A second order cone programming approach for semi-supervised learning

Gao Huang a, Shiji Song a,*, Jatinder N.D. Gupta b, Cheng Wu a

a Department of Automation, Tsinghua University, Beijing 100084, China
b College of Business Administration, The University of Alabama in Huntsville, Huntsville, AL 35899, USA

ARTICLE INFO

Article history:
Received 8 July 2012
Received in revised form 9 May 2013
Accepted 16 June 2013
Available online 26 June 2013

Keywords:
Semi-supervised learning
K-nearest-neighbor
Support vector machine
Second order cone programming

ABSTRACT

Semi-supervised learning (SSL) involves the training of a decision rule from both labeled and unlabeled data. In this paper, we propose a novel SSL algorithm based on the multiple clusters per class assumption. The proposed algorithm consists of two stages. In the first stage, we aim to capture the local cluster structure of the training data by using the k-nearest-neighbor (kNN) algorithm to split the data into a number of disjoint subsets. In the second stage, a maximal margin classifier based on the second order cone programming (SOCP) is introduced to learn an inductive decision function from the obtained subsets globally. For linear classification problems, once the kNN algorithm has been performed, the proposed algorithm trains a classifier using only the first and second order moments of the subsets without considering individual data points. Since the number of subsets is usually much smaller than the number of training points, the proposed algorithm is efficient for handling big data sets with a large amount of unlabeled data. Despite its simplicity, the classification performance of the proposed algorithm is guaranteed by the maximal margin classifier. We demonstrate the efficiency and effectiveness of the proposed algorithm on both synthetic and real-world data sets.

1. Introduction

Over the last decade, semi-supervised learning (SSL) has received considerable attention from the machine learning community [1–3]. Traditionally, supervised learning (SL) requires a fully labeled training data set with all the labels, while unsupervised learning (USL) uses only unlabeled patterns. As the name suggests, semi-supervised learning is halfway between SL and USL. It seeks to learn a decision rule from both labeled and unlabeled data. This is meaningful because in many practical problems, such as text classification and email spam filtering, obtaining training labels is nontrivial, while unlabeled data is easy to collect. Since SSL can utilize both labeled and unlabeled data to achieve higher prediction accuracy than pure SL and USL, it has become a significant focus of new research in machine learning.

The success of SSL relies on certain assumptions on the marginal distribution of the input data [1,3]. One common assumption in SSL is the cluster assumption, which states that if two points are in the same cluster, they are likely to be of the same class. Many SSL algorithms have been proposed based on this assumption, such as semi-supervised mixture models optimized with the EM algorithm [4–6]. Note that the cluster assumption may be equivalently expressed as the low density separation assumption: the decision boundary should lie in a low-density region and not cut through dense unlabeled data. This equivalent expression inspires a series of different SSL algorithms, such as the transductive support vector machine (TSVM) [7] and semi-supervised SVMs (S3VMs) [8–11]. Generally, the family of cluster assumption based SSL algorithms can learn explicit decision functions, resulting in inductive classifiers.

Another commonly adopted semi-supervised assumption that forms the basis of a number of SSL methods is the manifold assumption, which states that high dimensional input data lies on a low-dimensional manifold. The family of graph-based SSL methods [12–16] is based on this assumption. These techniques start by constructing an undirected graph from both the labeled and unlabeled data. The training patterns are represented as the vertices, and the similarities between the samples are set as edge weights. Once the graph is built, a learning algorithm is applied to assign labels to the unlabeled vertices in the graph. Consequently, most of the graph-based methods are inherently transductive, i.e., they aim to perform predictions of the unlabeled patterns supplied in the training stage instead of learning an inductive decision rule for unseen testing data.

Generally, the SSL algorithms based on the cluster assumption, such as semi-supervised mixture model and S3VMs, have the advantage of inductive learning, and typically have a relatively
small number of parameters. However, these algorithms tend to yield poor prediction performance when the cluster assumption is violated [10]. In comparison, the graph-based SSL methods have yielded poor prediction performance when the cluster assumption is small number of parameters. However, these algorithms tend to be less effective when the number of training points is large.

2. Algorithms and methods

We consider the problem of binary classification in a semi-supervised setting. The training set consists of \( l \) labeled patterns \( \{X_i, y_i\} \) and \( u \) unlabeled patterns \( X_u = \{x_i \}_{i=1}^u \), with \( x_i \in \mathbb{R}^d \) and \( y_i = \pm 1 \). For simplicity of notation, we will use \( X \) to denote both the labeled and unlabeled input patterns, i.e., \( X = X_l \cup X_u \). Let \( n = l + u \) be the total number of training patterns in \( X \). The goal of semi-supervised learning is to learn a classifier from both \( (X_l, y_l) \) and \( X_u \).

2.1. Stage 1: obtaining subsets

2.1.1. Partitioning the data

In the first stage of the proposed algorithm, we adopt the kNN algorithm to partition the data into a number of disjoint subsets. Classically, the kNN algorithm is a supervised learning method which does not take advantage of any unlabeled data. In the proposed algorithm, the kNN algorithm is used to explore the cluster structure of input data locally using both the labeled and unlabeled patterns. In this way, the unlabeled data naturally provides useful distribution information for the clusters, which can then be used for training a global inductive classifier in the following stage. The 1-nearest-neighbor (1NN) algorithm, a special case of kNN, classifies each unlabeled pattern according to its nearest labeled data. As one of the oldest methods for pattern classification, 1NN is simple but effective in practical applications.

When the Euclidean distance is used, the 1NN algorithm actually splits the input patterns into \( l \) Voronoi sets, which is defined by

\[
C_k = \{x_i \in X | ||x_i - x_k|| < ||x_i - x_j||, \quad \text{for all } x_i \in X, i \neq k\}, \quad k = 1, \ldots, l,
\]

where \( x_k \) is the \( k \)th target labeled pattern, and \( ||\cdot|| \) denotes the Euclidean norm.

It is clear that the elements in \( C_k \) are those input patterns in \( X \) for which \( x_k \) is the closest labeled vector, and \( X = C_1 \cup \cdots \cup C_l \). Each Voronoi set \( C_k \) contains one labeled pattern and \((n_k - 1) \) unlabeled patterns, i.e.,

\[
C_k = \{x_1^k, \ldots, x^{nk}_k\}, \quad k = 1, \ldots, l,
\]

where \( x_k^1 \) is the \( k \)th labeled pattern, \( \{x_1^k, \ldots, x^{nk}_k\} \) are those unlabeled patterns whose nearest labeled neighbor is \( x_k^1 \), and \( n_k = |C_k| \) is the number of patterns in \( C_k \).

