Multiple Graph Label Propagation by Sparse Integration

Masayuki Karasuyama and Hiroshi Mamitsuka

Abstract— **Graph-based approaches have been most successful in semisupervised learning. In this paper, we focus on label propagation in graph-based semisupervised learning. One essential point of label propagation is that the performance is heavily affected by incorporating underlying manifold of given data into the input graph. The other more important point is that in many recent real-world applications, the same instances are represented by multiple heterogeneous data sources. A key challenge under this setting is to integrate different data representations automatically to achieve better predictive performance. In this paper, we address the issue of obtaining the optimal linear combination of multiple different graphs under the label propagation setting. For this problem, we propose a new formulation with the sparsity (in coefficients of graph combination) property which cannot be rightly achieved by any other existing methods. This unique feature provides two important advantages: 1) the improvement of prediction performance by eliminating irrelevant or noisy graphs and 2) the interpretability of results, i.e., easily identifying informative graphs on classification. We propose efficient optimization algorithms for the proposed approach, by which clear interpretations of the mechanism for sparsity is provided. Through various synthetic and two real-world data sets, we empirically demonstrate the advantages of our proposed approach not only in prediction performance but also in graph selection ability.**

Index Terms— **Graph-based semisupervised learning, label propagation, multiple graph integration, sparsity.**

I. Introduction

Graph-based semisupervised learning [1]–[9] has received significant attention in machine learning and has been widely used in many practical applications, because of the flexibility and easiness of implementation. In the graph-based semisupervised learning, the key idea is so-called manifold assumption, in which instances connected by large weights on the graph have similar labels. A well accepted approach in this learning is label propagation (e.g., [4], [5], and [9]), which propagates the labels of each node in a given graph to their neighboring nodes according to their similarity and is formulated as the simple least squares problem.

The usefulness of the label propagation algorithms has been demonstrated so far, but their performance highly depends on the way of generating the input graph. That is, we need to choose the similarity measure (or the distance function) that determines the edges of the graph, and the distance function itself has the parameters to be tuned. More importantly, in many practical applications, we have a number of different graph data sources. For example, in gene function prediction, various information sources are available such as gene expression, gene sequences, and subcellular localization, which can all be given as graphs. We, however, cannot see the most important graph for prediction. In this paper, we address the issue of integrating multiple graphs under the label propagation framework. For this issue, we attempt to build a formulation and optimization algorithm, by which we do not need to choose one specific similarity measure and one data source a priori. Instead, our approach automatically estimates the optimal combination of given multiple graphs.

We can find some studies which considered the same problem in different applications and demonstrated that integrating multiple graphs improve the prediction performance. For example, in bioinformatics, a label propagation method of combining multiple graphs for protein function prediction is proposed [10]. This approach is however not robust against noisy graphs, and a more robust probabilistic model using Student-*t* distribution is proposed [11]. Similarly, in multimedia content analysis, a method for integrating multiple graphs for the video annotation problem proposed [12]. We will describe many existing methods (including the above methods) in more details in Section IV.

We propose a new approach for the issue of integrating multiple graphs under the label propagation framework. As already done by the most existing methods, our approach also combines multiple graph Laplacian matrices linearly and estimates their weight coefficients. However, our unique property is the sparsity of graph weights. That is, graph weights of our approach can be sparse, meaning that only a part of weights has nonzero values and the rest are equal to exactly 0. This important property provides the following two advantages.

1) In general, eliminating irrelevant or noisy graphs in integrating multiple graphs improves the classification performance. Conventional approaches however have cases of assigning nonzero weights to graphs which are irrelevant to classification (we define such graphs as irrelevant graphs), by which prediction performance is deteriorated, as irrelevant graphs are kept as the input. On the other hand, our sparseness property allows us to eliminate irrelevant graphs completely because their weights are estimated at zero.
2) Sparse weighted coefficients allow to identify the graphs which are important (or not needed) for classification easily. Furthermore, our formulation provides a clear

Manuscript received March 19, 2012; revised January 28, 2013; accepted June 22, 2013. Date of publication July 23, 2013; date of current version November 1, 2013.

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

Digital Object Identifier 10.1109/TNNLS.2013.2271327

interpretation of the mechanism of sparsity, and it also offers a kind of grouping effect, which is similar to that given by the elastic net [13].

The remainder of this paper is structured as follows. In Section II, we provide a short review on the standard label propagation algorithm for the single graph input. Section III introduces our proposed approach for the problem of integrating multiple graphs in the framework of label propagation. Section IV describes the relationships with several existing approaches. In Section V, we present our experimental results obtained by a variety of data sets including synthetic and real data sets, demonstrating the performance advantage of the proposed approach over other methods from various viewpoints. Finally, Section VI concludes the paper.

II. Label Propagation Algorithm

The label propagation algorithm propagates label information from labeled nodes to unlabeled nodes according to the edges. We here briefly review the standard formulation of the label propagation algorithm, following the local and global consistency method [5].

Suppose that we have n data points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell, \boldsymbol{x}_{\ell+1}, \ldots, \boldsymbol{x}_n\}$ in which the first ℓ data points are labeled by $\{y_1, \ldots, y_\ell\}$, where $y_i \in \{\pm 1\}$. The remaining $n - \ell$ data points are unlabeled. Our goal is to predict the labels of these unlabeled data points: $\{y_{\ell+1}, \ldots, y_n\}$. Let $\mathcal{G}$ be an undirected graph, where each node (or vertex) corresponds to each data point in $\mathcal{X}$. The graph $\mathcal{G}$ is represented by the affinity matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ where (i, j)-element W_{ij} is the strength of the edge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

The label propagation algorithm (e.g., [4], [5], and [9]) estimates the label of each node, based on the smoothness assumption on the graph. Concretely, the algorithm estimates scores $\{f_i\}_{i=1}^n$ for each node by solving the following regularized least squares problem:

$$\min_{\{f_i\}_{i=1}^n} \sum_{i,j}^n W_{ij} \left(\frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}} \right)^2 + \lambda \sum_{i=1}^n (y_i - f_i)^2$$

where $D_{ii} = \sum_{j=1}^n W_{ij}$, $y_i = 0$ $(i = \ell + 1, \ldots, n)$ and λ is a positive constant. The first term represents a penalty for the smoothness of the scores $\{f_i\}_{i=1}^n$ on the graph and the second term is a penalty for the inconsistency with the initial labeling. Using matrix-vector notation, the above problem is written as

$$\min_{\boldsymbol{f}} \ \boldsymbol{f}^\top \boldsymbol{L} \boldsymbol{f} + \lambda \|\boldsymbol{y} - \boldsymbol{f}\|_2^2$$

where $\boldsymbol{f} = (f_1, \ldots, f_n)^\top$ and $\boldsymbol{L}$ is the so-called graph Laplacian matrix defined by $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2} \boldsymbol{W} \boldsymbol{D}^{-1/2}$ ($\boldsymbol{D}$ is a diagonal matrix with the ith diagonal entry D_{ii}). Throughout the paper, we use this symmetric normalized version of graph Laplacian matrix. The other variants of the graph Laplacian matrices can also be used such as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ and $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1} \boldsymbol{W}$.

III. Integrating Multiple Graphs for Label Propagation Algorithm

We introduce our new approach of combining multiple graphs under the label propagation framework. After defining the problem formulation, we provide efficient optimization algorithms and then discuss several interesting properties of our approach.

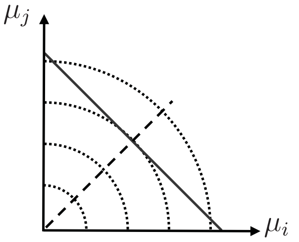


Fig. 1. Contour plot of the regularizer for $\boldsymbol{\mu}$ along with the feasible region (solid line) of the optimization problem given by (3). The dashed line is $\mu_i = \mu_j$. The minimizer of $\|\boldsymbol{\mu}\|_2^2$ in the feasible region is the intersection of the solid line with the dashed line. Thus, if we use large λ_2, the optimal solution of the problem given by (3) should be close to $\boldsymbol{\mu} = \boldsymbol{1}/K$.

A. Formulation

Suppose that we have K graphs $\{\boldsymbol{W}^{(k)}\}_{k=1}^K$ where (i, j)-entry is $W_{ij}^{(k)}$ and $\{\boldsymbol{L}_k\}_{k=1}^K$ is a set of the corresponding graph Laplacian matrices. To integrate the multiple graphs, we first consider the following minimization problem:

$$\begin{aligned} \min_{\boldsymbol{f}, \boldsymbol{\mu}} \quad & \sum_{k=1}^K \frac{\mu_k}{Z_k} \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} + \lambda \|\boldsymbol{y} - \boldsymbol{f}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{\mu}^\top \boldsymbol{1} = 1, \ \boldsymbol{\mu} \geq 0 \end{aligned}$$

where Z_k is a normalization constant that is defined by $Z_k = \|\boldsymbol{L}_k\|_F$. The first term linearly combines K graphs using weighting coefficients $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)^\top$ and $Z_k (k = 1, \ldots, K)$. The normalization is justified by $\boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} = \langle \boldsymbol{L}_k, \boldsymbol{f}\boldsymbol{f}^\top \rangle_F$, where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product that induces the Frobenius norm $\| \cdot \|_F = \sqrt{\langle \cdot, \cdot \rangle_F}$. Similar normalizations have also been employed in the context of multiple kernel learning (MKL) [14], [15]. Although this formulation seems to be natural, $\boldsymbol{\mu}$ has only one nonzero element at the stationary point. That is, the minimum in terms of $\boldsymbol{\mu}$ is attained by $\boldsymbol{\mu} = \boldsymbol{e}_{k_{\min}}$, where $\boldsymbol{e}_i$ is the ith unit vector (i.e., a vector with a one in the ith element and zeros elsewhere) and $k_{\min} = \arg\min_k 1/Z_k \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f}$, meaning that only one graph is selected from given multiple graphs but this is not always desirable.

To relax the strong sparsity on selecting graphs, we introduce an additional regularization term for $\boldsymbol{\mu}$ as follows:

$$\begin{aligned} \min_{\boldsymbol{f}, \boldsymbol{\mu}} \quad & \sum_{k=1}^K \frac{\mu_k}{Z_k} \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} + \lambda_1 \|\boldsymbol{y} - \boldsymbol{f}\|_2^2 + \frac{\lambda_2}{2} \|\boldsymbol{\mu}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{\mu}^\top \boldsymbol{1} = 1, \ \boldsymbol{\mu} \geq 0 \end{aligned} \tag{1}$$

where $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are the regularization parameters. Fig. 1 schematically illustrates the effect of this additional regularization term. If we set $\lambda_2 = \infty$, the solution is the uniform weight $\boldsymbol{\mu} = \boldsymbol{1}/K$. On the other hand, if we set $\lambda_2 = 0$, the solution is $\boldsymbol{\mu} = \boldsymbol{e}_{k_{\min}}$. As we will see later, between these two extremes $\lambda_2 \in (0, \infty)$, we can obtain sparse weighting coefficients.

