Pattern Recognition 43 (2010) 3977–3987

Contents lists available at ScienceDirect

Pattern Recognition

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

A hybrid SVM based decision tree

M. Arun Kumar*, M. Gopal

Control group, Department of Electrical Engineering, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India

ARTICLE INFO

Article history:
Received 14 July 2009
Received in revised form
1 June 2010
Accepted 10 June 2010

Keywords:
Support vector machines
Decision trees
Hybridization
Pattern recognition

ABSTRACT

We have proposed a hybrid SVM based decision tree to speedup SVMs in its testing phase for binary classification tasks. While most existing methods addressed towards this task aim at reducing the number of support vectors, we have focused on reducing the number of test datapoints that need SVM's help in getting classified. The central idea is to approximate the decision boundary of SVM using decision trees. The resulting tree is a hybrid tree in the sense that it has both univariate and multivariate (SVM) nodes. The hybrid tree takes SVM's help only in classifying crucial datapoints lying near decision boundary; remaining less crucial datapoints are classified by fast univariate nodes. The classification accuracy of the hybrid tree is guaranteed by tuning a threshold parameter. Extensive computational comparisons on 19 publicly available datasets indicate that the proposed method achieves significant speedup when compared to SVMs, without any compromise in classification accuracy.

1. Introduction

Support vector machines (SVMs), being computationally powerful tools for supervised learning, are widely used in classification and regression problems. The approach is systematic and motivated by statistical learning theory (SLT) and Bayesian arguments. The central idea of SVM classifier is to find the optimal separating hyperplane between positive and negative examples. The optimal hyperplane is defined as the one giving maximum margin between training examples that are closest to the hyperplane. SVM classifiers have been successfully applied to a variety of real-world problems like handwritten digits classification, particle identification, face recognition, text categorization and bioinformatics [1]. SVMs enjoy better generalization than many other classification techniques in these applications, but this improved generalization comes with a cost. SVMs are considerably slower in testing phase than other techniques. This is because the computational complexity of SVM's decision function scales with respect to the number of support vectors. Hence if the number of support vectors is very large, SVMs will take more time to classify a new datapoint.

Speeding up SVMs in testing phase has been an active research area for almost a decade and many solutions are already available in literature. Burges [2] proposed simplified SVM, which computes an approximate decision function based on reduced set of vectors. These reduced set of vectors are generally not support vectors. Burges achieved impressive results on NIST dataset with his method; however, the method proved to be computationally expensive and the approach was not pursued any further [3]. The method was also used to approximate decision function obtained by virtual support vector method in [4]. Osuna and Girosi [5] presented two approaches for decreasing the number of support vectors. One approach is to approximate the SVM's decision function using SVM regressor and the other is to solve a primal reformulation of the SVM optimization problem. Downs et al. [3] proposed an approach to discard unnecessary support vectors using linear dependence, keeping the SVM decision function unchanged. Li and Zhang [6] reduced the number of support vectors by iterative learning. Li et al. [7] proposed an adaptive algorithm based on vector correlation to reduce the number of support vectors.

It could be seen that almost all methods aim at reducing the number of support vectors to decrease testing time. In this paper, we propose to approximate the decision boundary of SVM using decision tree (DT) to speedup SVM in its testing phase. We call this approach SVM based DT (SVMDT), as several nodes of the DT are binary SVM. In SVMDT a single binary SVM is trained once and is positioned in some of the (multiple) leafs of the DT. SVMDT approach is different from the above mentioned works in the sense that, instead of attempting to reduce the number of support vectors, focus is on reducing the number of test datapoints that require SVM's decision. It uses both SVM and DT to achieve fast classification without any compromise in classification accuracy. Only crucial test datapoints which lie close to SVM's decision boundary are classified using SVM nodes; remaining less crucial test datapoints are classified using much faster univariate nodes.

* Corresponding author. Present address: Controls & Optimization Research Group, ABB Global Industries & Services Ltd., Bangalore. Tel: +91 9740219695; fax: +91 11 26581606.
E-mail address: sendtoarun@rediffmail.com (M. Arun Kumar).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.06.010

Thus only small portion of test datapoints need SVM's help in getting classified, and hence testing time of SVMDT drastically decreases.

The paper is organized as follows: In Section 2, we present a comparison study on SVM and DT. Based on the observations made in Section 2 we present the idea of SVMDT in Section 3. Experimental results are presented and discussed in Section 4 and Section 5 gives concluding remarks.

2. A comparison study on SVM and DT

In order to gain insight into how SVMs and DTs work, we carried out some trial experiments on eight commonly used subsets of adult dataset [8]. Table 1 gives a brief description of these eight subsets of adult datasets. These subsets were compiled by Platt [9] from UCI adult dataset. The objective is to predict whether a household has an income greater than \$50 k. We used Joachim's SVM^{light} [10] algorithm with Gaussian kernel to train SVM binary classifiers. The penalty parameter C and kernel parameter μ of SVM were tuned using a small validation dataset consisting of 30% of the training set. For DT, we used Quinlan's C4.5 algorithm implemented in Weka [11] with default parameter setting. Fig. 1(a) shows the testing accuracy comparison of SVM and DT on these eight datasets. It could be observed that, on all eight datasets SVMs performed better than DTs by at least 1%. Fig. 1(b) presents the testing time taken by SVM and DT over these eight datasets. From Fig. 1(b) it is clear that DTs are very fast when compared to SVMs in testing new datapoints. An interesting observation from SVM results is: the testing time increases when we proceed from adult1 to adult7 datasets, inspite of decreasing number of test datapoints. This is because, as we proceed from adult1 to adult7 datasets, the training size increases and hence the number of support vectors also increases (shown in Fig. 1(c)). Thus the increasing number of support vectors dominate over decreasing number of testing datapoints, thereby increasing the testing time of SVMs from adult1 to adult7 datasets. In fact, decreasing number of test datapoints seems to dominate over support vectors only on adult8 dataset. On the contrary, the testing time of DT decreases as we proceed from adult1 to adult8 datasets. As this is not clear in Fig. 1(b) we have plotted testing time variations of DT separately in Fig. 1(d). In fact with decision trees also, the size of the tree (total number of decision nodes+leafs) increases as we move from adult1 to adult8 (shown in Fig. 1(e)), similar to increasing number of support vectors in SVM. However, the decreasing number of testing instances dominates the increasing tree size and hence the testing time of DT decreases.

The outcomes of these experiments are:

(1) DTs are much faster than SVMs in classifying new instances.
(2) SVMs perform better then DTs in terms of classification accuracy.

Table 1
Description of subsets of adult dataset.

Dataset	# train	# test	# features
Adult1	1605	30956	123
Adult2	2265	30296	123
Adult3	3185	29376	123
Adult4	4781	27780	123
Adult5	6414	26147	123
Adult6	11220	21341	123
Adult7	16100	16461	123
Adult8	22696	9865	123

The first outcome is obvious, as testing a datapoint in DT just involves series of logical operations when compared to complex kernel operations (arithmetic) required by SVM. Testing a datapoint with SVM requires $O(MN_{sv})$ operations, where N_{sv} represents the number of support vectors and M represents the number of operations required in evaluating the kernel. For Gaussian kernel, M is $O(n)$, where n is the dimension of datapoints [1]. The computational complexity of testing a datapoint with DT depends on the depth of the tree [12]. If we assume equal splits at each node and total number of leaves to be equal to the number of support vectors, the computational complexity of testing a datapoint with DT will only be $O(\log N_{sv})$. However, in practice, total number of leaf nodes are far less when compared to total number of support vectors as indicated by our results. The second outcome is purely an empirical result which was also observed by many authors across various application domains [13–15]. Based on these outcomes, we aim at hybridizing DT and SVM to get benefits of both learning techniques.