Note that 1NN is not robust to outliers. To increase the robustness against data uncertainties, we can replace the 1NN algorithm in the above partitioning process by kNN with \( k > 1 \). When kNN (\( k > 1 \)) is adopted, we do not simply associate a candidate unlabeled pattern with only its nearest labeled point but consider its \( k \) nearest labeled neighbors. The candidate pattern will be labeled as the majority class among the \( k \) nearest neighbors, and will be assigned to the subset that corresponds to the nearest labeled pattern which has the same class label. That is, for each unlabeled pattern, we first classify it using kNN, then move it into the subset which contains its nearest labeled pattern of the same class. Actually, if the nearest labeled point of an unlabeled candidate pattern belongs to the same class as this candidate, then kNN will yield the same partitioning result for this candidate pattern as that obtained by 1NN. However, if this nearest labeled point happened to be an outlier, it will tend to have different class label as the candidate point. Consequently, the candidate pattern will not be assigned to the same subsets as this nearest neighbor, thus reducing the influence of outliers.

The kNN method is a simple SL algorithm and often leads to competitive results in certain problems, but it cannot benefit from the unlabeled data to improve classification performance. It also lacks the ability to learn a compact decision rule, thus it requires a relatively high computational cost for testing. In the proposed algorithm, these disadvantages are alleviated by introducing a maximal margin classifier described in Subsection 2.2.
For each subset $C_k$, we calculate its arithmetic mean and covariance matrix

$$\mu_k = \frac{1}{n_k} \sum_{l=1}^{n_k} X_l,$$

$$M_k = \frac{1}{n_k} \sum_{l=1}^{n_k} (X_l - \mu_k)(X_l - \mu_k)^T. \tag{3}$$

These are the empirical first and second moments of the subsets. In the second stage, we will manipulate these moments without considering individual patterns. Since the number of unlabeled patterns is usually much larger than the number of labeled patterns, the proposed algorithm significantly reduces the computational cost and, therefore, is capable of dealing with data sets involving large numbers of unlabeled data.

### 2.1.2. Subsets merging

We further perform a merging process on the subsets generated in the above process. Technically, after the partitioning process, the first and second order moments of the subsets can be used directly to train an inductive classifier using the method introduced in the second stage. However, here we propose to merge those subsets that have the same label and are lying “close” to each other. This is because when the number of labeled patterns $(l)$ is large, the partitioning process can generate many subsets, leading to higher computational cost in the following stage. The merging process can improve the efficiency of the proposed algorithm significantly in these scenarios. Moreover, our experimental results show that the classification performance can also be improved by proper subsets merging.

There are several ways to measure the closeness between two subsets to decide whether they should be merged. For example, the Euclidean or Mahalanobis distance between the means of two subsets are two of the simplest measures. The divergence is also a commonly used dissimilarity measure when we assume that the subsets are generated according to certain probability distribution. Specially, by making multivariate Gaussian assumption, it is possible to compute this divergence between sets [24,25]. In our algorithm, we prefer the Mahalanobis metric since it is easy to compute compared to the divergence, and it is more powerful than the Euclidean distance since it can take advantage of the second order moment information.

If the subset $C_k$ is of full rank, i.e., $\det(M_k) \neq 0$, the Mahalanobis distance from subset $C_j$ to subset $C_k$ is computed by

$$D(\mu_j, \mu_k) = (\mu_j - \mu_k)^T M_k^{-1} (\mu_j - \mu_k). \tag{5}$$

If the subset $C_k$ is singular, the inverse of the covariance matrix $M_k$ does not exist. In this case, we first compute the eigenvalues of $M_k$, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N = 0$, and their corresponding normalized eigenvectors, $v_1, \ldots, v_N$. Let $\lambda_j$ be the smallest non-zero eigenvalue of $M_k$, then the distance from subset $C_j$ to subset $C_k$ is computed by

$$D(\mu_j, \mu_k) = (\mu_j - \mu_k)^T V_k \Gamma_k V_k^T (\mu_j - \mu_k). \tag{6}$$

where $V_k = [v_1, \ldots, v_N]$, and $\Gamma_k$ is a diagonal matrix whose jth diagonal element is

$$\gamma_j = \frac{1}{\max(\lambda_j, \lambda_k)}. \tag{7}$$

In practice, we can also replace $\lambda_j$ in the above expression by a small regularization parameter $\epsilon > 0$.

From (5), (6) and (7), we can write the distance function in a unified form:

$$D(\mu_j, \mu_k) = (\mu_j - \mu_k)^T S_k (\mu_j - \mu_k), \tag{8}$$

where

$$S_k = \begin{cases} M_k^{-1}, & \text{det}(M_k) \neq 0, \\ V_k \Gamma_k V_k^T, & \text{det}(M_k) = 0. \end{cases} \tag{9}$$

Note that this distance measure is not symmetric with respect to its input arguments, namely, $D(\mu_j, \mu_k)$ may not equal $D(\mu_k, \mu_j)$.

We compute the distance $D(\mu_j, \mu_k)$ for all $C_j$ and $C_k$ that have the same class label. If one of the distances is less than a given threshold $\rho$, i.e.,

$$\min(D(\mu_j, \mu_k), D(\mu_k, \mu_j)) \leq \rho, \tag{10}$$

then we merge the two subsets into one subset $C_{k'} = \cup(C_j, C_k)$. It is easy to verify that the mean and covariance of $C_{k'}$ are given by

$$\mu_{k'} = \alpha \mu_j + \beta \mu_k, \quad M_{k'} = \alpha M_j + \beta M_k + \alpha \beta (\mu_j - \mu_k)(\mu_j - \mu_k)^T, \tag{11}$$

where $\alpha = n_k/(n_k + n_j)$, $\beta = n_j/(n_j + n_j)$.

Using the Eqs. (11) and (12), we obtain the first and second order information of the new subsets efficiently without considering individual patterns.

### 2.2. Stage 2: classification with subsets

In this section, we introduce a maximal margin method which can learn a decision function from the subsets obtained in stage one. We suppose that $p$ subsets have been obtained in the first stage, and the means and covariances are denoted by $[\mu_k]_{k=1}^p$ and $[M_k]_{k=1}^p$, respectively.

Note that if we treat the centers of all subsets and their corresponding labels as training patterns, the traditional SL method, such as SVM, can be adopted to learn a classifier in a supervised learning framework. However, doing this would ignore the second order moments of the subsets. By using both the first and second order moments of the subsets, the results can be significantly improved.

#### 2.2.1. A maximal margin classifier based on SOCP

Consider the soft margin linear SVM formulation [26]:

$$\min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_{k=1}^p \xi_k \tag{13a}$$

s.t. $y_k (w^T x_k + b) \geq 1 - \xi_k, \quad \forall k = 1, ..., p,$

$$\xi_k \geq 0, \quad \forall k = 1, ..., p. \tag{13b}$$

where $\xi = [\xi_1, \xi_2, \ldots, \xi_p]^T$ are slack variables, and $C$ is a positive penalty coefficient.

