Prototype Vector Machine for Large Scale Semi-Supervised Learning

Kai Zhang KAI_ZHANG@LBL.GOV
Life Science Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA, 94720

James T. Kwok JAMESK@CSE.UST.HK
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

Bahram Parvin B_PARVIN@LBL.GOV
Life Science Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA, 94720

Abstract

Practical data mining rarely falls exactly into the supervised learning scenario. Rather, the growing amount of unlabeled data poses a big challenge to large-scale semi-supervised learning (SSL). We note that the computational intensiveness of graph-based SSL arises largely from the manifold or graph regularization, which in turn lead to large models that are difficult to handle. To alleviate this, we proposed the *prototype vector machine* (PVM), a highly scalable, graph-based algorithm for large-scale SSL. Our key innovation is the use of "prototypes vectors" for efficient approximation on both the graph-based regularizer and model representation. The choice of prototypes are grounded upon two important criteria: they not only perform effective low-rank approximation of the kernel matrix, but also span a model suffering the minimum information loss compared with the complete model. We demonstrate encouraging performance and appealing scaling properties of the PVM on a number of machine learning benchmark data sets.

1. Introduction

In practical data analysis and mining, large-scale problems rarely fall exactly into the classic supervised learning scenario. In most cases, what dominates is the huge amount of unlabeled data acquired with an ever growing rate in the various scientific domains, while the fraction of labeled samples will always be tiny due to the expensive and tedious human intervention. Examples include but are not limited to document classification, gene functionality analysis, and phenotypic response prediction based on structural similarities. These situations make urgent the development of semi-supervised learning framework, where the unlabeled sample are used together with the labeled ones to improve the performance of classifiers. On the other hand, more challengingly, the semi-supervised techniques will have to scale themselves well to accommodate large data at hand to play practical roles.

Although semi-supervised learning has drawn considerable interest from the learning community, current methods have not yet handled large amount of data. Many graph-based methods to semi-supervised learning (Zhu et al., 2003; Belkin et al., 2006; Zhou et al., 2003) scale cubically with the data size, where manipulation of the kernel matrix defined on all the labeled and unlabeled samples is the primary bottleneck. As to the transductive SVM (Joachims, 1999), the formulation requires solving a combinatorial optimization that is computationally expensive, and relaxations into convex formulations (Bie & Cristianini, 2004; Xu et al., 2008) usually lead to semi-definite programs that scale at least $O(n^4)$. In (Lawrence & Jordan, 2003), the binary SSL problem is formulated in the Gaussian process framework, which needs to be scaled up by approximate, sparse GP-solvers.

Here, we summarize some advances in scaling up semi-supervised learning algorithms. For the transductive SVM, Fung and Mangasarian (2001) considered a sequential optimization procedure where a SVM is trained using the labeled samples while simultaneously assigning the unlabeled points to one of two classes to maximize the margin. The algorithm shows clear improvement in prediction performance, but it is de-

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

signed for the linear case and a special SVM with L_1-regularization. In (Collobert et al., 2006), a large-scale training method, based on the Concave-Convex procedure, is proposed for TSVMs. It iteratively optimizes the non-convex function through a series of conventional convex SVM training. The complexity is the best among the existing TSVMs. Empirically, it scales quadratically with the sample size but can still be expensive for large problems.

For graph-based methods, Zhu and Lafferty (2005) proposed an elegant framework that combines the generative mixture model with graph-based regularization, overcoming the limitations of scalability and non-inductive inference. However, as the authors commented, the curse of dimensionality could prevent a mixture model from fitting the data well. In (Delalleau et al., 2005), a nonparametric function induction framework is proposed which makes prediction based on a subset of samples. The graph regularization is also approximated by using only the selected subset of samples and their connections with the rest of the data. However, approximation of the graph regularizer is achieved by ignoring the inter-connections among the majority part of the data: those outside the pre-chosen small subset. This can weaken the smoothness of the resultant predictor. In (Gustavo et al., 2007), the Nyström method is used to numerically approximate the large matrix inverse used in (Zhou et al., 2003), which leads to an efficient algorithm. It is transductive and does not handle new testing points.

Note that graph-based methods can usually be accelerated to scale as $O(m^2 n)$ (Delalleau et al., 2005; Gustavo et al., 2007), where m is the number of samples selected, and n is the data size. Since $m \ll n$, this is almost linear in n. They are more efficient than transductive SVMs that scale at least quadratically (Collobert et al., 2006). Therefore, we will focus on graph-based methods, another advantage of which is their ease in handling multi-class problems. Our key observation is that the computational intensiveness of graph-based semi-supervised learning arises largely from the regularization term. On the one hand, this often requires manipulation of the $n \times n$ kernel matrix (such as multiplication or inverse) which is impractical for large problems. On the other hand, as shown in the representer theorem (Belkin et al., 2006), when the complete graph structure is used as the regularizer (such as the graph Laplacian), the resultant model will span over both the labeled and unlabeled samples, potentially leading to a very large model that is difficult to handle.

