Semisupervised Kernel Matrix Learning by Kernel Propagation

Enliang Hu, Songcan Chen, Daoqiang Zhang, and Xuesong Yin

***Abstract*— The goal of semisupervised kernel matrix learning (SS-KML) is to learn a kernel matrix on all the given samples on which just a little supervised information, such as class label or pairwise constraint, is provided. Despite extensive research, the performance of SS-KML still leaves some space for improvement in terms of effectiveness and efficiency. For example, a recent pairwise constraints propagation (PCP) algorithm has formulated SS-KML into a semidefinite programming (SDP) problem, but its computation is very expensive, which undoubtedly restricts PCPs scalability in practice. In this paper, a novel algorithm, called kernel propagation (KP), is proposed to improve the comprehensive performance in SS-KML. The main idea of KP is first to learn a small-sized sub-kernel matrix (named seed-kernel matrix) and then propagate it into a larger-sized full-kernel matrix. Specifically, the implementation of KP consists of three stages: 1) separate the supervised sample (sub)set $\mathcal{X}_l$ from the full sample set $\mathcal{X}$; 2) learn a seed-kernel matrix on $\mathcal{X}_l$ through solving a small-scale SDP problem; and 3) propagate the learnt seed-kernel matrix into a full-kernel matrix on $\mathcal{X}$. Furthermore, following the idea in KP, we naturally develop two conveniently realizable out-of-sample extensions for KML: one is batch-style extension, and the other is online-style extension. The experiments demonstrate that KP is encouraging in both effectiveness and efficiency compared with three state-of-the-art algorithms and its related out-of-sample extensions are promising too.**

***Index Terms*— Kernel propagation, out-of-sample extension, pairwise constraint, seed-kernel matrix learning, semidefinite programming.**

I. Introduction

KERNEL methods are very popular for dealing with nonlinear problems in modern pattern recognition and machine learning communities, and recently have attracted great attention toward kernel learning, which supposes that a learnt kernel can fit the given data better than a predefined kernel. In theory, a kernel corresponds to a reproducing kernel Hilbert space (RKHS) and kernel learning is equivalent to kernel matrix leaning (KML) when only the inner products of input data are involved. On the other hand, learning from both supervised and unsupervised samples, being well known as semisupervised learning (SSL), has grown into another hot topic in machine learning during last several years [1]. If taking SSL as kernel matrix design, SSL will turn to SS-KML. In SS-KML, besides a large number of unsupervised samples to reflect data structure [2], their partial supervised information such as pairwise constraint is available to deliver user preferences [3]. So, the principal goal of SS-KML is not only to fit the data structure better but also to cater for the user preferences closer.

Recently, the spectral transformation method [4], [5] has laid the necessary foundation for SS-KML researches. For example, cluster kernel [6], diffusion kernel [7], Gaussian-field kernel [8], and improved-order kernel [9] are all based on the assumption that the target kernel matrix has the same eigenvectors as those of graph Laplacian L while the eigenvalues are different. This implies that, if $L = \sum_i \lambda_i V_i V_i^T$, then $K = \sum_i \mu(\lambda_i) V_i V_i^T$, in which $\mu(\lambda)$ is a positive and nonincreasing function. For example, the Gaussian-field kernel corresponds to $\mu(\lambda) = 1/(\lambda + \epsilon)$ in which ϵ as a hyperparameter needs to be tuned through maximizing the kernel alignment score between K and training labels [9]. Different from the spectral transformation method, an SS-KML algorithm called pairwise constraints propagation (PCP) was recently suggested by Li *et al*. [10], which is formulated as a semidefinite programming (SDP) problem through optimizing the smooth measure while obeying the pre-given pairwise constraints as follows:

$$\begin{aligned} \min_{K} \quad & : \quad Tr(LK) \\ \text{s.t.} \quad & : \quad K(i,i) = 1, \qquad \forall x_i \in \mathcal{X} \\ & \quad\;\; K(i,j) = 1, \qquad \forall (x_i, x_j) \in \mathcal{M} \\ & \quad\;\; K(i,j) = 0, \qquad \forall (x_i, x_j) \in \mathcal{C} \\ & \quad\;\; K \succ 0 \end{aligned} \tag{1}$$

where $(Tr(LK) = 1/2 \sum_{i,j=1}^{n} w_{ij} \|\phi(x_i) - \phi(x_j)\|^2_{\mathcal{H}_K}$ ($Tr(\cdot)$ denotes the trace operator) measures the smoothness of K (smaller is smoother) and $\mathcal{M}(\mathcal{C})$ consists of must (cannot)-link constraints. Since a kernel implicitly corresponds to an embedding mapping ϕ [10], the first group constraints in (1) implies the unit-length constraint for each $\phi(x_i)$, which requires mapping each sample onto the unit hypersphere, and the rest require each pair of samples in $\mathcal{M}$ sharing the same embedding while those in $\mathcal{C}$ being orthogonal with each other.

Despite extensive researches for SS-KML, the current performances are still far from satisfaction in both effectiveness and efficiency [11]. For example, spectral transformation

Manuscript received April 10, 2010; revised September 1, 2010; accepted September 1, 2010. Date of current version November 3, 2010. This work was supported in part by National Natural Science Foundation of China under Grant 60773061, Grant 60875030, and Grant 61035003, and in part by the Natural Science Foundation of Jiangsu under Grant BK2008381.

E. Hu is with the Department of Mathematics, Yunnan Normal University, Kunming 650092, China (e-mail: helnuaa@nuaa.edu.cn).

S. Chen and D. Zhang are with the Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China (e-mail: s.chen@nuaa.edu.cn; dqzhang@nuaa.edu.cn).

X. Yin is with the School of Information and Engineering, Zhejiang Radio and TV University, Hangzhou 310030, China (e-mail: yinxs@nuaa.edu.cn).

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.2010.2076301

1045–9227/$26.00 © 2010 IEEE

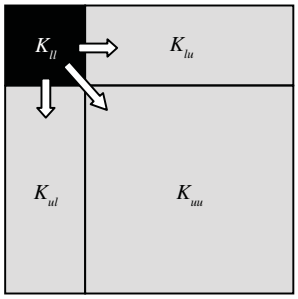

Fig. 1. Illustration of KP. An unknown full-kernel matrix K is split into four sub-blocks K_{ll}, K_{lu}, K_{ul} $(= K_{lu}^T)$, and K_{uu}. If K_{ll} equals a known seed-kernel matrix, then KP aims to propagate K_{ll} into the other unknown blocks K_{lu}, K_{ul}, and K_{uu} and thus makes the full-kernel matrix K to be known.

methods are so tightly dependent on the quality of the eigenvectors of graph Laplacian that a group of weak eigenvectors, despite being optimally combined, will lead to their ineffectiveness. The reason is that, it turns out that each high-density area uniquely corresponds to a representative eigenvector of graph Laplacian when the data are clustered [12], however, if the same-class data are distributed in multiple-different clusters, their corresponding eigenvectors will fail to conform to the class relationship so that they are almost uninformative and thus very weak. On the other hand, though good effectiveness was reported for PCP [10], solving its associated SDP problem often encounters a computational bottleneck, which makes the application of PCP impractical in large-scale problems.

