Data Selection Using Decision Tree for SVM Classification

Abstract—Support Vector Machine (SVM) is an important classification method used in many areas. Because the training of an SVM is computationally expensive, some methods to improve its training time have been proposed in recent years. Data selection methods for SVM select most important examples from training data sets to reduce the computational complexity of the related quadratic programming problem.

This paper introduces a novel data reduction method, it works by detecting clusters and then selects some examples from them. Different from other similar state of the art algorithms, the novel method uses a decision tree to form partitions that are treated as clusters, and then executes a guided random selection of examples from them. The aim of using trees to detect clusters is that they can be linearly separable. Taking advantage of the Eidelheit Separation Theorem, it is possible to reduce the size of training sets by carefully selecting examples from training sets.

The novel method was compared with LibSVM using public available data sets, experiments demonstrate an important reduction of the size of training sets whereas showing only a slight decreasing in the accuracy of classifier.

Index Terms—SVM, Decision tree, Eidelheit, Data reduction

I. INTRODUCTION

Support Vector Machine (SVM) is a classification method with strong mathematical background, it achieves good accuracy and it has been successfully applied in many areas [1][2][3].

A well known drawback of SVM is that it is computationally infeasible on large data sets, because it has \(O(n^3)\) time and \(O(n^2)\) space complexities [4]. This disadvantage comes from the fact that SVM needs to solve a quadratic programming problem (QPP) in order to find the optimal separation hyperplane.

Development of methods to apply SVM on large data sets is an active research area in recent years, these methods can be divided into four main types [5]: SVM variants (modifying original QPP problem), Data Reduction (reducing the size of training data sets), Decomposition (similar to optimization active set methods) and others (Lagrangian pre-initialization, geometric approach, etc.).

Main motivation of data reduction methods relies on the fact that training a SVM with all examples in training set produces the same result as using only the so-called support vectors (SV), further for most cases the SV are a small portion of whole training set [6][7]. It would result convenient to eliminate instances that are not SV before training, however there is no way to exactly know which instances are SV and which are not until the QPP has been solved.

The Decision Tree (DT) classifier is a method that has been used as a preprocessing step for SVM in recent years. In [8] 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 regions found by DT are less “complicated” than the region occupied by the entire training set. A SVM is applied to each region, the computational cost is less expensive compared with training a SVM with the whole data set. A similar approach was recently presented in [9], this consists in reducing the number of instances to train an SVM for classification. The central idea is to approximate the decision boundary of SVM using DT, i.e., capture the objects near decision boundary.

In [10] a DT is used to select examples close to decision boundary, the idea is use twice a SVM, the first time with a few randomly selected examples to obtain a separating hyperplane which is used to select examples close to the SV of separating hyperplanes using a DT. The SVM is trained again using this time those examples selected by the DT.

DT have been also combined with SVM to face multi class problems. According to the reviewed literature there is a greater number of proposals that combine SVM with DT to face multi class problem than those that have been developed to improve training time of former classifier. For multi class problems the binary tree of SVM method was presented in [11]. The method selects two classes for training in every node of a DT, and employs probabilistic outputs to measure the similarity between remaining samples and the two classes used in training. In [12] the multi class problem has been simplified by attempting to convert it into two-class classification problem by using C4.5 [13] algorithm to generate two subsets, all classes that are less separated by the margin are joined and treated as one class, the class with larger margin is considered as the opposite class. Two disadvantages with the method presented in [12] are that it needs to compute margins at each stage of DT, and that it shows good performance only with large number of classes. In [14] these drawbacks are solved by inserting information about separability (margin measures) in lists. Another algorithm that faces multi class problem was presented in [15], it creates a tree structure with binary classifiers based on SVM. The disadvantage of this method is that every node except leaf node defines a decision hyperplane that separates a class from the others, so as to the right and the left sub trees are imbalance, and it is known that SVM do not behaves well in such situation. Also, at the first nodes the SVM are trained with a large number of instances, whereas the higher nodes contain a few ones.

Summarizing, there exist algorithms that combine DT and SVM for improving training time of the last classifier, most
of the methods apply one SVM on each leaf in the DT.

In this paper we introduce a novel data reduction method for SVM based on DT. The method applies a DT to detect regions with low entropy, then a number of examples is chosen giving preference to those ones that are furthest from the center of that region. Only one SVM is then trained using the selected examples. The method is validated with public available data sets. According to the experiments, the method is effective and produces small size training sets which are used to train a SVM and obtain a classifier that achieves good accuracy, almost the same that the obtained with LibSVM in most cases.

