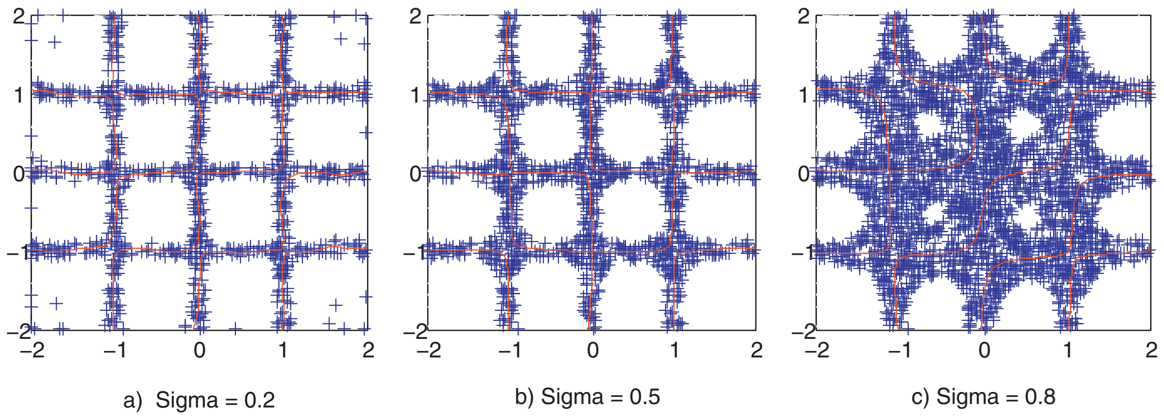

Fig. 7. Influence of parameter σ on the retrieved data-set.

Table 1
Data-sets used in the experiments.

Data-set	Training size	Testing size	Features
Fourclass	862	–	2
Mushroom	8124	–	112
German	1000	–	24
Diabetes	768	–	8
Breast_cancer	683	–	10
Australian	690	–	14
a1a	1605	30,956	123
a2a	2265	30,296	123
a3a	3185	29,376	123
a4a	4781	27,780	123
a5a	6414	26,147	123
a6a	11,220	21,341	123
a7a	16,100	16,461	123
a8a	22,696	9865	123
a9a	32,561	16,281	123
w1a	2477	47,272	300
w2a	3470	46,279	300
w3a	4912	44,837	300
w4a	7366	42,383	300
w5a	9888	39,861	300
w6a	17,188	32,561	300
w7a	24,692	25,057	300
w8a	49,749	14,951	300
Magic	19,020	–	10
IJCNN1	49,990	91,701	22
UCI Adult	48,842	–	14
Synthetic 1	109,000	–	2
Synthetic 2	500,000	–	2
Covtype Binary	581,012	–	54
Checkerboard	1,000,000	–	2

10^2, 10^3, 10^4, 10^5, 10^6]. In the experiments all data sets were normalized and 10-fold cross validation was used in order to validate the results. The experiments were performed on a 2.1GHz Intel Core i5 CPU, 4 GB RAM machine running Windows XP. The Synthetic 1 is the only linearly separable data-set used in the experiments. For this data-set, a linear kernel was used.

Table 2 shows a comparative between algorithms. In it *t* is the training time in seconds and *Acc* is a classification accuracy on test data-set. In some cases, the test data-set has the size of 20 percent of that of the entire data-set, and contains points that do not belong to the (80 percent) training set. In the case of UCI Adult, IJCNN1 and Magic data sets, there exist test data sets which contain data points that do not belong to the training data-set. Means and standard deviations (SD) over 30 independent runs are reported for all data sets.

In Table 2, the best results are shown in bold letters, standard deviation of the proposed method is shown in last column. It can be seen in Table 2 that proposed method requires more training time on small data sets. However, the proposed method requires several orders in less training time when the training data-set is big, while providing comparable classification accuracies. It is important because in many applications on the real world is necessary a reasonable trade-off between accuracy and training time.

The superiority of proposed method (in training time) becomes clearer in huge data-sets as can be seen in Table 2. In the case of Checkerboard, in Synthetic 1 and Synthetic 2 data sets, the SSVM algorithm requires substantially less time compared to LibSVM. However, is clear that where the data-sets is small, the proposed algorithm does not have a significant impact in terms of time. However, for large data-sets, a clear advantage can be observed.