3. Hybrid SVM based decision tree

3.1. Motivation and formulation

Based on the observations presented in the previous section, our objective is to hybridize SVM and DT to exploit the advantages of both the learning methodologies. From previous discussion it follows that, if we have a test dataset with *ltest* datapoints, the total computational complexity of testing the dataset with SVM is $ltest \times O(MN_{sv})$. Given the fact that $M=O(n)$, speeding up SVM is possible by decreasing any one of the three parameters: *ltest* or n or N_{sv}. All the algorithms that we have outlined in Section 1 aim to reduce N_{sv}. Any dimensionality reduction method can be used to reduce n. The approach we propose in this paper is an attempt to reduce *ltest* with the help of decision trees. In what follows we demonstrate how this can be achieved with a simple 2-D toy dataset (shown in Fig. 2(a)) for pedagogical reasons. The dataset was created by randomly generating 1000 points in the range [0,1], which was separated into two classes using a sinusoidal curve with mean 0.5. The points below the curve were assigned to class 1 and the points above the curve were assigned to class 2. The dataset also has randomly generated outliers (5%). Fig. 2(b) shows the classification decision boundary obtained using pruned C4.5 DT and Fig. 2(c) shows the classification decision boundary obtained using SVM ($C=2$; $\mu=8$) along with its predictions. It could be seen that SVM's decision boundary is very close to the true decision boundary; whereas DT's decision boundary is an approximate one. Consider for a moment that we are predicting whether a patient has cancer or not, in this toy example. Let class 1 datapoints represent patients with cancer and class 2 datapoints represent patients without cancer. Under this assumption it can be seen that, all datapoints near decision boundary represent crucial patients who need careful examination before arriving at a decision. As it is difficult to arrive at a decision, it is worth seeking advice from an expert who can analyze the case with all known symptoms and test results (features). For these crucial patients we will not be expecting a quick decision; rather we will expect an accurate decision. On the other hand consider datapoints far away from decision boundary representing patients who are not as crucial when compared with datapoints near decision boundary. Decision making is a lot easier when compared to previous case. Under this scenario, first of all, it is not required that we should consult the expert to arrive at a decision; rather we can take decisions from expert's assistants. This is based on the assumption that: the expert analyzes all symptoms and test

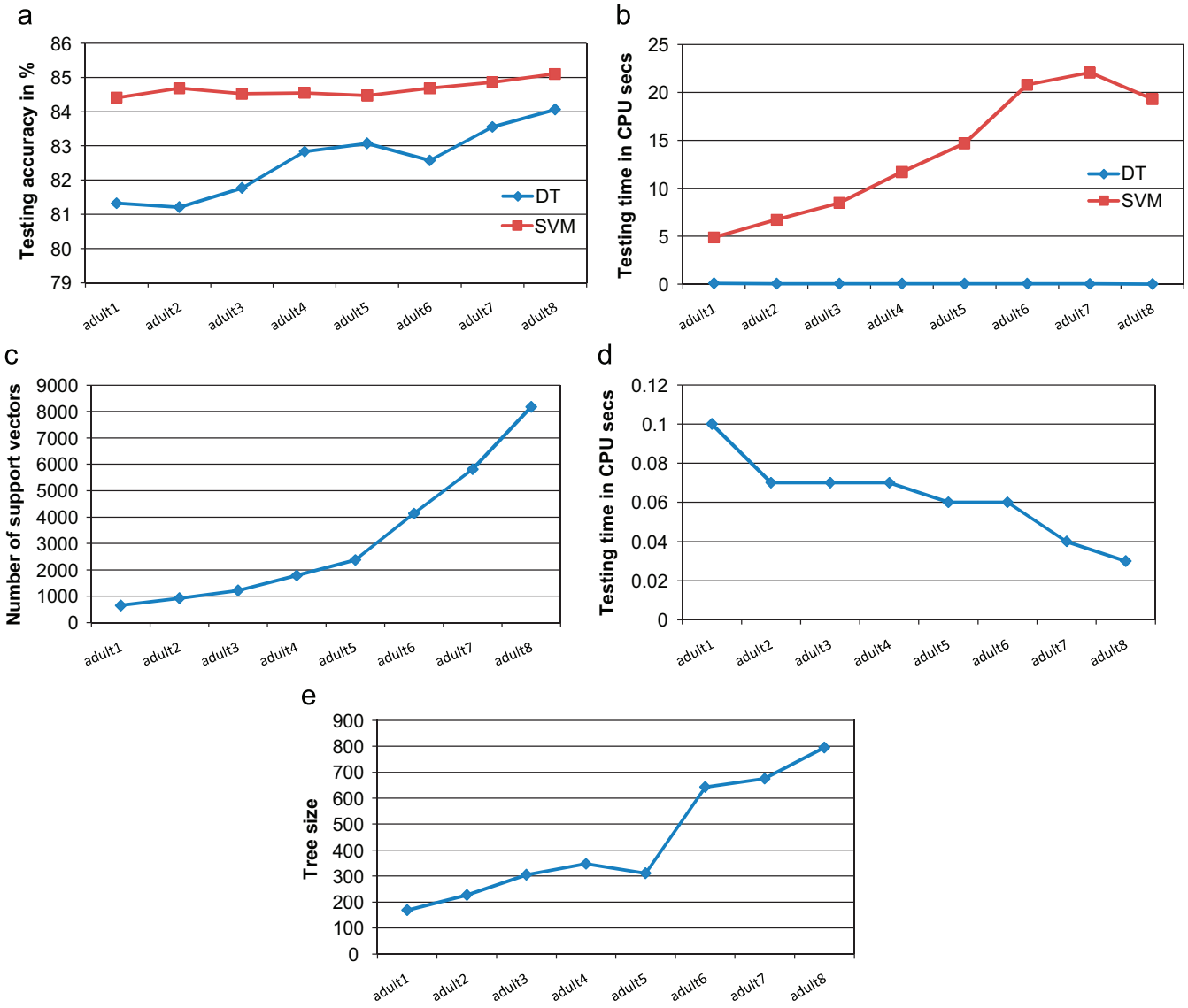

Fig. 1. (a) Testing accuracy comparison of SVM and DT, (b) testing time comparison of SVM and DT, (c) numbers of support vectors of SVM, and (d) testing time of DT.

results, and takes always fixed time to arrive at a decision, irrespective of whether the case is crucial or not.

Thus if we use SVM to classify all test datapoints, the computational complexity of testing a datapoint will remain fixed, irrespective of whether the datapoint is crucial or not. Hence we propose SVMDT in which, we use SVM only when a test datapoint is closer to its decision boundary. Test datapoints that are not closer to SVM's decision boundary will be predicted using much faster univariate decisions. In other words, we use DT to approximate a crucial region around SVM's decision boundary together with less crucial class 1 and class 2 regions.

3.2. SVMDT

In this paper, we use DT to identify three regions around SVM's decision boundary. Let us consider a SVM decision function

$$f(x)=\sum_{i=1}^{N_{sv}}\alpha_i D_{ii}K(x,X_i)+b \quad (1)$$

where $X_i \in \Re^n$ is a support vector, D_{ii} represents its corresponding target ($D_{ii}=1$ for class 2 datapoints and $D_{ii}=-1$ for class 1 datapoints), α_i represents Lagrangian multiplier ($0<\alpha_i<C$), $b \in \Re$ is bias, N_{sv} represents the number of support vectors, K represents the kernel function and $x \in \Re^n$ is the new datapoint to be tested. Given SVM's decision function $f(x)$, its decision boundary is defined as $f(x)=0$. This decision boundary dichotomizes the feature space into two mutually exclusive regions, positive region $f(x)>0$ (consisting of all training datapoints predicted as class 2) and negative region $f(x)<0$ (consisting of all training datapoints predicted as class 1). Based on this decision boundary, we define a closeness measure $S(x)$ such that, it will give small values for datapoints closer to $f(x)=0$ and large values for datapoints away from $f(x)=0$. By selecting a threshold parameter δ, we define a δ-region around the decision boundary, such that this δ-region will include all training datapoints with closeness measure $S(x)\leq\delta$. For the toy example under consideration, this δ-region is shown in Fig. 3(a). We re-label predictions for all training datapoints inside δ-region as class 3. Now the feature space will have training datapoints along with its predictions, representing three different regions (as shown in Fig. 3(b)):

