Efficient Algorithm for Localized Support Vector Machine

Haibin Cheng, Pang-Ning Tan, *Member, IEEE*, and Rong Jin, *Member, IEEE*

Abstract—This paper presents a framework called **Localized Support Vector Machine** (LSVM) for classifying data with nonlinear decision surfaces. Instead of building a sophisticated global model from the training data, LSVM constructs multiple linear SVMs, each of which is designed to accurately classify a given test example. A major limitation of this framework is its high computational cost since a unique model must be constructed for each test example. To overcome this limitation, we propose an efficient implementation of LSVM, termed **Profile SVM** (PSVM). PSVM partitions the training examples into clusters and builds a separate linear SVM model for each cluster. Our empirical results show that (1) LSVM and PSVM outperform nonlinear SVM for all twenty of the evaluated data sets; and (2) PSVM achieves comparable performance as LSVM in terms of model accuracy but with significant computational savings. We also demonstrate the efficacy of the proposed approaches in terms of classifying data with spatial and temporal dependencies.

Index Terms—Classification, Support Vector Machine, Kernel-based learning, Local learning.

1 Introduction

One of the important decisions one has to make in building classification models is whether to use a global or local learning approach. Global learning approaches attempt to build a complex model by using the global characteristics of the data. Nonlinear Support Vector Machine (SVM) is a popular global learning technique, which has been successfully applied to a wide range of applications, including text categorization [19], face detection [26], gene function prediction [7], and land cover classification [24]. The basic idea behind this technique is to find an optimal hyperplane in a high-dimensional feature space that maximizes the margin of separation between the closest training examples from different classes. However, one of the main difficulties in global learning techniques is the model selection problem. More precisely, one needs to select a suitable model and its corresponding parameters to represent the observed data. For example, a nonlinear SVM must employ sophisticated kernel functions to classify data sets with complex decision surfaces. Determining the right parameters for such kernel functions is not only challenging [12], the resulting models are also susceptible to overfitting due to their large VC dimensions [8]. Instead of fitting a global model to the entire training data, another strategy is to construct models based on the local neighborhood information of each test example. Recent research has demonstrated that such a local learning strategy is superior to global learning, especially on data sets that are not evenly distributed [6][35][13][23]. A well-known classification technique that employs such a strategy is the K-nearest neighbor (KNN) classifier [10]. The advantage of using KNN is that it does not require making a biased assumption about distribution of the data nor about the shape of the decision surfaces [2]. Nevertheless, because of its lazy learning strategy, classifying test examples can be computationally expensive.

This paper presents a framework known as *Localized Support Vector Machine* (**LSVM**), which leverages the strengths of both SVM and KNN. Instead of fitting a sophisticated kernel function to the training data, LSVM builds multiple linear SVM models by considering the training examples located in the vicinity of each test example. We empirically show that such a strategy often leads to significant performance improvement over nonlinear SVM and thus avoids the thorny issue of model selection.

However, since LSVM builds a unique model for each test example, it is impractical when the number of test examples is large. To overcome this limitation, we develop an efficient implementation of the algorithm, known as *Profile Support Vector Machine* (**PSVM**). The intuition behind this algorithm is that many test examples with similar neighboring training examples tend to share a common set of support vectors. The PSVM algorithm is therefore designed to exploit this property by employing a clustering algorithm to partition the training data into clusters and training a local SVM model for each cluster. The clustering objective in PSVM is somewhat different from standard unsupervised clustering algorithms. In addition to partitioning the training examples based on their similarities, each cluster must contain representative examples from both positive and negative classes to enable the construction of a local SVM model. After constructing the local models, a test example is classified by finding the nearest cluster centroid and invoking its corresponding local SVM model. Our experimental results show that this strategy enables PSVM to maintain the high accuracy of LSVM without its costly computational overhead.

We also extend the LSVM and PSVM framework to data sets with strong spatial or temporal dependencies, i.e., data sets whose classes are not evenly distributed across different locations or time. For example, the

Haibin Cheng is with the Yahoo! Labs, Santa Clara, CA, 95054. Email hcheng@yahoo-inc.com. Pang-Ning Tan is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 48823. Email ptan@msu.edu. Rong Jin is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 48823. Email rongjin@msu.edu.

classification of housing prices often depend on their regions due to social, economical, or environmental factors. Similarly, topics in blogs or email messages tend to exhibit temporal locality. Recent research on spatial and temporal prediction has demonstrated the superior performance of building local models compared with global models [13][23]. Our experimental results using real-world spatial and temporal data sets also demonstrate a significant performance improvement by LSVM and PSVM over both nonlinear SVM and KNN algorithms.

The remainder of this paper is organized as follows. In Section 2, we briefly review two classification techniques—the KNN classifier and support vector machine. Section 3 presents our proposed LSVM formulation, followed by a description of the PSVM algorithm in Section 4. Section 5 illustrates the application of LSVM and PSVM algorithms to spatial and temporal data. Section 6 presents our experimental results and Section 7 concludes with a summary of our contributions and suggestions for future work.

2 PRELIMINARIES

Consider a set of n labeled training examples, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \Re^d$ is a d-dimensional input vector for the i^{th} training example and $y_i \in \{-1, +1\}$ is its corresponding class label. Let $\overline{\mathcal{D}} = \{\overline{\mathbf{x}}_i\}_{i=1}^m$ denote a set of m unlabeled test examples. The goal of classification is to predict the class label of each test example $\overline{\mathbf{x}} \in \overline{\mathcal{D}}$ using information derived from $\mathcal{D}$. This section presents two well-known techniques—K-nearest neighbor classifier and support vector machine—that form the basis for our proposed LSVM framework.

2.1 Nearest Neighbor Classifier

The K-nearest neighbor [10] (KNN) classifier predicts the class label of a test example based on the distribution of training examples in its local neighborhood. One of the appealing features of the KNN classifier is its flexible representation of decision surfaces. Nevertheless, it is susceptible to the curse of dimensionality [5] problem and is sensitive to the choice of neighborhood size K [15]. When K is too small, the classifier may overfit the training data. On the other hand, a large K may also hurt the performance of the classifier by incorporating training examples from other classes that are unsuitable to the prediction of a test example.

To overcome the difficulty of choosing K, the weighted K-nearest neighbor [17] classifier was introduced. In this method, the influence of each training example $\mathbf{x}_i$ is weighted by its similarity to the test example $\overline{\mathbf{x}}$. The probability that a test example belongs to class y is computed as follows:

$$p(y|\overline{\mathbf{x}}) = \frac{\sum_{i=1}^n \delta(y, y_i)\sigma(\mathbf{x}_i, \overline{\mathbf{x}})}{\sum_{i=1}^n \sigma(\mathbf{x}_i, \overline{\mathbf{x}})}, \quad (1)$$

where $\sigma(\mathbf{x}_i, \overline{\mathbf{x}})$ denote the similarity between $\mathbf{x}_i$ and $\overline{\mathbf{x}}$ and

$$\delta(y, y_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

The similarity measure is often normalized so that it ranges between 0 and 1. One possible choice for the similarity measure is the radial basis function (RBF) kernel [27]

$$\sigma(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|_2^2/\mu\right), \quad (3)$$

where μ is a user-specified parameter.

Several variants of the nearest-neighbor classifier have been proposed in the literature to overcome some of its present limitations. Hastie et al. [16] and Weinberger et al. [34] attempted to address the curse of dimensionality problem by estimating the effective metric for computing the neighborhoods of the test examples. Domeniconi et al. [11] proposed a feature weighting scheme to alleviate the curse of dimensionality problem. Vincent and Bengio [33] developed the HKNN algorithm to fix the *missing sample* problem, which was shown to introduce artifacts in the decision surfaces produced by regular KNN. Both of these methods, however, are computationally expensive when the number of test examples is large. Another drawback is the difficulty in choosing the appropriate value for K [15].

2.2 Support Vector Machine

Support vector machine (SVM) is an effective technique for classifying high-dimensional data [19]. Unlike the nearest-neighbor classifier, SVM learns the optimal hyperplane that separates training examples from different classes by maximizing the classification margin [14][8]. It is also applicable to data sets with nonlinear decision surfaces by employing a technique known as the kernel trick [1], which projects the input data to a higher dimensional feature space, where a linear separating hyperplane can be found. SVM avoids the costly similarity computation in high-dimensional feature space by using a surrogate kernel function.

