Scaling Up Graph-Based Semisupervised Learning via Prototype Vector Machines

Kai Zhang, Liang Lan, James T. Kwok, Slobodan Vucetic, and Bahram Parvin

***Abstract*— When the amount of labeled data are limited, semi-supervised learning can improve the learner's performance by also using the often easily available unlabeled data. In particular, a popular approach requires the learned function to be smooth on the underlying data manifold. By approximating this manifold as a weighted graph, such graph-based techniques can often achieve state-of-the-art performance. However, their high time and space complexities make them less attractive on large data sets. In this paper, we propose to scale up graph-based semisupervised learning using a set of sparse prototypes derived from the data. These prototypes serve as a small set of data representatives, which can be used to approximate the graph-based regularizer and to control model complexity. Consequently, both training and testing become much more efficient. Moreover, when the Gaussian kernel is used to define the graph affinity, a simple and principled method to select the prototypes can be obtained. Experiments on a number of real-world data sets demonstrate encouraging performance and scaling properties of the proposed approach. It also compares favorably with models learned via ℓ_1-regularization at the same level of model sparsity. These results demonstrate the efficacy of the proposed approach in producing highly parsimonious and accurate models for semisupervised learning.**

***Index Terms*— Graph-based methods, large data sets, low-rank approximation, manifold regularization, semisupervised learning.**

I. Introduction

IN MANY data analysis and mining applications, the amount of unlabeled data available can be huge. However, the amount of labeled data remains scarce, due to the expensive and tedious human labeling process involved. Semisupervised learning [10], which can use unlabeled data together with the labeled data during training, is thus an effective approach to improve the learner's generalization performance.

In semisupervised learning, the label dependencies among the samples are captured by exploiting the intrinsic geometric structure of the data. This can be implemented using the cluster assumption, which encourages the separating hypersurface to pass through low-density regions [11], [19]. Another popular smoothness assumption is the manifold assumption, which assumes that the underlying function is smooth on a low-dimensional manifold formed by the data. Often, this manifold is approximated by a weighted graph defined on all the labeled and unlabeled data, and with the graph's affinity matrix encoding the similarities between the samples. This leads to the development of various graph-based semisupervised learning algorithms [4], [22], [26], [33], [45], [46], [48]. Besides these, techniques based on generative models [2], [27], self-training [39], co-training [7], and conditional random fields [20], [21] have also been used for semisupervised learning. An excellent survey can be found in [47].

In this paper, we will focus on the graph-based approach [4], [6], [48], which has a clear mathematical framework, good empirical performance and has also been widely applied in a variety of real-world problems. In particular, we will consider the manifold regularization framework [4]. It incorporates an additional regularizer to ensure that the learned function is smooth on the manifold; while at the same time the predictions on the labeled data are required to be consistent with the known class labels. In contrast to some other graph-based learning methods, this regularization framework is semisupervised (not transductive) and allows generalization to out-of-sample (unseen) patterns. Besides semisupervised learning, the use of manifolds has also been successfully applied in other learning problems such as manifold learning, dimension reduction, and clustering [3], [25], [29], [31], [34].

However, as graph-based semisupervised learning needs to manipulate the affinity (kernel) matrix of the graph (which is defined on all the available data), its space complexity has to grow (at least) quadratically with the data set size n [4], [45], [48]. On the other hand, its time complexity typically scales as n^3. These pose a big challenge in large-scale applications.

In recent years, many efforts have been devoted to scaling up graph-based semisupervised learning. For example, Zhu and Lafferty [49] proposed an elegant framework that combines the generative mixture model with graph-based regularization. Recently, this is further extended for semisupervised feature selection [38]. However, when the data

Manuscript received March 26, 2013; revised August 29, 2013 and January 3, 2014; accepted March 28, 2014. Date of publication April 21, 2014; date of current version February 16, 2015. This work was supported in part by the National Institute of Health under Grant R01-CA140663 and in part by the Lawrence Berkeley National Laboratory under Contract DE-AC02-05CH11231. The work of J. T. Kwok was supported by the Hong Kong Competitive Earmarked Research Grant under Project 614012.

K. Zhang is with NEC Laboratories America, Inc., Princeton, NJ 08540 USA (e-mail: kzhang@nec-labs.com).

L. Lan is with Huawei Noah's Ark Laboratory, Hong Kong (e-mail: lanliang@temple.edu).

J. T. Kwok is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: jamesk@cs.ust.hk).

S. Vucetic is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: vucetic@temple.edu).

B. Parvin is with the Life Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA (e-mail: b_parvin@lbl.gov).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2315526

are high-dimensional, the curse of dimensionality sets in and could prevent the mixture model from fitting the data well [49]. In [13], a nonparametric function induction framework is proposed, which performs prediction based on a subset of preselected samples. However, it cannot enforce smoothness of the target function over those unlabeled data not in the preselected sample subset. In [8], the Nyström method [37] is used to numerically approximate the large matrix inverse associated with the graph-based semisupervised learning method of [45]. This leads to an efficient, transductive algorithm. The Nyström low-rank approximation is also used to scale up support vector training recently in [43], and transfer learning in the Hilbert space [44]. Other very recent attempts, such as the primal Laplacian supported vector machine (LapSVM) [24] and the anchor graph [23] will be presented in Section II-B.

Our key observation is that the computational intensiveness of graph-based semisupervised learning arises from the regularization term. On the one hand, this requires manipulation (multiplication, or inverse) of the $n \times n$ kernel matrix, which is impractical for large problems. On the other hand, the representer theorem [4] shows that the learned model spans over both labeled and unlabeled samples, making it inefficient in both training and testing.

The main contribution of this paper is in improving the scalability of semisupervised learning algorithms with the use of prototypes. These refer to a small set of points in the input space that can be used as a replacement of the original data in obtaining an efficient yet accurate predictive solution. Specifically, there are two types of prototypes used in the paper. First, low-rank approximation prototypes, which are used to obtain a low-rank approximation of the kernel matrix. This in turn allows graph-based regularization to be performed efficiently without having to manipulate the $n \times n$ kernel matrix. Second, label-reconstruction prototypes, which form a set of basis functions to span the predictive model. Since the number of prototypes is much smaller than the sample size, a highly compact model can be obtained, leading to fast training and testing. Moreover, without this functional representation, the learned model can only be transductive, and is unable to handle out-of-sample patterns. It will also be shown that when the graph affinity is defined by the Gaussian kernel, both kinds of prototypes can be selected in a principled and simple manner. Experiments on a number of real-world data sets demonstrate encouraging performance and scaling properties of the proposed approach.

The rest of this paper is organized as follows. In Section II, we first give a brief review on graph-based semisupervised learning algorithms and low-rank approximation algorithms. In Section III, we discuss the two important roles of prototypes in scaling up graph-based semisupervised learning, namely approximating the graph-based regularizer and simplifying the model parametrization. We also show that when the Gaussian kernel is used to define the graph affinity, these two criteria lead to a simple and unified prototype selection scheme via k-means clustering. In Section IV, we demonstrate how to use the prototypes to reduce the size of the optimization problem associated with semisupervised learning.

In Section V, we perform experiments on a number of real-world data sets to demonstrate the performance of the proposed method. Finally, Section VI gives some concluding remarks. A preliminary version of this paper is published in [42].

In the sequel, the transpose of vector/matrix is denoted by the superscript$^\top$, the identity matrix by $\mathbf{I}$, and the n-dimensional vector of all ones by $\mathbf{1}_n$. Moreover, for matrix $\mathbf{A} = [a_{ij}]$, we use $\|\mathbf{A}\|_F = \sqrt{\sum_{ij} a_{ij}^2}$ for its Frobenius norm; and $\|\mathbf{A}\|_2$ for its spectral norm (which is equal to the maximum singular value of $\mathbf{A}$). Besides, diag($\mathbf{a}$) converts a n-dimensional vector $\mathbf{a}$ to a $n \times n$ diagonal matrix.

II. Related Work

Section II-A first reviews some standard graph-based semisupervised learning algorithms. Section II-B then introduces a number of representative recent developments that can handle larger data sets. Finally, the approach of low-rank approximation, which will be a core component of the proposed method, is introduced in Section II-C.

A. Graph-Based Semisupervised Learning

In semisupervised learning, we are given a set of l labeled samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and u unlabeled samples $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$, where $\mathbf{x}_i \in \mathbb{R}^d$, and $y_i \in \{-1, 1\}$ is the class label. For simplicity of exposition, we only consider binary classification problems here. The y_i's from all the labeled samples can be concatenated to form the vector $\mathbf{y}_l \in \{-1, 1\}^l$. For a given learning algorithm, let f be the learned function f, $\mathbf{f} \in \mathbb{R}^n$ be the vector of f's evaluations on the $n = l + u$ samples. This $\mathbf{f}$ can be divided into two subvectors: $\mathbf{f}_l \in \mathbb{R}^l$ for the labeled samples, and $\mathbf{f}_u \in \mathbb{R}^u$ for the unlabeled ones.

A number of graph-based semisupervised learning algorithms can be formulated as the following optimization problem:

$$\min_f \mathbf{f}^\top \mathbf{S}\mathbf{f} + C_1 L(\mathbf{f}_l, \mathbf{y}_l) + C_2 \Omega(f) \tag{1}$$

where C_1 and C_2 are the regularization parameters. The first term in (1) enforces smoothness of the predicted labels with respect to the data's manifold structure, which is captured via the matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$. Usually, $\mathbf{S}$ is chosen as the graph Laplacian matrix

$$\mathbf{S} = \mathbf{D} - \mathbf{K} \tag{2}$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$ is the kernel (adjacency) matrix of the graph [with associated kernel function $K(\cdot, \cdot)$], and $\mathbf{D} = \text{diag}(\mathbf{K}\mathbf{1}_n)$ is the corresponding degree matrix. Alternatively, the normalized graph Laplacian matrix

$$\mathbf{S} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{K} \mathbf{D}^{-\frac{1}{2}} \tag{3}$$

can also be used. The second term in (1) requires the predictions on the labeled samples be consistent with the known class labels, and the discrepancy is penalized by the empirical loss function $L(\cdot, \cdot)$. Finally, the third term enforces regularization on the learned function f. Popular choices for $\Omega(f)$ include the norm of f in some reproducing kernel Hilbert

space (RKHS) [4]. Alternatively, Zhou *et al.* [45] simply uses $\Omega(f) = \|\mathbf{f}_u\|$, which regularizes predictions on the unlabeled samples only.

Well-known examples that follow the formulation in (1) include:

1) the method of learning with local and global consistency (LGC) [45], where $\mathbf{S}$ is the normalized graph Laplacian, and the two regularization parameters C_1 and C_2 are set to be equal;
2) learning using Gaussian fields and harmonic functions [48], where $\mathbf{S}$ is the unnormalized graph Laplacian, C_1 approaches infinity, and $C_2 = 0$;
3) the Laplacian regularized least-squares (LapRLS) and the LapSVM algorithms [4], which will be described in more detail in the following section.