Algorithm 1 Alternating minimization algorithm for proposed approach.

```
1: Input: $\{\boldsymbol{L}_k\}_{k=1}^K$, $\lambda_1$, $\lambda_2$
2: Output: $\boldsymbol{f}$, $\boldsymbol{\mu}$
3: Initialize $\boldsymbol{\mu}$ (e.g., $\boldsymbol{\mu} \leftarrow \mathbf{1}/K$)
4: Optimize $\boldsymbol{f}$ according to (2)
5: Optimize $\boldsymbol{\mu}$ (e.g., by Algorithm 2)
6: Repeat step 2 and step 3 until convergence
```

Algorithm 2 Algorithm to optimize $\boldsymbol{\mu}$

```
1: Input: $\boldsymbol{v}$ (sorted in increasing order), $\lambda_2$
2: Output: $\boldsymbol{\mu}$
3: for $m \leftarrow 1$ to $K$ do
4:     $\eta \leftarrow (\lambda_2 + \sum_{k=1}^m v_k)/m$
5:     if $m = |\{k \mid \eta - v_k > 0,\ k = 1, \ldots, K\}|$ then
6:         break
7:     end if
8: end for
9: $\mu_k \leftarrow (\eta - v_k)/\lambda_2$ for $k = 1, \ldots, m$, and
10: $\mu_k = 0$ for $k = m + 1, \ldots, K$
```

B. Optimization

To solve the optimization problem given by (1), we alternately minimize the objective function with respect to $\boldsymbol{f}$ and $\boldsymbol{\mu}$ (Algorithm 1 provides the outline of our algorithm). First, we assume that $\boldsymbol{\mu}$ is fixed. We can then obtain the optimal $\boldsymbol{f}$ for the fixed $\boldsymbol{\mu}$ by the following linear system:

$$\boldsymbol{f} = \lambda_1 \left(\lambda_1 \boldsymbol{I} + \textstyle\sum_{k=1}^K \frac{\mu_k}{Z_k} \boldsymbol{L}_k\right)^{-1} \boldsymbol{y}. \tag{2}$$

On the other hand, if we fix $\boldsymbol{f}$, the optimal $\boldsymbol{\mu}$ is the solution of the following problem:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \boldsymbol{v}^\top \boldsymbol{\mu} + \tfrac{\lambda_2}{2} \|\boldsymbol{\mu}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{\mu}^\top \mathbf{1} = 1,\ \boldsymbol{\mu} \geq 0 \end{aligned} \tag{3}$$

where $\boldsymbol{v} = (v_1, \ldots, v_K)^\top$ and $v_k = 1/Z_k \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f}$. Because this problem is the convex quadratic programming (QP), it is solved by any QP solvers. However, we can derive easier ways to solve this problem using an analytical form of the optimal solution. Without loss of generality, we may assume that $\{v_k\}_{k=1}^K$ are sorted in increasing order: $v_1 \leq v_2 \leq \ldots \leq v_K$. Then, the following theorem gives analytic representations of the optimal solution for the problem given by (3).

Theorem 1: The optimal solution of the problem given by (3) is written as follows:

$$\mu_k = \frac{\eta - v_k}{\lambda_2}, \quad \text{for } k = 1, \ldots, m, \tag{4}$$

$$\mu_k = 0, \quad \text{for } k = m + 1, \ldots, K \tag{5}$$

where

$$\eta = \frac{\lambda_2 + \sum_{k=1}^I v_k}{m}, \tag{6}$$

and

$$m = |\{k \mid \eta - v_k > 0,\ k = 1, \ldots, K\}|. \tag{7}$$

The proof of this theorem is verified by that (4)–(7) are equivalent to the Karush–Kuhn–Tucker (KKT) conditions (see [16]) for the problem given by (3) (The proof is in Appendix B).

From Theorem 1, we see the following two important properties of our formulation.

1) The optimal $\boldsymbol{\mu}$ has only m nonzero elements and the remaining elements are exactly 0.
2) If we know the optimal m, we can calculate the optimal $\boldsymbol{\mu}$ analytically.

Thus determining the number of nonzero elements m is important for our approach. To find the optimal m, a simple way is to check the case of $m = 1$ to K one by one, as shown in Algorithm 2. Practically this naive manner of Algorithm 2 is efficient enough if the number of given graphs is kept at a moderate size.

Proposition 1: Algorithm 2 can find the optimal $\boldsymbol{\mu}$ for the optimization problem (3) by $O(K^2)$ computations.

Proof: Because each iteration of **for** loop in Algorithm 2 takes $O(K)$ computations to count the number of positive elements in $\eta\mathbf{1} - \boldsymbol{v}$ (line 5), the computational complexity of the entire process is obviously $O(K^2)$. Thus we need to prove that the output of Algorithm 2 satisfies the optimality conditions of the problem given by (3), i.e., (4)–(7). Lines 4, 9, and 10 in Algorithm 2 are the first three equations in the KKT conditions, i.e., (4)–(6), and the last condition (7) is satisfied by the condition in line 5. Therefore, the output of Algorithm 2 satisfies optimality conditions given by (4)–(7). ■

We note that we can further develop a slightly elaborate procedure, Algorithm 3, which allows to obtain the optimal $\boldsymbol{\mu}$ by only $O(K)$ computations.[1] The proof of the computational complexity for Algorithm 3 is given by Appendix C.

Proposition 2: Algorithm 3 can find the optimal $\boldsymbol{\mu}$ for the optimization problem given by (3) by $O(K)$ computations.

Although here we describe only the case of binary classification, our formulation is applied to multiclass classification using the same way as the original label propagation [5] (see Appendix A for detail).

C. Discussion

Here we describe three interesting characteristics of our approach.

1) Sparsity: As we see in (4) and (5), the graph weight $\boldsymbol{\mu}$ has only m nonzero elements and the rest (i.e., $K - m$ elements) are exactly 0. This allows our approach to obtain sparse solutions in terms of the graph weights, while we note that this important property is not considered in any of existing approaches. Our approach automatically selects important graphs and eliminates irrelevant or noisy graphs for discrimination. In other words, we can easily interpret the resultant combined graphs.

Fig. 2 shows the optimized $\boldsymbol{\mu}$ as a function of λ_2 regularization path for λ_2) on a synthetic data set. We can see that $\boldsymbol{\mu}$

[1]The input of the procedure is the sorted $\boldsymbol{v}$, and in general, sorting $\boldsymbol{v}$ needs $O(K \log K)$ computation.

Algorithm 3 Faster algorithm to optimize $\boldsymbol{\mu}$

```
1: Input: $\boldsymbol{v}$ (sorted in increasing order), $\lambda_2$
2: Output: $\boldsymbol{\mu}$
3: $v^{(\mathrm{sum})} \leftarrow v_1$, $\eta \leftarrow \lambda_2 + v_1$
4: Find $\widehat{m}$ such that $v_{\widehat{m}} < \eta \le v_{\widehat{m}+1}$
5: if $\widehat{m} = 1$ then
6:   $\mu_1 \leftarrow 1$ and $\mu_i = 0$ for $i = 2, \ldots, K$
7:   return
8: end if
9: $isInc \leftarrow false$
10: for $m \leftarrow 2$ to $K$ do
11:   $\eta^{(\mathrm{prev})} \leftarrow \eta$, $v^{(\mathrm{sum})} \leftarrow v^{(\mathrm{sum})} + v_m$,
      $\eta \leftarrow (\lambda_2 + v^{(\mathrm{sum})})/m$
12:   if $\eta > \eta^{(prev)}$ then
13:     $isInc \leftarrow true$
14:   end if
15:   if $isInc$ then
16:     Increase $\widehat{m}$ until $v_{\widehat{m}} < \eta \le v_{\widehat{m}+1}$
17:   else
18:     Decrease $\widehat{m}$ until $v_{\widehat{m}} < \eta \le v_{\widehat{m}+1}$
19:   end if
20:   if $\widehat{m} = m$ then
21:     break
22:   end if
23: end for
24: $\mu_k \leftarrow (\eta - v_k)/\lambda_2$ for $k = 1, \ldots, m$, and
25: $\mu_k = 0$ for $k = m+1, \ldots, K$
```

has only one nonzero element for small λ_2 and all elements of $\boldsymbol{\mu}$ come to the same value for large λ_2. In between these two extremes, we obtain sparse solutions in which only a part of weights has nonzero elements. We can efficiently explore this regularization path for λ_2 using warm-start-based path-following optimization [17].

Furthermore, from Theorem 1, we can present a clear interpretation of the mechanism of sparsity in our formulation. Because $\{v_k\}_{k=1}^K$ are sorted in the increasing order, a sequence of nonzero weights $\mu_1, \mu_2, \ldots, \mu_m$ should be in decreasing order by the definition of the optimal μ_k, given by (4). Because small v_k $(= 1/Z_k \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f})$ means that the score $\boldsymbol{f}$ is smooth on the graph $\boldsymbol{W}^{(k)}$, larger weights are assigned to smoother graphs and the rest ($K - m$ graphs $\{\boldsymbol{W}^{(k)}\}_{k=m+1}^K$ with $\mu_k = 0$) are not smooth, comparing to the first m graphs $\{\boldsymbol{W}^{(k)}\}_{k=1}^m$.

2) Grouping Effect: Our approach is interpreted as automatic variable grouping similar to the one done by the elastic net [13]. In our solution for $\boldsymbol{\mu}$ in (4) and (5), if v_i and v_j have similar values, then μ_i and μ_j tend to have similar values. We can easily verify this fact from $\mu_i - \mu_j = (v_i - v_j)/\lambda_2$. Let $\tilde{\boldsymbol{L}}_k = \boldsymbol{L}_k / Z_k$. Then, if $\mu_i - \mu_j \approx 0$, $(v_i - v_j)/\lambda_2$ can be written as

$$\frac{\boldsymbol{f}^\top \tilde{\boldsymbol{L}}_i \boldsymbol{f} - \boldsymbol{f}^\top \tilde{\boldsymbol{L}}_j \boldsymbol{f}}{\lambda_2} = \frac{\langle \tilde{\boldsymbol{L}}_i - \tilde{\boldsymbol{L}}_j, \boldsymbol{f}\boldsymbol{f}^\top \rangle_F}{\lambda_2} \approx 0.$$

From this equation, we can see that if $\tilde{\boldsymbol{L}}_i$ and $\tilde{\boldsymbol{L}}_j$ are similar to each other in terms of $\langle \tilde{\boldsymbol{L}}_i - \tilde{\boldsymbol{L}}_j, \boldsymbol{f}\boldsymbol{f}^\top \rangle_F$, then $\mu_i - \mu_j \approx 0$. Furthermore, from the above equation, the following inequality

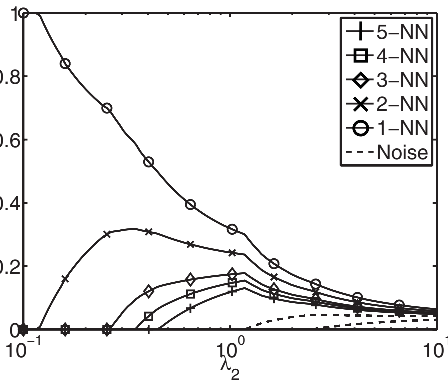


Fig. 2. Regularization path for λ_2 ($\boldsymbol{\mu}$ as a function of λ_2) on a synthetic data set with 25 graphs (i.e., $\boldsymbol{\mu} \in \mathbb{R}^{25}$). The first five graphs are generated by k-nearest neighbor (k-NN) graphs ($k = 1, \ldots, 5$, solid marked lines) and the rest are random graphs that are irrelevant to classification (dashed lines).

can be derived which also represents the grouping effect in our algorithm (see Appendix D for derivation):

$$|\mu_i - \mu_j| \le \frac{\lambda_1}{\lambda_2} \|\boldsymbol{y}\|_2^2 \sqrt{2(1-\rho)} \tag{8}$$

where $\rho = \langle \tilde{\boldsymbol{L}}_i, \tilde{\boldsymbol{L}}_j \rangle_F$. This means that two coefficients μ_i and μ_j for which graph Laplacian matrices are highly correlated in terms of $\langle \tilde{\boldsymbol{L}}_i, \tilde{\boldsymbol{L}}_j \rangle_F$ have similar values (tend to be selected together).