Formally, a nonlinear SVM model is trained to optimize the following objective function:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \beta \sum_{i=1}^n \xi_i \\ \text{s. t} \quad & y_i[\mathbf{w}^\top \varphi(\mathbf{x}_i) - b] \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, 2, \cdots, n, \end{aligned} \quad (4)$$

where $\varphi(\mathbf{x}_i)$ is a function that maps $\mathbf{x}_i$ to its higher dimensional feature space, ξ_i is the error in misclassification, and β is a parameter that controls the tradeoff between the classification margin and cost of misclassification. The preceding optimization function can be

transformed into its dual form using the Lagrange multiplier method:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \qquad (5) \\ \text{s. t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \beta, \ i = 1, 2, \cdots, n, \end{aligned}$$

where α_i is a Lagrange multiplier and $K(\mathbf{x}_i, \mathbf{x}_j)$ is the surrogate kernel function used to compute the dot product $\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$. Once the SVM model has been trained, the class label of a test example $\overline{\mathbf{x}}$ is predicted as follows:

$$\begin{aligned} y &= \text{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\overline{x}) \Big) \\ &= \text{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i K(\overline{\mathbf{x}}, \mathbf{x}_i) \Big). \qquad (6) \end{aligned}$$

Notice that the prediction depends only on training examples with non-zero weights α_i. Such training examples are known as the support vectors.

Finding the right kernel for a given data set can be quite a challenging task. On the one hand, a simple kernel may not capture all the intricacies of a complex decision surface. On the other hand, a kernel that is too flexible is susceptible to model overfitting. Although there have been several studies devoted to kernel learning [3], the proposed methods are generally expensive and do not scale well to very large data sets.

SVM can be extended to handle multi-class problems [18] using methods such as the one-versus-all strategy [29]. Given a data set with N classes, we may construct N binary SVM models $\{f_1, f_2, \cdots, f_N\}$, where each f_i is trained to separate training examples that belong to class i from all other classes. The class label of a test example $\overline{\mathbf{x}}$ can be predicted by applying the N binary SVM models and choosing the class i that yields the highest output value $f_i(\overline{\mathbf{x}})$.

3 Localized Support Vector Machine (LSVM)

Zhang et al. [35] have recently developed a hybrid algorithm called KNN-SVM [35] that constructs a local SVM model for each test example based on its K-nearest neighbors [28]. Their algorithm, which is a straightforward adaptation of KNN to SVM, has several limitations. First, the performance of the algorithm depends on the choice of K. Second, the model is not as flexible because it uses the same neighborhood size for each test example. Finally, the algorithm does not consider the similarity between the training and test examples when constructing the local SVM models. Once the nearest neighbors have been found, it will identify the local support vectors by solving the optimization problem given in (5).

The limitations of KNN-SVM have motivated us to develop a more general framework called Localized Support Vector Machine (LSVM), which incorporates the similarity information directly into SVM learning. LSVM is designed to find support vectors that are close to the test example and to reduce the influence of those located far away from a test example. This is accomplished by weighting the misclassification error of each training example according to its similarity to the test example.

Let $\sigma(\overline{\mathbf{x}}_s, \mathbf{x}_i)$ be the similarity between the test example $\overline{\mathbf{x}}_s$ and the training example $\mathbf{x}_i$. For each $\overline{\mathbf{x}}_s \in \overline{\mathcal{D}}$, we construct a local SVM model by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \beta \sum_{i=1}^{n} \sigma(\overline{\mathbf{x}}_s, \mathbf{x}_i)\xi_i \qquad (7) \\ \text{s. t} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \ i = 1, 2, \ldots, n \end{aligned}$$

The first term in the objective function corresponds to the classification margin while the second term penalizes the model if it misclassifies the training examples. Note that the more similar a training example is to the test example, the higher its penalty for misclassification. As a result, the local SVM model is more strongly influenced by training examples that are similar to the test example than by those that are dissimilar.

To further understand the effect of incorporating a similarity function into the SVM formulation, consider the dual form of the preceding optimization problem:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \qquad (8) \\ \text{s. t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \beta\sigma(\overline{\mathbf{x}}_s, \mathbf{x}_i), \ i = 1, 2, \ldots, n, \end{aligned}$$

where a linear kernel is employed for the local SVM. The details of the derivation of this dual form can be found in the Appendix section. Unlike the optimization problem given in (5) for nonlinear SVM, the upper bound for α_i is modified from β to $\beta\sigma(\overline{\mathbf{x}}_s, \mathbf{x}_i)$. Such modification leads to the following two consequences:

1) **Some of the support vectors for nonlinear SVM are no longer support vectors for LSVM.** As previously noted, a support vector for nonlinear SVM corresponds to a training example with $\alpha_i > 0$ while a non-support vector has $\alpha_i = 0$. Since the upper bound $\beta\sigma(\overline{\mathbf{x}}_s, \mathbf{x}_i)$ decreases to zero when $\overline{\mathbf{x}}_s$ and $\mathbf{x}_i$ are dissimilar, a support vector for nonlinear SVM that is dissimilar to the test example will have an upper bound close to zero, converting it into a non-support vector for LSVM.
2) **Some of the non-support vectors for nonlinear SVM become support vectors for LSVM**. As the α_i for support vectors that are dissimilar to the test example reduces to zero, some of the training

examples that are similar to the test example will be assigned weights larger than zero to ensure that the equality constraint $\sum_{i=1}^{n} \alpha_i y_i = 0$ is preserved.

There are two variants of LSVM considered in this study. If σ is a real-valued similarity measure defined on the closed interval $[0, 1]$, the resulting algorithm is called *Soft Localized Support Vector Machine* (**SLSVM**). We employ the RBF kernel as our similarity measure for SLSVM, similar to the approach taken by weighted KNN (see Equation 3). On the other hand, if σ is a binary-valued function, the resulting algorithm is called *Hard Localized Support Vector Machine* (**HLSVM**). The similarity measure $\sigma(\overline{\mathbf{x}}_s, \mathbf{x}_i)$ is equal to 1 if $\mathbf{x}_i$ is part of the nearest neighbor list for $\overline{\mathbf{x}}_s$, and 0 otherwise. For HLSVM, the upper bound constraint for α_i is equal to C, which is equivalent to the approach used in KNN-SVM. Thus, the KNN-SVM algorithm proposed by Zhang et al. in [35] is a special case of our proposed LSVM framework. Furthermore, LSVM can handle multi-class problems by utilizing the one-versus-all strategy described in Section 2.2.

4 Profile Support Vector Machine (PSVM)

Since LSVM builds a local SVM model for each test example, applying it to a large test set can be very costly. PSVM aims to improve the efficiency of LSVM by reducing the number of local SVM models that need to be constructed. To understand the intuition behind this approach, consider the optimization problem given in (8), which must be solved separately for each test example $\overline{x}_s$. Notice that the optimization problem is nearly identical for each test example, except for the upper bound constraint on α_i, which depends on $\sigma(\overline{\mathbf{x}}_s, \mathbf{x}_i)$. Let $\vec{\sigma}(\overline{\mathbf{x}}_s) = \Big[\sigma(\overline{\mathbf{x}}_s, \mathbf{x}_1), \sigma(\overline{\mathbf{x}}_s, \mathbf{x}_2), \ldots, \sigma(\overline{\mathbf{x}}_s, \mathbf{x}_n)\Big]^T$ denote a column vector of similarity values between a test example $\overline{\mathbf{x}}_s$ and the set of training examples $\{\mathbf{x}_i\}_{i=1}^n$. Since α_i determines whether $\mathbf{x}_i$ is a support vector, we expect test examples with highly correlated similarity vector $\vec{\sigma}$ to share many common support vectors. For such test examples, it may be sufficient to build a single LSVM model to classify them. To determine the set of test examples that can be classified together, we develop a supervised clustering algorithm called MagKmeans.

4.1 MagKmeans: A Supervised Clustering Algorithm for PSVM

Let $\Sigma = \big[\vec{\sigma}(\overline{\mathbf{x}}_1)\ \vec{\sigma}(\overline{\mathbf{x}}_2) \ldots \vec{\sigma}(\overline{\mathbf{x}}_m)\big]$ be an $n \times m$ matrix, where n is the training set size, m is the test set size, and the (i, j)-th entry of the matrix denote the similarity between the training example $\mathbf{x}_i$ and the test example $\overline{\mathbf{x}}_j$. Our clustering task is somewhat different than conventional unsupervised clustering. First, the data to be clustered is Σ, which contains the similarity between every training-test example pair. Second, conventional clustering methods consider only the proximity between examples and ignore their class distributions. As a result, some clusters may not contain enough representative examples from the different classes to construct a reliable local SVM model.

In this paper, we propose the "**MagKmeans**" algorithm, which extends regular k-means [20] by considering the class distribution of training examples within each cluster. Let $Y = (y_1, y_2, \ldots, y_n)^T$ be the class labels of the training examples. The objective function for MagKmeans is:

$$\min_{C,Z} \quad \sum_{j=1}^{\kappa} \sum_{i=1}^{n} Z_{i,j} \|X_i - C_j\|_2^2 + R \sum_{j=1}^{\kappa} \left| \sum_{i=1}^{n} Z_{i,j} y_i \right| \quad (9)$$

where X_i is the i-th row of the similarity matrix Σ, C_j is an $1 \times m$ row vector representing the centroid of the jth cluster, R is a non-negative scaling parameter, and each $Z_{ij} \in \{0, 1\}$ is an element of the cluster membership matrix, whose value is equal to one if the ith training example belongs to the jth cluster, and zero otherwise. The first term in the objective function corresponds to a cluster cohesion measure. Minimizing this term would ensure that training examples in the same cluster have highly correlated similarity vectors. The second term in the objective function measures the skewness of class distributions in each cluster. Minimizing this term would ensure that each cluster contains a balanced number of positive and negative examples.