The above problem can be efficiently solved by quadratic programming (QP) in its dual form. Here we reformulate (13) as a SOCP by introducing a new positive variable $t$

$$\min_{w, b, t} \quad t + C \sum_{k=1}^p \xi_k \tag{14a}$$

s.t. $y_k (w^T x_k + b) \geq 1 - \xi_k, \quad \forall k = 1, ..., p,$

$\|w\| \leq t. \tag{14b}$

$$\xi_k \geq 0, \quad \forall k = 1, ..., p. \tag{14c}$$

If each input pattern $x_k$ is treated as a random variable whose mean and covariance are $\mu_k$ and $M_k$, respectively, we can replace the constraints (14b) by a set of probabilistic constraints

$$\Pr_{x_k \sim (\mu_k, M_k)} [y_k (w^T x_k + b) \geq 1 - \xi_k] \geq \delta_k, \tag{15}$$

where $k = 1, ..., p$, and $\delta_k$ is a user defined parameter in $[0, 1)$.\n
This constraint requires that the random vector \( x_k \) lies on the correct side of the hyperplane with confidence greater than \( \delta_k \).

It is usually inefficient to solve an optimization problem with probabilistic constraints such as (15). Fortunately, the constraints (15) can be transformed into deterministic constraints [27,28] under certain conditions.

A. Normal distribution case: We first assume that \( x_k \) is drawn from a normal distribution with mean \( \mu_k \) and covariance \( M_k \). It is easy to verify that \( z_k = y_k(w^T x_k + b) \) is a normal random variable with mean \( \bar{M} \) and variance \( \sigma_k^2 = w^T M_k w \). Therefore, (15) can be expressed as

\[
\Pr_{z_k \sim \mathcal{N}(\bar{M}, \sigma_k^2)} ( \frac{z_k - \bar{M}}{\sigma_k} ) \geq \delta_k.
\]

(16)

or equivalently as

\[
\Pr_{z_k \sim \mathcal{N}(\bar{M}, \sigma_k^2)} ( \frac{z_k - \bar{M}}{\sigma_k} ) \geq \delta_k.
\]

(17)

This inequality can be further written as

\[
1 - \Phi((1 - \delta_k)/\sigma_k) \geq \delta_k,
\]

(18)

where \( \Phi(\cdot) \) is the cumulative distribution function (CDF) of a standard normal distribution.

Thus, the probabilistic constraints (15) are equivalent to

\[
y_k(w^T \mu_k + b) \geq 1 - \delta_k + \gamma \sqrt{w^T M_k w}.
\]

(19)

Note that for \( \delta_k \geq 0.5 \), the above inequality is a SOC constraint since \( \gamma \sqrt{w^T M_k w} \) can be expressed by \( \| M_k \|^2 / \| w \| \). When \( \delta_k < 0.5 \), the optimization problem becomes non-convex and will not be considered in this paper.

B. General case: If no information is available about the distribution of \( x_k \) except its mean and covariance, we can consider the following worse case condition instead [27,28]

\[
\inf_{x_k \sim \mathcal{N}(\mu_k, \rho_k)} \Pr(y_k(w^T x_k + b) \geq 1 - \delta_k) \geq \delta_k.
\]

(20)

or equivalently expressed as

\[
\sup_{x_k \sim \mathcal{N}(\mu_k, \rho_k)} \Pr(y_k(w^T x_k + b) \geq 1 - \delta_k) \leq 1 - \delta_k.
\]

(21)

We now transform the above expression into a tractable constraint using the multivariate Chebyshev inequality [29]. For a given closed convex set \( S \) we have

\[
sup_{x \in S} \Pr(x \in S) = \frac{1}{1 + d^2},
\]

(22)

where \( d^2 = \inf_{\mu \in S} \| M \|^{-1} (\mu - \mu) \).

Thus we have

\[
\sup_{x_k \sim \mathcal{N}(\mu_k, \rho_k)} \Pr(y_k(w^T x_k + b) \geq 1 - \delta_k) = \frac{1}{1 + d^2},
\]

(23)

where

\[
d^2 = \inf_{x_k \sim \mathcal{N}(\mu_k, \rho_k)} \| x_k - \mu_k \|^2 M_k^{-1} (x_k - \mu_k).
\]

(24)

From (21), (23) and (24) we obtain the following SOCP constraints for the general distribution case

\[
y_k(w^T \mu_k + b) \geq 1 - \delta_k + \gamma \sqrt{w^T M_k w}.
\]

(25)

2.2.2. Classification with subsets

We write (19) and (25) in a unified expression

\[
y_k(w^T \mu_k + b) \geq 1 - \delta_k + \gamma \sqrt{w^T M_k w}.
\]

(26)

where

\[
y_k = \begin{cases} \{ \hat{y}_k^N(\delta_k) = \Phi^{-1}(\delta_k), & \text{Normal case;} \\ \{ \hat{y}_k^C(\delta_k) = \sqrt{\delta_k/(1-\delta_k)}, & \text{General case.} \end{cases}
\]

(27)

It is easy to verify that \( \hat{y}_k^N(\delta_k) < \hat{y}_k^C(\delta_k) \) for all \( \delta_k \in (0.5, 1) \). This is due to the fact that in the general case we have no information about the distributional form of \( x_k \) except its mean and covariance. Thus, we can only obtain an upper bound of \( \gamma \) for all possible distributions with the same first and second order moments.

For subsets \( C_k \) obtained in the previous stage, we notice that \( \hat{y}_k^N(\delta_k) \) (for the normal distribution case) might be too optimistic, while \( \hat{y}_k^C(\delta_k) \) (for the general distribution case) might be too conservative. Therefore, we propose to set \( \gamma_k \) as

\[
y_k = \begin{cases} \min \{ \hat{y}_k^N(\delta_k), & \text{Normal case;} \\ \min \{ \hat{y}_k^C(\delta_k), & \text{General case.} \end{cases}
\]

(28)

where \( n_k \) is the number of training patterns in subset \( C_k \), \( n_k^{p_{pos}} \) and \( n_k^{p_{neg}} \) are, respectively, the numbers of positive and negative training patterns in the whole data set, and \( \gamma \) is a problem-dependent hyperparameter.

The underlying intuition of the above setting is quite simple. It is clear that \( \hat{y}_k^N \) and \( \hat{y}_k^C \) are both increasing functions of \( \delta_k \) on \([0.5, 1]\), which suggests that if we want \( x_k \) to be correctly classified with higher probability, i.e., higher value of \( \delta_k \), then a larger \( \gamma_k \) should be used. Therefore, by using (28), a subset with more training patterns is designed to be classified correctly with higher probability; while a subset with less training patterns will have a smaller effect on the decision rule. The hyperparameter \( \gamma \) is used to control the confidence level for each subset to be placed on the correct side.