To alleviate this problem, we propose to enforce the graph-based regularization in a more economical way by extracting a set of "prototypes vectors" from the data, in order to improve both the scalability and model simplicity for semi-supervised learning. On the one hand, these prototype vectors are applied in the low-rank approximation of the kernel matrix, which recovers the global graph structure crucial for regularization. On the other hand, with a certain information theoretic measure, we require the prototypes to span a prediction model with as much representation power as the complete model obtained from the representer theorem for graph-regularized structural risk minimization. Since the prototypes have a small size, they can be used to reformulate the optimization problem in SSL for improved scalability.

Our algorithm overcomes the difficulty with current practices in scaling up graph-based semi-supervised learning. It is non-parametric and makes no specific assumption about the sample distribution. On the other hand, the graph-based regularizer is enforced reliably based on a systematic approximation of the kernel matrix, guaranteeing the smoothness of the resultant model with regard to the overall manifold structure. Our algorithm differs from a direct application of low-rank approximation technique in speeding up existing graph-based SSL algorithms. Instead, it combines the insight from low-rank kernel matrix approximation with the learning of inductive models.

The rest of the paper is organized as follows. In Section 2, we give a brief introduction on graph-based semi-supervised learning algorithms. In Section 3, we discuss two important roles played by the prototypes, i.e., approximating the graph-based regularizer and simplifying the model parametrization. We also show that with the use of the Gaussian kernel, the two criteria lead to a simple and unified prototype selection scheme via k-means clustering. In Section 4, we demonstrate how to use the prototypes to reduce the problem size in SSL, under both the L_2 and hinge loss functions. In Section 5, we make empirical evaluations on a number of algorithms to demonstrate the performance of the proposed prototype vector machine. The last section gives some concluding remarks.

2. Related Work

Suppose that we are given a set of l labeled samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$ and u unlabeled samples $\{\mathbf{x}_i\}_{i=l+1}^{l+u}$, where $n = l + u$, $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{y}_i \in \mathbb{R}^c$ is the label vector such that if $\mathbf{x}_i$ belongs to the kth class ($1 \leq k \leq c$), the kth entry in $\mathbf{y}_i$ is 1 and all the other entries are 0's. A number of graph-based semi-supervised learning algorithms (Zhou et al., 2003; Zhu et al., 2003) can be placed under the following optimization framework

$$\min_{\mathbf{f}=[\mathbf{f}_l^\top \mathbf{f}_u^\top]^\top \in \mathbb{R}^{n\times c}} \operatorname{tr}(\mathbf{f}^\top \boldsymbol{\mathcal{S}} \mathbf{f}) + C_1 L(\mathbf{f}_l, \mathbf{Y}_l) + C_2 \|\mathbf{f}_u\|_F^2. \quad (1)$$

Here $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{n\times n}$ is the graph-based regularization matrix, $\mathbf{Y}_l \in \mathbb{R}^{l\times c}$ is the class assignment matrix for the labeled points, $\mathbf{f} \in \mathbb{R}^{n\times c}$ is the predicted class labels for all the data points, which is divided into the labeled part $\mathbf{f}_l \in \mathbb{R}^{l\times c}$ and the unlabeled part $\mathbf{f}_u \in \mathbb{R}^{u\times c}$, C_1 and C_2 are positive regularization parameters, and L is the loss function. The first term in (1) enforces the smoothness of the predicted labels $\mathbf{f}$ with regard to the manifold structure of the graph. The second term requires that the prediction $\mathbf{f}$ should be consistent with the known class labels, and the third term enforces some kind of regularization on predicting the unlabeled samples.

Usually $\boldsymbol{\mathcal{S}}$ is chosen as the graph Laplacian matrix $\boldsymbol{\mathcal{S}} = \mathbf{D} - \mathbf{K}$, where $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of the graph (or the kernel matrix) and $\mathbf{D} = \operatorname{diag}(\mathbf{K}\mathbf{1}_n)$ is the degree matrix; or sometimes the normalized graph Laplacian matrix $\boldsymbol{\mathcal{S}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$. For example, in (Zhou et al., 2003), $\boldsymbol{\mathcal{S}}$ is chosen as the normalized Laplacian, and the two regularization parameters C_1 and C_2 are chosen the same; in (Zhu et al., 2003), $\boldsymbol{\mathcal{S}}$ is chosen as the graph Laplacian, C_1 approaches infinity, and $C_2 = 0$. Note that these algorithms are transductive and do not handle new test points. In (Belkin et al., 2006), it is shown that in order to optimize

$$\min \sum_{i=1}^{l} L\left(f(\mathbf{x}_i), \mathbf{y}_i\right) + \gamma_A \|f\|_{\mathcal{K}} + \gamma_I \|f\|_{\mathcal{G}}, \quad (2)$$

where $L(\cdot,\cdot)$ is an empirical loss function, γ_A and γ_I are regularization parameters to control the RKHS-norm ($\|\cdot\|_{\mathcal{K}}$) and graph-based smoothness ($\|\cdot\|_{\mathcal{G}}$) of the model f, respectively, the model will expand over all the labeled and unlabeled points in the form of

$$f(\mathbf{x}) = \sum_{i=1}^{l+u} \alpha_i K(\mathbf{x}, \mathbf{x}_i), \quad (3)$$

where α_i's are the multipliers. By plugging (3) into (2), the authors extended standard regularization algorithms to their Laplacian-regularized versions. For example, the Laplacian regularized least-squares minimizes

$$\min \|\mathbf{Y}_l - \mathbf{K}\boldsymbol{\alpha}\|_F^2 + \gamma_A \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} + \gamma_I \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\mathcal{S}} \mathbf{K} \boldsymbol{\alpha},$$

where $\boldsymbol{\alpha}$ is the vector of α_i's. This can be deemed as the inductive counterpart of (1) with a certain choice of the regularization parameters γ_A and γ_I.

