Laplacian Embedded Regression for Scalable Manifold Regularization

Lin Chen, Ivor W. Tsang, and Dong Xu

***Abstract*—Semi-supervised learning (SSL), as a powerful tool to learn from a limited number of labeled data and a large number of unlabeled data, has been attracting increasing attention in the machine learning community. In particular, the manifold regularization framework has laid solid theoretical foundations for a large family of SSL algorithms, such as Laplacian support vector machine (LapSVM) and Laplacian regularized least squares (LapRLS). However, most of these algorithms are limited to small scale problems due to the high computational cost of the matrix inversion operation involved in the optimization problem. In this paper, we propose a novel framework called Laplacian embedded regression by introducing an intermediate decision variable into the manifold regularization framework. By using ϵ-insensitive loss, we obtain the Laplacian embedded support vector regression (LapESVR) algorithm, which inherits the sparse solution from SVR. Also, we derive Laplacian embedded RLS (LapERLS) corresponding to RLS under the proposed framework. Both LapESVR and LapERLS posses a simpler form of a transformed kernel, which is the summation of the original kernel and a graph kernel that captures the manifold structure. The benefits of the transformed kernel are two-fold: 1) we can deal with the original kernel matrix and the graph Laplacian matrix in the graph kernel separately and 2) if the graph Laplacian matrix is sparse, we only need to perform the inverse operation for a sparse matrix, which is much more efficient when compared with that for a dense one. Inspired by kernel principal component analysis, we further propose to project the introduced decision variable into a subspace spanned by a few eigenvectors of the graph Laplacian matrix in order to better reflect the data manifold, as well as accelerate the calculation of the graph kernel, allowing our methods to efficiently and effectively cope with large scale SSL problems. Extensive experiments on both toy and real world data sets show the effectiveness and scalability of the proposed framework.**

***Index Terms*—Laplacian embedding, large scale semi-supervised learning, manifold regularization.**

I. Introduction

LABELED data are expensive to obtain in terms of both monetary cost and labeling time in many applications. On the other hand, a large amount of unlabeled data is often cheaper to collect in many domains, such as computer vision, information retrieval, natural language processing, and speech recognition. Semi-supervised learning (SSL), which exploits the prior knowledge from the unlabeled data to improve classification performance, has attracted considerable attention in recent years [1]–[10]. There are two typical SSL approaches: learning with the cluster assumption [11], [12] and learning with the manifold assumption [13].

The cluster assumption requires that data within the same cluster are more likely to have the same label. In other words, if two data points are connected with an edge passing through a high density region, they are likely to share the same class label or the decision boundary should lie in some low density regions. Many algorithms have been developed based on the cluster assumption. The most prominent example is the transductive support vector machine (TSVM), which was originally proposed by Vapnik in [11]. Since the objective function of TSVM is non-convex, its performance is easily influenced by the initialization. To achieve a better solution, numerous strategies have been proposed, such as S^3VM [14], SVM$^{\text{light}}$ [15], ∇SVM [16], concave convex procedure-TSVM [17], and S^2LS-SVM [4].

On the other hand, the manifold assumption states that each class lies on a separate low-dimensional manifold embedded in a higher dimensional space. Many forms of real-world data, such as handwritten digits [18], [19], faces [20], webpages [20]–[22], medical data (e.g., cancer) [8], speech data [13], GPS data [23], and WiFi signals [24], have been demonstrated to exhibit such kind of intrinsic geometric structure, which is commonly approximated by a weighted graph. Thus, the label dependencies among data points can be captured by this graph, leading to a branch of graph-based SSL methods, including local and global consistency [22], Gaussian fields and harmonic functions [21], and spectral graph transducer [25]. However, most of them are designed for transductive learning, and thus cannot be easily extended to out-of-sample data points.

In this paper, we focus on the manifold regularization framework proposed in [13], which is one of the most successful SSL works due to the convexity of its optimization problem, its out-of-sample prediction, and solid theoretical foundations. In this framework, the underlying geometric structure of the marginal distribution is estimated from the unlabeled data and is incorporated into reproducing kernel Hilbert space (RKHS) to form a data-dependent regularization term (*a.k.a. manifold regularizer*). This regularizer ensures that the learned decision function is smooth along the manifold. Thus, kernel methods can then be integrated into this RKHS, leading to Laplacian SVM (LapSVM) and Laplacian regularized least

Manuscript received August 4, 2011; revised December 9, 2011; accepted February 9, 2012. Date of publication April 18, 2012; date of current version May 10, 2012. This research is supported by the Singapore National Research Foundation under its Interactive & Digital Media (IDM) Public Sector R&D Funding Initiative and administered by the IDM Programme Office.

The authors are with the School of Computer Engineering, Nanyang Technological University, 639798, Singapore (e-mail: chen0631@ntu.edu.sg; ivortsang@ntu.edu.sg; dongxu@ntu.edu.sg).

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.2012.2190420

squares (LapRLS) (see Section II), which have shown the state-of-the-art performance [26]. Recent work [27] also shows that the *manifold regularizer* is useful for the feature selection.

Despite their great successes, manifold regularization methods are limited to only small scale problems because of the matrix inversion operation involved in the dual problem [13]. Sindhwani *et al.* [28] proposed a linear manifold regularization that is able to deal with a large number of data with a few (or large but highly sparse) features. However, the flexibility of the linear model has to be compromised, and the linear model may not work well when the classification task is complicated. Considering the great success of nonlinear kernel methods in many applications [29], it is still more attractive to develop nonlinear kernel methods. Recently, Melacci *et al.* [20] proposed to solve the primal LapSVM (PLapSVM) problem by using the preconditioned conjugate gradient (PCG) to avoid the matrix inversion operation in the dual problem. Moreover, Tsang *et al.* [30] proposed to accelerate the training process by changing the form of *manifold regularizer*.

As aforementioned, the algorithms (LapSVM and LapRLS) derived from the manifold regularization framework are limited to small scale problems because of the matrix inversion operation involved in the dual problem, and another problem is that this matrix is always dense. Taking Laplacian support vector regression (LapSVR)[1] for example, it is very expensive to compute $(I + \mu LK)^{-1}$ in its dual form, because it usually requires $O(n^3)$ time complexity, where $L \in \mathcal{R}^{n\times n}$ is the graph Laplacian matrix with n being the total number of labeled and unlabeled data, $K \in \mathcal{R}^{n\times n}$ is the kernel matrix, $I \in \mathcal{R}^{n\times n}$ is the identity matrix, $\mu > 0$ is a scalar (see Section II-B for more details). In addition, while L is often sparse, $(I + \mu LK)$ is always dense because K is dense. Thus, it makes the problem difficult to simplify.

To tackle this issue, we first introduce an intermediate decision variable into the manifold regularization framework and obtain the novel framework of Laplacian embedded regression. The algorithms based on this novel framework possess a simpler transformed kernel, which is of the form: $\tilde{Q} = K + (\Lambda + \mu L)^{-1}$, where Λ is an $n \times n$ diagonal matrix (see Section III-A for more details). In such a case, the computational cost mainly depends on $(\Lambda+\mu L)^{-1}$. When $(\Lambda+\mu L)$ is sparse, there are efficient ways to compute $(\Lambda+\mu L)^{-1}$ [31]. However, in some cases (such as the iterated graph Laplacian matrix L^p used in the *manifold regularizer*), it is impractical to compute $(\Lambda+\mu L^p)^{-1}$ as L^p is no longer sparse for $p > 1$, and in turn it leads to a huge computational and memory cost for large data sets. Inspired by kernel principal component analysis (PCA), we further propose to project the introduced intermediate variable into a subspace spanned by a few eigenvectors of the graph Laplacian matrix in order to better reflect the data manifold. Then, we obtain a simpler transformed kernel that can be efficiently computed. Specifically, we can compute the transformed kernel matrix in $O(n_{ev}(n+n_{ev}+\bar{k})n)$ time complexity and $O((d + \bar{k} + n_{ev})n)$ space complexity, where $n_{ev} \ll n$ is the number of eigenvectors spanning the subspace, and $\bar{k}$ is the average number of non-zero elements in each row of L and d is the feature dimension. Because of the above characteristics, the algorithms Laplacian embedded SVR (LapESVR) and Laplacian embedded RLS (LapERLS) derived based on our framework are more efficient for very large scale problems. Moreover, the QP problem in LapESVR can be readily solved by using the QP solver in LIBSVM [32], which adopts sequential minimal optimization (SMO) without requiring the precomputed kernel matrix. Therefore, our method LapESVR is scalable to large scale data sets.

[1]LapSVR and LapSVM are similar, with the only difference being in the variable domain of their quadratic programming (QP) problems. For simplicity, we take LapSVR as an example to show the disadvantages.

The main contributions of this paper are outlined as follows.

1) We propose a novel framework, called Laplacian embedded regression, which extends the manifold regularization framework for large scale SSL. Based on our proposed framework, we develop two algorithms LapESVR and LapERLS. We further improve the efficiency for the calculation of the transformed kernel in the proposed framework by exploiting only the first few eigenvectors of the graph Laplacian matrix such that our methods are scalable to large scale problems.
2) We conduct extensive experiments on both toy and real world data sets, and the experimental results show that our methods achieve comparable or even better prediction performance when compared with the state-of-the-art algorithms, and at the same time our methods have much better scalability.

The rest of this paper is organized as follows. In Section II, we briefly review the manifold regularization and highlight its limitations by taking LapSVR as an example. Our proposed Laplacian embedded regularization framework is presented in Section III. We discuss related works in Section IV and the experiments are described in Section V. We conclude this paper in Section VI.

II. Manifold Regularization

In this section, we briefly review the manifold regularization method [13]. Let us first introduce some notations. We define $\mathcal{S} = \{\mathbf{x}_i \in \mathcal{X} \subset \mathcal{R}^d, i = 1, \ldots, l + u\}$ as the set of training samples, which consists of the first l labeled samples with the label of each sample being $y_i \in \{-1, 1\}$ and the remaining are u unlabeled samples. We also denote $n = l + u$ as the total number of training samples and $\mathcal{S} = \mathcal{L} \cup \mathcal{U}$, where $\mathcal{L} = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ is the set of labeled samples drawn according to the joint distribution P defined on $\mathcal{X} \times \mathcal{R}$, and $\mathcal{U} = \{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}\}$ is the set of unlabeled samples drawn according to the marginal distribution $P_{\mathcal{X}}$ of P.