Again, Fig. 2 shows the grouping effect of our regularizer. For example, we can see that 3-NN, 4-NN, and 5-NN have similar values. This means that these three graphs are similar in terms of the above measure. This result shows that our approach provides grouping information for a number of given different graphs.

3) Computational Efficiency: We can prove that our alternating minimization algorithm (Algorithm 1), is equivalent to the concave–convex procedure (CCCP) [18], which is widely used to solve nonconvex problems in many machine learning tasks. Substituting (2) into (1), we can obtain

$$\frac{\lambda_2}{2}\|\boldsymbol{u}\|_2^2 + \lambda_1\|\boldsymbol{y}\|_2^2 - \lambda_1^2 \boldsymbol{y}^\top \left(\lambda_1 \boldsymbol{I} + \sum_{k=1}^K \frac{\mu_k}{Z_k}\boldsymbol{L}_k\right)^{-1} \boldsymbol{y}. \tag{9}$$

In (9), the first two terms are obviously convex and the last term is concave with respect to $\boldsymbol{\mu}$. Solving QP with respect to $\boldsymbol{\mu}$ for (3) can be interpreted as solving the problem that linearizes the concave part of (9) (The details are in Appendix E). Rapid convergence of CCCP is empirically demonstrated in various applications. Thus we can expect that our approach inherits the nice convergence properties of CCCP and its theoretical analyses are also applicable to out approach (e.g., [19]).

The computational complexity of each iteration in our algorithm (line 4 and line 5 in Algorithm 1) is nearly $O(E + K)$, where E is the number of nonzero entries in $\sum_{k=1}^K \mu_k \tilde{\boldsymbol{L}}_k$: First, for line 4, the graph Laplacian matrix is usually sparse, and so the computational complexity of the linear system given by (2) is nearly linear in the number of nonzero entries of the graph Laplacian [20]. Second, line 5 is calculated by $O(K)$ computations, as shown in Section III-B.

TABLE I
SUMMARY OF FORMULATIONS AND OPTIMIZERS OF FIVE EXISTING METHODS AND OUR PROPOSED METHOD [SPARSE MULTIPLE GRAPH INTEGRATION (SMGI)]

Methods	Objective	Constraints	Optimization
TSS	$\min_{\boldsymbol{\alpha}} \boldsymbol{y}^\top \left(\boldsymbol{I} + \sum_{k=1}^K \alpha_k \boldsymbol{L}_k\right)^{-1} \boldsymbol{y}$	s.t. $0 \le \alpha_k \le c_0,\ \sum_{k=1}^K \alpha_k \le c.$	Matlab fmincon
RLPMN	$\max_{\boldsymbol{f}} -\frac{\beta_{\mathrm{y}}}{2}(\boldsymbol{f}-\boldsymbol{y})\boldsymbol{G}(\boldsymbol{f}-\boldsymbol{y}) - \frac{\beta_{\mathrm{bias}}}{2}\|\boldsymbol{f}\|_2^2 - \frac{\beta_{\mathrm{net}}}{2}\sum_{k=1}^K \bar{\mu}_k \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f}$	s.t. $\bar{\mu}_k = \int_0^\infty du_k \hat{q}(u_k) u_k.$	EM algorithm
OMGSSL	$\min_{\boldsymbol{f},\boldsymbol{\alpha}} \sum_{k=1}^K \alpha_k^r \left(\boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} + \lambda\|\boldsymbol{f}-\boldsymbol{y}\|_2^2\right)$	s.t. $\sum_{k=1}^K \alpha_k = 1.$	EM-like iterative optimization
GeneMANIA	$\min_{\boldsymbol{\mu}} \|\boldsymbol{T} - \boldsymbol{W}_\ell\|_F^2$	s.t. $\boldsymbol{W}_\ell = \sum_{k=0}^K \mu_k \boldsymbol{W}_\ell^{(k)},\ \mu_k \ge 0.$	Iterative ridge regression
GMKL	$\min_{\boldsymbol{f},\boldsymbol{\mu}} \sum_{i=1}^\ell V(y_i, f_i) + \gamma\|\boldsymbol{f}\|_{\boldsymbol{K}(\boldsymbol{\mu})}$	s.t. $\sum_{k=1}^K \mu_k = 1,\ \mu_k \ge 0,\ \boldsymbol{f} \in \mathcal{H}_{\boldsymbol{K}(\boldsymbol{\mu})}.$	Saddle-point based approach [21]
SMGI	$\min_{\boldsymbol{f},\boldsymbol{\mu}} \sum_{k=1}^K \frac{\mu_k}{Z_k} \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} + \lambda_1\|\boldsymbol{y}-\boldsymbol{f}\|_2^2 + \frac{\lambda_2}{2}\|\boldsymbol{\mu}\|_2^2$	s.t. $\sum_{k=1}^K \mu_k = 1,\ \mu_k \ge 0.$	CCCP

IV. RELATED WORK

We here raise five methods which address the same issue as that of our approach. They are all methods we found in the literature for the same problem setting as that of our approach. We note that we focus on only approaches that combine a number of different graphs in the context of graph-based semisupervised learning. In fact, combining multiple graphs can be interpreted as a special case of graph construction problem (see [22] and [23]), for which there are a lot of work, and several clustering papers also proposed methods combining different graphs [24], [25]. We however do not consider the methods in these directions in this paper, and we also do not try to compare with other paradigms in semisupervised learning (such as large margin methods) in which a variety of approaches have been studied so far (e.g., [26]–[31]). Multiview learning (e.g., [32]–[34]) is another line of approach to combining multiple different data sources (views of features). Several papers have also proposed methods incorporating multiple graph views of input data [35]–[38]. However, unlike our case, most of these algorithms do not intend to learn weights for combination of multiple views. They often deal with weights as hyper-parameters of learning algorithms and optimize them by some model selection procedure such as cross-validation. In our case, selecting optimal weights by model selection is computationally expensive (e.g., if we use grid search with 10 grid points in each dimension of weights, we have to run each algorithm at least 10^K times).

Table I summarizes the formulations of the five methods and their optimizers, which are employed by their original papers or implementations. See Section IV for details of each method and qualitative advantages of our approach over these existing methods.

A. TSS [10]

The formulation is given by the following optimization problem:

$$\begin{aligned}\min_{\boldsymbol{f},\{\xi_k\}_{k=1}^K,\gamma} \quad & (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + c\gamma + c_0 \textstyle\sum_{i=1}^K \xi_k \\ \text{s.t.} \quad & \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} \le \gamma + \xi_k,\ \xi_k \ge 0,\ \gamma \ge 0\end{aligned}$$

where c and c_0 are positive constants. The last term of the objective function penalizes the smoothness of the score $\boldsymbol{f}$ on all K graphs using slack variables $\{\xi_k\}_{k=1}^K$. The dual problem is

$$\begin{aligned}\min_{\boldsymbol{\alpha}} \quad & \boldsymbol{y}^\top \left(\boldsymbol{I} + \textstyle\sum_{k=1}^K \alpha_k \boldsymbol{L}_k\right)^{-1} \boldsymbol{y} \\ \text{s.t.} \quad & 0 \le \alpha_k \le c_0,\ \textstyle\sum_{k=1}^K \alpha_k \le c.\end{aligned}$$

Here, dual variable α_k can be interpreted as weights for given graphs. We can see that α_k has nonzero values for graphs with large $\boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f}$, due to the complimentary conditions of the KKT [11]. This means that TSS tends to impose large weights to relatively nonsmooth graphs. Thus we can say that TSS is not robust against graphs irrelevant to classification.

B. RLPMN [11]

This method combines multiple graphs by a maximum a posterior (MAP) estimation approach using the Gamma distribution as the prior for weighting coefficients μ_k

$$p(\mu_k) = \mathrm{Gamma}\left(\mu_k; \frac{1}{2}\nu, \frac{1}{2}\nu\right)$$

where ν is a positive constant and

$$\mathrm{Gamma}(\mu; a, b) = \Gamma(a)^{-1} b^a \mu^{a-1} \exp(-b\mu)$$

with Gamma function $\Gamma(a)$. The objective function can be written as the following form:

$$\begin{aligned}\max_{\boldsymbol{f}} \quad & -\tfrac{\beta_{\mathrm{y}}}{2}(\boldsymbol{f}-\boldsymbol{y})\boldsymbol{G}(\boldsymbol{f}-\boldsymbol{y}) - \tfrac{\beta_{\mathrm{bias}}}{2}\|\boldsymbol{f}\|_2^2 - \tfrac{\beta_{\mathrm{net}}}{2}\textstyle\sum_{k=1}^K \bar{\mu}_k \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} \\ \text{s.t.} \quad & \bar{\mu}_k = \textstyle\int_0^\infty du_k \hat{q}(u_k) u_k\end{aligned}$$

where $\boldsymbol{G}$ is a diagonal matrix with $G_{ii} = 1$ for $i = 1, \ldots, \ell$, and $G_{ii} = 0$ for $i = \ell + 1, \ldots, n$, $\{\beta_{\mathrm{y}}, \beta_{\mathrm{bias}}, \beta_{\mathrm{net}}\}$ is a set of positive constant parameters and

$$\hat{q}(\mu_k) = \mathrm{Gamma}\left(u_k; \frac{\nu+n}{2}, \frac{\nu}{2} + \frac{\beta_{\mathrm{net}}}{2}\boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f}\right).$$

To obtain optimal $\boldsymbol{f}$ and weighting coefficients, they derived the EM algorithm that maximizes the objective function under fixed $\bar{\mu}_k$ and calculates the expectation $\bar{\mu}_k$ under fixed $\boldsymbol{f}$, alternately. The expectation is efficiently computed by the following updating rule:

$$\bar{\mu}_k = \frac{\nu + n}{\nu + \beta_{\text{net}} \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f}}.$$

This equation means that, in Robust Label Propagation on Multiple Networks (RLPMN), $\bar{\mu}_k$ cannot be 0 even for graphs irrelevant to classification. RLPMN outperformed TSS, but still assigns relatively large weight values to graphs irrelevant to classification, by which sparse weights are not obtained.

C. OMGSSL [12]

The formulation is given as follows:

$$\begin{aligned} \min_{\boldsymbol{f},\boldsymbol{\alpha}} \quad & \textstyle\sum_{k=1}^K \alpha_k^r \left(\boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} + \lambda \|\boldsymbol{f} - \boldsymbol{y}\|_2^2 \right) \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^K \alpha_k = 1 \end{aligned}$$

where $r \geq 1$ and the weights can be represented as $\beta = \alpha_i^r / (\sum_k \alpha_k^r)$. The optimization is done by an EM-style iterative algorithm, which updates $\boldsymbol{f}$ and $\boldsymbol{\alpha}$, alternately. Because the minimum of $\sum_{k=1}^K \alpha_k^r$ is achieved at $\alpha_k = 1/K$, the optimal α_k is $1/K$ if $r = \infty$. On the other hand, if we decrease r to 1, the weights converge to $\boldsymbol{e}_{k_{\min}}$. However, the weights cannot be exactly 0 except for the case $r = 1$, as shown by the updating rule of α_k

$$\alpha_k = \frac{\left(\frac{1}{\boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} + \lambda \|\boldsymbol{f} - \boldsymbol{y}\|_2^2} \right)^{\frac{1}{r-1}}}{\sum_{k=1}^K \left(\frac{1}{\boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f} + \lambda \|\boldsymbol{f} - \boldsymbol{y}\|_2^2} \right)^{\frac{1}{r-1}}}. \tag{10}$$

This shows that Optimal Multi-Graph Semi-Supervised Learning (OMGSSL) always gives nonzero weights to every graph (except for the case with $r = 1$) including graphs irrelevant to classification. This point will cause performance deterioration. In addition, α_k is very sensitive to the value of exponential coefficient r, which is easily affected by noise and is hard to choose the optimal value by cross-validation.