It is observed from the results shown in Table 2 that on large data sets, LibSVM algorithm performs better than the proposed algorithm in almost all data sets. However, the difference is very small. Moreover, the proposed algorithm requires substantially less time compared to LibSVM algorithm.

Some studies have shown that there is a relationship between the accuracy and the number of support vectors in the training

Table 2
Comparisons in terms of accuracy and time on data-sets.

Data-set	LibSVM		Proposed method		
	t in seconds	*Acc*	*t* in seconds	*Acc*	*σ*
Fourclass	0.0008	**98.5**	0.1140	**98.5**	0.023
Mushroom	0.731	**100**	0.472	**100**	0.310
German	0.00658	**80.03**	0.0081	79.95	0.075
Diabetes	0.00235	**80.11**	0.004761	79.91	0.058
Breast_cancer	0.000520	**98.5**	0.003569	98.4	0.043
Australian	0.000909	**86.413**	0.01139	86.2319	0.081
a1a	0.126	**84.38**	0.176	84.19	0.028
a2a	0.242	**84.71**	0.183	84.38	0.062
a3a	0.439	**84.52**	0.187	83.93	0.085
a4a	0.98	**84.53**	0.477	84.49	0.093
a5a	1.71	84.44	0.864	**84.51**	0.153
a6a	5.42	84.58	2.95	**84.67**	0.278
a7a	11.28	84.66	4.95	**84.80**	0.318
a8a	23.43	85.04	8.71	**85.13**	0.271
a9a	50.64	85.05	20.88	**85.12**	0.249
w1a	0.52	97.23	0.12	**97.73**	0.048
w2a	1.057	97.40	0.23	**97.52**	0.098
w3a	2.05	**97.51**	0.394	97.44	0.083
w4a	10.91	97.60	1.52	**98.01**	0.036
w5a	26.73	97.66	2.20	**97.95**	0.249
w6a	89.93	97.87	4.13	**98.20**	0.485
w7a	177.89	97.94	7.03	**98.78**	0.552
w8a	690.61	**99.46**	21.09	99.29	0.310
Magic	103.67	91.5	22.97	**91.69**	0.379
IJCNN1	9.188	**98.59**	2.108	98.41	0.824
Synthetic 1	120	**100**	12.07	**100**	0.018
Synthetic 2	223.5	**99.9**	32.81	**99.9**	0.059
Covtype Binary	9532	**91.48**	4761	91.45	0.364
Checkerboard	12,632	**98.5**	857	**98.5**	0.074

set. In all data sets shown in Table 2 the number of support vectors represent only a fraction of the entire training set. However, in Covtype Binary many data points are support vectors. Preliminary studies show that over one hundred thousands of support vectors are required for a good approximation on Covtype Binary dataset.

Fig. 8 shows the results obtained with the proposed method and LibSVM algorithms.

Accuracy and training time of the proposed method depend of each training data set. Figs. 8 and 9 show the comparisons with UCI

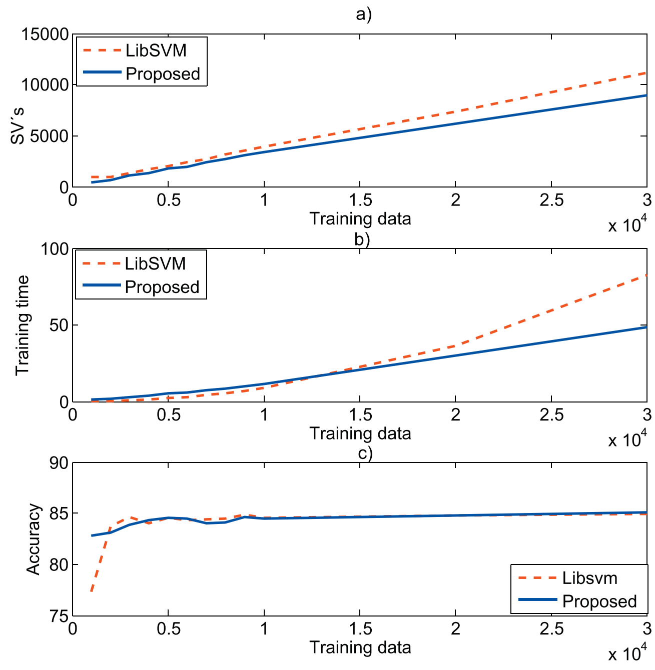

Fig. 8. Comparisons with UCI Adult data-set in terms of accuracy, training time and SV.