The standard representer theorem in kernel learning can be extended to graph-based semisupervised learning. Specifically, Belkin *et al.* [4] showed that the learned function in (1) admits a representation as an expansion over all the labeled and unlabeled samples

$$f(\mathbf{x}) = \sum_{i=1}^{l+u} \alpha_i K(\mathbf{x}, \mathbf{x}_i) \tag{4}$$

where $\alpha_i \in \mathbb{R}$.

1) Laplacian Regularized Least-Squares: In this paper, we will focus on two loss functions commonly used in (1), the square and the hinge loss. The square loss function leads to the LapRLS algorithm [4], which solves the optimization problem

$$\min_f \frac{1}{l}\sum_{i=1}^{l}(y_i - f(\mathbf{x}_i))^2 + \gamma_A\|f\|_K^2 + \frac{\gamma_I}{(l+u)^2}\mathbf{f}^\top\mathbf{S}\mathbf{f} \tag{5}$$

where $\|f\|_K$ is the norm of f in the RKHS induced by the kernel function K, and γ_A, γ_I are regularization parameters related to the C_1, C_2 in (1) as $C_1 = (l+u)^2/\gamma_I l$ and $C_2 = \gamma_A(l+u)^2/\gamma_I$. By plugging in the form of f in (4), we obtain

$$\min_{\boldsymbol{\alpha}} \frac{1}{l}(\mathbf{y}-\mathbf{J}\mathbf{K}\boldsymbol{\alpha})^\top(\mathbf{y}-\mathbf{J}\mathbf{K}\boldsymbol{\alpha}) + \gamma_A\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha} + \frac{\gamma_I}{(l+u)^2}\boldsymbol{\alpha}^\top\mathbf{K}\mathbf{S}\mathbf{K}\boldsymbol{\alpha}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{l+u}]^\top \in \mathbb{R}^{l+u}$, $\mathbf{y} = [y_1, \ldots, y_l, 0, \ldots, 0]^\top \in \mathbb{R}^{l+u}$, and $\mathbf{J} = \text{diag}([\underbrace{1, \ldots, 1}_{l}, \underbrace{0, \ldots, 0}_{u}]^\top) \in \mathbb{R}^{(l+u)\times(l+u)}$. It can be shown that the solution of $\boldsymbol{\alpha}$ is given by

$$\boldsymbol{\alpha}^* = \left(\mathbf{J}\mathbf{K} + \gamma_A l\mathbf{I} + \frac{\gamma_I l}{(l+u)^2}\mathbf{S}\mathbf{K}\right)^{-1}\mathbf{y}. \tag{6}$$

2) Laplacian SVM: Instead of using the square loss in (5), the LapSVM [4] uses the hinge loss, which then leads to

$$\min_f \frac{1}{l}\sum_{i=1}^{l}(1 - y_i f(\mathbf{x}_i))_+ + \gamma_A\|f\|_K^2 + \frac{\gamma_I}{(l+u)^2}\mathbf{f}^\top\mathbf{S}\mathbf{f}$$

where $(1 - yf(\mathbf{x}))_+ = \max(0, 1 - yf(\mathbf{x}))$. Again, by plugging in (4), we obtain the primal problem

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{\xi}}\ & \frac{1}{l}\sum_{i=1}^{l}\xi_i + \gamma_A\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha} + \frac{\gamma_I}{(l+u)^2}\boldsymbol{\alpha}^\top\mathbf{K}\mathbf{S}\mathbf{K}\boldsymbol{\alpha} \\ \text{s.t.}\quad & y_i\left(\sum_{j=1}^{l+u}\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, l, \\ & \xi_i \geq 0, \quad i = 1, \ldots, l\end{aligned}$$

where $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_l]^\top \in \mathbb{R}^l$. This leads to the solution

$$\boldsymbol{\alpha}^* = \left(2\gamma_A\mathbf{I} + 2\frac{\gamma_I}{(l+u)^2}\mathbf{S}\mathbf{K}\right)^{-1}\mathbf{J}^\top\mathbf{Y}\boldsymbol{\beta}^* \tag{7}$$

where $\mathbf{J} = [\mathbf{I}_l \quad \mathbf{0}] \in \mathbb{R}^{l\times(l+u)}$ with $\mathbf{I}_l$ as the $l \times l$ identity matrix, $\mathbf{Y} = \text{diag}([y_1, \ldots, y_l]^\top)$, and $\boldsymbol{\beta}^* \in \mathbb{R}^l$ is obtained by solving the following optimization problem:

$$\begin{aligned}\boldsymbol{\beta}^* = \max_{\boldsymbol{\beta}}\ & \sum_{i=1}^{l}\beta_i - \frac{1}{2}\boldsymbol{\beta}^\top\mathbf{Q}\boldsymbol{\beta} \\ \text{s.t.}\quad & \sum_{i=1}^{l}\beta_i y_i = 0 \\ & 0 \leq \beta_i \leq \frac{1}{l}, \quad i = 1, \ldots, l\end{aligned}$$

and

$$\mathbf{Q} = \mathbf{Y}\mathbf{J}\mathbf{K}\left(2\gamma_A\mathbf{I} + 2\frac{\gamma_I}{(l+u)^2}\mathbf{S}\mathbf{K}\right)^{-1}\mathbf{J}^\top\mathbf{Y} \in \mathbb{R}^{l\times l}.$$

As can be observed from (6) and (7), both LapRLS and LapSVM require the storing of a number of $(l+u)\times(l+u)$ matrices (such as $\mathbf{K}$, $\mathbf{J}$ and $\mathbf{S}$), and expensive matrix inversion of an $(l+u)\times(l+u)$ matrix. Consequently, this costs $O(n^3)$ time and $O(n^2)$ space (recall that $n = l+u$). Besides, testing is also expensive as the representation of f in (4) can involve n terms. In many real-world applications, u can be huge as there are often an abundance of unlabeled samples. Hence, both the algorithms can become impractical on large data sets.

B. Scaling Up Graph-based Semisupervised Learning

In this section, we briefly review some of the recent advances in scaling up graph-based semisupervised learning algorithms.

1) Nyström Method for Learning With LGC: A popular method for graph-based semisupervised learning is based on LGC [45]. Its predictions on the set of n (labeled and unlabeled) samples are given by

$$(\mathbf{I} - \eta\mathbf{Q})^{-1}\mathbf{y} \tag{8}$$

where $\mathbf{y} = [y_1, \ldots, y_l, 0, \ldots, 0]^\top \in \mathbb{R}^n$, $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is the normalized similarity matrix on the samples, and η is a regularization parameter. As (8) involves the inverse of the $n \times n$ matrix $\mathbf{I} - \eta\mathbf{Q}$, it is expensive when n is large. To alleviate this problem, Camps-Valls *et al.* [8] used the Nyström method (which will be reviewed in Section II-C) to speed up the computation of $(\mathbf{I} - \eta\mathbf{Q})^{-1}$. Specifically, they

use the Nyström method to efficiently obtain an approximate eigenvalue decomposition $\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ [where $\mathbf{V} \in \mathbb{R}^{n\times m}$ and $\mathbf{\Lambda} \in \mathbb{R}^{m\times m}$] of $\mathbf{Q}$ in $O(m^2 n)$ time. Using the Woodbury formula, (8) can then be written as $\mathbf{y} - \mathbf{V}(\mathbf{\Lambda}\mathbf{V}^\top\mathbf{V} - \eta^{-1}\mathbf{I})^{-1}\mathbf{\Lambda}\mathbf{V}^\top\mathbf{y}$, which can be computed in $O(m^2 n)$ time. By choosing $m \ll n$, this approach is thus very suitable for large-scale applications. However, as will be demonstrated in Section V-B, empirically this approximation can hamper the prediction accuracy.

2) Primal LapSVM: As discussed in Section II-A1, traditionally the LapSVM is solved in the dual formulation. Very recently, new strategies are proposed to solve the LapSVM in the primal [24], which can be efficiently performed with preconditioned conjugate gradient [30]. Convergence can be further speed up using an early stopping criterion based on the predictions on the unlabeled data or validation examples. The resultant time complexity is reduced to $O(cn^2)$, where c is an empirical constant much smaller than n. Though better than the original $O(n^3)$ complexity, this quadratic complexity is still more expensive than other competitive methods.

3) Fast Nonparametric Function Induction: Delalleau *et al.* [13] proposed a nonparametric approach for estimating the continuous labels of unlabeled examples. For an out-of-sample example $\mathbf{x}$, its prediction

$$f(\mathbf{x}) = \frac{\sum_{j=1}^{n} W(\mathbf{x}, \mathbf{x}_j) f(\mathbf{x}_j)}{\sum_{j=1}^{n} W(\mathbf{x}, \mathbf{x}_j)} \tag{9}$$

has the same form as Parzen window regressors. Here, $f(\mathbf{x}_j)$ is the estimated label for training sample $\mathbf{x}_j$ (which may be unlabeled) obtained in the training phase. A straightforward implementation, however, involves $O(n^3)$ time and $O(n^2)$ space on training. To reduce these complexities, Delalleau *et al.* [13] force the summations in (9) to only involve a preselected sample subset S, $|S| \ll n$, and do not enforce smoothness among the large number of unlabeled samples not in S. As will be empirically demonstrated in Section V-B, these restrictions can hamper the prediction accuracy.

4) Anchor Graph: Liu *et al.* [23] borrowed the idea of [13] and [42] and proposed to use the so-called anchor points (which are the same as prototypes in [42]) to build sparse graphs for semisupervised learning. The adjacency matrix is computed by a k-nearest-neighbor graph, or by solving a quadratic program (QP) similar to that in locally linear embedding [29] with additional sparsity constraints. In both cases, the computational cost is very expensive for large-scale applications.

