Discriminative Graph Embedding for Label Propagation

Canh Hao Nguyen and Hiroshi Mamitsuka

Abstract— **In many applications, the available information is encoded in graph structures. This is a common problem in biological networks, social networks, web communities and document citations. We investigate the problem of classifying nodes' labels on a similarity graph given only a graph structure on the nodes. Conventional machine learning methods usually require data to reside in some Euclidean spaces or to have a kernel representation. Applying these methods to nodes on graphs would require embedding the graphs into these spaces. By embedding and then learning the nodes on graphs, most methods are either flexible with different learning objectives or efficient enough for large scale applications. We propose a method to embed a graph into a feature space for a discriminative purpose. Our idea is to include label information into the embedding process, making the space representation tailored to the task. We design embedding objective functions that the following learning formulations become spectral transforms. We then reformulate these spectral transforms into multiple kernel learning problems. Our method, while being tailored to the discriminative tasks, is efficient and can scale to massive data sets. We show the need of discriminative embedding on some simulations. Applying to biological network problems, our method is shown to outperform baselines.**

Index Terms— **Graph embedding, label propagation, multiple kernel learning.**

I. Introduction

WE INVESTIGATE the problem of propagating labels on a proximity graph *without* descriptive information on the nodes of the graph. An example is in biomedical research, where the knowledge of gene and protein relations is conveniently encoded in networks [1]. The target is to use graph structures to learn a function that label the graphs' nodes in a reliable manner. Graph structures have shown to contain necessary information for various tasks. Graph structures are used in graph-based semi-supervised learning or learning with side information [2]. However, these methods require a premier information source of the nodes such as kernels or Euclidean spaces, which we assume not available. In these methods, the graphs are usually constrained to have parametric representations of subspaces of the original spaces that the nodes reside in. In our problem setting, there is no kernels or Euclidean representations of the nodes. Therefore, the graphs are not constrained to any parametric representation, allowing more flexibility in their possible space representations. Graph structures are also used in manifold learning, which are usually for an unsupervised purpose. We aim here for the supervised purpose.

By having only a graph representation of data, the conventional learning methods cannot be applied directly as they operate on Euclidean spaces or kernels. Therefore, it is the common solution to convert the graphs into conventional representations, in other words, to embed the graphs into Euclidean or kernel spaces [3]. Converting data on graphs into Euclidean or kernel spaces poses some difficulties. First, it is the difficulty of extracting and preserving the information that closely reflects the graph structures [4]. It is also difficult to extract only the relevant information for the task at hand, making it easier for the learning algorithms to work on. Another difficulty is to be able to deal with large scale datasets in an efficient manner.

We consider the supervised node classification problem given only a graph structure (edges) among the nodes, without nodes' descriptive attributes. An example can be found in Fig. 1. There are different approaches to the problem but none of them deliver an efficient and flexible solution for discriminative tasks. A major approach to this problem is to construct a kernel or an embedding that contains smooth functions on the graph [3]–[5]. This approach is unsupervised, and therefore the space may not contain the information required to construct the function of interest, hindering its effectiveness for discriminative task. This approach can be refined by using a spectral transform specifically for a supervised purpose [6], [7], but the learned function still resides in the same embedding space, which is not meant to contain discriminative information. A more direct approach is to use graph structure as a smoothness constraint [8], [9]. However, this approach is limited to the objective of a least square loss function on training labels. We will also show that this approach is equivalent to a limited choice of kernels with inappropriate decision functions, making it less reliable. The most flexible approach in terms of objective function is to use any loss function with a smoothness constraint. By using a general loss function [10] coupled with graph weight learning, the formulation is general but inefficient. The solution provided in [10] is the kernel learning approach with a simple graph Laplacian kernel.

We propose a method, named DisEmp, targeting both efficiency and effectiveness. The idea is to embed the graph into a Euclidean space enriched with discriminative information. From the embedded data in the space, a globally optimal and reliable learning step is carried out. It is shown to have the interpretation of combining principal component analysis (PCA) and linear discriminant analysis (LDA) [11] for graph embedding. It is made efficient by using a generalized eigenvalue

Manuscript received July 30, 2010; revised June 14, 2011; accepted June 19, 2011. Date of publication July 22, 2011; date of current version August 31, 2011. This work was supported in part by the Japan Science and Technology Agency (JST) through the Institute for Bioinformatics Research and Development (BIRD). The work of C. H. Nguyen was supported in part by the Japan Society for the Promotion of Science (JSPS).

The authors are with the Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho 611-0011, Japan (e-mail: canhhao@kuicr.kyoto-u.ac.jp; mami@kuicr.kyoto-u.ac.jp).

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/TNN.2011.2160873

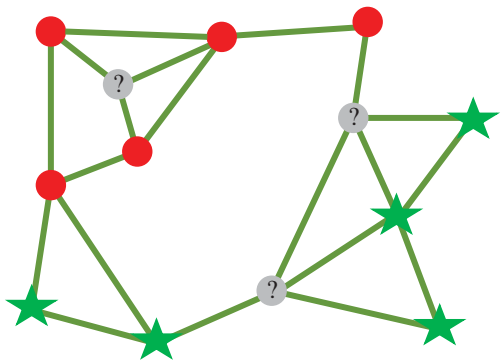

Fig. 1. Label propagation on graph, given a network connectivity of nodes with labels (ball or star), predict the labels of unlabeled nodes.

problem. The embedding makes the following learning step more flexible and reliable, meaning that the problem is *easier* to be learned. For the learning step, we specifically formulate an optimal feature weighting and large margin-based learning objective function, which is known for its effectiveness. This formulation leads to a multiple kernel learning (MKL) problem [12], which is well studied and has large-scale, efficient solutions. As this formulation is a large margin reformulation of the method in [7] equipped with efficient solutions, it is a contribution *per se*. By separating the embedding and learning steps, the total time of the method is the sum of the time of each step, which is known to be efficient.

This paper is organized as follows. We state the problem, review previous works and their limitations in Section II. We then describe our method in Section III. We elaborate more on its properties in Section IV. We present some simulations to show essential properties of the method in Section V. We then present an application of the method on gene function prediction problems in Section VI and conclude this paper.

II. Problem and Related Work

We state the problem as follows: Given a set of nodes $X = \{x_i\}_{i=1}^n$ and a graph $G = (X, W)$, in which G is a graph with edges $W = \{w_{ij}\}_{i,j=1}^n$, $w_{ij} \in \mathbb{R}$ (matrix of edge weights) and the node set X. A supervised learning problem is that given a set of labels, y_i of node x_i for some i, we want to learn a function f such that for an unlabeled node x, $f(x)$ is a reliable label. A pictorial description of the problem can be seen in Fig. 1. It is a common way to learn f that $f(x_i) \cong y_i$ and f is *regularized* in some forms. In this problem setting, only graph structure information is used to induce unknown labels on a graph from known ones.