3. Approximations via Prototypes

This section introduces the basic idea of using prototypes for solving SSL problems. By prototypes, we mean a small set of points in the input space that can be used as a replacement of the original data to obtain efficient yet accurate solutions. More specifically, in this paper we will need two kinds of prototypes: (i) low-rank-approximation prototype for the graph-based regularization (Section 3.1); (ii) label-reconstruction prototype for model representation (Section 3.2).

3.1. Graph Regularization

Graph-based regularization in semi-supervised learning is usually in the form of the quadratic term $\mathbf{f}^\top \boldsymbol{\mathcal{S}} \mathbf{f}$, where $\boldsymbol{\mathcal{S}}$ is usually chosen as the (normalized) graph Laplacian matrix, and $\mathbf{f}$ is the realization of the target function (the class labels) on the whole data set. The computation of $\boldsymbol{\mathcal{S}}$ is based on all the labeled and unlabeled samples, which not only requires storing the $n \times n$ kernel matrix, but may also involve expensive operations such as the matrix inverse in solving the optimization problem. As a result, it will dominate the overall training complexity.

Note that the kernel matrix encountered in practice usually have low numerical-rank compared with the matrix size (Williams & Seeger, 2000). This opens up the possibility of reducing the computational burdens of enforcing the graph-based regularization by performing low-rank approximation on the kernel matrix $\mathbf{K}$. Theoretically, the optimal rank-k approximation of a kernel matrix $\mathbf{K}$ is provided by its eigenvectors associated with the dominant k eigenvalues. Therefore, efficient alternatives have to be adopted. In this paper, we use the Nyström method (Williams & Seeger, 2001), a well-known sampling-based approach that approximates the kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ by

$$\mathbf{K} \approx \mathbf{K}_{nm} \mathbf{K}_{mm}^{-1} \mathbf{K}_{nm}^\top. \quad (4)$$

Here, $\mathbf{K}_{nm}$ denotes the sub-matrix of $\mathbf{K}$ with m columns, and $\mathbf{K}_{mm}$ is the sub-matrix that corresponds to the intersection of the chosen columns and rows.

The Nyström low-rank approximation can be interpreted as a kind of prototype approximation, i.e., choosing a subset of "prototype" points $\{\mathbf{u}_i\}_{i=1}^m$ from the given data $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, computing the kernel matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$ on the prototype points

$$\mathbf{W}_{ij} = K(\mathbf{u}_i, \mathbf{u}_j), \ 1 \le i, j \le m,$$

the cross-similarity matrix $\mathbf{E} \in \mathbb{R}^{n\times m}$ between the whole data $\mathbf{X}$ and the prototype points,

$$\mathbf{E}_{ij} = K(\mathbf{x}_i, \mathbf{u}_j), \ 1 \le i \le n, 1 \le j \le m,$$

and then apply $\mathbf{K} \approx \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top$. Therefore, we call $\mathbf{u}_i$'s low-rank-approximation prototypes. With this approximation, the graph-based regularization term can be approximated as

$$\mathbf{f}^\top \boldsymbol{\mathcal{S}} \mathbf{f} \quad \approx \quad \mathbf{f}^\top (\tilde{\mathbf{D}} - \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top)\mathbf{f},$$

where $\tilde{\mathbf{D}} = \mathrm{diag}(\mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top \mathbf{1}_n)$ and $\mathbf{1}_n \in \mathbb{R}^n$ is the vector of all 1's. Usually, the prototype points are simply chosen as a random subset of the samples (Williams & Seeger, 2001). Recently, Zhang and Kwok (2008) showed that the Nyström low-rank approximation error is related to the encoding errors of the landmark points in summarizing the data, for a number of commonly used kernels (Gaussian, linear, and polynomial). Based on this error analysis, we will choose the k-means clustering centers as the low-rank approximation prototypes.

3.2. Label Reconstruction

As has been discussed, the inclusion of a graph-based regularizer results in a model that will expand over all the labeled and unlabeled samples (3). This not only leads to slow testing, but in turn causes computations to be huge by applying such model in the structural risk minimization (2) as considered in (Belkin et al., 2006). Actually, in practice a classifier rarely needs the whole basis set. In Figure 1, we illustrate this using a toy data set. We choose a set of prototypes (crosses) from each class, whose labels depend on the classes they are chosen from. To predict the label of a sample $\mathbf{x}$, we compute its similarity with all the prototypes, and linearly combine the prototype labels using the similarities as weights. The decision boundary is shown in blue. We also train a SVM on the full data set and labels, and plot the support vectors (circles) and the decision boundary in green. As can be seen, the two decision boundaries are very similar.