C. Low-Rank Approximation

Given a symmetric, positive semidefinite matrix (such as the kernel matrix) $\mathbf{K} \in \mathbb{R}^{n\times n}$, a low-rank approximation algorithm [14], [17] produces an approximation of the form $\mathbf{G}\mathbf{G}^\top$, where $\mathbf{G} \in \mathbb{R}^{n\times m}$ and $m \ll n$. If the columns of $\mathbf{G}$ are linearly independent, $\mathbf{G}\mathbf{G}^\top$ is of rank m. It is well known that the optimal rank-m approximation $\mathbf{K}^{(m)}$ of $\mathbf{K}$ (with respect to the spectral or the Frobenius norm) can be easily constructed from its k leading eigenvectors and eigenvalues. Specifically, let the eigenvalue decomposition of $\mathbf{K}$ be $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$, where the columns of $\mathbf{U}$ contain its eigenvectors, and $\mathbf{\Sigma} = \text{diag}[(\sigma_1, \sigma_2, \ldots, \sigma_n)]$ contains the corresponding eigenvalues in descending order. Then, $\mathbf{K}^{(m)} = \mathbf{U}^{(m)}\mathbf{\Sigma}^{(m)}{\mathbf{U}^{(m)}}^\top$, where $\mathbf{U}^{(m)}$ contains the m leading eigenvectors and $\mathbf{U}^{(m)}$ contains the m leading eigenvalues. The corresponding low-rank approximation errors with respect to the spectral norm and Frobenius norm are $\|\mathbf{K} - \mathbf{K}^{(m)}\|_2 = \sigma_{m+1}$ and $\|\mathbf{K} - \mathbf{K}^{(m)}\|_F^2 = \sum_{i=m+1}^{n} \sigma_i^2$, respectively. However, while $\mathbf{K}^{(m)}$ can be obtained easily from the leading eigenvectors and eigenvalues of $\mathbf{K}$, directly finding this eigenvalue decomposition takes $O(n^3)$ time. For large $\mathbf{K}$, this can be expensive and hence more efficient alternatives have to be sought.

In this paper, we will use an efficient low-rank approximation approach called the Nyström method [15], [16], [37]. It first chooses a subset of m rows/columns from $\mathbf{K}$. Let $\mathbf{K}_{nm} \in \mathbb{R}^{n\times m}$ be the submatrix of $\mathbf{K}$ containing these selected columns, and $\mathbf{K}_{mm} \in \mathbb{R}^{m\times m}$ the submatrix containing the intersection of the selected columns and rows. Using the eigenvalue decomposition $\mathbf{K}_{mm} = \mathbf{U}_m\mathbf{\Lambda}_m\mathbf{U}_m^\top$, the matrix containing the m leading eigenvectors of $\mathbf{K}$ is approximated by the Nyström extension as $\mathbf{U}^{(m)} \simeq \mathbf{U} = \mathbf{K}_{nm}\mathbf{U}_m\mathbf{\Lambda}_m^{-1}$ [37]. A useful consequence is that the whole matrix can then be approximated as

$$\mathbf{K} \simeq \mathbf{U}\mathbf{\Lambda}_m\mathbf{U}^\top = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{nm}^\top. \tag{10}$$

Note that, $\mathbf{K}_{nm}$ can be computed in $O(mn)$ time, and $\mathbf{K}_{mm}^{-1}$ in $O(m^3)$ time. Thus, using (10), $\mathbf{K}$ can be written as $\mathbf{G}\mathbf{G}^\top$, where $\mathbf{G} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1/2}$, in $O(m^2 n + m^3)$ time, which is linear in the sample size n.

Traditionally, the m columns are sampled randomly from $\mathbf{K}$ [37] or by a probabilistic sampling scheme [15]. More generally, they can be chosen as landmark points $\{\mathbf{u}_i\}_{i=1}^m$ obtained from summarizing the data. The approximation in (10) is then applied as

$$\hat{\mathbf{K}} = \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top \tag{11}$$

where $\mathbf{W} \in \mathbb{R}^{m\times m}$ [with $\mathbf{W}_{ij} = K(\mathbf{u}_i, \mathbf{u}_j)$] is the kernel matrix evaluated on the landmark points, and

$$\mathbf{E} = [K(\mathbf{x}_i, \mathbf{u}_j)] \in \mathbb{R}^{n\times m} \tag{12}$$

is the cross-similarity matrix between the whole data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and the landmark points. For a number of commonly used kernels (such as the Gaussian, linear, and polynomial kernels), it has been shown that the approximation error $\|\mathbf{K} - \hat{\mathbf{K}}\|_F$ is bounded by the encoding errors of these landmark points [40], [41]. Based on this error analysis, we can choose the k-means cluster centers of the data as the landmark points. In the implementation, we further fix the number of k-means iteration to a small number (such as five). The complexity of k-means is then linear in the sample size and dimension (assuming that each kernel evaluation takes time linear in the dimension), and is thus suitable for large-scale problems.

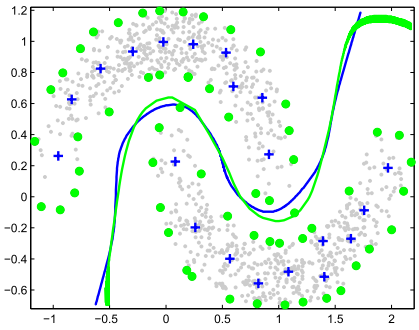


Fig. 1. Decision boundaries of the SVM (green) and a prototype-based model (blue) on a toy data set. Circles are the support vectors and crosses are the prototypes.

III. Approximations via Prototypes

This section discusses the basic idea of using two types of prototypes to enable the optimization of graph-based semisupervised learning more efficient. By prototypes, we mean a small set of points in the input space that can be used as a replacement of the original data in obtaining an efficient yet accurate predictive solution. In this paper, there are two types of prototypes serving different purposes: 1) low-rank approximation prototypes that are used for low-rank approximation of the graph Laplacian matrix (Section III-A) and 2) label-reconstruction prototypes that are used in label-reconstruction for accurate model representation (Section III-B).

A. Low-Rank Approximation Prototypes

Recall that the graph Laplacian matrix $\mathbf{S}$ in (2) or (3) is constructed from the kernel matrix $\mathbf{K}$ of the graph. In practice, $\mathbf{K}$ usually has a rank much smaller than its size [37]. This offers the opportunity of reducing the computational burden of graph-based regularization by performing low-rank approximation (Section II-C).

Using the low-rank approximation $\mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top$ of $\mathbf{K}$ in (11), the unnormalized graph Laplacian $\mathbf{S}$ in (2) can be approximated as

$$\mathbf{S} \simeq \tilde{\mathbf{D}} - \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top \tag{13}$$

where $\tilde{\mathbf{D}} = \text{diag}(\mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top\mathbf{1}_n)$. The corresponding (approximated) graph-based regularizer is then $\mathbf{f}^\top\mathbf{S}\mathbf{f} \simeq \mathbf{f}^\top(\tilde{\mathbf{D}} - \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top)\mathbf{f}$. Similarly, when $\mathbf{S}$ is the normalized graph Laplacian, it is approximated as

$$\mathbf{S} \simeq \mathbf{I} - \mathbf{E}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{W}^{-1}\mathbf{E}^\top\tilde{\mathbf{D}}^{-\frac{1}{2}} \tag{14}$$

and the corresponding regularizer becomes $\mathbf{f}^\top\mathbf{S}\mathbf{f} \simeq \mathbf{f}^\top(\mathbf{I} - \mathbf{E}\tilde{\mathbf{D}}^{-1/2}\mathbf{W}^{-1}\mathbf{E}^\top\tilde{\mathbf{D}}^{-1/2})\mathbf{f}$.

B. Label-Reconstruction Prototypes

1) Motivating Example: As discussed in Section II-A, the inclusion of a graph-based regularizer results in the model (4) that expands over all the labeled and unlabeled samples. This leads to slow training and testing when n is large. In practice, a classifier seldom needs all the samples as basis functions. Fig. 1 shows a motivating example on a binary classification problem. We first train a standard SVM using the whole data set. The obtained decision boundary (in green) and support vectors (circles) are shown. To construct an alternative classifier, we choose a set of label-reconstruction prototypes (crosses) from each class. To predict the label of a test sample $\mathbf{x}$, we compute its similarities with all the label-reconstruction prototypes, and then linearly combine their labels using the similarities as weights. The resultant decision boundary is shown in blue. As can be observed, the two decision boundaries are very similar. This illustrates that instead of using the whole data set, a small set of properly chosen label-reconstruction prototypes can successfully reconstruct the decision boundary.

2) Formulation: The above idea can be formalized as follows. We assume that the label of any sample $\mathbf{x}$ can be reconstructed as a weighted combination of the labels of r label-reconstruction prototypes $\{\mathbf{v}_i\}_{i=1}^r$

$$\sum_{i=1}^{r} f_{\mathbf{v}_i} K(\mathbf{x}, \mathbf{v}_i) \tag{15}$$

where $f_{\mathbf{v}_i}$ is the label of prototype $\mathbf{v}_i$. This resembles the nearest-neighbor classifier in that the predicted label of $\mathbf{x}$ is mostly dependent on prototypes $\mathbf{v}_j$ having large similarities (i.e., $K(\mathbf{x}, \mathbf{v}_j)$ values) with $\mathbf{x}$. Intuitively, as long as there are enough label-reconstruction prototypes filling the input space, the label of any sample can be reconstructed reliably from these nearby prototypes.

To make the model more flexible, we allow each $f_{\mathbf{v}_i}$ in (15) to be real-valued even for classification problems. Note that, (15) can be written more compactly as

$$\mathbf{H}\mathbf{f}_v \tag{16}$$

where $\mathbf{f}_v = [f_{\mathbf{v}_1}, f_{\mathbf{v}_2}, \ldots, f_{\mathbf{v}_r}]^\top \in \mathbb{R}^r$ and

$$\mathbf{H} = [K(\mathbf{x}_i, \mathbf{v}_j)] \in \mathbb{R}^{n\times r} \tag{17}$$

is the kernel submatrix defined between the whole data set $\mathbf{X}$ and prototypes $\mathbf{v}_i$'s. On large data sets, one can additionally enforce sparsity on $\mathbf{H}$ by only computing the similarities between the label-reconstruction prototypes and their k-nearest samples. This has been found useful in the construction of large graphs in semisupervised learning [23].

3) Choosing the Label-Reconstruction Prototypes: We now consider how to choose the label-reconstruction prototypes. Note that, (15) can be deemed as an approximation of the complete model in (4). Hence, we want to minimize

$$D\left(\sum_{i=1}^{l+u} \alpha_i K(\mathbf{x}, \mathbf{x}_i), \sum_{i=1}^{r} f_{\mathbf{v}_i} K(\mathbf{x}, \mathbf{v}_i)\right) \tag{18}$$

where $D(\cdot,\cdot)$ measures the difference between the complete model $\sum_{i=1}^{l+u} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$ in (4) and the approximate model $\sum_{i=1}^{r} f_{\mathbf{v}_i} K(\mathbf{x}, \mathbf{v}_i)$ in (15).

