Applied Soft Computing 64 (2018) 356–365

Contents lists available at ScienceDirect

Applied Soft Computing

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

MS-SVM: Minimally Spanned Support Vector Machine

Rupan Panja[a], Nikhil R. Pal[b,*]

[a] Computer Science Department, Chandernagore College, Strand Road, Chandernagore, West Bengal 712136, India
[b] Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700108, India

A R T I C L E I N F O

Article history:
Received 10 May 2017
Received in revised form 25 October 2017
Accepted 9 December 2017
Available online 15 December 2017

Keywords:
Support vector machine
Minimum spanning tree
Prim's algorithm
Kernel method

成比例

分离超平面

生成树

先用最小生成缩减训练集的数量，
之后再使用SVM建立模型

A B S T R A C T

For a Support Vector Machine (SVM) algorithm, the time required for classifying an unknown data point is proportional to the number of support vectors. For some real time applications, use of SVM could be a problem if the number of support vectors is high. Depending on the complexity of the class structure, sometimes the number of support vectors of a SVM model increases with the number of training data points. Here our objective is to reduce the number of support vectors, yet, maintaining more of less the same level of accuracy as that of a normal SVM that does not use any reduction of support vectors. An SVM finds a separating hyperplane maximizing the margin of separation and hence, the location of the hyperplane is primarily dependent on a set of "boundary points". Here, we first identify some boundary points using a minimum spanning tree on the training data to obtain a reduced training set. The SVM algorithm is then applied on the reduced training data to generate the classification model. We call this algorithm, Minimally Spanned Support Vector Machine (MS-SVM). We also assess the performance by relaxing the definition of boundary points. Moreover we extend the algorithm to a feature space using a kernel transformation. In this case, an MST is generated in the feature space using the associated kernel matrix. Our experimental results demonstrate that the proposed algorithm can considerably reduce the number of support vectors without affecting the overall classification accuracy. This is true irrespective of whether the MST is generated in the input space or in the feature space. Thus the MS-SVM algorithm can be used instead of SVM for efficient classification.

1. Introduction

The Support Vector Machine (SVM) algorithm proposed by Vapnik [1] has been successfully applied in many applications over the past decades. The good generalization capability of SVM to separate two classes has made it widely acceptable. However, the run time of an SVM model to classify a data point is directly related to the number of support vectors and this could be problematic for some real time applications like BCI applications where an instantaneous decision has to be made based on continuous signals. To classify a data point, an SVM computes the dot product of the given test point with every support vector either in the input space or in the feature space after transformation via a kernel function. Thus, the execution time increases with an increase in the number of support vectors.

问题

In the past, researchers have proposed various methods to reduce the number of support vectors [2–10,13,14]. In [2] authors proposed the ν-SVM algorithm, where the parameter ν is used to control the number of support vectors. However, it may not always give a small set of support vectors [3]. Li et al. [3] presented an algorithm where, the training data points of each class are clustered separately using the k-means algorithm and only the cluster centers are used to train an SVM model. The number of support vectors is controlled by the choice of k for each class; when k is set to the number of data points in each class, it reduces to the normal SVM. In this algorithm, determining the desired value of k for each class requires further investigation. It is worth noting here that, if only the cluster centers are used for training, the SVM may not be able to capture the boundary of classes properly – a cluster center represents the core of a cluster but not the boundary, while boundary points are more important for defining the classifier by an SVM. In [4] Downs et al. proposed an algorithm to remove linearly redundant support vectors. The authors of [6] heuristically divided the training data into smaller chunks such that every training data points is just in one chunk. Each of these chunks is then used to find support vectors, and only these support vectors are kept for further training [8]. Though, the time required to train an SVM model using this algorithm will be considerably high, it has an added advantage of tackling the memory constraint. Dagl et al. [7] introduced the concept of fuzziness for identifying irrelevant

启发式
较小的块

* Corresponding author.
E-mail address: nikhil@isical.ac.in (N.R. Pal).

https://doi.org/10.1016/j.asoc.2017.12.017
1568-4946/© 2017 Elsevier B.V. All rights reserved.

training data points. The class membership of each training data point is calculated from its k-nearest neighbors. The training data 清晰 points with crisp class membership are discarded as irrelevant and the SVM model is trained with the remaining data points. The k-means algorithm has also been used in [8,9] to remove irrelevant data points. In [8] the radii of the clusters obtained using k-means are increased/decreased to make them largest possible crisp clusters with at least a minimum number of data points. The training data points from the core of these crisp clusters are removed as irrelevant. In [10] Kumar et al. also clustered each class separately using different clustering algorithms and have then used the cluster centers to train SVMs, thereby reducing the number of support vectors as in [3]. Note that, the philosophy of [8] is to discard the core of each cluster, while that of [3,10] is to use the core of a cluster as represented by the center of a cluster. In [13], the authors have proposed an interesting method to reduce the number of support vectors of a fuzzified version of SVM. They have used an l_0-norm of the vector of Lagrangian multipliers as a regularizer to the dual problem of the fuzzy SVM. Although the use of l_0-norm is a natural choice for reducing the number of support vectors, it makes the optimization problem non-convex, which is then solved using a continuous approximation function. In [14] authors use the generalized support vector machine along with a rectangular kernel. Here while finding the nonlinear separating surface, authors solves an optimization problem with a few variables (corresponding to about 10% of the randomly selected data points) but use the entire dataset as constraints such that once the decision surface is found, almost 90% of the data can be discarded.

In this paper we propose algorithms to reduce the number of support vectors for SVMs. We call this method Minimally-Spanned Support Vector Machine (MS-SVM). A minimum spanning tree algorithm is first applied on the training dataset to remove data points which are far away from the expected decision boundary. The SVM model is then built on the remaining data points. The effectiveness of the proposed algorithm has been demonstrated on a set of benchmark datasets. We have considered both linear and non-linear SVMs. The MST can be obtained either in the input space 增强版本 or in the feature space. An augmented version of the algorithm is also proposed for data sets with very well separated classes. The rest of the paper is organized as follows: Section 2 gives a brief introduction to minimum spanning tree and SVM algorithm, then the minimally-spanned support vector machine algorithm is proposed. In Section 3, experimental results on a set of benchmark datasets are discussed, and finally in Section 4 the paper is concluded.

2. Methods

2.1. Minimum spanning tree

In a connected undirected weighted graph $G(V, E)$ a minimum spanning tree (MST) is a sub-graph $G'(V, E')$ such that, the number 顶点 of edges of G' is equal to one less than the number of vertices (i.e. $|E'| = |V| - 1$) connecting all the vertices and the sum of weight of edges in E' being minimal [11]. Note that, a graph may have multiple MSTs. There are several algorithms for finding a minimum spanning tree of a graph. In this work we have used Prim's algorithm [12]. Since an MST is a minimum-weight spanning tree, it has many applications in diverse areas like telecommunication and pattern recognition.