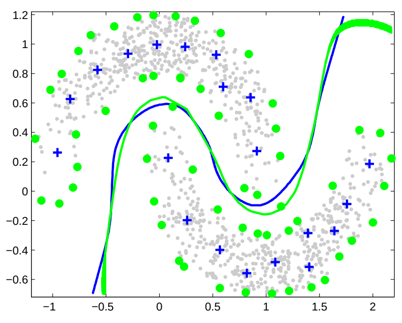


Figure 1. Decision boundaries of a prototype-based model (blue) and a SVM trained on the full data set (green).

This example illustrates the value of prototypes, i.e., a set of properly chosen prototype vectors, together with their labels (have to be learned, though), is supposed to reconstruct the whole landscape of sample labels. To formalize this, we assume that the label of any sample point $\mathbf{x}$ can be recovered by a linear combination of the labels of a set of prototypes $\{\mathbf{v}_i\}_{i=1}^k$

$$g(\mathbf{x}) = \sum_{i=1}^{k} \mathbf{f}_i K(\mathbf{x}, \mathbf{v}_i), \tag{5}$$

where $\mathbf{f}_i$'s are the estimated labels for $\mathbf{x}_i$. Therefore, we call $\mathbf{v}_i$'s the label-reconstruction prototypes. In matrix form, this can be written as

$$\mathbf{f} = \mathbf{H}\mathbf{f}_{\mathbf{v}}, \tag{6}$$

where $\mathbf{f}$ is the predicted labels on all the samples, $\mathbf{f}_{\mathbf{v}} \in \mathbb{R}^{k\times c}$ is the label of prototypes $\mathbf{v}_i$'s, and $\mathbf{H} \in \mathbb{R}^{n\times k}$ is the cross-similarity matrix between the whole data $\mathbf{X}$ and the prototype points $\mathbf{v}_i$'s,

$$\mathbf{H}_{ij} \quad = \quad K(\mathbf{x}_i, \mathbf{v}_j), \;\; 1 \le i \le n, 1 \le j \le k.$$

We may also want to normalize $\mathbf{H}$ by dividing each row of $\mathbf{H}$ by the sum of entries in that row. This can prevent too large/small values in predicting the label.

The prototype reconstruction (5) resembles the effect of a nearest neighbor decision rule. Due to the similarity-based weighting scheme, the reconstructed label of $\mathbf{x}$ will be mostly dependent on those prototypes having large similarities with $\mathbf{x}$. Therefore, as long as there are enough prototypes "filling" the space of the input data, the sample label of a point can always be reconstructed reliably by finding prototypes close to it. Our goal is to find a compact set of prototypes that can faithfully reconstruct the labels.

Next, we examine how to choose the label reconstruction prototypes for (5). Note that (5) can be deemed as an approximation to the complete model (3) from the representer theorem of graph-regularized structural risk minimization. In supervised learning, the kernel expansions in the model are all over "support vectors" lying on the class boundaries. However, in semi-supervised scenario, the small amount of labeled data never suffice in delineating the complete class boundary. In this case, the classifier has to be expanded over unlabeled samples as well, many of which could reside far off the class boundaries. Now, our goal is to avoid allowing the model to be expanded over the whole data set from scratch by finding a small set of prototypes that can generate similar decision boundaries. Mathematically, this will require minimizing the distance between the complete model f (3) and the approximate model g (5) as $D\left(\sum_{i=1}^{l+u} \alpha_i K(\mathbf{x}, \mathbf{x}_i), \sum_{i=1}^{m} \beta_i K(\mathbf{x}, \mathbf{v}_i)\right)$, where $D(\cdot,\cdot)$

is a distance between functions, and we rewrite the model g (5) as $g(\mathbf{x}) = \sum_{i=1}^{m} \beta_i K(\mathbf{x}, \mathbf{v}_i)$ because the estimated labels of the prototypes usually take real values instead of $\{\pm 1\}$ (binary) or $\{0, 1\}$ (multi-class).

In practice, the coefficients in the two models are unknown during the training phase, so it is infeasible to directly minimize their distance. However, note that both models are expanded over their respective dictionaries of basis functions: f is spanned by the kernel function $K(\cdot,)$'s over the complete sample set, and g by those over the prototypes. Therefore, the approximate model g is expected to have similar representation power as the complete model f, if the basis in f can be reliably coded by those from g. More specifically, the prototypes in g can be chosen to minimize

$$\mathcal{Q} = \sum_{i=1}^{l+u} \sum_{j=1}^{k} \min D((K(\mathbf{x}, \mathbf{x}_i), K(\mathbf{x}, \mathbf{v}_j)) \,. \qquad (7)$$

This can be imagined as the error of coding each component in f with the best prototype in g.