With the above formulas, the maximal margin formulation is given by

\[
\min_{w, b, t} \quad t + C \sum_{k=1}^{p} \xi_k
\]

(29a)

s.t. \[
y_k(w^T \mu_k + b) \geq 1 - \xi_k + \gamma_k \sqrt{w^T M_k w},
\]

\[
\forall k = 1, \ldots, p,
\]

(29b)

\[
\| w \| \leq \delta_t,
\]

(29c)

\[
\xi_k \geq 0, \quad \forall k = 1, \ldots, p.
\]

(29d)

This formulation is a SOC which can be solved efficiently by optimization methods such as interior point method (IPM). For an optimal solution to problem (29) denoted \( w^* \) and \( b^* \), the decision function is given by

\[
f(x; w^*, b^*) = \text{Sign}(w^T x + b^*).
\]

(30)

2.3. The complete linear SSL algorithm

Given the aforementioned formulas, the proposed linear SSL algorithm is summarized in Algorithm 1.

Algorithm 1. The proposed linear SSL algorithm.

Input:
The labeled patterns, \( X_l, y_l \);
The unlabeled patterns, \( X_u \);

Output:
The parameters of the decision function \( f(x; w^*, b^*) \); Stage 1
1-a) Generate initial subsets
Split the input data into \( l \) subsets using the kNN algorithm as introduced in Subsection 2.1.1;

1-b) Subset Merging
while There exists two subsets \( C_k \) and \( C_i \) such that \( \min_{\mu_k, \mu_i} D(\mu_k, \mu_i) \leq \rho \) do
Merge the two subsets into one, \( C_k' = \bigcup (C_k, C_i) \);
end while

Stage 2
Train the maximal margin classifier (29) using the first and second order moments of the subsets generated in Stage 1;
return \( w^* \) and \( b^* \);

2.4. An illustrative example of the proposed algorithm

We now illustrate the proposed method on a synthetic binary classification problem. As shown in Fig. 1, each of the two classes is separately drawn from two independent Gaussian distributions, and each class contains two clusters. For each class (shown by squares and crosses, respectively), we generate 4 labeled points (bold) and 1000 unlabeled points. Fig. 1 also shows the supervised decision boundary given by a standard SVM, and the decision boundary given by a semi-supervised EM algorithm based on a single cluster per class assumption. The supervised SVM ignores any distribution of the unlabeled data, and gives a classification error rate of 5.67%; while the semi-supervised EM algorithm gives an error rate of 8.08% under the wrong model assumption.

In the first stage of our algorithm, the 1NN algorithm is adopted to split the training data into \( l \) subsets. In this example, 8 subsets are obtained, and 1NN yields an error rate of 3.86% (as shown in Fig. 2). We then calculate the mean and covariance of each subset, which carry the first and second moments information of the corresponding subset. Fig. 3 visualizes the means and covariances by ellipsoids, centered at the subset means. Then we merge the subsets that have the same class label and are lying ‘close’ to each other into larger subsets, whose mean and covariance are shown as ellipsoids in Fig. 4.

In the second stage, a decision rule is learned from the subsets obtained in the above stage. This is achieved by training the maximal margin classifier (29) using the means and covariances of the subsets. Fig. 4 shows the decision boundary given by the proposed algorithm, which yields an error rate of 1.55%.

3. Kernelized formulation

In this section, we introduce the kernelized version of our semi-supervised algorithm for linearly nonseparable problems.

3.1. Stage 1

Denote \( \varphi : X \to \mathcal{F} \) as mapping function that maps the input patterns into a high dimensional feature space \( \mathcal{F} \). Usually, the mapping function is required to satisfy Mercer’s condition such
that
\[ \phi(x_i)^T \phi(x_j) = k(x_i, x_j), \]  
(31)

where \( k(\cdot, \cdot) \) is a kernel function. Commonly used kernel functions include

- Gaussian kernel
\[ k^{\text{G}}(x_i, x_j) = \exp\left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right), \sigma \in \mathbb{R}. \]  
(32)

- Polynomial kernel
\[ k^{\text{P}}(x_i, x_j) = (1 + x_i^T x_j)^p, \ p \in \mathbb{N}^+. \]  
(33)

Though the nonlinear mapping \( \phi \) is usually unknown, the Euclidean distance between input patterns in the feature space can be computed with the following kernel trick
\[ ||\phi(x_i) - \phi(x_j)||^2 = (\phi(x_i) - \phi(x_j))^T (\phi(x_i) - \phi(x_j)) = \phi(x_i)^T \phi(x_i) - 2\phi(x_i)^T \phi(x_j) + \phi(x_j)^T \phi(x_j) = k(x_i, x_i) - 2k(x_i, x_j) + k(x_j, x_j). \]  
(34)

With the distance computed from (34), the kNN algorithm can be applied in the feature space. Analogous to the Voronoi sets in the input space, the Voronoi sets in feature space can be defined as follows:
\[ \tilde{C}_k = \{ \phi(x_i) \mid ||\phi(x_i) - \phi(x_j)|| < ||\phi(x_i) - \phi(x_k)||, x_i \in \mathcal{X}, \text{ for all } x_i \in \mathcal{X}_i, i \neq k \}. \]  
(35)

Thus we can split the input data into \( l \) subsets similar to that introduced in Subsection 2.1.1. However, when the mapping function \( \phi \) is usually unknown, it is impossible to compute the mean and covariance of \( \tilde{C}_k \) explicitly. In the following stage, we will use the kernel trick to handle this difficulty.

3.2. Stage 2

Linear classification can be performed in the feature space defined by the mapping \( \phi \). From (14), the formulation of a linear classifier in the feature space is
\[ \min_{\hat{w}, b, t} t + C \sum_{k=1}^{p} \xi_k \]  
(36a)
\[ \text{s.t. } y_k (\hat{w}^T \phi(x_k) + b) \geq 1 - \xi_k, \ \forall k = 1, \ldots, p, \]  
(36b)
\[ \|\hat{w}\| \leq t, \]  
(36c)
\[ \xi_k \geq 0, \ \forall k = 1, \ldots, p. \]  
(36d)

where \( \hat{w} \) and \( b \) are the parameters of the classification hyperplane in the feature space.

By repeating the derivation of Subsection 2.2.1 in the feature space, it is straightforward to obtain the maximal margin classifier in the feature space
\[ \min_{\hat{w}, b, t} t + C \sum_{k=1}^{p} \xi_k \]  
(37a)
\[ \text{s.t. } y_k (\hat{w}^T \hat{\mu}_k + \hat{b}) \geq 1 - \xi_k + y_k \sqrt{\hat{w}^T \hat{M}_k \hat{w}}, \]  
(37b)
\[ \|\hat{w}\| \leq t, \]  
(37c)
\[ \xi_k \geq 0, \ \forall k = 1, \ldots, p. \]  
(37d)

where \( \hat{\mu}_k \) and \( \hat{M}_k \) are, respectively, the mean and covariance matrix of the subset \( \tilde{C}_k \).