2.2. Support vector machine

The objective of a support vector machine training algorithm for a two-class problem is to find the optimal separating hyperplane, that maximises the margin of separation between the two classes. However, if data points from the two classes are not linearly separable in the input space, then they are transformed into a higher dimensional space using a nonlinear mapping $\phi(x)$ with a hope that the given set of points would be separable in the high dimensional space. Thus, for a dataset having N points $\{\mathrm{x_i} \epsilon \mathbb{R}^p\}_{i=1}^N$ with labels $y_i \epsilon \{-1, 1\}_{i=1}^N$ an SVM algorithm solves the following problem [5]

$$\min_{w,b,\xi_i} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \tag{1}$$

such that

$$y_i(w^T\phi(\mathbf{x}) + b) \geq 1 - \xi_i; \quad i = 1, 2, \ldots, N. \tag{2}$$

and

$$\xi_i \geq 0; \quad i = 1, 2, \ldots, N. \tag{3}$$

Here $C \geq 0$ is a tradeoff between complexity and training error [5] and $\xi_i; i = 1, 2, \ldots, N$ are positive slack variables. The dual Lagrangian 拉格朗日 of the above SVM optimization problem can be written as follows:

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(\mathrm{x_i}, \mathrm{x_j})\alpha_i\alpha_j - \sum_{i=1}^{N}\alpha_i \tag{4}$$

such that

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{5}$$

and

$$0 \leq \alpha_i \leq C; \quad i = 1, 2, \ldots, N. \tag{6}$$

Here K is a kernel function of the form $K(\mathbf{x}_i, \mathbf{x}_j) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle$, where $\langle\mathbf{x}, \mathbf{y}\rangle$ denotes the dot product of $\mathbf{x}$ and $\mathbf{y}$.

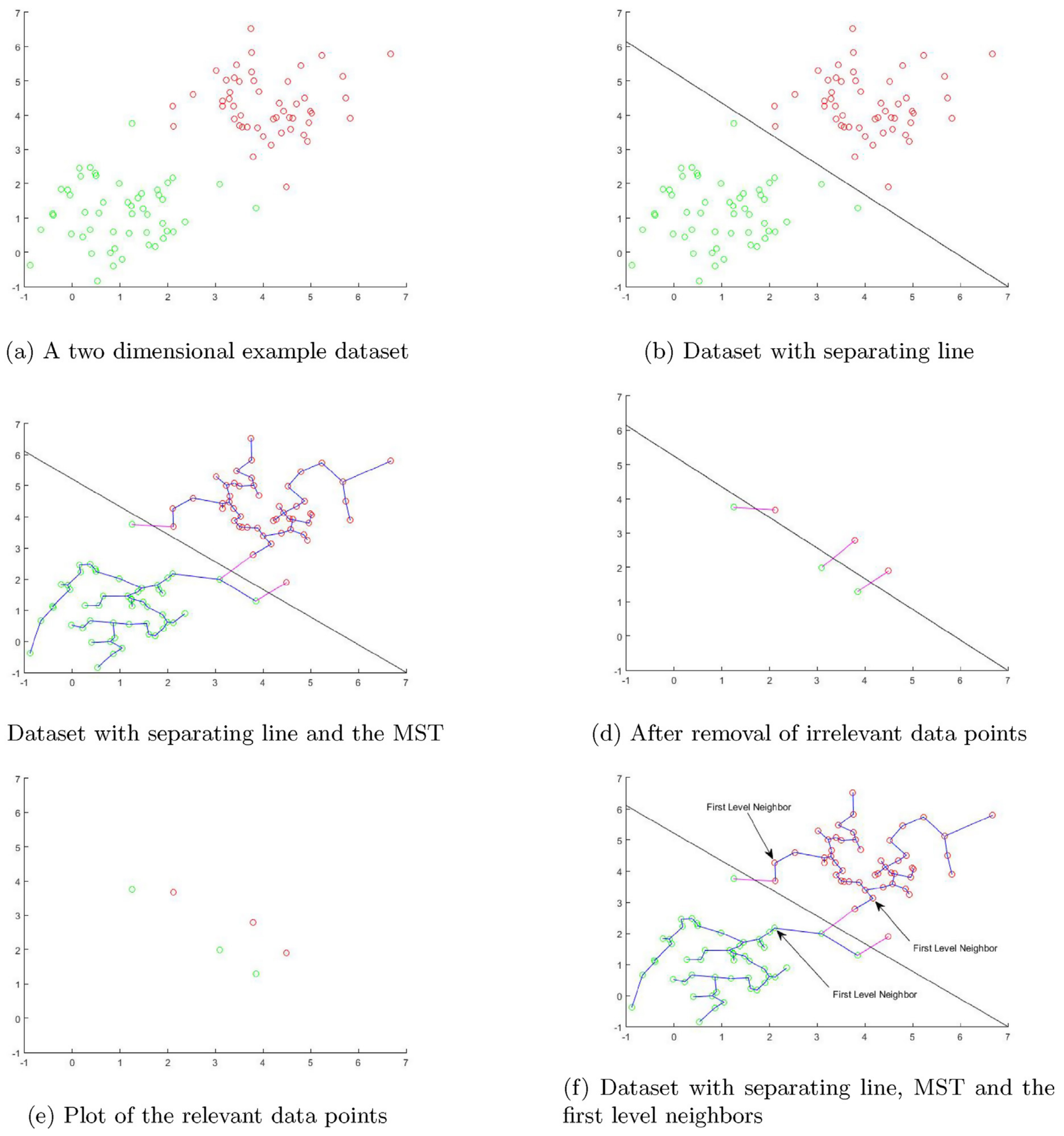
The predicted class of an unknown data point $\mathbf{x}$ is obtained by the decision function:

$$sign\left(\sum_{i=1}^{N}\alpha_i y_i K(\mathbf{x}, \mathrm{x_i}) + b\right). \tag{7}$$

Note here that if $\alpha_i = 0$, then $\mathbf{x}_i$ does not contribute in decision making. The set of $\mathbf{x}_i$ for which $\alpha_i > 0$ is called the support vectors.

2.3. Minimally-spanned support vector machine

In the MS-SVM algorithm the objective is to reduce the number of support vectors by initially removing "irrelevant" training data points. A natural question comes, how to define irrelevant points. An SVM finds a separating hyperplane maximizing the margin of separation. Thus, primarily, the separating hyperplane is defined by "boundary" points between classes. Although we do not have a precise definition of boundary points, we can certainly say that points which are deep inside a class, say points with all k-nearest neighbors from the same class, are not expected to influence the SVM hyperplane. By "irrelevant" points we mean such points which are not expected to influence the boundary between classes as defined by a SVM separating hyper-plane. Ideally when the data points of the two classes are linearly separable, the separating hyperplane of the SVM passes through the data in such a manner that the data points of the two classes lie on its opposite sides. So, the data points of each class which are towards the boundary or near to the separating hyperplane mostly determine the separating hyperplane. And, the data points of each class which are away from the separating hyperplane or deep inside the cluster of a class are irrelevant. Now, assuming that the data points of the two classes are linearly separable, at least approximately, if we build a minimum spanning tree on the training dataset the separating hyperplane is likely



(a) A two dimensional example dataset
(b) Dataset with separating line
(c) Dataset with separating line and the MST
(d) After removal of irrelevant data points
(e) Plot of the relevant data points
(f) Dataset with separating line, MST and the first level neighbors

Fig. 1. Diagrammatic representation of the concept of minimally spanned support vector machine.

to pass mostly through those edges having data points from two different classes on its either side. Thus, the data points from different classes which are connected by a single edge of the MST are important for finding the separating hyperplane, at least we can say that they are important for defining the class boundary. Thus these points are very relevant to define the class boundary. In our MS-SVM algorithm we build an MST on the training dataset to remove the irrelevant points. The SVM classifier is then learned using the remaining data points.