Manifold regularization exploits the geometry of the marginal distribution $P_{\mathcal{X}}$, and assumes the conditional distributions $P(y|\mathbf{x}_1)$ and $P(y|\mathbf{x}_2)$ are similar if the two data points $\mathbf{x}_1, \mathbf{x}_2$ are close to each other in the intrinsic geometry of $P_{\mathcal{X}}$. This constraint is enforced as an *intrinsic regularizer* $\|f\|_I^2$ in the objective function. Given a Mercer kernel K and its induced RKHS $\mathcal{H}_k$, the objective function of manifold regularization is

$$\min_{f\in\mathcal{H}_k} \frac{1}{l}\sum_{i=1}^{l} \ell(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) + \gamma_A \Omega(\|f\|_{\mathcal{H}_k}) + \gamma_I \|f\|_I^2 \quad (1)$$

where $f(\mathbf{x})$ is the decision function, γ_A and γ_I are two parameters balancing different terms, $\|f\|_I^2$ is the *intrinsic regularizer* or *manifold regularizer*, $\Omega(\|f\|_{\mathcal{H}_k})$ is the *ambient regularizer*, and ℓ is the loss function. According to the Representer Theorem, the optimal $f^*(\mathbf{x})$ can be shown to have the form

$$f^*(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + \int_{\mathcal{M}} \alpha(z) k(\mathbf{x}, z) dP_{\mathcal{X}}(z) \quad (2)$$

where $\mathcal{M}$ is the support of the marginal distribution $P_{\mathcal{X}}$, and $\alpha(z)$ is the coefficient corresponding to z.

A. Graph Laplacian

In most applications, $P_{\mathcal{X}}$ is not available. Therefore, the *manifold regularizer* is usually approximated by the graph Laplacian matrix associated with $\mathcal{S}$. Assume the support $\mathcal{M}$ of $P_{\mathcal{X}}$ is a compact sub-manifold, the *manifold regularizer* $\|f\|_I^2$ has the form of $\int_{\mathcal{M}} < \nabla f, \nabla f >$, where ∇ is the gradient of f along the manifold $\mathcal{M}$. Then, the approximation of $\|f\|_I^2$ is

$$\|f\|_I^2 = \sum_{i,j=1}^{l+u} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 W_{ij} = \boldsymbol{f}^\top L \boldsymbol{f} \quad (3)$$

where $\boldsymbol{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{l+u})]^\top$, and W is the affinity matrix defining the similarity between any pair of samples. $L = D - W$ is the graph Laplacian matrix, where D is the diagonal matrix with diagonal elements $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$. The normalized graph Laplacian matrix is defined as $L = D^{-1/2}(D - W)D^{-1/2}$. For the graph construction, a kNN graph is commonly used due to its simplicity and effectiveness [20], [30], [33]. In a kNN graph, the affinity matrix W is calculated as $W_{ij} = exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_g^2)$, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected, and 0 otherwise.

It is worth mentioning that recent theoretical and empirical results [5] demonstrate that the solution of manifold regularization using L is not well posed at the limit of infinite unlabeled data. For example, the solution may degenerate to a constant function with "spikes" near the labeled data points in the case of high-dimensional data. As suggested in [8], one possible way to cope with this problem is to use other forms of the graph Laplacian matrix, such as the iterated graph Laplacian matrix L^p, where $p \in \mathcal{N}$ is the power of the graph Laplacian matrix [28], [34].

B. LapSVR

Taking LapSVR as an example, we instantiate the loss function in (1) as an ϵ-insensitive loss

$$\ell(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) = \begin{cases} |f(\mathbf{x}_i) - y_i| - \epsilon, & \text{if } |f(\mathbf{x}_i) - y_i| \geq \epsilon \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Then we obtain the primal form of LapSVR as

$$\min_{f\in\mathcal{H}_k, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \frac{1}{2}\|f\|_{\mathcal{H}_k}^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) + \frac{\mu}{2}\boldsymbol{f}^\top L\boldsymbol{f} \quad (5)$$

$$\text{s.t.} \quad f(\mathbf{x}_i) - y_i \leq \epsilon + \xi_i$$
$$y_i - f(\mathbf{x}_i) \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0, i = 1, \ldots, l \quad (6)$$

where C and μ are two parameters balancing different terms, $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_l]^\top$ and $\boldsymbol{\xi}^* = [\xi_1^*, \ldots, \xi_l^*]^\top$. According to the Representer Theorem, the optimal f^* of (5) can be written as $f^*(\mathbf{x}) = \sum_{i=1}^{l+u} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$. When the bias term is added, we have $f^*(\mathbf{x}) = \sum_{i=1}^{l+u} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b$. By introducing the Lagrange multipliers β_i and β_i^* into the inequality constraints (6), we arrive at the following dual problem:

$$\min_{\boldsymbol{\beta}, \boldsymbol{\beta}^*} \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\top Q(\boldsymbol{\beta} - \boldsymbol{\beta}^*) \quad (7)$$
$$+ \epsilon\sum_{i=1}^{l}(\beta_i + \beta_i^*) + \sum_{i=1}^{l} y_i(\beta_i - \beta_i^*)$$
$$\text{s.t.} \sum_{i=1}^{l}(\beta_i - \beta_i^*) = 0, \beta_i, \beta_i^* \in [0, C]$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_l]^\top$, $\boldsymbol{\beta}^* = [\beta_1^*, \ldots, \beta_l^*]^\top$ and

$$Q = JK(I + \mu LK)^{-1}J^\top \quad (8)$$

is the transformed kernel, with $K \in \mathcal{R}^{(l+u)\times(l+u)}$ being the kernel matrix over both labeled and unlabeled data, $L \in \mathcal{R}^{(l+u)\times(l+u)}$ being the graph Laplacian matrix, and $J = [I \ \ 0] \in \mathcal{R}^{l\times(l+u)}$. After obtaining the dual variables $\boldsymbol{\beta}$ and $\boldsymbol{\beta}^*$, the final solution is obtained by solving the linear system

$$\boldsymbol{\alpha} = (I + \mu LK)^{-1}J^\top(\boldsymbol{\beta}^* - \boldsymbol{\beta}) \quad (9)$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{l+u}]^\top$. Moreover, the final decision function is written as

$$f(\mathbf{x}) = \sum_{i=1}^{l+u} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b. \quad (10)$$

The calculation of the transformed kernel Q involves a matrix inversion operation with the size of the matrix being $(l+u) \times (l+u)$, and the recovery of the expansion coefficients $\boldsymbol{\alpha}$ also requires solving a linear system of a matrix with the size of $(l+u)$. Both operations require $O((l+u)^3)$ time complexity. This may be impractical for large data sets. Moreover, it is very hard to simplify this problem because L and K are combined together in LK. In next section, we will present our novel formulation of Laplacian embedded regression, which has a much simpler form of transformed kernel and is scalable to large scale problems.

III. Proposed Formulation

In order to solve the aforementioned issues and make the manifold regularization scalable to large scale problems, we propose the following optimization problem by introducing an intermediate decision variable $\boldsymbol{g}$ into the manifold regularization framework in (1)

$$\min_{f\in\mathcal{H}_k, \boldsymbol{g}\in\mathcal{R}^{(l+u)}} \frac{1}{l+u}\sum_{i=1}^{l+u}\ell(\mathbf{x}_i, g_i, f(\mathbf{x}_i)) + \gamma_C\sum_{i=1}^{l}(g_i - y_i)^2$$
$$+\gamma_A\Omega(\|f\|_{\mathcal{H}_k}) + \gamma_I\|\boldsymbol{g}\|_I^2 \quad (11)$$

where $\Omega(\|f\|_{\mathcal{H}_k})$ is the *ambient regularizer*, and $\boldsymbol{g} = [g_1, \ldots, g_{l+u}]^\top$. Note that $\boldsymbol{g}$ plays a similar role as the decision function $f(\mathbf{x})$ but $\boldsymbol{g}$ is not used for final classification.

Essentially, we enforce this intermediate decision variable $\boldsymbol{g}$ to be close to the labels of the labeled data and also to be smooth with respect to the graph manifold. On the other hand, the *ambient regularizer* is another term to regulate the linear/nonlinear prediction function $f(\mathbf{x})$ in a RKHS. As discussed in prior work [7], the prediction function $f(\mathbf{x})$ may not fit well for the graph manifold structure. To relax such a restriction in traditional manifold regularization, the first term in (11) is introduced to penalize the mismatch between the decision variable $\boldsymbol{g}$ for the graph manifold and the prediction function $f(\mathbf{x})$ in the RKHS. According to prior work [7], the introduction of this regularizer helps improve the generalization performance of the traditional manifold regularization. The more interesting part is that the multiplication between K and L (i.e., LK) in the formulation of traditional manifold regularization (e.g., LapSVR) can be decoupled in our formulation by regularizing $\boldsymbol{g}$ instead of $\boldsymbol{f}$. We will derive our solutions to show the advantages of this proposed formulation in the following sections.

Proposition 1: The optimal solution f^* of the optimization problem (11) is of the form

$$f^*(\mathbf{x}) = \sum_{i=1}^{l+u} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \tag{12}$$

with $\alpha_i \in \mathcal{R}, i = 1, , \ldots, l+u$ being the expansion coefficients.

Proof: For any fixed $\boldsymbol{g}$, the optimization problem (11) reduces to

$$\min_{f\in\mathcal{H}_k} \frac{1}{l+u}\sum_{i=1}^{l+u} \ell(\mathbf{x}_i, g_i, f(\mathbf{x}_i)) + \gamma_A \Omega(\|f\|_{\mathcal{H}_k}). \tag{13}$$

According to the Representer Theorem, the solution of the above-reduced problem can be represented as

$$f(\mathbf{x}; \boldsymbol{g}) = \sum_{i=1}^{l+u} \alpha_i(\boldsymbol{g}) k(\mathbf{x}_i, \mathbf{x})$$

where $\alpha_i(\boldsymbol{g})$ is the coefficient α_i with respect to the current values of $\boldsymbol{g}$. It is clear that the optimal solution of (11) can be represented in the form of (12).

A. LapESVR

By instantiating the loss function in (11) as ϵ-insensitive loss in (4), and re-organizing each term and its corresponding parameter according to the formulation of LapSVR in (5), we obtain the following optimization problem:

$$\begin{aligned}\min_{f\in\mathcal{H}_k, \boldsymbol{g}, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad & \frac{1}{2}\|f\|^2_{\mathcal{H}_k} + C\sum_{i=1}^{l+u}(\xi_i + \xi_i^*) \\ & +\frac{1}{2}(\boldsymbol{g}-\mathbf{y})^\top \Lambda(\boldsymbol{g}-\mathbf{y}) + \frac{\mu}{2}\boldsymbol{g}^\top L\boldsymbol{g} \qquad (14)\\ \text{s.t.} \quad & f(\mathbf{x}_i) - g_i \le \epsilon + \xi_i \\ & g_i - f(\mathbf{x}_i) \le \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0, i = 1, \ldots, l+u \qquad (15)\end{aligned}$$

where Λ is a diagonal matrix of trade-off parameters with $\Lambda_{ii} = \lambda$ if $\mathbf{x}_i$ is a labeled data, and $\Lambda_{ii} = 0$ if $\mathbf{x}_i$ is an unlabeled data and $\mathbf{y} = [y_1, \ldots, y_l, 0, \ldots, 0]^\top \in \mathcal{R}^{l+u}$. The *ambient regularizer* is replaced by $\Omega(\|f\|_{\mathcal{H}_k}) = 1/2\|f\|^2_{\mathcal{H}_k}$, and the *manifold regularizer* is replaced by $\boldsymbol{g}^\top L\boldsymbol{g}$. The parameters C, λ and μ control the trade-off of different terms.