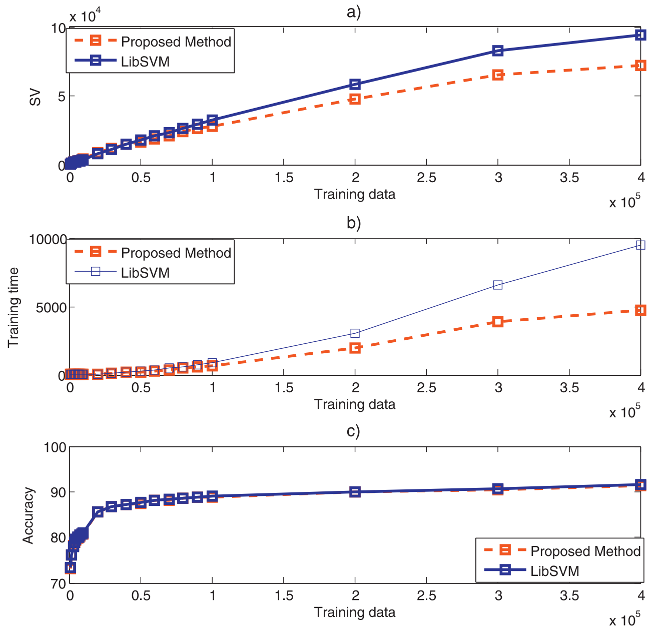

Fig. 9. Comparisons with Covtype data-set in terms of accuracy, training time and SV.

Adult and Covtype binary data-sets respectively. Comparisons are made in terms of accuracy, training time and SVs with different size of the training time. The number of support vectors significantly affects the training time of any SVM implementation and in this case the proposed approach is affected too. Data sets with many supports vectors can provoke that the training time of proposed method will be affected when the proportion between support vectors and instances in the data set is small.

As it can be seen from Fig. 8a the number of support vectors obtained with Libsvm is similar to the proposed method in this data set. It provokes that the training time of the proposed method is similar to LibSVM as can be seen in Fig. 8b.

On the other hand, as it can be seen from Fig. 9c, in this case, the number of support vectors is very high for both data sets too. However, in this data set, the proposed method reduces considerably the input data set without loss accuracy.

Note that when the training set is small, more training patterns bring in additional information useful for classification and so the number of support vectors is very high and training time is very similar to LibSVM. However, after processing around 100K patterns, both the time and space requirements of the proposed method begin to exhibit a constant scaling with the training set size. With the entire data-set, the training time of the proposed method is only a fraction of the training time used with LibSVM. In this case, the accuracy obtained with LibSVM is better than the proposed method. However, standard deviation over 30 independent runs shows in Table 3 demonstrates that accuracy means of LibSVM is better than the proposed method but the training time is very high.

Moreover, Fig. 9b shows that the means of accuracy of the proposed method is similar to LibSVM. In the case of Checkerboard data-set, we generate the data-set with class-overlap in order to do more complex the training process. The experimental results show that the training time of proposed algorithm is only 52.63 s in comparison with 857 s with LibSVM.

6. Performance analysis

In this section we show the complexity of the proposed algorithm. It is difficult to analyze the complexity of SVM algorithm precisely. This operation involves multiplication of matrices of size n, which has a complexity $O(n^2)$ and $O(n^3)$ at worst [11,12,24]. Is clear that without a decomposition method, is almost impossible for normal SVM to obtain the optimal hyperplane when the training data size is huge. On the other hand, in the SVM, the input data of p dimensions are loaded into the memory. The data type is float, so the size of each data point is 4 bytes. If we use normal SVM classification, the memory size for the input data should be $4(n \times p)^2$ at worst. Is clear that, in modern SVM implementations, it is not needed that the entire kernel matrix be put into the memory simultaneously. In the following, we assume that a QP implementation of each stage of SVM takes $O(n^3)$ time.