D. GeneMANIA [39]–[41]

This method linearly combines graphs (affinity matrices) as follows:

$$\boldsymbol{W} = \sum_{k=1}^K \mu_k \boldsymbol{W}^{(k)}$$

where $\{\mu_k\}_{k=1}^K$ are a set of graph weights. Optimal weights are obtained using ℓ labeled nodes only and optimizing one type of kernel-target alignment (KTA) [42] between an integrated graph and target graph $\boldsymbol{T} \in \mathbb{R}^{\ell \times \ell}$.

$$\boldsymbol{T} = \boldsymbol{t}\boldsymbol{t}^\top$$

where $\boldsymbol{t} = (t_1, \ldots, t_\ell)^\top$ is defined by

$$t_i = \begin{cases} \ell^+/\ell & \text{for } y_i = -1, \\ -\ell^-/\ell & \text{for } y_i = 1 \end{cases}$$

with ℓ^+ and ℓ^- are the number of positive and negative instances, respectively, in labeled ℓ nodes. Using this target graph $\boldsymbol{T}$, the following squared error is minimized:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \|\boldsymbol{T} - \boldsymbol{W}_\ell\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{W}_\ell = \textstyle\sum_{k=0}^K \mu_k \boldsymbol{W}_\ell^{(k)}, \ \mu_k \geq 0 \end{aligned}$$

where $\boldsymbol{W}_\ell^{(k)}$ is a subgraph of $\boldsymbol{W}^{(k)}$ that has only labeled nodes of the original graph $\boldsymbol{W}^{(k)}$ and $\boldsymbol{W}_\ell^{(0)}$ is a constant graph defined by $\boldsymbol{W}_\ell^{(0)} = \boldsymbol{1}\boldsymbol{1}^\top$, and coefficient μ_0 is a bias term (meaning that $\boldsymbol{W}_\ell^{(0)}$ is discarded when we construct the final graph). The weights are estimated by one type of ridge regression, where weights are kept as nonnegatives. That is, if a weight becomes a negative value: $\mu_k < 0$, this weight is set to be $\mu_k = 0$. Thus in this method, weights μ_k can be exactly zero. However, this approach creates the graph using only labeled nodes, by which the score smoothness on unlabeled nodes is not considered. Note that this point is in general the most essential part of graph-based semisupervised learning, by which the resultant integrated graph cannot be optimal in terms of the score smoothness over the entire graph.

E. GMKL [15]

This is a MKL-based approach that combines multiple graphs based on the graph regularization framework. The graph kernel for the kth graph is defined as the pseudoinverse of the corresponding graph Laplacian: $\boldsymbol{K}^{(k)} = \boldsymbol{L}_k^+$, where $^+$ indicates the matrix pseudoinverse. This approach estimates the optimal convex combination of these kernels using a formulation similar to kernel learning [14]

$$\begin{aligned} \min_{\boldsymbol{f},\boldsymbol{\mu}} \quad & \textstyle\sum_{i=1}^\ell V(y_i, f_i) + \gamma \|\boldsymbol{f}\|_{\boldsymbol{K}(\boldsymbol{\mu})}^2 \\ \text{s.t.} \quad & \textstyle\sum_{k=1}^K \mu_k = 1, \ \mu_k \geq 0, \ \boldsymbol{f} \in \mathcal{H}_{\boldsymbol{K}(\boldsymbol{\mu})} \end{aligned} \tag{11}$$

where V is a loss function, γ is a positive constant, $\mathcal{H}_{\boldsymbol{K}(\boldsymbol{\mu})}$ is a reproducing kernel Hilbert space (RKHS) defined on the combined kernel $\boldsymbol{K}(\boldsymbol{\mu}) = \sum_{k=1}^K \mu_k \boldsymbol{K}^{(k)}$ and $\| \cdot \|_{\boldsymbol{K}(\boldsymbol{\mu})}$ is an induced norm (see [6] and [15] for detail). This problem formulation was solved by transforming into an equivalent saddle-point problem, which was presented by [21]. Graph-based Multiple Kernel Learning (GMKL) does not explicitly enforce sparsity, but in our experiments, we observe that sparse weights are obtained occasionally. However, we note that because of the unclearness of the mechanism, it is hard to directly control the sparsity of weights in GMKL. Furthermore, unlike our approach, GMKL cannot provide any grouping effect, by which for similar graphs in $\{\boldsymbol{W}^{(k)}\}_{k=1}^K$, weights cannot be always similar.

Several papers have developed elastic net-based regularization for MKL [43]–[46]. To compare this approach in our experiments, we consider the following formulation that introduces elastic net regularization to (11):

$$\min_{\boldsymbol{f}, \{\boldsymbol{f}_k\}_{k=1}^K} \sum_{i=1}^{\ell} V(y_i, f_i) + C \sum_{k=1}^{K} \left((1-\lambda)\|\boldsymbol{f}_k\|_{\boldsymbol{K}^{(k)}} + \lambda \|\boldsymbol{f}_k\|_{\boldsymbol{K}^{(k)}}^2\right)$$
$$\text{s.t. } \boldsymbol{f} = \sum_{k=1}^{K} \boldsymbol{f}_k \tag{12}$$

where $C \geq 0$ and $\lambda \in [0, 1]$ are regularization parameters (See Appendix F for detail). We derived this formulation based on [44], and we refer to this formulation as elastic-net GMKL (EGMKL) in this paper. In both of GMKL and EGMKL, combined graphs are represented through kernels: $\boldsymbol{K}(\boldsymbol{\mu}) = \sum_{k=1}^{K} \mu_k \boldsymbol{K}^{(k)}$. Because the kernel $\boldsymbol{K}^{(k)}$ is set as the pseudoinverse of graph Laplacian, the interpretation of that combination in terms of the original graphs is not clear.

V. Experiment

We empirically evaluated our approach using six synthetic and two real-world data sets, comparing with six existing methods: TSS, RLPMN, OMGSSL, GeneMultiple Association Network Integration Algorithm (GeneMANIA), GMKL and EGMKL, which are all introduced in the Section IV . For brevity, we hereafter refer to our proposed approach as Sparse Multiple Graph Integration (SMGI) and GeneMANIA as GMANIA. We use the original implementations of TSS and GMKL that are available from the authors' web sites.[2] Hyper-parameters in each method (λ_1 and λ_2 in SMGI) are optimized by 10-fold cross-validation.[3] Note that cross-validation can be unreliable when we only have few labeled instances. Although stable model selection is still one of the challenging problems in semisupervised learning [47], we do not tackle this problem in this paper. We first generated graphs according to given node information $\mathcal{X}$, and we call these graphs relevant graphs. For relevant graphs, in most cases, we use the following nearest-neighbor graph (NN graph):

$$W_{ij} = \begin{cases} \exp\left(-\frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{2\sigma^2}\right), & i \in \mathcal{N}_j \text{ or } j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases}$$

where $\mathcal{N}_i$ is an index set of k nearest-neighbors of point $\boldsymbol{x}_i$, $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a distance metric function, and σ is a positive constant parameter. We selected σ from $\{10^{-1}\sigma_0, 10^{1}\sigma_0, 10^{1}\sigma_0\}$ by cross-validation, where σ_0 is heuristically defined in each experiment (See later section for detail). On the other hand, the other graphs are those irrelevant to classification which we call irrelevant graphs.

A. Synthetic Data Sets

We first used six synthetic data sets in Fig. 3. The size of each data set is $n = 300$. We investigated the performance of each method in the following two scenarios.

[2]TSS is available at http://www.cbrc.jp/ tsuda/code/eccb05.html and GMKL is at http://ttic.uchicago.edu/ argyriou/code/index.html

[3]For SMGI (or OMGSSL with $r = 1$), if $f_i = 0$ is obtained for unlabeled nodes, we increased λ_2 (or r in OMGSSL) to have prediction for such nodes from already eliminated graphs.

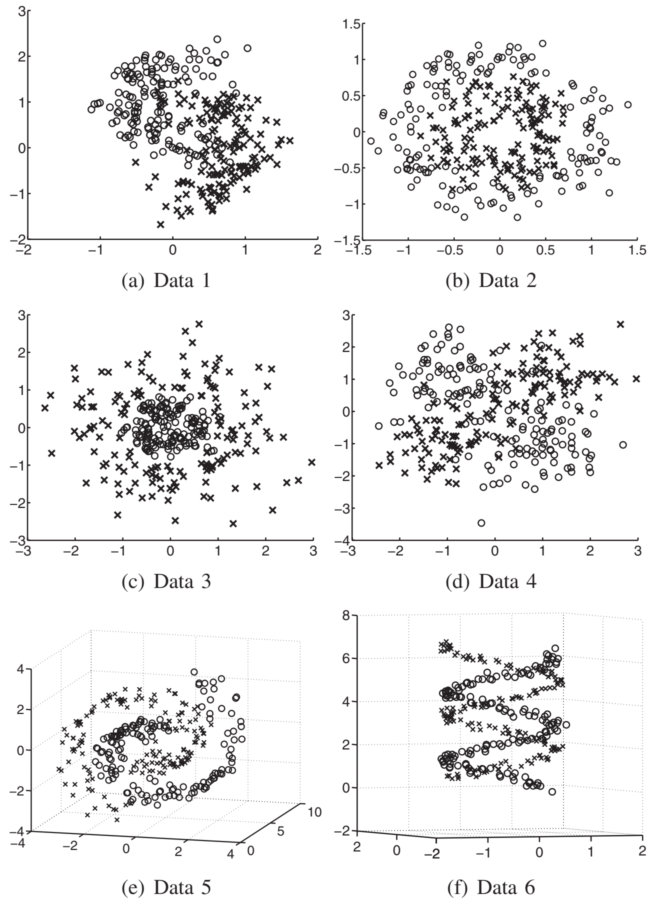


Fig. 3. Synthetic data sets. (a) Data 1. (b) Data 2. (c) Data 3. (d) Data 4. (e) Data 5. (f) Data 6.

1) For relevant graphs, we constructed the k-NN graphs $\boldsymbol{W}^{(k)}$ for $k = 1, \ldots, 5$, where σ_0 is computed by $\sigma_0 = \sum_{i=1}^{n} d(\boldsymbol{x}_i, \boldsymbol{x}_{i_k})/n$ and $\boldsymbol{x}_{i_k}$ is the k-th nearest neighbor of $\boldsymbol{x}_i$ [48]. We also created two types of irrelevant graphs to evaluate graph selection ability of each method. The first type is generated from 5-NN graphs by randomly shuffling the index order of $\{\boldsymbol{x}_i\}_{i=1}^{n}$. The second type is constructed by randomly generating the element W_{ij} from the uniform distribution in [0, 1]. For each of the two types, graphs are generated ten times with different randomizations. Thus the total number of graphs is $K = 25 = 5 + 20$.
2) We used graphs in which only a small part of nodes have edges to their neighbors, two example graphs being shown in Fig. 4. In this case, one single graph cannot provide complete information of graph structures, and integrating multiple graphs properly is the key to achieve good performance. First, we randomly selected ten center nodes from the entire data set, and picked up k' nearest-neighbor data points for each of those center nodes as small subsets of data points. After that, 5-NN graphs are created using each small subset of data points. Here, k' is randomly chosen from $\{10, 11, \ldots, 40\}$. The intuition behind this setting is that each of these ten graphs corresponds to a differently extracted community or a different subnetwork of the entire network, and these ten different networks should be integrated well

to estimate the original, gold-standard graph. Here σ_0 was set as the mean of all data point pair distances. We then generated 20 irrelevant graphs by using the same procedure in scenario 1. Thus, we have $K = 30 = 10 + 20$ graphs.