According to Proposition 1, we have $f(\mathbf{x}) = \sum_{i=1}^{l+u}\alpha_i k(\mathbf{x}_i, \mathbf{x})$. Again, a bias term may be added, and we have $f(\mathbf{x}) = \sum_{i=1}^{l+u}\alpha_i k(\mathbf{x}_i, \mathbf{x}) + b$. In order to derive its dual formulation, we introduce the Lagrangian multipliers β_i, β_i^*, η_i, and η_i^* for the inequality constraints in (15) and obtain the Lagrangian as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{g}, \boldsymbol{\xi}, \boldsymbol{\xi}^*, b) = & \frac{1}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} + C\mathbf{1}^\top(\boldsymbol{\xi}+\boldsymbol{\xi}^*) \\ & +\frac{1}{2}(\boldsymbol{g}-\mathbf{y})^\top\Lambda(\boldsymbol{g}-\mathbf{y}) + \frac{1}{2}\mu\boldsymbol{g}^\top L\boldsymbol{g} \\ & -\boldsymbol{\beta}^\top(\epsilon\mathbf{1} + \boldsymbol{\xi} + \boldsymbol{g} - K\boldsymbol{\alpha} - b\mathbf{1}) - \boldsymbol{\eta}^\top\boldsymbol{\xi} \\ & -\boldsymbol{\beta}^{*\top}(\epsilon\mathbf{1} + \boldsymbol{\xi}^* - \boldsymbol{g} + K\boldsymbol{\alpha} + b\mathbf{1}) - \boldsymbol{\eta}^{*\top}\boldsymbol{\xi}^*\end{aligned} \tag{16}$$

where $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_{l+u}]^\top$, $\boldsymbol{\xi}^* = [\xi_1^*, \ldots, \xi_{l+u}^*]^\top$, $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_{l+u}]^\top$, $\boldsymbol{\eta}^* = [\eta_1^*, \ldots, \eta_{l+u}^*]^\top$, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{l+u}]^\top$ and $\boldsymbol{\beta}^* = [\beta_1^*, \ldots, \beta_{l+u}^*]^\top$. By setting the derivatives of (16) with respect to all the variables to be zeros, we have

$$\begin{aligned}\partial_{\boldsymbol{\alpha}}\mathcal{L} &= K\boldsymbol{\alpha} + K(\boldsymbol{\beta} - \boldsymbol{\beta}^*) = 0, \\ \partial_b\mathcal{L} &= \mathbf{1}^\top(\boldsymbol{\beta} - \boldsymbol{\beta}^*) = 0, \\ \partial_{\boldsymbol{\xi}}\mathcal{L} &= C\mathbf{1} - \boldsymbol{\beta} - \boldsymbol{\eta} = 0, \\ \partial_{\boldsymbol{\xi}^*}\mathcal{L} &= C\mathbf{1} - \boldsymbol{\beta}^* - \boldsymbol{\eta}^* = 0, \\ \partial_{\boldsymbol{g}}\mathcal{L} &= \Lambda(\boldsymbol{g} - \boldsymbol{y}) + \mu L\boldsymbol{g} - \boldsymbol{\beta} + \boldsymbol{\beta}^* = 0.\end{aligned} \tag{17}$$

Substituting these equations back into (14), we can arrive at the dual of (14) as

$$\begin{aligned}\min_{\boldsymbol{\beta}, \boldsymbol{\beta}^*} \quad & \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^*)^\top\tilde{Q}(\boldsymbol{\beta} - \boldsymbol{\beta}^*) \\ & +\epsilon\mathbf{1}^\top[\boldsymbol{\beta} + \boldsymbol{\beta}^*] + \tilde{\mathbf{y}}^\top[\boldsymbol{\beta} - \boldsymbol{\beta}^*] \\ \text{s.t.} \quad & \mathbf{1}^\top\boldsymbol{\beta} = \mathbf{1}^\top\boldsymbol{\beta}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^* \in [0, C]\end{aligned} \tag{18}$$

where

$$\tilde{Q} = K + (\Lambda + \mu L)^{-1} \tag{19}$$

is the transformed kernel matrix, and

$$\tilde{\mathbf{y}} = (\Lambda + \mu L)^{-1}\Lambda\mathbf{y} \tag{20}$$

is the transformed label vector or virtual label vector in which the labels of the whole data set are propagated from the labeled data. It is worth noting that the transformed kernel is a summation of two kernels: the original kernel and a graph kernel denoted as $K_G = (\Lambda + \mu L)^{-1}$ [35] that is essential to capture the data manifold.

After solving for $\boldsymbol{\beta}$ and $\boldsymbol{\beta}^*$, $\boldsymbol{\alpha}$ can be recovered as

$$\boldsymbol{\alpha} = \boldsymbol{\beta}^* - \boldsymbol{\beta} \tag{21}$$

according to the Karush–Kuhn–Tucher (KKT) condition (17) and the final decision function is

$$f(\mathbf{x}) = \sum_{\alpha_i \neq 0}\alpha_i k(\mathbf{x}, \mathbf{x}_i) + b. \tag{22}$$

The optimization problem has the following advantages.

1) It shares the same form as the dual of standard ϵ-SVR [36]. The only differences in our method are that we have a different kernel matrix $\tilde{Q}$ and a different label vector $\tilde{\mathbf{y}}$. It thus can be solved efficiently using the QP solver such as the one in LIBSVM [32].
2) Sharing the same dual with ϵ-SVR, the solutions of β_i and β_i^* also inherit the good characteristics of sparsity as in ϵ-SVR. This can be verified by the KKT conditions: $\beta_i(\epsilon + \xi_i + g_i - f(\mathbf{x}_i)) = 0$ and $\beta_i^*(\epsilon + \xi_i^* - g_i + f(\mathbf{x}_i)) = 0$. As many values of $f(\mathbf{x}_i)$ lie inside the ϵ-tube, most values of β_i and β_i^* are zero. Moreover, we have $\boldsymbol{\alpha} = \boldsymbol{\beta}^* - \boldsymbol{\beta}$, so $\boldsymbol{\alpha}$ is sparse as well.
3) The recovery of expansion coefficients $\boldsymbol{\alpha}$ from the dual variables $\boldsymbol{\beta}$, $\boldsymbol{\beta}^*$ is much simpler than the traditional LapSVR, which needs to solve a linear system of a matrix of the size $l+u$, i.e., the complexity is $O((l+u)^3)$.
4) The resultant transformed kernel matrix in (19) is also much simpler, and we can independently cope with K and the graph Laplacian matrix L in the graph kernel K_G. Moreover, the sparsity of L is preserved in $\Lambda + \mu L$, which benefits the consequent inverse operation.

B. Efficient Evaluation of the Transformed Kernel

1) Motivation: Although the transformed kernel (19) has a much simpler form, it is still quite expensive to directly calculate K_G using full eigen-decomposition, because the time complexity is still $O(n^3)$. We observe that the graph Laplacian matrix L is usually highly sparse if kNN is used to construct the graph as described in Section II-A. In this case, there are more efficient techniques (i.e., the Jaccobi, the Arnoldi, and the Hebbian methods [31], [37]) to compute only a few eigenvectors and eigenvalues. These methods conduct iterative matrix-vector products at each step. When the matrix is sparse, they can be implemented more efficiently.

On the other hand, according to [38], the relevant information for a classification problem is contained in a finite number of leading kernel PCA components if the kernel matches the underlying learning problem. In other words, a small number of eigenvectors corresponding to the largest eigenvalues of the kernel matrix are sufficient to capture the data manifold, and those corresponding to smaller eigenvalues are more likely to be dominated by noise. In the following lemma, we will show the relationship between the eigenvectors corresponding to the smallest eigenvalues of graph Laplacian matrix, and the eigenvectors corresponding to the largest eigenvalues of affinity matrix W.

Lemma 1: Let us denote W as the affinity matrix of a graph, $\tilde{W} = D^{-1/2} W D^{-1/2}$ and $L = I - \tilde{W}$ as the normalized graph Laplacian matrix. If σ_i, $\Phi_{\cdot i}$ are the eigenvalue and eigenvector of L (i.e., the solution to $L\Phi_{\cdot i} = \sigma_i \Phi_{\cdot i}$), then $(1 - \sigma_i)$, $\Phi_{\cdot i}$ are, respectively, the eigenvalue and eigenvector of W (i.e., the solution to $\tilde{W}\Phi_{\cdot i} = (1 - \sigma_i)\Phi_{\cdot i}$).

Proof: $L\Phi_{\cdot i} = \sigma_i \Phi_{\cdot i} \Rightarrow (I - \tilde{W})\Phi_{\cdot i} = \sigma_i \Phi_{\cdot i} \Rightarrow \tilde{W}\Phi_{\cdot i} = (1 - \sigma_i)\Phi_{\cdot i}$. So we prove Lemma 1.

Meanwhile, it is easy to show that $\tilde{W}$ is a kernel matrix as it is symmetric and positive semidefinite. Based on the analysis in [38], we can reach the conclusion that the relevant information for a classification problem is contained in the first few eigenvectors of the graph Laplacian matrix L corresponding to the smallest eigenvalues up to a negligible error.

2) Simplification of the Transformed Kernel: Based on the above analysis, we propose to project $\boldsymbol{g}$ onto a subspace spanned by only a few eigenvectors of the graph Laplacian matrix L that are associated with the smallest eigenvalues. In practice, we enforce $\boldsymbol{g}$ to lie in the subspace spanned by these eigenvectors (bases). We denote $\Phi_{\cdot i}, \sigma_i$ as the eigenvector and eigenvalue of the graph Laplacian matrix L (i.e., $L\Phi_{\cdot i} = \sigma_i \Phi_{\cdot i}$, $i = 0, \ldots, n-1$). Without loss of generality, we assume $\sigma_0 \leq \cdots \leq \sigma_{n-1}$. Then, we have $\boldsymbol{g} = \sum_{i=0}^{n_{ev}-1} \zeta_i \Phi_{\cdot i} = \bar{\Phi}\bar{\boldsymbol{\zeta}}$, where $n_{ev} < n$, $\bar{\Phi} = [\Phi_{\cdot 0}, \ldots, \Phi_{\cdot n_{ev}-1}]$ and $\bar{\boldsymbol{\zeta}} = [\zeta_0, \ldots, \zeta_{n_{ev}-1}]^\top$. Thus, the optimization problem of (14) becomes

$$\min_{f \in \mathcal{H}_k, \bar{\zeta}, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \frac{1}{2}\|f\|_{\mathcal{H}_k}^2 + C\sum_{i=1}^{l+u}(\xi_i + \xi_i^*) + \frac{1}{2}(\bar{\Phi}\bar{\boldsymbol{\zeta}} - \mathbf{y})^\top \Lambda (\bar{\Phi}\bar{\boldsymbol{\zeta}} - \mathbf{y}) + \frac{\mu}{2}\bar{\boldsymbol{\zeta}}^\top \bar{\Phi}^\top L \bar{\Phi}\bar{\boldsymbol{\zeta}} \tag{23}$$

$$\text{s.t.} \quad f(\mathbf{x}_i) - \bar{\Phi}_{i\cdot}\bar{\boldsymbol{\zeta}} \leq \epsilon + \xi_i$$
$$\bar{\Phi}_{i\cdot}\bar{\boldsymbol{\zeta}} - f(\mathbf{x}_i) \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0, i = 1, \ldots, l+u \tag{24}$$

where $\bar{\Phi}_{i\cdot}$ is the row vector representing the ith row of $\bar{\Phi}$. Under similar derivation procedures, we can arrive at the same dual as (18), with only different $\tilde{Q}$ and $\tilde{\mathbf{y}}$ as follows:

$$\tilde{Q} = K + \bar{\Phi} S^{-1} \bar{\Phi}^\top \tag{25}$$
$$\tilde{\mathbf{y}} = \bar{\Phi} S^{-1} \bar{\Phi}^\top \Lambda \mathbf{y} \tag{26}$$

where

$$S = \bar{\Phi}^\top \Lambda \bar{\Phi} + \mu \bar{\Sigma}_L \tag{27}$$

$\bar{\Sigma}_L = \text{diag}\{\sigma_0, \ldots, \sigma_{n_{ev}-1}\}$ and $S \in R^{n_{ev} \times n_{ev}}$. If $n_{ev} \ll n$, the computational cost to calculate S^{-1} can be reduced significantly.