6.1. Algorithm complexity

The complexity of the proposed algorithm can be approximated as follows. The complexity of finding a small data sets is $O(l)$. $O(\log l)$ in the worst case, where l is the size of small input data. The approximate complexity of the two SVM training is $O(l^3)+O[(\sum_{i=1}^{l} n_i + m)^3]$. The total complexity of two stage classification is

$$O(l)+O(l^3)+O(m^3) \qquad (17)$$

where l is the number of data points selected in the first SVM stage, m are the data points closest to the hyperplane which were recovered using the decision tree, is clear that, the number of data points depends of σ. The choice of l and σ is very important to obtain fast convergence and to obtain a good accuracy. A good choice of l and σ will conduce to obtain fast convergence and high degree of generalization. On the other hand, the algorithmic complexity of C4.5 is $O(mn \log n)$ as proved in [16]. The total complexity of two stage classification via decision trees is

$$O(l)+O(l^3)+O(mp \log p)+O(m^3) \qquad (18)$$

where p is the number of attributes. Obviously complexity in 17 is smaller than the complexity of a normal SVM $O(n^3)$, because in the most cases $n \gg (l+mp)$.

Table 3
Comparisons in terms of SV, accuracy and time on data-sets.

Data-set	Data	SV1	RDS	SVs Proposed/LibSVM	Acc Proposed/LibSVM	Time in seconds Proposed/LibSVM
IJCNN1	49,900	209	10,703	2561/2711	98.41/ **98.59**	1.93/9.07
w8a	49,749	471	12,962	6820/24,520	99.29/**99.46**	21.09/590.61
a9a	32,561	1428	13,847	9754/19,059	**85.12**/85.05	20.88/50.64
Synthetic 1	109,000	3	395	3/3	**100/100**	12.07/120
Synthetic 2	500,000	122	5893	253/257	**99.9/99.9**	22.81/223.5
Checkerboard	1,000,000	1428	33,690	17,858/18,024	**98.5 / 98.5**	52.63/857
Covtype	100,000	849	47,237	32,180/32,091	88.81/**88.92**	684/887
	200,000	1452	78,076	56,473/57,676	89.95/**90.04**	1966/3063
	300,000	2073	97,136	75,356/81,413	90.48/**90.69**	3914/6542
	350,000	3095	113,384	76,814/92,2163	91.45/**91.45**	4731/8028

7. Discussions

The performance of SVM is intimately related to Support Vectors (SV), which are the most important data points in the entire data-set, because they represent the solution of the related QPP. An important improvement of training time of SVM is possible if we eliminate the data with less possibilities of being SV.

In this study an algorithm that reduces size of large training set is proposed. It uses twice a SVM. The first time, it recovers enough statistical information from SV and provides it to a DT, which uses this information to recover all SV candidates in the entire data-set. During the second time that SVM is used, it refines the solution to obtain the optimal solution.

Since the performance of algorithm depends directly on the first selected subset, it is very important to implement an efficient mechanism to select objects that represent the entire data-set. The quality of the reduced training set affects directly the accuracy and training time of the classifier.

In order to reduce the entire data-set, two important aspects must be considered: the imbalance of the training set and the size of it. In skewed data-sets, the proposed algorithm modify the selection process benefiting the minority class. In balanced data-sets, the selection process is random. On the other hand, size of the entire data-set is very important to select a small data-set and to use SVM in the first stage. It was observed in the experiments that the accuracy and the training time depend on how close to the suboptimal hyperplane the recovered candidates are. In [16], the authors used a δ – region to control the influence of the method on the training data-set. In this research, the parameter used to recover the most important objects is the parameter σ of the RBF kernel from SVM first stage. Furthermore, the training time rely on the proper choice of σ parameter, which is selected in this research by grid search. In the special case of the checkerboard data-set, using the smallest data-set the training time is reduced considerably. However, in some cases it is not the best solution and we have to find the best value of σ that reduces the training time without affecting the classifier accuracy.

Table 3 shows more results. It is clear that the proposed method is suboptimal, because it obtains a subset from the original data-set eliminating data points that the algorithm identify as less important to the learning process. However, sometimes, important data points can be eliminated improving the accuracy. The experimental results show that in some cases the accuracy obtained with the proposal method is better than LibSVM with the entire data-set. This anomaly is an interesting area for future research but a possible hypothesis for this situation could be that the proposed algorithm eliminates SV that introduce noise in the generalization process. However, in the most data-sets, the number of SV affects directly the classifier's performance. An example of this argument is shown by Covtype data-set (Table 3). The proposed method obtains less SV's than LibSVM and the performance is affected by it; although it is very small, the performance of LibSVM is better in all the cases than the proposed method.