To further improve the performance for SS-KML, we propose a novel algorithm called kernel propagation (KP) in this paper. Analogous to the fact that label propagation (LP) [13]–[16] propagates the seed labels from the labeled samples to the unlabeled samples, KP aims to learn and then propagate a small-sized sub-kernel matrix into a large-sized full-kernel matrix. More specifically, KP performs the three stages: 1) separate the supervised sample (sub)set $\mathcal{X}_l$ from the full sample set $\mathcal{X}$; 2) learn a sub-kernel matrix, called seed-kernel matrix on the $\mathcal{X}_l$; and 3) propagate the learnt seed-kernel matrix into a full-kernel matrix as the final target. Furthermore, following the idea in KP, we respectively develop a batch-style and an online-style out-of-sample extensions (BE and OE) to deal with KML for out-of-sample data [17], [18], meaning that for the newly received data in future a kernel matrix can be straightforwardly captured in batch or online mode without the need of relearning. The encouraging results from the experiments show the superiority of KP in effectiveness and efficiency compared with the three state-of-the-art algorithms.

The rest of this paper is organized as follows. In Section II, we briefly review LP. In Section III, we first introduce KP, and then develop the two out-of-sample extensions following the idea in KP. In Section IV, the experimental results are provided. Conclusions are presented in the last section.

II. Brief Review of LP

The underlying intuition of graph transductive learning lies in that two nodes (or data points) are likely to share the same class label if they can be connected by the graph paths

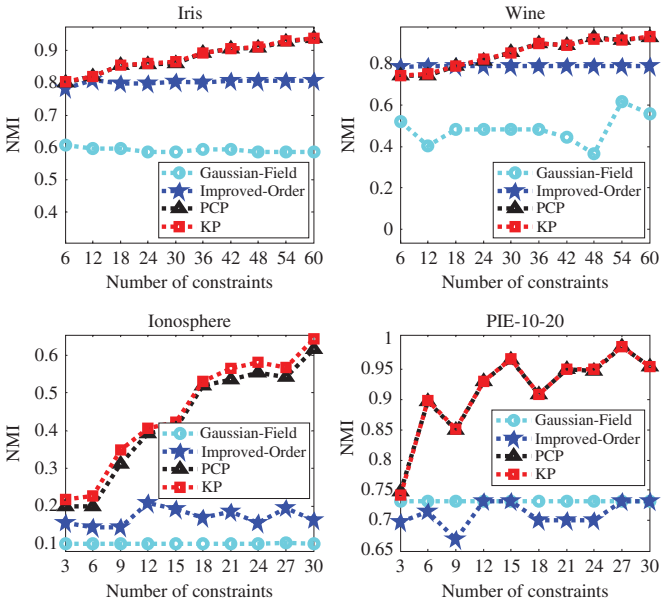

Fig. 2. Comparison of average classification accuracies [in normalized mutual information (NMI)] for varying number of pairwise constraints across four datasets: Iris, Wine, Ionosphere, and PIE-10-20. Higher is better.

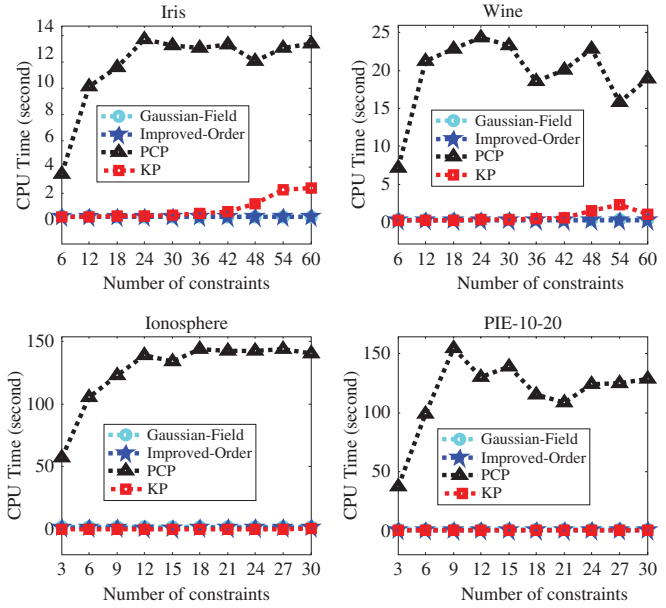

Fig. 3. Comparison of average running times (in seconds) for varying number of pairwise constraints across four datasets: Iris, Wine, Ionosphere, and PIE-10-20. Lower is better.

passing through high-density areas [12]. As a special method for transductive learning, LP [14] focuses on propagating the seed labels from the labeled samples to the unlabeled samples. Given a sample subset $\mathcal{X}_l = \{x_i\}_{i=1}^{l}$ plus its label vector $\mathcal{Y}_l$ and an unlabeled-sample subset $\mathcal{X}_u = \{x_i\}_{i=l+1}^{l+u}$, a graph $\mathcal{G} = (v, \varepsilon, W)$ can be first constructed in terms of its node set $v = \mathcal{X}_l \cup \mathcal{X}_u (|v| = n)$, edge set ε, and edge-weight matrix $W = (w_{ij})_{n\times n}$ in which w_{ij} reflects the similarity between nodes x_i and x_j. Then, a so-called normalized graph Laplacian can be calculated as $L = D^{-1/2}(D - W)D^{-1/2}$, where $D = \text{diag}([d_{ii}]_{n\times n})$ and $d_{ii} = \sum_j w_{ij}$.

TABLE I
KP-ALGORITHM FOR SS-KML

Input:
$\mathcal{X} = \{x_i\}_{i=1}^n$ — Full sample set;
$\mathcal{M} = \{(x_i, x_j)\}_{i,j}$— Set of must-link constraints;
$\mathcal{C} = \{(x_i, x_j)\}_{i,j}$— Set of cannot-link constraints;
L — Graph Laplacian on $\mathcal{X}$.
Output: K^* — Full kernel matrix on $\mathcal{X}$.

Step 1: Split $\mathcal{X}$ into $\mathcal{X}_l = \{x_i | (x_i, \cdot) \in \mathcal{M} \cup \mathcal{C}\}$ and $\mathcal{X}_u = \mathcal{X}/\mathcal{X}_l$;
Step 2: Learn a seed-kernel matrix K_{ll} on $\mathcal{X}_l$ through solving (8);
Step 3: Obtain a full-kernel matrix K after putting K_{ll} into (5);
Step 4: Return $K^* = \hat{D}^{-1/2} K \hat{D}^{-1/2}$ in terms of $\hat{D} = \mathrm{diag}(K)$.