The cluster centroids C and cluster membership matrix Z are estimated iteratively as follows. First, we fix the cluster centroids and use them to determine the cluster membership matrix. Next, the revised cluster membership matrix is used to update the centroids. This procedure is repeated until the algorithm converges to a local minimum. To compute the cluster membership matrix Z, we transform the original optimization problem into the following form using κ slack variables t_j:

$$\begin{aligned} \min_{\{Z,t\}} \quad & \sum_{j=1}^{\kappa} \sum_{i=1}^{n} Z_{i,j} \|X_i - C_j\|_2^2 + R \sum_{j=1}^{\kappa} t_j \\ \text{s. t.} \quad & -t_j \leq \sum_{i=1}^{n} Z_{i,j} y_i \leq t_j \\ & t_j \geq 0,\ 0 \leq Z_{i,j} \leq 1 \\ & \sum_{j=1}^{\kappa} Z_{i,j} = 1 \end{aligned} \quad (10)$$

The preceding optimization problem can be solved using linear programming (LP) [31]. Note that the original LP formulation given in (9) assumes that each $Z_{i,j} \in \{0, 1\}$, which makes it an NP-hard problem. Although an optimal solution can be found using branch-and-bound techniques, such techniques are computationally expensive. To make the problem solvable in polynomial time, we relax the condition by allowing Z to be real-valued, ranging between the interval $[0, 1]$. While the solution of the relaxed LP problem may be suboptimal,

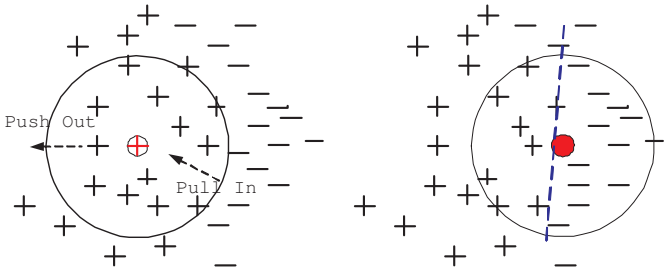

Fig. 1. An illustration of the MagKmeans clustering algorithm. The cluster in the left figure contains only positive exampels. After several iterations, some of the positive examples will be expelled from the cluster while some negative example are absorbed into the cluster to achieve balance in the class distribution (the right figure).

our experimental results suggest that the clusters are still good enough for building local SVM models that are as accurate as the SLSVM solution. After the cluster membership matrix has been obtained, the cluster centroids are updated based on the following equation:

$$C_j = \frac{\sum_{i=1}^{n} Z_{i,j} X_i}{\sum_{i=1}^{n} Z_{i,j}} \tag{11}$$

Figure 1 illustrates how the MagKmeans algorithm works. The initial cluster (the left figure) contains only positive examples. As the algorithm progresses, some positive examples are expelled from the cluster and replaced by the nearby negative examples (the right figure). By ensuring that the cluster has almost equal representation from each class, a local SVM model can be constructed from the training examples. A summary of the MagKmeans algorithm is given in Algorithm 1.

Algorithm 1 MagKmeans algorithm

Input: Similarity matrix Σ, class vector Y, and number of clusters κ
Output: Cluster membership matrix Z and the centroid matrix C

1: Randomly initialize the centroid matrix C.
2: **repeat**
3: Update the cluster membership matrix $Z_{i,j}$ by solving the linear programming problem given in (10).
4: Update the centroid matrix C using Equation (11).
5: **until** convergence

We illustrate the difference between the clustering results produced by regular k-means and MagKmeans using the synthetic data set shown in Figure 2. The data set contains two groups of labeled examples, where each group corresponds to a particular class. The figure in the middle shows the results of k-means, which partitions the data according to their classes, whereas the figure on the right shows the results of MagKmeans, which produces clusters that contain a mixture of training examples from both classes. The latter clustering is clearly more desirable for constructing local SVM models.

4.2 Model Building and Testing

After applying the MagKmeans algorithm, a local linear SVM model is constructed for each cluster. The SVM model for the kth cluster is obtained by solving the following optimization problem:

$$\begin{aligned} \max_{\widetilde{\alpha}} \quad & \sum_{i=1}^{n} \widetilde{\alpha}_i - \frac{1}{2} \sum_{i,j=1}^{n} \widetilde{\alpha}_i \widetilde{\alpha}_j y_i y_j \mathbf{x}_i \mathbf{x}_j \\ \text{s. t.} \quad & \sum_{i=1}^{n} \widetilde{\alpha}_i y_i = 0 \\ & 0 \le \widetilde{\alpha}_i \le \beta Z_{i,k}, \ i = 1, 2, \ldots, n \end{aligned}$$

Since Z is a binarized version of the cluster membership matrix found in (10), only the training examples assigned to the cluster will be used to build the local LSVM model.

To classify the test set, we need to determine the local SVM model that should be invoked for each test example. Let C be the $\kappa \times m$ centroid matrix produced by the MagKmeans algorithm. Since the (i, j)-th element of the centroid matrix indicates the average similarity between the test example $\overline{\mathbf{x}}_j$ and the training examples in cluster i, we assign $\overline{\mathbf{x}}_j$ to the cluster with highest similarity. We then apply the local SVM model associated with the cluster to predict the class label of $\overline{\mathbf{x}}_j$. The procedure for model building and testing of the PSVM algorithm is summarized in Algorithm 2. in which the Build_Local_SVM function is used to construct a local SVM model for each cluster identified by Z.

Algorithm 2 Profile SVM algorithm

Input: Training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, test set $\overline{\mathcal{D}} = \{\overline{\mathbf{x}}_i\}_{i=1}^{m}$, and number of clusters κ
Output: Class labels for the test set $\{y_j\}_{j=1}^{m}$

1: Compute the $n \times m$ similarity matrix Σ.
2: $(Z, C) \leftarrow$ MagKmeans(Σ, Y, κ)
3: **for** k = 1 to κ **do**
4: SVM$_k \leftarrow$ Build_Local_SVM($\mathcal{D}$, Z, k)
5: **end for**
6: **for** each test example $\overline{\mathbf{x}}_j \in \overline{\mathcal{D}}$ **do**
7: $k = \arg\max_i C_{i,j}$
8: y = SVM$_k(\overline{\mathbf{x}}_j)$
9: **end for**

In short, to avoid training a separate LSVM model for each test example, the PSVM algorithm partitions the data into κ clusters and trains a local SVM model for each cluster. κ is typically chosen to be considerably smaller than the number of test examples m to reduce its high computational cost. The scaling parameter R handles the tradeoff between the two terms in our objective function: cluster cohesion and class imbalance. We set R to $1/\kappa$ times the diameter of the data set in all of our experiments.

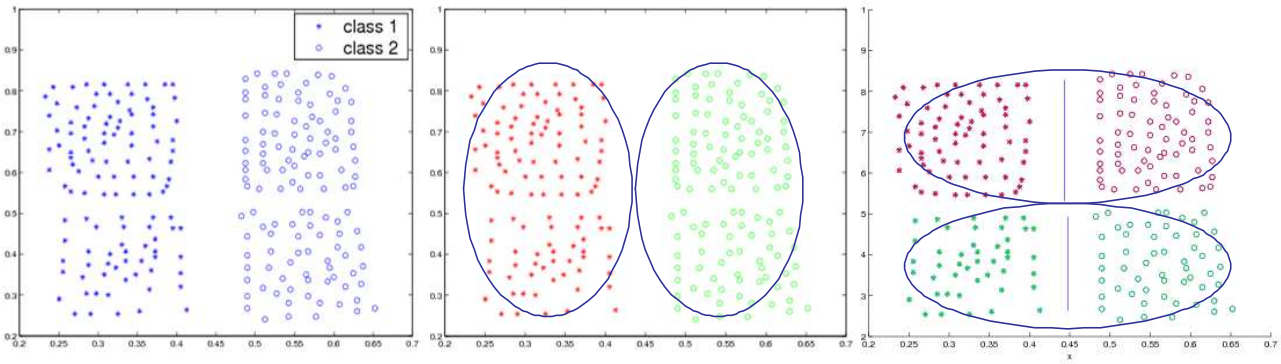

Fig. 2. Comparison between the clustering results of regular K-means (middle figure) and MagKmeans (right figure).

Finally, note that our PSVM formulation uses information from unlabeled data during the training process. However, unlike the transductive SVM formulation developed by Vapnik [32], our method is quasi-transductive because the unlabeled data is used for partitioning the input space and is not included as part of the objective function for building the SVM models.