Other important point is the training time. In small data-sets, the training time of the proposed algorithm is bigger than LibSVM, because the operation is more expensive in small data sets and LibSVM grows quadratically. However, the proposed method significantly outperforms LibSVM in large data sets in training time and sometimes in accuracy. Finally, our algorithm uses a data filter algorithm which find a function that discriminates between support and non-support vectors from a small data-set in order to select the best data points in the entire data-set. This step may often provoke improvement of the generalization performance, maintaining a fast convergence. The proposed method captures the pattern of the data and it provides enough information to obtain a good performance. The data filter of the proposed algorithm captures the statistical summaries of the entire data-set.

Table 3 shows results obtained with Checkerboard data-set, we generated it with a small intersection between classes. The case without intersection is very simple, the accuracy is always almost 100% and the SV's are a small quantity. In this case, the proposed algorithm obtains the same accuracy with less SVs.

8. Conclusions and future work

In this paper was presented a new algorithm for training SVM for classification. The proposed algorithm works very fast even with large data sets and outperforms the current state of the art SVM implementations without substantial reduction of accuracy.

The proposed method applies a data filter based on a decision tree that scans the entire data and obtains a small subset of data points. The proposed approach is conceptually simple, easy to implement, and for some experiments faster than other SVM training algorithms (SMO, LibSVM and SSVM) because it avoids calculating margins for nonsupport vector examples. The superiority of the proposed algorithm is experimentally demonstrated for some real life data sets in terms of training time.

The results of experiments on synthetic and real data sets show that the proposed approach is scalable for very large data sets while generating high classification accuracy.

As future work, proposed method ideas can be adapted to stream data mining SDM, which is a relatively new research area in data mining. The challenge in SDM for SVM rests in that the underlying model must consider concept drift phenomena and SV must be updated very quickly.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments and suggestions in order to make the paper more complete. This research is supported by a grant from the Universidad Autónoma del Estado de México (UAEMex) Research project (3771/2014/CI).

References

[1] J. Huang, X. Shao, H. Wechsler, Face pose discrimination using support vector machines (SVM), in: Proceedings of the 14th International Conference on Pattern Recognition (ICPR), vol. 1, 1998, pp. 154–156.
[2] J. Huang, J. Lu, C. Ling, Comparing naive Bayes, decision trees, and SVM with AUC and accuracy, in: Third IEEE International Conference on Data Mining, 2003. ICDM 2003, 2003, pp. 553–556, http://dx.doi.org/10.1109/ICDM.2003.1250975.
[3] D.C. Tran, P. Franco, J.-M. Ogier, Accented handwritten character recognition using SVM – application to French, in: Proceedings of the 2010 12th International Conference on Frontiers in Handwriting Recognition, ICFHR '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 65–71, http://dx.doi.org/10.1109/ICFHR.2010.16.
[4] B. Waske, J. Benediktsson, Fusion of support vector machines for classification of multisensor data, IEEE Trans. Geosci. Remote Sens. 45 (12) (2007) 3858–3866, http://dx.doi.org/10.1109/TGRS.2007.898446.
[5] H.M. Azamathulla, F.-C. Wu, Support vector machine approach to for longitudinal dispersion coefficients in streams, Appl. Soft Comput. 11 (2) (2011) 2902–2905.
[6] J. Ruan, X. Wang, Y. Shi, Developing fast predictors for large-scale time series using fuzzy granular support vector machines, Appl. Soft Comput. 13 (9) (2013) 3981–4000, http://dx.doi.org/10.1016/j.asoc.2012.09.005.
[7] A. Chaudhuri, K. De, Fuzzy support vector machine for bankruptcy prediction, Appl. Soft Comput. 11 (2) (2011) 2472–2486, http://dx.doi.org/10.1016/j.asoc.2010.10.003, The Impact of Soft Computing for the Progress of Artificial Intelligence.
[8] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, 1st ed., Cambridge University Press, 2000.
[9] F. Chang, C.-Y. Guo, X.-R. Lin, C.-J. Lu, Tree decomposition for large-scale SVM problems, J. Mach. Learn. Res. 9999 (2010) 2935–2972.
[10] C. Chih-Chung, L. Chih-Jen, Libsvm, A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 1–27.