It goes without saying that the graph edges' weights are interpreted as similarity scores between pairs of nodes. For that, it is the common goal of all the approaches to learn a function f that is *smooth* on the graph in the sense that $||f(x_i) - f(x_j)||$ must be small in presence of an edge with a large weight between x_i and x_j. Most methods quantify the smoothness of f by using graph Laplacians [13] as follows. Denote D as a diagonal matrix of the same size, in which its diagonal entries are calculated as $d_{ii} = \sum_{j=1}^n w_{ij}$. The (unnormalized) graph Laplacian is defined to be $L \stackrel{\text{def}}{=} D - W$. The smoothness of the function f on the graph is then measured as

$$f^T L f = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} (f_i - f_j)^2 \tag{1}$$

given that f is normalized, i.e., $f^T f = 1$.

The smaller $f^T L f$, the smoother f is. This measure is used for many different purposes [14]. It is used for semi-supervised learning [2], [15] where a function needs to be smooth on both labeled and unlabeled data. It is used in spectral clustering [9], [16] where the smoothness on clusters is sought. It is also used for dimensionality reduction [4] as the mappings based on eigenvectors of a graph Laplacian yield the most faithful embedding of the graph data on a Euclidean space in the sense that it gives the smoothest embedding on local neighborhoods.

There are two main lines of methods. The first line is called here *feature extraction methods*. They first extract data into a common representation such as on Euclidean spaces and then learning can be done efficiently and flexibly from there. The second line is called *integrated methods* for the reason of having both targets, label and graph structure information in a single objective function to be optimized. We will analyze all these methods in three criteria, the features extracted, the ways to weight features to build kernels, and the learning objective functions.

A. Feature Extraction Methods

We elaborate on the methods that fall into the category of extracting features from graphs for learning purposes. In these methods, features are extracted, weighted and then combined to form kernels. Different kernels only mean different features extracted and different ways to combined them.

1) Unsupervised Feature Extraction: One of the common methods for learning over a graph is to extract features or induce kernels from the graph structure, for example, diffusion kernels [5]. Diffusion kernels use connectivity between two nodes on the graphs to define a kernel value between the nodes. Suppose the eigen-decomposition of L is $L = \sum_{i=1}^n \lambda_i e_i e_i^T$, then the kernel

$$K \stackrel{\text{def}}{=} \exp(-\beta L) = \sum_{i=1}^n \exp(-\beta \lambda_i) e_i e_i^T.$$

The idea is extended further by using different transformations of the eigenvalue of L, each has a different interpretations as in [3]. The same idea is used for Euclidean embedding of manifolds constructed from neighborhood graphs [4]. Another variant is in [6] where parameter β of the diffusion kernel is set to optimize the von Neumann Entropy of the kernel K, subject to constraints on some smoothness measures.

2) Supervised Feature Combination: Given the features are eigenvectors, in [7], features are combined using label information directly for a discriminative purpose. It is a non-parametric spectral transform maximizing kernel target alignment (KTA) [17]. In this method, to guarantee smoothness, a kernel from the graph is limited to the family of

$K_\mu \stackrel{\text{def}}{=} \mu(\lambda_i) e_i e_i^T$, where μ is a non-increasing function. It is noteworthy here that this formulation offers a family of kernels that encompasses all the families suggested in [3]. The optimal transformation μ^* is determined by

$$\mu^* = \arg\max_{\mu} KTA(K_\mu, yy^T) \tag{2}$$

with y as the vector of labels and

$$KTA(K_\mu, yy^T) = \frac{\langle K, y \cdot y^T \rangle_F}{m\sqrt{(\langle K, K \rangle_F)}}$$

where $\langle , \rangle_F$ is the Frobenius inner product of matrices.

The problem of this feature weighting method is that using KTA is not theoretically sound as from the analysis in [18]. KTA optimization only means a higher classification in a strict condition. When the condition is not satisfied, higher KTA value does not mean a higher discriminative ability. Therefore, it is desirable to use other well-studied criteria such as margin to optimize feature weights.

These weighted features are then combined to build kernels. Learning is then applied in kernels, such as with large margin objective functions.

3) Remark: The following is observed for these methods.

a) Feature extraction: eigenvectors of graph Laplacian.
b) Feature weighting: with KTA as the objective function. It is not theoretically sound to optimize KTA. However, this method offers the most flexible kernel family compared to the non-learning methods in [3], [4], and [6].
c) Decision function: linear classification with large margin objective function.

B. Integrated Methods

These methods combine feature extraction, weighting, and learning into a single objective function. This enables weighting of smoothness, i.e., weighting of graph Laplacians, to be optimal for a supervised purpose. However, it is tricky to incorporate both steps to have an efficient, effective, and flexible objective function. The main reason is that to be efficient, most methods use the least square loss for a discriminative purpose instead of margin-based ones. To use margin-based losses, the solution is no longer computationally efficient [10].

1) Efficient Formulation: The typical methods, such as in, [8], [9], and [19], for one or more graphs, are formulated to optimize the following problem:

$$\arg\min_f (f - y)^T (f - y) \text{ s.t. } f^T L_i f \le \gamma \qquad \forall i. \tag{3}$$

Through a different procedure, the resulting solution of the above formulation coincides with the result of the step 2 of [20] or spectral learning with constraints in [21] and [22], that is[1]

$$f = (I + L)^{-1} y \stackrel{\text{def}}{=} Ky \tag{4}$$

for L as some conic combination of L_i.

[1]We denote M^{-1} here for both inverse and pseudoinverse of a matrix M for convenience.

We will show here that these methods [8], [9], [19], [20] are equivalent to using regularized graph kernels and the decision functions learned can be read from the kernel-induced feature spaces. In light of this analysis, we will show that there are two problems with all these approaches attributed to this formulation: 1) limited choices of kernels from the limited feature weighting options, and 2) inappropriate decision functions associated with the kernels.

The first problem 1) is as follows. Suppose that $L = \sum \lambda_i e_i e_i^T$ as the eigen-decomposition of L, then

$$K = \sum_{i=1}^{n} \frac{1}{\lambda_i + 1} e_i e_i^T.$$

This means that K is a regularized graph kernels based on L with a specific regularization function $r\colon \mathbb{R} \to \mathbb{R}$ that

$$r(\lambda) = \frac{1}{\lambda + 1}. \tag{5}$$

This means that implicit features used in these methods are still eigenvectors of graph Laplacians.

Furthermore, this also means that feature weighting in these methods follows the formulation in (5). This is a restricted kernel family that may not deliver the highest possible performance. There are many more parametric kernel families that can be applied, for examples, in [3]. The nonparametric spectral transform in [7], being the most flexible transform compared to that in [3], offers an even greater kernel family.