TABLE II
ONLINE OUT-OF-SAMPLE EXTENSION FOR KML

Input:
$\mathcal{X}_\mathrm{I}$ — In-sample data ($|\mathcal{X}_\mathrm{I}| = n$);
$\mathcal{K}_\mathrm{II}$ — Kernel matrix on $\mathcal{X}_\mathrm{I}$;
$\mathcal{X}_\mathrm{O} = \{x_{n+1}, \ldots, x_{n+m}\}$— A sequence of out-of-sample data.
Output:
K_Full — Full kernel matrix on $\mathcal{X}_l \cup \{x_{n+1}, \ldots, x_{n+m}\}$.

Initialization: Set $\mathrm{I} = \{1, 2, \ldots, n\}$ and $K^{(n)} = \mathcal{K}_\mathrm{II}$.
For $\mathrm{i} = n+1$ to $n+m$
Step 1: Update graph Laplacian on $\mathcal{X}_\mathrm{I} \cup \{x_i\}$ as
$$L^{(\mathrm{i})} = \begin{pmatrix} L_\mathrm{II} & L_\mathrm{Ii} \\ L_\mathrm{Ii}^T & L_\mathrm{ii} \end{pmatrix};$$
Step 2: Update $Q_{(\mathrm{i})}$ and $K^{(\mathrm{i})}$ as
$$Q_{(i)} = \begin{pmatrix} I_\mathrm{I} \\ \frac{1}{-L_\mathrm{ii}} L_\mathrm{Ii}^T \end{pmatrix}, K^{(\mathrm{i})} = Q_{(i)} K^{(\mathrm{i}-1)} Q_{(\mathrm{i})}^T;$$
Step 3: Update $\mathrm{I} \longleftarrow \mathrm{I} \cup \{\mathrm{i}\}$;
End
Return $K_\mathrm{Full} = K^{(n+m)}$.

Let $f = [f_l, f_u]^T$ as a label function to be determined, and the LP in [28] is formulated as

$$\begin{aligned} \min_f \quad &: \quad Tr(f^T L f) \\ \text{s.t.} \quad &: \quad f_l = \mathcal{Y}_l \end{aligned} \tag{2}$$

after a simple calculation to (2), we can obtain $f_u = -L_{uu}^{-1} L_{ul} \mathcal{Y}_l$ and set it as the label vector of $\mathcal{X}_u$, where L_{uu} is the sub-part L (i.e., restricted to $\mathcal{X}_u$).

III. KP FOR SS-KML

A. KP Formulation

Inspired by LP, the proposed KP borrows a similar principle of LP. However, instead of propagating seed labels as in LP, KP aims to learn first and then propagate a seed-kernel matrix into a full-kernel matrix.

TABLE III
SOME DESCRIPTIONS OF THE SEVEN DATASETS

Dataset	# Samples	# Classes	# Dimensions	Citation
Iris	150	3	4	[10]
Wine	178	3	13	[10]
Ionosphere	351	2	34	[10]
PIE-10-20	340	2	1024	[10]
USPS0123	3000	4	256	[10]
MNIST0123	3000	4	784	[10]
Control	600	6	60	[23]

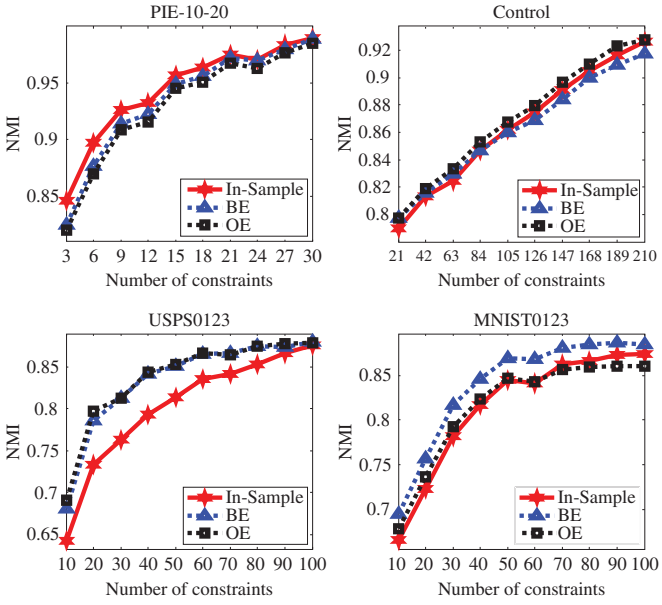

Fig. 4. Comparison of average classification accuracies (in NMI) for varying number of pairwise constraints for KP on in-sample data, BE and OE on out-of-sample data across four datasets: PIE-10-20, Control, USPS0123, and MNIST0123. Higher is better.

In order to easily understand KP, we first give an illustration in Fig. 1, where an unknown full-kernel matrix K is split into the four sub-blocks K_{ll}, K_{lu}, K_{ul} ($= K_{lu}^T$ for symmetry), and K_{uu}. Without loss of generality, let K_{ll} be a known seed-kernel matrix; then KP aims to propagate K_{ll} to the other unknown sub-blocks K_{lu}, K_{ul}, and K_{uu} so that it produces a known full-kernel matrix on $\mathcal{X}$. If the user preferences have been encoded into K_{ll}, KP can be viewed to diffuse such preferences from K_{ll} to the other sub-parts.

As is well known, a kernel matrix K corresponds to an embedding mapping. $\phi : \mathcal{X} \mapsto \mathcal{H}_K$ Li *et al.* [10] utilized $Tr(LK) = 1/2 \sum_{i,j=1}^n w_{ij} \|\phi(x_i) - \phi(x_j)\|_{\mathcal{H}_K}^2$ to measure the smoothness of K (or ϕ), where $\mathcal{H}_K$ is the RKHS induced by K. Likewise, depending on the above smoothness measure, we formulate KP to be a minimization problem as

$$\begin{aligned} \min_K \quad &: \quad Tr(LK) \\ \text{s.t.} \quad &: \quad {}_g K_g{}^T = K_{ll,g} = [I_l, 0] \\ &\quad\quad K \succeq 0 \end{aligned} \tag{3}$$

where K is reorganized as $K = \begin{pmatrix} K_{ll} & K_{lu} \\ K_{lu}^T & K_{uu} \end{pmatrix}$, of which the sub-part $K_{ll} = {}_g K_g{}^T$ equals to a known seed-kernel matrix

TABLE IV

COMPARISON OF AVERAGE CLASSIFICATION ACCURACIES (IN NMI) AND RUNNING TIMES (IN SECONDS) FOR VARYING NUMBER OF PAIRWISE CONSTRAINTS ON CONTROL (SIZE OF 600). BEST RESULTS ARE UNDERLINED AND IN BOLD. (IN COLUMNS 2–5, EACH CELL HAS TWO ROWS: THE UPPER IS THE NMI ACCURACY PLUS ITS STANDARD DEVIATION AND THE LOWER IS THE RUNNING TIME)