[11] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: Adv. Kernel Methods Support Vector Mach., 1998, pp. 185–208.
[12] S. Vishwanathan, M. Narasimha Murty, Geometric SVM: a fast and intuitive SVM algorithm, in: 16th International Conference on Pattern Recognition, 2002. Proceedings, vol. 2, 2002, pp. 56–59, http://dx.doi.org/10.1109/ICPR.2002.1048235.
[13] N.E.I. Karabadji, H. Seridi, I. Khelf, N. Azizi, R. Boulkroune, Improved decision tree construction based on attribute selection and data sampling for fault diagnosis in rotating machines, Eng. Appl. Artif. Intell. 35 (0) (2014) 71–83, http://dx.doi.org/10.1016/j.engappai.2014.06.010.
[14] S. Piramuthu, Input data for decision trees, Expert Syst. Appl. 34 (2) (2008) 1220–1226, http://dx.doi.org/10.1016/j.eswa.2006.12.030.
[15] R. Wang, Y.-L. He, C.-Y. Chow, F.-F. Ou, J. Zhang, Learning ELM-tree from big data based on uncertainty reduction, Fuzzy Sets Syst. (2014), http://dx.doi.org/10.1016/j.fss.2014.04.028 (in press).
[16] M. Arun Kumar, M. Gopal, A hybrid SVM based decision tree, Pattern Recognit. 43 (12) (2010) 3977–3987, http://dx.doi.org/10.1016/j.patcog.2010.06.010.
[17] J. Cervantes, Clasificación de grandes conjuntos de datos vía máquinas de vectores soporte y aplicaciones en sistemas biológicos, Computer Science Department, Cinvestav, 2009, August (Ph.D. thesis).
[18] J. Cervantes, A. Lopez, F. Garcia, A. Trueba, A fast SVM training algorithm based on a decision tree data filter, in: I. Batyrshin, G. Sidorov (Eds.), Advances in Artificial Intelligence, Vol. 7094 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2011, pp. 187–197.
[19] K.P. Bennett, E.J. Bredensteiner, Geometry in learning, in: Geometry at Work, 1997.
[20] Y.-J. Lee, S.-Y. Huang, Reduced support vector machines: a statistical theory, IEEE Trans. Neural Netw. 18 (1) (2007) 1–13.
[21] F. Zhu, J. Yang, N. Ye, C. Gao, G. Li, T. Yin, Neighbors distribution property and sample reduction for support vector machines, Appl. Soft Comput. 16 (0) (2014) 201–209, http://dx.doi.org/10.1016/j.asoc.2013.12.009.
[22] X. Li, J. Cervantes, W. Yu, Fast classification for large data sets via random selection clustering and support vector machines, Intell. Data Anal. 16 (6) (2012) 897–914, http://dx.doi.org/10.3233/IDA-2012-00558.
[23] B. Gartner, E. Welzl, A simple sampling lemma: analysis and applications in geometric optimization, Discrete Comput. Geomet. 25 (4) (2001) 569–590.
[24] S. Canu, L. Bottou, S. Canu, Training Invariant Support Vector Machines Using Selective Sampling, MIT Press, 2007 http://eprints.pascal-network.org/archive/00002954.
[25] J.L. Balcázar, Y. Dai, J. Tanaka, O. Watanabe, Provably fast training algorithms for support vector machines, Theory Comput. Syst. 42 (4) (2008) 568–595.
[26] Y.-G. Liu, Q. Chen, R.-Z. Yu, Extract candidates of support vector from training set, in: 2003 International Conference on Machine Learning and Cybernetics, vol. 5, 2003, pp. 3199–3202, http://dx.doi.org/10.1109/ICMLC.2003.1260130.
[27] Z.-W. Li, J. Yang, J.-P. Zhang, Dynamic Incremental SVM Learning Algorithm for Mining Data Streams, 2007, pp. 35–37, http://dx.doi.org/10.1109/ISDPE.2007.115.
[28] A. Shigeo, I. Takuya, Fast training of support vector machines by extracting boundary data, in: ICANN '01: Proceedings of the International Conference on Artificial Neural Networks, Springer-Verlag, London, UK, 2001, pp. 308–313.
[29] J. Wang, P. Neskovic, L.N. Cooper, Selecting data for fast support vector machines training., in: K. Chen, L. Wang (Eds.), Trends in Neural Computation, Vol. 35 of Studies in Computational Intelligence, Springer, 2007, pp. 61–84.
[30] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, ACM, New York, NY, USA, 1992, pp. 144–152, http://dx.doi.org/10.1145/130385.130401.