The second problem 2) of this formulation is its decision function, which is set to be Ky as in (4). As K here is regarded as a regularized graph kernel [3], we show that the decision function with the current setting of y does not give a sensible statistical interpretation on the space induced by the regularized graph kernel. Suppose that this kernel K induces a feature space representation $\Phi = [\phi_1, \phi_2 \ldots \phi_n]$ in some space, i.e., $K_{ij} = \phi_i^T \cdot \phi_j$, then

$$Ky = (\Phi^T \cdot \Phi) \cdot y = \Phi^T \cdot (\Phi \cdot y) = \Phi^T \cdot w \tag{6}$$

for $w = \Phi \cdot y$. In other words, this is a linear decision function with the weight vector w. In all these methods, components of y are set to be ± 1 depending on their classes, then

$$w = \sum_{y_i=1} \phi_i - \sum_{y_j=-1} \phi_j.$$

w is then the vector between the sums of all the points in the two classes. This means that w is not a meaningful weight vector for linear classification as in (6).

This decision function, for w as proposed in these methods, is efficient to compute but does not expect to deliver a high performance as no loss optimizations are performed once K is fixed.

2) General Formulation: The most versatile approach is in [10] that incorporates loss function with smoothness measure together

$$\arg\min_{f,\mu} \sum l(x_i, y_i, f) + \mu_i f^T L_i f \tag{7}$$

for any loss function l. This approach, from a graph kernel viewpoint, use the kernel $K = L^{-1}$, which is not regularized.

Input: *A graph G with training node labels y.*
1) *Constructing graph Laplacian L, training label matrix Y as in Section III-A.*
2) Step 1: *Embedding the graph into $\mathbb{R}^m$. Extracting the first m features in Section III-A.*
3) Step 2: *Weighting the features with a large margin objective function and learning a SVM classifier in Section III-B via MKL.*
4) *Inferring label of test nodes from the classifier.*
Output: *label function.*

Fig. 2. DisEmp: Discriminative graph embedding for label propagation.

This approach is very expensive as it must carry out the whole learning process as an internal step together with optimizing weights of graph Laplacians as the external loop. For only one graph with hinge loss, it becomes the non-optimized version in the feature weighting sense compared to other feature extraction methods in Section II-A.

3) Remark: For the efficient formulation, the following interpretations are concluded.

a) Feature extraction: not explicit but equivalent to eigenvectors of graph Laplacians.
b) Feature weighting: a specific form of regularization in (5).
c) Decision function: none-learning version. It is not optimal nor interpretable in the kernel-induced feature spaces.

The general formulation is too expensive computationally [10].

III. DisEmp: Discriminative Graph Embedding for Label Propagation

As we see in Section II, previous methods do not satisfy the need of both efficiency and effectiveness at the same time. We propose a method, named DisEmp, to classify labels on a graph for classification problems. A graph is first embedded into a Euclidean space that targets both, being faithful to the graph structure and having a discriminative ability. This step improves the embeddings without label information (in [3], [4], [6]) for a supervised purpose, therefore more information-rich. From the embedded distribution, a nonparametric spectral transform is used to optimally weight eigenvectors together with learning labels with a margin-based loss function. This step is both efficient and robust (compared to [7], [8], and [9]) because of the MKL formulation. An overview of our method is in Fig. 2.

A. Step 1: Discriminative Graph Embedding via Variance/Distance on Graphs

The purpose is to extract m features from a graph with two types of information: descriptive information of the graph, i.e., graph smoothness, and discriminative information with respect to label information. We describe both types of information before arriving at the final formulations. Given a graph G with the graph Laplacian L. In the context of supervised learning, we assume a label vector $y = \{y_i, y_2, \ldots, y_n\}^T$, $y_i \in \mathbb{R}$. In case of binary classification tasks, assume $y_i = (1/n_+)$ or $y_i = (-1/n_-)$ depending on whether x_i is a positive or negative example, respectively, where n_+ and n_- are corresponding numbers of examples in the classes. Testing data will have $y_i = 0$. Define Y to be a $n \times n$ matrix of labels, setting $Y_{ij} = y_i y_j$ as the (pairwise) *label matrix*. $f \in \mathbb{R}^n$ is denoted as any function on the graph.

1) Feature Objectives: The overall objective is to extract m features $(e_1, e_2 \ldots e_m)$, $e_i \in \mathbb{R}^n$ from a graph to represent it in a Euclidean space for discriminative learning (classification or regression). Therefore, it is desirable to have e_i already containing discriminative information, i.e., e_i depends on the label information of the nodes. It is a convention to seek e_i that is close to the label y in some senses. Therefore, our objective is to make e_i containing label information as well as respecting graph structures.

The first target is to extract m features with discriminative ability, named *variance to distance ratio*, we use the angle between y and f as follows:

$$(e_1, e_2 \ldots e_m) = \arg\min_f \frac{f^T f}{f^T Y f}. \tag{8}$$

The above target is solely for discriminative features, hence it may not respect the graph structures. Therefore, it is necessary to have e_i smooth on the graph. For this target, one could use a conventional setting

$$(e_1, e_2 \ldots e_m) = \arg\min_f \frac{f^T L f}{f^T f}. \tag{9}$$

The objective of our discriminative feature extraction from a graph is to achieve both targets (8) and (9). Since they are usually not satisfied at the same time, we propose ways to compromise both targets and depending on applications, users can choose a suitable compromise. We show in Section IV that the objectives are interpreted as LDA and PCA ways of embedding graphs.

2) Discriminative Graph Embedding: As we want to have both graph structure information and label information into the space, our discriminative embedding is a combination of the two targets in (8) and (9). When combining the two objectives into one, we need to compromise them by assigning a weight to each objective and combine them with weights in some ways. We propose three variations of objective combinations for DisEmp: *additive*, *subtractive*, or *multiplicative* DisEmp. The m features to be extracted are the m eigenvectors associated with the m smallest eigenvalues of these generalized eigenvalue problems.

Multiplicative DisEmp

$$(e_1, e_2 \ldots e_m) = \arg\min_f \left(\frac{f^T f}{f^T Y f}\right)^c \cdot \left(\frac{f^T L f}{f^T f}\right)^{(1-c)}. \tag{10}$$

It is expensive to raise to the powers c, therefore we can only recommend to use the case where $c = 0.5$ as follows:

$$\begin{aligned}(e_1, e_2 \ldots e_m) &= \arg\min_f \frac{f^T f}{f^T Y f} \cdot \frac{f^T L f}{f^T f} \\ &= \arg\min_f \frac{f^T L f}{f^T Y f}.\end{aligned} \tag{11}$$

Knowing that setting $c = 0.5$ does not allow different weightings of smoothness and label information, therefore, this combination is not supposed to give the highest possible performance.