形象的

In Fig. 1 we try to motivate our algorithm pictorially. Fig. 1a shows a synthetically generated two dimensional dataset. The data points of the two classes are shown in two different colors, red and green. A line separating the classes is added in Fig. 1b. In Fig. 1c the edges of a minimum spanning tree built on this dataset are added, the edges with data points from the same class on its either side are in blue while edges with data points from different classes are in magenta. As previously explained, we can see that a separating line passes through the edges which are magenta in color, i.e. edges with data points from different classes. We therefore mark the data points of the edges in magenta as relevant and remove the remaining irrelevant data points as shown in Fig. 1d. The SVM model is then build on the relevant data points, as shown in Fig. 1e, to get the separating line. Fig. 1a shows all data points in the dataset and Fig. 1e shows only the relevant data points as found by the MS-SVM algorithm, and thereafter used for learning the model.

合成的

洋红色

In the minimally spanned support vector machine algorithm we can increase the relevant set by including data points on the neighboring edges of the already included edges. The reason behind this is that in real life data, the classes may not be linearly separable or the class boundary could be more complex and hence, it may require more data points to get a better model of the class boundary.

As the first level neighbor, we include every data point which is adjacent to an already selected data point of the same class in the MST. So, as per Fig. 1, the new edges are blue in color, as the data

邻近的

这里是说也要加上相关数据点的邻近点们

points on either side of the newly added edges belong to the *same* class. In Fig. 1c three points (two from the red class and one from the green class) will be included as the first level neighbor. These points are marked in Fig. 1f. If we continue adding the neighboring edges iteratively, after some iterations the whole dataset will be included and MS-SVM will be the same as the normal SVM. The number of support vectors may increase on including more and more neighbors, which may also increase the overall classification accuracy. Algorithm 1 describes the proposed MS-SVM method.

Algorithm 1. Algorithm MS-SVM.

```
Input: The dataset.
Output: The classification accuracy.
Create first level folds $train_i$-$test_i$, $i \in (1,...,10)$
For each $train_i$ create the second level folds $train_{ij}$-$test_{ij}$, $j \in (1,...,10)$
$SVMAccuracy = 0$, $MSSVMAccuracy = 0$
for $i = 1$ to 10 do
  for $j = 1$ to 10 do
    $MST_{ij}$ = Build a MST on $train_{ij}$
    $RelevantPoints_{ij}$ = Adjacent datapoints in $MST_{ij}$ belonging to different classes
    for $p = 1$ to No. of level of Neighbors do
      $AdjacentPoints$ = Datapoints adjacent to $RelevantPoints_{ij}$
      $RelevantPoints_{ij} = RelevantPoints_{ij} \bigcup AdjacentPoints$
    end for
    for all SVM-Parameter-Combinations do
      $SVM_{ij}$ = Train a SVM on $train_{ij}$
      $SVMAccuracy_{ij}$ = Testing $test_{ij}$ using $SVM_{ij}$
      $MSSVM_{ij}$ = Train a SVM on $RelevantPoints_{ij}$
      $MSSVMAccuracy_{ij}$ = Testing $test_{ij}$ using $MSSVM_{ij}$
    end for
  end for
  $SVMParameter_i = argmax(SVMAccuracy_{ij})$
  $MSSVMParameter_i = argmax(MSSVMAccuracy_{ij})$
  $MST_i$ = Build a MST on $train_i$
  $RelevantPoints_i$ = Adjacent datapoints in $MST_i$ belonging to different classes
  for $p = 1$ to No. of level of Neighbors do
      $AdjacentPoints$ = Datapoints adjacent to $RelevantPoints_i$
      $RelevantPoints_i = RelevantPoints_i \bigcup AdjacentPoints$
  end for
  $SVM_i$ = Train a SVM on $train_i$ using $SVMParameter_i$
  $SVMAccuracy_i$ = Testing $test_i$ using $SVM_i$
  $SVMAccuracy = SVMAccuracy + SVMAccuracy_i$
  $MSSVM_i$ = Train a SVM on $RelevantPoints_i$ using $MSSVMParameter_i$
  $MSSVMAccuracy_i$ = Testing $test_i$ using $MSSVM_i$
  $MSSVMAccuracy = MSSVMAccuracy + MSSVMAccuracy_i$
end for
$SVMAccuracy = SVMAccuracy/10$
$MSSVMAccuracy = MSSVMAccuracy/10$
```

3. Results

The effectiveness of the minimally-spanned support vector machine algorithm is demonstrated on 12 benchmark datasets (including three high dimensional ones) and one synthetically generated two-class dataset. The synthetic dataset, named, *Synthetic*, consists of 200 points; 100 2-dimensional points in each of two classes. Data points in each class are randomly generated from a Gaussian distribution. The results depicted in subsequent tables are generated using a two-level 10-fold cross validation averaged over ten runs. There are three sets of experiments. The first set uses linear SVM (Tables 1 and 2). The parameter, C, of the linear SVM is selected using a two-level cross validation technique from the set $\{0.1, 1, 5, 10, 50, 100, 500\}$. Table 1 summarizes the results of MS-SVM using only the boundary points; while Table 2 depicts the results when the first level neighbors are also used to obtain the linear SVM. In Table 1 we find that in five of the ten cases the performance of MS-SVM is either very comparable or marginally better than SVM; while Table 2 reveals that for all the datasets but Satellite Image, the performance of MS-SVM is comparable or marginally better. We note that the accuracy obtained for the Satellite Image dataset using MS-SVM, with only the boundary points, is poor. This is possibly because of elimination of some relevant points. If that is so, the performance should improve, when we use higher level neighbors. The performance of MS-SVM, as expected, is improved much when we consider the third level neighbors. The classification accuracy for Satellite Image dataset obtained by MS-SVM, considering the third level neighbors, is 87.66% with a standard deviation of 1.66. On an average the number of support vectors used is 1033.47 reducing 20.58% of support vectors. The performance of MS-SVM is now comparable with that of SVM, although the reduction in the number of support vectors is still more than 20%. From Table 2, we find that the reduction in number of support vectors varies between 4.94% and 73.39% (for the banana dataset). It is worth noting that even with a reduction of 73.39% of support vectors, MS-SVM marginally outperforms SVM. It is also to be remembered that, the datasets whose MS-SVM classification accuracy is comparable or better than that of SVM without considering the first level neighbors (as shown in Table 1), need not include the first level neighbors as it would be redundant and the gain in classification accuracy is negligible compared to the increase in support vectors.