We employ the Euclidean distance for d. The ratio of the number of labeled data points ℓ to the number of the entire data points n, i.e., ℓ/n, is set at 0.1. Another item of note is that the original implementation of GMKL normalizes kernel matrices so that they have the same Frobenius norm, but we do not employ it because we found that this normalization gives dominant weights to irrelevant graphs in our experiments.

Table II shows the average Area Under the Curve (AUC) values of seven competing methods for the two scenarios, each value being averaged over 30 runs. For scenario 1, SMGI, GMKL, and EGMKL achieve the highest performances while for scenario 2, SMGI and OMGSSL achieve the highest performances, being compared to other methods. This means that the performance of SMGI is the highest among all seven competing methods. Another note is that SMGI, GMKL, EGMKL, and OMGSSL are comparable with each other in performance, whereas the performances of TSS, RLPMN, and GMANIA are much worse than the top four methods, i.e., SMGI, OMGSSL, GMKL, and EGMKL. Additionally, the result of scenario 2 shows that GMKL and EGMKL could not combine small graphs well. OMGSSL could combine small graphs in scenario 2, but in scenario 1, in which larger proportion of input graphs is irrelevant, AUC values of OMGSSL lower than SMGI in 5 out of 6 data sets. These results suggest that SMGI is more robust to the existence of irrelevant graphs.

To evaluate graph selection abilities, we present the slightly modified AUC in Table II that is defined by the following manner: Suppose graphs have their labels $y_i^g \in \{\pm 1\}$ $(i = 1, \ldots, K)$ that indicate whether each graph is a relevant graph or an irrelevant graph. For example, in scenario 1, $y_i^g = 1$ is for $i = 1, \ldots, 5$ and $y_i^g = -1$ for $i = 6, \ldots, 25$. Here, the graph weights obtained by each method is considered as scores for classifying y_i^g, and we can compute AUC over the weights and the corresponding labels. The standard AUC penalizes the case of $u_i = 0$ and $u_j = 0$ for $y_i^g = 1$ and $y_j^g = -1$, respectively. This case is however not a problem for us, and so we slightly modified the formulation of the standard AUC as follows:

$$\frac{1}{K^P K^N} \sum_{(i,j)\in\{(i,j)|y_i=1,y_j=-1\}} I((u_i > u_j) \text{ or } (u_i = 0, u_j = 0))$$

where K^P is the number of relevant graphs, K^N is the number of irrelevant graphs, and I is the indicator function. We hereafter refer to this as the modified AUC. Table II shows the modified AUC values for SMGI, OMGSSL, GMANIA, GMKL, and EGMKL. SMGI achieves the highest performances among the five competing methods. We emphasize that GMKL and EGMKL does not perform well in graph selection despite of the high performance of scenario 1 in Table II. OMGSSL achieves good performance, but we note that OMGSSL usually does not have sparse weights, as mentioned in Section IV.

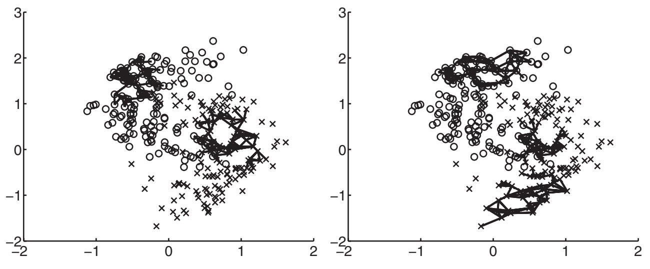


Fig. 4. Graph examples in scenario 2. (a) Example 1. (b) Example 2.

Table II shows the average false positive rates obtained by SMGI, OMGSSL, GMANIA, GMKL, and EGMKL. The average false positive rate means the average ratio of the number of nonzero weights on irrelevant graphs to the total number of irrelevant graphs. We can see that the false positive rates of SMGI are clearly lowest among those of the five competing methods. OMGSSL always gave nonzero weights to all graphs, resulting in very large false positive rates, which makes hard to select relevant graphs using OMGSSL, even if the classification performance of OMGSSL is high. The false positive rates of GMKL and EGMKL are also large, comparing to those of SMGI, implying the hardness of selecting relevant graphs.

Data sets 5 and 6 have 2-D or 1-D manifold structure embedded in 3-D space, respectively. For these data sets, each graph should represent underlying manifold structure accurately. Fig. 5 shows the 3-NN graph (which approximately represents manifold structure) and the random graph (where we show only a part of edges for graphical simplicity). In Table II, false positive rates of our algorithm for data sets 5 and 6 are zero. This result means that SMGI can clearly find graphs representing manifold because those graphs lead smoother score function than the random (noise) graphs.

Finally, as the regularization path for λ_2 ($\boldsymbol{\mu}$ as a function of λ_2) is shown for SMGI in Fig. 2, Fig. 6 shows the weights as a function of r in OMGSSL and γ in GMKL for Data 1 in Scenario 1. It is easy to see from this figure that in OMGSSL, all graphs, including irrelevant graphs, have nonzero weights for almost all parameter values, while weights change very rapidly for small r, implying that the weights are highly sensitive to r. On the other hand, GMKL does not give large weights to irrelevant graphs, meaning that the weight assignment by GMKL is more stable. However, in GMKL, we can see that all the relevant graphs does not necessarily have high weight values. For example, 2-NN and 3-NN graphs have relatively high values, while 1-NN, 4-NN, and 5-NN have very small values. This means that it is hard for GMKL to find some clear rule on larger weights that the relevant graphs can have, by which it will be difficult to retrieve all relevant graphs. We emphasize that this is a clear and important difference between GMKL and SMGI, which allows to retrieve all relevant graphs, eliminating irrelevant graphs, as shown in Fig. 2.

Overall these results indicate that SMGI could eliminate irrelevant graphs efficiently, by which SMGI achieves the highest predictive performance among all seven competing methods under the setting of synthetic data.

TABLE II

PERFORMANCE COMPARISON USING SIX SYNTHETIC DATA SETS. THE RESULTS ARE THE AVERAGE OF 30 RUNS. THE BEST METHOD AND THOSE COMPARABLE WITH THE BEST ACCORDING TO THE t TEST WITH 5% SIGNIFICANCE LEVEL ARE SHOWN BY BOLDFACE

(a) AUC and standard deviation.

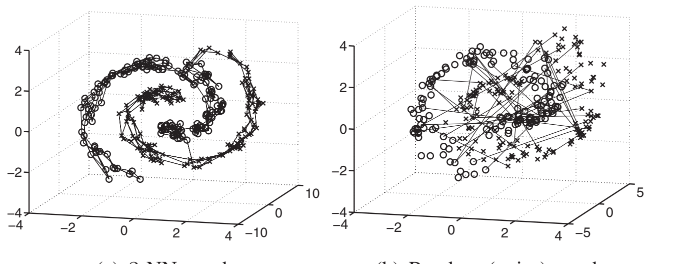
Scenario	Data	SMGI	TSS	RLPMN	OMGSSL	GMANIA	GMKL	EGMKL
1	1	0.86 (0.09)	0.62 (0.04)	0.64 (0.05)	0.83 (0.11)	0.74 (0.13)	**0.93 (0.04)**	**0.92 (0.05)**
	2	**0.84 (0.06)**	0.61 (0.05)	0.63 (0.04)	0.79 (0.12)	0.76 (0.10)	**0.84 (0.09)**	**0.81 (0.10)**
	3	**0.95 (0.03)**	0.64 (0.04)	0.67 (0.04)	0.91 (0.08)	0.80 (0.12)	**0.95 (0.06)**	**0.95 (0.06)**
	4	**0.75 (0.08)**	0.58 (0.05)	0.59 (0.04)	0.72 (0.10)	0.62 (0.11)	**0.78 (0.11)**	0.73 (0.14)
	5	**0.99 (0.05)**	0.65 (0.04)	0.68 (0.04)	**0.99 (0.02)**	0.86 (0.10)	**1.00 (0.00)**	**1.00 (0.00)**
	6	**1.00 (0.00)**	0.64 (0.04)	0.68 (0.05)	0.99 (0.01)	0.84 (0.09)	0.56 (0.13)	0.64 (0.19)
2	1	**0.87 (0.05)**	0.59 (0.05)	0.61 (0.05)	**0.86 (0.08)**	0.70 (0.11)	0.62 (0.10)	0.62 (0.09)
	2	**0.84 (0.05)**	0.58 (0.03)	0.60 (0.04)	0.82 (0.06)	0.66 (0.10)	0.62 (0.09)	0.65 (0.07)
	3	**0.92 (0.03)**	0.59 (0.03)	0.62 (0.04)	**0.90 (0.06)**	0.78 (0.12)	0.72 (0.12)	0.72 (0.10)
	4	**0.76 (0.08)**	0.56 (0.03)	0.58 (0.04)	**0.76 (0.10)**	0.63 (0.12)	0.64 (0.10)	0.64 (0.08)
	5	**0.98 (0.02)**	0.61 (0.04)	0.63 (0.04)	**0.97 (0.03)**	0.77 (0.11)	0.92 (0.07)	0.94 (0.06)
	6	0.97 (0.03)	0.62 (0.03)	0.65 (0.04)	**0.99 (0.02)**	0.78 (0.12)	0.51 (0.05)	0.57 (0.13)

(b) Modified AUC and standard deviation.

Sc.	Data	SMGI	OMGSSL	GMANIA	GMKL	EGMKL
1	1	**1.00 (0.00)**	**0.92 (0.23)**	0.73 (0.34)	0.79 (0.40)	0.40 (0.33)
	2	**0.98 (0.09)**	**0.92 (0.23)**	0.85 (0.24)	0.59 (0.49)	0.60 (0.31)
	3	**1.00 (0.00)**	**0.97 (0.13)**	0.71 (0.34)	0.86 (0.34)	0.57 (0.29)
	4	**1.00 (0.00)**	**0.92 (0.27)**	0.80 (0.28)	0.80 (0.41)	0.45 (0.33)
	5	**1.00 (0.00)**	**1.00 (0.00)**	0.86 (0.28)	0.56 (0.50)	0.50 (0.13)
	6	**1.00 (0.00)**	**1.00 (0.00)**	0.72 (0.33)	0.50 (0.48)	0.40 (0.35)
2	1	**1.00 (0.00)**	**1.00 (0.00)**	0.75 (0.32)	0.78 (0.34)	0.28 (0.34)
	2	**1.00 (0.00)**	**1.00 (0.00)**	0.65 (0.31)	0.86 (0.30)	0.23 (0.34)
	3	**1.00 (0.00)**	**1.00 (0.00)**	0.76 (0.30)	0.83 (0.35)	0.43 (0.41)
	4	**1.00 (0.00)**	**1.00 (0.00)**	0.67 (0.32)	0.75 (0.39)	0.33 (0.38)
	5	**1.00 (0.00)**	**1.00 (0.00)**	0.86 (0.24)	0.55 (0.50)	0.58 (0.49)
	6	**1.00 (0.00)**	**1.00 (0.00)**	0.83 (0.29)	0.90 (0.18)	0.41 (0.35)

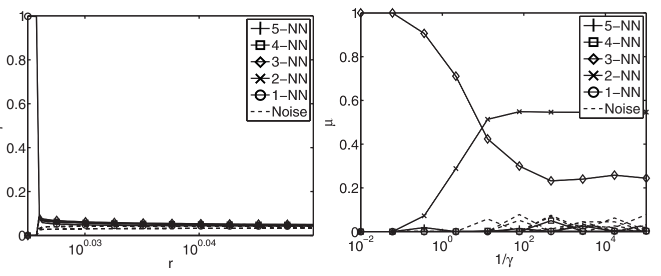
(c) False positive rates and standard deviation.