3) Relationship to Laplacian Eigenmaps: The above approximation is also related to Laplacian eigenmaps [33]. In Laplacian eigenmaps, the first m eigenvectors of the graph Laplacian matrix after discarding the trivial eigenvector corresponding to zero eigenvalue are used as the embedding of the data in m-dimensional Euclidean space, i.e., $\mathbf{x}_j \rightarrow [\Phi_{j1}, \ldots, \Phi_{jm}]^\top$, where Φ_{ji} is the j^{th} element of column vector $\Phi_{\cdot i}$. We note $\bar{\Phi} = [\Phi_{\cdot 0}, \ldots, \Phi_{\cdot n_{ev}-1}]$ is also composed of the first n_{ev} eigenvectors of L corresponding to the smallest eigenvalues, and is similar to Laplacian eigenmaps. For a better comparison, we conduct an eigen-decomposition, namely, $S = V\Sigma_S V^\top$. Then, we have $\bar{\Phi} S^{-1} \bar{\Phi}^\top = \bar{\Phi} V \Sigma_S^{-1} V^\top \bar{\Phi}^\top = \tilde{\Phi}\tilde{\Phi}^\top$, where

$$\tilde{\Phi} = \bar{\Phi} V \Sigma_S^{-\frac{1}{2}} \tag{28}$$

can be treated as a new embedding of the data. In the extreme case when there is no labeled data, i.e., $\Lambda_{ii} = 0, \forall i$, according to (27) and (28), we have $S = \mu \bar{\Sigma}_L$, $\tilde{\Phi} = \bar{\Phi} S^{-1/2}$. Let us denote $\tilde{\Phi}_{\cdot i}$ and $\bar{\Phi}_{\cdot i}$ as the ith column of $\tilde{\Phi}$ and $\bar{\Phi}$, respectively. Then, we have $\tilde{\Phi}_{\cdot i} = \bar{\Phi}_{\cdot i}/\sqrt{\mu \sigma_i}$, so $\tilde{\Phi}$ is an embedding similar to Laplacian eigenmaps, except that each eigenvector is normalized by the corresponding eigenvalue, in which a higher weight is assigned to the eigenvector with a smaller eigenvalue, which plays a more important role for the smoothness of the manifold structure of the data. Although our method is similar to Laplacian eigenmaps, our method is different from it in the following two aspects.

a) Our new representation is affected by the labeled data as it is a transformed version of $\bar{\Phi}$. When we have some labeled data, according to (27), S is influenced by Λ that integrates the label information. Thus, $\tilde{\Phi}$ is also influenced by the label information according to (28).

b) Referring to the representation $\boldsymbol{g} = \bar{\Phi}\bar{\boldsymbol{\zeta}}$, we do not simply put the eigenvectors together. Instead, we seek for a better combination by learning the combination coefficients $\bar{\boldsymbol{\zeta}}$.

4) Extension to the Iterated Graph Laplacian: Our method can easily deal with a graph Laplacian matrix of a higher order. As $\Phi_{\cdot i}, \sigma_i, i = 0, \ldots, n_{ev} - 1$ are the eigenvectors and eigenvalues of the graph Laplacian matrix L, it is easy to show that $\Phi_{\cdot i}, \sigma_i^p, i = 0, \ldots, n_{ev} - 1$ are the eigenvectors and eigenvalues of L^p, $p \in \mathcal{N}$. Our method can then easily deal with L^p by replacing (27) with

$$S = \bar{\Phi}^\top \Lambda \bar{\Phi} + \mu \bar{\Sigma}_L^p \tag{29}$$

where $\bar{\Sigma}_L^p = \text{diag}\{\sigma_0^p, \ldots, \sigma_{n_{ev}-1}^p\}$.

C. Algorithm and Complexity Analysis

The algorithm to train LapESVR is listed in Algorithm 1. It takes the data matrix X, the label vector $\mathbf{y}$, and some parameters $C, \lambda, \mu, n_{ev}, p$ as inputs, and outputs the LapESVR classifier. Specifically, we first construct a kNN graph to compute the graph Laplacian matrix L, and then compute the n_{ev} eigenvectors of L corresponding to the n_{ev} smallest eigenvalues. Based on these eigenvectors and eigenvalues, we compute the new representation $\tilde{\Phi}$ from line 4 to 6. Then, we solve the QP problem in line 9 based on the transformed kernel $\tilde{Q}$ computed in line 8. Note we do not need to precompute $\tilde{Q}$ and K. Inspired by LIBSVM [32], we can directly calculate $\tilde{Q}$ as follows:

$$\tilde{Q}(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \tilde{\Phi}_{i\cdot}\tilde{\Phi}_{j\cdot}^\top \tag{30}$$

where $\tilde{\Phi}_{i\cdot}$ is the ith row of $\tilde{\Phi}$. Also, we compute the bias b in the same way as LIBSVM.

The time complexity of the proposed algorithm involves two parts: computing the transformed kernel, and solving the QP program in (18). As we use LIBSVM to solve the QP problem, the time complexity is no. of iterations $\times O(n)$ in the best case and no. of iterations $\times O(nd)$ in the worst case [32], where d is the feature dimension. Empirically, the time complexity lies between $O(n)$ and $O(n^{2.3})$ [3], [39].

As for the complexity for the calculation of transformed kernel, we only focus on the complexity for the calculation of graph kernel $K_G = (\Lambda + \mu L)^{-1}$ without considering the computational cost for constructing the graph Laplacian matrix L. The most computationally expensive part is to find the first n_{ev} eigenvectors of the graph Laplacian matrix L.

Algorithm 1 LapESVR

```
1: Input: $X=[\mathbf{x}_1, \ldots, \mathbf{x}_l, \mathbf{x}_{l+1}, \mathbf{x}_n] \in \mathcal{R}^{d\times n}$, $\mathbf{y}, C, \lambda, \mu, n_{ev}$.
2: Construct a $k$NN graph and compute the corresponding graph Laplacian matrix $L$ as described in Section II-A.
3: Compute the $n_{ev}$ eigenvectors $\Phi_{\cdot i}$ of $L$ associated with the $n_{ev}$ smallest eigenvalues $\sigma_i$, $i = 1, \ldots, n_{ev}$, and denote $\bar{\Phi} = [\Phi_{\cdot 1}, \ldots, \Phi_{\cdot n_{ev}}]$ and $\bar{\Sigma}_L = \text{diag}\{\sigma_1, \ldots, \sigma_{n_{ev}}\}$.
4: $S = \bar{\Phi}^\top \Lambda \bar{\Phi} + \mu \bar{\Sigma}_L^p$.
5: Do eigen-decomposition $S = V\Sigma_S V^\top$.
6: $\tilde{\Phi} = \bar{\Phi} V \Sigma_S^{-\frac{1}{2}}$.
7: $\tilde{\mathbf{y}} = \tilde{\Phi}\tilde{\Phi}^\top \Lambda \mathbf{y}$.
8: $\tilde{Q} = K + \tilde{\Phi}\tilde{\Phi}^\top$.
9: Solve (18) to obtain $\boldsymbol{\beta}$, $\boldsymbol{\beta}^*$ and $b$ using the QP solver in LIBSVM.
10: $\boldsymbol{\alpha} = \boldsymbol{\beta}^* - \boldsymbol{\beta}$.
11: Output: LapESVR classifier: $f(\mathbf{x}) = \sum_{\alpha_i \neq 0} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b$.
```

In our implementation, the MATLAB function *eigs* is used, which implements the Arnoldi technique [31]. The Arnoldi method requires the computation of matrix-vector products at each iterative step, but it is efficient when L is sparse. It takes roughly $O(n_{ev}\bar{k}n)$ time complexity to calculate the first n_{ev} eigenvectors of L, where $\bar{k}$ is the average number of non-zero entries in each column of L. Then, computing S in (27) requires $O(n_{ev}l^2 + n_{ev})$ time complexity, where l is the number of labeled data. Finally, computing S^{-1} and the matrix chain product in (25) require $O(n_{ev}^3)$ and $O(n_{ev}^2 n + n_{ev}n^2)$ time complexity, respectively, leading to a total time complexity of $O(n_{ev}\bar{k}n + n_{ev}l^2 + n_{ev} + n_{ev}^3 + n_{ev}^2 n + n_{ev}n^2)$. Usually, we have $l, n_{ev}, \bar{k} \ll n$, and the time complexity for computing the transformed kernel in (25) reduces to $O(n_{ev}(n + n_{ev} + \bar{k})n)$. Here, we omit the time complexity for computing K, which requires $O(n^2)$ time complexity. In the contrary, LapSVR involves an expensive inversion operation for an $n \times n$ matrix and it requires $O(n^3)$ time complexity. Moreover, when the SMO strategy is employed, we do not need to explicitly compute K_G and only need to compute the embedding in (28), which requires $O(n_{ev}\bar{k}n + 2n_{ev}^3 + nn_{ev}^2)$ time complexity. When $n_{ev} \ll n$, the time complexity reduces to $O((\bar{k} + n_{ev})n_{ev}n)$. As for space complexity, we only need to store the features X and the sparse graph Laplacian matrix L and the new data representation $\tilde{\Phi}$, leading to a total space complexity of $O((d + \bar{k} + n_{ev})n)$, which is linear with respect to n.