(a) δ-region: This region consists of training datapoints (predicted as class 3) that are closer to SVM's decision boundary with closeness measure $S(x)\leq\delta$. Any test datapoint falling in

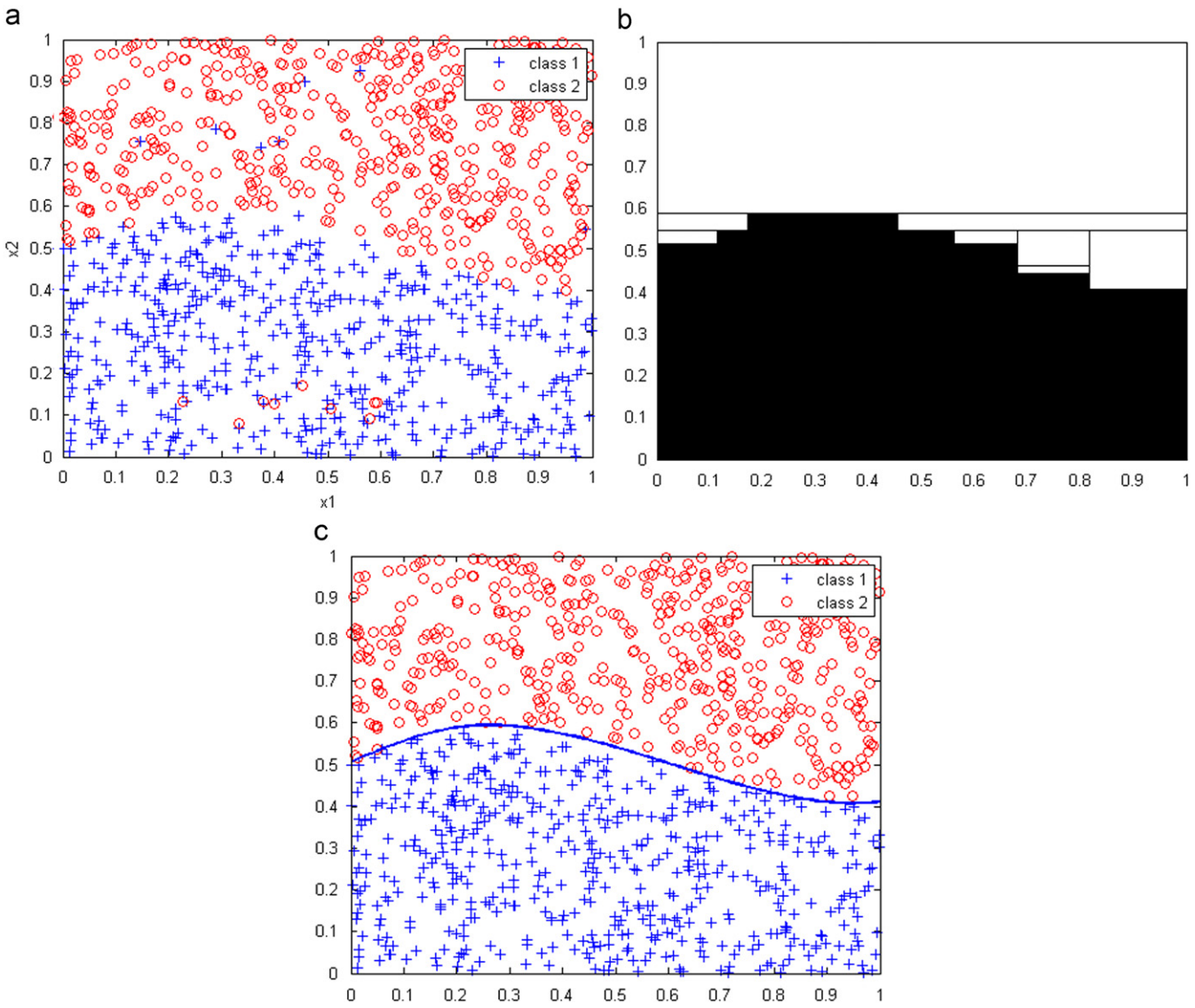

Fig. 2. (a) 2-D toy dataset, (b) decision boundary obtained by DT, and (c) decision boundary obtained by SVM.

this region is considered crucial and we expect SVMDT to classify this datapoint using one of its SVM nodes.

(b) New positive region: This region consists of training datapoints (predicted as class 2) on the positive side of decision boundary ($f(x)>0$) with closeness measure $S(x)>\delta$. Any test datapoint falling in this region is considered less crucial when compared to test datapoints in δ-region and hence we expect SVMDT to classify it using fast univariate decision nodes bypassing SVM nodes.

(c) New negative region: This region consists of training datapoints (predicted as class 1) on the negative side of decision boundary ($f(x)<0$) with closeness measure $S(x)>\delta$. Any test datapoint falling in this region is expected to be classified by SVMDT using fast univariate decision nodes bypassing SVM nodes.

Once training datapoints with its predictions are available to represent these three regions, SVMDT can be obtained in two steps. First we train a DT with this 3-class dataset to approximately identify these three regions. Fig. 3(c) shows the three regions identified by C4.5 DT for the toy dataset under consideration. After training DT, the second step is to replace each class 3 leaf by a subtree with binary SVM and two leaves as shown in Fig. 4. This gives our final SVMDT where a single binary SVM trained once is positioned in several leafs of the DT. Fig. 5 shows the final SVMDT obtained for the toy dataset. In Fig. 5, $\times 1$ and $\times 2$ represent the two features and the value inside brackets indicates the corresponding split value at the node. SVMDT consists of both conventional univariate decision nodes and multivariate decision nodes (SVMs). The univariate decision nodes help in arriving at a quick decision for less crucial test datapoints without consulting the multivariate SVM. On the other hand if test datapoints are crucial, univariate decision nodes direct them to multivariate SVM. Hence only a small portion of test datapoints gets classified with SVM nodes and the rest are classified using much faster univaraite decision nodes, thereby decreasing the overall testing time.

3.3. Closeness measure $S(x)$

For identifying the δ-region, a closeness measure between training datapoints and decision boundary $f(x)=0$ is to be defined. We propose to use probabilistic output of SVM as closeness measure in this paper. In general, the SVM decision function $f(x)$ outputs uncalibrated values and can be converted to posterior probability estimates by fitting a sigmoidal function at its output. A discussion on different types of probabilistic outputs for SVM