5 LSVM and PSVM Formulations for Temporal and Spatial Data

This section describes how the LSVM and PSVM formulations can be extended to data with temporal [21] and spatial attributes [22]. For temporal data, each example is associated with an arrival time t and the prediction of a test example is often influenced by recently arrived training examples. Let $\overline{\mathbf{x}}_s$ be a test example that arrives at time $\bar{t}_s$ and $\mathbf{x}_i$ be a training example that arrives at an earlier time t_i. We may define a temporal similarity function $\sigma(t_i, \bar{t}_s)$ to measure the proximity between two examples in terms of their arrival times. Such timing information can then be incorporated into the LSVM framework by constraining the upper bound for α_i to $\beta\sigma(t_i, \bar{t}_s)$. The modified optimization problem for LSVM becomes:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s. t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \beta\sigma(t_i, \bar{t}_s), \ i = 1, 2, \ldots, n \end{aligned}$$

A PSVM formulation can also be developed to incorporate the temporal constraints. In this case, clustering is performed on the temporal similarity matrix $\Sigma = [\sigma(t_i, \bar{t}_s)]$ to partition the input space into homogeneous regions. Depending on the distribution of arrival times for training examples from each class, one could either apply MagKmeans or regular k-means to cluster the data. If the temporal neighborhood of a test example contains training examples from all classes, as in the case of the Enron data set considered in our experiments, it is sufficient to apply regular k-means. A local SVM model can then be constructed for each cluster partition.

Spatial constraints can be incorporated into the proposed framework in a similar manner. Let $\overline{\mathbf{x}}_s$ be a test example located at $\bar{l}_s$ and $\mathbf{x}_i$ is a training example located at l_i. The spatial proximity between the examples is given by the spatial similarity function $\sigma(l_i, \bar{l}_s)$. The function can be incorporated into LSVM by changing the upper bound for α_i from β to $\beta\sigma(l_i, \bar{l}_s)$. Therefore, the optimization problem for spatial LSVM is

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s. t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \beta\sigma(l_i, \bar{l}_s), \ i = 1, 2, \ldots, n \end{aligned}$$

We may reduce the computational overhead of spatial LSVM by performing clustering on the spatial similarity matrix $\Sigma = [\sigma(l_i, \bar{l}_s)]$ and building a spatial LSVM model for each cluster. Unlike temporal data, the spatial distribution of the classes are not as uniform, thus requiring a supervised algorithm such as MagKmeans to partition the data.

6 Experimental Evaluation

This section describes the experimental results obtained by applying the proposed algorithms to a variety of data sets. The LSVM algorithm is implemented by modifying the C++ code for the LIBSVM tool developed by Chang and Lin [9]. LIBSVM uses the one-versus-all approach as described in Section 2.2 for multi-class problems. For PSVM, we have also implemented the MagKmeans algorithm to cluster the similarity matrix Σ. Our experiments were conducted on a Windows XP machine with a 3.0GHz CPU and 1.0GB of RAM. The main objectives of our experiments are:

1) To compare the difference between the support vectors and decision surfaces obtained using LSVM and nonlinear SVM (Section 6.1).

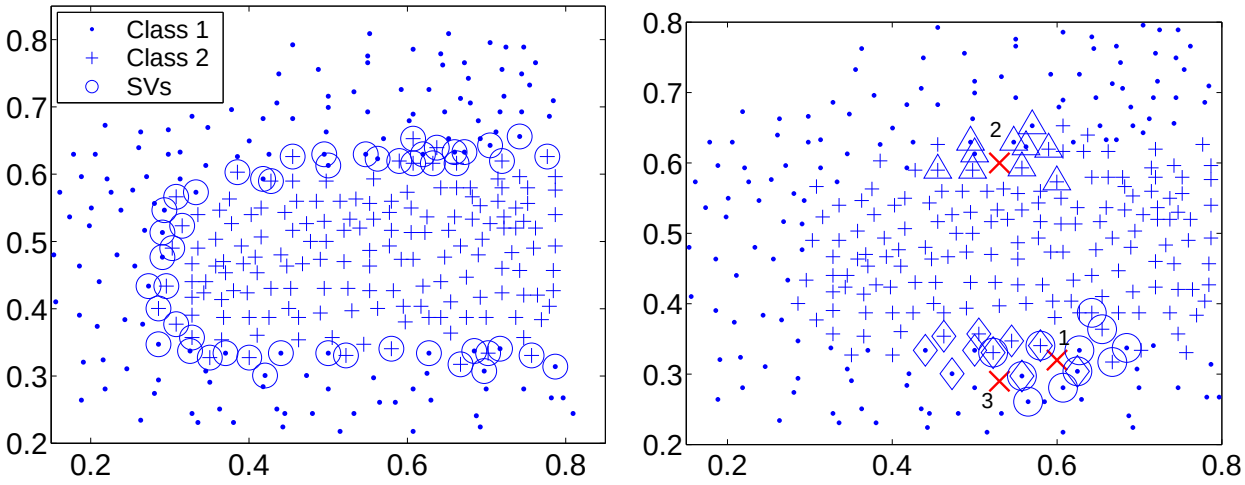

Fig. 3. Comparison between LSVM and nonlinear SVM. The left diagram shows the support vectors obtained by nonlinear SVM; whereas the right diagram shows the support vectors obtained for three test examples (marked as $\times$) using LSVM.

2) To compare the relative performance of KNN, SVM, LSVM, and PSVM on a variety of real-world data sets (Section 6.2).
3) To illustrate the application of LSVM and PSVM to temporal data (Section 6.3).
4) To illustrate the application of LSVM and PSVM to spatial data (Section 6.4).
5) To perform sensitivity analysis on parameters of the algorithms.

6.1 Comparison between LSVM/PSVM and Nonlinear SVM

For this experiment, we use synthetic data sets to demonstrate the difference between the support vectors and decision boundaries found by LSVM/PSVM against those found by nonlinear SVM. As previously mentioned in Section 3, LSVM reduces the influence of support vectors (for nonlinear SVM) that are far away from a test example and converts some of the training examples in its vicinity into support vectors.

6.1.1 Comparison of Support Vectors

Consider the synthetic data set depicted in Figure 3. The left diagram shows the data distributions for the two classes, represented as $\cdot$ and $+$, respectively. The support vectors found using a nonlinear SVM with an RBF kernel function are represented by the symbols $\odot$ and $\oplus$. Observe that the support vectors are located at the boundaries between the two classes. The right diagram of Figure 3 shows the corresponding support vectors found by LSVM for three selected test examples. The locations of the test examples are marked by the symbol $\times$ while their support vectors are represented by symbols O,$\triangle$ and $\diamondsuit$. Clearly, the number of support vectors associated with each test example is fewer than those produced by nonlinear SVM. Some support vectors for nonlinear SVM are not chosen as support vectors for LSVM because they are far away from the test examples. In addition, some non-support vectors for nonlinear SVM became support vectors for LSVM because of their proximity to the test examples. Furthermore, the test examples marked as 1 and 3 share several common support vectors because of their close proximity to each other. Even though their support vectors are not exactly identical, they have the potential of sharing the same decision surface, which justifies our motivation for using clustering to reduce the computational cost of LSVM.

6.1.2 Decision Surfaces for PSVM

Consider the three synthetic data sets shown in the top panel of Figure 4. The bottom panel shows the corresponding decision surfaces generated by PSVM. For the left-most data set, MagKmeans partitions the training data into six clusters, each of which is shown in a different color. After constructing their corresponding local SVMs, observe that the circular decision boundary of the original data is now approximated by a hexagon consisting of six linear decision boundaries. For the second data set shown in the middle diagram of Figure 4, the horseshoe shaped decision boundary is approximated by 11 piecewise linear decision boundaries. For the third data set shown in the right-most diagram, the spiral-shaped decision boundary is also approximated by 11 piecewise linear decision boundaries. In summary, the results of this experiment demonstrate the ability of PSVM to fit a complex decision boundary using multiple piecewise linear segments.