The second set of experiments involve Kernel SVM. Here also we use an MST computed on the same graph as in our previous experiment. For the two-level cross validation, when using the RBF kernel, the inner level selects the optimal C and γ for the outer level using: $C \in \{0.1, 1, 5, 10, 50, 100, 500\}$ and $\gamma \in \{0.001, 0.01, 0.1, 1, 5, 10, 50\}$. Tables 3 and 4 compare the performance of MS-SVM with that of SVM under two scenarios: using only the boundary points and boundary points along with the first level neighbors, respectively. Table 3 reveals that for all but Ionos and Banana datasets the classification accuracy obtained by MS-SVM is almost the same or even in some cases marginally better than that of SVM with all data points. And, for each of these datasets the number of support vectors is reduced significantly, specially for Synthetic, Satellite Image, Scene, and Yeast ML8 datasets the reductions are huge (more than 67%). In this case, for eight of the datasets the performance of MS-SVM is comparable or even marginally better than that of SVM. And, on considering the first level neighbors, as shown in Table 4, the classification accuracy for all the datasets is either comparable or marginally better than that of SVM. We further note that, even after considering the first level neighbors the reduction in the number of support vectors is significantly large for most of the datasets. Moreover, based on the results of our limited experiments shown in Tables 3 and 4, we find that usually the reduction in the number of support vectors is more when the number of data points and the dimensionality are high. The low standard deviation of the classification accuracy achieved by MS-SVM also emphasizes the consistency of the results and hence of the algorithm.

In the third set of experiments, we take a different approach and instead of finding the MST in the input space we do it in the feature space. Here, we consider the RBF Kernel, where the kernel function K is defined as:

$$K(\mathbf{x}, \mathbf{y}) = \exp^{(-\gamma\|\mathbf{x}-\mathbf{y}\|^2)} \tag{8}$$

Now, since $K(\mathbf{x}, \mathbf{y})$ gives the similarity between the datapoints $\mathbf{x}$ and $\mathbf{y}$, the MST is build on the dissimilarity measure $1 - K(\mathbf{x}, \mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$. The RBF kernel parameters C and γ are chosen by two-level 10 fold cross validation, from $C \in \{0.1, 1, 5, 10, 50, 100, 500\}$ and $\gamma \in \{0.001, 0.01, 0.1, 1, 5, 10, 50\}$, as in the previous set of experiments. In Table 5 we compare the performance of SVM with that of MS-SVM considering only the boundary points and in Table 6 the performance of SVM is compared with that of MS-SVM considering the boundary points along with the first level neighbours, where in both cases the MST is built in the feature space. It is evident from Table 5 that, the classification accuracy obtained by MS-SVM for all but Ionos, Liver and Banana datasets are either comparable or marginally better than that of SVM and the classification accuracy for the datasets Ionos, Liver and Banana are expected to improve if the first level neighbors are also con-

Table 1
Performance comparison of linear SVM and linear MS-SVM.

Datasets (no. of points, no. of features)	SVM		MS-SVM		% Reduction of # of SVs
	Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Synthetic (200,2)	91.65 (0.41)	39.75	91.65 (0.47)	26.67	32.91
Haberman (306,3)	72.42 (0.44)	157.09	73.10 (0.41)	126.26	19.41
Ionos (351,34)	86.30 (1.51)	79.5	82.76 (1.06)	55.76	29.86
Liver (345,6)	68.17 (0.65)	225.21	64.35 (1.39)	146.96	34.75
Satellite Image (6435,36)	89.73 (1.15)	1302.18	52.04 (1.67)	361.02	72.26
Sonar (208,60)	74.57 (1.23)	99.78	71.06 (2.96)	54.93	44.95
Scene (2407,294)	92.64 (0.04)	447.44	85.45 (0.42)	284.67	36.38
Yeast ML8 (2417,103)	92.64 (0)	618.82	92.64 (0.0)	328.40	46.93
Banana (5300,2)	54.42 (1.20)	4110.20	55.19 (0.14)	868.90	78.86
US Crime (1994,100)	94.08 (0.21)	258.71	93.36 (0.19)	208.97	19.23

Table 2
Performance comparison of linear SVM and linear MS-SVM with first level neighbors.

Datasets	SVM		MS-SVM		% Reduction of # of SVs
	Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Synthetic	91.65 (0.41)	39.19	92.25 (0.35)	30.98	22.38
Haberman	72.48 (0.46)	157	72.84 (0.45)	147.34	6.15
Ionos	86.44 (1.40)	79.80	83.93 (1.65)	58.45	26.75
Liver	68.14 (0.59)	225.19	66.52 (1.02)	189.83	15.70
Satellite Image	89.79 (1.02)	1301.22	60.96 (0.54)	744.66	42.77
Sonar	74.90 (1.72)	99.14	75.24 (2.91)	73.69	25.67
Scene	92.64 (0.05)	447.26	92.41 (0.18)	358.79	19.78
Yeast ML8	92.64 (0)	623.35	92.64 (0)	425.66	31.71
Banana	54.41 (1.22)	4124.55	55.01 (0.43)	1097.34	73.39
US Crime	94.07 (0.18)	258.67	93.73 (0.20)	245.89	4.94

Table 3
Performance comparison of non-linear SVM and non-linear MS-SVM.

Datasets	SVM		MS-SVM		% Reduction of # of SVs
	Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Synthetic	91.80 (0.71)	95.38	91.65 (0.58)	27.57	71.10
Haberman	72.68 (1.02)	164.22	72.58 (1.32)	122.78	25.23
Ionos	94.62 (0.37)	82.79	89.97 (1.25)	52.37	36.74
Liver	68.49 (0.82)	227.46	67.19 (1.71)	163.21	28.25
Satellite Image	95.12 (0.17)	1617.81	94 (0.09)	564.92	68.42
Sonar	85.29 (1.89)	114.96	83.85 (1.40)	67.95	41.23
Scene	92.72 (0.13)	1994.18	92.64 (0.05)	328.39	83.53
Yeast ML8	92.59 (0.05)	1163.56	92.64 (0)	381.76	67.19
Banana	90.47 (0.09)	1192.50	87.47 (0.51)	837.26	29.79
US Crime	94.02 (0.12)	285.66	93.41 (0.24)	211.98	25.79

Table 4
Performance comparison of non-linear SVM and non-linear MS-SVM with first level neighbors.

Datasets	SVM		MS-SVM		% Reduction of # of SVs
	Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Synthetic	91.85 (0.63)	95.02	91.25 (1.27)	32.45	65.85
Haberman	72.88 (0.91)	166.10	72.88 (1.08)	144.30	13.12
Ionos	94.62 (0.43)	83.34	92.59 (0.87)	59.29	59.29
Liver	68.46 (0.77)	227.28	68.58 (1.01)	204.60	9.98
Satellite Image	95.12 (0.17)	1619.61	94.12 (0.09)	805.30	50.28
Sonar	85.82 (2.16)	116.55	85.19 (2.30)	87.48	24.94
Scene	92.71 (0.13)	1994.05	92.74 (0.07)	578.99	70.96
Yeast ML8	92.59 (0.06)	1094.96	92.64 (0)	595.53	45.61
Banana	90.44 (0.09)	1195.86	89.96 (0.11)	986.45	17.51
US Crime	94 (0.12)	286.80	93.76 (0.14)	247.12	13.84

sidered. The experimental setup for Tables 3 and 5 are the same except in Table 3 the MST is built in the input space and in Table 5 the MST is built in the feature space. The similarity between the results obtained in Tables 3 and 5 leads to similar observations for Table 5 as we made for Table 3. The consideration of the first level neighbors, as expected, improves the classification accuracies for the datasets Ionos, Liver and Banana as shown in Table 6, and, thus, for all datasets the classification accuracies for MS-SVM

Table 5
Performance comparison of non-linear SVM and non-linear MS-SVM where the MST is build in the kernel space.