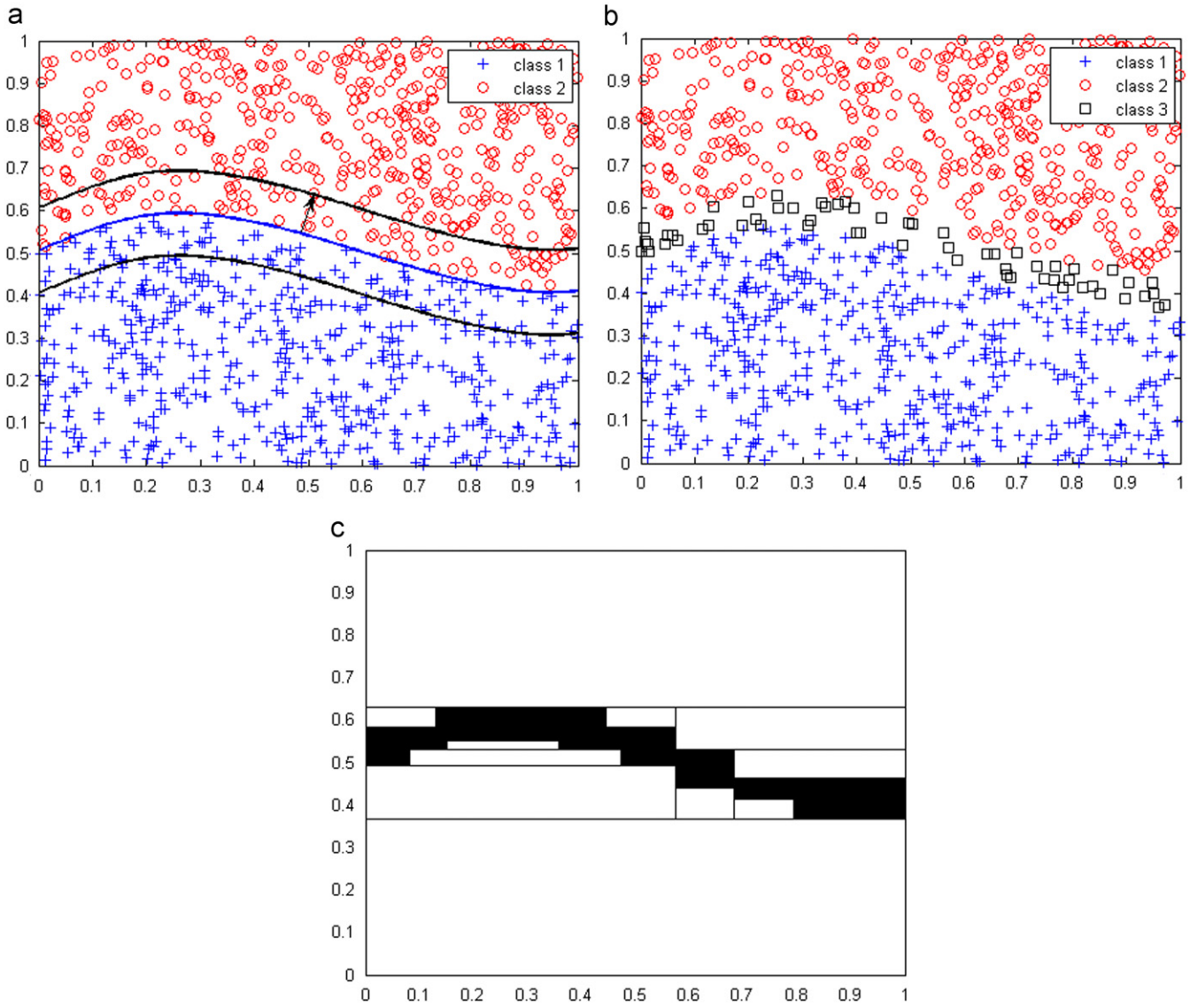

Fig. 3. (a) $\delta-$region around decision boundary, (b) datapoints representing three regions around decision boundary, and (c) three regions identified by DT.

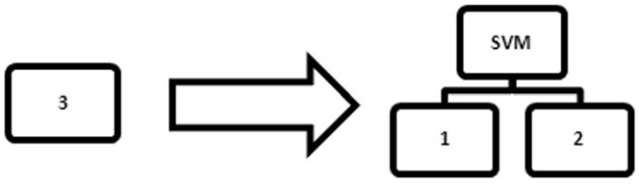

Fig. 4. SVMDT—second step.

can be found in [16]. Following [16], we adopted the posterior probability estimate given by

$$P(\text{Class } 2|f(x)) = \frac{1}{1+\exp(-f(x))}$$
$$P(\text{Class } 2|f(x)) = 0.5 \quad \text{when } f(x) = 0 \tag{2}$$

The above expression can be modified as

$$\Delta P(x) = P(\text{Class } 2|f(x)) - 0.5 = \frac{1}{1+\exp(-f(x))} - 0.5 \tag{3}$$

where $|\Delta P(x)|$ indicates the closeness measure $S(x)$ between training datapoints and decision boundary $f(x)=0$. It could be noted that, for $f(x)=0$; $S(x)=0$, when $f(x)\to\infty$; $S(x)\to 0.5$, and when $f(x)\to-\infty$; $S(x)\to 0.5$. Thus, $f(x)$ in the range $(-\infty,\infty)$ gets mapped to $S(x)$ in the range [0,0.5), which also becomes a valid range for selection of threshold parameter δ.

3.4. SVMDT algorithm

We now give an explicit statement of the algorithm for our SVMDT approach. Description of data structures used in the algorithm is as follows:

Train_data: set of training datapoints
Train_target: corresponding target for Train_data
New_target: targets to be used for DT training

Algorithm 3.1. SVMDT training

Given a binary dataset, SVMDT training can be performed with the following steps:

(a) Select a kernel function (Gaussian/poly, etc.) with parameter μ and penalty parameter C for SVM. Usually these parameters are selected based on validation.

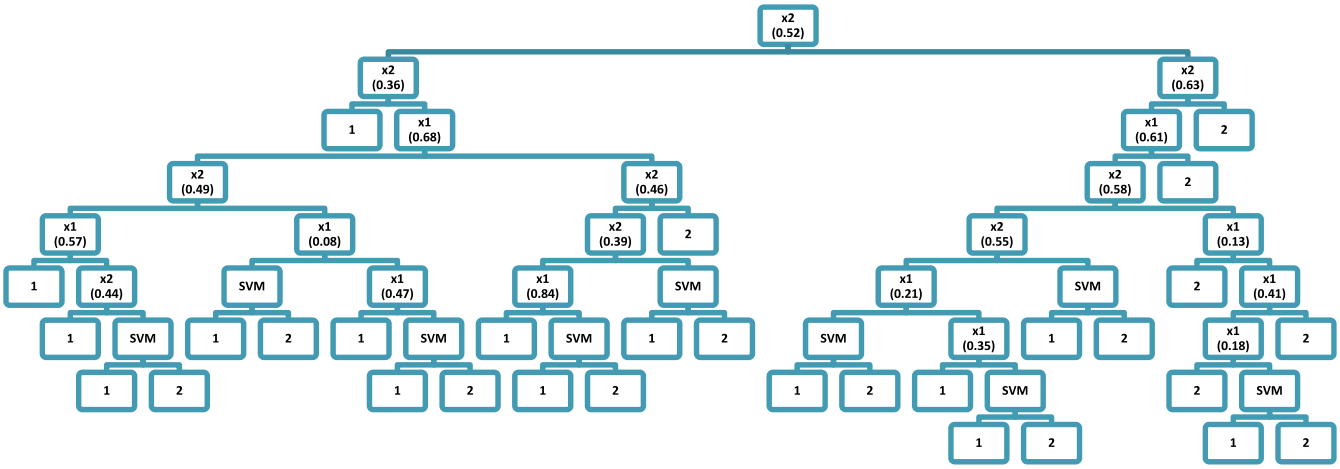

Fig. 5. SVMDT for the toy dataset.

(b) Train SVM classifier with Train_data and Train_target to obtain its decision function $f(x)$.
(c) Classify Train_data with $f(x)$ into class 1 or class 2. Save these predictions in New_target.
(d) Select a threshold value δ between 0 and 0.5.
(e) Identify datapoints in Train_data with $S(x) \leq \delta$ and change their corresponding predictions in New_target to class 3.
(f) Train a DT with Train_data and New_target.
(g) Replace all class 3 leaves of DT by a subtree with SVM and two leaves as shown in Fig. 4.
(h) Save the tree and return.

3.5. Significance of the threshold δ

The threshold δ plays significant role in the proposed SVMDT approach. It can vary between 0 and 0.5. Below, we will give special attention to SVMDT with three values of δ.

$\delta=0$: It is easy to see that, when $\delta=0$, there is no δ-region and hence SVMDT will have only univariate nodes. In this case SVMDT just boils down to a conventional DT. As all test datapoints will be classified by univariate decision nodes, we can expect SVMDT with $\delta=0$ to be the fastest classifier among family of SVMDTs with all possible values of δ. Though SVMDT with $\delta=0$ is a conventional DT, it is to be remembered that it is different from a DT trained for the same dataset. This is because SVMDT is trained with different targets (predictions given by SVM) as opposed to original targets with which DT will be trained.

$\delta=0.5$: When $\delta=0.5$, δ-region covers the entire feature space and hence SVMDT will have single SVM node with two leaves. In this case thus SVMDT just boils down to conventional SVM classification. As all test datapoints will be classified by SVM, we can expect SVMDT with $\delta=0.5$ to be the slowest classifier among family of SVMDTs with all possible values of δ.