According to the reviewed literature, there is currently no methods that combine DT and a guided random selection to improve training time of SVM for classification task in a similar way to the presented in this paper. The simple random sampling is commonly used to under sample large training sets, however in most methods it is necessary to train SVM more than once and most important, random sampling can yield to unrepeatable results.

The rest of the paper is organized as follows. Section II contains a brief review on SVM, Karush-Khun Tucker conditions and decision trees. The fundamentals, the underlying idea and details on implementation of the the proposed method are explained in section III. Performance analysis is presented in section IV. The experiments, results and discussion are shown in section V. This paper finishes with a conclusion and some ideas to extend the method to online environments.

II. PRELIMINARIES

A. Support Vector Machines

SVM classifier obtains the largest margin separating hyperplane by solving the QPP (1).

\[
\min_{\omega} \|\omega\|^2 + C \sum_{i=1}^{N} \xi_i^p \\
\text{s.t.} y_i (\omega^T x_i + b) \geq 1 - \xi_i, i = 1, \ldots, N \\
\xi_i \geq 0
\]

\( \omega \) is the vector normal to separating hyperplane, \( \xi_i \) is a slack variable used to permit miss classifications, which enable to work with non linearly separable data sets.

The Wolfe dual form of the (1) for SVM is (4)

\[
\min_{\alpha_i} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{i=1}^{n} \alpha_i, \ i = 1, \ldots, n
\]

such that

\[ 0 \leq \alpha_i \leq C, \ \forall i = 1, \ldots, n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0 \]

The constant \( C > 0 \) controls the trade-off between the separation margin and the errors.

The KKT conditions (5) for QPP (4) are:

\[
\alpha_i = 0 \Rightarrow y_i (\langle \omega, x_i \rangle + b) > 1 \text{ and } \xi_i = 0 \\
0 < \alpha_i < C \Rightarrow y_i (\langle \omega, x_i \rangle + b) = 1 \text{ and } \xi_i = 0 \\
\alpha_i = C \Rightarrow y_i (\langle \omega, x_i \rangle + b) < 1 \text{ and } \xi_i \geq 0
\]

The examples with \( \alpha_i > 0 \) define the optimal separating hyper plane. The \( \alpha_i \) is zero if the the \( i^{th} \) example is not a SV and therefore it is not close to separating hyperplane, see figure 1.

This means that examples close to decision boundaries are the most important for SVM, however it is not possible to exactly know the location of those boundaries before solving (1). This observation will be taken into account in the design of the proposed method.

B. Decision Trees

DT work by partitioning recursively the input space into successive each time “purer” subsets, where purer refers to subsets containing (most or all) instances of the same class. The number of splits (branching factor) can vary at each stage, but typically is fixed to two. The partitions are realized based on an attribute, this is selected considering a measure of impurity that determines the best way to partition a set, or in other words the degree of impurity of the produced partitions. The measures of impurity are defined in terms of the class distribution of the instances before and after splitting a subset. The most common impurity measures are entropy, misclassification and the Gini impurities, see equations (6) to (8).

\[
\text{Entropy}(t) = - \sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)
\]

\[
\text{Gini} = 1 - \sum_{i=0}^{c-1} [ p(i|t) ]^2
\]

\[
\text{Classification error}(t) = 1 - \max[p(i|t)]
\]
The main advantages of DT over other classification methods are interpretability, fast training and classification times. On the other hand, the decision boundaries created by DT are usually perpendicular to the feature axes which is different from SVM, this can be seen as a disadvantage in certain cases, however any decision boundary can be approximated by using a large enough DT [16]. Pruning can be used to avoid over-fitting. Most common DT are C4.5 [13] and CART [17].

III. PROPOSED METHOD

In this section the underlying idea and the basis of a novel data reduction method for SVM are presented.

A. Foundations of the method

Most of the clustering based methods to improve training time of SVM use centroids of detected clusters and then train SVM multiple times to refine the hyperplane [18][19][20].

The novel method is similar to those methods in the sense that it first creates clusters of examples and then trains SVM using a summarized version of the clusters.

The main difference between the proposed method and the other ones is that there is no necessity of guessing the number of clusters in training set, which is a drawback in some of those algorithms, the method does not use any distance measure to create the clusters and in addition a SVM is trained only once.