Datasets	SVM		MS-SVM		% Reduction of # of SVs
	Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Synthetic	91.55 (0.60)	98.61	91.80 (0.75)	27.52	72.09
Haberman	72.55 (1.18)	159.53	72.55 (0.87)	122.35	23.31
Ionos	94.44 (0.41)	82.18	90.22 (1.40)	51.36	37.50
Liver	68.93 (0.89)	227.23	66.67 (1.69)	164.11	27.78
Satellite Image	95.12 (0.18)	1630.68	94.00 (0.09)	564.92	65.36
Sonar	85.29 (1.27)	113.12	84.38 (1.18)	67.85	40.02
Scene	92.78 (0.10)	2030.60	92.65 (0.05)	326.49	83.92
Yeast ML8	92.62 (0.05)	1179.22	92.64 (0.02)	375.39	68.17
Banana	90.47 (0.12)	1218.82	86.82 (0.53)	837.22	31.31
US Crime	94.02 (0.16)	287.33	93.35 (0.29)	211.88	26.26

Table 6
Performance comparison of non-linear SVM and non-linear MS-SVM with first level neighbors where the MST is build in the kernel space.

Datasets	SVM		MS-SVM		% Reduction of # of SVs
	Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Synthetic	91.55 (0.76)	96.55	91.30 (1.32)	32.81	66.02
Haberman	72.55 (1.12)	160.42	72.32 (0.93)	152.52	4.92
Ionos	94.33 (0.43)	82	92.68 (0.96)	60.32	26.44
Liver	68.87 (0.88)	227.46	68.26 (1.18)	203.20	10.67
Satellite Image	95.12 (0.18)	1630.68	94.12 (0.09)	805.3	50.62
Sonar	85.34 (1.29)	114.36	84.66 (1.33)	85.60	25.15
Scene	92.78 (0.11)	2026.75	92.78 (0.03)	1702.56	16
Yeast ML8	92.62 (0.05)	1203.03	92.64 (0)	1087.91	9.57
Banana	90.47 (0.12)	1217.25	89.95 (0.11)	988.48	18.79
US Crime	93.99 (0.15)	286.31	93.84 (0.18)	246.03	14.07

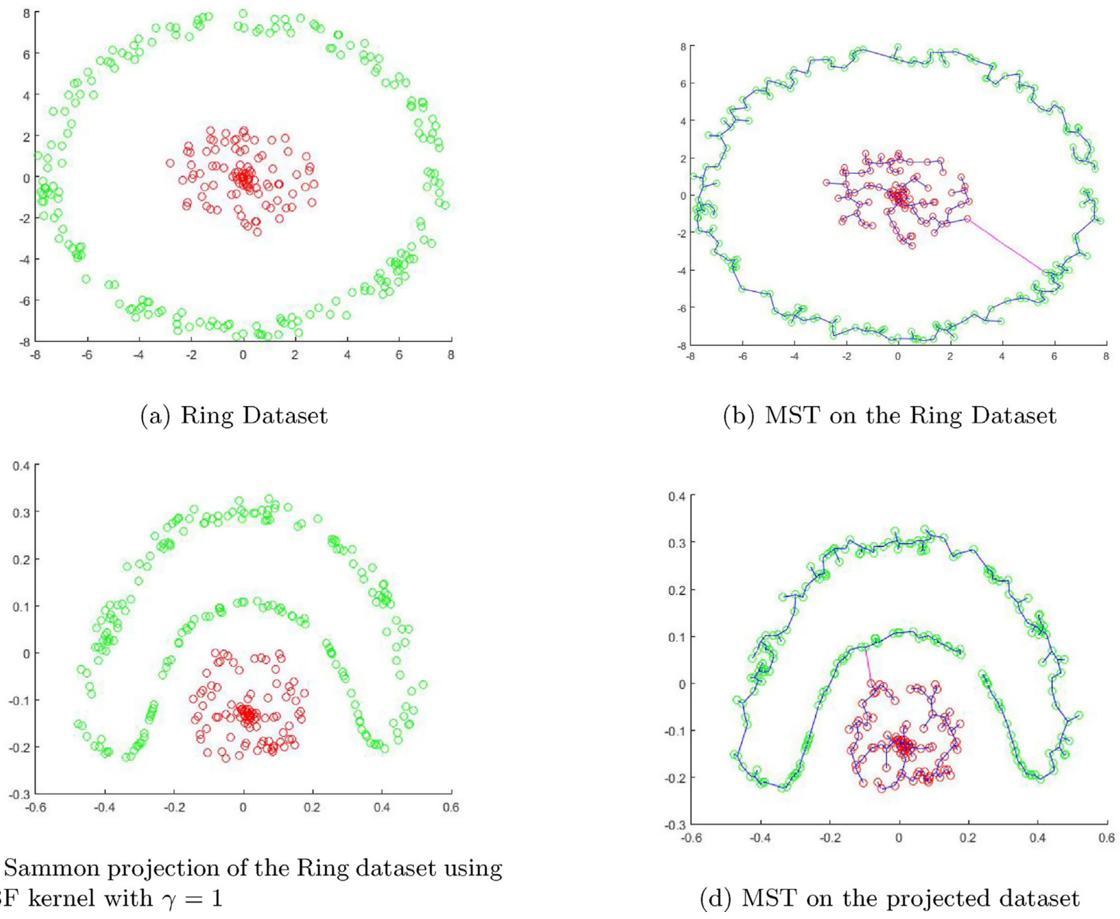


Fig. 2. Diagrammatic representation of our study on the Ring dataset.

Table 7
Performance comparison of SVM and MS-SVM for Ring dataset.

Datasets (no. of points, no. of features)	SVM		MS-SVM		% Reduction of # of SVs
	Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Ring (300,2)	100 (0)	51.75	99.67 (0.42)	9.25	82.13

algorithm are either comparable or marginally better than that of SVM. The huge reduction in the number of support vectors by the MS-SVM algorithm compared to SVM, again proves the effectiveness of the MS-SVM algorithm. Again, the experimental setup for Tables 4 and 6 are the same except in Table 4 the MST is built in the input space and in Table 6 it is built in the feature space. The results obtained in Tables 4 and 6 are also very similar except for the number of support vectors in case of Scene and YeastML8 datasets. However, we observe that for Scene and YeastML8 datasets the results are good even without considering the first level neighbors. The similarity in the results between Tables 3 and 5 and that between Tables 4 and 6 reveals that the MST can be build either in the input space or in the feature space without affecting the overall classification accuracy. We note that, for nonlinear SVM, irrespective of whether MST is found in the input space or in the feature space, inclusion of the first level neighbors does not improve the performance much in most cases. Thus, as emphasized earlier, if the performance of MS-SVM (without first level neighbors) is comparable with that of SVM, then we need not consider the first level neighbors.