# Constraints	KP	PCP	Improved-order	Gaussian-field
21	**0.87** ± 2.32E-02 **0.65**	**0.87** ± 3.15E-02 213.39	0.82 ± 1.92E-02 3.10	0.81 ± 0.82E-02 5.02
42	**0.90** ± 3.46E-02 **1.41**	**0.90** ± 3.31E-02 413.79	0.82 ± 2.43E-02 3.07	0.81 ± 0.82E-02 5.03
63	**0.92** ± 3.59E-02 3.3	**0.92** ± 3.50E-02 542.23	0.83 ± 1.72E-02 **3.08**	0.81 ± 0.80E-02 5.05
84	**0.93** ± 2.93E-02 8.48	**0.93** ± 3.03E-02 547.01	0.83 ± 0.87E-02 **3.07**	0.81 ± 0.85E-02 5.05
105	**0.95** ± 2.68E-02 9.21	**0.95** ± 2.71E-02 543.23	± 0.83E-02 **3.08**	0.82 ± 0.87E-02 0.83 5.06
126	**0.97** ± 1.30E-02 19.43	**0.97** ± 1.36E-02 529.59	0.83 ± 0.15E-02 **3.08**	0.82 ± 0.88E-02 5.09
147	**0.98** ± 1.55E-02 37.79	**0.98** ± 1.44E-02 522.14	0.83 ± 0.43E-02 **3.08**	0.82 ± 0.89E-02 5.06
168	**0.98** ± 2.01E-02 58.63	**0.98** ± 2.02E-02 538.87	0.83 ± 0.73E-02 **3.07**	0.82 ± 0.89E-02 5.12
189	**0.98** ± 0.83E-02 74.73	**0.98** ± 0.81E-02 548.22	0.83 ± 0.64E-02 **3.08**	0.82 ± 0.89E-02 5.10
210	**0.98** ± 0.92E-02 83.86	**0.98** ± 0.91E-02 552.53	0.83 ± 0.06E-02 **3.08**	0.81 ± 0.84E-02 5.11
Average score	**0.94** ± 2.16E-02 29.75	**0.94** ± 2.23E-02 495.10	0.83 ± 0.97E-02 **3.08**	0.82 ± 85E-02 5.07

K_{ll} and I_l is the identity matrix of proper size. Because of the fewer constraints imposed, less computation is needed to optimize the SDP for (1), the intention behind the formulation (3) is that, to reduce the number of unit-length constraints in (1), we will only impose the "$K(i,i) = 1$" constraint to each sample $x_i \in \mathcal{M} \cup \mathcal{C}$, while for all samples not in $\mathcal{M} \cup \mathcal{C}$ we defer unit-length constraints to a postprocessing step after finishing the SDP optimization, i.e., we perform a diagonal normalization to the solution matrix obtained by solving the SDP (detailed in Table I).

For (3), K has the decomposition of $K = BB^T$ due to its symmetry and positive semidefiniteness, where B is a real matrix. After re-permuting and denoting

$$B = \begin{pmatrix} B_l \\ B_u \end{pmatrix}, \qquad L = \begin{pmatrix} L_u & L_{lu} \\ L_{lu}^T & L_{uu} \end{pmatrix}$$

we can reformulate (3) as

$$\begin{aligned} \min_{B} \quad & : \quad Tr\left(B^T L B\right) \\ \text{s.t.} \quad & : \quad {}_g B B^T g^T = K_{ll} \\ & \quad g = [I_l, 0] \\ \Rightarrow \min_{B_u} \quad & : \quad Tr\left(B_l^T L_{ll} B_l + 2B_u^T L_{lu}^T B_l + B_u^T L_{uu} B_u\right) \\ \text{s.t.} \quad & : \quad B_l B_l^T = K_{ll} \\ \Rightarrow \min_{B_u} \quad & : \quad \left[2Tr\left(B_u^T L_{lu}^T B_l\right) + Tr\left(B_u^T L_{uu} B_u\right)\right]. \end{aligned} \tag{4}$$

For minimizing (4), we compute and set its derivative w.r.t. B_u to zero, thus have

$$\begin{aligned} & 2L_{lu}^T B_l + 2L_{uu} B_u = 0 \\ \Rightarrow \quad & B_u = -L_{uu}^{-1} L_{lu}^T B_t \\ \Rightarrow \quad & B = \begin{pmatrix} I_l \\ -L_{uu}^{-1} L_{lu}^T \end{pmatrix} B_l. \end{aligned}$$

Denoting $Q = \begin{pmatrix} I_l \\ -L_{uu}^{-1} L_{lu}^T \end{pmatrix}$, we can rewrite the solution of (3) as

$$\begin{aligned} & K = BB^T = Q B_l B_l^T Q^T = Q K_{ll} Q^T \\ \Rightarrow \quad & K = \begin{pmatrix} K_{ll} & -K_{ll} L_{lu} L_{uu}^{-1} \\ -L_{uu}^{-1} L_{lu}^T K_{ll} & L_{uu}^{-1} L_{lu}^T K_{ll} L_{lu} L_{uu}^{-1} \end{pmatrix}. \end{aligned} \tag{5}$$

Equation (5) means that (3) uniquely has a closed-form solution and it is of relatively low rank due to $\text{rank}(K) \leq \text{rank}(K_{ll}) \leq l \ll n$. Furthermore, after inserting (5) into (3), its objective (i.e., smoothness measure) can be reformulated as

$$\begin{aligned} & Tr(LK) = Tr\left(LQK_{ll}Q^T\right) \\ = \quad & Tr(L_{ll}K_{ll}) - Tr\left(L_{lu}L_{uu}^{-1}L_{lu}^T K_{ll}\right) \\ = \quad & Tr\left[\left(L_{ll} - L_{lu}L_{uu}^{-1}L_{lu}^T\right)K_{ll}\right]. \end{aligned} \tag{6}$$

From (5) and (6), we notice that the closed-form solution and the objective of (3) *just* depend on the seed-kernel matrix K_{ll}. In other words, if obtaining K_{ll} then obtaining the desired full-kernel matrix K. However, K_{ll} is usually unknown and

TABLE V

COMPARISON OF AVERAGE CLASSIFICATION ACCURACIES (IN NMI) AND RUNNING TIMES (IN SECONDS) FOR VARYING NUMBER OF PAIRWISE CONSTRAINTS ON USPS0123 (SIZE OF 1000). BEST RESULTS ARE UNDERLINED AND IN BOLD. (IN COLUMNS 2–5, EACH CELL HAS TWO ROWS: THE UPPER IS THE AVERAGE NMI ACCURACY PLUS ITS STANDARD DEVIATION AND THE LOWER IS THE AVERAGE RUNNING TIME)