In practice, as α_i's, $f_{\mathbf{v}_i}$'s, and $\mathbf{v}_i$'s are unknown during training, it is infeasible to directly minimize (18) with respect to all these variables. Recall that $\{K(\cdot, \mathbf{x}_i)\}_{i=1}^{l+u}$ forms a basis for the complete model and $\{K(\cdot, \mathbf{v}_i)\}_{i=1}^{r}$ forms a basis for the approximate model. In order for these two models to have similar representation power, we expect that the two basis can

be reliably coded by each other. More specifically, we choose the label-reconstruction prototypes $\mathbf{v}_i$'s so as to minimize the error of coding each component in $\{K(\cdot, \mathbf{x}_i)\}_{i=1}^{l+u}$ with the best element in $\{K(\cdot, \mathbf{v}_i)\}_{i=1}^{r}$

$$\min_{\mathbf{v}_1,\ldots,\mathbf{v}_r} \mathcal{Q} \equiv \sum_{i=1}^{l+u} \Big(\min_{j=1,2,\ldots,r} D(K(\cdot, \mathbf{x}_i), K(\cdot, \mathbf{v}_j)) \Big). \quad (19)$$

Obviously, the choice of $\mathbf{v}_i$'s depends on the specific kernel $K(\cdot, \cdot)$ and discrepancy measure $D(\cdot, \cdot)$. In the following, we consider the use of the Gaussian kernel $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / 2h^2\right)$. Since it has the same form as the normal distribution, we use the cross entropy for probability distribution functions (pdfs) as $D(\cdot, \cdot)$ in (19). Recall that the cross entropy between the two pdfs p and q is

$$H(p, q) = H(p) + D_{\mathrm{KL}}(p \| q) \quad (20)$$

where $H(p) = -\int p(\mathbf{x}) \log(p(\mathbf{x})) d\mathbf{x}$ is the entropy of p, and $D_{\mathrm{KL}}(p\|q) = \int p(\mathbf{x}) \log\left(p(\mathbf{x})/q(\mathbf{x})\right) d\mathbf{x}$ is the Kullback–Leibler (KL) divergence between p and q [18]. Since $\{\mathbf{x}_1, \ldots, \mathbf{x}_{l+u}\}$ is fixed, minimizing $D(K(\cdot, \mathbf{x}_i), K(\cdot, \mathbf{v}_j)) = H(K(\cdot, \mathbf{x}_i), K(\cdot, \mathbf{v}_j))$ in (19) is the same as minimizing their KL-divergence, which can be written as [18][1]

$$D_{\mathrm{KL}}\left(K(\cdot, \mathbf{x}_i) \| K(\cdot, \mathbf{v}_j)\right) = \frac{1}{4h^2} \|\mathbf{x}_i - \mathbf{v}_j\|^2. \quad (21)$$

The objective in (19) then simplifies as

$$\mathcal{Q} = \frac{1}{4h^2} \sum_{j=1}^{r} \sum_{i \in S_j} \|\mathbf{x}_i - \mathbf{v}_j\|^2$$

where S_j is the subset of kernels in $\{K(\cdot, \mathbf{x}_i)\}_{i=1}^{l+u}$ whose closest label-reconstruction prototype is $K(\cdot, \mathbf{v}_j)$. Interestingly, this is exactly the objective of the k-means clustering (apart from the scaling factor $1/4h^2$). As in Section III-A, the prototypes $\mathbf{v}_i$'s in (15) can again be chosen as the k-means cluster centers. In other words, the low-rank approximation prototypes in Section III-A and the label-reconstruction prototypes introduced here are indeed the same. Consequently, $\mathbf{E}$ in (12) is also the same to $\mathbf{H}$ in (17).

For other types of kernels, the cross entropy in (20) may no longer be suitable and other distance measures will be investigated in the future.

IV. Prototype Vector Machine

In Section III, we introduced two types of prototypes: 1) low-rank-approximation prototypes $\{\mathbf{u}_i\}_{i=1}^{m}$, which avoid the computation of an $n \times n$ graph Laplacian matrix and 2) label-reconstruction prototypes $\{\mathbf{v}_i\}_{i=1}^{r}$, which avoid the use of all the labels in the whole data set. In this section, using the associated simplified equations (13) [or (14)] and (16), we will consider how to realize the proposed prototype vector machine (PVM) with the square and hinge loss functions. As will be seen, since $m, r \ll n$ (the sample size), the optimization of problem (1) can be made much more efficient.

[1]This is a special case of the problem considered in [18], where the simplified model has varying bandwidths and the components in the original model are weighted.

Algorithm 1 PVM Using the Square Loss

1: Perform k-means clustering on the given labeled and unlabeled samples, and use the cluster centers as prototypes.
2: Use (17) to compute $\mathbf{H}$ (and thus obtain its submatrices $\mathbf{H}_l$ and $\mathbf{H}_u$).
3: Substitute the approximate graph Laplacian [(13) or (14)] into (23) and obtain $\mathbf{f}_v^*$.
4: Prediction
   1) on the given unlabeled samples: $\mathbf{f}_u = \mathbf{H}_u \mathbf{f}_v^*$.
   2) on an unseen $\mathbf{x}$: $[K(\mathbf{x}, \mathbf{v}_1), \ldots, K(\mathbf{x}, \mathbf{v}_r)]\mathbf{f}_v^*$.

A. Training

1) Square Loss: Using the square loss, the objective in (1) can be written as

$$\min_{\mathbf{f}_v} (\mathbf{H}\mathbf{f}_v)^\top \mathbf{S}(\mathbf{H}\mathbf{f}_v) + C_1 \|\mathbf{H}_l \mathbf{f}_v - \mathbf{y}_l\|^2 + C_2 \|\mathbf{H}_u \mathbf{f}_v\|^2 \quad (22)$$

where $\mathbf{H}_l \in \mathbb{R}^{l \times r}$ and $\mathbf{H}_u \in \mathbb{R}^{u \times r}$ are submatrices of $\mathbf{H}$ containing the rows corresponding to the labeled and unlabeled samples, respectively, and $\mathbf{S}$ is as defined in (13). Replacing $\mathbf{S}$ with the normalized graph Laplacian (14) is straightforward. Moreover, as in [45], we have used $\|\mathbf{H}_u \mathbf{f}_v\|^2$ to regularize f. Alternatively, one can use the RKHS norm of f (as in [4]). On using (15), this leads to a slightly different regularizer $\mathbf{f}_v^\top \mathbf{K}_v \mathbf{f}_v$, where $\mathbf{K}_v$ is the kernel matrix defined on the r label-reconstruction prototypes, and the results in the sequel can be easily extended.

By setting the derivative of the objective in (22) with respect to $\mathbf{f}_v$ to zero, we obtain the optimal solution of $\mathbf{f}_v$ as

$$\mathbf{f}_v^* = C_1 (\mathbf{H}^\top \mathbf{S}\mathbf{H} + C_1 \mathbf{H}_l^\top \mathbf{H}_l + C_2 \mathbf{H}_u^\top \mathbf{H}_u)^{-1} \mathbf{E}_l^\top \mathbf{Y}_l. \quad (23)$$

The complete algorithm is shown in Algorithm 1. Note that, if k-means clustering is used as the selection scheme for both the low-rank-approximation prototypes and label-reconstruction prototypes, then we simply have $\mathbf{E}_l = \mathbf{H}_l$ in (23).

On using k-means as the unified prototype selection scheme, we have $m = r$. Step 1 of Algorithm 1 then takes $O(ndm)$ time and $O(nm)$ space. Step 2 takes $O(ndr) = O(ndm)$ time (assuming that each kernel evaluation takes $O(d)$ time) and $O(nr) = O(nm)$ space. In step 3, with the use of the m low-rank approximation prototypes, we have $\mathbf{H}^\top \mathbf{S}\mathbf{H} = \mathbf{H}^\top \tilde{\mathbf{D}}\mathbf{H} - (\mathbf{H}^\top \mathbf{E})\mathbf{W}^{-1}(\mathbf{E}^\top \mathbf{H})$ from (13). Since $\tilde{\mathbf{D}}$ is diagonal, $\mathbf{H} \in \mathbb{R}^{n \times r}$ and $\mathbf{E} \in \mathbb{R}^{n \times m}$, $\mathbf{H}^\top \mathbf{S}\mathbf{H}$ can be computed in $O(r^2 n + r^2 m + m^2 r + mnr) = O(nr(m+r)) = O(nm^2)$ time and $O(n \max(r, m)) = O(nm)$ space. Moreover, the inverse operation in (23) involves a $r \times r$ matrix, and takes $O(r^3)$ time. Thus, the total time for computing $\mathbf{f}_v^*$ is $O(nr(m+r) + r^3) = O(nr(m+r)) = O(nm^2)$. Since $m \ll n$, the total space and time complexities are both linear in the sample size, which is much faster than the $O(n^3)$ time for LapRLS (Section II-A).

2) Hinge Loss: For classification problems (with labels in $\{\pm 1\}$), we can use the hinge loss as popularized by the SVM. The other two terms in (22) remain the same, and can be combined together as $\mathbf{f}_v^\top \mathbf{A}\mathbf{f}_v$, where[2]

$$\mathbf{A} = \mathbf{H}^\top \mathbf{S}\mathbf{H} + C_2 \mathbf{H}_u^\top \mathbf{H}_u. \tag{24}$$

Let $\mathbf{e}_k$ be the transpose of the kth row in $\mathbf{H}_l$

$$\mathbf{H}_l = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_l]^\top. \tag{25}$$

The optimization problem in (1) can then be written as

$$\begin{aligned} \min_{\mathbf{f}_v} \ & \frac{1}{2}\mathbf{f}_v^\top \mathbf{A}\mathbf{f}_v + C_1 \sum_{i=1}^{l} \xi_i \\ \text{s.t.} \ & y_i \mathbf{e}_i^\top \mathbf{f}_v \geq 1 - \xi_i, \ i = 1, \ldots, l \\ & \xi_i \geq 0, \ i = 1, \ldots, l \end{aligned}$$

where the first term performs regularization, and the second term is the hinge loss incurred by approximating the labels of $\mathbf{X}_l$ with the reconstructed ones ($\mathbf{H}_l \mathbf{f}_v$). The Lagrangian can be written as

$$\mathcal{L} = \frac{1}{2}\mathbf{f}_v^\top \mathbf{A}\mathbf{f}_v + C_1 \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \beta_i (y_i \mathbf{e}_i^\top \mathbf{f}_v - 1 + \xi_i) - \sum_{i=1}^{l} \gamma_i \xi_i.$$

By setting its derivative with respect to $\mathbf{f}_v$ and ξ_i's to zero, we obtain $C_1 = \beta_i + \gamma_i$, and

$$\mathbf{A}\mathbf{f}_v = \sum_{i=1}^{l} \beta_i y_i \mathbf{e}_i = \mathbf{H}_l^\top \boldsymbol{\beta} \odot \mathbf{y}_l \tag{26}$$

on using (25). Here, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_l]^\top$, and $\odot$ denotes the element wise product. Plugging these back into the Lagrangian, we obtain the dual problem

$$\begin{aligned} \max_{\boldsymbol{\beta}} \ & -\frac{1}{2}(\mathbf{H}_l^\top \boldsymbol{\beta} \odot \mathbf{y}_l)^\top \mathbf{A}^{-1} (\mathbf{H}_l^\top \boldsymbol{\beta} \odot \mathbf{y}_l) + \sum_{i=1}^{l} \beta_i \\ \text{s.t.} \ & 0 \leq \beta_i \leq C_1, \ i = 1, 2, \ldots, l \\ = \ & \max_{\boldsymbol{\beta}} -\frac{1}{2}\boldsymbol{\beta}^\top \mathbf{Q}\boldsymbol{\beta} + \mathbf{1}_l^\top \boldsymbol{\beta} \\ \text{s.t.} \ & 0 \leq \beta_i \leq C_1, \ i = 1, 2, \ldots, l \end{aligned} \tag{27}$$

where

$$\mathbf{Q} = \mathbf{H}_l \mathbf{A}^{-1} \mathbf{H}_l^\top \odot \mathbf{y}_l \mathbf{y}_l^\top \in \mathbb{R}^{l \times l}. \tag{28}$$

After solving this QP, labels of the prototypes $\mathbf{v}_i$'s can be recovered using the Karush–Kuhn–Tucker condition in (26), as

$$\mathbf{f}_v^* = \mathbf{A}^{-1} \mathbf{H}_l^\top \boldsymbol{\beta} \odot \mathbf{y}_l. \tag{29}$$

The complete algorithm is shown in Algorithm 2.