For the sake of notational convenience, we denote
\[ \phi_j = \phi(x_j), \ j = 1, \ldots, n, \]  
\[ \phi_k = \phi(x_k), \ k = 1, \ldots, p, \]
\[ \tau = 1, \ldots, n_k, \]

where \( x_j \in \mathcal{X} \) denotes the \( j \)th pattern in the whole training set, and \( x_k \in \mathcal{X} \) denotes the \( k \)th pattern in the \( k \)th subset.

It follows that
\[ \hat{\mu}_k = \frac{1}{n_k} \sum_{\tau=1}^{n_k} \phi_{k,\tau}, \]  
(38)
\[ \hat{M}_k = \frac{1}{n_k} \sum_{\tau=1}^{n_k} (\phi_{k,\tau} - \hat{\mu}_k)(\phi_{k,\tau} - \hat{\mu}_k)^T. \]  
(39)

Define the kernel matrix \( K \in \mathbb{R}^{n \times n} \) as
\[ [K]_{ij} = \phi_i^T \phi_j = k(x_i, x_j), \ i, j = 1, \ldots, n. \]  
(40)

Note that \( \hat{w} \) is in the span of the training data points in the feature space, such that
\[ \hat{w} = \sum_{j=1}^{n} \theta_j \phi_j, \]  
(41)
where \( \theta = [\theta_1, \ldots, \theta_n]^T \in \mathbb{R}^n \) is a coefficient vector.

Thus, we have
\[ \|\hat{w}\|^2 = \hat{w}^T \hat{w} = \left( \sum_{j=1}^{n} \theta_j \phi_j \right)^T \left( \sum_{j=1}^{n} \theta_j \phi_j \right) = \theta^T K \theta, \]  
(42)
\[ \hat{w}^T \hat{\mu}_k = \left( \sum_{j=1}^{n} \theta_j \phi_j \right)^T \left( \frac{1}{n_k} \sum_{\tau=1}^{n_k} \phi_{k,\tau} \right) = \frac{1}{n_k} \theta^T K \theta_k, \]  
(43)
where \( \theta_k = k(x_i, \tilde{C}_k) \) and \( \theta_k \) is a \( n \)-dimensional vector with
\[ [1_k] = \begin{cases} 1, & j \in [i] \mid \phi(x_j) \in \tilde{C}_k; \\ 0, & \text{otherwise}. \end{cases} \]  
(44)

Similarly, we have
\[ \hat{w}^T \hat{M}_k \hat{w} = \left( \frac{1}{n_k} \sum_{\tau=1}^{n_k} (\phi_{k,\tau} - \hat{\mu}_k)(\phi_{k,\tau} - \hat{\mu}_k)^T \right) \hat{w} = \frac{1}{n_k} \sum_{\tau=1}^{n_k} \hat{w}^T (\phi_{k,\tau} - \hat{\mu}_k)(\phi_{k,\tau} - \hat{\mu}_k)^T = \frac{1}{n_k} \sum_{\tau=1}^{n_k} \left( \theta_k^T K \theta_k - \frac{1}{n_k} \theta_k^T K \theta_k \right)^2 \]  
(45)
where \( \theta_k \) is a \( n \)-dimensional vector with the element that corresponds to \( x_i \) equal to 1, and all other elements 0.

We further denote \( \theta_k^T K \theta_k \) and \( \theta_k^T K \theta_k - \frac{1}{n_k} \theta_k^T K \theta_k \)
\[ \bar{1}_k = \frac{1}{n_k} \sum_{\tau=1}^{n_k} \left( \theta_k^T K \theta_k - \frac{1}{n_k} \theta_k^T K \theta_k \right) \]  
(46)

The expression (45) can be simplified as
\[ \hat{w}^T \hat{M}_k \hat{w} = \theta^T \bar{1}_k \theta. \]  
(47)

With the expressions (42), (43), and (47), the kernelized formulation (37) becomes
\[ \min_{\theta, b, t} t + C \sum_{k=1}^{p} \xi_k \]  
(48a)
\[ \text{s.t. } y_k (\theta^T \bar{1}_k + \hat{b}) \geq 1 - \xi_k + y_k \sqrt{\theta^T \bar{M}_k \theta}, \]  
(48b)
\[ \|\theta\| \leq t, \]  
(48c)
\[ \xi_k \geq 0, \ \forall k = 1, \ldots, p. \]  
(48d)
\begin{equation}
\forall k = 1, \ldots, p, \tag{48b}
\end{equation}
\begin{equation}
\sqrt{\theta^T K \theta} \leq t, \tag{48c}
\end{equation}
\begin{equation}
\xi_k \geq 0, \quad \forall k = 1, \ldots, p. \tag{48d}
\end{equation}

Note that the above optimization problem is a SOCP since \((48b)\) and \((48c)\) are all SOCP constraints.

We mention that for an optimal solution \(\theta^*\) and \(b^*\) to problem \((48)\), the classification function is given by

\begin{equation}
f(x; \theta^*, b^*) = \text{Sign} \left( \sum_{k=1}^{p} \alpha_k^* \left( \frac{1}{n_k} \sum_{i=1}^{n_k} k(x_i, x) \right) + b^* \right). \tag{49}
\end{equation}

3.3. The proposed kernelized SSL algorithm

The proposed kernelized SSL algorithm is summarized in Algorithm 2.

**Algorithm 2.** The kernelized SSL algorithm.

**Input:**
- The labeled patterns, \(X_l, y_l\);
- The unlabeled patterns, \(X_u\);

**Output:**
- The parameters of the decision function \((49)\), \(\theta^*\) and \(b^*\);

**Stage 1**
- Split the input data into \(l\) subsets according to \((34)\) and \((35)\);

**Stage 2**
- Train the maximal margin classifier \((48)\) using the subsets generated in stage 1;

**return** \(\theta^*\) and \(b^*\);

4. Simulation examples

In this section, we empirically test the proposed algorithm on a number of synthetic and real world data sets. The SOCP is solved by SeDuMi Matlab toolbox, version 1.3 \([30,31]\).

4.1. Model selection

Due to the lack of labeled data, model selection becomes challenging in semi-supervised learning. There are several methods that have been widely used for hyperparameter selection for SSL algorithms. One of the simplest ways to find the optimal model is by minimizing the test error directly. This requires that the labels of the testing data are known at the training stage, which is not possible in practice. However, the results obtained by this method can be used for evaluating the potential of a given algorithm. We will use this model selection method on the transductive learning benchmarks in Subsection 4.2.

To test the real world performance of a SSL algorithm, only the small set of labeled points are used to be used for model selection. Under this case, the classification error rate becomes an unreliable performance criterion since a wide range of hyperparameter values may yield the same error rate. In our experiment, we adopt the empirical error criterion introduced in \([32]\) for model selection, which is defined by

\begin{equation}
E = \sum_{i=1}^{N} |t_i - p_i|, \tag{50}
\end{equation}

where

\begin{equation}
t_i = (y_i + 1)/2, \tag{51}
\end{equation}

\begin{equation}
p_i = 1/(1 + \exp(-A \cdot f(x_i) + B)), \tag{52}
\end{equation}

and the parameters \(A\) and \(B\) are fitted by minimizing the cross-entropy error. In our experiment, we fix the values to \(A = 1\) and \(B = 0\).