Sc.	Data	SMGI	OMGSSL	GMANIA	GMKL	EGMKL
1	1	**0.08 (0.25)**	0.57 (0.50)	0.58 (0.25)	**0.21 (0.40)**	0.60 (0.33)
	2	**0.03 (0.18)**	0.90 (0.31)	0.40 (0.24)	0.42 (0.49)	0.43 (0.34)
	3	**0.00 (0.00)**	0.80 (0.41)	0.54 (0.27)	0.14 (0.34)	0.43 (0.29)
	4	**0.10 (0.28)**	0.80 (0.41)	0.55 (0.27)	**0.20 (0.41)**	0.55 (0.33)
	5	**0.00 (0.00)**	0.63 (0.49)	0.51 (0.27)	0.44 (0.50)	0.50 (0.13)
	6	**0.00 (0.00)**	0.60 (0.50)	0.53 (0.24)	0.50 (0.48)	0.65 (0.32)
2	1	**0.02 (0.09)**	1.00 (0.00)	0.52 (0.28)	0.31 (0.41)	0.78 (0.28)
	2	**0.03 (0.18)**	1.00 (0.00)	0.55 (0.27)	**0.15 (0.31)**	0.80 (0.31)
	3	**0.00 (0.00)**	1.00 (0.00)	0.59 (0.25)	0.30 (0.44)	0.80 (0.28)
	4	**0.07 (0.25)**	1.00 (0.00)	0.50 (0.27)	0.32 (0.42)	0.83 (0.30)
	5	**0.00 (0.00)**	1.00 (0.00)	0.55 (0.26)	0.84 (0.38)	0.65 (0.40)
	6	**0.00 (0.00)**	1.00 (0.00)	0.54 (0.27)	0.13 (0.24)	0.78 (0.28)



(a) 3-NN graph (b) Random (noise) graph

Fig. 5. Graph examples for Data 5. (a) 3-NN graph. (b) Random (noise) graph



(a) OMGSSL (b) GMKL

Fig. 6. Weights for 25 different graphs of Data 1 as a function of r in OMGSSL and γ in GMKL. The solid marked lines are corresponding to k-NN graphs ($k = 1, \ldots, 5$) and the dashed lines are 20 random graphs. (a) OMGSSL. (b) GMKL.

B. Real-World Data Sets

We used the following two real-world data sets:

1) Protein function prediction data set (Protein).
This is available at http://noble.gs.washington.edu/proj/sdp-svm/, containing three types of input graphs: protein interaction network and two sequence-based networks, one being by BLAST and the other by the Smith–Waterman algorithm (see [49] for detail). For

TABLE III

PERFORMANCE COMPARISON USING TWO REAL-WORLD DATA SETS. THE RESULTS ARE THE AVERAGE OVER 30 RUNS. THE BEST METHOD AND THOSE COMPARABLE WITH THE BEST ACCORDING TO THE *t* TEST WITH 5% SIGNIFICANCE LEVEL ARE SHOWN BY BOLDFACE

(a) AUC and standard deviation.

Labels		SMGI	OMGSSL	GMKL	EGMKL
Protein	5%	**0.87 (0.05)**	**0.86 (0.05)**	0.74 (0.12)	0.77 (0.08)
	10%	**0.91 (0.04)**	**0.91 (0.03)**	0.84 (0.06)	0.85 (0.05)
	15%	0.93 (0.03)	**0.95 (0.03)**	0.87 (0.06)	0.90 (0.02)
WebKB	5%	0.91 (0.05)	0.88 (0.03)	0.93 (0.02)	**0.94 (0.02)**
	10%	**0.96 (0.03)**	0.91 (0.02)	0.93 (0.02)	**0.96 (0.02)**
	15%	0.95 (0.03)	0.93 (0.02)	0.94 (0.02)	**0.97 (0.01)**

(b) Modified AUC and standard deviation.

Labels		SMGI	OMGSSL	GMKL	EGMKL
Protein	5%	**1.00 (0.00)**	**1.00 (0.00)**	0.39 (0.46)	0.23 (0.31)
	10%	**1.00 (0.00)**	**1.00 (0.00)**	0.16 (0.36)	0.12 (0.22)
	15%	**1.00 (0.00)**	**1.00 (0.00)**	0.00 (0.00)	0.12 (0.22)
WebKB	5%	**1.00 (0.00)**	**1.00 (0.00)**	**0.93 (0.25)**	0.30 (0.43)
	10%	**1.00 (0.00)**	**1.00 (0.00)**	0.63 (0.49)	0.35 (0.48)
	15%	**1.00 (0.00)**	**1.00 (0.00)**	0.67 (0.48)	0.18 (0.38)

(c) False positive rates and standard deviation.

Labels		SMGI	OMGSSL	GMKL	EGMKL
Protein	5%	**0.07 (0.25)**	1.00 (0.00)	0.61 (0.46)	0.85 (0.30)
	10%	**0.07 (0.25)**	1.00 (0.00)	0.84 (0.36)	0.88 (0.22)
	15%	**0.03 (0.18)**	1.00 (0.00)	1.00 (0.00)	0.90 (0.20)
WebKB	5%	**0.00 (0.00)**	0.97 (0.18)	**0.07 (0.25)**	0.70 (0.43)
	10%	**0.03 (0.18)**	0.93 (0.25)	0.37 (0.49)	0.65 (0.48)
	15%	**0.00 (0.00)**	0.97 (0.18)	0.33 (0.48)	0.83 (0.38)

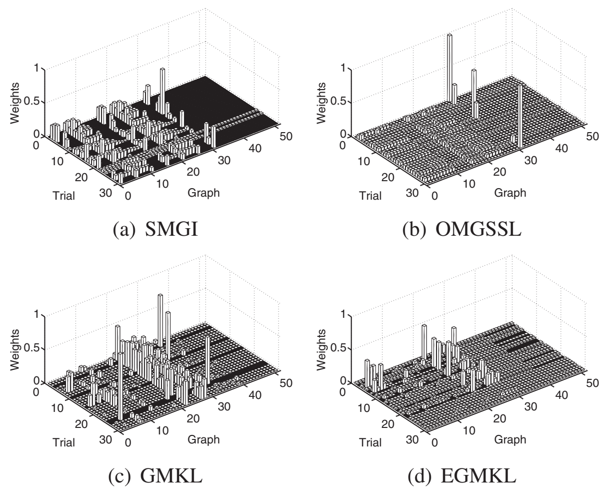
(d) AUC and standard deviation without irrelevant graphs.

Labels		SMGI	OMGSSL	GMKL	EGMKL
Protein	5%	**0.90 (0.05)**	**0.90 (0.05)**	0.76 (0.15)	0.81 (0.08)
	10%	**0.92 (0.05)**	**0.93 (0.05)**	0.85 (0.08)	0.90 (0.03)
	15%	**0.96 (0.02)**	**0.96 (0.02)**	0.90 (0.03)	0.92 (0.02)
WebKB	5%	0.91 (0.05)	0.92 (0.04)	0.93 (0.02)	**0.95 (0.02)**
	10%	**0.97 (0.02)**	0.95 (0.03)	0.94 (0.02)	0.96 (0.02)
	15%	0.96 (0.03)	**0.97 (0.03)**	0.94 (0.02)	**0.97 (0.01)**

the protein interaction network, we calculated distances between two nodes as the difference between the corresponding interaction patters. Using the interaction matrix (a binary matrix indicating interactions between protein pairs), we define the interaction pattern as a row of the interaction matrix. We also use binary interaction network directly as one of the input graphs $\boldsymbol{W}^{(k)}$. The task is to predict whether or not each protein is ribosomal, meaning binary class labels. The number of nodes is $n = 1040$.

2) Web page classification data set (WebKB).
   This is available at http://vikas.sindhwani.org/MR.zip, with two input graphs: the Web page text content called page and the anchor-text on links called link. WebKB has binary classes: course and noncourse. The entire size is $n = 1051$.

For relevant graphs, we have generated two sets of k-NN graphs with $k \in \{1, \ldots, 5\}$, one set being by the Euclidean distance and the other by the cosine distance, for both Protein and WebKB. We set σ_0 by the same way as the scenario 1 in the experiments of synthetic data sets. We added 20 irrelevant graphs, which are generated in the same manner as those for



(a) SMGI (b) OMGSSL (c) GMKL (d) EGMKL

Fig. 7. Comparison of resultant weights for Protein data. The 50 graphs of Protein data are generated as follows: Graph 1–10 are NN graphs from BLAST sequence data. Graph 11–20 are sequence data from Smith–Waterman. Graph 21–30 are NN graphs from protein–protein interaction data. For each of the three 10 graphs, the first 5 and the rest of the 5 graphs are generated by Euclidean distance and cosine similarity, respectively. Graph 31 is protein–protein interaction network itself. Graph 32–51 are just noise graphs. (a) SMGI. (b) OMGSSL. (c) GMKL. (c) GMKL.

the synthetic data sets. So totally we have 51 $(= 5 \times 2 \times 3 + 1 + 20)$ different graphs for Protein and 40 $(= 5 \times 2 \times 2 + 20)$ different graphs for WebKB. We randomly choose only 5%, 10%, and 15% of all nodes as labeled nodes, keeping the number of positives equal to that of negatives, and the rest are unlabeled.

In this experiment, as competing methods, we focused on OMGSSL, GMKL, and EGMKL only, both achieve relatively comparable performances against SMGI. Table III shows the average AUC of SMGI, taken over 30 runs, comparing with those of OMGSSL, GMKL and EGMKL. In terms of the AUC values for Protein data set in Table III, SMGI outperformed OMGSSL, GMKL and, EGMKL in all cases except only one. For WebDB data set, although EGMKL has the highest AUC values, SMGI also has comparable high prediction performance. Table III is the modified AUC of the three methods, showing that both SMGI and OMGSSL achieve the highest values (i.e., 1). Table III reports the false positive rates of all the three competing methods, showing that SMGI has significantly lower false positive rates than OMGSSL, GMKL, and EGMKL. Overall, the real-world data set results also show the highest performance of SMGI, keeping the nice property of graph selection. Table III shows the AUC values without irrelevant graphs. We can see that relative performance among all the methods is almost the same as Table III.