As in Section IV-A1, $\mathbf{H}^\top \mathbf{S}\mathbf{H}$ can be computed in $O(nm^2)$ time. Hence, matrix $\mathbf{A}$ in step 3 can also be computed in $O(nr(m+r) + r^2 u) = O(nr(m+r)) = O(nm^2)$ time, and its inverse in $O(r^3)$ time. Moreover, note that the QP in (27) is of the same form as a standard SVM, and so can be solved readily with state-of-the-art SVM solvers. There are only l optimization variables in this QP, and modern SVM implementations typically have a time complexity that scales linearly with l [36]. So, the total time complexity is $O(nr(m+r)+r^3) = O(nr(m+r)) = O(nm^2)$. Since $m \ll n$, this is again much more efficient than the $O(n^3)$ time required of the standard LapSVM (Section II-A) and the $O(n^2)$ time required of the primal LapSVM (Section II-B2). Moreover, it can be easily seen that its space complexity is $O(mn)$, the same as that when the square loss is used. As a short summary, Table I compares the time and space complexities of a number of semisupervised learning algorithms that will be further studied empirically in Section V-B.

[2]If the RKHS norm of f is used as the regularizer instead of $\|\mathbf{H}_u \mathbf{f}_v\|^2$ as discussed in Section IV-A1, then $\mathbf{A}$ becomes $\mathbf{H}^\top \mathbf{S}\mathbf{H} + C_2 \mathbf{K}_v$, and the results in Section IV-A2 can also be easily extended.

Algorithm 2 PVM Using the Hinge Loss

1: Perform k-means clustering on the given labeled and unlabeled samples, and use the cluster centers as prototypes.
2: Use (17) to compute $\mathbf{H}$ (and thus obtain its submatrices $\mathbf{H}_l$ and $\mathbf{H}_u$).
3: Use the approximate graph Laplacian [(13) or (14)] to compute $\mathbf{A}$ in (24), and subsequently $\mathbf{Q}$ in (28).
4: Solve the QP in (27) to obtain $\boldsymbol{\beta}$.
5: Obtain $\mathbf{f}_v^*$ from (29).
6: Prediction
   1) on the given unlabeled samples: $\text{sgn}(\mathbf{f}_u) = \text{sgn}(\mathbf{H}_u \mathbf{f}_v^*)$.
   2) on an unseen $\mathbf{x}$: $\text{sgn}([K(\mathbf{x}, \mathbf{v}_1), \ldots, K(\mathbf{x}, \mathbf{v}_r)]\mathbf{f}_v^*)$.

B. Prediction

After obtaining $\mathbf{f}_v^*$, the predicted labels on the given unlabeled samples can be computed as $\mathbf{f}_u = \mathbf{H}_u \mathbf{f}_v^*$, and that on an unseen test sample $\mathbf{x}$ as $[K(\mathbf{x}, \mathbf{v}_1), \ldots, K(\mathbf{x}, \mathbf{v}_r)]\mathbf{f}_v^*$. Since it only depends on the r label-reconstruction prototypes, testing takes $O(r)$ time. This is much faster than LapRLS and LapSVM, which take $O(n)$ time.

C. Number of Prototypes

The number of prototypes is an important factor that affects the performance of the proposed method. More prototypes lead to more accurate approximation on both the kernel matrix as well as the class labels. Therefore, this tends to give a better performance. However, the time consumption also grows. Hence, this involves a tradeoff between the accuracy and the efficiency. Empirically, we choose m as 5% of the sample size (or fewer), which gives satisfactory results. In our empirical evaluations (Section V-A.3), we also observe that when m is beyond a certain threshold the performance would reach a plateau.

We are also interested in performance of PVM with respect to choice of the number of prototype vectors as well as the number of labeled and unlabeled samples. First, the larger the number of prototypes, the more complicated concept (e.g., classification boundary) can be potentially delineated. Therefore, more prototypes will lead to a better performance, as has been shown in our empirical evaluations.

TABLE I

TIME AND SPACE COMPLEXITIES FOR THE TRAINING PROCESS OF VARIOUS SEMISUPERVISED LEARNING ALGORITHMS (DESCRIPTIONS OF THE ALGORITHMS CAN BE FOUND IN SECTION V-B). HERE, n IS THE TOTAL NUMBER OF LABELED AND UNLABELED SAMPLES, AND m IS THE NUMBER OF PROTOTYPES IN THE PVM (WHICH IS EQUAL TO THE SAMPLE SUBSET SIZE USED IN THE OTHER ALGORITHMS)

	LGC [45]	LapRLS [4]	LapSVMp [24]	NYS-LGC [8]	NFI [13]	Anchor [23]	PVM (sqr)	PVM (hinge)
time	$O(n^3)$	$O(n^3)$	$O(m^2n)$	$O(n^2)$	$O(m^2n)$	$O(m^2n)$	$O(m^2n)$	$O(n^2)$
space	$O(n^2)$	$O(n^2)$	$O(n^2)^\dagger$	$O(mn)$	$O(mn)$	$O(n^2)^\dagger$	$O(mn)$	$O(mn)$

†In the implementation of LapSVMp and Anchor method, the kernel matrix is partitioned into sub-blocks that are loaded into memory sequentially, which can further reduce the $O(n^2)$ memory consumption to $O(np)$, where p is a user defined number.

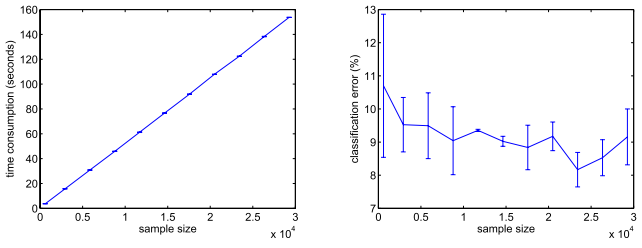
In considering the number of labeled samples, note that given a fixed level of intrinsic difficulty of the learning problem (in terms of the complexity of the classification boundaries), more labeled samples will lead to a better prediction performance. In this case, together with more prototypes, we will also expect a better performance.

If the unlabeled data can improve the performance of learning, more prototypes will be able to summarize the distribution of unlabeled data better, therefore the performance will improve with more prototypes and unlabeled data. However, sometimes, the unlabeled data may not help in semisupervised learning. For example, if the low-density separation assumption is violated [11], [19], [28] then it would be hard to predict whether more prototypes will improve performance, since more prototypes will make the effect of unlabeled samples more significant.

In practice, the decision boundary can be determined in a complicated manner by labeled and unlabeled samples together in semisupervised learning. For example, in [32], it is stated that if the complexity of the distribution under consideration is too high to be learned using labeled data points, but is small enough to be learned using unlabeled data points, then semisupervised learning can improve the performance of the supervised learning task. In this case, we may expect a better performance with more prototypes as well as labeled and unlabeled samples. However, in some cases, the needed relation between labeled and unlabeled samples might not be satisfied. For example, Ben-David *et al.* [5] concluded that using unlabeled data cannot provide sample size guarantees that are better than those using labeled data only, unless one assumes a very strong relationship between the labeled and unlabeled data distribution. Therefore, in case the desired condition is not satisfied, it would be hard to guarantee that more prototypes will lead to improved performance with more labeled and unlabeled samples.

V. EXPERIMENTS

In Section V-A, we investigate how the performance of the proposed PVM varies with sample size, the number of labeled samples and prototypes. In Section V-B, the PVM is compared with other state-of-the-art semisupervised learning algorithms on a number of real-world data sets. Experiments will be performed on both binary and multiclass classification problems. Note that for PVM using the square loss, the formulation in (22) can be extended for multiclass classification in a straightforward manner. For PVM using the hinge loss, we first decompose a C-class classification problem into



(a) (b)

Fig. 2. Performance of the PVM with different sample sizes. (a) Training time versus sample size (note that the error bars here are very small). (b) Error versus sample size.

C one-versus-rest binary classification problems. Prediction is then made by assigning the sample to the class with the highest decision function value.

PVM is a sparse model, as only a small number of prototypes are involved in (15). Another popular approach for the construction of sparse models is using the ℓ_1-regularizer, which encourages model coefficients to approach zero. Recently, this ℓ_1-penalized approach has drawn considerable interest in learning sparse predictive models. For example, lasso [35], the most well-known ℓ_1-penalized sparse model, adds the ℓ_1-regularizer to the square loss function. Hence, a natural question is how the parsimonious model produced by the PVM compares with that obtained by ℓ_1-regularization. This will be addressed in Section V-D.

A. Performance of the PVM

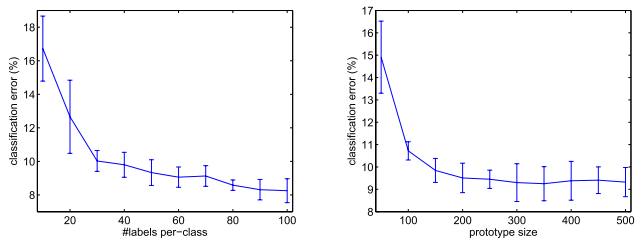
In this section, we study the performance of the PVM by performing experiments on a subset of the MNIST digits data set.[3] We use the five digits (represented as 784-D feature vectors) 3, 5, 6, 8, and 9, leading to a total of 29 270 samples. The Gaussian kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\beta\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{30}$$

is used. As discussed in Section III, this leads to a unified prototype selection scheme, namely that the k-means cluster centers can be used as both the low-rank approximation prototypes and the label-reconstruction prototypes.