D. LapERLS

We can also easily extend the RLS to LapERLS under our framework. The objective function of LapERLS is

$$\min_{f \in \mathcal{H}_k, \boldsymbol{g}, \xi_i} \quad \frac{1}{2}\|f\|_{\mathcal{H}_k}^2 + \frac{1}{2}C\sum_{i=1}^{l+u}\xi_i^2 + \frac{1}{2}(\boldsymbol{g} - \mathbf{y})^\top \Lambda (\boldsymbol{g} - \mathbf{y}) + \frac{1}{2}\mu \boldsymbol{g}^\top L \boldsymbol{g} \tag{31}$$

$$\text{s.t.} \quad \xi_i = f(\mathbf{x}_i) - g_i, i = 1, \ldots, l + u. \tag{32}$$

According to Proposition 1, we can easily show that $f(\mathbf{x}) = \sum_{\alpha_i \neq 0} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$. Let us denote $\boldsymbol{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^\top$.

Then we have $\boldsymbol{f} = K\boldsymbol{\alpha}$. By introducing it into (31), we arrive at the following optimization problem:

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{g}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} + \frac{1}{2}C\|\boldsymbol{\xi}\|^2 \\ & +\frac{1}{2}(\boldsymbol{g}-\mathbf{y})^\top \Lambda(\boldsymbol{g}-\mathbf{y}) + \frac{1}{2}\mu \boldsymbol{g}^\top L\boldsymbol{g} \\ \text{s.t.} \quad & \boldsymbol{\xi} = K\boldsymbol{\alpha} - \boldsymbol{g}. \end{aligned} \tag{33}$$

By introducing the Lagrange multipliers β_i into the constraints in (33), we can easily reach the following dual problem:

$$\min_{\boldsymbol{\beta}} \ \frac{1}{2}\boldsymbol{\beta}^\top \tilde{Q}\boldsymbol{\beta} + \frac{1}{2}\boldsymbol{\beta}^\top \tilde{\mathbf{y}} \tag{34}$$

where

$$\tilde{Q} = K + (\Lambda + \mu L)^{-1}, \quad \tilde{\mathbf{y}} = (\Lambda + \mu L)^{-1}\Lambda\mathbf{y}$$

are the same as (19) and (20), respectively. Therefore, the same techniques for the efficient calculation of the transformed kernel in LapESVR are also applicable to LapERLS here. We can readily develop the algorithm for learning the LapERLS classifier by replacing the QP solver in line 9 of Algorithm 1. After solving $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$ is recovered as $\boldsymbol{\alpha} = -\boldsymbol{\beta}$ according to KKT condition. Thus, the final decision function is

$$f(\mathbf{x}) = \sum_{\alpha_i \neq 0} \alpha_i k(\mathbf{x}, \mathbf{x}_i). \tag{35}$$

E. Discussions on the Proposed Formulation

Now, we present more detailed analysis about the proposed formulation in (11).

1) Relationship to Standard Regularization: Our proposed formulation is a semi-supervised extension of standard regularization.

Proposition 2: When setting $\mu = 0$ and $\lambda \to \infty$: 1) LapESVR reduces to SVR and 2) LapERLS reduces to RLS.

Proof: We first prove 1). Let us assume $\mu = 0$ and i_u represents the indeces of unlabeled data. Then, we have $\tilde{Q}_{i_u i_u} = \infty$ according to (19) and $\tilde{y}_{i_u} = 0$ according to (20). Therefore, the dual (18) has trivial solutions: $\beta_{i_u} = \beta^*_{i_u} = 0$ for the unlabeled data, and it reduces to

$$\begin{aligned}\min_{\boldsymbol{\beta}_l,\boldsymbol{\beta}_l^*} \quad & \frac{1}{2}(\boldsymbol{\beta}_l - \boldsymbol{\beta}_l^*)^\top \tilde{Q}_{l\times l}(\boldsymbol{\beta}_l - \boldsymbol{\beta}_l^*) \\ & +\epsilon\mathbf{1}^\top(\boldsymbol{\beta}_l + \boldsymbol{\beta}_l^*) + \mathbf{y}_l^\top(\boldsymbol{\beta}_l - \boldsymbol{\beta}_l^*) \\ \text{s.t.} \quad & \mathbf{1}^\top\boldsymbol{\beta}_l = \mathbf{1}^\top\boldsymbol{\beta}_l^*, \boldsymbol{\beta}_l, \boldsymbol{\beta}_l^* \in [0, C]\end{aligned}$$

where $\tilde{Q}_{l\times l} = K_{l\times l} + \Lambda^{-1}_{l\times l}$, $\boldsymbol{\beta}_l$ (*resp.* $\boldsymbol{\beta}_l^*$) is the sub-vector of $\boldsymbol{\beta}$ (*resp.* $\boldsymbol{\beta}^*$) corresponding to labeled data, and $\mathbf{y}_l$ is the ground truth label of labeled data. When setting $\lambda \to \infty$, we have $\tilde{Q}_{l\times l} = K_{l\times l}$, then LapESVR reduces to SVR.

Moreover, LapERLS reduces to LapRLS when setting $C \to \infty$.

2) Relationship to Regularization on a Graph: A typical form of regularization on a graph [21], [22], [34] is

$$\min_{\boldsymbol{g}} \ \frac{1}{l}\sum_{i=1}^{l}(g_i - y_i)^2 + \gamma_C \boldsymbol{g}^\top L\boldsymbol{g}. \tag{36}$$

By comparing with (11), we find that our formulation incorporates the above objective. Specifically, the loss function and the regularization term used in (36) are parts of our objective in (11). Therefore, our formulation can be considered as an out-of-sample extension[2] to the objective in (36).

IV. Discussion of Related Works

Due to the extensive literature on SSL, our review focuses on methods that are scalable to large scale problems. Interested readers can refer to [26] and [40] for two comprehensive surveys.

A. Linear Methods

Linear methods can always scale to large scale problems because of optimization techniques dedicated to the linear case (e.g., linear version of TSVM, S^3VM, LapRLS, and LapSVM). We only introduce the linear LapRLS, which is one of the examples from the linear manifold regularization framework [28]. The objective of linear LapRLS is

$$\begin{aligned}\min_{\mathbf{w},b} \quad & \frac{1}{l}\sum_{i=1}^{l}(y_i - \mathbf{w}^\top\mathbf{x}_i - b)^2 \\ & +\gamma_A\|\mathbf{w}\|^2 + \gamma_I \mathbf{w}^\top X^\top LX\mathbf{w}\end{aligned}$$

where $X = [\mathbf{x}_1, \ldots, \mathbf{x}_{l+u}]$. The solution can be obtained by solving the following linear system

$$(X_l^\top X_l + \gamma_A lI + \gamma_I X^\top LX)\mathbf{w} = X_l^\top \mathbf{y}_l$$

where X_l is the submatrix of X corresponding to the labeled data, and $\mathbf{y}_l$ is the label for the labeled data. This is a d-by-d linear system and it is independent of the number of data $(l + u)$. When d is small or X is highly sparse, it can be solved efficiently.

There are many other linear methods, such as linear TSVM and linear S^3VM [14], which can also scale to large scale problems. Generally, the linear models have achieved promising performance for some simple classification tasks like text categorization. However, in many real world applications, we need nonlinear methods to achieve better results. Unfortunately, it is much more difficult to develop nonlinear methods for large scale SSL.

B. Nonlinear Methods

1) Sparsified Laplacian Core Vector Machines (SLapCVM): In [30], Tsang *et al.* focused on the change of the *intrinsic norm* $\|f\|_I^2$. They applied ϵ-insensitive loss to the *intrinsic norm*. The dual of their optimization problem is also a QP problem with the size equal to the summation of the number of labeled data and the number of edges in the graph. They cast it to a minimum enclosing ball problem, which can be solved efficiently using CVM[3] and has shown success in

[2]It is worth noting that the out-of-sample extension of (36) was first introduced by Belkin *et al.* in [13]. Since our work is an extension of the work in [13], so our method is also an out-of-sample extension of (36).

[3]Available at http://c2inet.sce.ntu.edu.sg/ivor/cvm.html.

solving large scale problems [41]–[44]. However, the resultant QP is still very large, especially when the number of nearest neighbors used to construct the graph Laplacian matrix is large. In addition, their method cannot deal with the iterated graph Laplacian matrix (i.e., L^p), while our method can easily deal with L^p and at the same time solve a much smaller QP problem with the size being only the number of training samples.

2) Cluster Kernels: Cluster kernel methods focus on the change of data representation such that two points in the same cluster are preferred to be from the same class. A typical work is the one proposed by Chapelle *et al.* in [12]. The new kernels were generated by modifying the eigenspectrum of the kernel matrix, and the linear classifier was trained based on the new representation. However, it is quite expensive as it involves diagonalizing a matrix with the size of n, leading to time complexity of $O(n^3)$ [16]. In addition, their method is expensive in dealing with test samples, while our method provides a natural out-of-sample extension based on the decision function in (22).

3) PLapSVM: Recently, Melacci *et al.* [20] proposed to solve the PLapSVM problem by using the PCG method. Their primal algorithm can avoid the matrix inversion operation in the dual problem. However, their algorithm needs to store the entire kernel matrix and thus it has limited scalability. In addition, several early stopping heuristics are employed to reduce the PCG computation, resulting in immature solutions that may be unstable. In contrast, our method is more stable without using any early stopping heuristics.

4) Approximation on Graph Laplacian: Researchers have made efforts to compute the approximated eigenvectors and eigenvalues of the graph Laplacian matrix. The Nyström method in [45] was used by Talwalkar *et al.* to accelerate the kernel machine [46] by finding a few approximated eigenvectors of the graph Laplacian matrix. The Nyström method heavily depends on the choice of landmark points, and it will fail when the landmark points cannot adequately represent the distribution of data. Zhang *et al.* [47], [48] improved the Nyström method by using the k-means clustering centers as landmark points, which is also applied to large scale manifold learning and dimension reduction. Meanwhile, they also proposed a prototype vector machine (PVM) [3], which uses the k-means clustering centers as the prototypes and predicts the label of test data by using a weighted combination of the labels from these prototypes. However, the convexity of the objective function cannot be guaranteed in PVM, and thus it may suffer from local minima.

In order to guarantee the convexity of the objective function, in [6] Liu *et al.* proposed a method called Anchor graph regularization (AnchorGraphReg) by enforcing the constraints that the combination weights are nonnegative. They also explored a local linear embedding-like method to compute these weights with additional nonnegative constraints to enforce sparsity. In AnchorGraphReg, a similarity matrix Z was constructed between all the data points and a few of the anchor points (landmarks). Each row of Z is then normalized to unit length and the approximation of W is $\hat{W} = Z\Lambda^{-1}Z^\top$, where $\Lambda = \text{diag}(Z^\top\mathbf{1})$. Essentially, AnchorGraphReg works by first propagating the label information from the labeled data to the anchor points and then propagating back to the unlabeled data.