$\delta=0.231$: An interesting observation is: for support vectors, $f(x)=\pm 1$ and $S(x)=0.231$; hence selecting $\delta=0.231$ will make the δ-region to coincide with the margin area around SVM's decision boundary. Hence we can expect that SVMDT will classify all test datapoints falling inside margin area with SVM nodes and remaining datapoints outside margin area with univariate nodes.

As δ increases from 0 to 0.5, the δ-region becomes wider and hence more and more test datapoints will be classified by SVM nodes in SVMDT. Thus the threshold δ acts as a soft switch between, classification using DT and classification using SVM.

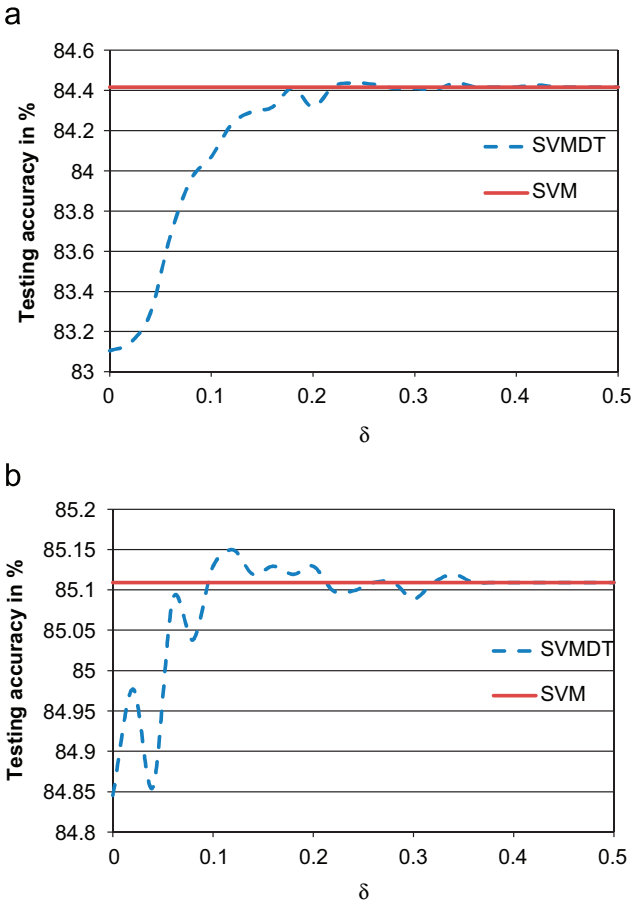

Fig. 6. Testing accuracy vs. threshold δ for SVMDT: (a) adult1 and (b) adult8.

Hence by varying δ, we can control the testing time of SVMDT. In general, small δ will witness less testing time and large δ will take more testing time. Also as δ moves from 0 to 0.5, the testing accuracy of SVMDT moves from testing accuracy of DT to testing accuracy of SVM. While mathematical analysis of testing accuracy variation with respect to threshold δ is out of the scope of this paper; taking empirical observations made in Section 2 into consideration, one can expect increase in testing accuracy for δ varying from 0 to 0.5. Thus the testing accuracy of SVMDT is

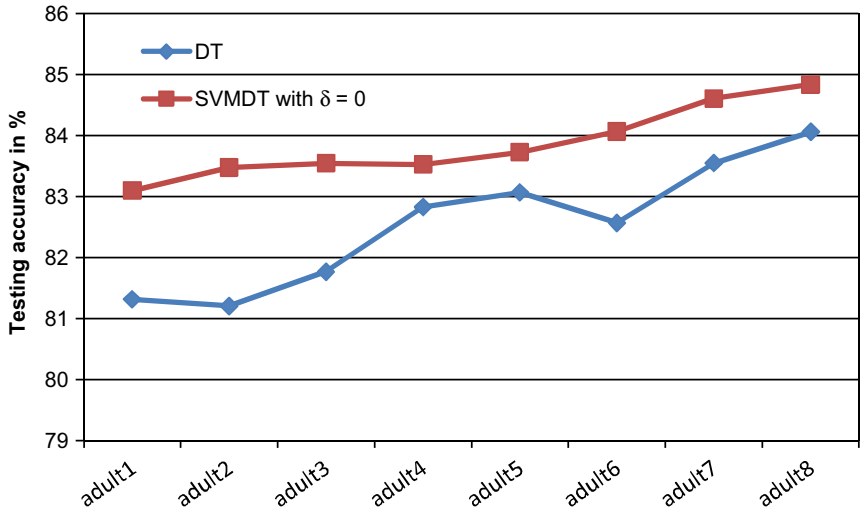

Fig. 7. Testing accuracy comparisons of DT and SVMDT with $\delta=0$.

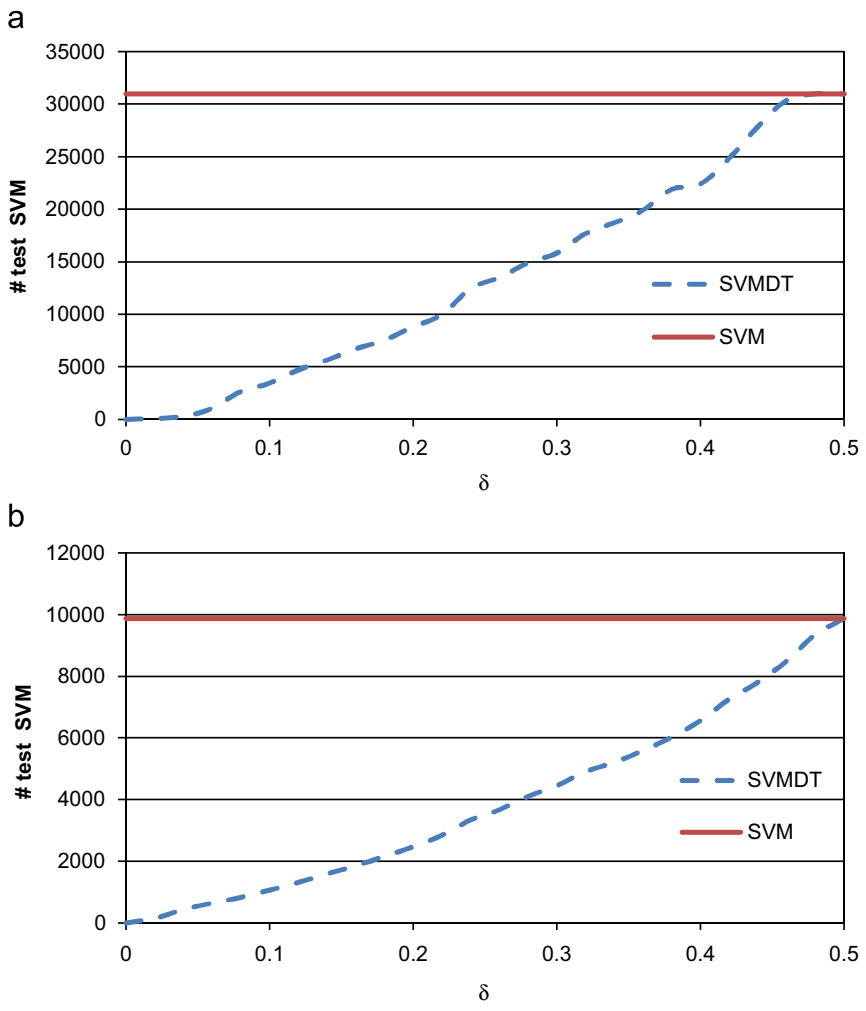

Fig. 8. Number of test datapoints classified using SVM nodes in SVMDT vs. threshold δ.: (a) adult1 and (b) adult8.

guaranteed by tuning the threshold δ. Also, it is to be noted that parameter tuning (in some form) is very common and inherent among the approaches that aim to speedup SVMs. For example, in simplified SVM [2,4] the number of reduced set of vectors is to be selected a priori, in the work of Li and Zhang [6], the acceptable reduction in training accuracy is to be defined a priori, and in the work of Li et al. [7] the approximation error to control the number of the feature vectors is to be tuned.