Additive DisEmp

$$(e_1, e_2 \dots e_m) = \arg\min_f \frac{c \cdot f^T L f + (1-c) \cdot f^T f}{c \cdot f^T f + (1-c) \cdot f^T Y f} = \arg\min_f \frac{f^T (c \cdot L + (1-c) \cdot I) f}{f^T (c \cdot I + (1-c) \cdot Y) f}. \quad (12)$$

Subtractive DisEmp: Recall that the objective in (8) can be rephrased as

$$(e_1, e_2 \dots e_m) = \arg\min_f -\frac{f^T Y f}{f^T f}.$$

Then we have

$$(e_1, e_2 \dots e_m) = \arg\min_f c \cdot \frac{f^T L f}{f^T f} - (1-c) \cdot \frac{f^T Y f}{f^T f} = \arg\min_f \frac{f^T (c \cdot L - (1-c) \cdot Y) f}{f^T f}. \quad (13)$$

B. Step 2: Optimal Spectral Transform

Having obtained a set of m eigenvectors as features for the graph nodes, one can learn a kernel from this by weighting these eigenvectors for node labels prediction. We propose an efficient and effective method to combine features (eigenvectors) for a discriminative purpose (classification or regression) as in the algorithm in Fig. 2. As it transforms eigenvalues, it becomes a (nonparametric) spectral transform method. The formulation is robust because we use the large margin objective function. The problem is then formulated into a MKL problem, which is well-studied and has efficient solutions, for example in [12]. We show two versions of learning weights, one *with order constraints* and one *without order constraints* for completeness. However, our experiments later on only use the latter version (without order constraints).

1) Spectral Transform with Order Constraints: Denote $K_k = \sum_{i=1}^k e_i e_i^T$ as a sum of the first k kernels generated by the first k eigenvectors. Suppose that we take m eigenvectors, i.e., embedding the graph into an m-dimensional space. Then, for any non-increasing function μ on real numbers

$$\begin{aligned} K &= \sum_{i=1}^m \mu(\lambda_i) e_i e_i^T \quad \text{s.t. } \mu(\lambda_i) \geq \mu(\lambda_{i+1}) \\ &= \mu(\lambda_m) \sum_{i=1}^m e_i e_i^T \\ &\quad + (\mu(\lambda_{m-1}) - \mu(\lambda_m)) \sum_{i=1}^{m-1} e_i e_i^T \\ &\quad + \cdots + (\mu(\lambda_1) - \mu(\lambda_2)) \sum_{i=1}^{1} e_i e_i^T. \\ &= \mu(\lambda_m) K_m + \sum_{k=1}^{m-1} (\mu(\lambda_k) - \mu(\lambda_{k+1})) K_k. \end{aligned} \quad (14)$$

Denoting $\mu_m = \mu(\lambda_m)$, $\mu_k = \mu(\lambda_k) - \mu(\lambda_{k+1})$ for $k = 1, 2, \dots (m-1)$, we have a new formulation of K as

$$K = \sum_{k=1}^m \mu_k K_k \text{ for any } \mu_k \geq 0.$$

The key difference in this formulation compared to that of [7] is the use of auxiliary kernels K_k that removes the order constraints. The order constraints become the positivity constraints of differences. Corresponding coefficients of K_k are no longer the absolute transformed values of the eigenvalues of the graph Laplacian, but their differences.

Our method is then defined to be: finding $\mu_i \geq 0$ such that K has the largest margin (subject to total sum of μ_i being 1). It is formulated as follows:

$$\begin{aligned} \max_{\mu} \min_{\alpha} \quad & \sum_{k=1}^m \mu_k \left\{ \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K_k(i,j) - \sum_{i=1}^n \alpha_i \right\} \\ \text{w.r.t.} \quad & \alpha \in \mathbb{R}^n, \ \mu \in \mathbb{R}^m \\ \text{s.t.} \quad & 0 \leq \alpha \leq C, \ \mathbf{0} \leq \mu, \ \sum_{i=1}^n \alpha_i y_i = 0 \\ & \text{and } \sum_{k=1}^m \mu_k = 1. \end{aligned} \quad (15)$$

This formulation is a standard MKL problem [12]. One can use other variations of MKL problem as well. A solution to this formulation can be made efficient by using the well-studied MKL solution directly.

It is noteworthy that this is a large margin reformulation of the nonparametric spectral transform in [7]. By using large margin objective function, this formulation is theoretically sound, statistically interpretable and has computationally efficient solutions. Therefore, this formulation is a contribution *per se*.

2) Spectral Transform without Order Constraints: Given that the weight of each eigenvector (kernel) is computed to optimize the margin, it may not necessary to keep the order constraints. Instead, we let the weights to be determined for a large margin purpose. This may not keep the nonparametric spectral transforms decreasing. This is an extension of the transforms with order constraints, offering a larger kernel family to search from. The method then can be formulated as follows.

Denote $K_k^* = e_k e_k^T$ as the kernel formed by the k-th eigenvector. The problem is to find a (nonnegative) weight for each K_k^* independently. It is formulated as

$$\begin{aligned} \max_{\mu} \min_{\alpha} \quad & \sum_{k=1}^m \mu_k \left\{ \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K_k^*(i,j) - \sum_{i=1}^n \alpha_i \right\} \\ \text{w.r.t.} \quad & \alpha \in \mathbb{R}^n, \ \mu \in \mathbb{R}^m \\ \text{s.t.} \quad & 0 \leq \alpha \leq C, \ \mathbf{0} \leq \mu, \ \sum_{i=1}^n \alpha_i y_i = 0 \\ & \text{and } \sum_{k=1}^m \mu_k = 1. \end{aligned} \quad (16)$$

This also results in a MKL problem where previous efficient formulation and software are utilized, for example, in [12], among others. In our implementation, we use this formulation with the code from [12].

3) Discussion: Experiments in this paper follow the version in Section III-B2, nonparametric spectral transform *without* order constraints.

As in [7], spectral transform is used to generate kernels from graphs. One of its drawbacks is the objective function of spectral transform, being KTA, which is not theoretically sound. Another drawback is that it results in a standard second-order cone programming (SOCP). We emphasize that our proposed formulation overcomes these drawbacks. By using the large margin objective function, our formulation is expected to be more robust.

IV. Properties

We show some properties of the method, its special cases, efficiency, interpretations and comparisons with other methods.

1) The method estimates labels using margin-based loss or anything else, which is expected to be more robust than square loss-based ones.
2) The method incurs an expense of an additional parameter, i.e., trade-off parameter between graph smoothness and label information.
3) The method is transductive, i.e., testing data should be included at training time. This is because the method is an embedding method. This is different from other methods on Euclidean spaces or kernels because these methods are projection ones. The solutions of these methods have prior parametric representations (usually in some subspaces).
4) Feature weighting (spectral transform of eigenvalues) is globally optimal (Section III-B). Our formulations directly improve from the method in [7] and have efficient solutions.

A. Special Cases

Our main contribution is to add label information into embedding process (by setting $0 < c < 1$). However, even without label information, our proposed method has the following special cases.

1) Nearest neighbors (NN): having $c \to 0$ makes these combinations converge to the case of embedding the whole class to a single point. In this case, large margin classifiers become NN algorithm, a test point is classified to the class that the most of its neighbors belong to. A test point makes the embedding along its adjacent edges. If most of the adjacent edges link to a class, then the test point is closest to the collapsing point of that class.
2) Large margin spectral transform (LMST): having $c = 1$ makes our method become LMST of the graph Laplacians. This is a learning version of methods in [3]–[6], weighting each eigenvector optimally to optimize the margins of classifiers. This is supposed to be an improvement from the method in [7], replacing the theoretically not sound KTA criterion with the theoretically sound margin-based one. By having a margin reformulation instead of KTA in [7], LMST is theoretically sound. LMST alone can be regarded as a contribution of our method, but we use it as a baseline to assess the merit of our main contribution, adding label information into the embedding process.