After successful implementation of the MS-SVM algorithm on nine benchmark datasets and one synthetically generated dataset, we try to study the behavior of the proposed MS-SVM algorithm on a different class of data where classes are well separated but a linear hyperplane cannot classify the data. As an example consider a 2D data, called, Ring as shown in Fig. 2a. The Ring dataset consists of 300 points belonging to two classes, 100 points in the inner class and 200 points in the outer class. It is evident from Fig. 2a, that using a linear SVM we cannot discriminate between the two classes. Now, using RBF kernels if we apply the MS-SVM algorithm on the Ring dataset, in the first step we get the minimum spanning tree as shown in Fig. 2b. In Fig. 2b, we can see that, there is only one edge joining data points of the two classes, and thus only two relevant data points(the data points joined by the edge). Now, there can be many lines or curves separating the two relevant points but they will fail to discriminate the two classes, as the selected two data points do not portray the dataset. Now, instead of building the MST in the input space we build it in the feature space. The Sammon's projection [15] of the Ring dataset using RBF kernel with γ equal to 1 is shown in Fig. 2c, where the output dimensionality is taken as two. The MST on the projected dataset, as shown in Fig. 2d, poses the same problem of selecting only two relevant data points by the MS-SVM algorithm. This is a problem.

描绘
数据集

Now, to solve the problem, similar to the one with the Ring dataset, we take a different plausible approach. It is evident from Fig. 2d that, instead of one edge connecting the two classes if multiple such edges are found, the separating curve can be learned easily. To achieve this, we remove the selected relevant points and rebuild the MST on the remaining data points. This gives another set of relevant points which along with the previously selected relevant points gives a better understanding of the separating curve. The process of removing selected relevant points and rebuilding the MST to get another set of relevant points is continued till the difference in the training accuracy with normal SVM reaches an acceptable mark. The experimental results for the Ring dataset in Table 7 are also generated by two-level 10-fold cross validation averaged over ten runs, with $C \in \{0.1, 1, 5, 10, 50, 100, 500\}$ and $\gamma \in \{0.001, 0.01, 0.1, 1, 5, 10, 50\}$. The process of building MSTs to get additional boundary points is continued till the difference in training accuracies between the normal SVM and MS-SVM becomes less than 5%. The result obtained by the modified MS-SVM algorithm is comparable with that of the SVM and the number of support vectors obtained by our algorithm is almost negligible 微不足道 compared to that obtained by normal SVM. This modified MS-SVM algorithm can be applied to datasets where interclass separability causes a single edge or a few edges with end data points belonging to different classes in the MST resulting into low classification accuracy. In Algorithm 2 we provide this slightly modified MS-SVM algorithm used to address the stated problem.

似是而非

有的数据集中相关点很少，这时候我们可以去除已经得出的相关点，
利用剩下的点来重建最小生成树，再次得到一组相关点

Algorithm 2. Algorithm modified MS-SVM.

```
Input: The dataset and Threshold as the acceptable difference between the
       training accuracies.
Output: The classification accuracy.
Create first level folds $train_i$-$test_i$, $i \in (1,\ldots,10)$
For each $train_i$ create the second level folds $train_{ij}$-$test_{ij}$, $j \in (1,\ldots,10)$
$SVMAccuracy = 0$, $MSSVMAccuracy = 0$
for $i = 1$ to 10 do
  for $j = 1$ to 10 do
    for all SVM-Parameter-Combinations do
      $SVM_{ij}$ = Train a SVM on $train_{ij}$
      $SVMAccuracy_{ij}$ = Testing $test_{ij}$ using $SVM_{ij}$
      $SVMTrainAccuracy_{ij}$ = Testing $train_{ij}$ using $SVM_{ij}$
      $RelevantPoints_{ij} = GetRelevantPts(train_{ij}, SVMTrainAccuracy_{ij}, Threshold,$
      $SVM\text{-}Parameter\text{-}Combination)$
      $MSSVM_{ij}$ = Train a SVM on $RelevantPoints_{ij}$
      $MSSVMAccuracy_{ij}$ = Testing $test_{ij}$ using $MSSVM_{ij}$
    end for
  end for
  $SVMParameter_i = argmax\,(SVMAccuracy_{ij})$
  $MSSVMParameter_i = argmax\,(MSSVMAccuracy_{ij})$
  $SVM_i$ = Train a SVM on $train_i$ using $SVMParameter_i$
  $SVMAccuracy_i$ = Testing $test_i$ using $SVM_i$
  $SVMAccuracy = SVMAccuracy + SVMAccuracy_i$
  $SVMTrainAccuracy_i$ = Testing $train_i$ using $SVM_i$
  $RelevantPoints_i = GetRelevantPts\,(train_i, SVMTrainAccuracy_i, Threshold,$
  $MSSVMParameter_i)$
  $MSSVM_i$ = Train a SVM on $RelevantPoints_i$ using $MSSVMParameter_i$
  $MSSVMAccuracy_i$ = Testing $test_i$ using $MSSVM_i$
  $MSSVMAccuracy = MSSVMAccuracy + MSSVMAccuracy_i$
end for
$SVMAccuracy = SVMAccuracy/10$
$MSSVMAccuracy = MSSVMAccuracy/10$
```

Algorithm 3. GetRelevantPts.

```
Input: $train, SVMTrainAccuracy, Threshold, SVMParaComb$
Output: $RelevantPoints$
$traindummy = train$
repeat
  $MST$ = Build a MST on $traindummy$
  $NewRelevantPoints$ = Adjacent datapoints in $MST$ belonging to different classes
  $RelevantPoints = RelevantPoints \bigcup NewRelevantPoints$
  $MSSVM$ = Train a SVM on $RelevantPoints$ using SVMParaComb
  $MSSVMTrainAccuracy$ = Testing $train$ using $MSSVM$
  if $SVMTrainAccuracy\_MSSVMTrainAccuracy > Threshold$ then
    $traindummy = traindummy - NewRelevantPoints$
    $flag = False$
  else
    $flag = True$
  end if
until $flag = True$
return RelevantPoints
```

For further investigation, we have studied the performance of SVM and MS-SVM algorithms on some high dimensional bench-

Table 8
Performance comparison of linear SVM and linear MS-SVM for high dimensional datasets.

Datasets (no. of points, no. of features)	Level	SVM		MS-SVM		% Reduction of # of SVs
		Avg. acc.(Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Colon (62, 2000)	Original	81.13 (1.71)	30.56	72.58 (2.63)	19.23	37.07
	First		30.56	78.23 (2.97)	27.11	11.29
	Second		30.56	80.16 (2.95)	29.61	3.11
Leukemia1 (34, 7129)	Original	88.53 (3.24)	20.14	79.12 (3.52)	9.36	53.53
	First		20.14	86.18 (3.93)	14.51	27.95
	Second		20.14	87.35 (3.12)	18.04	10.43
Leukemia2 (38, 7129)	Original	97.37 (0)	20.23	91.05 (1.84)	7.32	63.82
	First		20.23	95.26 (1.66)	12.67	37.37
	Second		20.23	96.05 (2.56)	15.15	25.11

Table 9
Performance comparison of non-linear SVM and non-linear MS-SVM for high dimensional datasets.