Obviously, the selection of the prototypes $\mathbf{v}_i$'s in (7) will vary depending on the specific kernel $K(\cdot, \cdot)$ and distance measure $D(\cdot, \cdot)$ adopted. In the following, we consider the case where K is the Gaussian kernel and $D(\cdot, \cdot)$ is the cross-entropy. The cross-entropy is defined as $H(p, q) = H(p) + D_{KL}(p||q)$, where p and q are basis functions from the two models f and g, respectively, $H(p) = -\int p(\mathbf{x}) \log(p(\mathbf{x})) d\mathbf{x}$ is the entropy of p, and $D_{KL}(p||q) = \int p(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) d\mathbf{x}$ is the Kullback-Leibler divergence. Since our variables (the prototypes $\mathbf{v}_i$'s) rest with the basis in g, $H(p)$ will be constant and we only need to consider the KL-divergence part. Using the Gaussian kernel $K(\mathbf{z}_1, \mathbf{z}_2) = \exp\left(-\|\mathbf{z}_1 - \mathbf{z}_2\|^2 / 2h^2\right)$, the KL-divergence can be written as (Goldberger & Roweis, 2005)[1]

$$D_{KL}\left[K(\mathbf{x}, \mathbf{x}_i) || K(\mathbf{x}, \mathbf{v}_j)\right] = \frac{1}{4h^2} \|\mathbf{x}_i - \mathbf{v}_j\|^2. \qquad (8)$$

Then the objective (7) can be simplified as

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

where S_i is the set of those kernels $K(\mathbf{x}_j, \cdot)$ in f whose closest prototype in g is $K(\mathbf{v}_i, \cdot)$. As can be seen, interestingly, this is exactly the objective of k-means clustering, and the prototypes $\mathbf{v}_i$'s in (5) can be chosen as the k-means cluster centers as well. For other types of kernels (linear or polynomial), the cross-entropy is no longer suitable and new distance measures will be investigated in the future.

[1]This is a special case of the problem considered in (Goldberger & Roweis, 2005), where the model g has varying bandwidths and the components in f are weighted.

4. Prototype Vector Machine

We have used two kinds of prototypes, the low-rank-approximation prototypes $\{\mathbf{u}_i\}_{i=1}^{m}$ and the label-reconstruction prototypes $\{\mathbf{v}_i\}_{i=1}^{k}$ as discussed in Section 3. Since we require the prototype sizes (m and k) to be much smaller than the sample size (n), the prototypes can be used to reformulate problem (1) for a more efficient optimization. More specifically, the low-rank-approximation prototypes avoid computing the $n \times n$ kernel matrix in enforcing the graph-based regularization. On the other hand, the label-reconstruction prototypes avoid optimizing the labels of all the samples $\mathbf{f} \in \mathbb{R}^{n \times c}$, by just focusing on the labels of the k prototypes $\mathbf{f}_\mathbf{v} \in \mathbb{R}^{k \times c}$. Suppose that we have chosen the prototype sets $\mathbf{u}_i$'s and $\mathbf{v}_i$'s. Then by plugging (5) and (6) into (1) and considering the use of different loss functions, we can formulate the prototype vector machine in the following.

4.1. L_2 Loss Function

By choosing the loss function L (1) as the L_2-norm (or the Frobenius norm in the multi-class case), the objective can be written as

$$\min_{\mathbf{f}_\mathbf{v} \in \mathbb{R}^{m \times k}} \quad \operatorname{tr}\left((\mathbf{H}\mathbf{f}_\mathbf{v})^\top \boldsymbol{\mathcal{S}} (\mathbf{H}\mathbf{f}_\mathbf{v})\right) + C_1 \|\mathbf{H}_l \mathbf{f}_\mathbf{v} - \mathbf{Y}_l\|_F^2 \\ + C_2 \|\mathbf{H}_u \mathbf{f}_\mathbf{v}\|_F^2.$$

Here, $\mathbf{H}_l \in \mathbb{R}^{l \times k}$ and $\mathbf{H}_u \in \mathbb{R}^{u \times k}$ are the rows in $\mathbf{H}$ that correspond to the labeled and unlabeled points, respectively, and $\boldsymbol{\mathcal{S}}$ is as defined in (5). By setting the derivative w.r.t. $\mathbf{f}_\mathbf{v}$ to zero, we obtain

$$\mathbf{f}_\mathbf{v}^* = \left(\mathbf{H}^\top \boldsymbol{\mathcal{S}} \mathbf{H} + C_1 \mathbf{H}_l^\top \mathbf{H}_l + C_2 \mathbf{H}_u^\top \mathbf{H}_u\right)^{-1} \mathbf{E}_l^\top \mathbf{Y}_l. \quad (9)$$

Note that $\mathbf{H}^\top \boldsymbol{\mathcal{S}} \mathbf{H} = \mathbf{H}^\top \tilde{\mathbf{D}} \mathbf{H} - (\mathbf{H}^\top \mathbf{E}) \mathbf{W}^{-1} (\mathbf{H}^\top \mathbf{E})^\top$ can be computed in $O((m+k) \cdot nk)$ time, therefore the overall complexity is bounded by $O((m+k)^2 \cdot n)$.