TABLE I
SUMMARY OF THE DATA SETS. n_k IS THE NUMBER OF NEAREST NEIGHBORS USED TO CONSTRUCT THE GRAPH LAPLACIAN MATRIX, AND p IS THE DEGREE (i.e., POWER) OF THE GRAPH LAPLACIAN MATRIX

Data set	**No. of attributes**	**No. of points**	**No. of labeled**	n_k	p
Two moons	2	400	2	6	1
g241c	241	1500	10	5	2
g241d	241	1500	10	5	2
Digit1	241	1500	10	5	2
USPS	241	1500	10	5	2
COIL2	241	1500	10	5	2
Text	11 960	1500	10	50	5
FACEMIT	361	31 022	20	6	1
MNIST	780	60 000	100	20	1
extended USPS	676	75 383	20	6	1

Fergus *et al.* [2] extended the eigenvectors of the normalized graph Laplacian matrix to the eigenfunctions of the weighted Laplace–Beltrami operators. Thereafter, they estimated the marginal distribution $P_{\mathcal{X}}$ using the histogram. They then constructed a smaller graph Laplacian matrix based on the virtual discrete points at which the associated probabilities are estimated from a histogram. The entire eigenvectors were then obtained by using the interpolation according to the eigenvectors of the smaller graph Laplacian matrix. However, they assumed the marginal distribution has a product form and each dimension is independent, so their method will fail when this assumption is violated, which is often the case in real applications. Similarly, the approximated eigenvectors can also be used in our framework.

V. EXPERIMENTS

In this section, we perform experiments to show the performance and scalability of our framework. We start with a toy problem on the famous two-moons data set, and then we compare the results of existing (semi-)supervised algorithms on the benchmark data sets in [16] to demonstrate the performance of our methods. In order to show the scalability of our methods, we perform experiments on three large scale real world data sets: FACEMIT, MNIST, and extended USPS. Finally, we compare the out-of-sample performance of different algorithms on the full extended USPS data set. Table I is a summary of the data sets used in the experiments, in the transductive setting. Gaussian kernel, $k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\rho\|\mathbf{x}_i - \mathbf{x}_j\|^2}$, with $\rho = n^2/\sum_{i,j=1}^n \|\mathbf{x}_i - \mathbf{x}_j\|^2$ is used as the default kernel for all the methods. The affinity matrix W is also defined as $W_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|/2\sigma_g^2)$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are adjacent, and set to zero otherwise. σ_g is set to be the mean distance among neighbors. The number of nearest neighbors n_k and the degree of the graph Laplacian matrix p for all the data sets are shown in Table I. Their parameters are set according to

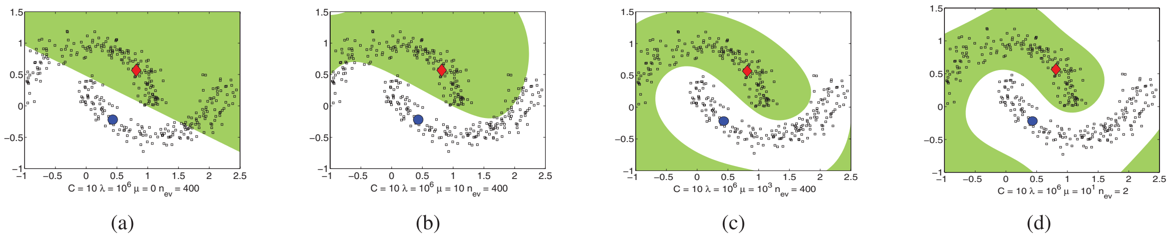

Fig. 1. Separating planes of LapESVR with different μ and n_{ev}. Two labeled points are shown in different colors, while the other points are unlabeled samples. When setting $\mu = 0$, LapESVR reduces to the supervised method (see (a)). With the increase of μ, the separating plane becomes better (see (a)–(c)). Selecting the optimal number of eigenvectors can also lead to a better separating plane (see (b) and (d)).

TABLE II
TEST ERRORS (MEAN ± STANDARD DEVIATION%) ON THE BENCHMARK DATA SETS. THE NUMBER IN PARENTHESIS IS THE RANK OF EACH ALGORITHM IN TERMS OF ITS MEAN. THE LAST COLUMN SHOWS THE AVERAGE RANK OF EACH ALGORITHM OVER ALL THE DATA SETS. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

	g241c	g241d	Digit1	USPS	COIL2	Text	Average Rank
SVR(Linear)	45.41 ± 4.37(5)	43.55 ± 5.17(5)	22.98 ± 5.16(6)	20.09 ± 0.24(8)	43.57 ± 5.38(8)	44.90 ± 6.16(6)	6.33
SVR(RBF)	47.02 ± 4.53(6)	45.55 ± 5.21(6)	25.01 ± 7.87(7)	19.26 ± 1.82(6)	43.03 ± 3.79(7)	45.37 ± 6.32(7)	6.50
LapRLS	42.29 ± 2.89(4)	41.95 ± 3.25(4)	**5.64 ± 2.49(1)**	**14.75 ± 2.52(1)**	31.20 ± 2.98(2)	35.10 ± 8.40(3)	2.50
LapSVR	47.02 ± 4.53(6)	45.55 ± 5.21(6)	21.00 ±10.55(4)	16.10 ± 2.52(3)	41.23 ± 5.21(5)	35.56 ± 8.23(4)	4.67
PLapSVM [20]	47.13 ± 4.55(8)	45.77 ± 5.23(8)	21.21 ±13.33(5)	18.47 ± 1.59(5)	40.25 ± 5.26(4)	39.28 ±10.06(5)	5.83
Eigenfunction [2]	40.88 ± 3.02(3)	**38.79 ± 3.95(1)**	26.64 ± 3.83(8)	19.59 ± 2.66(7)	42.18 ± 4.27(6)	46.31 ± 2.19(8)	5.50
LapERLS	39.49 ± 8.58(2)	41.57 ± 3.23(3)	6.45 ± 4.60(2)	15.87 ± 3.89(2)	**30.93 ± 8.04(1)**	32.88 ± 5.74(2)	2.00
LapESVR	**33.86 ±12.69(1)**	41.26 ± 3.40(2)	7.19 ± 3.59(3)	18.31 ± 4.58(4)	35.10 ± 5.91(3)	**32.81 ± 7.09(1)**	2.33

previous works [20], [26], [30]. For all the three large scale data sets, we fix $p = 1$, because the performances using $p = 1$ are already very good as shown in the experimental results. In addition, we use the sign of the decision function, $\text{sign}(f(\mathbf{x}))$, for the final classification, and we adopt the mean test error for the performance evaluation.

A. Two-Moons Data Set

The two-moons data set is shown in Fig. 1, which contains 400 points with one labeled point for each class. To evaluate the performance variations of LapESVR, we fix $C = 10$, $\lambda = 10^6$, $n_{ev} = 400$ while setting $\mu = \{0, 10, 10^3\}$. Note $\lambda = 10^6$ is sufficiently large for this toy data set. When setting $\mu = 0$, LapESVR reduces to SVR (see Fig. 1(a)). Fig. 1(a)–(c) shows the results when using different μ. It can be observed that the separating plane becomes better with the increase of μ. We also fix μ, in order to evaluate the influence of parameter n_{ev}. Note that when setting $\mu = 10$, the separating plane is not satisfactory (see Fig. 1(b)). We then change n_{ev} and draw the corresponding separating plane. We observe that the separating plane becomes better when setting $n_{ev} = 2$ (see Fig. 1(d)). The observation that fewer eigenvectors can achieve better performance demonstrates that not all of the eigenvectors of the graph Laplacian matrix are useful, and the manifold structure can be better reflected by a few eigenvectors associated with the smaller eigenvalues, which is analyzed in Section III-B.

B. Benchmark Data Sets

The second experiment is carried out on the SSL benchmark data sets as shown in Table I. Because the BCI data set has no manifold structure, we do not test it in the experiment. For all the other benchmark data sets, we strictly follow the experimental setup in [13] and [26] and adopt the transductive setting. The experiments are repeated 12 times, and the mean test error and standard derivation are reported.

We compare our proposed methods with PLapSVM [20], LapRLS and LapSVR [13]. We also report the results of SVR with radial basis function (RBF) kernel and linear kernel as two baselines, and refer to them as SVR(RBF) and SVR(Linear), respectively. The results of the large scale SSL method proposed in [2], which is referred to as *Eigenfunction*, is also reported for comparison.

There are too few labeled data in these data sets, so it is not meaningful to conduct cross validation to determine the optimal parameters. Following the setting in [16], we tune the parameters for all the methods and report their best results from the optimal parameters using the test data set for a fair comparison. For the baselines SVR(RBF) and SVR(Linear), we set C to be in the set of $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^4, 10^6\}$. The optimal parameters γ_A and γ_I of LapRLS and PLapSVM are selected from $\{10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 1, 10, 10^2\}$. The parameters related to the stopping criteria are set as default values for PLapSVM. For LapSVR, we set C and μ in the set of $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^4, 10^6\}$. Before conducting *Eigenfunction* [2], PCA is performed as suggested in [2] to de-correlate each feature dimension. For *Eigenfunction* [2], we set the total number of bins in the set of $\{100, 500, 1000\}$, $n_{ev} = 2d$ with d being the feature dimension, and $\lambda = \{1, 10, 10^2, 10^3\}$ for all the data sets. For LapESVR and LapERLS, we also set C and μ in the set of $\{10^{-2}, 10^{-1}, 1, 10, 10^2, 10^4, 10^6\}$ and set $\lambda = \{10^2, 10^4, 10^6\}$.

As the parameter n_{ev} in our LapESVR and LapERLS influences not only the training time, but also the performance, we set n_{ev} to different values between 1 and 1000.

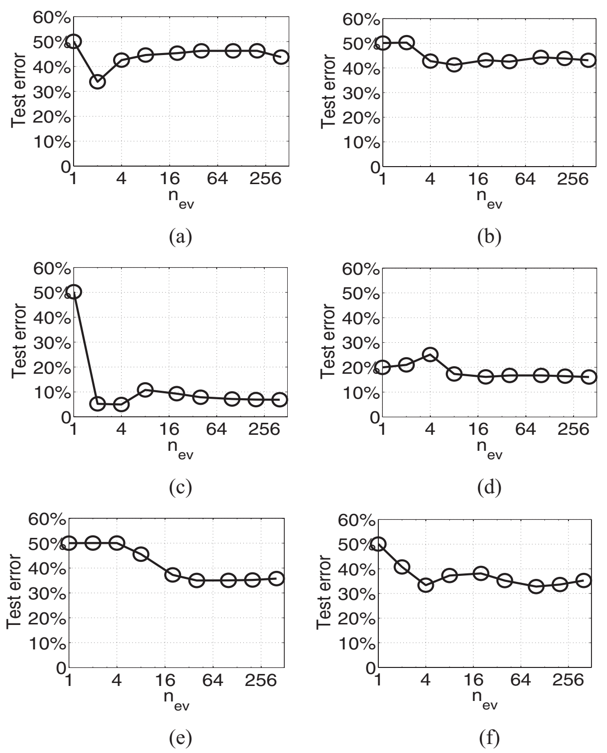

Fig. 2. Test error of LapESVR with different n_{ev} on the benchmark data sets. (a) g241c. (b) g241d. (c) Digit1. (d) USPS. (e) COIL2. (f) Text.