The proposed method is also somewhat similar to random sampling methods, however a uniform distribution of examples is not supposed like simple random sampling or other algorithms do, further a SVM is not trained several times, this is an important difference to other randomized methods.

In this novel method the centroids are not used as summarization of training sets as typical in methods such as [21], but a number of examples is randomly selected. The probability of an example to be chosen changes with respect to its distance to the center of cluster, in this way the examples located on exterior boundaries of clusters are preferred because they are probably SV, according to the observation made in [9]. This mechanism is a guided random selection and not a simple random sampling.

In general, a DT finds regions in which almost all of the contained examples are of the same class. Those regions are specified by the leaves of the DT which can be seen as a kind of cluster whose boundaries are given by splits orthogonal to axes of features. Different from specific clustering methods such as KNN or Fuzzy C-Means which form clusters explicitly by grouping examples using distance or density measures, a DT discovers pure regions using a purity function such as the Gini index or entropy gain. An advantage of DT is that there is no necessity of specifying the number of clusters.

The figure 2 shows a toy example and the decision boundaries discovered by a DT C4.5, it can be seen that the number of clusters (leaves) for this example is four and that all clusters are not pure.

At this time is useful to remember that according the KKT conditions (5) the examples close to decision boundaries are the most important examples for SVM. Intuition says that the points close the center of clusters are not too important and can be safely discarded. Other methods that exploit this observation in a different way are [18][22][23].

This intuition can be supported on Eidelheit Separation theorem, which states that given two convex sets $K_1$ and $K_2$ in a real vector space $X$, such that $K_1$ contains interior points and $K_2$ contains no interior points of $K_1$. Then there is a closed hyperplane $H$ separating $K_1$ and $K_2$, i.e., there is a $x^* \in X$ such that

$$\sup_{x \in K_1} \langle x, x^* \rangle \leq \inf_{x \in K_2} \langle x, x^* \rangle \quad (9)$$

In other words, $K_1$ and $K_2$ lay in the opposite half-spaces determined by the hyperplane $H$.

The Eidelheit separation theorem allows to design a new and fast algorithm using the clusters discovered by DT as follows. Consider two adjacent clusters, say C and D in figure 2, one can remove the minority class examples from each one, so the the clusters become pure ones.

Without loss of generality we can say that each cluster is in a convex set, further, they do not intersect. The split created by the DT perfectly separates these two clusters, so the split is a separating boundary $h_{split}$, although probably not the optimal one. According to Eidelheit separation theorem the closest examples to the $h_{split}$ that belong to the convex sets define a supporting hyperplane.

It is possible to select the closest pair of examples that produce the optimal separating hyperplane for clusters C and D by solving the QPP (10), however solving it is computationally expensive considering that this must be repeated for each pair of opposite class clusters.

$$\min_{x_i \in C_1, x_j \in C_2} \|x_i - x_j\|$$

(10)

$C_1$ is a cluster containing examples with opposite class to cluster $C_2$.

Instead of solving this costly problem we randomly select examples from each cluster. However, different from simple...
random sampling or other randomized methods for SVM, in this method the probability of an example to be selected from a cluster is given by (11).

\[
p(x_i) = 1 - \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(\delta)^2}{2\sigma^2}}
\]  

(11)

Where \(\delta\) is the (normalized) distance of the example to the center of cluster and \(\sigma\) the standard deviation. The greater the distance of an example with respect to the center of cluster, the higher its probability to be chosen.

Fig. 3. Probabilities within clusters represented in a gray scale

The figure 3 shows the probability values represented in grayscale for the toy data set. The darker a region is the higher its probability to contain more important examples.

B. Implementation details

The proposed method is forward to implement, it consists in three simple steps:

1) Train a DT using whole data set. In this paper we used C4.5.
2) Recover all the leaves of DT, these are treated as clusters with low entropy.
3) Select examples from clusters weighting their probability to be chosen as a function to distances to the center.
4) Train SVM.

The Algorithm 1 shows the pseudo code for the three first steps (data selection). The DT used in the algoritm was the C4.5 of Weka [24], it is called as J48 class within Weka.