To perform model selection with the empirical error criterion, we adopt the nested \(k\)-fold cross validation (CV) technique as introduced in \([33]\), which consists of an outer CV and an inner (nested) CV. In the outer CV, we randomly divide the data set into \(k\) subsets of roughly equal sizes. Here we make sure that each subset contains approximately the same proportion of negative and positive patterns as the overall labeled data sets. Then the learning algorithm is trained using \(k-1\) of these subsets, and tested on the hold out subset. This procedure is repeated \(k\) times, each with a different subset used for testing. The model selection is performed with an inner \(k\)-folds CV using only the labeled training data. The average performance obtained from the outer \(k\)-fold CV is used as the performance measure.

Other than the above two model selection methods, we also propose a particular method to find the optimal hyperparameters for our algorithm. We will discuss this technique in detail in Subsection 4.5.

4.2. Transductive learning on benchmarks

We first test our algorithm on several benchmarks that have been elaborately designed for evaluating SSL algorithms (chapter 21 of \([1]\)). The benchmarks include three artificial data sets, \(g241c, g241d,\) and \(Digit1\), and four real data sets, \(USPS, COIL, BCI,\) and \(Text\). Table 1 lists the characteristics of these data sets. We use the same experimental setup as in \([1]\): 12 groups of training and testing sets are generated for each benchmark. Each group contains 10 labeled training points, and all the remaining data are used as unlabeled training data which also serve as the testing data. For the sake of comparison, we also include the results obtained by several state-of-the-art SSL algorithms as presented in \([1]\), including transductive SVM (TSVM) \([16]\), cluster-kernel (C-K) \([34]\), low density separation (LDS) \([35]\), data dependent regularization (DDR) \([36]\), Laplacian regularized least squares (LapRLS) \([37]\), and semi-supervised EM (Semi-EM) \([4]\). The results given by two classical supervised learning methods, i.e., 1NN and SVM, are also reported.

On all these benchmarks, the linear form of the proposed method is used. In this experiment, the number of nearest neighbors is fixed to \(k = 1\), and the threshold for subsets merging is fixed to \(\rho = N\) (\(N\) is the dimension of the input data). The hyperparameters \(C\) and \(\gamma\) are, respectively, selected from the exponential sequence \([10^{-1}, \ldots, 10^5]\) and \([2^{-5}, 2^{-4}, \ldots, 2^{10}]\) by minimizing the test error directly, or by 5-fold CV on the labeled training data (last column in Tables 2 and 3).

The classification error rate and the AUC score (area under the receiver operating characteristic (ROC) curve) are used as performance measures. Note that for skewed data sets (the numbers of patterns in different classes are imbalanced), the AUC score is considered to be a more appropriate measure. Tables 2 and 3, respectively, show the prediction error rates and AUC scores.

<table>
<thead>
<tr>
<th>Data set</th>
<th># Points</th>
<th># Dimensions</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>g241c</td>
<td>1500</td>
<td>241</td>
<td>2</td>
</tr>
<tr>
<td>g241d</td>
<td>1500</td>
<td>241</td>
<td>2</td>
</tr>
<tr>
<td>Digit1</td>
<td>1500</td>
<td>241</td>
<td>2</td>
</tr>
<tr>
<td>USPS</td>
<td>1500</td>
<td>241</td>
<td>2</td>
</tr>
<tr>
<td>COIL</td>
<td>1500</td>
<td>241</td>
<td>6</td>
</tr>
<tr>
<td>BCI</td>
<td>400</td>
<td>117</td>
<td>2</td>
</tr>
<tr>
<td>Text</td>
<td>11,960</td>
<td>1500</td>
<td>2</td>
</tr>
</tbody>
</table>

G. Huang et al. / Pattern Recognition 46 (2013) 3548–3558

Table 1
Summary of the benchmarks.
is no apparent difference between the three settings of the assumption. The data sets are considered to be cluster-like data sets, on which the proposed algorithm achieves competitive performance with the other state-of-the-art SSL methods. Among all the SSL algorithms shown in this table, the LapRLS is based on the manifold assumption while all the other algorithms are based on the cluster assumption. The data sets g241c, g241d and Text are usually considered to be cluster-like data sets, on which the proposed algorithm achieves highly competitive results with the other cluster-based SSL algorithms. On the manifold-like data sets, USPS, COIL and BCI, the results obtained by the proposed algorithm are comparable to those obtained by the manifold based LapRLS algorithm. From these experimental results, we can observe that in comparison to various SSL algorithms, the proposed algorithm yields satisfactory performance overall.

We also conduct an experiment to study the impact of the hyperparameter \( k \) on classification performance. Three values of \( k \), i.e., \( k = 1,3 \) and 5, are tested, and the other hyperparameters are selected as introduced above. The test AUC scores are summarized in Table 4. From the experiment results, we can observe that there is no apparent difference between the three settings of \( k \). The 1NN algorithm seems to give slightly better performance than the other two algorithms. This may due to the fact that the labeled data in the benchmarks contains few outliers. One can expect that under the case of high data noise, a value of \( k \) larger than 1 may give more stable performance.

4.3. Inductive learning on UCI data sets

As discussed in the Introduction, inductive learning is of great importance in many situations. To test the inductive performance of the proposed algorithm, we run our algorithm on five real world data sets from the UCI database [38] and StatLib repository [39]. The characteristics of these data sets are summarized in Table 5.

We perform nested 5-fold cross validation as discussed in Subsection 4.1 on these data sets. Note that only the hyperparameters \( C \) and \( \gamma \) are cross validated, while \( k \) and \( \rho \) are fixed to 1 and \( N \), respectively. For the data sets Ionosphere, the kernelized form of our algorithm is used, and the hyperparameter \( \sigma \) in Gaussian kernel is set to 2. For all the other data sets, we adopt the linear classifier. For the training set, we remove most of the labels, leaving only few (10, 20 and 50) labeled patterns. The nested 5-fold CV is performed 4 times, and we report the averaged AUC scores along with the standard deviations over the 20 trials.

Table 2: Testing error rate (%) on the benchmarks.