Datasets	Level	SVM		MS-SVM		% Reduction of # of SVs
		Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs	
Colon	Original	64.52 (1.45)	55.80	42.26 (3.47)	19.66	64.77
	First		55.80	59.68 (5.38)	35.26	36.81
Leukemia1	Original	58.82 (0)	30.60	44.12 (5.88)	9.46	69.08
	First		30.60	37.65 (10.07)	18.56	39.35
Leukemia2	Original	71.05 (0)	34.20	35.79 (4.84)	7.32	78.60
	First		34.20	27.89 (5.98)	13.81	59.62

mark datasets, the results of which are reported in Table 8 and Table 9. The comparison between Linear SVM and Linear MS-SVM in Table 8 shows that MS-SVM substantially lags behind SVM, though the number of support vectors are remarkably low for the MS-SVM algorithm. However, when the first level neighbors are considered for learning the model, as shown in the second row of Table 8 for each dataset, the performance of MS-SVM algorithm comes within an acceptable range with that of SVM. Note that, the MS-SVM algorithm has drastically reduced the number of support vectors for Leukemia1 and Leukemia2 datasets, while that of Colon is considerably low. In the third row of Table 8 for each dataset we report the results when the second level neighbors are considered, where the classification accuracies obtained by MS-SVM algorithm almost matches with that of SVM at the cost of more support vectors.

However, the performance of both kernal SVM and kernal MS-SVM as reported in Table 9 on these high dimensional datasets is poor. Comparing between Table 8 and Table 9 we can say that SVM in the feature space after transforming the data points is unable to find a separating hyperplane and thus is unable to match up with its linear version. Similarly, the kernal MS-SVM with reduced number of data points fails to match its linear counterpart. In this case the number of support vectors in Table 9 are of no importance as the classification accuracies for both kernal SVM and kernal MS-SVM are not acceptable for these datasets.

To complete the work, we have also compared the proposed MS-SVM algorithm with an existing algorithm for reduction of number of support vectors as in [3,10]. In [3,10], the data points of the two classes are clustered separately using *K-Means* clustering algorithm and only the cluster centers are used for training the SVM. In principle, these algorithms are not comparable to ours because we use data points, but those algorithms use cluster centers. Moreover, the centers of *K-Means* will fall in dense regions, and if the value of k is small the centers will be away from the boundary, consequently the centers may not be effective to define class boundaries. However, if we increase the value of k, there may be some centers relatively close to the boundary. Since, most of the centers in this existing method is expected to be away from the boundary, which are not likely to influence the separating hyperplane of SVM, the number of support vectors using this method will be low compared to that of the MS-SVM algorithm. Moreover, to use the algorithms in [3,10] one needs to decide on the choice of k, which appears difficult as this k is certainly not the number of natural clusters. The comparative results are given in Table 10 and Table 11. For a fair comparison we have set the value of k for each class equal to the number of data points selected after applying the minimum spanning tree algorithm on the training data by MS-SVM. In Table 10 we compare the performance of linear MS-SVM and KMeans-SVM using the original algorithm and with first level neighbors. Similarly, Table 11 report the same results with the kernal version of the two algorithms. These results show that the classification accuracy obtained by KMeans-SVM and MS-SVM are similar, at least by considering the first level neighbors. MS-SVM is found to beat KMeans-SVM in terms of classification accuracy, usually when the dimension and/or the number of data points are high. The number of support vectors obtained by KMeans-SVM is mostly marginally less than that obtained by MS-SVM. The KMeans-SVM algorithm is, however, not able to compete with MS-SVM for the Ring dataset as shown in Table 12. In this case the average classification accuracy obtained by KMeans-SVM is much lower than that of MS-SVM. Moreover, KMeans-SVM shows a very high standard deviation. Which is likely, since the initial centers are randomly chosen by KMeans algorithm resulting into different positioning of the separating hyperplane.

The Kmeans-SVM and MS-SVM algorithms produce apparently similar outputs, however, as mentioned earlier KMeans-SVM requires the value of k as an external input. Note that, since in KMeans-SVM the two different classes are clustered separately, two values of k are to be provided as external inputs, where the ideal value of k for each class will depend upon the data distribution. The procedure to find the ideal value of k is not studied in [10] or [3]. In [9] the authors have proposed a way for finding the value of k,

Table 10
Performance comparison of linear KMeans-SVM and linear MS-SVM.

Datasets	Level	KMeans-SVM		MS-SVM	
		Avg. acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs
Synthetic	Original	92.00 (0.78)	11.63	91.65 (0.47)	26.67
	First	92.15 (0.75)	13.95	92.25 (0.35)	30.98
Haberman	Original	74.77 (0.65)	101.12	73.10 (0.41)	126.60
	First	75.20 (0.29)	126.55	72.84 (0.45)	147.34
Ionos	Original	85.47 (1.07)	37.22	82.76 (1.06)	55.76
	First	84.59 (1.32)	53.59	83.93 (1.65)	58.45
Liver	Original	65.86 (2.16)	130.61	64.35 (1.39)	146.96
	First	67.04 (1.19)	181.3	66.52 (1.02)	189.83
Satellite Image	Original	63.63 (0.90)	355.30	52.04 (1.67)	361.20
	First	69.69 (1.10)	597.95	60.96 (0.54)	744.66
Sonar	Original	72.88 (2.66)	41.24	71.06 (2.96)	54.93
	First	73.41 (2.02)	60.34	75.24 (2.91)	73.69
Scene	Original	76.14 (0.65)	223.87	85.45 (0.42)	284.67
	First	89.92 (0.31)	327.87	92.41 (0.18)	358.79
Yeast ML8	Original	92.64 (0)	311.97	92.64 (0)	328.40
	First	92.63 (0.01)	361.39	92.64 (0)	445.66
Banana	Original	51.91 (0.71)	858.52	55.19 (0.14)	868.90
	First	54.54 (0.71)	1104.89	55.01 (0.43)	1097.34
US Crime	Original	87.11 (0.46)	112.52	93.36 (0.19)	208.97
	First	90.13 (0.45)	147.77	93.73 (0.20)	245.89
Colon	Original	81.29 (2.43)	11.92	72.58 (2.63)	19.23
	First	79.52 (3.88)	20.97	78.23 (2.97)	27.11
Leukemia1	Original	91.76 (1.86)	8.21	79.12 (3.52)	9.36
	First	90.88 (2.92)	13.79	86.18 (3.93)	14.51
Leukemia2	Original	96.84 (2.08)	5.86	91.05 (1.84)	7.32
	First	97.11 (0.83)	10.09	95.26 (1.66)	12.67

Table 11
Performance comparison of non-linear KMeans-SVM and non-linear MS-SVM.