4.2. Hinge Loss Function

Here, we consider the binary classification case where the labels $\mathbf{Y}_l \in \{\pm 1\}^{l \times 1}$, and L is chosen as the hinge loss used in the SVM. Let $\mathbf{e}_k$ be the transpose of the kth row in $\mathbf{H}_l$, i.e., $\mathbf{H}_l = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_l]^\top$, and

$$\mathbf{A} = \mathbf{H}^\top \boldsymbol{\mathcal{S}} \mathbf{H} + C_2 \mathbf{H}_u^\top \mathbf{H}_u. \qquad (10)$$

Then the optimization problem (1) can be written as

$$\begin{aligned} \min_{\mathbf{f_v} \in R^{m\times 1}} \quad & \frac{1}{2}\mathbf{f_v}^\top \mathbf{A} \mathbf{f_v} + C_1 \sum_{i=1}^{l} \xi_i \\ \text{s.t.} \quad & y_i \mathbf{e}_i^\top \mathbf{f_v} \geq 1 - \xi_i, \;\; \xi_i \geq 0. \end{aligned}$$

The first term is the regularization part, and the second term is the hinge loss induced by approximating the labels of $\mathbf{X}_l$ with the reconstructed ones ($\mathbf{H}_l \mathbf{f_v}$). The Lagrangian function 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 the derivative w.r.t. $\mathbf{f_v}$ and ξ_i's to zero,

$$\begin{aligned} \mathbf{A}\mathbf{f_v} &= \sum_{i=1}^{l} \beta_i y_i \mathbf{e}_i, \qquad (11) \\ C_1 &= \beta_i + \gamma_i. \end{aligned}$$

Plugging back into the Lagrangian, we have

$$\max -\frac{1}{2}\left(\sum_{i=1}^{l} \beta_i y_i \mathbf{e}_i^\top\right) \mathbf{A}^{-1} \left(\sum_{i=1}^{l} \beta_i y_i \mathbf{e}_i\right) + \sum_{i=1}^{l} \beta_i.$$

The above optimization problem can be written as

$$\begin{aligned} \max \quad & -\frac{1}{2}\boldsymbol{\beta}^\top \mathbf{Q} \boldsymbol{\beta} + \mathbf{1}_l^\top \boldsymbol{\beta} \\ \text{s.t.} \quad & 0 \leq \beta_i \leq C_1, \; i = 1, 2, \ldots, l, \end{aligned}$$

where $\mathbf{Q} = \mathbf{H}_l \mathbf{A}^{-1} \mathbf{H}_l^\top \odot \mathbf{Y}_l \mathbf{Y}_l^\top$, and $\odot$ is the element-wise product. This is a convex QP problem with a similar form as the standard SVM with only l variables. In practice, we will first remove the negative eigenvalues of $\mathbf{A}$ (10) and then compute matrix $\mathbf{Q}$. Since $\mathbf{A}$ can be deemed as an approximation to the psd matrix, its negative eigenvalues are all of vanishing magnitudes and can be removed safely. Then the resultant matrix $\mathbf{Q}$ is guaranteed to be positive semi-definite. After solving the QP, labels of the prototypes $\mathbf{v}_i$'s can be recovered using the KKT condition (11), as $\mathbf{f}_\mathbf{v}^* = \mathbf{A}^{-1}\mathbf{H}_l^\top \boldsymbol{\beta} \odot \mathbf{Y}_l$. Note that the matrix $\mathbf{A}$ (10) can be computed efficiently (similar to $\boldsymbol{\mathcal{S}}$). On the other hand, modern SVM implementations typically have an empirical time complexity that scales between $O(l)$ and $O(l^{2.3})$ for solving the QP (Platt, 1999). So the overall complexity is still bounded by $O((m+k)^2 \cdot n)$ since the labeled set is usually small. After obtaining $\mathbf{f}_\mathbf{v}^*$, the predicted labels of $\mathbf{X}_u$ are computed by $\mathbf{f}_u = \mathbf{H}_u \mathbf{f}_\mathbf{v}^*$.

5. Experiments

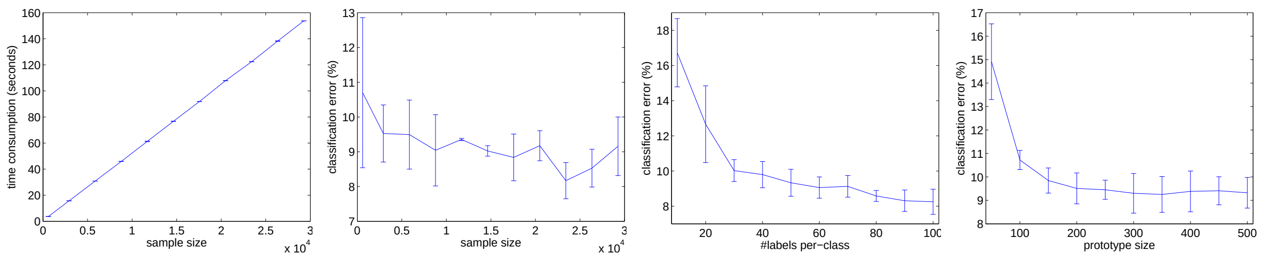
This section compares the following algorithms: (1) LGC: local and global consistency method (Zhou et al., 2003); (2) Lap-RLS: Laplacian regularized least squares (Belkin et al., 2006); (3) NYS-LGC: Direct acceleration of LGC using Nyström low-rank approximation (use the Woodbury formula plus Nyström method to accelerate the matrix inverse in LGC); (4) NFI: nonparametric function induction (Delalleau et al., 2005); (5) PVM(1): our method using the L_2 loss; (6) PVM(2): our method using the hinge loss (only for binary classification). All codes are in matlab and run on a Intel(R) T2400 1.83GHz Laptop with 1GB memory.