B. Efficiency

The computational time of the method is the sum of the time of a generalized eigenvalue problem and a MKL problem. While the worst case could be expensive, for the real data, our proposed method is very efficient for the following reasons.

1) The first step can be made very efficient by using some simple tricks. It is a generalized eigenvalue problem on non-sparse matrices. However, the graphs are usually sparse in our applications or in other methods (such as NN graphs). Therefore, computing $f^T L f$ is efficient using sparse matrix representation. Y is not sparse but has a structure (of rank 1). $f^T Y f$ has a closed form and can be computed efficiently. All in all, extracting eigenvectors and eigenvalues in (12) and (13) is very efficient (in comparison with general generalized eigenvalue problem) using the functional form of the numerators and denominators of our proposed objective functions. The eigen-decomposition procedure from [23] is used to take advantage of the functional form of the matrices. The overall running time is mainly on the second step of the method.
2) The overall computational time is then dominated by the complexity of the second step, which is a MKL problem. We refer to the solution in [12], which is a semi-indefinite linear program. As in [12], the method is efficient and scales to massive datasets.

In term of computational complexity, in comparison with PCA or LDA-based feature extraction methods, our method does not incur additional expenses for sparse networks. The reason is that computational burden of our method and PCA/LDA-based ones are all eigen-decomposition. As analyzed above, even our method results in eigen-decomposition problem of a non-sparse matrices, we use the functional form of matrices to take advantage of its special structure to ensure the same computational complexity. Given that most of the time are consumed by the common learning step, the difference in term of computational complexity of our method and PCA/LDA-based ones is negligible.

For dense networks without any special network structure, all the methods are based on eigen-decomposition of dense matrices, requiring the same computational burden.

C. PCA/LDA Combination Interpretation

As in [24], Laplacian eigenmaps are interpreted as a PCA way of embedding graphs, which coincides with the smoothness feature objective of DisEmp in (9). The same argument can be applied here to show that our DisEmp method consists of different variations of combining PCA and LDA for some

TABLE I
COMPARING APPROACHES IN TERMS OF LOSS FUNCTIONS AND OPTIMIZATION PROBLEMS. MOST METHODS USE ON QUADRATIC LOSS FUNCTION TO BE EFFICIENT

Methods	Loss function	Optimization Problem
Tsuda's [8]	square	LP
Zhu's [7]	KTA	SOCP
Zhou's [9]	square	QP
Argyriou's [10]	margin/any	QP
DisEmp	margin/any	SILP

specific graph kernels. In fact, LDA- and PCA-based feature extractions are just special cases.

Proposition 1: The formulation in (9) is equivalent to kernel PCA in terms of feature extraction.

Proof: The features extracted from the formulation in (9) are the eigenvectors of the graph Laplacian L. Setting $K = L^{-1}$, then K has exactly the same eigenvector set (except for the trivial ones) with eigenvalues are inversed. Hence, eigenvectors with smallest eigenvalues of L are the eigenvectors with the largest eigenvalues of K. Therefore, this formulation gives the same result (feature set) as kernel PCA on the kernel K [25]. The same argument can be found in [24]. ■

Proposition 2: The formulation in (8), namely variance to distance formulation, is equivalent to kernel LDA in terms of feature extraction.

Proof: Using the same argument as in [24], we take $K = L^{-1}$ as a kernel. Having $L\mathbf{1} = \mathbf{0}$ where $\mathbf{1} = \{1, 1, \ldots 1\}^T \in \mathbb{R}^n$, $\mathbf{0} = \{0, 0, \ldots 0\}^T \in \mathbb{R}^n$, then $K\mathbf{1} = \mathbf{0}$, meaning data are centered from this kernel. This means that any projection of data from this kernel, f, we have $f^T\mathbf{1} = 0$.

Given that f is any projection of data from the kernel K as defined above, from the formulation in (8), the numerator is $f^T f = \sum_i f_i^2$, which is a scale of the variance of the embedded data (because $\sum_i f_i = 0$). This is the numerator.

The denominator is

$$f^T Y f = \left(\frac{\sum_{y_i=+1} f_i}{n_+} - \frac{\sum_{y_i=-1} f_i}{n_-} \right)^2$$

which is the between class variance of the embedded data.

Therefore, the formulation in (8) is equivalent to LDA feature extraction on a kernel induced from the graph. ■

Corollary 1: The multiplicative, subtractive, and additive DisEmp are different alternatives to combine the objectives of PCA and LDA for feature extraction.

From a multiobjective optimization viewpoint, these formulations are just different ways of combining objective functions. The characteristics of multiplicative DisEmp is that only some specific parameter settings of multiplicative DisEmp can have efficient solutions. Other formulations allow continuous parameter variations without breaking the problem structure, therefore giving more flexibility.

D. Discussion

Our problem setting is different from other approaches: Laplacian support vector machines [2], local fisher discriminant analysis [26], marginal fisher analysis [27], [28], local discriminant embedding [29], their extensions and variations [30]–[32], that all these methods use a primary information source of nodes represented in some Euclidean spaces or by kernels. The difference to our problem setting is that these methods restrict the nodes to have a parametric representation, while our setting is not restricted to that. For this reason, all these methods are for projections, with graph embedding as an additional information. Our approach is for the case when only graph structures exist, and therefore, is purely a graph embedding technique. Another difference is that these approaches are either equivalent to either PCA or LDA while our method compromises both objectives. At the expense of a trade-off parameter, our method has the potential of finding the most suitable compromise for the highest performance.

Our method is different from unsupervised methods such as in [4] and [7] using graph Laplacians, having the identity matrix I in place of Y. It is different from [26] and [33] as these methods are for data in Euclidean spaces, which can be reformulated for kernels. These methods are not applicable for the case having only graph structures among data objects.

We summarize objective functions used in different methods in Table I. The optimization problems are linear programming (LP), quadratic programming (QP), SOCP, and semi-indefinite linear programming (SILP).

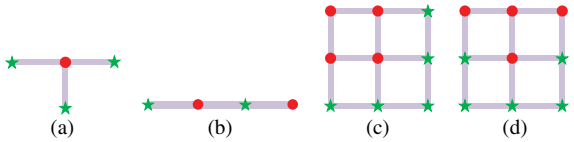

Fig. 3. Multimodal graphs: each class consists of several clusters. Clusters from the same class are not directly connected to each other, but connected via clusters of the other classes. (a) T junction. (b) Linear. (c) L shape. (d) T shape.

V. SIMULATION

We design some simulations on toy graph data with clear structure interpretations to show the scenarios where discriminative embedding methods are supposed to be useful for a discriminative purpose. The key idea in these simulations is the different graph topologies.