[31] P.E. Hart, The condensed nearest neighbor rule, IEEE Trans. Inf. Theory 14 (1968) 515–516.
[32] G. Gates, The reduced nearest neighbor rule, IEEE Trans. Inf. Theory IT-8 (3) (1972) 431–433.
[33] G.L. Ritter, H.B. Woodruff, S.R. Lowry, T.L. Isenhour, An algorithm for a selective nearest neighbor decision rule (corresp.)., IEEE Trans. Inf. Theory 21 (6) (1975) 665–669.
[34] R. Wang, S. Kwong, Sample selection based on maximum entropy for support vector machines, in: 2010 International Conference on Machine Learning and Cybernetics (ICMLC), vol. 3, 2010, pp. 1390–1395, http://dx.doi.org/10.1109/ICMLC.2010.5580848.
[35] H. Shin, S. Cho, Neighborhood property-based pattern selection for support vector machines, Neural Comput. 19 (3) (2007) 816–855, http://dx.doi.org/10.1162/neco.2007.19.3.816.
[36] X. Jiantao, H. Mingyi, W. Yuying, F. Yan, A fast training algorithm for support vector machine via boundary sample selection, in: Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, 2003, vol. 1, 2003, pp. 20–22, http://dx.doi.org/10.1109/ICNNSP.2003.1279203.
[37] M. Doumpos, An experimental comparison of some efficient approaches for training support vector machines, Oper. Res. 4 (2004) 45–56, http://dx.doi.org/10.1007/BF02941095.
[38] N. List, S. Hans-Ulrich, A general convergence theorem for the decomposition method, in: COLT, 2004, pp. 363–377.
[39] G. Wang, A survey on training algorithms for support vector machine classifiers, in: Proceedings of the 2008 Fourth International Conference on Networked Computing and Advanced Information Management – Vol. 1 of NCM '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 123–128, http://dx.doi.org/10.1109/NCM.2008.103.
[40] C.-W. Hsu, C.-J. Lin, A simple decomposition method for support vector machines, Mach. Learn. 46 (2002) 291–314, http://dx.doi.org/10.1023/A:1012427100071.
[41] R.-E. Fan, P.-H. Chen, C.-J. Lin, Working set selection using second order information for training support vector machines, J. Mach. Learn. Res. 6 (2005) 1889–1918.
[42] T. Joachims, Making large-scale SVM learning practical, in: Adv. Kernel Methods Support Vector Learn., 1998, pp. 169–184.
[43] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (1999) 293–300, http://dx.doi.org/10.1023/A:1018628609742.
[44] G. Fung, O.L. Mangasarian, Incremental support vector machine classification, in: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 77–86.
[45] Lee Y. jye, O.L. Mangasarian, RSVM: reduced support vector machines, in: Data Mining Institute, Computer Sciences Department, University of Wisconsin, 2001, pp. 00-07.
[46] H.P. Graf, E. Cosatto, L. Bottou, I. Durdanovic, V. Vapnik, Parallel support vector machines: the cascade SVM, in: In Advances in Neural Information Processing Systems, MIT Press, 2005, pp. 521–528.
[47] L. Bao-Liang, W. Kai-An, W. Yi-Min, Comparison of parallel and cascade methods for training support vector machines on large-scale problems, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 2004, vol. 5, 2004, pp. 3056–3061, http://dx.doi.org/10.1109/ICMLC.2004.1378557.
[48] R. Collobert, Y. Bengio, S. Bengio, Scaling large learning problems with hard parallel mixtures, Int. J. Pattern Recognit. Artif. Intell. (IJPRAI) 17 (3) (2003) 349–365.
[49] J. xiong Dong, A. Krzyzak, C. Suen, Fast SVM training algorithm with decomposition on very large data sets, IEEE Trans. Pattern Anal. Mach. Intell. 27 (4) (2005) 603–618, http://dx.doi.org/10.1109/TPAMI.2005.77.
[50] S. Qiu, T. Lane, Parallel computation of RBF kernels for support vector classifiers, in: Proc. 5th SIAM International Conference on Data Mining (SDM05), 2005, pp. 334–345.