# Constraints	KP	PCP	Improved-order	Gaussian-field
10	0.69 ± 7.89E-02 **1.72**	0.69 ± 8.77E-02 387.92	0.75 ± 4.50E-02 12.97	**0.76** ± 1.33E-02 17.21
20	**0.79** ± 5.16E-02 **1.79**	0.77 ± 5.88E-02 1460.44	**0.79** ± 4.77E-02 12.94	0.76 ± 1.23E-02 17.22
30	**0.82** ± 3.31E-02 **2.01**	0.80 ± 4.01E-02 1938.96	0.80 ± 4.94E-02 12.94	0.76± 1.11E-02 17.24
40	**0.85** ± 3.17E-02 **2.19**	0.84 ± 3.23E-02 2336.51	0.78 ± 4.24E-02 12.94	0.76 ± 0.74E-02 17.16
50	**0.87** ± 2.93E-02 **2.64**	0.86 ± 3.08E-02 2303.51	0.77 ± 4.96E-02 12.94	0.76 ± 1.42E-02 17.11
60	**0.87** ± 4.00E-02 **3.64**	0.86 ± 3.96E-02 2347.76	0.79 ± 4.37E-02 12.94	0.76 ± 0.94E-02 17.19
70	**0.90** ± 2.74E-02 **4.06**	0.89 ± 2.8E-02 2469.94	0.79 ± 4.34E-02 12.94	0.76 ± 0.93E-02 17.12
80	**0.91** ± 2.67E-02 **4.56**	0.90 ± 2.73E-02 2413.87	0.79 ± 3.84E-02 12.94	0.76 ± 1.00E-02 17.20
90	**0.91** ± 2.01E-02 **5.75**	**0.91** ± 2.12E-02 2463.38	0.77 ± 4.29E-02 12.94	0.76 ± 1.57E-02 17.08
100	**0.91** ± 2.02E-02 **10.58**	**0.91** ± 2.07E-02 2688.62	0.78 ± 3.72E-02 12.94	0.76 ± 0.72E-02 17.15
Average score	**0.85** ± 3.60E-02 **3.89**	0.84 ± 3.86E-02 2081.09	0.78 ± 4.40E-02 12.94	0.76 ± 1.10E02 17.17

thus still needs to be learnt in practice, so we will concentrate on how to perform seed-KML in next section.

B. Seed-KML

In addition to class label, there exists another kind of supervised information such as pairwise constraint [3]. Pairwise constraint is more inherently general than class label since pairwise constraints can be directly derived from labeled data but not vice versa [19]. Thus, we will assume pairwise constraint as supervised information in the rest of this paper.

For notational convenience, we still denote $\mathcal{X}_l$ and $\mathcal{X}_u$ as supervised and unsupervised sample sets, respectively, that is, $\mathcal{X}_l = \{x_i | (x_i, \cdot) \in \mathcal{M} \cup \mathcal{C}\}$ consists of the samples with constraints, but no constraint on $\mathcal{X}_u$. Now, if we insert (5) into (1) and only impose the unit-length constraint to each sample in $\mathcal{X}_l$ (instead of $\mathcal{X}$), the PCP problem [in (1)] can be changed as

$$\begin{aligned} \min_{K_u} \quad & : Tr(LK) \\ \text{s.t.} \quad & : K = QK_{ll}Q^T \\ & \quad K_{ll}(i,i) = 1, \qquad \forall x_i \in \mathcal{X}_l \\ & \quad K_{ll}(i,j) = 1, \qquad \forall (x_i, x_j) \in \mathcal{M} \\ & \quad K_{ll}(i,j) = 0, \qquad \forall (x_i, x_j) \in \mathcal{C} \\ & \quad K_{ll} \succ 0. \end{aligned} \tag{7}$$

Equation (7) also requires that each pair of samples in $\mathcal{M}$ share the same embedding while those in $\mathcal{C}$ are orthogonal to each other, however, its fundamental differences from PCP lie in that: 1) the constraints "$K_{ii} = 1$" for $\forall x_i \in \mathcal{X}$ in PCP are just limited to the ones for $\forall x_i \in \mathcal{X}$ in (7); 2) the rank of the solution matrix for (7) is at most l, but unnecessarily so for PCP; and 3) (7) just needs to solve a seed-kernel matrix K_{ll} of size $l \times l$, while PCP needs to solve a full-kernel matrix of size $n \times n$, implying that solving (7) will be far more easier and efficient than solving PCP when $l \ll n$.

After plugging (6) into (7), we have

$$\begin{aligned} \min_{K_{ll}} \quad & : Tr\left(\left(L_{ll} - L_{lu}L_{uu}^{-1}L_{lu}^T\right)K_{ll}\right) \\ \text{s.t.} \quad & : K_{ll}(i,i) = 1, \qquad \forall x_i \in \mathcal{X}_l \\ & \quad K_{ll}(i,j) = 1, \qquad \forall (x_i, x_j) \in \mathcal{M} \\ & \quad K_{ll}(i,j) = 0, \qquad \forall (x_i, x_j) \in \mathcal{C} \\ & \quad K_{ll} \succeq 0 \end{aligned} \tag{8}$$

where, if all constraints in $\mathcal{M} \cup \mathcal{C}$ are consistent with each other, (8) will be clearly a convex programming and has a feasible solution $Y_l Y_l^T (Y_l \in \mathbb{R}^{l \times c}$, which is a class indicator matrix in terms of c classes, e.g., $Y_l(i,s) = 1$ if $x_i \in \mathcal{X}_l$ has a label $s \in \{1, 2, \ldots, c\}$ and $Y_l(i,s) = 0$ otherwise. Thus, optimizing (8) can yield one of the globally optimal solutions. However, the feasible set of (8) may be empty when $\mathcal{M} \cup \mathcal{C}$ contains some conflicting constraints such as $(x_1, x_2) \in \mathcal{M}$ and $(x_1, x_3) \in \mathcal{M}$ but $(x_2, x_3) \in \mathcal{C}$, for such a case, we have Proposition 1 below.

Proposition 1: The solution of (8) exists *if and only if* all given constraints are consistent with each other.

Proof: The proof of the sufficiency is trivial, so we just give a necessity proof by the counterproof. If there is a subset $\mathcal{U} \subseteq \mathcal{M} \cup \mathcal{C}$ consisting of those contradictive (or inconsistent) constraints, no mapping ϕ can map all the samples in $\{x|(x,\cdot) \in \mathcal{U}\}$ onto a unit hypersphere and make them to satisfy the constraints in $\mathcal{U}$ consistently, meaning that the feasible set of (8) will be empty and thus the solution of (8) does not exist. □

Remark 1: For Proposition 1, an inconsistent constraint set containing $(x_1, x_2) \in \mathcal{M}, (x_1, x_3) \in \mathcal{M}$, and $(x_2, x_3) \in \mathcal{C}$ means a contradiction [i.e., requiring $\phi(x_2) = \phi(x_1) = \phi(x_3)$ while $\phi(x_2) \neq \phi(x_3)$], so such a ϕ never exists. However, even though some users provide some inconsistent (noisy) constraints, we can still remove them in advance by an initialization step such as transitive closure method as suggested in [20]. For example, if $(x_1, x_2) \in \mathcal{M}$ and $(x_1, x_3) \in \mathcal{M}$, the set $\{x_1, x_2, x_3\}$ forms a must-link closure in terms of the transitive property of must-link constraints, and for such a situation $(x_2, x_3) \in \mathcal{C}$ can be identified and then removed due to its inconsistency to the closure.