<table>
<thead>
<tr>
<th></th>
<th>INN</th>
<th>SVM</th>
<th>TSVM</th>
<th>C-K</th>
<th>LDS</th>
<th>DDR</th>
<th>SS-EM</th>
<th>LapRLS</th>
<th>Proposed</th>
<th>Proposed-5cv</th>
</tr>
</thead>
<tbody>
<tr>
<td>g241c</td>
<td>44.05</td>
<td>47.32</td>
<td>24.71</td>
<td>48.28</td>
<td>28.85</td>
<td>41.25</td>
<td>44.05</td>
<td>43.95</td>
<td>35.08</td>
<td>37.99</td>
</tr>
<tr>
<td>g241d</td>
<td>43.22</td>
<td>46.66</td>
<td>50.08</td>
<td>42.05</td>
<td>50.63</td>
<td>45.89</td>
<td>43.22</td>
<td>45.68</td>
<td>37.32</td>
<td>41.16</td>
</tr>
<tr>
<td>Digit1</td>
<td>23.47</td>
<td>30.60</td>
<td>17.77</td>
<td>18.73</td>
<td>15.63</td>
<td>12.49</td>
<td>22.80</td>
<td>5.44</td>
<td>16.33</td>
<td>19.08</td>
</tr>
<tr>
<td>USPS</td>
<td>19.82</td>
<td>20.03</td>
<td>25.20</td>
<td>19.41</td>
<td>17.57</td>
<td>17.96</td>
<td>19.82</td>
<td>18.99</td>
<td>18.70</td>
<td>20.85</td>
</tr>
<tr>
<td>COIL</td>
<td>65.91</td>
<td>68.36</td>
<td>67.50</td>
<td>67.32</td>
<td>61.90</td>
<td>63.65</td>
<td>61.23</td>
<td>54.54</td>
<td>55.60</td>
<td>62.57</td>
</tr>
<tr>
<td>BCI</td>
<td>46.74</td>
<td>49.85</td>
<td>49.15</td>
<td>48.31</td>
<td>49.27</td>
<td>50.21</td>
<td>48.68</td>
<td>48.97</td>
<td>45.56</td>
<td>48.26</td>
</tr>
<tr>
<td>Text</td>
<td>39.44</td>
<td>45.37</td>
<td>31.21</td>
<td>42.72</td>
<td>27.15</td>
<td>-</td>
<td>32.12</td>
<td>33.68</td>
<td>32.24</td>
<td>34.21</td>
</tr>
</tbody>
</table>

Table 3: AUC (area under curve) (%) on the benchmarks.

<table>
<thead>
<tr>
<th></th>
<th>INN</th>
<th>SVM</th>
<th>TSVM</th>
<th>C-K</th>
<th>LDS</th>
<th>DDR</th>
<th>SS-EM</th>
<th>LapRLS</th>
<th>Proposed</th>
<th>Proposed-5cv</th>
</tr>
</thead>
<tbody>
<tr>
<td>g241c</td>
<td>64.68</td>
<td>82.41</td>
<td>61.63</td>
<td>77.35</td>
<td>63.43</td>
<td>55.98</td>
<td>59.23</td>
<td>74.91</td>
<td>68.36</td>
<td></td>
</tr>
<tr>
<td>g241d</td>
<td>63.04</td>
<td>50.65</td>
<td>77.68</td>
<td>48.70</td>
<td>65.92</td>
<td>57.03</td>
<td>57.07</td>
<td>69.72</td>
<td>65.73</td>
<td></td>
</tr>
<tr>
<td>Digit1</td>
<td>88.38</td>
<td>86.98</td>
<td>89.49</td>
<td>90.10</td>
<td>96.22</td>
<td>78.85</td>
<td>99.50</td>
<td>91.54</td>
<td>89.62</td>
<td></td>
</tr>
<tr>
<td>USPS</td>
<td>75.56</td>
<td>68.21</td>
<td>74.28</td>
<td>75.88</td>
<td>84.91</td>
<td>64.76</td>
<td>85.70</td>
<td>78.13</td>
<td>75.77</td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>51.59</td>
<td>50.92</td>
<td>51.77</td>
<td>49.75</td>
<td>50.31</td>
<td>51.34</td>
<td>51.69</td>
<td>54.80</td>
<td>51.99</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>67.97</td>
<td>73.42</td>
<td>73.09</td>
<td>80.68</td>
<td>-</td>
<td>72.23</td>
<td>76.55</td>
<td>74.12</td>
<td>73.24</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: AUC with respect to different choice of \( k \) in kNN.

<table>
<thead>
<tr>
<th></th>
<th>( K=1 )</th>
<th>( K=3 )</th>
<th>( K=5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>g241c</td>
<td>74.91 ± 5.35</td>
<td>74.62 ± 7.02</td>
<td>73.65 ± 8.03</td>
</tr>
<tr>
<td>g241d</td>
<td>69.72 ± 8.42</td>
<td>69.02 ± 8.01</td>
<td>66.07 ± 7.55</td>
</tr>
<tr>
<td>Digit1</td>
<td>93.54 ± 3.10</td>
<td>92.46 ± 4.11</td>
<td>93.76 ± 2.55</td>
</tr>
<tr>
<td>USPS</td>
<td>78.13 ± 6.04</td>
<td>71.70 ± 6.04</td>
<td>69.94 ± 5.34</td>
</tr>
<tr>
<td>BCI</td>
<td>54.80 ± 3.19</td>
<td>54.32 ± 2.19</td>
<td>56.15 ± 3.50</td>
</tr>
<tr>
<td>Text</td>
<td>74.12 ± 2.23</td>
<td>74.08 ± 3.01</td>
<td>74.21 ± 3.44</td>
</tr>
</tbody>
</table>

Table 5: Summary of the UCI data sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>#Points</th>
<th>#Dim</th>
<th>#Testing points</th>
<th>#Training points</th>
<th>#Labeled points</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast</td>
<td>893</td>
<td>10</td>
<td>137</td>
<td>546</td>
<td>(10, 20, 50)</td>
</tr>
<tr>
<td>heart</td>
<td>270</td>
<td>13</td>
<td>138</td>
<td>216</td>
<td>(10, 20, 50)</td>
</tr>
<tr>
<td>australian</td>
<td>690</td>
<td>14</td>
<td>138</td>
<td>552</td>
<td>(10, 20, 50)</td>
</tr>
<tr>
<td>sonar</td>
<td>208</td>
<td>60</td>
<td>42</td>
<td>166</td>
<td>(10, 20, 50)</td>
</tr>
<tr>
<td>ionosphere</td>
<td>351</td>
<td>34</td>
<td>71</td>
<td>280</td>
<td>(10, 20, 50)</td>
</tr>
</tbody>
</table>

Table 6: AUC (area under curve) (%) on the UCI data sets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM</th>
<th>TSVM</th>
<th>SS-EM</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast</td>
<td>97.02 ± 1.01</td>
<td>99.45 ± 0.32</td>
<td>97.29 ± 1.26</td>
<td>99.53 ± 0.34</td>
</tr>
<tr>
<td>heart</td>
<td>91.95 ± 1.01</td>
<td>99.42 ± 0.87</td>
<td>98.27 ± 1.13</td>
<td>99.59 ± 0.33</td>
</tr>
<tr>
<td>australian</td>
<td>99.44 ± 0.84</td>
<td>99.45 ± 0.94</td>
<td>98.52 ± 0.91</td>
<td>99.57 ± 0.28</td>
</tr>
<tr>
<td>sonar</td>
<td>80.78 ± 9.27</td>
<td>79.25 ± 8.96</td>
<td>75.97 ± 6.49</td>
<td>79.63 ± 5.22</td>
</tr>
<tr>
<td>ionosphere</td>
<td>80.66 ± 4.95</td>
<td>75.08 ± 6.88</td>
<td>89.46 ± 3.89</td>
<td>92.36 ± 2.18</td>
</tr>
<tr>
<td>Proposed</td>
<td>98.52 ± 5.15</td>
<td>88.34 ± 7.93</td>
<td>95.46 ± 3.96</td>
<td>94.25 ± 3.96</td>
</tr>
</tbody>
</table>