![Algorithm 1 Data Selection](image)

Algorithm 1 Data Selection

Input: \(X\) A training set

Output:

- \(X_r\) A subset of \(X\)

Train a C4.5 DT on \(X\)

for each cluster do

//Each leaf of DT is seen as a cluster

for each \(x_i\) in current cluster do

Compute \(p(x_i)\) using (11)

Add \(x_i\) to \(X_r\) randomly according to its \(p(x_i)\)

end for

end for

return \(X_r\)

Where \(k\) is the current example being examined, \(j = 1, \ldots, d\) is the \(j^{th}\) feature of \(k^{th}\) example, \(i = 1, \ldots, m\), \(m\) is the number of examples in cluster, \(\min\{x_{i,j}\}\) and \(\max\{x_{i,j}\}\) is the minimum and maximum values of feature \(j\) in the cluster respectively.

The “distance” \(\delta\) of example \(k\) to the center of cluster is computed with (13). Because the previous normalization, the center of every cluster has always 0.5 value in each feature or dimension.

\[
\delta_k = \sqrt{\frac{\sum j = 1 (x_{kj} - 0.5)^2}{d}}
\]

(13)

with \(d\) the number of features of training set.

The use of \(d\) in (13) is to produce

\[0 \leq \delta_k \leq 1\]

Parameter \(\sigma\) affects number of selected examples by the method. The figure 4 shows an example of guided selection for a uniformly distributed set of examples for different values of \(\sigma\): (A) Original set, (B) \(\sigma = 0.10\), (C) \(\sigma = 0.40\), (D) \(\sigma = 0.70\).

IV. PERFORMANCE ANALYSIS

The proposed method uses the C4.5 DT to detect clusters and then randomly selects examples giving a higher probability to be chosen as those that are located on the exterior boundaries of each cluster.

The C4.5 has the following training time for non numeric features.

\[O(d \cdot m \cdot \log_2(m))\]

with \(d\) the number of features and \(m\) the number of examples.

The original Weka implementation of C4.5 does not use any discretization of continuous-valued attributes, however for the presented method we introduce this modification at the expense of degrading accuracy of such classifier.

The resulting complexity of the slightly modified C4.5 is shown in (14), \(L\) is a non negative integer number. Several
values were tested and the value was finally set to $L = 20$ for the experiments.

$$O(L \cdot d \cdot m \cdot \log_2(m))$$

(A)

(14)

At each node the DT tests for the best attribute considering only $L$ bins, without considering all the examples in the node.

Once the DT has been trained, its leaves are used as clusters and some examples in them are selected. The selection is executed in linear time, so the time complexity of the method is

$$O(L \cdot d \cdot m \cdot \log_2(m)) + \sum_{i=1}^{C} N_i$$

(B)

(15)

With $C$ the number of clusters and $N_i$ the number of examples in each cluster.

Observe that

$$\sum_{i=1}^{C} N_i = m$$

Solving the QPP to compute the optimal separating hyperplane adds up a quadratic term to the final training time of SVM, however because the total number of selected examples is always lesser than the original size of training set, the proposed method is more efficient than other implementations.

V. EXPERIMENTS AND DISCUSSION

In this section we present the results of the method applied on nine different data sets. Because some of the selected datasets are for multi class problems, we create binary versions of them by selecting two classes. The total number of datasets was finally thirteen.

A. Data sets

In order to test the effectiveness of the proposed method, it was tested on typical datasets used for classification, the table I shows the main characteristics of them. Most of the training sets have numeric features, however one of them has only nominal values and another set has mixed features.

B. Setup

The method was implemented using Java as programming language and Weka as base platform. The C4.5 decision tree implemented in J48 class of Weka was slightly modified to obtain access to some of its members.

All the experiments were run on a Laptop with Intel core i7 2670QM CPU at 2.2 GHz 8 GB RAM, Windows 7 Operating System installed.

The LibSVM library was used to train SVM, the kernel used for SVM was the RBF function, the $\gamma$ parameter was arbitrarily set to 0.01 for all the experiments. The amount of memory given to the JVM was set to 1,200 MB. Among all current implementations of training algorithms for SVM, LibSVM was selected because it outperformed other methods in previous tests.

For each run, the training sets were randomly partitioned in two sets: training (70%) and testing (30%), the presented results correspond to the average of 50 runs of each experiment, these results are shown in the Table II.

The meaning of columns and rows in the Table II is the following.

The name of training sets is shown in bold face, the respective size of training set (in parenthesis) follows the name.

The column $Size_{red}$ is the size of the reduced training set after the proposed method (identified as the rows DSDT in the Table II) was applied.

The column $T_{training}$ is the training time. For the DSDT method, that column is the sum of the time to reduce the training set and the time to train SVM.