4. Experimental results

4.1. Adult datasets

To demonstrate the performance of SVMDT, we conducted several experiments on real-world datasets. First we will discuss the results on eight subsets of adult dataset introduced in Section 2. All experiments were implemented in MATLAB 7.3.0 (R2006b) [17] environment on a PC with Intel Core2Duo processor (2.13 GHz), 1 GB RAM. We used SVM^{light} [10] with Gaussian kernel and C4.5 of Weka [11] for training SVMDT. We used the same values of C and μ tuned for SVM, with SVMDT (for SVM nodes). To gain insight into the effect of threshold δ on SVMDT, we trained a family of SVMDTs for δ varying between 0 and 0.5 in steps of 0.02. Fig. 6 presents the testing accuracy of this family of SVMDTs for adult1 and adult8 datasets. In both the figures, SVMDT with $\delta=0$ has the least testing accuracy (83.10 & 84.84%, respectively); however, they exceed the corresponding testing accuracies of C4.5 DT (81.32 & 84.06%). Though SVMDT with $\delta=0$, is also a conventional DT without any SVM nodes, its accuracy is better than C4.5 tree because SVMDT is generated from a dataset guided by SVM. SVM acts as data pre-processor for the DT thereby filtering any noise/re-labeling outliers. In fact on all the eight adult datasets, SVMDT with $\delta=0$ was having the least testing accuracy among the family of SVMDTs and it outperformed C4.5's testing accuracy (this is shown in Fig. 7). Returning to Fig. 6, it could be observed that, in general the testing accuracy of SVMDT increases with increase in δ and the testing accuracy of SVMDT approaches testing accuracy of SVM when δ is close to 0.5. Fig. 8 shows the influence of δ over number of test datapoints classified by SVM nodes in SVMDT. Fig. 9 presents variations in testing time with respect to threshold δ. These figures together with Fig. 6 personify our discussion on threshold δ (Section 3.5). In fact we could observe similar patterns on all other datasets as well.

These illustrations presented above throw light on the influence of threshold δ over performance of SVMDT algorithm. We now present a tabular form of numerical results comparing these algorithms in Table 2. On each dataset, we report SVMDT results for the smallest value of δ, such that its testing accuracy is closest with testing accuracy of SVM. Table 2 gives the following details: training accuracy (Train %), testing accuracy (Test %), SVM parameters C and μ, threshold δ used in SVMDT, number of support vectors (# SV), number of test datapoints predicted by SVM/SVM nodes (# ts. SVM), testing time taken on test dataset (ts. time) and reduction rate (r. rate). We defined reduction rate as an index to measure the amount of reduction in testing time

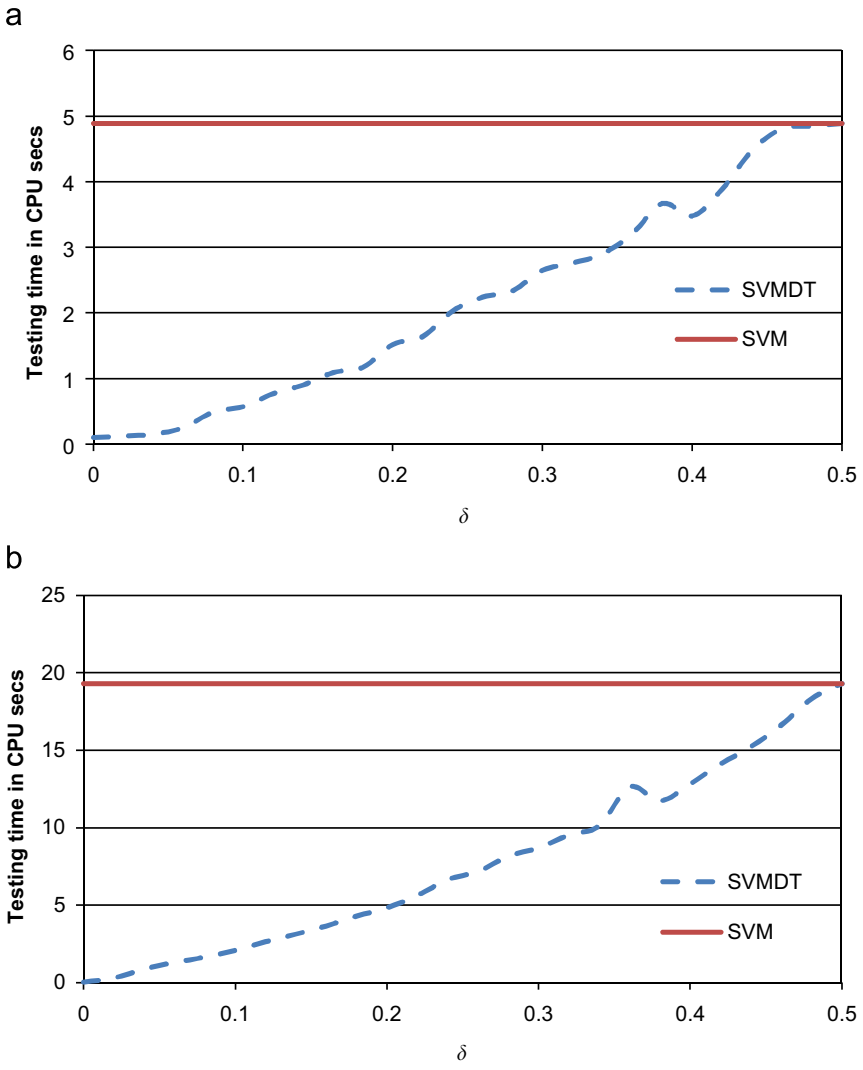

Fig. 9. Testing time of SVMDT vs. threshold δ.: (a) adult1 and (b) adult8.

Table 2
Comparison results of DT, SVM, and SVMDT on eight subsets of adult dataset.

Details		DT	SVM	SVMDT		DT	SVM	SVMDT
Train (%)	Adult1	91.02	85.29	85.23	Adult5	87.93	85.18	85.18
Test (%)		81.32	84.41	84.41		83.07	84.47	84.47
C,μ		–	8, 0.0078	8, 0.0078		–	8, 0.0078	8, 0.0078
δ		–	–	0.18		–	–	0.26
# SV		–	649	649		–	2376	2376
# ts. SVM		–	30 956	7535		–	26 147	10 355
ts. time		**0.1**	**4.89**	**1.17**		**0.06**	**14.7**	**5.58**
r. rate (%)		–	**0**	**76.07**		–	**0**	**62.04**
Train (%)	Adult2	90.11	85.12	85.16	Adult6	89.19	84.78	84.74
Test (%)		81.21	84.68	84.67		82.57	84.68	84.70
C,μ		–	16, 0.0078	16, 0.0078		–	32, 0.001	32, 0.001
δ		–	–	0.24		–	–	0.14
# SV		–	926	926		–	4131	4131
# ts. SVM		–	30 296	11 906		–	21 341	3493
ts. time		**0.07**	**6.73**	**2.64**		**0.06**	**20.8**	**3.65**
r. rate		–	**0**	**60.77**		–	**0**	**82.45**
Train (%)	Adult3	90.51	85.74	85.68	Adult7	88.19	85.05	85.00
Test (%)		81.77	84.52	84.52		83.55	84.86	84.87
C,μ		–	16, 0.0078	16, 0.0078		–	128, 0.001	128, 0.001
δ		–	–	0.24		–	–	0.12
# SV		–	1220	1220		–	5807	5807
# ts. SVM		–	29 376	11 001		–	16 461	2271
ts. time		**0.07**	**8.48**	**3.22**		**0.04**	**22.06**	**3.08**
r. rate		–	**0**	**62.03**		–	**0**	**86.04**
Train (%)	Adult4	89.24	85.48	85.48	Adult8	87.86	84.83	84.84
Test (%)		82.83	84.55	84.55		84.06	85.10	85.12
C,μ		–	16, 0.0078	16, 0.0078		–	128, 0.001	128, 0.001
δ		–	–	0.24		–	–	0.1
# SV		–	1786	1786		–	8172	8172
# ts. SVM		–	27 780	9681		–	9865	1067
ts. time		**0.07**	**11.7**	**4.05**		**0.03**	**19.3**	**2.09**
r. rate		–	**0**	**65.38**		–	**0**	**89.17**