6.2 Performance Comparison on Real-World Data

We use twenty data sets from the UCI Machine Learning repository [25] to compare the performances of HLSVM, SLSVM, and PSVM against KNN and nonlinear SVM in terms of their accuracy and execution time. The characteristics of the data sets are summarized in Table

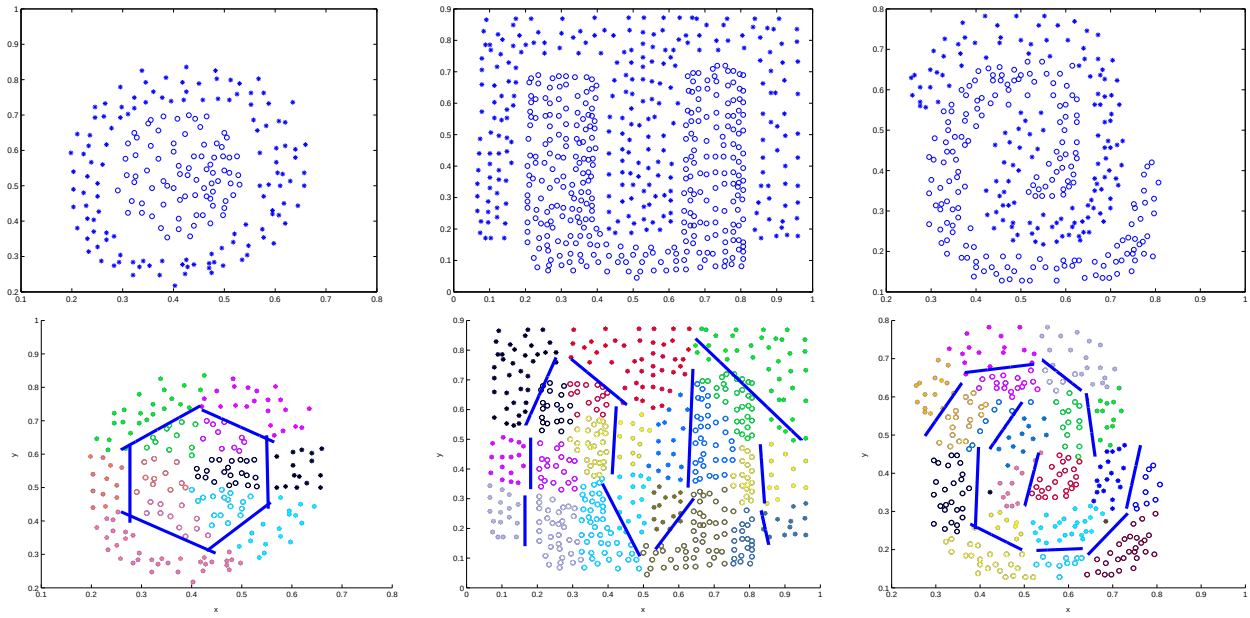

Fig. 4. The diagrams in the top panel show the distribution for three synthetic data sets. Each data set comprises of two classes marked by $\circ$ and $*$, respectively. The diagrams in the bottom panel show the decision boundaries generated by PSVM. Each color represents one of the clusters produced by the MagKmeans algorithm.

1. Some of the data sets such as `Breast`, `Glass`, `Iris`, `KDDCup`, `Physics`, `Yeast`, `Robot`, and `Ecoli` initially contain more than two classes. Since some of the classes have only a few examples, they are not quite suitable for multiclass SVM. Instead, we divide the classes in each of these data sets into two groups and relabel one group as the positive class and the other group as negative class. For some of the larger data sets such as `KDDCup`, `Physics`, and `Robot`, we randomly sample 4500 examples from each class to form the data sets. We repeat the experiment 30 times and report the average accuracy. Each attribute in the data sets has been normalized so that it ranges between 0 and 1.

TABLE 1
Description of the 20 UCI datasets.

DATA	NUMBER OF EXAMPLES	NUMBER OF ATTRIBUTES
BREAST	699	10
GLASS	214	9
IRIS	150	4
KDDCUP	1015062	38
PHYSICS	171017	10
YEAST	1484	8
ROBOT	173841	26
COVTYPE	383467	54
ECOLI	336	7
OZONE	2536	75
ARCENE	200	10000
ARRHYTH	452	279
BALANCE	625	4
CAR	1728	6
OPTDIGIT	3477	64
CONTRACE	1473	9
DERMAT	366	34
SONAR	208	60
GISETTE	6000	5000
KRKOPT	28056	6

6.2.1 *Model Selection*

All the classification algorithms considered in this study require one or more user-specified parameters. For the KNN classifier, this corresponds to the number of nearest neighbors to be considered for each test example. For nonlinear SVM with RBF kernel, the user must specify the kernel width μ as well as the scaling parameter β that controls the tradeoff between classification margin and cost of misclassification. The same two parameters are also needed in SLSVM (except μ is used to define the similarity function σ). For HLSVM, the similarity function is equal to 1 if the training example $\mathbf{x}_i$ belongs to the K nearest neighbor list of the test example $\overline{\mathbf{x}}_s$, and 0 otherwise. Thus, it requires two user-specified parameters: β and K.

In principle, although there are four parameters in PSVM, two of them are automatically determined from characteristics of the data. As noted in Section 4, the scaling parameter R used in MagKmeans clustering is set to the diameter of the data set. Furthermore, the similarity function σ is used for the clustering step only. Thus, changing the kernel width μ in σ will not affect the clustering results significantly. For our experiments, μ is set to the standard deviation of the data. The only two parameters in PSVM that must be specified by the user are the number of clusters (κ) and β.

Throughout our experiments, all the remaining parameters to be set by the user are selected using ten-fold cross validation on the training set. For example, the

TABLE 2
Classification accuracies (%) for SVM, KNN, HLSVM (KNN-SVM), SLSVM , and PSVM on the 20 UCI datasets.

DATA	SVM	KNN	HLSVM	SLSVM	PSVM
BREAST	94.57	95.70	95.42	96.85	96.52
GLASS	64.33	54.30	62.67	66.39	66.91
IRIS	92.75	89.75	74.71	96.42	97.06
KDDCUP	98.14	95.21	98.01	99.71	99.29
PHYSICS	82.98	67.82	83.57	86.42	85.57
YEAST	92.31	93.36	94.62	96.00	95.83
ROBOT	85.35	78.64	85.87	87.23	86.23
COVTYPE	86.21	67.40	73.33	90.22	90.39
ECOLI	93.41	93.30	93.33	94.89	94.44
OZONE	93.35	92.30	91.56	94.80	94.38
ARCENE	63.90	68.94	70.06	72.30	71.93
ARRHYTH	68.78	66.74	68.45	70.22	69.86
BALANCE	89.10	88.70	90.62	92.02	89.94
CAR	79.16	78.24	75.90	81.65	79.22
OPTDIGIT	97.76	96.98	98.18	99.31	98.42
CONTRACE	69.85	69.01	71.79	75.01	72.44
DERMAT	97.65	97.37	98.41	98.68	98.72
SONAR	73.29	69.58	72.57	75.21	74.79
GISETTE	85.91	78.53	85.93	88.14	86.99
KRKOPT	69.04	69.09	68.16	72.15	71.44

parameter K for the KNN classifier is chosen based on the number of nearest neighbors that yields the highest accuracy according to ten-fold cross validation on the training set. The same approach is also used to determine the parameters μ and β (for SLSVM and nonlinear SVM), β and K (for HLSVM), and β and κ (for PSVM). For our experiments, κ usually takes a value between $\sqrt{n}/2$ and $\sqrt{n}$.

6.2.2 Accuracy Comparison

The experimental results reported in this study are obtained by applying five-fold cross validation on the data sets. To make the problem more challenging, we use one-fifth of the data for training and the remaining four-fifths for testing. Each experiment is also repeated ten times and the accuracy reported is obtained by averaging the results over ten trials. Table 2 summarizes the results of our experiments. First, observe that nonlinear SVM outperforms the KNN algorithm in 16 out of the 20 data sets. For the `Covtype` data set, the accuracy for SVM (86.21%) is significantly higher than that for KNN (67.40%). Second, HLSVM does not seem to show significant accuracy improvement over nonlinear SVM. In fact, its accuracy is worse than nonlinear SVM for 10 of the 20 data sets. For `Iris`, the classification accuracy drops significantly from 92.75% (for nonlinear SVM) to 74.71% (for HLSVM). One possible explanation for HLSVM's poor performance is that it uses the same number of nearest neighbors for each test example, which makes it hard to find the optimal parameter for the entire data set. In contrast, the SLSVM algorithm consistently outperforms nonlinear SVM in all twenty data sets. With the exception of `KDDCup`, the observed differences in their classification accuracies are found to be statistically significant according to Student's t-test. This should not come as a surprise because SVM can be considered as a special case of SLSVM by setting the kernel width μ to ∞. Finally, we observe that PSVM, which is an efficient implementation of LSVM, achieves comparable accuracy as SLSVM but still outperforms nonlinear SVM for all the data sets.