A. Simulation Setting

In all the following experiments, as depicted in the figures, a ball or star means a fully connected cluster of 20 nodes with an equal weight w. A gray line between balls means a full bipartite graph between the two balls, i.e., any node in each cluster connects to any node in the other cluster with weight $1-w$. The experiments are done by 80%/20% train/test splits and repeated 50 times. The means of the areas under the receiver operating characteristic curve (AUC scores) are reported. Graphs are embedded into a 3-D space ($m = 3$). We vary the values of within cluster weight w. We also vary trade-off parameters c. We use the additive DisEmp version.

B. Multimodal Case

We simulated different topologies of graphs as shown in Fig. 3(a) and (b). The idea of these topologies is that each

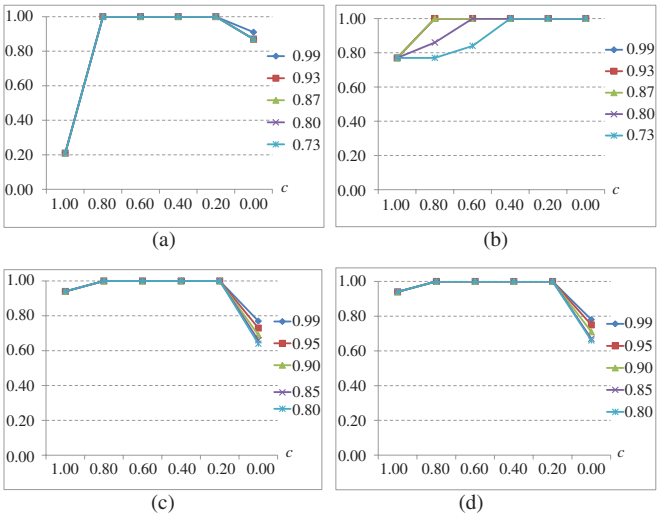

Fig. 4. AUC scores of different graph topologies. The legends are different weight w values. Vertical axes are AUC scores. Horizontal axes show different c values. These figures show that different trade-off c values give different AUC scores, and the highest performances happen at $0 < c < 1$. (a) T junction. (b) Linear. (c) L shape. (d) T shape.

class consists of several clusters dispatched to different places, not directly connected to each other. We name this type of topology *multimodal*. When the inter-cluster weight is high enough, we expect the label propagation algorithm to be able to correctly predict labels in these graphs.

Average AUC scores are visualized in Fig. 4(a) and (b). Note that horizontal axes are different trade-off parameter c values. As we can see, setting $c = 0$, which is equivalent to Laplacian eigenmaps followed by a supervised feature weighting (by MKL), did not give a high performance. By utilizing label information in the embedding (setting $c < 1$), data from the same class in the Euclidean space were pulled together, making the embedding easier to be learned (giving very high AUC scores). As a result, using label information proves to be essential in these cases.

Remark 1: The method shows advantage when clusters of one class are distributed to different places, i.e., the class is multimodal. They are not directly connected, but through clusters of the other class.

C. Complicated Boundary Case

We also simulated different topologies as shown in Fig. 3(c) and (d). The key idea is that in these topologies, even the whole class is connected, but as in the figure, the boundary between the two classes is complicated. These topologies require a *nonlinear* function in the figure to separate them. We expect that in the embedding space, a nonlinear decision function is also required to classify the two classes to achieve the highest result.

From the results in Fig. 4(c) and (d), we see that without using label information ($c = 1.00$), i.e., Laplacian eigenmap followed by MKL, the performance was not the highest. By using some label information (by setting $c < 1$) together with graph smoothness, our DisEmp method achieved the maximum performance.

Remark 2: Even though clusters of the same class are connected, given a complicated connectivity to clusters of the other class (*nonlinear* boundary in the figures), the method is helpful to increase performance.

In conclusion, in these simulations, we have shown that in these multimodal and *nonlinear* boundary cases, having label information is beneficial to increase performance. This is a result of using label information to have data of the same class closer to each other, making the classification problem easier. This means that our method of discriminative embedding of graph has more discriminative power compared to the method not considering label information. This also signifies the use of label information in embedding graphs for a discriminative purpose.

VI. Application: Gene Function Prediction

We applied the method to the problem of predicting gene function. We used two biological networks of Saccharomyces cerevisiae, gene network and protein-protein interaction (PPI) network. The three gene function categories: amino acid metabolism, carbohydrate metabolism, and nucleotide metabolism were selected for having the largest numbers of genes. We used only the largest connected component from original networks to form the final graphs.

We carried out gene function prediction from these networks. Due to class overlapping, we made three *one versus the rest* classification problems. The experiment setting was as follows. The whole network was split with 90% of the number of nodes for training and the rest for testing. We ran this train/test split 50 times. We used mean AUC scores over 50 runs as the performance measure. In all following experiments, we used five eigenvectors associated with five smallest eigenvalues. At the learning step using MKL, we chose regularization parameter C from the set $\{0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500\}$, and selected the parameter value with the highest mean AUC score. As for trade-off parameter c, we varied it from 1 to 0.001 and showed AUC scores for each parameter value.

A. Gene Network

We show an application of gene function prediction from a gene network. We created a gene network from Kyoto Encyclopedia of Genes and Genomes (KEGG) metabolic pathways [34]. A gene was connected to another gene if both of them catalyze the same chemical compound, making an undirected edge with weight 1: 1) we obtained a gene network of 422 genes, in which 192 are for amino acid; 2) 173 are for carbohydrate; and 3) 110 are for nucleotide metabolism. There are 39 genes are shared between 1) and 3), 7 are shared between 3) and each of 1) and 2).

Subtractive DisEmp version (13) of graph smoothness and label information result is presented in Fig. 5(a). We can see that Laplacian eigenmap followed by a margin-based weighting, i.e., $c = 1$, did not give a very high classification performance (0.86, 0.77, and 0.91). By adding some label information ($c < 1$), AUC scores reached their peaks. In particular, the scores could be very close to 1.00 for nucleotide.