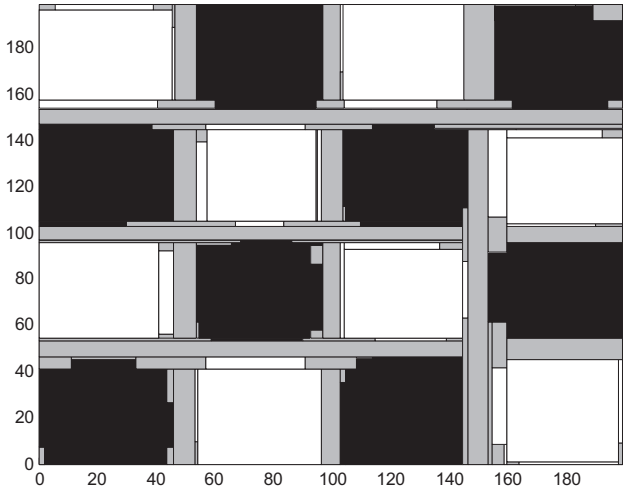

Fig. 10. Three regions identified by SVMDT for checkerboard dataset.

achieved by SVMDT over testing time of SVM. Reduction rate can be expressed as

$$\text{r.ratein\%} = \left(1 - \frac{\text{ts. time of SVMDT}}{\text{ts. time of SVM}}\right) \times 100 \quad (4)$$

From Table 2 it could be observed that, on all the eight datasets, SVMDT could achieve significant reductions in testing time over SVM, for the same testing accuracy. In particular, on adult8 SVMDT achieved its highest reduction rate of 89.17. In Appendix A we have presented more results of SVMDT on adult datasets. Comparing these results with Table 1, it could be observed that by making 0.2–0.3% compromise in testing accuracy, testing time can further be decreased.

4.2. Checkerboard dataset

The checkerboard dataset [18] is often considered to be a highly nonlinear dataset. We generated 5000 black and white datapoints in $\Re^2$ taken from sixteen black and white squares of a checkerboard; we used 4000 datapoints for training and 1000 for testing. Fig. 10 shows the three regions identified by SVMDT with $\delta = 0.24$. The classification accuracy of SVMDT was same as that of SVM; however, only 194 test datapoints were classified by SVM nodes as against the original 1000. This results shows that SVMDT is capable of approximating even highly nonlinear decision boundaries.

4.3. Other binary datasets

We also conducted experiments on 18 binary datasets obtained from various sources [8,19–22] apart from adult datasets. Descriptions of these datasets and comparison results of SVM, SVMDT are presented in Appendix B. The pre-processing step includes encoding all categorical attributes into binary attributes and normalization of continuous attributes to be in the range $[-1,+1]$. For around 16 datasets we conducted ten-fold cross-validation experiments as they were not having test set (or test set was very small). We report average testing accuracy (Test %) and average number of test datapoints predicted by SVM/SVM nodes (# ts. SVM) over ten-fold for these cases. For other two datasets we report standard results with corresponding train sets and test sets. We do not provide testing time for these datasets as testing time was very less because of smaller test sets. Instead we provide # ts. SVM as an indicator for test computational

complexity. It is easy to see that, testing time of SVM and SVMDT is proportional to # ts. SVM with # SV and n remaining fixed for any given dataset. These results corroborate the observations made in previous sections over a wide variety of datasets. It is also interesting to see that on two datasets, SVMDT with $\delta=0$ gave almost the same testing accuracy as SVM; which means all test datapoints get classified with univariate nodes.

4.4. Comparison study

In this section we present comparison results of SVMDT with recently proposed feature vector selection (FVS) algorithm [7] for reducing the number of support vectors. FVS algorithm is an adaptive algorithm that selects feature vectors from support vectors solutions based on vector correlation principle. We conducted experiments on five datasets namely cmc-2 [8], German [8], Ripley [20], svmguide1 [19] and svmguide3 [19] used in the previous section. All experiments were carried out in MATLAB 7.3.0 (R2006b) [17] environment on a PC with Intel Core2Duo processor (2.13 GHz), 1 GB RAM. SVM optimization problems were solved using fast interior point solvers of Mosek optimization toolbox for MATLAB [23] and C4.5 of Weka [11] was used for training SVMDT. Appendix C presents these results. For ten-fold datasets we report average testing accuracy (Test %), average testing time (ts. time), average number of test datapoints predicted by SVM/SVM nodes (# ts. SVM), and average number of support vectors (#SV). For the other two datasets we report standard results with corresponding train sets and test sets. For FVS algorithm we tuned the approximation error parameter similar to threshold of SVMDT as suggested in [7]. We have also presented the number of feature vectors (#FV) for FVS algorithm corresponding to the approximation error. It could be observed that on all datasets SVMDT achieved the least testing time when compared to SVM/FVS. At this point it is worth mentioning that any of the existing methods that aim to decrease the number of support vectors (similar to FVS) can be used along with SVMDT to further speedup SVM classification. This could be achieved by just replacing the true SVM decision function present in SVM nodes of SVMDT with the reduced ones retuned by any of the support vector reduction algorithm. Thus SVMDT is a novel way of decreasing testing time of SVMs and it does not contradict with the existing approaches.

5. Conclusion and future work

In this paper we have proposed a hybrid SVM based DT (SVMDT) to speedup SVMs in its testing phase for binary classification tasks. While existing methods to speedup SVMs in

Table A1

Details/Dataset	Adult1		Adult2		Adult3		Adult4	
δ	0.16	0.14	0.22	0.16	0.14	0.12	0.1	0.08
Train (%)	84.98	85.11	84.98	84.72	85.68	85.43	85.21	85.08
Test (%)	84.32	84.29	84.61	84.55	84.50	84.47	84.49	84.25
# ts. SVM	6706	5644	9029	6089	5100	4450	2964	2223
ts. time	**0.99**	**0.8**	**2.08**	**1.4**	**1.52**	**1.31**	**1.21**	**0.93**
r. rate (%)	**79.75**	**83.64**	**69.09**	**79.19**	**82.07**	**84.55**	**89.65**	**92.05**
Details/Dataset	Adult5		Adult6		Adult7		Adult8	
δ	0.16	0.06	0.1	0.06	0.06	0.02	0.06	0.02
Train (%)	85.07	84.76	84.69	84.53	84.84	84.67	84.76	84.53
Test (%)	84.42	84.23	84.65	84.48	84.77	84.67	85.08	84.98
# ts. SVM	4259	1768	2493	1637	1069	230	646	151
ts. time	**2.34**	**1.01**	**2.56**	**1.67**	**1.65**	**0.37**	**1.28**	**0.27**
r. rate (%)	**84.08**	**93.12**	**87.69**	**91.97**	**92.52**	**98.32**	**93.36**	**98.60**