6.2.3 Runtime Comparison

The purpose of this experiment is to compare the efficiency of LSVM against PSVM. Recall from Section 3 that the main limitation of LSVM is its high computational cost since a unique LSVM model must be constructed for each test example. PSVM attempts to overcome this limitation by partitioning the training examples into a small number of clusters and building a linear SVM model for each cluster. Figure 5 shows the execution time (in seconds) for training and testing different classification models using the `Physics` data set. To evaluate its performance, we choose the two largest classes of the data set and randomly sample 600 records from each class to form the training set. We then apply LSVM and PSVM separately on the data set, while varying the number of test examples from 600 to 4800. The diagram on the left panel of Figure 5 shows the total time spent by each algorithm for building models and testing. Notice that the time consumed by SLSVM, HLSVM, and PSVM grows linearly with the number of test examples. However, the execution times for both HLSVM and SLSVM are considerably longer than PSVM. We may further decompose the run time for PSVM into two parts: PSVM-Cluster, which is the time needed to perform MagKmeans clustering, and PSVM-LSVM, which is the time needed to build a separate model for each cluster. The diagram on the top right panel of Figure 5 shows that most of the execution time for PSVM is spent on clustering. If the clustering time is excluded, the remaining time needed to build a linear SVM model for each cluster as well as the time to evaluate each test example is considerably shorter than the overall training and testing times for nonlinear SVM, as demonstrated in the bottom right panel of Figure 5.

6.2.4 Summary

The SLSVM algorithm generally outperforms both SVM and KNN but at the expense of longer computational time. PSVM helps to improve its computational efficiency, while achieving comparable accuracy as the SLSVM algorithm.

6.3 Classification of Temporal Data

To demonstrate the effectiveness of LSVM and PSVM on temporal data, we apply them to the Enron email data set, which contains 517,431 email messages from about 150 users. The mailbox for each user is organized into a number of topical and non-topical folders. A topical

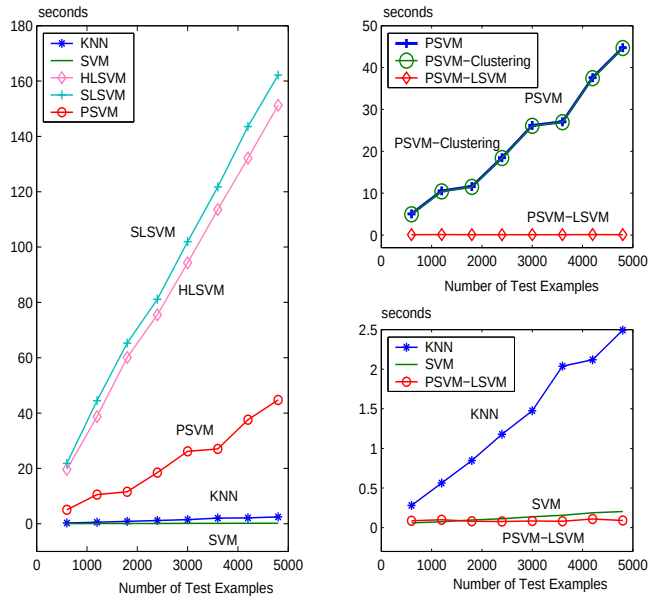

Fig. 5. The left panel shows the runtime comparison among SVM, KNN, HLSVM, SLSVM, and PSVM. The right top panel shows the amount of time spent for MagK-means clustering by PSVM (i.e., PSVM-Clustering) as opposed to the amount of time spent for training LSVMs and applying them to the test examples (i.e., PSVM-LSVM). The right bottom panel compares the time spent by each algorithm to predict the class labels of test examples.

folder contains emails on a particular subject (e.g., golf, ebay, credit, or trading), whereas the non-topical folders include messages that are stored in general-purpose folders such as inbox, contacts, deleted_item, and sent. The prediction task here is to assign each email into its corresponding folder. We use a subset of the Enron data created in [4], which contains emails for 7 users—beck-s, farmer-d, kaminski-v, kitchen-l, lokay-m, sanders-r, and williams-w3. During preprocessing, the non-topical folders and folders that contain only a few messages are removed from each user. We then extract the timestamp for each email and use the information to compute the temporal similarity matrix Σ. Table 3 summarizes the characteristics of the Enron data. The second column indicates the number of classes (folders) assigned to the emails of each user. The third column in the table indicates the actual number of classes selected for our experiments while the last column shows the number of emails contained in these classes. As mentioned in Section 5, we use regular k-means to cluster the email messages. A multi-class LSVM model is then constructed for each cluster using the one-versus-all [29] scheme.

We compared the performance of PSVM against nonlinear SVM, HLSVM, and T-KNN. T-KNN is a variant of the K nearest neighbor classifier where the neighborhood is determined based on temporal locality. Table 4 shows the accuracies of the various algorithms when applied to the emails of various users. In this experiment,

TABLE 3
Data description of Enron testbed from 7 users for email stream classification.

USERS	TOTAL NUMBER OF CLASSES	NUMBER OF SELECTED CLASSES	NUMBER OF EMAILS
BECK-S	101	70	1266
FARMER-D	25	20	1883
KAMINSKI-V	41	30	2093
KITCHEN-L	47	20	2772
LOKAY-M	11	10	1383
SANDERS-R	30	20	679
WILLIAMS-W3	16	16	2769

90% of the data is randomly selected for training and the remaining 10% is used for testing. First, observe that both HLSVM and PSVM achieve significantly higher accuracies than nonlinear SVM and T-KNN for almost all the users. In particular, the accuracy for the user "beck-s" improves from less than 30% to more than 40%. Second, PSVM outperforms HLSVM for all the users. With the exception of "lokay-m" and "williams-w3", the accuracy improvements are statistically significant according to Student's t-test. These results suggest that PSVM is a promising technique for classifying temporal data such as the Enron email data. Figure 6 shows the runtime comparison for both training and testing. The results agree with the observations in Section 6.2, where HLSVM is found to be the most inefficient algorithm. PSVM is slightly more efficient than nonlinear SVM and is comparable to T-KNN. Although the bulk of the computational time for PSVM is spent on clustering the data, there is very little time needed to partition the email messages using regular k-means.

In summary, the results of this experiment support our assertion that PSVM is effective at classifying data with temporal dependencies (compared to nonlinear SVM) and is significantly faster than HLSVM.

6.4 Classification of Spatial Data

For spatial data, we have applied the proposed LSVM and PSVM algorithms to the California housing price

TABLE 4
Classification accuracy for SVM, T-KNN, HLSVM(KNN-SVM) and PSVM for the email stream data from 7 different users.

USERS	T-KNN	SVM	HLSVM	PSVM
BECK-S	21.80	26.34	43.44	47.85
FARMER-D	44.23	38.25	55.71	61.99
KAMINSKI-V	26.73	35.64	42.57	46.29
KITCHEN-L	35.64	34.50	44.90	49.46
LOKAY-M	38.36	36.39	41.10	42.16
SANDERS-R	52.54	50.51	59.60	67.80
WILLIAMS-W3	70.16	85.26	95.28	95.28

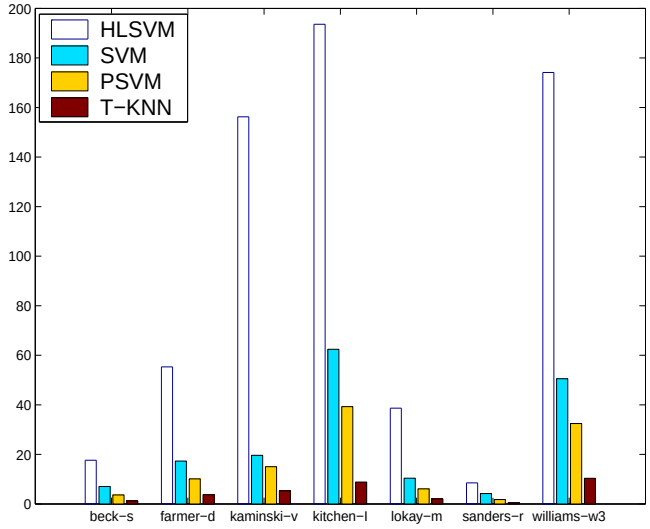

Fig. 6. Runtime comparison among T-KNN, SVM, HLSVM (KNN-SVM) and PSVM on the classification of Enron email data.

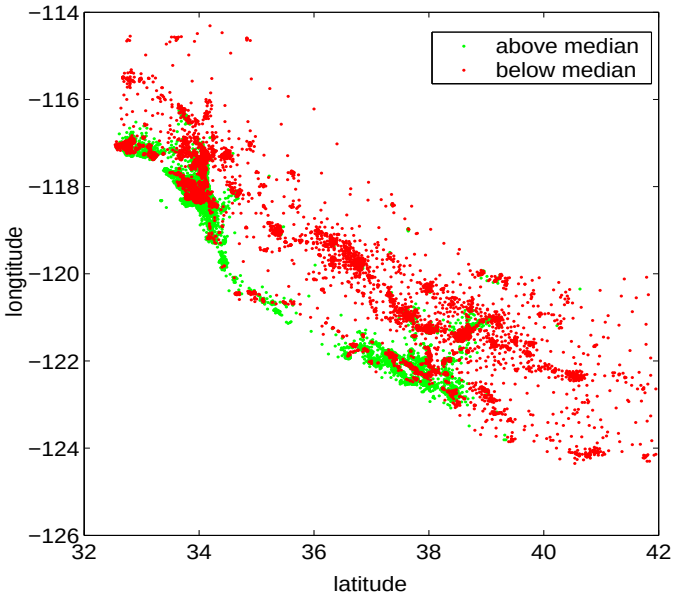

Fig. 7. California housing price data collected from the 1990 census. The data is divided into two classes—house prices above median (●) and house prices below median (●).

data, which was collected during a census conducted in 1990. Each training and test example corresponds to a block unit, which on average, contains about 1425 individuals living in a geographically dense area. The spatial location of each block unit is specified in terms of its latitude and longitude. Each block unit is also characterized by 7 other variables—median house value, median income, median house age, average number of rooms, average number of bedrooms, total population in the block unit, and number of households. We convert the median house value into a binary class by comparing it against the median value for the entire state of California and used it as the target variable for our prediction task. There are 20,640 examples in the data set, whose class distributions are illustrated in Figure 7. Positive examples are shown in green while negative examples are shown in red. The figure clearly shows a spatial relationship between the class attribute and the location of a block unit. This motivates us to consider applying LSVM and PSVM algorithms to the data set using latitude and longitude to define the spatial neighborhood of a test example.