Figs. 7 and 8 are the plots of resultant weights for each method in all 30 trials. In both of Figs. 7(a) and 8(a), our approach clearly eliminates irrelevant graphs and gives nonzero weights for similar graphs simultaneously. OMGSSL [Figs. 7(b) and 8(b)] gives nonzero weights for all graphs. Although GMKL and EGMKL also obtain sparse weights in some trials, the results are difficult to interpret compared to SMGI. By selecting only important graphs using sparse constraint, resultant weights of SMGI are stable than the other methods. This would also make average performance

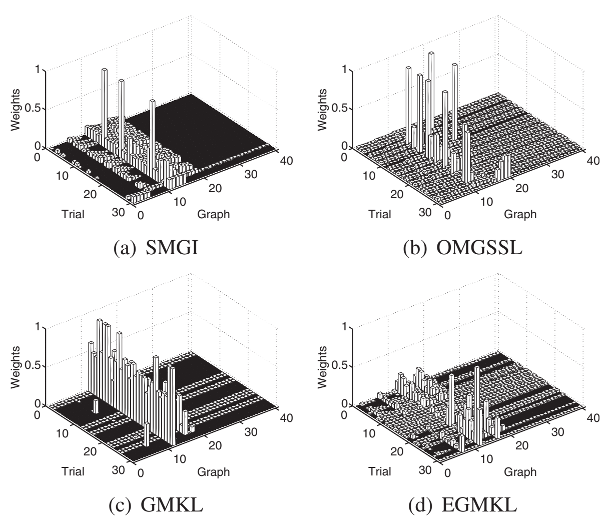


Fig. 8. Comparison of resultant weights for WebKB data. Graph 40 of WebKB data are generated as follows: Graph 1–10 are NN graphs from the page text data. Graph 11–20 are the link data. For each of the three 10 graphs, the first 5 and the rest of the 5 graphs are generated by Euclidean distance and cosine similarity, respectively. Graph 21–40 are just noise graphs. (a) SMGI. (b) OMGSSL. (c) GMKL. (d) EGMKL.

of SMGI better and enable SMGI to eliminate irrelevant graphs.

VI. Conclusion

We have proposed a new method, SMGI, of integrating multiple graphs for label propagation. SMGI has an appealing property: the sparsity of graph weights, by which irrelevant graphs can be easily eliminated (provided that we had some reliable model selection criteria). We stress that this feature was highly important for integrating multiple graphs in label propagation and had not been achieved by any existing methods. We have presented the efficient optimization algorithms in SMGI which allow the clear interpretation of our formulation. Experimental results have demonstrated that SMGI has a superior graph selection ability as well as the highest prediction performance among the state-of-the-art methods for integrating multiple graphs under the problem setting of label propagation.

Appendix A
Extention to Multiclass Problem

Our approach can also be applied to multiclass problems. Suppose that the output label has c classes, i.e., $y_i \in \{1, \dots, c\}$. Let $\boldsymbol{Y} \in \mathbb{B}^{n \times c}$ be the binary label matrix defined by

$$\boldsymbol{Y}_{ij} = \begin{cases} 1, \ i \in \{1, \dots, \ell\}, \ y_i = j, \\ 0, \ \text{otherwise}. \end{cases}$$

Then, using a score matrix $\boldsymbol{F} \in \mathbb{R}^{n \times c}$, our objective function can be defined by

$$\sum_{k=1}^{K} \frac{\mu_k}{Z_k} \operatorname{tr}\left(\boldsymbol{F}^\top \boldsymbol{L}_k \boldsymbol{F}\right) + \lambda_1 \|\boldsymbol{Y} - \boldsymbol{F}\|_F^2 + \lambda_2 \|\boldsymbol{\mu}\|_2.$$

We can use the same alternating minimization approach to minimize this objective function.

Appendix B
Proof of Theorem 1

Proof: We first show that the optimization problem given by (3) is equivalent to the following problem:

$$\min_{\boldsymbol{\mu}} \quad \boldsymbol{v}^\top \boldsymbol{\mu} + \tfrac{\lambda_2}{2} \|\boldsymbol{\mu}\|_2^2 \qquad (13)$$
$$\text{s.t.} \quad \boldsymbol{\mu}^\top \mathbf{1} \geq 1, \ \boldsymbol{\mu} \geq 0.$$

The only difference from the original problem by (3) is the inequality constraint $\boldsymbol{\mu}^\top \mathbf{1} \geq 1$. We can prove that $\boldsymbol{\mu}^\top \mathbf{1} = 1$ holds at the optimal of this problem. Suppose that $\boldsymbol{\mu}^*$ is the optimal solution of this problem but $\boldsymbol{\mu}^{*\top} \mathbf{1} > 1$. Because $\boldsymbol{L}_k$ is positive semidefinite, $v_k = \frac{1}{Z_k} \boldsymbol{f}^\top \boldsymbol{L}_k \boldsymbol{f}$ is nonnegative: $v_k \geq 0$. Thus if $\boldsymbol{\mu}^{*\top} \mathbf{1} > 1$, we can further decrease the objective function by multiplying $\boldsymbol{\mu}^*$ by $1/(\boldsymbol{\mu}^{*\top} \mathbf{1})$ without violating the constraints. As a result, we see that $\boldsymbol{\mu}^{*\top} \mathbf{1} = 1$ must hold at the optimal of (13).

The Lagrangian function of (13) is

$$L = \boldsymbol{v}^\top \boldsymbol{\mu} + \frac{\lambda_2}{2} \boldsymbol{\mu}^\top \boldsymbol{\mu} + \eta(1 - \boldsymbol{\mu}^\top \mathbf{1}) - \boldsymbol{\xi}^\top \boldsymbol{\mu}$$

where $\eta \geq 0$ and $\boldsymbol{\xi} \geq \mathbf{0}$ are the Lagrangian multipliers. By setting the derivatives of the Lagrangian with respect to μ_k as 0, we obtain

$$\mu_k = \frac{\eta - v_k + \xi_k}{\lambda_2}.$$

This derives (4) and (5) in the following manner. First, we consider the case where $\eta - v_k > 0$. Because $\xi_k \geq 0$, we see $\mu_k > 0$ in this case. At the optimal solution, because of the KKT complimentary conditions (see, e.g., [16]), we have $\mu_k \xi_k = 0$. Thus, we see $\xi_k = 0$ if $\mu_k > 0$. On the other hand, if $\eta - v_k < 0$, then $\xi_k > 0$ to satisfy the nonnegative constraint of μ_k. Again, from the KKT complimentary conditions, $\mu_k = 0$ in this case. In the case of $\eta - v_k = 0$, we also see $\mu_k = 0$ for a similar reason. Putting them altogether, the optimality conditions are summarized as follows:

$$\mu_k = \frac{\eta - v_k}{\lambda_2}, \text{for } k \in \{k \mid \eta - v_k > 0\}, \qquad (14)$$
$$\mu_k = 0, \text{for } k \in \{k \mid \eta - v_k \leq 0\}. \qquad (15)$$

Because $\{v_k\}_{k=1}^K$ are sorted in increasing order, using the positive integer $m = |\{k \mid \eta - v_k > 0\}|$, these equations are written as (4) and (5). Then, by substituting (4) and (5) into the equality constraint $\boldsymbol{\mu}^\top \mathbf{1} = 1$, we can obtain the optimal η as (6). ■

Appendix C
Proof of Proposition 2

Proof: Algorithm 3 exploits a property of η as a function of m. Let $\eta_m = (\lambda_2 + \sum_{k=1}^m v_k)/m$. Then η_m has the only one local minimum in terms of m. In other words, $\eta_1 \geq \eta_2 \ldots \geq \eta_t \leq \ldots \leq \eta_K$, where $t \in \{1, \dots, K\}$. This is seen from the following equations:

$$\eta_{m+1} = \frac{\lambda_2 + \sum_{k=1}^{m+1} v_k}{m+1} = \frac{m\frac{\lambda_2+\sum_{k=1}^{m} v_k}{m} + v_{k+1}}{m+1} = \frac{m\eta_m + v_{m+1}}{m+1}.$$

Using the last equation, we see the following relationships:

$$v_{m+1} \le \eta_m \ \Rightarrow\ \eta_{m+1} \in [v_{m+1}, \eta_m],$$
$$v_{m+1} \ge \eta_m \ \Rightarrow\ \eta_{m+1} \in [\eta_m, v_{m+1}].$$

From the first condition, if $v_{m+1} \le \eta_m$, then $\eta_{m+1} \le \eta_m$, meaning that η_m decreases with increase of m, while the first condition is satisfied. On the other hand, because the elements of $\boldsymbol{v}$ are sorted in increasing order, the second condition holds for some $t \in \{1, \dots, K\}$: $v_{t+1} \ge \eta_t$. Then we see $\eta_{t+1} \in [\eta_t, v_{t+1}]$ and obtain $\eta_1 \ge \eta_2 \dots \ge \eta_t \le \dots \le \eta_K$. Using this fact, we do not need to investigate the entire K elements of $\eta\mathbf{1} - \boldsymbol{v}$ in each iteration to count the number of positive elements in all the K elements. In Algorithm 3, $\widehat{m}$ is the number of positive elements in current $\eta\mathbf{1} - \boldsymbol{v}$. After being initialized in line 4, $\widehat{m}$ is gradually decreased in line 18 at every iteration of **for** loop (lines 10–23) until $\widehat{m} = t$. From the next iteration ($m = t+1$), $\widehat{m}$ is gradually increased in line 16 because η should increase at every iteration after $m = t$. This process takes only $O(K)$ computations. ■

Appendix D
Derivation for Difference Between μ_i and μ_j

For the difference between μ_i and μ_j, we can derive the following inequality:

$$(\mu_i - \mu_j)^2 = \left(\frac{\langle \tilde{\boldsymbol{L}}_i - \tilde{\boldsymbol{L}}_j, \boldsymbol{f}\boldsymbol{f}^\top\rangle_F}{\lambda_2}\right)^2 \le \frac{1}{\lambda_2^2}\|\tilde{\boldsymbol{L}}_i - \tilde{\boldsymbol{L}}_j\|_F^2\ \|\boldsymbol{f}\boldsymbol{f}^\top\|_F^2 = \frac{2}{\lambda_2^2}(1 - \langle\tilde{\boldsymbol{L}}_i, \tilde{\boldsymbol{L}}_j\rangle)^2\|\boldsymbol{f}\|_2^4. \quad (16)$$

In the last equation, we use $\|\tilde{\boldsymbol{L}}_i\|_F = \|\tilde{\boldsymbol{L}}_j\|_F = 1$. For $\|\boldsymbol{f}\|_2$, we can also obtain the following bound:

$$\|\boldsymbol{f}\|_2 \le \lambda_1 \left\|\left(\lambda_1\boldsymbol{I} + \sum_{k=1}^K \tfrac{\mu_k}{Z_k}\boldsymbol{L}_k\right)^{-1}\right\|_2 \|\boldsymbol{y}\|_2 \le \sqrt{\lambda_1}\ \|\boldsymbol{y}\|_2.$$

The last inequality is derived from the positive semidefiniteness of $\sum_{k=1}^K \frac{\mu_k}{Z_k}\boldsymbol{L}_k$ and a property of Euclidean norm for matrix: for any real valued matrix $\boldsymbol{M}$, $\|\boldsymbol{M}\|_2^2$ is equal to the maximum eigenvalue of $\boldsymbol{M}$. Combining this bound with (16), we can obtain (8).