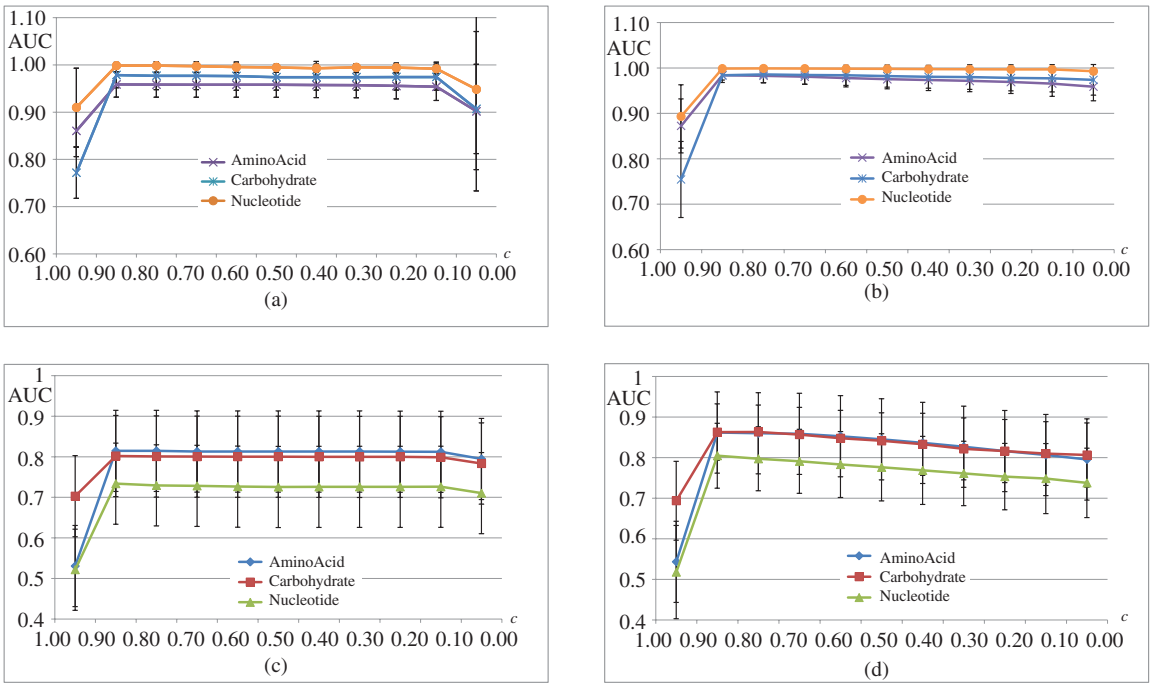

Fig. 5. Results of DisEmp on gene and PPI networks. Vertical axes are AUC scores. Horizontal axes show different c values. Different trade-off c values give different AUC scores, and the highest performances happen at $0 < c < 1$. (a) Subtractive DisEmp on gene network. (b) Additive DisEmp on gene network. (c) Subtractive DisEmp on PPI network. (d) Additive DisEmp on PPI network.

For any problem, by selecting the most suitable parameter c ($0 < c < 1$), AUC scores became be significantly higher. Another point is that AUC scores became very high right after c dropped from 1. This means that only a little label information could significantly improve the discriminative performance. As c went to very close to 0, mean AUC scores dropped significantly and standard deviations increased. This means that graph structure is a vital information source and the learned label function must obey the graph structure to certain extent, that is, it must be reasonably smooth.

The result of the additive DisEmp version (12) is presented in Fig. 5(b). The same conclusion can be induced as in the subtractive DisEmp version. Using only Laplacian eigenmaps with spectral transforms gave only 0.87, 0.75, and 0.89 mean AUC scores for the three problems. By adding some label information into the embedding ($c < 1$), mean AUC scores went up to 0.98, 0.99, and 1.00. This shows that adding label information significantly improves its performance. This also justifies the combination of both graph smoothness and label information to embed graphs for a discriminative purpose. It is also the case that one must not use very little graph information, meaning that c should not be set close to 0. One minor difference to the subtractive one was that additive DisEmp tended to give a slightly more reliable performance in the sense that standard deviations were smaller and peak performances happened at a larger range of trade-off parameter c.

B. PPI Network

We extracted the PPI network as in [1]. In total, we obtained a graph of 413 proteins: 1) there are 84 proteins for amino acid; 2) 144 proteins for carbohydrate; and 3) 98 proteins for nucleotide. The remaining proteins do not belong to these large categories. The performance of the subtractive DisEmp version is shown in Fig. 5(c) and of the additive DisEmp version is shown in Fig. 5(d).

TABLE II

COMPARISON WITH BASELINES. c PARAMETER OF (ADDITIVE) DISEMP IS SET BY CROSS-VALIDATION ON TRAINING DATA. THREE AUC VALUES IN EACH CELL ARE THE MEAN VALUES FOR AMINO ACID, CARBOHYDRATE, AND NUCLEOTIDE CATEGORIES

	LMST	DisEmp
Gene Network	0.87 (±0.06)	0.98 (±0.01)
	0.75 (±0.08)	0.98 (±0.02)
	0.89 (±0.07)	1.00 (±0.00)
PPI Network	0.54 (±0.09)	0.86 (±0.06)
	0.69 (±0.10)	0.86 (±0.07)
	0.52 (±0.11)	0.80 (±0.08)

From the tables, we can also obtain similar conclusions as for the gene networks. Without label information embedding ($c = 1$), both methods performed poorly. Take additive DisEmp for example, AUC scores of 0.54, 0.69, and 0.52 were reported for the three classes. However, using an appropriate amount of label information, AUC scores reached 0.86, 0.86, and 0.80 correspondingly.

We also observe that on the PPI network, compared to the gene network, our proposed method gave lower means and higher standard deviations. This is possible because PPI network is more unreliable. Gene network from KEGG metabolic pathways is, on the other hand, well studied and reliable.

C. Comparison

As DisEmp is supposed to improve from feature extraction methods by putting label information into the embedding

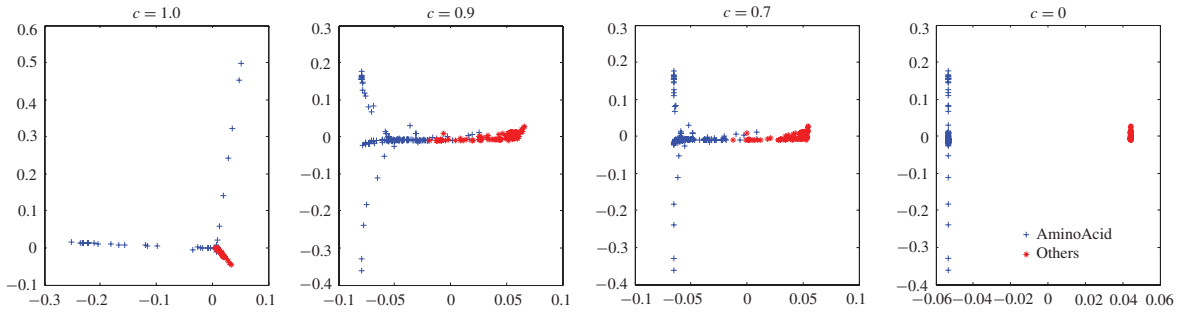

Fig. 6. Embedding of amino acid metabolism gene classification at different c parameters. The leftmost figure is the embedding of Laplacian eigenmap. The rightmost one is the embedding with very little graph smoothness.

TABLE III
EMPIRICAL RUNNING TIMES (IN SECONDS) OF THE TWO STEPS

	Step 1	Step 2
Gene network	1.61 ± 0.03	19.55 ± 6.95
PPI network	1.50 ± 0.03	10.00 ± 6.68

process, we compared DisEmp to LMST. The reason is that LMST does not take label information into the embedding step. It is noteworthy that LMST is a flexible learning version of [3]–[6]. It is also a large-margin version of [7]. LMST is supposed to be more robust to KTA-based as in [7] because of the theoretical soundness of margin criterion.