We use 15 benchmark data sets on binary and multi-class semi-supervised classification, including: data sets from (Olivier Chapelle & Zien, 2006)[2] and the libsvm webset[3] (training and testing sets are merged for some libsvm data sets). The number of labels/per-class is chosen as 50 for most tasks, except the usps3589 data (20 labels) and the 2-moon data (1 labels). We used the Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 \cdot b\right)$ in our experiments. As discussed in Section 3, this will lead to a unified prototype selection scheme, i.e., we will perform the k-means and use the cluster centers for both types of prototypes in our algorithm. The prototype size m (or subset size in methods 3,4) is chosen as follows: $m = 20$ for 2-moon; $m = 10\% n$ if $n \leq 3000$; $m = 200$ if $n > 3000$. Parameters of the different algorithms are chosen based on cross-validation as follows. For the kernel parameter b, we choose among $b_0 \times \{2^{-5} - 2^5\}$ where b_0 is the reciprocal of the averaged distance between data points; for regularization parameters, those in method 2 are chosen among $\{10^{-6} - 10^1\}$; while the other algorithms will choose from $\{10^{-3} - 10^5\}$; and in PVM(1) and PVM(2) we simply set the regularization parameter $C_2 = 0$ and only tune C_1. For all the algorithms, we randomly pick the labeled samples for 30 times and report the averaged error and time consumption.

Results are given in Table 1 (LGC and LAP-RLS need $O(n^2)$ space and cannot handle the largest 3 data sets). Among the 15 data sets, PVM(1) and PVM(2) give the most accurate solution among 6 data sets. For other data sets their performances are close to the full SSL algorithms (LGC, LAP-RLS), but the speed is 1 to 3 orders of magnitude faster. On the other hand, PVM outperforms NYS-LGC that directly uses low-rank approximation to numerically scale up the LGC. The NFI method is very efficient but is less accurate on some data sets. We speculate that it is due to the ignorance of the graph-based regularization on a large portion of the data (Delalleau et al., 2005).

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

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



(a) time vs sample size. (b) error vs sample size. (c) error vs #labels. (d) error vs prototype-size.

Figure 2. Case study of PVM on the USPS digits 3,5,6,8,9.

We perform a case study of PVM(1) on a 5-class classification task of mnist digit 3,5,6,8,9, with $n = 29270$ and $d = 784$. First, we gradually increase the sample size, and for each subset size we perform PVM 30 times while fixing $m = 200$ and record the averaged time consumption and error. As can be seen, PVM scales exactly linearly with the sample size (Figure 2(a)), and the accuracy is stable as the sample size increases (Figure 2(b)). This coincides with our intuition that the prototype size needed depends largely on the distribution of the data and the class boundaries, but not on the data size. From Figure 2(c), we can see that the accuracy of PVM increases steadily with more and more labeled points. Figure 2(d) examines the error of PVM over the prototype size m (while fixing #labels). As can be seen, larger prototype size will lead to better performance. However, the improvement diminishes when m is beyond a certain value. In this example, $m = 200$ already suffices to give a good performance, which is less than 1% of the whole data size.

6. Conclusion

In this paper, we proposed the prototype vector machine to scale up the graph-based semi-supervised learning. By using the prototype vectors to approximate the graph-based regularizer as well as the complete prediction model, we can drastically reduce the problem size and apply our algorithm to large-scale real-world problems. The PVM demonstrates appealing scaling behavior (linear in sample size) and competitive performance. In the future, we will explore different ways to extend it. For example, we will consider different label reconstruction schemes based on local data geometry. We can also derive the prototype version for the Laplacian-regularized algorithms in (Belkin et al., 2006). Finally, our current prototype selection treats the labeled and unlabeled data as equally important. This can be extended to a weighted version that assigns importance-based weighting based on some prior knowledge.

Acknowledgement

This research was supported in part by U.S. Department of Energy, Office of Biological and Environmental Research, under contract number DE-AC02-05CH11231 with the University of California. James T. Kwok acknowledged support by the Research Grants Council of the Hong Kong Special Administrative Region under grant 614508.

References

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Bie, T. D., & Cristianini, N. (2004). Convex methods for transduction. *Advances in Neural Information Processing Systems 16* (pp. 73–80).

Collobert, R., Sinz, F., Weston, J., Bottou, L., & Joachims, T. (2006). Large scale transductive SVMs. *Journal of Machine Learning Research*, *7*, 1687–1712.

Delalleau, O., Bengio, Y., & Roux, N. (2005). Efficient non-parametric function induction in semi-supervised learning. *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics* (pp. 96–103).

Fung, G., & Mangasarian, O. L. (2001). Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, *15*, 29–44.

Goldberger, J., & Roweis, S. (2005). Hierarchical clustering of a mixture model. *Advances in Neural Information Processing Systems 17* (pp. 505–512).

Gustavo, C., Marsheva, T., & Zhou, D. (2007). Semi-supervised graph-based hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *45*, 3044–3054.

Table 1. Classification errors (%) and time consumptions (seconds) of different algorithms.