Fig. 2(a)–(f) shows the test error of LapESVR with different n_{ev}. We can observe that the performances tend to be stable for the Digit1, USPS, COIL2, and Text data sets when setting $n_{ev} \geq 100$. We then fix $n_{ev} = 100$ for these data sets and empirically set $n_{ev} = 2$ for the g241c data set, and $n_{ev} = 8$ for the g241d data set as these parameters are already very good. The performances of all the algorithms are shown in Table II. For each data set, we report the rank of each algorithm in parentheses, and the last column shows the average rank of each algorithm over all the data sets. While LapSVR and PLapSVM share a similar formulation with only different loss functions, from Table II, we observe that LapSVR performs slightly better than PLapSVM in terms of average rank, possibly because of the instability of the solution using PCG and their early stopping heuristics in PLapSVM. Interestingly, *Eigenfunction* is only better than the baselines and PLapSVM, and it is much worse than other SSL methods LapRLS, LapSVR and our LapERLS and LapESVR, because of the large approximation error from the eigenfunction when using a limited number of bins. Our methods LapESVR and LapERLS outperform the other methods in terms of average rank. An explanation is that we use only parts of the eigenvectors of the graph Laplacian matrix that can better discover the data manifold.

C. Large Scale Data Sets

In this section, we conduct experiments on the three large data sets: FACEMIT, MNIST, and extended USPS, in order to test the scalability of our proposed approach LapESVR. We also test the out-of-sample performance of different algorithms on the extended USPS data set. All the experiments are conducted on an IBM server with 32G RAM and multicore 2.67GHz processors.

Considering that LapERLS and LapESVR achieve similar results on the benchmark data sets, we take LapESVR as an example to test our framework on the large scale data sets. For comparison, we also report the results of PLapSVM and *Eigenfunction*. PLapSVM cannot be compared on the MNIST and extended USPS data sets, because its current implementation[4] needs the precomputed kernel and it requires large memory that is beyond the maximum memory of our server. We do not report the results of LapSVR and LapRLS with RBF kernel, because it is computationally prohibitive to run them on large data sets. Instead, we report the results of linear LapRLS [referred to as LapRLS(Linear)] and a recently proposed method (referred to as AnchorGraphReg) in [6].

1) Data Sets and Experimental Setup: The details of the three large scale data sets are listed as follows.

a) **FACEMIT Data Set**: The FACEMIT data set[5] contains 6977 training samples (2429 positives, 4548 negatives), and 24 045 test samples (472 positives, 23 573 negatives). In our experiment, we combine them together to form a data set with 31 022 samples in total.
b) **MNIST Data Set**: The MNIST data set[6] is for handwritten digit recognition and it contains 60 000 training images and 10 000 test images. We use only the training set for binary classification under two settings: 1) the first five versus the last five digits in MNIST1 and 2) the odd digits versus the even digits in MNIST2.
c) **Extended USPS Data Set**: The extended USPS data set[7] is from [42], which aims to classify digit zero and digit one. The original USPS data set contains 1005 zeros and 1194 ones in the training set, and 359 zeros and 264 ones in the test set. Following [42], we extend this data set by first converting the resolution from 16×16 to 26×26, and then generate new images by shifting the original ones in all directions up to five pixels. Thus, the extended USPS dataset contains 266 079 images in the training set and 75 383 images in the test set. We use only the test set which contains 75 383 data points in the transductive setting.

We determine the optimal parameters by using cross-validation. For SVR(Linear), SVR(RBF), LapRLS(Linear), PLapSVM, *Eigenfunction* and our method LapESVR, we employ the same sets of parameters as those used in the benchmark data sets. The data are partitioned into three subsets S_l, S_v, and S_u for each data set, in which S_l is the labeled data set, S_v is the validation data set, and S_u is the test data set. For each data set, we construct S_l by randomly sampling ten images from each class and fix the number of samples in S_v as 1000. The remaining images are used as S_u.

In the cross-validation process, S_l and S_v are combined together to conduct the transductive inference where S_v is

[4]Available at http://www.dii.unisi.it/~melacci/lapsvmp/index.html.

[5]CBCL Face Database #1 Available at http://www.ai.mit.edu/projects/cbcl.

[6] Available at http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/multiclass.html#mnist.

[7]Available at http://c2inet.sce.ntu.edu.sg/ivor/cvm.html.

TABLE III

TEST ERRORS (MEAN ± STANDARD DEVIATION %) OF DIFFERENT ALGORITHMS ON THE FACEMIT, MNIST AND EXTENDED USPS. WE HAVE TWO SETTINGS FOR THE MNIST DATA SET: 1) THE FIRST FIVE DIGITS VERSUS THE LAST FIVE DIGITS IN MNIST1 AND 2) THE ODD DIGITS VERSUS THE EVEN DIGITS IN MNIST2. FOR ANCHORGRAPHREG [6], THE RESULTS USING DIFFERENT NUMBERS OF ANCHOR POINTS ARE REPORTED. THE RESULTS SHOWN IN BOLDFACE ARE SIGNIFICANTLY BETTER THAN THE OTHERS, JUDGED BY THE PAIRED T-TEST WITH A SIGNIFICANCE LEVEL OF 0.05

		FACEMIT	MNIST1	MNIST2	extended USPS
SVR(Linear)		16.26±7.39	25.71±1.60	18.56±0.94	27.98±4.89
SVR(RBF)		15.35±4.47	19.08±1.70	14.13±1.13	17.56±3.04
LapRLS(Linear)		17.72±5.86	26.45±1.74	19.12±1.37	41.13±3.64
PLapSVM [20]		3.63±0.85	-	-	-
Eigenfunction [2]		17.31±5.63	25.53±1.46	22.90±1.15	26.44±3.03
AnchorGraphReg [6]	500	13.24±4.19	12.11±0.52	11.52±1.00	5.77±5.77
	1000	10.66±4.33	11.49±0.93	10.70±1.04	4.54±5.20
	2000	9.10±3.48	11.23±0.58	9.76±1.03	4.07±5.28
	5000	10.35±4.53	9.37±1.15	7.50±1.10	3.77±4.72
LapESVR		**2.22±0.18**	**5.43±0.83**	**6.00±1.44**	**0.41±0.61**

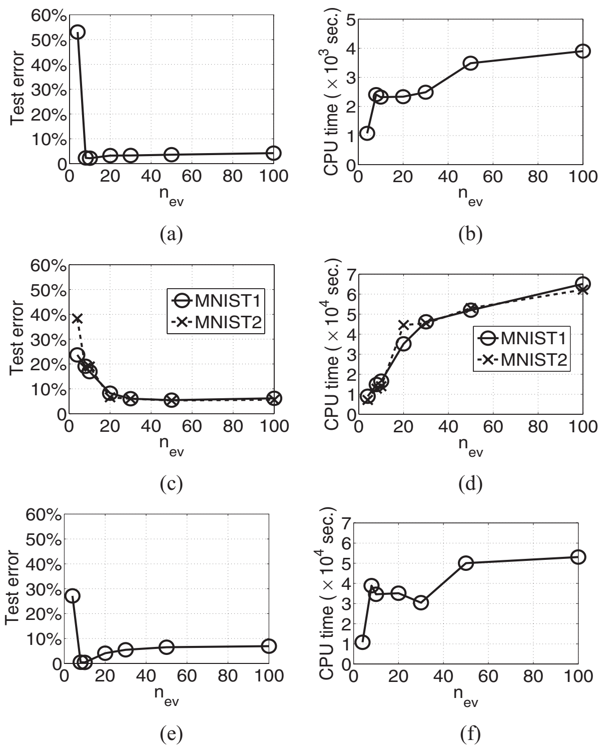

Fig. 3. Test error and training CPU time (in seconds) of LapESVR with different n_{ev} on three data sets. (a), (c), (e) show the test errors and (b), (d), (f) show the training CPU time. Two settings MNIST1 (the first five digits versus. the last five digits) and MNIST2 (the odd digits versus. the even digits) are used for MNIST data set.

used as the unlabeled data. We randomly run the experiments ten times with different random data partitions. Except for *Eigenfunction*, the best parameters of other algorithms are determined according to the mean test error over the ten rounds of experiments. In the testing step, S_l and S_u are combined together to perform the transductive inference using the best parameters decided by cross-validation, and we report the mean test errors and the standard deviations from the ten rounds of experiments in Table III. For *Eigenfunction*, we report the best mean test error from different number of bins and different values of λ using the test data set S_u because the parameters determined by cross-validation are generally not the optimal parameters. For AnchorGraphReg, we report the results using different numbers of anchor points (i.e., in the set of {500, 1000, 2000, 5000}) in order to test its influence on the final performance. We also empirically set $n_{ev} = \{4, 8, 10, 20, 30, 50, 100\}$ for LapESVR. Fig. 3(a)–(f), respectively, shows the test errors and training CPU time with respect to n_{ev} on the three large scale data sets. When setting $n_{ev} \geq 10$, the performances tend to be stable on the FACEMIT and extended USPS data sets, so we empirically fix $n_{ev} = 10$ when conducting the experiments on the FACEMIT and extended USPS data sets. For MNIST, the performances become stable when setting $n_{ev} = 20$, so we fix $n_{ev} = 20$ for this data set.

2) In-Sample Results: The experimental results are shown in Table III. We observe that LapESVR achieves the best results on all the three data sets in terms of the mean test error, and it is also significantly better than the other methods, judged by the paired t-test with a significance level of 0.05. *Eigenfunction* is even worse than SVR with the RBF kernel, possibly because of the approximation error. Note it is also reported in [6] that *Eigenfunction* is worse than SVR in some cases. LapRLS(Linear) performs poorly because the linear kernel is not appropriate for the nonlinear classification task. While AnchorGraphReg performs better than *Eigenfunction*, it is still worse than PLapSVM on the FACEMIT data set. An explanation is that a limited number of anchor points are not sufficient to represent the data distribution. While LapESVR also uses only a few eigenvectors corresponding to the smallest eigenvalues, those eigenvectors can better preserve the most useful information for discovering the manifold structure. AnchorGraphReg generally performs better when the number of anchor points increases. However, it becomes much slower when using more anchor points and it is still worse than our LapESVR even when setting the number of anchor points to 5000.