1) Variation With Sample Size: In this experiment, we first study how the performance of PVM (using the square loss) varies when the total number (n) of labeled and unlabeled samples is gradually increased from 1000 to 29 270. For each class, 50 samples are labeled. The number of prototypes m is fixed at 200. To reduce statistical

[3]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/



(a) (b)

Fig. 3. Performance of the PVM at different settings. (a) Different numbers of labeled samples per class. (b) Different numbers of prototypes.

variability, results are based on averages over 30 random repetitions.

Fig. 2(a) shows the variation of training time (which includes the time for k-means clustering in step 1 of Algorithms 1 and 2) with n. As can be seen, the time scales linearly with n, which agrees with our analysis in Section IV-A. Fig. 2(b) shows the classification error on the unlabeled samples. Note that although the number of prototypes is fixed, the accuracy remains fairly stable with increasing sample size. This agrees with our intuition that the number of prototypes required for building an accurate model depends largely on the data distribution and class boundary, but not directly on the sample size.

2) Variation With the Number of Labeled Samples: Next, we vary the number of labeled samples used for training, as $l = \{100, 200, 300, \ldots, 1000\}$. The number of unlabeled samples is then $u = n - l$ with $n = 29\,270$. Fig. 3(a) shows the resultant variation of the classification error (on the unlabeled samples). As can be seen, similar to the other semisupervised learning models, the accuracy increases when more labeled information is available. The improvement in accuracy is particularly prominent when the number of labeled samples is small. This agrees with the observation in [9] that the labeled samples reduce the error exponentially fast.

3) Variation With the Number of Prototypes: Finally, Fig. 3(b) examines the variation of the classification error with the number of prototypes m, while fixing the number of labeled samples per class at 50, and the sample size at $n = 29\,720$. As can be seen, using more prototypes leads to better performance, though the improvement diminishes when m is beyond a certain value. In this experiment, using $m = 200$, which is less than 1% of the whole data set size, already suffices to give a good performance.

B. Comparison With the State-of-the-Art

In this section, we compare both versions of PVM: PVM(sqr), using the square loss (Section IV-A1), and PVM(hinge), using the hinge loss (Section IV-A2), with the supervised learner SVM (using only the labeled samples) and a number of state-of-the-art semisupervised learning algorithms, including:

1) *LGC:* learning with LGC [45];
2) *LapRLS:* Laplacian regularized least squares [4] (Section II-A1);
3) *NYS-LGC:* acceleration of LGC using Nyström low-rank approximation [8] (Section II-B1);
4) *LapSVMp:* primal LapSVM [4] trained with preconditioned conjugate gradient [24] (Section II-B2);
5) *Nonparametric function induction (NFI):* fast nonparametric function induction [13] (Section II-B3);
6) *Anchor*[5]: the method based on anchor graph [23] (Section II-B4).

TABLE II
DATA SETS USED IN THE EXPERIMENT. l IS THE NUMBER OF LABELED SAMPLES, u IS THE NUMBER OF UNLABELED SAMPLES, AND m IS THE NUMBER OF PROTOTYPES

DATA	DIM	#CLASSES	l	u	m
G241C	241	2	100	1,400	150
G241D	241	2	100	1,400	150
DIGIT1	241	2	100	1,400	150
USPS	241	2	100	1,400	150
COIL_2	241	2	100	1,400	150
COIL	241	6	300	1,200	150
BCI	117	2	100	300	40
TEXT	11,960	2	100	1,400	150
SPLICE	60	2	100	900	100
SEGMENT	29	7	350	1,960	231
DNA	180	3	150	3,036	200
SVMGD1A	4	2	100	6,989	200
USPS-FULL	256	10	500	6,791	200
SATIMAGE	36	6	300	6,135	200

All codes are in MATLAB and run on an Intel T2400 1.83-GHz laptop with 1-GB memory.

1) Setup: We use a variety of binary and multiclass benchmark data sets from [10][6] and the LIBSVM data archive.[7] The number of labeled samples per class is 50 (except for the two-moon data, which has only one labeled sample per class). A summary of the data sets is shown in Table II.

For PVM, the number of prototypes is chosen as $m = 0.1n$ when $n \leq 3000$; and $m = 200$ otherwise. The same number of samples is used in forming the subset for NFI and NYS-LGC. Other parameters of the various algorithms are chosen by fivefold cross-validation.[8] For example, the β parameter of the Gaussian kernel is chosen from $\{2^{-5}\beta_0, 2^{-4}\beta_0, \ldots, 2^4\beta_0, 2^5\beta_0\}$, where β_0 is the reciprocal of average distance between the samples. The regularization parameters in LapRLS are chosen from $\{10^{-6}, 10^{-5}, \ldots, 1, 10\}$; while those for the other algorithms are from $\{10^{-3}, 10^{-2}, \ldots, 10^4, 10^5\}$. For PVM, we simply set the regularization parameter $C_2 = 0$ and only tune C_1. To reduce statistical variability, results are based on averages over 30 random selections of the labeled samples.

[4]The MATLAB code is from http://sourceforge.net/projects/lapsvmp/

[5]The MATLAB code is from http://www.ee.columbia.edu/ln/dvmm/downloads/WeiGraphConstructCode/dlform.htm

[6]http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html

[7]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[8]In each of the 5 repetitions, the semisupervised model is trained using 4/5 of the labeled data and the unlabeled data, and is validated on the remaining 1/5 of the labeled data. The parameter that yields the lowest average validation error is selected. Using this parameter, the model is retrained with all the labeled and unlabeled data.

TABLE III

CLASSIFICATION ERRORS (%) ON THE UNLABELED DATA OBTAINED BY THE DIFFERENT ALGORITHMS. THE BEST RESULTS AND THOSE THAT ARE NOT STATISTICALLY WORSE (ACCORDING TO THE TWO-SAMPLE ONE-TAILED t-TEST, WITH A p-VALUE OF 0.05) ARE IN BOLD. NOTE THAT, LGC AND LAPRLS CANNOT BE RUN ON THE LARGE DATA SETS OF SVMGD1A, USPS-FULL, AND SATIMAGE

	(SUPERVISED)	(SEMI-SUPERVISED)							
DATA	SVM	LGC	LAPRLS	LAPSVMP	NYS-LGC	NFI	ANCHOR	PVM(SQR)	PVM(HINGE)
G241C	22.34±1.41	**21.92±1.90**	**22.02±1.73**	25.82±1.75	24.19±3.06	28.07±2.71	35.15±1.45	24.50±2.49	23.21±1.95
G241D	25.14±2.03	28.10±3.27	**22.36±1.78**	25.89±1.12	30.98±4.22	30.82±5.36	36.61±2.07	25.15±2.58	24.85±2.70
DIGIT1	4.70±1.09	5.74±1.09	5.74±1.50	**1.84±0.37**	6.68±1.63	9.83±1.43	3.56±0.45	4.18±1.17	3.72±1.07
USPS	9.30±2.81	**4.57±1.27**	6.11±0.63	5.23±2.34	9.72±1.82	5.49±0.78	15.90±3.41	5.29±0.73	6.35±1.33
$COIL_2$	11.17±0.91	14.37±3.63	10.83±1.94	12.74±2.16	16.90±3.00	13.98±2.48	**8.11±1.64**	11.69±2.47	14.85±2.36
COIL	12.21±1.81	12.38±1.99	21.17±1.71	**7.79±0.74**	18.75±1.48	30.93±6.22	12.41±0.67	13.41±1.29	12.26±1.01
BCI	35.17±3.29	44.43±2.28	**29.16±3.56**	46.23±3.16	45.45±2.62	45.67±2.61	48.37±2.61	33.59±3.01	31.65±2.86
TEXT	24.35±2.14	**23.09±1.43**	23.99±1.58	**22.75±1.30**	34.40±3.63	32.54±2.66	27.28±1.01	30.4±4.46	26.29±2.58
SPLICE	24.33±1.13	22.85±1.47	**19.78±2.41**	33.76±2.53	30.56±3.16	34.56±1.97	29.11±0.97	23.47±1.59	25.32±2.48
SEGMENT	9.91±1.06	**8.97±1.20**	9.58±1.73	12.15±2.13	13.58±1.95	15.71±2.15	13.75±1.58	10.15±1.21	**9.06±1.15**
DNA	15.88±1.17	27.31±2.58	17.72±1.29	15.27±0.41	29.53±2.12	43.38±3.94	28.60±1.81	15.87±1.44	**14.19±1.28**
SVMGD1A	5.85±1.65	–	–	**4.23±0.40**	6.32±1.88	14.21±2.92	**4.54±0.62**	5.24±1.09	6.08±1.55
USPS-FULL	6.39±0.50	–	–	**4.16±0.13**	17.68±1.57	14.43±0.89	7.43±0.29	7.35±0.62	5.88±0.91
SATIMAGE	14.59±0.71	–	–	**13.81±0.22**	16.36±0.64	19.27±3.74	14.80±0.56	14.64±0.50	**13.27±0.53**

TABLE IV

TOTAL TRAINING AND TESTING TIME (S) OF THE DIFFERENT ALGORITHMS

DATA	SVM	LGC	LAPRLS	LAPSVMP	NYS-LGC	NFI	ANCHOR	PVM(SQR)	PVM(HINGE)
G241C	0.54	140.84	129.86	33.47	0.86	0.48	11.75	3.30	3.19
G241D	0.55	129.78	142.65	11.95	0.84	0.49	12.33	3.31	3.16
DIGIT1	0.57	140.51	131.08	7.72	0.84	0.48	12.79	3.31	3.15
USPS	0.48	139.23	131.59	21.35	0.74	0.47	12.89	3.28	3.14
$COIL_2$	0.57	151.36	120.48	14.57	0.87	0.48	11.51	3.26	3.47
COIL	1.58	146.92	115.22	31.54	0.79	0.49	13.01	3.35	3.51
BCI	0.08	3.08	1.94	1.12	0.53	0.22	5.89	0.71	1.09
TEXT	25.6	139.67	216.37	59.27	9.14	13.26	47.54	30.24	34.24
SPLICE	1.69	1622.51	1439.51	33.12	2.49	0.83	23.71	4.87	4.24
SEGMENT	4.51	1319.67	1389.80	69.71	1.96	31.67	7.15	6.13	8.81
DNA	2.85	1566.91	1463.75	368.91	3.07	1.22	52.91	8.92	7.57
SVMGD1A	2.71	–	–	1129.44	3.22	1.66	69.21	8.06	5.38
USPS-FULL	19.46	–	–	8920.13	3.96	2.87	125.70	22.48	28.21
SATIMAGE	8.90	–	–	2671.56	3.34	2.57	189.56	11.56	14.32

2) Results: Table III shows the classification errors on the unlabeled samples, and Table IV shows the total training and testing time. The following observations can be made.