We created three separate data sets from the original data collection—cahouse1, cahouse2, and cahouse3. The number of examples in each data set is 1000, 3000, and 5000, respectively. We apply 10-fold cross validation on each data set to compare the performance of LSVM and PSVM against nonlinear SVM and S-KNN. The latter is a variant of the K-nearest neighbor classifier, where the neighborhood is defined based on spatial proximity. For nonlinear SVM, all the variables, including latitude and longitude, are used to create the feature set of the training and test examples. The parameter K in S-KNN is chosen using ten-fold cross validation on the training set. The same K will also be used for HLSVM.

Table 5 shows the classification accuracies for all the algorithms on the three spatial data sets. Observe that both HLSVM and PSVM achieve significantly higher accuracies than SVM and S-KNN. However, the performance of PSVM is slightly worse than HLSVM. The execution times for training and testing are shown in Figure 8. Again, HLSVM is the most time consuming algorithm because it has to construct a local model for each test example. The time spent by PSVM is considerably less than HLSVM. These results suggest that PSVM is more efficient than HLSVM while maintaining a comparable accuracy.

TABLE 5
Classification accuracies for SVM, S-KNN, HLSVM (KNN-SVM), and PSVM on the California housing price data.

USERS	S-KNN	SVM	HLSVM	PSVM
CAHOUSE-1	73.86	53.24	82.16	80.29
CAHOUSE-2	71.57	51.29	80.07	79.68
CAHOUSE-3	70.07	50.34	78.13	77.78

6.5 Sensitivity Analysis

The number of clusters κ is an important parameter in PSVM. In this experiment, we perform sensitivity analysis on this parameter using the `Robot` data set, though we observe similar behavior in other data sets. We sampled 4500 records each class from the data set and varied κ from 2 to 46. For each value of κ, we performed ten-fold cross validation, using one-tenth of the data for training and the rest for testing. The experiment was repeated ten times. Figure 9 shows the average

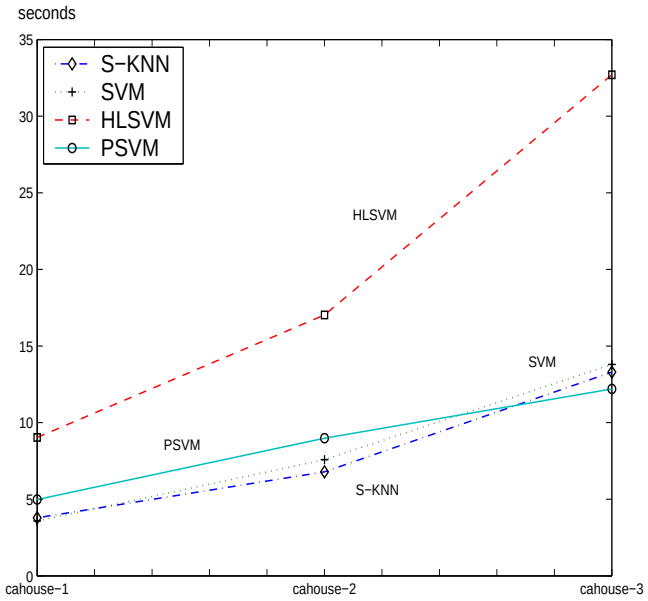

Fig. 8. Runtime comparison among S-KNN, SVM, HLSVM (KNN-SVM), and PSVM on the California housing price data.

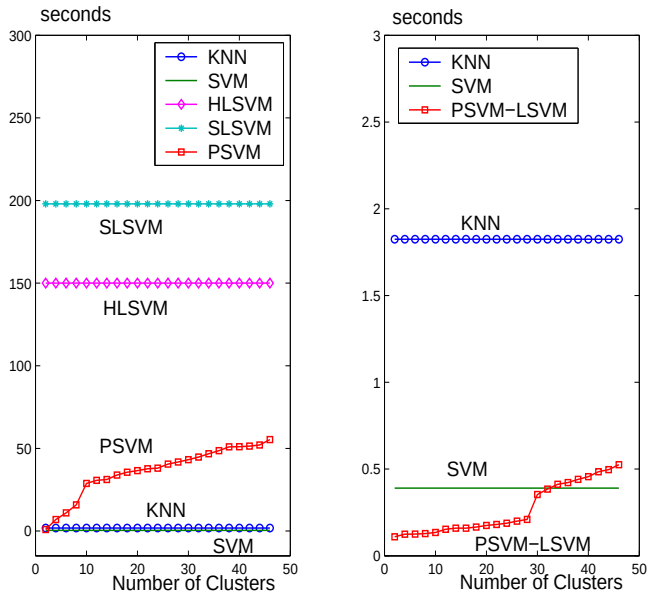

Fig. 10. Runtime comparison among SVM, KNN, HLSVM, SLSVM, and PSVM when varying the number of clusters κ in PSVM.

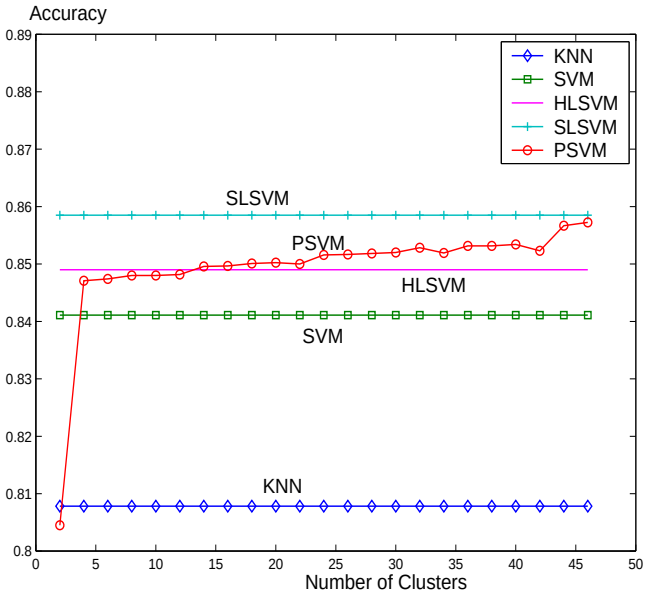

Fig. 9. Effect of varying the number of clusters κ on the performance of PSVM.

accuracies of PSVM for each κ along with those of SVM, KNN, SLSVM and HLSVM. The accuracy of PSVM increases rapidly until κ reaches 5. PSVM outperforms HLSVM after $\kappa = 15$ and becomes comparable to SLSVM when κ is sufficiently large. The execution times for SVM, KNN, HLSVM, SLSVM and PSVM are reported in Figure 10. The results confirmed our earlier observation that PSVM takes considerably less time than SLSVM and HLSVM. Furthermore, if we exclude the clustering time, the execution time for the training and testing parts of PSVM is less than nonlinear SVM when $\kappa \leq 30$, as shown in the right panel of Figure 10.

7 Conclusion

In this paper, we proposed a framework for Localized Support Vector Machine, which incorporates similarity values between training and test examples into SVM learning. We tested the proposed framework on a number of real-world data sets and showed its superior performance over both KNN and nonlinear SVM. Nevertheless, LSVM is expensive to compute since it requires training a separate SVM model for each test example. To address this limitation, we develop a more efficient implementation of the algorithm called Profile SVM. PSVM reduces the computational time by extracting a small number of clusters using a supervised algorithm called MagKmeans. Our analysis showed that PSVM achieves comparable accuracy as LSVM but is much more computationally efficient. We also demonstrated the effectiveness of the proposed algorithms on both temporal and spatial data.

For future work, we will investigate the applicability of the framework to the spatial-temporal domain, where the neighborhood of an example is defined in terms of both space and time [30]. Such extension will be useful for a variety of application domains including climate modeling and dynamic network analysis. Robustness of the framework to noise and skewed class distributions are additional factors that need to be investigated. Finally, the work described in the paper focuses on classification problems only. Extending the proposed framework to other learning tasks such as regression and time series prediction are other possible directions to pursue.