Datasets	Level	KMeans-SVM		MS-SVM	
		Avg. Acc. (Std. Dv.)	Avg. no. of SVs	Avg. acc. (Std. Dv.)	Avg. no. of SVs
Synthetic	Original	91.90 (0.84)	14.98	91.65 (0.58)	27.57
	First	91.55 (0.98)	18.12	91.25 (1.27)	32.45
Haberman	Original	74.22 (1.00)	105.1	72.58 (1.32)	122.78
	First	74.35 (0.62)	126.89	72.88 (1.08)	144.30
Ionos	Original	93.50 (0.73)	44.04	89.97 (1.25)	52.37
	First	93.96 (0.97)	53.45	92.59 (0.87)	59.29
Liver	Original	68.26 (2.01)	145.41	67.19 (1.71)	163.21
	First	68.99 (1.24)	191.65	68.58 (1.01)	204.60
Satellite Image	Original	92.97 (0.10)	588.74	94 (0.09)	564.92
	First	93.16 (0.11)	576.08	94.12 (0.09)	805.3
Sonar	Original	84.23 (0.93)	61.64	83.85 (1.40)	67.56
	First	86.15 (1.39)	86.37	85.19 (2.30)	87.48
Scene	Original	92.76 (0.08)	330.02	92.64 (0.05)	328.39
	First	92.84 (0.07)	626.09	92.74 (0.07)	578.99
Yeast ML8	Original	92.64 (0)	359.82	92.64 (0)	381.76
	First	92.63 (0.02)	515.70	92.64 (0)	595.53
Banana	Original	90.47 (0.13)	1220.16	86.85 (0.51)	837.26
	First	90.33 (0.17)	338.04	89.96 (0.11)	986.45
US Crime	Original	92.68 (0.10)	212.55	93.41 (0.24)	211.98
	First	92.48 (0)	313.88	93.76 (0.14)	247.12
Colon	Original	42.23 (3.47)	19.66	42.26 (3.47)	19.66
	First	59.68 (5.38)	35.26	59.68 (5.38)	35.26
Leukemia1	Original	44.12 (5.88)	9.46	44.12 (5.88)	9.46
	First	37.65 (10.07)	18.56	37.65 (10.07)	18.56
Leukemia2	Original	35.79 (4.84)	7.32	35.79 (4.84)	7.32
	First	27.89 (5.98)	13.81	27.89 (5.98)	13.81

Table 12
Performance comparison between KMeans-SVM and MS-SVM for Ring dataset.

Datasets	KMeans-SVM		MS-SVM	
	Average accuracy (Std. Dv.)	Average no. of SVs	Average accuracy (Std. Dv.)	Average no. of SVs
Ring	88.37 (11.3675)	8.91	99.67 (0.4157)	9.25

where it starts with k as the number of points in about 5% of the dataset and then k is increased till some criterion on accuracy of the classifier is satisfied. This is a tedious procedure and assumes the same value of k for both classes, which may not be desirable. The proposed MS-SVM algorithm however do not suffer this drawback.

4. Conclusion and discussion

The support vector machine algorithm for classification is widely used in various application domains. But, the time required for classification, which increases with the number of support vectors, may not make it feasible for some real time applications when the support vectors are large in number. Again, training the SVM with a reduced dataset will reduce the learning time, as the optimization problem will have less data points, especially when the actual dataset is large. In this work we have proposed a novel algorithm, Minimally Spanned Support Vector Machine (MS-SVM), for reducing the number of support vectors. The results of the MS-SVM algorithm on various benchmark datasets reveal that we have been able to reduce the number of support vectors to a great extent maintaining almost same overall classification accuracy and even sometimes surpassing it. We have also demonstrated that, if the overall classification accuracy is low, we can include the neighboring edges iteratively till acceptable overall classification accuracy is obtained. Again, the MS-SVM algorithm turns into normal SVM algorithm if we keep on adding the neighboring edges iteratively. We have even shown that the minimum spanning tree can be build in the feature space instead of the input space. The problem thrown by the Ring dataset is solved by a slightly modified MS-SVM algorithm. The modification made to solve the Ring dataset can also be applied to any other dataset if the associated MST has a very few edges connecting data points from the two classes and the MS-SVM yields a poor classification accuracy. The MS-SVM algorithm has also been compared with an existing algorithm, and found to produce similar results. However, the existing method suffers from the drawback of external input, which can be tedious to fix. The MS-SVM algorithm, on the other hand, has no such drawback.

Our sample selection method is, in some sense, tied up with the SVM classifier. SVM does not make use of density information of the data. It finds a separating hyperplane maximizing the margin and hence use of the boundary points is good enough to learn the classifier. However, most other classifiers implicitly or explicitly depend on class densities/distribution/shape of the training data. For such classifiers, although this philosophy is expected to work, we cannot say that the selected samples will always do a good job. For example, we have done some experiments, using the first two benchmark datasets, Haberman and Ionos with Multilayer Perceptron (MLP) as the classifier. For these two datasets, the performance of the MLP designed using the selected samples and that of the system designed with all data points are very similar. On the other hand, for the Ring data set, with a few training data points, an MLP may not do a god job. However, we may make the sample selection method tied up to the specific classifier that will use the sample. Thus, we need to use an MLP, instead of an SVM, to decide on the number of samples to choose from, when the classifier is an MLP. If we do so, even an MLP does a reasonably good job for this data set.

References

[1] V. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag, 1995.
[2] B. Scholkopf, A. Smola, R.C. Williamson, P.L. Bartlett, New support vector algorithms, Neural Comput. 12 (2000) 1207–1245.
[3] Q.-A. Tran, Q.-L. Zhang, X. Li, Reduce the number of support vectors by using clustering techniques, Proceedings of the Second International Conference on Machine Learning Cybernetics, Xi'an (2003) 1245–1248.
[4] T. Downs, K.E. Gates, A. Masters, Exact simplification of support vector solutions, J. Mach. Learn. Res. 2 (2001) 293–297.
[5] D. Geebelen, J.A. Suykens, J. Vandewalle, Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation, IEEE Trans. Neural Netw. Learn. Syst. 23 (4) (2012) 682–688.
[6] T. Joachims, Making Large-Scale SVM Learning Practical. Advances in Kernel Methods-Support Vector Learning, MIT-Press, 1999, pp. 169–184.
[7] S. Sohn, C.H. Dagl, Advantages of using fuzzy class memberships in self-organizing map and support vector machines, Proceedings of the International Joint Conference on Neural Networks (2001).
[8] R. Koggalage, S. Halgamuge, Reducing the number of training samples for fast support vector machine classification, Neural Inf. Process. 2 (3) (2004) 57–65.
[9] X.-L. Xia, M.R. Lyu, T.-M. Lok, G.-B. Huang, Method of decreasing the number of support vectors via K-mean clustering, Adv. Intell. Comput. 3644 (2005) 717–726.
[10] P. Kumar, I. Makki, D. Filev, A Non-Intrusive Three-Way Catalyst Diagnostics Monitor Based on Support Vector Machines, SMC, 2014.
[11] R.L. Graham, P. Hell, On the history of the minimum spanning tree problem, Ann. Hist. Comput. 7 (1) (1985) 43–57.
[12] R.C. Prim, Shortest connection networks and some generalizations, Bell Syst. Tech. J. 36 (6) (1957) 1389–1401.
[13] N. Manh Cuong, N. Van Thien, A method for reducing the number of support vectors in fuzzy support vector machine, in: T. Nguyen, T. van Do, H. An Le Thi, N. Nguyen (Eds.), Advanced Computational Methods for Knowledge Engineering, Advances in Intelligent Systems and Computing, 453, Springer, Cham, 2016.
[14] Y.-J. Lee, O.L. Mangasarian, RSVM: reduced support vector machines, Proceedings of the SIAM International Conference on Data Mining (2001).
[15] J.W. Sammon Jr., A nonlinear mapping for data structure analysis, IEEE Trans. Comput. C-18 (May (5)) (1969).