1) In general, the semisupervised learning algorithms are more accurate than the baseline SVM. In particular, based on the paired student-t test, the proposed PVM(hinge) outperforms SVM on the DIGIT1, USPS, BCI, SEGMENT, DNA, USPS-FULL, and SATIMAGE data sets at the 95% confidence level. It has a comparable performance as the SVM on the G241C, G241D, SPLICE, SVMGD1A data sets, and is only outperformed by the SVM on the two data sets of COIL2 and TEXT.
2) LGC, LapRLS, and LapSVMp do not employ any approximation to scale up training. Hence, they are usually the most accurate, but also computationally most expensive. Indeed, recall that LGC and LapRLS require $O(n^2)$ space. Hence, they cannot be run on the three largest data sets (svmgd1a, usps-full, and satimage). Moreover, though LapSVMp is faster than LapRLS and LGC, it is still computationally expensive on the large data sets as its time complexity is quadratic in n.
3) LGC-NYS and NFI are computationally very efficient but not as accurate, especially on the more difficult data sets of TEXT and DNA. As discussed in Section II-B.3, we speculate that it is because NFI cannot enforce smoothness of the function on the unlabeled samples not belonging to the basis set [13].
4) The anchor method is more efficient than LGC, LapRLS, and LapSVMp. However, since it needs to compute the combination coefficients for each sample using QP, its complexity can be huge for large data sets. As can be observed from Table VI, its time consumption is at least several times larger than that of PVM. The larger the data set, the larger the difference.
5) The accuracies of both PVMs are comparable with LGC, LapRLS, and LapSVMp; while their speeds can be several hundred times faster. On larger data sets, this speedup ratio is expected to be more significant. Overall, PVM achieves a good balance between speed and accuracy.

We are also interested in training the full semisupervised learning algorithms (LapRLS and LapSVMp) using only a small subset of representative points (such as cluster centers) as unlabeled data. Results and discussions are in Table VII. As can be observed, using representative points in such a way

TABLE V
CLASSIFICATION ERRORS (%) ON THE UNLABELED DATA OBTAINED BY CB-LAPRLS, CB-LAPSVMP, AND PVM

DATA	CB-LAPRLS	CB-LAPSVMP	PVM (SQR)	PVM (HINGE)
G241C	27.14±2.15	28.32±2.91	24.50±2.49	**23.21±1.95**
G241D	28.11±1.78	27.57±1.43	25.15±2.58	**24.85±2.70**
DIGIT1	6.78±1.91	6.84±1.72	4.18±1.17	**3.72±1.07**
USPS	19.67±3.46	19.45±2.25	**5.29±0.73**	6.35±1.33
$COIL_2$	14.98±2.15	17.16±2.81	**11.69±2.47**	14.85±2.36
COIL	21.17±1.71	12.39±1.45	13.41±1.29	**12.26±1.01**
BCI	**29.16±3.56**	46.23±3.16	33.59±3.01	31.65±2.86
TEXT	27.18±2.77	**25.31±1.91**	30.4±4.46	26.29±2.58
SPLICE	**22.35±3.48**	28.82±3.97	23.47±1.59	25.32±2.48
SEGMENT	9.19±1.56	11.21±1.81	10.15±1.21	**9.06±1.15**
DNA	16.57±1.41	15.87±0.44	15.87±1.44	**14.19±1.28**
SVMGD1A	7.71±0.56	6.92±0.55	**5.24±1.09**	6.08±1.55
USPS-FULL	14.15±3.41	11.15±2.17	7.35±0.62	**5.88±0.91**
SATIMAGE	17.85±2.15	15.34±1.66	14.64±0.50	**13.27±0.53**

TABLE VI
TOTAL TRAINING AND TESTING TIME (S) OF CB-LAPRLS, CB-LAPSVMP, AND PVM

DATA	CB-LAPRLS	CB-LAPSVMP	PVM (SQR)	PVM (HINGE)
G241C	10.15	0.47	3.30	3.19
G241D	12.11	0.65	3.31	3.16
DIGIT1	12.54	0.72	3.31	3.15
USPS	13.76	0.35	3.28	3.14
$COIL_2$	11.48	0.57	3.26	3.47
COIL	115.22	0.54	3.35	3.51
BCI	1.15	0.22	0.71	1.09
TEXT	62.15	28.35	30.24	34.24
SPLICE	6.51	3.42	4.87	4.24
SEGMENT	12.17	5.31	6.13	8.81
DNA	15.16	7.41	8.92	7.57
SVMGD1A	23.94	5.58	8.06	5.38
USPS-FULL	57.14	19.85	22.48	28.21
SATIMAGE	39.58	11.29	11.56	14.32

can improve the computational efficiency of the algorithm but lead to less accurate results.

C. Learning With Only the Cluster Centroids

In the PVM, the prototypes are selected from the centroids of k-means clustering. Recall that both the LapRLS and LapSVM (trained in either the primal or dual) have large computational complexities. Hence, one may consider using a computationally less expensive version that replaces the whole unlabeled set with the set of k-means cluster centroids. On the other hand, all the labeled samples are still used, as they are usually much fewer in number. In this section, we compare the performance of these cluster-based (CB) versions (denoted CB-LapRLS and CB-LapSVMp) with PVM.

Tables V and VI show the comparison in terms of the classification error and time consumption, respectively. For easy reference, results for the PVM are also copied from Tables III and IV. As can be seen, though CB-LapRLS and CB-LapSVMp are much faster than the original LapRLS and LapSVMp, their accuracies are often much inferior because of the loss of unlabeled data information. In contrast, the PVM adopts the representative points (cluster centers) in a systematic way for approximations of the whole kernel matrix and the decision function, thus yielding a much better performance in general.

D. Comparison With ℓ_1-Penalized Sparse Models

In this section, we compare the sparse models produced by PVM and ℓ_1-regularization. For ℓ_1-regularization, we use the kernel regressor, which involves the optimization problem

$$\min_{\boldsymbol{\alpha}} \ \|\mathbf{y}_l - \mathbf{K}_l\boldsymbol{\alpha}\|^2 + \lambda_1(\mathbf{K}\boldsymbol{\alpha})^\top \mathbf{S}(\mathbf{K}\boldsymbol{\alpha}) + \lambda_2\|\boldsymbol{\alpha}\|_1 \qquad (31)$$

where $\mathbf{K}\boldsymbol{\alpha}$ (for some $\boldsymbol{\alpha} \in \mathbb{R}^n$) is the vector of predictions on all n labeled and unlabeled samples, and $\mathbf{K}_l \in \mathbb{R}^{l\times n}$ is the kernel submatrix. The $(\mathbf{K}\boldsymbol{\alpha})^\top \mathbf{S}(\mathbf{K}\boldsymbol{\alpha})$ regularization term encourages smoothness over the data manifold, and the ℓ_1-regularizer $\|\boldsymbol{\alpha}\|_1$ encourages sparseness of the coefficient vector $\boldsymbol{\alpha}$.

In the following, we compare the performance of PVM and the ℓ_1-penalized formulation in (31) at different sparsity levels. Note that how the sparsity is achieved by the PVM is quite different from that of (31). The former is obtained by extracting a highly compact and representative set of prototypes, from which the structure of the kernel matrix can be faithfully preserved. Hence, for the PVM, sparsity can be explicitly controlled by specifying the number of prototypes. On the other hand, (31) enforces sparsity by penalizing via the ℓ_1-norm. There is no known algorithm that can produce a model with exactly the desired number of nonzero coefficients. Instead, we can only try different regularization parameter settings to obtain a model with approximately the desired size.[9] This parallel the comparison between VQ-based codebook selection and sparse-coding-based codebook selection in dictionary learning [12]. Moreover, to allow (31) to be used on large data sets, low-rank approximation is first applied on the kernel matrix $\mathbf{K}$. In the sequel, the method will be denoted L1-LRK.

The classification errors on the unlabeled data versus sparsity for the two algorithms are reported in Fig. 4. As can be seen, at the same sparsity level, PVM often has better accuracy than L1-LRK. This demonstrates the efficacy of PVM in building highly efficient, parsimonious, and accurate models in the semisupervised learning setting. In particular, PVM tends to perform better than L1-LRK when the data are high dimensional or have a lot of classes.

Table VII compares the time consumption of PVM with L1-LRK. Here, we have chosen the regularization parameters such that the number of nonzero coefficients in the obtained model is around 10% of the sample size. As can be seen, PVM is much more efficient. This is because for the L1-LRK, to transform the problem (31) to a standard LASSO-type problem, $O(n^2)$ space and $O(n^3)$ time are needed.

[9] In the experiments, λ_1 and λ_2 are selected from $\{10^{-4}, 10^{-3}, \ldots, 10^1, 10^2\}$.

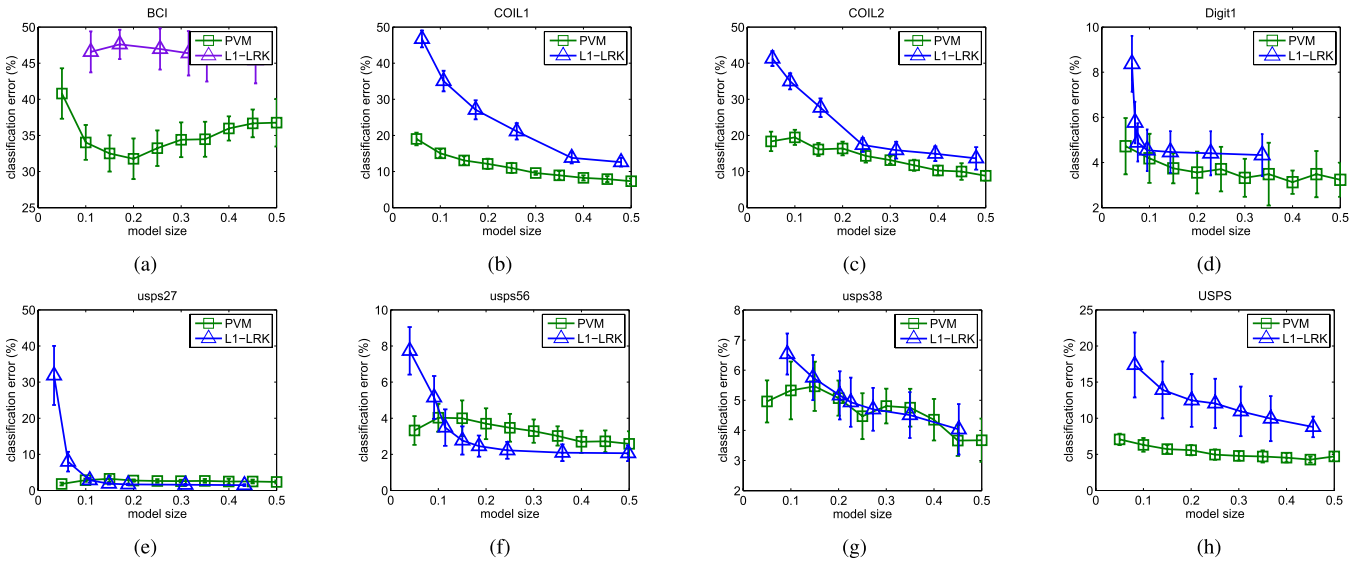


Fig. 4. Performance of PVM and the ℓ_1-penalized formulation at different levels of model sparsity. The x-axis is the ratio of nonzero coefficients in the predictive model to the sample size. (a) BCI. (b) COIL. (c) COIL_2. (d) Digit1. (e) USPS-2v7. (f) USPS-5v6. (g) USPS-3v8. and (h) USPS-FULL.