1Source code of the proposed method and datasets used in experiments available at http://www.mediafire.com/download.php?id=7xzj05qa29fzr, the password: ieeeicdm2012

TABLE I

DATASETS FOR EXPERIMENTS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Dim</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Feat. type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris-setosa</td>
<td>100</td>
<td>4</td>
<td>50</td>
<td>50</td>
<td>Numeric</td>
</tr>
<tr>
<td>Iris-versicolor</td>
<td>100</td>
<td>4</td>
<td>50</td>
<td>50</td>
<td>Numeric</td>
</tr>
<tr>
<td>Iris-virginica</td>
<td>100</td>
<td>4</td>
<td>50</td>
<td>50</td>
<td>Numeric</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>34</td>
<td>126</td>
<td>225</td>
<td>Numeric</td>
</tr>
<tr>
<td>diabetes</td>
<td>768</td>
<td>8</td>
<td>500</td>
<td>268</td>
<td>Numeric</td>
</tr>
<tr>
<td>svmguide3</td>
<td>1,243</td>
<td>22</td>
<td>296</td>
<td>947</td>
<td>Numeric</td>
</tr>
<tr>
<td>Waveform-0</td>
<td>3,308</td>
<td>40</td>
<td>1,653</td>
<td>1,655</td>
<td>Numeric</td>
</tr>
<tr>
<td>Waveform-1</td>
<td>3,347</td>
<td>40</td>
<td>1,692</td>
<td>1,655</td>
<td>Numeric</td>
</tr>
<tr>
<td>Waveform-2</td>
<td>3,347</td>
<td>40</td>
<td>1,692</td>
<td>1,653</td>
<td>Numeric</td>
</tr>
<tr>
<td>Mushroom</td>
<td>8,124</td>
<td>112</td>
<td>3,916</td>
<td>4,208</td>
<td>Numeric</td>
</tr>
<tr>
<td>ijcnnl</td>
<td>35,000</td>
<td>22</td>
<td>3,415</td>
<td>31,585</td>
<td>Numeric</td>
</tr>
<tr>
<td>bank-full</td>
<td>45,211</td>
<td>16</td>
<td>39,922</td>
<td>5,289</td>
<td>Mixed</td>
</tr>
<tr>
<td>cod-rna</td>
<td>59,535</td>
<td>8</td>
<td>19,845</td>
<td>36,690</td>
<td>Numeric</td>
</tr>
</tbody>
</table>
The rows \textit{LibSVM} correspond to the results obtained with the method with the same name using the whole training set.

\textbf{C. Results and discussion}

<table>
<thead>
<tr>
<th>Dataset (size)</th>
<th>Method</th>
<th>\textit{Size}_{red} (avg.)</th>
<th>\textit{T}_{training} (ms)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris-setosa (70)</td>
<td>DSDT ((\sigma = 0.01))</td>
<td>59.20</td>
<td>3.51</td>
<td>92.85</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>3.77</td>
<td>100.00</td>
</tr>
<tr>
<td>Iris-versicolor (70)</td>
<td>DSDT ((\sigma = 0.99))</td>
<td>22.88</td>
<td>1.94</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>2.73</td>
<td>100.00</td>
</tr>
<tr>
<td>Iris-virginica (70)</td>
<td>DSDT ((\sigma = 0.99))</td>
<td>22.20</td>
<td>2.79</td>
<td>100.00</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>3.71</td>
<td>100.00</td>
</tr>
<tr>
<td>Ionosphere (246)</td>
<td>DSDT ((\sigma = 0.70))</td>
<td>99.72</td>
<td>28.70</td>
<td>92.92</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>34.54</td>
<td>93.58</td>
</tr>
<tr>
<td>Diabetes (538)</td>
<td>DSDT ((\sigma = 0.1))</td>
<td>398.23</td>
<td>48.38</td>
<td>67.0</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>110.00</td>
<td>66.41</td>
</tr>
<tr>
<td>Waveform-0 (2,316)</td>
<td>DSDT ((\sigma = 0.5))</td>
<td>914.55</td>
<td>381.11</td>
<td>93.92</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>942.50</td>
<td>92.94</td>
</tr>
<tr>
<td>Waveform-1 (2,343)</td>
<td>DSDT ((\sigma = 0.5))</td>
<td>933.44</td>
<td>481.77</td>
<td>91.90</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>1,308.90</td>
<td>90.65</td>
</tr>
<tr>
<td>Waveform-2, (2,342)</td>
<td>DSDT ((\sigma = 0.5))</td>
<td>925.33</td>
<td>481.88</td>
<td>92.61</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>1,096.00</td>
<td>89.88</td>
</tr>
<tr>
<td>ijcnn1 (24,500)</td>
<td>DSDT ((\sigma = 0.4))</td>
<td>4,818.60</td>
<td>9,149.65</td>
<td>91.06</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>22,342.00</td>
<td>92.13</td>
</tr>
<tr>
<td>bank-full (31,647)</td>
<td>DSDT ((\sigma = 0.6))</td>
<td>6,474.90</td>
<td>38,274.06</td>
<td>88.62</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>mem error</td>
<td>mem error</td>
<td></td>
</tr>
<tr>
<td>svmguide3 (870.09)</td>
<td>DSDT ((\sigma = 0.3))</td>
<td>65.00</td>
<td>217.00</td>
<td>76.35</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>360.00</td>
<td>78.82</td>
</tr>
<tr>
<td>cod-rna (41,675)</td>
<td>DSDT ((\sigma = 0.6))</td>
<td>14,529.00</td>
<td>59,800.75</td>
<td>88.74</td>
</tr>
<tr>
<td></td>
<td>Libsvm</td>
<td>-</td>
<td>393,554.00</td>
<td>89.13</td>
</tr>
</tbody>
</table>