Appendix E
Equivalence Between Alternating Minimization and CCCP

Let

$$E_{vex}(\boldsymbol{\mu}) = \tfrac{\lambda_2}{2}\|\boldsymbol{u}\|_2^2$$

and

$$E_{cave}(\boldsymbol{\mu}) = -\lambda_1^2\boldsymbol{y}^\top\left(\lambda_1\boldsymbol{I} + \sum_{k=1}^K \tfrac{\mu_k}{Z_k}\boldsymbol{L}_k\right)^{-1}\boldsymbol{y}.$$

Note that $E_{vex}(\boldsymbol{\mu})$ is a convex function and $E_{cave}(\boldsymbol{\mu})$ is a concave function. Using $E_{vex}(\boldsymbol{\mu})$ and $E_{cave}(\boldsymbol{\mu})$, the objective function of our problem is

$$E_{vex}(\boldsymbol{\mu}) + E_{cave}(\boldsymbol{\mu}).$$

Here, we omit the constant term $\lambda_1\|\boldsymbol{y}\|_2^2$ in (9). Suppose that $\boldsymbol{\mu}^t = (\mu_1^t, \dots, \mu_K^t)^\top$ is the solution at the tth iteration, CCCP updates the solution by solving the following problem:

$$\boldsymbol{\mu}^{t+1} = \arg\min_{\boldsymbol{\mu}}\ E_{vex}(\boldsymbol{\mu}) + \boldsymbol{\mu}^\top\tfrac{\partial E_{cave}(\boldsymbol{\mu}^t)}{\partial\boldsymbol{\mu}} \quad \text{s.t.}\quad \boldsymbol{\mu}^\top\mathbf{1} = 1,\ \boldsymbol{\mu} \ge 0. \quad (17)$$

This problem linearly approximates the concave part of the original objective function and the derivative of $E_{cave}(\boldsymbol{\mu})$ is written as

$$\frac{\partial E_{cave}(\boldsymbol{\mu}^t)}{\partial\mu_k} = (\boldsymbol{f}^t)^\top\frac{1}{Z_k}\boldsymbol{L}_k\boldsymbol{f}^t$$

where

$$\boldsymbol{f}^t = \lambda_1\left(\lambda_1\boldsymbol{I} + \sum_{k'=1}^K \frac{\mu_{k'}^t}{Z_{k'}}\boldsymbol{L}_{k'}\right)^{-1}\boldsymbol{y}.$$

Because $v_k = 1/Z_k\boldsymbol{f}^\top\boldsymbol{L}_k\boldsymbol{f}$, the problem given by (17) is equivalent to the problem given by (3) in our alternating minimization algorithm.

Appendix F
Derivation for EGMKL

Here, we describe a derivation of the elastic net regularization for GMKL based on [44] and [50]. The objective function of EGMKL (12) can be written as

$$\min_{\{f_k\}_{k=1}^K,\boldsymbol{\mu}}\ \sum_{i=1}^{\ell}(y_i - f_i)^2 + C\sum_{k=1}^K\left(\frac{\|f_k\|_{\boldsymbol{K}^{(k)}}^2}{2\mu_k} - g^*\left(\frac{1}{2\mu_k}\right)\right)$$

where $g^*(1/(2\mu_k)) = ((1-\lambda)^2\mu_k)/(2 - 2\lambda\mu_k)$. The following simple update equation for μ_k can be derived from this formulation:

$$\mu_k = \frac{\|\boldsymbol{f}_k\|_{\boldsymbol{K}^{(k)}}}{1 - \lambda + \lambda\|\boldsymbol{f}_k\|_{\boldsymbol{K}^{(k)}}}.$$

Using the representer theorem, we can solve the above problem in terms of $\boldsymbol{f}_k$ easily (see [44] and [50] for detail). We simply optimize $\boldsymbol{f}_k$ and $\boldsymbol{\mu}$ with alternating minimization.

References

[1] A. Blum and S. Chawla, “Learning from labeled and unlabeled data using graph mincuts,” in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 19–26.

[2] M. Szummer and T. Jaakkola, “Partially labeled classification with Markov random walks,” in *Advances in Neural Information Processing Systems*, vol. 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA, USA: MIT Press, 2001, pp. 945–952.

[3] T. Joachims, “Transductive learning via spectral graph partitioning,” in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 290–297.

[4] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning: From Gaussian fields to Gaussian processes," in *Proc. 20th Annu. Int. Conf. Mach. Learn.*, 2003.

[5] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems*, vol. 16, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA, USA: MIT Press, 2004.

[6] M. Herbster, M. Pontil, and L. Wainer, "Online learning over graphs," in *Proc. 22nd Annu. Int. Conf. Mach. Learn.*, 2005, pp. 305–312.

[7] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: From transductive to semi-supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 824–831.

[8] 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, Jan. 2006.

[9] Y. Bengio, O. Delalleau, and N. Le Roux, "Label propagation and quadratic criterion," in *Semi-Supervised Learning*, O. Chapelle, B. Schölkopf, and A. Zien, Eds. Cambridge, MA, USA: MIT Press, 2006, pp. 193–216.

[10] K. Tsuda, H. Shin, and B. Scholkopf, "Fast protein classification with multiple networks," *Bioinformatics*, vol. 21, pp. 59–65, Sep. 2005.

[11] T. Kato, H. Kashima, and M. Sugiyama, "Robust label propagation on multiple networks," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 35–44, Jan. 2009.

[12] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, "Unified video annotation via multigraph learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 5, pp. 733–746, May 2009.

[13] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. R. Stat. Soc. Ser. B*, vol. 67, no. 2, pp. 301–320, 2005.

[14] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, Dec. 2004.

[15] A. Argyriou, M. Herbster, and M. Pontil, "Combining graph Laplacians for semi-supervised learning," in *Advances in Neural Information Processing Systems*, vol. 18, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA, USA: MIT Press, 2006, pp. 67–74.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[17] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, "Pathwise coordinate optimization," *Ann. Appl. Stat.*, vol. 1, no. 2, pp. 302–332, 2007.

[18] A. L. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Comput.*, vol. 15, no. 4, pp. 915–936, 2003.

[19] B. Sriperumbudur and G. Lanckriet, "On the convergence of the concave-convex procedure," in *Advances in Neural Information Processing Systems*, vol. 22, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Cambridge, MA, USA: MIT Press, 2009, pp. 1759–1767.

[20] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proc. 36th Annu. ACM Symp. Theory Comput.*, 2004, pp. 81–90.

[21] A. Argyriou, C. A. Micchelli, and M. Pontil, "Learning convex combinations of continuously parameterized basic kernels," in *Proc. 18th Annu. Conf. Learn. Theory*, vol. 3559. 2005, pp. 338–352.

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

[23] P. P. Talukdar, "Topics in graph construction for semi-supervised learning," Dept. Comput. Inf. Sci., Univ. Pennsylvania, Philadelphia, PA, USA, Tech. Rep. MS-CIS-09-13, 2009.

[24] W. Tang, Z. Lu, and I. S. Dhillon, "Clustering with multiple graphs," in *Proc. 9th IEEE ICDM*, Dec. 2009, pp. 1016–1021.

[25] S. Yu, X. Liu, L.-C. Tranchevent, W. Glänzel, J. A. K. Suykens, B. De Moor, and Y. Moreau, "Optimized data fusion for k-means Laplacian clustering," *Bioinformatics*, vol. 27, no. 1, pp. 118–126, Jan. 2011.

[26] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th Annu. Int. Conf. Mach. Learn.*, 1999, pp. 200–209.

[27] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. 10th Int. Workshop Artif. Intell. Stat.*, 2005, pp. 57–64.

[28] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. 24th Annu. Int. Conf. Mach. Learn.*, 2007, pp. 759–766.

[29] 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.

[30] R. Soares, H. Chen, and X. Yao, "Semisupervised classification with cluster regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 11, pp. 1779–1792, Nov. 2012.

[31] Y. Wang, S. Chen, and Z.-H. Zhou, "New semi-supervised classification method based on modified cluster assumption," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 5, pp. 689–702, May 2012.

[32] V. R. de Sa, "Learning classification with unlabeled data," in *Advances in Neural Information Processing Systems*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. San Francisco, CA, USA: Morgan Kaufmann, 1993, pp. 112–119.

[33] J. D. R. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szedmak, "Two view learning: SVM-2K, theory and practice," in *Advances in Neural Information Processing Systems*, vol. 18, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA, USA: MIT Press, 2006, pp. 355–362.

[34] U. Brefeld, T. Gärtner, T. Scheffer, and S. Wrobel, "Efficient co-regularised least squares regression," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 137–144.

[35] V. Sindhwani, P. Niyogi, and M. Belkin, "A co-regularization approach to semi-supervised learning with multiple views," in *Proc. 22nd ICML Workshop Learn. Multiple Views*, 2005, pp. 1–6.

[36] D. Zhou and C. J. C. Burges, "Spectral clustering and transductive learning with multiple views," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1159–1166.

[37] M. Culp, G. Michailidis, and K. Johnson, "On multi-view learning with additive models," *Ann. Appl. Stat.*, vol. 3, no. 1, pp. 292–318, Mar. 2009.

[38] M. Culp, "On propagated scoring for semisupervised additive models," *J. Amer. Stat. Assoc.*, vol. 106, no. 493, pp. 248–259, 2011.

[39] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris, "GeneMANIA: A real-time multiple association network integration algorithm for predicting gene function," *Genome Biol.*, vol. 9, pp. S4–S15, Jun. 2008.

[40] S. Mostafavi and Q. Morris, "Fast integration of heterogeneous data sources for predicting gene function with limited annotation," *Bioinformatics*, vol. 26, no. 14, pp. 1759–1765, 2010.

[41] D. Warde-Farley, S. L. Donaldson, O. Comes, K. Zuberi, R. Badrawi, P. Chao, M. Franz, C. Grouios, F. Kazi, C. T. Lopes, A. Maitland, S. Mostafavi, J. Montojo, Q. Shao, G. Wright, G. D. Bader, and Q. Morris, "The GeneMANIA prediction server: Biological network integration for gene prioritization and predicting gene function," *Nucleic Acids Res.*, vol. 38, pp. W214–W220, May 2010.

[42] N. Cristianini, J. Shawe-Taylor, and J. Kandola, "On kernel target alignment," in *Advances in Neural Information Processing Systems*, vol. 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA, USA: MIT Press, 2002, pp. 367–373.

[43] J. Shawe-Taylor, "Kernel learning for novelty detection," in *Proc. NIPS Workshop, Kernel Learn., Autom. Sel. Optim. Kernels*, 2008, pp. 1–45.

[44] R. Tomioka and T. Suzuki, "Sparsity-accuracy trade-off in MKL," in *Proc. NIPS Workshop, Understand. Multiple Kenrel Learn.*, 2009, pp. 1–8.

[45] F. Orabona and J. Luo, "Ultra-fast optimization algorithm for sparse multi kernel learning," in *Proc. 28th ICML*, Jun. 2011, pp. 249–256.

[46] Z.-P. Wu and X.-G. Zhang, "Elastic multiple kernel learning," *Acta Autom. Sinica*, vol. 37, no. 6, pp. 693–699, 2011.

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

[48] W. Liu and S.-F. Chang, "Robust multi-class transductive learning with graphs," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 381–388.

[49] G. R. G. Lanckriet, T. D. Bie, N. Cristianini, M. I. Jordan, and W. S. Noble, "A statistical framework for genomic data fusion," *Bioinformatics*, vol. 20, no. 16, pp. 2626–2635, 2004.

[50] T. Suzuki and R. Tomioka, "SpicyMKL: A fast algorithm for multiple kernel learning with thousands of kernels," *Mach. Learn.*, vol. 85, nos. 1–2, pp. 77–108, 2011.

Masayuki Karasuyama received the B.E, M.E., and Ph.D. degrees from the Nagoya Institute of Technology, Nagoya, Japan, in 2006, 2008, and 2011, respectively.

He is currently an Assistant Professor with the Bio-Knowledge Engineering Research Laboratory, Bioinformatics Center, Institute for Chemical Research, Kyoto University, Kyoto, Japan. His current research interests include statistical machine learning and its applications to 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 from the University of Tokyo, Tokyo, Japan, in 1988, 1991, and 1999, respectively.

He is involved in research on machine learning, data mining, and bioinformatics. His current research interests include mining from graphs and networks in biology and chemistry