For simplicity, we assume that all given constraints are consistent with each other throughout the rest part.

C. KP Algorithm

From the above results, we can illustrate the proposed KP algorithm in Table I.

Instead of directly returning K, we make a postprocessing step to diagonally normalize K to K^* and let K^* be the final target. The reason for doing so is that we likewise make the samples in $\mathcal{X}_u$ being unit length by this post-normalization since the samples in $\mathcal{X}_l$ have been constrained to being unit length. Undoubtedly, the so-obtained K^* will be difficult to guarantee its optimality w.r.t. the objective of (7). However, our empirical results show that adding such a post normalization can lead to better performances than not doing so.

The next thing we need to clarify is whether there is a significant difference between two solutions obtained respectively from KP and PCP since, after all, in both formulations there is seeming similarity to a great extent except for different utilizations *just* for unit-length constraints on the samples in $\mathcal{X}_u$. However, at present, we fail to prove this *theoretically*, instead, we *experimentally* provide an illustration of their essential difference (for more details, please refer to Appendix Part-A), which indicates that the rank, as one of matrix's algebraic invariants, of the full-kernel matrix resulted by KP is far less than that by PCP. Else, Table VII in Appendix Part-A also tells us that the rank of KPs kernel matrix gradually increases while that of PCPs gradually decreases as the number of constraints grows. Therefore, we can naturally draw a conclusion that KP indeed yields a significantly different solution from PCP, and particularly, KP actually leads to a far lower rank kernel matrix than PCP.

D. Time Complexity Compared with PCP

The main running time of KP algorithm is consumed by "Step 2." Let q be the number of the given pairwise constraints, then according to the analysis as offered in [21], solving the SDP problem in terms of seed-KML in "Step 2" needs the complexity $\mathcal{O}(l^3 + (l+q)^3)$ at least, and else, calculating L_{uu}^{-1} for (5) in "Step 3" needs the complexity $\mathcal{O}(u^3)$ generally, so KP needs the complexity $\mathcal{O}(l^3 + (l+q)^3 + u^3)$ in total. In contrast, PCP needs the complexity $\mathcal{O}(n^3 + (n+q)^3)$ at least to solve its SDP problem in (1). These analyzes again indicate that KP will be far more efficient than PCP when $l \ll n$.

Clearly, KPs running time will be also very high in the face of a large pairwise constraint set. Nevertheless, for such a case, we can adopt a resampling for the constraint set and then an ensemble technique as developed in our previous work [19] to step over this obstacle. More specifically, like the bootstrap, we first resample a large constraint set to generate a set of constraint subsets with fixed but smaller size on each of which KP is performed, then combine these individual KPs results to perform SS-KML for the large-constraint problem. However, this is out of scope of this paper.

E. Out-of-Sample Extension of KML

It is well known that an extension from in-sample case to out-of-sample case means to provide a timesaving procedure without the need of relearning for newly received data in future [17], [18]. To our best knowledge, some existing KML algorithms such as the ones in [8]–[10] do not address the out-of-sample extension issue, thus, an out-of-sample extension is desired.

Fortunately, following the previous KPs idea, we can naturally develop a conveniently realizable out-of-sample extension for KML. If we have learnt a kernel matrix K_{II} on in-sample data, then we can treat K_{II} as a "seed-kernel" matrix and propagate it into a full-kernel matrix on both in-sample and out-of-sample data. Specifically, denoting $\mathcal{X}_{\mathrm{I}}$ and $\mathcal{X}_{\mathrm{O}}$ as in-sample and out-of-sample data (sets) respectively, on both we can construct a full-graph Laplacian $L_{\mathrm{Full}} = \begin{pmatrix} L_{\mathrm{II}} & L_{\mathrm{IO}} \\ L_{\mathrm{OI}} & L_{\mathrm{OO}} \end{pmatrix}$ in which I and O are the index sets of $\mathcal{X}_{\mathrm{I}}$ and $\mathcal{X}_{\mathrm{O}}$, respectively. Further, denoting $Q = \begin{pmatrix} I_{\mathrm{I}} \\ -L_{\mathrm{OO}}^{-1} L_{\mathrm{IO}}^T \end{pmatrix}$, according to (5), we can directly propagate K_{II} into a full-kernel matrix as

$$K_{\mathrm{Full}} = Q \mathcal{K}_{\mathrm{II}} Q^T. \tag{9}$$

In particular, if out-of-sample data $\mathcal{X}_{\mathrm{O}}$ arrive in batch style, the kernel matrix can be directly calculated by

$$K_{\mathrm{OO}} = L_{\mathrm{OO}}^{-1} L_{\mathrm{OI}} K_{\mathrm{II}} L_{\mathrm{IO}} L_{\mathrm{OO}}^{-1}. \tag{10}$$

However, if $\mathcal{X}_{\mathrm{O}}$ arrive in sequence style (i.e., one sample at a time), Q needs to be updated sequentially. Let $Q_{(i)} \begin{pmatrix} I_{\mathrm{I}} \\ (-1/L_{\mathrm{ii}}) L_{\mathrm{Ii}}^T \end{pmatrix}$ be an update after the ith sample arrives, and an online style out-of-sample extension can be naturally gotten as given in Table II.

By contrast with the OE in Table II, we call the BE to deal with whole "block" data in (10). Clearly, the main running time of BE is consumed for computing L_{OO}^{-1} while it just turns into a reciprocal computation in OE, so the main running time of OE is consumed for updating $L^{(\mathrm{i})}$ at a time, and such an update can be fast computed for a sparse graph [22].

IV. Experiments

We compare KP with three state-of-the-art algorithms: PCP [10], improved-order [9], and Gaussian-field [27] on seven datasets depicted in Table III. Iris, Wine, Ionosphere, and Control are from UCI Machine Learning Repository (http://archive.ics.uci.edu/ml). USPS0123 and MNIST0123, respectively, come from USPS and MNIST datasets and both of them contain digit classes 0 to 3, USPS0123 consisting of the first 250 samples for each class while MNIST0123 the first 750 samples for each class. PIE-10-20 is drawn from CMU PIE dataset and consists of five frontal poses (C05, C07, C09, C27, and C29) of two individuals indexed as 10 and 20, which is sampled under different illuminations and expressions.