Data(#cls)	LGC	LAP-RLS	NYS-LGC	NFI	PVM(1)	PVM(2)
g241c(2)	**21.92±1.90**	**22.02±1.73**	24.19±3.06	28.07±2.71	24.50±2.49	23.21±1.95
g241d(2)	28.10±3.27	**22.36±1.78**	30.98±4.22	30.82±5.36	25.15±2.58	24.85±2.70
digit1(2)	5.74±1.09	5.74±1.50	6.68±1.63	9.83±1.43	4.18±1.17	**3.72±1.07**
USPS(2)	**4.57±1.27**	6.11±0.63	9.72±1.82	5.49±0.78	5.29±0.73	6.35±1.33
coil$_2$(2)	14.37±3.63	**10.83±1.94**	16.90±3.00	13.98±2.48	11.69±2.47	14.85±2.36
coil(6)	**12.38±1.99**	21.17±1.71	18.75±1.48	30.93±6.22	13.41±1.29	–
BCI(2)	44.43±2.28	**29.16±3.56**	45.45±2.62	45.67±2.61	33.59±3.01	31.65±2.86
Text(2)	**23.09±1.43**	23.99±1.58	34.40±3.63	32.54±2.66	30.4±4.46	26.29±2.58
2-moon(2)	0.95±0.71	7.87±4.74	1.12±0.84	16.72±6.45	0.092±0.028	**0.07±0.06**
usps3589(4)	**2.46±0.73**	4.54±1.42	6.89±1.76	7.14±1.63	3.66±1.18	–
splice(2)	22.85±1.47	**19.78±2.41**	30.56±3.16	34.56±1.97	23.47±1.59	25.32±2.48
dna(3)	27.31±2.58	17.72±1.29	29.53±2.12	43.38±3.94	**15.87±1.44**	–
svmgd1a(2)	–	–	6.32±1.88	14.21±2.92	**5.24±1.09**	6.08±1.55
usps-full(10)	–	–	17.68±1.57	14.43±0.89	**7.35±0.62**	–
satimage(6)	–	–	16.36±0.64	19.27±3.74	**14.97±0.50**	–
data(size/dim)						
g241c(1500/241)	140.84±0.21	129.86±0.69	0.86±0.031	0.48±0.27	3.30±0.44	3.19±0.049
g241d(1500/241)	129.78±0.14	142.65±1.46	0.84±0.029	0.49±0.034	3.31±0.032	3.16±0.056
digit1(1500/241)	140.51±3.03	131.08±2.01	0.84±0.27	0.48±0.12	3.31±0.42	3.15±0.064
USPS(1500/241)	139.23±0.36	131.59±1.12	0.74±0.026	0.47±0.033	3.28±0.34	3.14±0.073
coil$_2$(1500/241)	151.36±0.059	120.48±0.92	0.87±0.22	0.48±0.074	3.26±0.024	3.47±0.075
coil(1500/241)	146.92±0.43	115.22±1.54	0.79±0.081	0.49±0.032	3.35±0.057	–
BCI(400/117)	3.08±0.027	1.94±0.057	0.53±0.023	0.22±0.029	0.71±0.23	1.09±0.049
Text(1500/11960)	139.67±0.095	216.37±0.59	9.14±0.12	13.26±0.13	30.24±0.41	34.24±0.28
2-moon(1000/2)	49.76±0.11	16.11±0.026	0.026±0.017	0.24±0.031	0.083±0.025	0.21±0.081
usps3589(719/64)	13.94±0.045	13.13±0.01	0.15±0.019	0.086±0.032	0.37±0.022	–
splice(3175/60)	1622.5± 11.7	1439.5±10.5	2.49±0.78	0.83±0.051	4.87±0.091	4.24±0.11
dna(3186/180)	1566.91±7.45	1463.75±8.13	3.07±0.75	1.22±0.044	8.92±0.014	–
svmgd1a(7089/4)	–	–	3.22±0.91	1.66±0.052	8.06±0.032	5.38±0.17
usps-full(7291/256)	–	–	3.96±0.064	2.87±0.064	22.48±0.15	–
satimage(6435/36)	–	–	3.34±1.11	2.57±0.16	11.56±0.19	–

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *International Conference on Machine Learning* (pp. 200–209). Morgan Kaufmann.

Lawrence, N., & Jordan, M. (2003). Semi-supervised learning via gaussian processes. *Advances in Neural Information Processing Systems 14*.

Olivier Chapelle, B. S., & Zien, A. (2006). *Semi-supervised learning*. MIT.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods – Support vector learning*, 185–208.

Williams, C., & Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. *Proceedings of the 17th International Conference on Machine Learning* (pp. 1159–1166).

Williams, C., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems 13* (pp. 682 – 688).

Xu, Z., Jin, R., Zhu, J., King, I., & Lyu, M. (2008). Efficient convex relaxation for transductive support vector machine. In *Advances in neural information processing systems 20*, 1641–1648.

Zhang, K., & Kwok, J. (2008). Improved Nyström low rank approximation and error analysis. *Proceedings of the 25th international conference on Machine learning* (pp. 1232–1239).

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2003). Learning with local and global consistency. *Neural Information Processing Systems 16* (pp. 321–328).

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *In ICML* (pp. 912–919).

Zhu, X., & Lafferty, J. (2005). Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. *The 22nd International Conference on Machine Learning* (pp. 1052–1059).