The Table II shows the comparative between \textit{LibSVM} and the DSDT method. It can be seen that for all the experiments the DSDT reduces the training time of SVM. The achieved accuracy is improved for some data sets. For some training sets the accuracy is degraded although acceptable.

For small size data sets it is no necessary to apply a data reduction method such as DSDT. The main motivation to apply our method on small data sets was to explore why the accuracy of the proposed method is sometimes degraded with respect to results obtained when using the entire training set.

Taking as particular example the Iris-setosa set, the proposed method gave the worst result in accuracy. Examining both the original training set (see figure 5) and the reduced version after applying the proposed method, it can be verified that the method successfully detects examples that are close to the decision boundary (see figure 5) and discards those elements that are far from it. In general if there are regions where classes overlap, the accuracy is degraded. Because in practice most data sets have this characteristic, it is necessary to adjust the parameters of QPP solver to improve accuracy.

The method removes examples that have less chance to contribute to define the optimal separating hyperplane, during this process some SV can be accidentally deleted. Although there was not used in the presented results of Table II, by adjusting the parameters \(\gamma\), penalty factor \(C_i\) in QPP solver the accuracy of classifier can be improved. For example, using \(\gamma = 0.85\) and \(C = 2\), the achieved accuracy for Iris-setosa is 94.81\%. This effect is also presented in other data reduction methods.

The parameter \(\sigma\) can be easily adjusted using the grid method. The values \(\sigma = 0.1, 0.2, \ldots, 0.9\) were tested to select the ones that met the best training time and accuracy. For the modified Iris data sets, this value was calibrated using a finer grid, because the training times are very short.

![Fig. 5. Class Distribution for Iris-setosa Data set](image1)

The results become more noticeable with the larger data sets. The accuracy is competitive to the achieved with \textit{LibSVM} using all training example. The cases where the accuracy is slightly degraded can be attributed to the fact that some SV were not included during the selection phase.

\textbf{VI. Conclusions}

SVM is an important classifier with a good generalization power and strong mathematical background. A well known drawback of SVM is that its training time is costly, therefore...
methods to tackle this drawback have been proposed in last years.

In this work a novel data reduction method was developed, the method is based on the observation that separating boundaries of partitions discovered by decision trees can be linearly separable. Classification of linearly separable sets is equivalent to solving a closest pair of points which is a QPP, but solving it is computationally expensive.

In order to avoid solving costly optimization problems, the proposed method apply random sampling, however different from other methods that use simple random sampling, we guide the selection giving more chance to be selected to those examples that are on the boundaries of clusters discovered by a decision tree.

Experiments on different data sets commonly used for classification task show the defectiveness of the proposed method. The method gives best results when the training sets are large.

We are currently working on several fronts, we are studying how to improve the accuracy of classifier by detecting SV. Working with (opposite class) neighbors of a cluster and then selecting the closest examples is an underlying idea of such improvement. We are also analyzing how to adapt the method to the VFDT and enable our method to work on data stream classification.

REFERENCES