Since both Gaussian-field and improved-order belong to the spectral transformation method, their resulting kernel matrices are assumed to have the same eigenvectors as those of the graph Laplacian L while the eigenvalues are different, that is, if $L = \sum_i \lambda_i V_i V_i^T$, then $K = \sum_i \mu_i V_i V_i^T$ in which the μ_i's need to be tuned by maximizing the kernel alignment score in terms of the target matrix (T_{ij}) with $T_{ij} = +1$ if $label(x_i) = label(x_j)$ and -1 if $label(x_i) \neq label(x_j)$, otherwise $T_{ij} = 0$. It is necessary to note that both Gaussian-field and improved-order algorithms can be naturally generalized to the pairwise constraint setting as long as we redefine $T_{ij} = +1$ if $(x_i, x_j) \in \mathcal{M}$ and -1 if $(x_i, x_j) \in \mathcal{C}$, otherwise $T_{ij} = 0$.

As suggested in [10], we utilize the kernel K-means as a postprocessing for SS-KML algorithm evaluation, that is, the learnt full-kernel matrix is plugged into the kernel K-means to accomplish classification. The kernel K-means is an extension of the standard K-means algorithm, which maps data points from input space to a RKHS induced by the kernel (matrix) and minimizes the classification error in this RKHS. Also, as suggested in [24], we employ the NMI as a classification accuracy to compare the ground-truth class label vector Z with the class label vector Z' obtained by the kernel K-means. NMI value is calculated as

$$\mathrm{NMI}(Z, Z') = \frac{I(Z, Z')}{\sqrt{H(Z)H(Z')}}$$

where $I(Z, Z')$ is the mutual information measure between Z and Z', and $H(Z)$, $H(Z')$, denote the entropy measures of Z and Z', respectively. Note that $0 \leq \mathrm{NMI}(Z, Z') \leq 1$ and $\mathrm{NMI} = 1$ when $Z = Z'$, a larger NMI value means a better quality for the tested kernel (matrix).

A. Parameter Setting

For fairness of comparison, all the compared algorithms use a same k-nearest neighbor graph $\mathcal{G}$ to calculate graph Laplacian L, where K takes ten uniformly for each dataset as suggested in [9]. Each edge weight of $\mathcal{G}$ is computed by $w_{ij} = \exp\{-\|x_i - x_j\|^2/(2\sigma^2)\}$, in which σ is set as the averaged Euclidean distance from each data point to its ten nearest neighbors. In order to avoid unconnected component [25], we guarantee $\mathcal{G}$ to be connected by adding the edges generated by the maximum spanning tree algorithm.

To investigate how the evaluation measure varies according to the number of pairwise constraints, we vary the constraint number as suggested in [10], that is, r must-links for each class and r cannot-links for every two classes are produced randomly, such that $r \times (c + c(c-1)/2)$ pairwise constraints are generated in total for each r, where c is the ground-truth class number of the test dataset and r runs from 1 to 10 with the gap 1. For the K-means classification phase, we set K as the class number, i.e., $K = c$.

We use the standard solver CSDP 6.0.11 [26] to solve the SDP problems involved in KP and PCP and the quadratically constrained quadratic programming problem in the improved-order algorithm. For the Gaussian-field algorithm, its hyperparameter ϵ is computed through the "fminbnd" function of MATLAB as suggested in [9]. All the codes are implemented in MATLAB 7.1(R14), and all the experiments are carried out on a PC running Windows XP with Pentium(R)-Dual-CPU (1.6 GHz) and 1G RAM.

B. Experimental Results for In-Sample Case

In addition to NMI accuracy for effectiveness evaluation, we use the running time of SS-KML phase to assess efficiency. Both effectiveness and efficiency are tested for varying number of pairwise constraints. Over 20 trials, the average NMI accuracies of the kernel K-means on Iris, Wine, Ionosphere, and PIE-10-20 are illustrated in Fig. 2, and the average running times (in seconds) of the SS-KML phase in Fig. 3.

For a further quantitative examination, we also tabulate the comparisons of average NMI accuracies and running times respectively in Tables IV–VI for varying number of pairwise constraints on Control, USPS0123, and MNIST0123. Where there are two rows in each cell for each table, the upper row is the average NMI accuracy plus its standard deviation and the lower one is average running time in seconds, the highest NMI accuracies and the least running times are underlined and in bold. Additionally, we calculate the whole-average NMI accuracy, standard deviation, and run time over all the varying constraint numbers, and list them in the bottom row of each table.

From Figs. 2 and 3 and Tables IV–VI, we derive some observations and insights as follows.

1) KP gets almost identical NMI accuracies as PCP on all the datasets used here. However, it does not mean that KP and PCP share the same solution. In fact, as shown in Table VII of Appendix Part-A, KP yields a significantly different solution from PCP. This indicates that, despite KP and PCP getting almost identical NMI accuracies in our experiments, their solution matrices, as intermediate results of the whole classification learning procedure, are not necessarily close to each other mathematically.
2) In NMI accuracy, both KP and PCP significantly and consistently outperform both improved-order and Gaussian-field almost on all the datasets. Meanwhile, as the number of constraints grows, the improvements of KP and PCP are more significant than those of both improved-order and Gaussian-field. These results partially demonstrate that, in contrast with the spectral transformation method, KP and PCP are more flexible and thus more effective to cater for relatively complicated data. For this point, we have provided a toy in

TABLE VI

COMPARISON OF AVERAGE CLASSIFICATION ACCURACIES (IN NMI) AND RUNNING TIMES (IN SECONDS) FOR VARYING NUMBER OF PAIRWISE CONSTRAINTS ON MNIST0123 (SIZE OF 3000). BEST RESULTS ARE UNDERLINED AND IN BOLD. (IN COLUMNS 2–5, EACH CELL HAS TWO ROWS: THE UPPER IS THE AVERAGE NMI ACCURACY PLUS ITS STANDARD DEVIATION AND THE LOWER IS THE AVERAGE RUNNING TIME). (*EACH RUNNING TIME OF PCP $>$ 10000 s ON MNIST0123)

# Constraints	KP	PCP*	Improved-order	Gaussian-field
10	0.75 ± 2.88E-02 **28.04**	—	**0.82** ± 8.03E-02 307.97	0.76 ± 3.85E-02 343.89
20	0.78 ± 5.85E-02 **27.44**	—	**0.84** ± 8.27E-02 307.87	0.75 ± 0.04E-02 343.74
30	0.80 ± 6.72E-02 **28.23**	—	**0.84** ± 8.59E-02 307.86	0.75 ± 0.03E-02 343.05
40	**0.85** ± 5.69E-02 **28.38**	—	**0.85** ± 8.47E-02 307.85	0.75 ± 0.04E-02 343.77
50	**0.86** ± 3.37E-02 **29.31**	—	0.85 ± 8.55E-02 307.86	0.75 ± 0.03E-02 343.54
60	**0.90** ± 1.77E-02 **29.98**	—	0.86 ± 7.97E-02 307.86	0.75 ± 0.04E-02 344.22
70	**0.91** ± 2.61E-02 **31.11**	—	0.82 ± 8.57E-02 307.86	0.76 ± 3.84E-02 343.98
80	**0.91** ± 1.29E-02 **31.74**	—	0.84 ± 8.25E-02 307.87	0.75 ± 0.04E-02 343.84
90	**0.91** ± 1.53E-02 **35.42**	—	0.84 ± 8.85E-02 307.87	0.75 ± 0.04E-02 344.28
100	**0.92** ± 0.56E-02 **33.87**	—	0.86 ± 8.08E-02 307.86	0.75 ± 0.04E-02 344.02
Average score	**0.86** ± 3.23E-02 **30.35**	—	0.84 ± 8.36E-02 307.87	0.75 ± 0.80E-02 343.83