From Table II for both data sets, we can see that DisEmp outperforms LMST. This means that discriminative embedding, compromising graph smoothness, and discriminative information, usually gives a higher performance should the trade-off parameter is set properly.

D. Effect of Trade-Off Parameter

We show the effect of trade-off parameter c to the embedding of the graph in Fig. 6. For visualization purpose, we used all data for training and plotted the distribution in a 2-D space spanned by two eigenvectors with smallest eigenvalues (excluding the trivial eigenvector, the one with a constant coordinate for all data points). It is expected that not using label information ($c = 1$), we can see that the two classes were fused with each other. As more label information was added, we can see that the two classes were pulled away from each other. To the extreme $c = 0$, the embedding does not respect the graph structure, the two classes were completely separated from each other. What we expect is that some values $0 < c < 1$ should give the *right* trade-off between smoothness on the graph and label information, giving the highest prediction performance.

E. Running Times of the Steps

To demonstrate that the running time of the method depends mainly on the running times of the step 2 (analyzed in Section IV-B), we show the actual running times of the two steps in Table III. We could observe that the step 2 was much more expensive.

VII. CONCLUSION

We have proposed a method to propagate labels of nodes on a graph by embedding the graph into a Euclidean space in a discriminative way. By incorporating training label information, the embedding step compromises between smoothness of labels on the graph and discriminative information. The embedding was interpreted as combining PCA and LDA feature extractions for graphs. From the embedding space, a globally optimal learning method that optimizes the margin for classification was formulated. The embedding step was made efficient by formulating it into a generalized eigenvalue problem taking only a few eigenvectors with smallest eigenvalues. By using an embedding step, the method is also efficient as the learning step inherits solutions from MKL problems. Using a margin-based objective function, the method is robust and theoretically sound. Simulations showed that the method was beneficial on multimodal graphs or the graphs with complicated between-class graph structures. We showed an application of predicting gene metabolism function from a gene network and a PPI network. Our proposed method achieved a significantly higher performance compared to baselines. This shows that by putting label information into the embedding process, the distribution becomes easier to be learned. This justifies our method to embed the graph with label information for discriminative tasks.

ACKNOWLEDGMENT

The authors are grateful to laboratory members for their advice. They acknowledge the editors and reviewers for constructive comments.

REFERENCES

[1] C. Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork, "Comparative assessment of large-scale data sets of protein–protein interactions," *Nature*, vol. 417, no. 6887, pp. 399–403, May 2002.

[2] 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, Nov. 2006.

[3] A. J. Smola and R. I. Kondor, "Kernels and regularization on graphs," in *Proc. COLT*, 2003, pp. 144–158.

[4] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.

[5] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 315–322.

[6] K. Tsuda and W. S. Noble, "Learning kernels from biological networks by maximizing entropy," *Bioinformatics*, vol. 20, no. 1, pp. 326–333, 2004.
[7] X. Zhu, J. S. Kandola, Z. Ghahramani, and J. D. Lafferty, "Nonparametric transforms of graph kernels for semi-supervised learning," in *Advances in Neural Information Processing System*. Cambridge, MA: MIT Press, 2004, pp. 1–8.
[8] K. Tsuda, H. Shin, and B. Schölkopf, "Fast protein classification with multiple networks," *Bioinformatics*, vol. 21, no. 2, pp. 59–65, 2005.
[9] D. Zhou and C. J. C. Burges, "Spectral clustering and transductive learning with multiple views," in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, 2007, pp. 1159–1166.
[10] A. Argyriou, M. Herbster, and M. Pontil, "Combining graph Laplacians for semi-supervised learning," in *Advances in Neural Information Processing System*. Cambridge, MA: MIT Press, 2005, pp. 1–8.
[11] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. San Diego, CA: Academic, 1990.
[12] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *J. Mach. Learn. Res.*, vol. 7, pp. 1531–1565, Jul. 2006.
[13] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI: AMS, 1997.
[14] M. Hein, J.-Y. Audibert, and U. Luxburg, "Graph Laplacians and their convergence on random neighborhood graphs," *J. Mach. Learn. Res.*, vol. 8, pp. 1325–1370, May 2007.
[15] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing System*. Cambridge, MA: MIT Press, 2003, pp. 1–8.
[16] U. Luxburg, M. Belkin, and O. Bousquet, "Consistency of spectral clustering," *Ann. Stat.*, vol. 36, no. 2, pp. 555–586, Apr. 2008.
[17] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. S. Kandola, "On kernel-target alignment," in *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press, 2001, pp. 367–373.
[18] C. H. Nguyen and T. B. Ho, "Kernel matrix evaluation," in *Proc. Int. Joint Conf. Artif. Intell.*, 2007, pp. 987–992.
[19] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CALD-02-107, Jun. 2002.
[20] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 985–992.
[21] X. Ji and W. Xu, "Document clustering with prior knowledge," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2006, pp. 405–412.
[22] X. Ling, W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Spectral domain-transfer learning," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2008, pp. 488–496.
[23] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM J. Sci. Comput.*, vol. 23, no. 2, pp. 517–541, 2001.
[24] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 47.
[25] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Mar. 1998.
[26] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1027–1061, May 2007.
[27] S. Yan, D. Xu, B. Zhang, and H.-J. Zhang, "Graph embedding: A general framework for dimensionality reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Washington D.C., Jun. 2005, pp. 830–837.
[28] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
[29] H.-T. Chen, H.-W. Chang, and T.-L. Liu, "Local discriminant embedding and its variants," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Washington D.C., Jun. 2005, pp. 846–853.
[30] Y. Fu, S. Yan, and T. S. Huang, "Correlation metric for generalized feature extraction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 12, pp. 2229–2235, Dec. 2008.
[31] X. Liu, S. Yan, and H. Jin, "Projective nonnegative graph embedding," *IEEE Trans. Image Process.*, vol. 19, no. 5, pp. 1126–1137, May 2010.
[32] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *IEEE Trans. Neural Netw.*, vol. 20, no. 4, pp. 729–735, Apr. 2009.
[33] X. He and P. Niyogi, "Locality preserving projections," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.
[34] M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi, "KEGG for linking genomes to life and the environment," *Nucl. Acids Res.*, vol. 36, no. 1, pp. 480–484, 2008.

Canh Hao Nguyen received the B.S. degree from the University of New South Wales, Sydney, Australia, in 2002, and the M.S. and Ph.D. degrees from Japan Advanced Institute of Science and Technology, Ishikawa, Japan, in 2006 and 2009, respectively.

He is currently a Post-Doctoral Fellow in the Japan Society for the Promotion of Science, Kojimachi, Japan. His current research interests include machine learning methods with applications in network and biological data.

Hiroshi Mamitsuka received the B.S. degree in biophysics and biochemistry, the M.E. degree in information engineering, and the Ph.D. degree in information sciences, all from the University of Tokyo, Tokyo, Japan, in 1988, 1991, and 1999, respectively.

He has been working in machine learning, data mining, and bioinformatics. His current research interests include mining from graphs and networks in biology and chemistry.