TABLE VII
COMPARISON OF THE TOTAL TRAINING AND TESTING TIME (S) BETWEEN PVM AND L_1-PENALIZED SPARSE MODEL

	BCI	COIL	COIL_2	DIGIT1	USPS-2v7	USPS-5v6	USPS-3v8	USPS-FULL
L1-LRK	178.5	169.1	135.7	165.4	135.0	144.1	108.5	-
PVM (SQUARED)	0.71	3.35	3.26	3.31	4.12	3.98	2.73	8.06

VI. CONCLUSION

In this paper, we proposed the PVM to scale up graph-based semisupervised learning. Using the prototypes to approximate the graph-based regularizer as well as the predictive model, we can drastically reduce the problem size while at the same time guarantee smoothness of the prediction over the data manifold. Experiments on a number of real-world data sets demonstrate that the PVM has appealing scaling behavior (linear in sample size) and competitive performance.

In the future, we will study various extensions of the PVM. For example, we will consider other kernels (besides the Gaussian kernel) and other label reconstruction schemes that can take the local geometrical structure of the samples into account. Moreover, the current prototype selection scheme (k-means clustering) treats the labeled and unlabeled data as equally important. One possibility is to incorporate the label information, and extend the selection scheme to a weighted version so that different importance-based weightings are assigned based on prior knowledge. A similar idea has been successfully used in the context of low-rank approximation [1]. Moreover, we will further study the relationships between the labeled samples, unlabeled samples, and the prototypes.

REFERENCES

[1] F. Bach and M. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, Aug. 2005, pp. 33–40.

[2] S. Baluja, "Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data," in *Proc. Adv. NIPS*, vol. 11. 1999, pp. 854–860.

[3] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. NIPS*, vol. 14. 2002, pp. 585–591.

[4] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, no. 11, pp. 2399–2434, 2006.

[5] S. Ben-David, T. Lu, and D. Pal, "Does unlabeled data provably help? Worst-case analysis of the sample complexity of semi-supervised learning," in *Proc. 21st Annu. Conf. Learn. Theory*, 2008, pp. 33–44.

[6] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proc. 18th Int. Conf. Mach. Learn.*, San Francisco, CA, USA, 2001, pp. 19–26.

[7] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, New York, NY, USA, Jul. 1998, pp. 209–214.

[8] G. Camps-Valls, T. V. B. Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens. E*, vol. 45, no. 10, pp. 3044–3054, Oct. 2007.

[9] V. Castelli and T. Cover, "On the exponential value of labeled samples," *Pattern Recognit. Lett.*, vol. 16, no. 1, pp. 105–111, 1995.

[10] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[11] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, Jan. 2005.

[12] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proc. 28th Int. Conf. Mach. Learn.*, Bellevue, WA, USA, Jun. 2011, pp. 921–928.

[13] O. Delalleau, Y. Bengio, and N. L. Roux, "Efficient non-parametric function induction in semi-supervised learning," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, 2005, pp. 96–103.

[14] P. Drineas, R. Kannan, and M. Mahoney, "Fast Monte Carlo algorithms for matrices II: Computing a low rank approximation to a matrix," *SIAM J. Comput.*, vol. 36, no. 1, pp. 158–183, 2006.

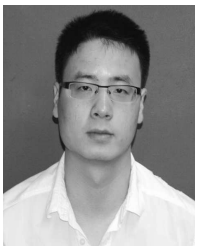
[15] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning," *J. Mach. Learn. Res.*, vol. 6, pp. 2153–2175, Dec. 2005.

[16] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, Feb. 2004.

[17] A. Frieze, R. Kannan, and S. Vempala, "Fast Monte-Carlo algorithms for finding low-rank approximations," in *Proc. 39th Annu. Symp. Found. Comput. Sci.*, Palo Alto, CA, USA, 1998, pp. 370–378.
[18] J. Goldberger and S. Roweis, "Hierarchical clustering of a mixture model," in *Proc. Adv. NIPS*, vol. 17. 2005, pp. 505–512.
[19] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th Int. Conf. Mach. Learn.*, San Francisco, CA, USA, 1999, pp. 200–209.
[20] J. Lafferty, X. Zhu, and Y. Liu, "Kernel conditional random fields: Representation and clique selection," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, AB, Canada, Jul. 2004, p. 64.
[21] C.-H. Lee, S. Wang, F. Jiao, D. Schuurmans, and D. Greiner, "Learning to model spatial dependency: Semi-supervised discriminative random fields," in *Proc. Adv. NIPS*, vol. 19. Cambridge, MA, USA, 2007.
[22] L. Chen, I. W. Tsang, and D. Xu, "Laplacian embedded regression for scalable manifold regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 902–915, Jun. 2012.
[23] W. Liu, J. He, and S. F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, Jun. 2010, pp. 679–686.
[24] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *J. Mach. Learn. Res.*, vol. 12, no. 3, pp. 1149–1184, 2011.
[25] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. NIPS*, vol. 14. 2001, pp. 849–856.
[26] Y. Huang, D. Xu, and F. Nie, "Semi-supervised dimension reduction using trace ratio criterion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 3, pp. 519–526, Mar. 2012.
[27] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Mach. Learn.*, vol. 39, nos. 2–3, pp. 103–134, 2000.
[28] P. Rigollet, "Generalization error bounds in semi-supervised classification under the cluster assumption," *J. Mach. Learn. Res.*, vol. 8, pp. 1369–1392, Dec. 2007.
[29] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[30] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., 1994.
[31] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[32] A. Singh, R. D. Nowak, and X. Zhu, "Unlabeled data: Now it helps, now it doesn't," in *Proc. Adv. NIPS*, 2008, pp. 1513–1520.
[33] R. G. F. Soares, H. Chen, and X. Yao, "Semisupervised classification with cluster regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 11, pp. 1779–1792, Nov. 2012.
[34] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[35] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist., Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.
[36] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, Dec. 2005.
[37] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Adv. NIPS*, vol. 13. 2001, pp. 682–688.
[38] Z. Xu, I. King, M. R.-T. Lyu, and R. Jin, "Discriminative semi-supervised feature selection via manifold regularization," *IEEE Trans. Neural Netw.*, vol. 21, no. 7, pp. 1033–1047, Jul. 2010.
[39] D. Yarowsky, "Unsupervised word-sense disambiguation rivaling supervised methods," in *Proc. 33rd Annu. Meet. Assoc. Comput. Linguistics*, Stroudsburg, PA, USA, 1995, pp. 189–196.
[40] K. Zhang and J. T. Kwok, "Improved Nyström low rank approximation and error analysis," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jun. 2008, pp. 1232–1239.
[41] K. Zhang and J. T. Kwok, "Clustered Nyström method for large scale manifold learning and dimension reduction," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1576–1587, Oct. 2010.
[42] K. Zhang, J. T. Kwok, and B. Parvin, "Prototype vector machine for large scale semi-supervised learning," in *Proc. 26th Int. Conf. Mach. Learn.*, Montreal, QC, Canada, Jun. 2009, pp. 1233–1240.
[43] K. Zhang, L. Lan, Z. Wang, and F. Moerchen, "Scaling up kernel SVM on limited resources: A low-rank linearization approach," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2012, pp. 1425–1434.
[44] K. Zhang, V. Zheng, Q. Wang, J. Kwok, Q. Yang, and I. Marsic, "Covariate shift in Hilbert space: A solution via surrogate kernels," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, Jun. 2013, pp. 388–395.
[45] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Adv. NIPS*, vol. 16. 2003, pp. 321–328.
[46] Y. Wang, S. Chen, and Z.-H. Zhou, "New semi-supervised classification method based on modified cluster assumption," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 5, pp. 689–702, May 2012.
[47] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1530, 2008.
[48] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington, DC, USA, Aug. 2003, pp. 912–919.
[49] X. Zhu and J. Lafferty, "Harmonic mixtures: Combining mixture models and graph-based methods for inductive and scalable semi-supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, Aug. 2005, pp. 1052–1059.

Kai Zhang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 2008.

He joined the Life Science Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA. He is currently a Research Staff Member with NEC Laboratories America, Inc., Princeton, NJ, USA. His current research interests include large scale machine learning, matrix approximation, and applications in bioinformatics, time series, and complex networks.



Liang Lan received the B.S. degree in bioinformatics from the Huazhong University of Science and Technology, Wuhan, China, and the Ph.D. degree in computer and information sciences from Temple University, Philadelphia, PA, USA, in 2007 and 2012, respectively.

He is currently a Researcher with the Huawei Noah's Ark Laboratory, Hong Kong. His current research interests include data mining and machine learning.

James T. Kwok received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 1996.

He is a Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His current research interests include kernel methods, machine learning, and pattern recognition.

Prof. Kwok has been a Program Co-Chair for a number of international conferences, and served as an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2006 to 2012. He is currently an Associate Editor of the *Neurocomputing* journal. He was a recipient of the 2004 IEEE Outstanding Paper Award, and the Second Class Award in Natural Sciences from the Ministry of Education, China, in 2008.

Slobodan Vucetic received the B.S. and M.S. degrees in electrical engineering from the University of Novi Sad, Novi Sad, Serbia, and the Ph.D. degree in electrical engineering from Washington State University, Pullman, WA, USA, in 1994, 1997, and 2001, respectively.

He is currently an Associate Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. His current research interests include data mining and machine learning.

Bahram Parvin is a Principal Scientist with the Lawrence Berkeley National Laboratory, Berkeley, CA, USA, and has an adjunct appointment with the Electrical Engineering Department, University of California, Berkeley. His laboratory focuses on the development of machine learning techniques for realization of pathway pathology, and elucidating molecular signatures of aberrant morphogenesis in normal and engineered matrices.

Prof. Parvin was the General Chair of the IEEE International Symposium on Biomedical Imaging: From Nano to Macro, in 2013. He is also a Steering Committee Member for the IEEE Bioimaging and Signal Processing and the IEEE Bioengineering and Health Care.