[51] T. Serafini, L. Zanni, G. Zanghirati, Some improvements to a parallel decomposition technique for training support vector machines, in: B. Di Martino, D. Kranzlmller, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface, Vol. 3666 of Lecture Notes in Computer Science, 2005, pp. 9–17.
[52] T. Eitrich, B. Lang, On the optimal working set size in serial and parallel support vector machine learning with the decomposition algorithm, in: AusDM '06: Proceedings of the Fifth Australasian Conference on Data Mining and Analytics, Australian Computer Society, Inc, Darlinghurst, Australia, 2006, pp. 121–128.
[53] F. Poulet, Multi-way distributed SVM algorithms, in: Parallel and Distributed computing for Machine Learning. In conjunction with the 14th European Conference on Machine Learning (ECML'03) and 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03), Cavtat-Dubrovnik, Croatia, 2003.
[54] G. Zanghirati, L. Zanni, A parallel solver for large quadratic programs in training support vector machines, Parallel Comput. 29 (4) (2003) 535–551, http://dx.doi.org/10.1016/S0167-8191(03)00021-8.
[55] Y. Ming-Hsuan, A. Narendra, A geometric approach to train support vector machines, in: Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2000, p. 1430, http://dx.doi.org/10.1109/CVPR.2000.855851.
[56] Crisp, Burges, A geometric interpretation of υ-SVM classifiers, NIPS 12 (2000) 244–250.
[57] M.E. Mavroforakis, M. Sdralis, S. Theodoridis, A geometric nearest point algorithm for the efficient solution of the SVM classification task, IEEE Trans. Neural Netw. 18 (5) (2007) 1545–1549.
[58] Z. Liu, J.G. Liu, C. Pan, G. Wang, A novel geometric approach to binary classification based on scaled convex hulls, IEEE Trans. Neural Netw. 20 (7) (2009) 1215–1220.
[59] E. Osuna, O.D. Castro, Convex hull in feature space for support vector machines, in: Proceedings of the 8th Ibero-American Conference on AI: Advances in Artificial Intelligence, IBERAMIA 2002, Springer-Verlag, London, UK, 2002, pp. 411–419.
[60] X. Peng, Efficient geometric algorithms for support vector machine classifier, in: 2010 Sixth International Conference on Natural Computation (ICNC), vol. 2, 2010, pp. 875–879, http://dx.doi.org/10.1109/ICNC.2010.5583913.
[61] Z.-Q. Zeng, H.-R. Xu, Y.-Q. Xie, J. Gao, A geometric approach to train SVM on very large data sets, in: 3rd International Conference on Intelligent System and Knowledge Engineering, 2008. ISKE 2008, vol. 1, 2008, pp. 991–996, http://dx.doi.org/10.1109/ISKE.2008.4731074.
[62] D. DeCoste, K. Wagstaff, Alpha seeding for support vector machines, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00, ACM, New York, NY, USA, 2000, pp. 345–349, http://dx.doi.org/10.1145/347090.347165.
[63] D. Feng, W. Shi, H. Guo, L. Chen, A new alpha seeding method for support vector machine training, in: L. Wang, K. Chen, Y. Ong (Eds.), Advances in Natural Computation, Vol. 3610 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2005, p. 418.
[64] J.C.R. Santos, J.S. Aguilar-Ruiz, M. Toro, Finding representative patterns with ordered projections, Pattern Recognit. 36 (4) (2003) 1009–1018.
[65] Y. Caises, A. Gonzlez, E. Leyva, R. Prez, Combining instance selection methods based on data characterization: an approach to increase their effectiveness, Inf. Sci. 181 (20) (2011) 4780–4798.

[66] A. Almas, M. Farquad, N. Avala, J. Sultana, Enhancing the performance of decision tree: a research study of dealing with unbalanced data, in: 2012 Seventh International Conference on Digital Information Management (ICDIM), 2012, pp. 7–10.
[67] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[68] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, 1984.
[69] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
[70] R. Akbani, S. Kwek, N. Japkowicz, Applying support vector machines to imbalanced datasets, in: ECML'04, 2004, pp. 39–50.