3) Out-of-Sample Results: The experiments in Sections V-B and V-C2 are carried out in the transductive setting. As discussed before, our LapESVR can be naturally used to predict the out-of-sample data because it is an inductive

TABLE IV
TEST ERRORS (MEAN ± STANDARD DEVIATION %) AND TESTING CPU TIME (SECONDS) OF DIFFERENT ALGORITHMS ON THE OUT-OF-SAMPLE DATA FROM THE EXTENDED USPS DATA SET. THE RESULTS SHOWN IN BOLDFACE ARE SIGNIFICANTLY BETTER THAN THE OTHERS, JUDGED BY THE PAIRED T-TEST WITH A SIGNIFICANCE LEVEL OF 0.05

		Test error	Testing CPU time
SVR(Linear)		28.40±4.70	1.45×10^{-5}
SVR(RBF)		18.67±3.12	1.50×10^{-5}
AnchorGraphReg	500	6.86±6.63	1.16×10^{-2}
	1000	5.26±5.98	1.69×10^{-2}
	2000	4.53±6.24	2.62×10^{-2}
	5000	4.45±5.89	5.92×10^{-2}
LapESVR		**0.31±0.64**	2.45×10^{-2}

algorithm. Now, we test our method on the out-of-sample data by employing the full extended USPS data set with 341 462 data points in total. Recall that, we have learned ten models from ten rounds of experiments by using the 75 383 training data points (see Section V-C2). In this paper, we directly use the ten models to predict the labels of the remaining 266 079 test data points. We also report the results from the baseline algorithms of SVR(Linear) and SVR(RBF). The AnchorGraphReg algorithm can be naturally extended to deal with the out-of-sample data using the method proposed in [49]. The test error and the average CPU time in the testing process are shown in Table IV. We observe that LapESVR outperforms AnchorGraphReg in terms of the mean test error and it is also significantly better than the other methods, judged by the paired t-test with a significance level of 0.05. AnchorGraphReg performs better when the number of anchor points increases. However, it is still worse than our LapESVR even when setting the number of anchor points as 5000 and the average testing time of AnchorGraphReg also increases quickly when using more anchor points.

4) Influence of Unlabeled Data: We also study the performance variations of our method LapESVR when using different number of unlabeled data on the extended USPS dataset. We fix the number of labeled data to be 20 and gradually increase the number of unlabeled data to train a set of LapESVR classifiers. The trained LapESVR classifiers are then applied to predict the labels of the 266 079 test data points of the extended USPS dataset. The experiments are repeated ten times with different randomly sampled unlabeled data and the average performance is reported. Fig. 4(a) and (b), show the test error and training CPU time with respect to the number of unlabeled data, respectively. With more unlabeled data, we can learn a better model that achieves lower test error while longer time is needed to train the model.

D. Discussion on the Eigenvectors of the Graph Laplacian

The number of eigenvectors n_{ev} used to represent the intermediate decision variable $\boldsymbol{g}$ is an important parameter for both test error and running time. The traditional method, using the entire graph Laplacian matrix, is equivalent to exploiting all the eigenvectors. However, as discussed in Section III-B, the utilization of more eigenvectors might not gain much in performance but it is less efficient. Usually, the best performance is achieved when setting n_{ev} in a certain range. For example, the performance on the g241c and g241d data sets is already very good when setting $n_{ev} = 2, 8$, respectively. Both data sets are artificially generated from a Gaussian distribution. The g241c data set satisfies the cluster assumption, while the g241d data set does not have obvious cluster structure. For large scale data sets like FACEMIT, MNIST, and extended USPS, it is sufficient to use $10 \sim 30$ eigenvectors [see Fig. 3(a), (c), and (e)], which is much smaller than the total number of samples.

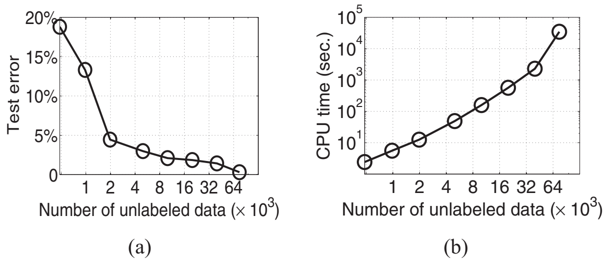

Fig. 4. Test error and training CPU time (in seconds) of LapESVR with respect to the number of unlabeled data on the extended USPS dataset. (a) Test Error. (b) Training CPU time.

VI. CONCLUSION

Observing that the successful SSL methods derived from the manifold regularization framework suffer from high computational cost when calculating the inverse of a transformed kernel, we have proposed a novel framework called Laplacian embedded regression by introducing an intermediate decision variable into manifold regularization for scalable SSL. Under the proposed framework, we have developed two algorithms LapESVR and LapERLS, whose transformed kernels have a much simpler form as the summation of the original kernel and a graph kernel that can capture the data manifold. We also proposed to project the intermediate decision variable into a subspace spanned by a few eigenvectors of the graph Laplacian matrix associated with the smallest eigenvalues, which further reduces the computational complexity and at the same time it can also improve the performance. Our extensive experiments on both toy and real world data sets not only show the effectiveness but also the scalability of our proposed methods.

REFERENCES

[1] D. Xu and S. Yan, "Semi-supervised bilinear subspace learning," *IEEE Trans. Image Process.*, vol. 18, no. 7, pp. 1671–1676, Jul. 2009.
[2] R. Fergus, Y. Weiss, and A. Torralba, "Semi-supervised learning in gigantic image collections," in *Proc. Neural Inf. Process. Syst.*, 2009, pp. 522–530.
[3] K. Zhang, J. T. Kwok, and B. Parvin, "Prototype vector machine for large scale semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, 2009, pp. 1233–1240.
[4] M. M. Adankon, M. Cheriet, and M. A. Biem, "Semisupervised least squares support vector machine," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1858–1870, Dec. 2009.

[5] B. Nadler, N. Srebro, and X. Zhou, "Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data," in *Proc. Neural Inf. Process. Syst.*, 2009, pp. 1330–1338.

[6] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 679–686.

[7] F. Nie, D. Xu, I. W. Tsang, and C. Zhang, "Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction," *IEEE Trans. Image Process.*, vol. 19, no. 7, pp. 1921–1932, Jul. 2010.

[8] X. Zhou and M. Belkin, "Semi-supervised learning by higher order regularization," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2011, pp. 1–9.

[9] Y. Yang, F. Nie, D. Xu, J. Luo, Y. Zhuang, and Y. Pan, "A multimedia retrieval framework based on semi-supervised ranking and relevance feedback," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 723–742, Apr. 2012.

[10] 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.

[11] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, Sep. 1998.

[12] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster kernels for semi-supervised learning," in *Proc. Neural Inf. Process. Syst.*, 2002, pp. 585–592.

[13] 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, pp. 2399–2434, Dec. 2006.

[14] K. P. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *Proc. Neural Inf. Process. Syst.*, 1998, pp. 368–374.

[15] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. Int. Conf. Mach. Learn.*, 1999, pp. 200–209.

[16] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. Int. Workshop Artif. Intell. Stat.*, 2005, pp. 57–64.

[17] R. Collobert, F. H. Sinz, J. Weston, and L. Bottou, "Large scale transductive SVMs," *J. Mach. Learn. Res.*, vol. 7, pp. 1687–1712, Aug. 2006.

[18] G. E. Hinton, P. Dayan, and M. Revow, "Modeling the manifolds of images of handwritten digits," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 65–74, Jan. 1997.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[20] S. Melacci and M. Belkin, "Laplacian support vector machines trained in the primal," *J. Mach. Learn. Res.*, vol. 12, pp. 1149–1184, Mar. 2011.

[21] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 912–919.

[22] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. Neural Inf. Process. Syst.*, 2004, pp. 321–328.

[23] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, "A probabilistic approach to WLAN user location estimation," *Int. J. Wireless Inf. Netw.*, vol. 9, no. 3, pp. 155–164, 2002.

[24] S. J. Pan, J. T. Kwok, Q. Yang, and J. J. Pan, "Adaptive localization in a dynamic WiFi environment through multi-view learning," in *Proc. AAAI Conf. Artif. Intell.*, 2007, pp. 1108–1113.

[25] T. Joachims, "Transductive learning via spectral graph partitioning," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 290–297.

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

[27] Z. Xu, I. King, M. R. 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.

[28] V. Sindhwani, P. Niyogi, M. Belkin, and S. Keerthi, "Linear manifold regularization for large scale semi-supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, Aug. 2005, pp. 80–83.

[29] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorihtms," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 181–201, Mar. 2001.

[30] I. W. Tsang and J. T. Kwok, "Large-scale sparsified manifold regularization," in *Proc. Neural Inf. Process. Syst.*, 2006, pp. 1401–1408.

[31] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Philadelphia, PA: SIAM, 1998.

[32] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[33] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Neural Inf. Process. Syst.*, 2001, pp. 585–591.

[34] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *Proc. Annu. Conf. Learn. Theory*, 2004, pp. 624–638.

[35] A. J. Smola and R. I. Kondor, "Kernels and regularization on graphs," in *Proc. Annu. Conf. Learn. Theory*, 2003, pp. 144–158.

[36] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[37] G. Gorrell, "Generalized hebbian algorithm for incremental singular value decomposition in natural language processing," in *Proc. Eur. ACL*, 2006, pp. 97–104.

[38] M. L. Braun, J. M. Buhmann, and K.-R. Müller, "Denoising and dimension reduction in feature space," in *Proc. Neural Inf. Process. Syst.*, 2006, pp. 185–192.

[39] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[40] X. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, Tech. Rep. 1530, 2005.

[41] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Large-scale maximum margin discriminant analysis using core vector machines," *IEEE Trans. Neural Netw.*, vol. 19, no. 4, pp. 610–624, Apr. 2008.

[42] 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, Apr. 2005.

[43] I. W. Tsang, J. T. Kwok, and J. M. Zurada, "Generalized core vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1126–1140, Sep. 2006.

[44] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Simpler core vector machines with enclosing balls," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 911–918.

[45] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Neural Inf. Process. Syst.*, 2001, pp. 682–688.

[46] A. Talwalkar, S. Kumar, and H. A. Rowley, "Large-scale manifold learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[47] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved Nyström low rank approximation and error analysis," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1232–1239.

[48] 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.

[49] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graph," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1–8.

Lin Chen received the B.E. degree from the University of Science and Technology of China, Hefei, China, in 2009. He is currently pursuing the Ph.D. degree with the School of Computer Engineering, Nanyang Technological University, Singapore.

His current research interests include computer vision, machine learning, especially large scale image retrieval and related techniques.

Ivor W. Tsang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2007.

He is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University (NTU), Singapore. He is the Deputy Director of the Center for Computational Intelligence, NTU.

Dr. Tsang received the prestigious IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2006 and the 2008 National Natural Science Award (Class II), China, in 2009. His co-authored papers also received the Best Student Paper Award at the 23rd IEEE Conference on Computer Vision and Pattern Recognition in 2010, the Best Paper Award at the 23rd IEEE International Conference on Tools with Artificial Intelligence in 2011, the 2011 Best Student Paper Award from PREMIA, Singapore, in 2012, and the Best Paper Award from the IEEE Hong Kong Chapter of Signal Processing Postgraduate Forum in 2006. He was also conferred with the Microsoft Fellowship in 2005.

Dong Xu received the B.E. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2001 and 2005, respectively.

He was with Microsoft Research Asia, Beijing, China, during his Ph.D. study, and the Chinese University of Hong Kong, Shatin, Hong Kong, for more than two years. He was a Post-Doctoral Research Scientist with Columbia University, New York, NY, for one year. He is currently an Associate Professor with Nanyang Technological University, Singapore. His current research interests include computer vision, statistical learning, and multimedia content analysis.

Dr. Xu was the co-author of a paper that won the Best Student Paper Award in the prestigious IEEE International Conference on Computer Vision and Pattern Recognition in 2010.