Table 6 presents the results obtained by our algorithm, and two other cluster assumption based SSL algorithms, transductive SVM (TSVM) and semi-supervised EM (Semi-EM). The results obtained
by the supervised SVM algorithm are also included as a baseline. The highest AUC score in each row is shown in italics. We perform a Bonferroni–Dunn test [40,33] on the results to test for significant differences between the performance of different algorithms. The numbers shown in bold indicate that these results are significantly better than the others (under the significance level of 95%). From Table 6, one can observe that the proposed algorithm outperforms the supervised SVM significantly on almost all the data sets, which indicates that by incorporating unlabeled points into training, the proposed algorithm has consistently improved the classification performance. In contrast, the classification performance of both TSVM and Semi-EM deteriorate on several data sets. Based on these computational results, it can be concluded that on these data sets, the performance of the proposed algorithm is highly competitive with the other state-of-the-art SSL algorithms.

### 4.4. Performance on imbalanced data set

Traditional SSL algorithms usually lead to poor performance on imbalanced data sets, especially when there are multiple clusters in one class. In this experiment, we further evaluate the robustness of the proposed algorithm on imbalanced data sets. The USPS data set used here is a 10-class handwritten digit data set, which contains 7291 training patterns and 2007 testing patterns. The ten classes respectively correspond to the handwritten digit numbers from zero to nine. The dimensionality of the input patterns are reduced to 50 using principle component analysis (PCA). Each time, we choose one class of digits as positive patterns and the remaining nine classes of digits as negative patterns. In this way, the positive and negative classes are imbalanced, and the negative class tends to have multiple clusters. For each class of digits, we randomly label 5 points. Thus, there are 50 labeled points in total, of which 5 are positive and the other 45 are negative. The hyperparameter $C$ and $\gamma$ are selected using 5-fold CV on the labeled training data. We run our algorithm 10 times with randomly generated labeled points. The AUC scores averaged over the 10 trials are reported.

Table 7 presents the AUC scores obtained by the proposed algorithm, semi-supervised EM, and supervised SVM. The highest AUC scores obtained by these algorithms are shown in italics, and significantly better results are shown in bold (with a significance level of 95% in the Bonferroni–Dunn test). From Table 7, it is clear that the proposed algorithm yields better, or at least comparable results compared to the supervised SVM and the Semi-EM algorithm. This demonstrates that the proposed algorithm is robust to imbalanced data and is capable of handling data sets with multiple clusters in one class.

### 4.5. Sensitivity to hyperparameters and a particular model selection method for the proposed algorithm

In this section, we conduct a set of experiments to investigate the sensitivity of our algorithm to the hyperparameters $C$ and $\gamma$. Based on the experimental results, we also propose a particular method for selecting $\gamma$.

Fig. 5 shows the performance of our algorithm with different values of $C$. We can observe that the AUC score remains almost unchanged over a wide range of $C$ values (approximately from $10^0$ to $10^{12}$). However, the performance of our algorithm is sensitive to the hyperparameter $\gamma$. From Figs. 6–9, we can observe that the test error rates decrease roughly monotonically as $\gamma$ increases, while
the errors increase dramatically when $\gamma$ increases to a certain point. One can conjecture that a turning point exists for $\gamma$ such that on the left side of this point, we obtain higher prediction accuracy with larger values of $\gamma$; while on the right side of this point, the classifier breaks down.

In fact, we can confirm the above conjecture from the following two observations. First, from the original SVM formulation (13), we know that $1/\|w\|$ corresponds to the margin between the support vectors and the decision hyperplane. Second, from expressions (15), (27) and (28), we notice that assigning a larger value to $\gamma$ corresponds to ensuring that a subset is classified correctly with higher probability. Certainly, if we want to ensure that the subsets be placed on the right side of the decision hyperplane with higher confidence (equivalent to assigning larger values to $\gamma$), we will obtain a narrower margin. One can verify this observation from Figs. 6–9: $1/\|w\|$ decreases to zero as $\gamma$ grows from 0 to the turning point. As $\gamma$ further increases, no hyperplane that satisfies the requirement can be found, and the SOCP tends to yield a trivial solution. From Figs. 6–9, we can observe that $1/\|w\|$ grows quickly as $\gamma$ moves to the right side of the turning point.

Based on the above observations, we suggest that the best value of hyperparameter $\gamma$ can be found by finding the turning point at which the margin ($1/\|w\|$) approaches 0. The underlying idea is the optimal decision hyperplane is the one that maximizes the probability that each subset is classified correctly. To test this suggestion, we conduct an experiment on the USPS data set with this model selection technique. We fix C = 100, and let $\gamma$ increase from 0 with a step-size of 0.1. The model selection process is terminated when $1/\|w\|$ ceases to decrease, and the $\gamma$ value at the turning point is selected.

The experimental results are presented in Table 8. One can observe that the obtained classification results are comparable to that obtained by 5-fold CV. However, the computational cost of this method is considerably less than that required by 5-fold CV.

5. Conclusions

In this paper, we have introduced a novel inductive semi-supervised learning algorithm by combining the efficient kNN algorithm with a maximal margin classifier. We have tested our algorithm on a variety of data sets, and showed that our algorithm outperforms several state-of-the-art SSL algorithms on most cluster-like data sets, and is also comparable with graph-based SSL algorithms on manifold-like data sets. Furthermore, it rarely produced degraded performance compared to the pure SL algorithms that are trained with the labeled patterns, which indicates that the proposed algorithm is robust across a variety of application domains.

Future research work may focus on studying how to partition the training data into subsets in a more appropriate way. For example, instead of using the kNN algorithm, we can adopt clustering techniques to explore the cluster structure of the data. It is expected that the performance of our algorithm can be further improved by partitioning the data into subsets more effectively. Another problem for future research is the model selection method mentioned in Subsection 4.5. There may exist efficient ways to find the optimal $\gamma$ automatically without performing cross validation.

Conflict of interest

None declared.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 61273233, the Research Fund for the Doctoral Program of Higher Education under Grant 20120002110035, the National Key Technology R&D Program under Grant 2012BAF01B03, the Project of China Ocean Association under Grant DY125-25-02, and Tsinghua University Initiative Scientific Research Program under Grants 2011THZ07132.

References