Table B1

Datasets	#train	#test	#features	Source	Test %		# ts. SVM		δ
					SVM	SVMDT	SVM	SVMDT	
Australian	690	–	14	[8]	87.11	87.11	69.0	11.7	0.1
Breast cancer	263	–	9	[22]	75.7	76.09	26.3	2.0	0.06
Breast cancer (Wisconsin)	683	–	10	[8] [21]	97.22	97.22	68.3	9.3	0.26
Class level-KC1	145	–	94	[8]	80.81	82.9	14.5	7.2	0.24
Cmc-2	1473	–	9	[19]	70.88	70.88	147.3	84.4	0.22
Fourclass	862	–	2	[8]	100	100	86.2	63.4	0.36
German	1000	–	24	[8]	76.6	77.1	100.0	30.9	0.16
Heart-statlog	270	–	13	[8]	87.04	87.4	27.0	10.8	0.24
Heart-c	303	–	13	[8]	87.08	87.08	30.3	15.6	0.24
Horsecolic	300	–	27	[19]	96.99	98.67	30.0	0	0
Mushrooms	8124	–	112	[8]	100	100	812.4	0	0
Pima	768	–	8	[20]	77.73	78.0	76.8	33.2	0.16
Ripley	250	1000	2	[8]	90.9	91.1	1000	206	0.2
Spambase	4601	–	57	[19]	94.02	94.0	460.1	82.1	0.24
Svmguide1	3089	4000	4	[19]	97.15	97.1	4000	126	0.12
Svmguide3	1284	–	21	[8]	85.69	86.31	128.4	81.3	0.34
Wpbc	198	–	34	[8]	81.88	83.91	19.8	9.1	0.26
Votes	435	–	16	[8]	97.27	97.27	43.5	4.6	0.3

Table C1

Datasets	Details	SVM	FVS		SVMDT	
Cmc-2	δ/appr. error	–	4e-5	1e-6	0.04	0.22
	Test (%)	**70.88**	**70.61**	**70.88**	**70.53**	**70.88**
	ts. time	3.26	1.44	2.24	0.20	1.81
	# ts. SVM	147.3	147.3	147.3	9.6	84.4
	# SV/#FV	924	491	664	924	924
German	δ/appr. error	–	0.0008	4e-5	0.12	0.16
	Test (%)	**76.6**	**76.3**	**76.6**	**76.5**	**77.1**
	ts. time	1.22	0.53	0.71	0.32	0.37
	# ts. SVM	100.0	100.0	100.0	25.6	30.9
	# SV/#FV	502.4	213.7	280.7	502.4	502.4
Ripley	δ/appr. error	–	0.15	0.05	0.1	0.2
	Test (%)	**90.9**	**90.8**	**90.9**	**90.7**	**91.1**
	ts. time	3.79	0.91	1.21	0.26	0.58
	# ts. SVM	1000	1000	1000	81	206
	# SV/#FV	147	40	54	147	147
Svmguide1	δ/appr. error	–	0.0128	0.0032	0.12	0.22
	Test (%)	**97.15**	**97.125**	**97.15**	**97.1**	**97.15**
	ts. time	30.87	19.08	26.16	0.9833	2.56
	# ts. SVM	4000	4000	4000	126	340
	# SV/#FV	335	211	289	335	335
Svmguide3	δ/appr. error	–	0.0008	1e-6	0.3	0.34
	Test (%)	85.69	84.88	85.64	85.69	86.31
	ts. time	1.26	0.67	0.88	0.54	0.78
	# ts. SVM	128.4	128.4	128.4	65.1	81.3
	# SV/#FV	428.1	224.0	276.1	428.1	428.1

testing phase concentrate on reducing the number of support vectors, we have focused on reducing the number of test datapoints that need SVM's decision in getting classified. SVMDT is a hybrid tree which generally has both univariate and multivariate (SVM) nodes. Less crucial datapoints get classified with fast univariate decision nodes and crucial datapoints get classified with SVM nodes. The accuracy of SVMDT is guaranteed by tuning the threshold δ. In its best case, SVMDT ($\delta=0$) may boil down to a fast conventional DT; in its worst case SVMDT ($\delta=0.5$) just becomes SVM itself. Thus the threshold δ acts as a trade-off between SVM and DT classification. We have conducted extensive experiments on 27 publicly available datasets to demonstrate the effectiveness of SVMDT. On all the datasets, SVMDT has shown impressive results with significant speedup when compared to SVM, without any compromise in classification accuracy. Further SVMDT is a novel way of decreasing testing time of SVMs and it does not contradict with existing approaches. Our subject for future work is to realize the potential of SVMDT in multiclass classification.

Appendix A

SVMDT results on adult datasets are given in Table A1.

Appendix B

SVMDT results on other binary datasets used are given in Table B1.

Appendix C

SVMDT comparison with FVS is given in Table C1.

References

[1] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (1998) 1–43.
[2] C.J. Burges, Simplified support vector decision rules, in: Proceedings of the 13th International Conference on Machine Learning, Italy, 1996, pp. 71–74.
[3] T. Downs, K. Gates, A. Masters, Exact simplification of support vector solutions, Journal of Machine Learning Research 2 (2001) 293–297.
[4] C.J. Burges, B. Scholelkopf, Improving the accuracy and speed of support vector machines, Advances in Neural Information Processing Systems 9 (1997) 375–381.
[5] E. Osuna, F. Girosi, Reducing the run-time complexity of support vector machines. Advances in Kernel Methods—Support Vector Learning, MIT Press, 1999, pp. 271–283.
[6] Y. Li, W. Zhang, Simplify support vector machines by iterative learning, Neural Information Processing—Letters and Reviews 10 (1) (2006) 11–17.
[7] Q. Li, L. Jiao, Y. Hao, Adaptive simplification of solution for support vector machines, Pattern Recognition 40 (2007) 972–980.
[8] C.L. Blake, C.J. Merz, in: UCI repository for machine learning databases. Department of Information and Computer Sciences, University of California, Irvine, 1998 ⟨http://www.ics.uci.edu/m/earn/MLRepository.html⟩.
[9] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, Advances in Kernel Methods—Support Vector Learning, MIT Press, 1999.
[10] T. Joachims, Making large-scale SVM learning practical, in: B. Scholkopf, C.J. Burges, A.J. Smola (Eds.), Advances in Kernel Methods—Support vector Learning, MIT Press, 1999.
[11] I.H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
[12] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd Edition, Wiley-Interscience, 2000.
[13] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: Proceedings of ECML-98, 10th European Conference on Machine Learning, 1998, pp. 137–142.
[14] C. Cortes, V. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.
[15] K.P. Bennett, J.A. Blue, A support vector machine approach to decision trees, Department of Mathematical Sciences Math Report No. 97-100, Rensselaer Polytechnic Institute, 1997, pp. 2396–2401.
[16] B. Fei, J. Liu, Binary tree of SVM: a new fast multiclass training and classification algorithm, IEEE Transactions on Neural Networks 17 (2006) 696–704.
[17] MATLAB Software: ⟨http://www.mathworks.com⟩, 2009.
[18] L. Kaufman, Solving the quadratic programming problem arising in support vector classification, Advances in Kernel Methods—Support Vector Learning, MIT Press, 1999, pp. 147–168.
[19] C.C. Chang, C.J. Lin, LibSVM: A Library for Support Vector Machine, 2001, software available at: ⟨http://www.csie.ntu.edu.tw/~cjlin/libsvm/⟩.
[20] Ripley's Dataset, available at: ⟨http://www.stats.ox.ac.uk/pub/PRNN/⟩.
[21] Promise Dataset Repository, ⟨http://promise.site.uottawa.ca/SERepository/⟩.
[22] Gunnar's Benchmark Repository, ⟨http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm⟩.
[23] Mosek optimization toolbox for MATLAB, 2009 ⟨http://www.mosek.com⟩.

M. Arun Kumar did his Bachelors/Masters in Electronics/Controls/Instrumentation from Annamalai University, Tamil Nadu, India, in 2001 and 2004, respectively. He is currently working towards his Ph.D. with the Department of Electrical Engineering, Indian Institute of Technology, Delhi. His current research interests are in the areas of machine learning, data mining, optimization and information retrieval.

M. Gopal is currently Professor with the Department of Electrical Engineering, Indian Institute of Technology, Delhi. His teaching and research stints span over three decades at prestigious institutes. He is the author/co-author of six books. His video course on control engineering is being transmitted periodically through EKLAVYA Technology channel of IIT. He is the author of an interactive web compatible multimedia course on control engineering (available at http://nptel@iitm.ac.in). A large number of research publications are to his credit. His current research interests are in the areas of machine learning, soft computing, and intelligent control.