APPENDIX
PROOF OF THE DUAL FORM FOR LOCALIZED SUPPORT VECTOR MACHINE

Let $\overline{\mathcal{D}} = \{\overline{x}_1, \overline{x}_2, ..., \overline{x}_m\}$ be a set of m test examples and $\sigma(\mathbf{x}_i, \overline{\mathbf{x}}_s)$ be the similarity between the training example $\mathbf{x}_i$ and the test example $\overline{\mathbf{x}}_s$. For each $\overline{\mathbf{x}}_s \in \overline{\mathcal{D}}$, we construct its primal local SVM model by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \beta \sum_{i=1}^{n} \sigma(\mathbf{x}_i, \overline{\mathbf{x}}_s)\xi_i \\ \text{s. t} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 - \xi_i, \\ & \xi_i \geq 0,\ i = 1, 2, \ldots, n \end{aligned} \tag{12}$$

We introduce the Lagrangian multiplier α_i for each inequality condition $\mathbf{y}_i((w)^\top \mathbf{x}_i - b) \geq 1 - \xi_i$, and $\mathbf{u}_i$ for $\xi_i \geq 0$. As a result, the Lagrangian for the optimization problem can be written as:

$$\begin{aligned} \mathcal{L} \quad = \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + \beta \sum_{i=1}^{n} \sigma(\mathbf{x}_i, \overline{\mathbf{x}}_s)\xi_i - \\ & \sum_{i=1}^{n} \alpha_i(\mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i - b) - 1 + \xi_i) - \sum_{i=1}^{n} \mathbf{u}_i \xi_i \end{aligned}$$

Then, taking deritives of $\mathcal{L}$ with regard to $\mathbf{w}$, $\mathbf{b}$, ξ_i, we obtain the KKT conditions for the primal problem:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i \mathbf{y}_i \tag{13}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = 0 \Rightarrow -\sum_{i=1}^{n} \alpha_i \mathbf{y}_i = 0 \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \Rightarrow \beta\sigma(\mathbf{x}_i, \overline{\mathbf{x}}_s) - \alpha_i - \mathbf{u}_i = 0 \tag{15}$$

$$\mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i - b) - 1 + \xi_i \geq 0 \tag{16}$$

$$\xi_i \geq 0 \tag{17}$$

$$\alpha_i \geq 0 \tag{18}$$

$$\mathbf{u}_i \geq 0 \tag{19}$$

$$\alpha_i\{\mathbf{y}_i(\mathbf{w}^\top \mathbf{x}_i - b) - 1 + \xi_i\} = 0 \tag{20}$$

$$\mathbf{u}_i \xi_i = 0 \tag{21}$$

From the last two KKT complementarity conditions, we can determine the threshold b. Furthermore, if $\alpha_i < \beta\sigma(\mathbf{x}_i, x)$, then $\xi_i = 0$. So any training point in which $0 < \alpha_i < \beta\sigma(\mathbf{x}_i, x)$ can be used to compute b. By replacing Equations (13)−(15) into the objective function in (13), the optimization problem can be reduced to:

$$\begin{aligned} \mathcal{L} \quad = \quad & \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j + \sum_{i=1}^{n} (\alpha_i + \mathbf{u}_i)\xi_i - \\ & \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j - \sum_{i=1}^{n} \alpha_i(-b\mathbf{y}_i - 1 + \xi_i) - \sum_{i=1}^{n} \mathbf{u}_i \xi_i \\ = \quad & -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j + \sum_{i=1}^{n} \alpha_i \end{aligned} \tag{22}$$

Thus, we obtain the dual form of the LSVM optimization problem, which is given as follows:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \\ \text{s. t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq \beta\sigma(\mathbf{x}_i, \overline{\mathbf{x}}_s),\ i = 1, 2, \ldots, n \end{aligned}$$

ACKNOWLEDGMENTS

The authors would like to thank Dr Ron Bekkerman, Dr Chih-Chung Chang and Dr Chih-Jen Lin for sharing the Enron email data and LIBSVM [9] tools. This work is partially supported by NSF-III Grant No. 0712987.

REFERENCES

[1] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control 25*, pages 821–837., 1964.

[2] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, April 1997.

[3] F. Bach, G. Lanckriet, and M. Jordan. *Fast Kernel Learning using Sequential Minimal Optimization*. UCB/CSD-04-1307. EECS Department, University of California, Berkeley, 2004.

[4] R. Bekkerman, A. McCallum, and G. Huang. Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora. Technical report, Department of CSE, UMASS, 2004.

[5] R. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[6] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

[7] M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. Furey, M. A. Jr, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, January 2000.

[8] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. In *Knowledge Discovery and Data Mining*, page 2, 1998.

[9] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.

[10] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions in Information Theory*, pages 21–27, 1967.

[11] C. Domeniconi, D. Gunopulos, and P. Jing. Large margin nearest neighbor classifiers. *IEEE Transactions on Neural Networks*, 16(4):899–909, 2005.

[12] H. Frohlich and A. Zell. Efficient parameter selection for support vector machines in classification and regression via model-based global optimization. *Proceedings of IEEE International Joint Conference on Neural Networks*, 3:1431–1436, July 2005.

[13] N. Gilardi and S. Bengio. Local machine learning models for spatial data analysis. *Geographic Information and Decision Analysis*, 4(1):11–28, 2000.

[14] S. R. Gunn. Support vector machines for classification and regression. Technical report, University of Southampton, 1998.

[15] D. Hand and V. Vinciotti. Choosing k for two-class nearest neighbour classifiers with unbalanced classes. *Pattern Recognition Letters*, 24(9-10):1555–1562, June 2003.

[16] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence,Volume 18,Issue 6*, pages 607–616, June,1996.

[17] K. Hechenbichler and K. Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. *Discussion Paper 399, SFB 386*, 2006.

[18] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *Technical report, Department of Computer Science and Information Engineering, National Taiwan University*, 2001.
[19] T. Joachims. Transductive inference for text classification using support vector machines PRODIGY. In *International Conference on Machine Learning*, San Francisco, 1999. Morgan Kaufmann.
[20] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7)*, pages 881–892, 2002.
[21] S. Kiritchenko, S. Matwin, and S. Abu-Hakima. Email classification with temporal features. *Proceedings of Intelligent Information Systems, New Trends in Intelligent Information Processing and Web Mining*, pages 523–534, 2004.
[22] K. Koperski, J. Han, and N. Stefanovic. An efficient two-step method for classification of spatial data. *Proceedings of Internatial Symposium on Spatial Data Handling, Vancouver, Canada*, 1998.
[23] K. W. Lau and Q. H. Wu. Local prediction of non-linear time series using support vector regression. *Pattern Recognition*, 41(5):1556–1564, 2008.
[24] F. Melgani and L. Bruzzone. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(8):1778–1790, August 2004.
[25] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/$\sim$mlearn/MLRepository, 1998.
[26] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Conference on Computer Vision and Pattern Recognition*, pages 130–136, San Juan, Puerto Rico, June 1997.
[27] M. Pawlak and M. Ng. On kernel and radial basis function techniques for classification and function recovering. In *International Conference on Pattern Recognition*, pages B:454–456, 1994.
[28] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.
[29] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.
[30] J. Roddick and K. Hornsby, editors. *Temporal, Spatial, and Spatio-Temporal Data Mining*, volume 2007 of *Lecture Notes in Computer Science*. Springer, 2001.
[31] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and sons, 1998.
[32] V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
[33] P. Vincent and Y. Bengio. K-local hyperplane and convex distance nearest neighbor algorithms. *Advances IN Neural Information Processing Systems*, pages 985–992, 2001.
[34] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in Neural Information Processing Systems 18*, pages 1473–1480, 2006.
[35] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: discriminative nearest neighbor for visual object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

Haibin Cheng is a research scientist at Yahoo! Labs. He received his Ph.D. degree in computer science from the Michigan State University and his B.S. degree in computer science from the University of Science and Technology of China. His research interests include data mining, machine learning, and information retrieval. He has published numerous technical papers in data mining conferences and workshops.

Pang-Ning Tan is an Assistant Professor in the Department of Computer Science and Engineering at Michigan State University. He received his Ph.D. degree in computer science from the University of Minnesota. His research interests include data mining, Web intelligence, and machine learning. He is the first author of the textbook *Introduction to Data Mining*, published by Addison Wesley. He is a member of ACM and IEEE.

Rong Jin is an Associate Professor in the Department of Computer Science and Engineering at Michigan State University. He received his Ph.D. degree in computer science from Carnegie Mellon University. His research interests include machine learning, information retrieval, and bioinformatics. He has published numerous technical papers in machine learning journals and conferences. He has also served on the program committees for many international conferences.