TABLE VII

COMPARISON OF THE RANKS OF TWO KERNEL MATRICES RESULTED BY KP AND PCP, RESPECTIVELY, FOR VARYING NUMBER OF CONSTRAINTS ON WINE AND PIE-10-20

Wine (size of 178)	# Constraints		**6**	**12**	**18**	**24**	**30**	**36**	**42**	**48**	**54**	**60**
	Rank	K_{KP}	9	16	23	29	35	39	45	48	51	55
		K_{PCP}	147	144	142	138	135	133	129	128	125	125
PIE-10-20 (size of 351)	# Constraints		**3**	**6**	**9**	**12**	**15**	**18**	**21**	**24**	**27**	**30**
	Rank	K_{KP}	4	8	12	15	19	23	26	29	31	35
		K_{PCP}	338	336	334	332	330	328	326	324	322	320

Appendix Part-B to illustrate and clarify why improved-order and Gaussian-field are sometimes ineffective.

3) More importantly, KP always gains significantly higher efficiency than PCP, e.g., on MNIST0123, PCP executes quite slowly so that its single run almost takes 3 h in our experimental environment, but KP only takes about 30 s on average, which is an over 300-times speedup. Such a higher efficiency of KP is mainly due to the fact that it just needs to solve a far smaller sized SDP problem than PCP when the number of constrained samples is far less than that of the whole samples.
4) Besides significantly higher accuracy, KP also has a higher efficiency than both improved-order and Gaussian-field under the settings of a relatively small number of constraints. Nevertheless, as the number of pairwise constraints grows, the running time of KP is higher than those of improved-order and Gaussian-field. This is because the more the number of constraints, the more the time needed for KP to solve the SDP problem in (8).

In sum, KP gets a comparable effectiveness as PCP and a comparable efficiency as improved-order and Gaussian-field. In addition, it is worth noting that, on the same datasets used here, Li *et al.* [10] have demonstrated that in effectiveness PCP outperforms two related algorithms, i.e., spectral learning (SL) [27] and semi-supervised kernel K-means (SSKK) [28].

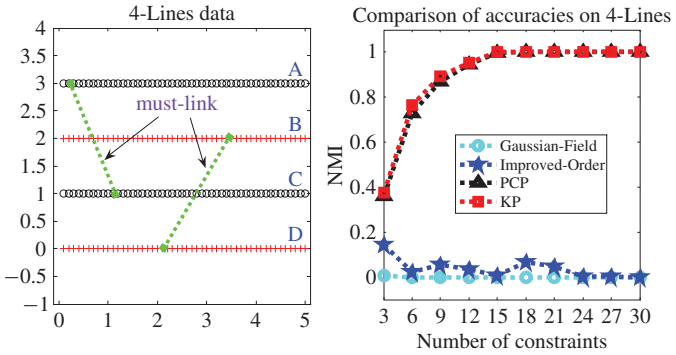

Fig. 5. Illustration to show that when the same-class data are distributed in multiple clusters, their leading eigenvectors derived from graph Laplacian will be unrepresentative and thus uninformative. Left: Four-lines data. Right: Comparison of average classification accuracies (in NMI) by four algorithms on four-lines data.

Thereby, we can expect that KP will also outperform SL and SSKK in effectiveness on these datasets.

C. Experimental Results for Out-of-Sample Extension

For the effectiveness evaluation to out-of-sample extensions developed in Section III-E, we test both BE and OE on PIE-10-20, Control, USPS0123, and MNIST0123. In order to generate out-of-sample and in-sample data, we randomly split a dataset into two halves in each trial, one half as out-of-sample data, and the other half as in-sample data. In particular, we assume that the pairwise constraints are only available on in-sample data. Over 20 trials, the average NMI accuracies are shown in Fig. 4, from which we can derive some observations and insights as follows.

1) When the number of pairwise constraints is small, the in-sample accuracies are higher than the out-of-sample accuracies on PIE-10-20. The reason is partially due to the fact that pairwise constraints, as supervised information, are *directly* injected into the in-sample data while *indirectly* propagated into the out-of-sample data. Intuitively, there would be more benefits from directly imposing constraints than indirectly doing so. However, contrarily, the out-of-sample accuracies are higher than the corresponding in-sample ones on both USPS0123 and MNIST0123, which is due to the fact that not only in-sample data but also out-of-sample ones are used to estimate the structure of out-of-sample graph (that is to say, according to BE or OE algorithm, we first construct the in-sample graph using in-sample data, and then update it into an out-of-sample graph when out-of-sample data are added), which, intuitively, should yield a better fit to the out-of-sample data and thus lead to the higher out-of-sample accuracies.
2) The accuracies of BE (or OE) are gradually closer to those of in-sample on all the four datasets as the number of constraints grows, which partially implies that, when sufficient in-sample constraints are provided, their constraint information can also be propagated to out-of-sample data well and thus the accuracies on out-of-sample and in-sample data approach to consistency.
3) The two accuracies of BE and OE are correspondingly close to each other on all four datasets, which demonstrates that the batch and online out-of-sample extensions are effective and consistent with each other.

As an intuition, it seems that BE should be better than OE because it takes advantage of all paired similarity on out-of-sample data points while OE does not. However, our experiments seem counterintuitive to some extent, so an in-depth investigation for such a violation is needed in future.

V. Conclusion

Despite the considerable research that has been carried out on SS-KML, a comprehensive performance in both effectiveness and efficiency is still far from satisfactory currently [11]. So, in this paper, we developed the KP algorithm to narrow such a gap. Though two typical kinds of partially supervised information, i.e., class label and pairwise constraint, are mostly concerned in practice, we mainly investigated KP's behavior under the pairwise constraint setting since pairwise constraint is inherently more general than class label [19].

Our main contributions include the following. 1) By KP, the procedure of SS-KML is roughly separated into, first, to learn a seed-kernel matrix through solving a small-sized SDP problem, and then to propagate the learnt seed-kernel matrix into a large-sized full-kernel matrix as the final target. 2) Following the idea in KP, two conveniently realizable out-of-sample extensions (i.e., BE and OE) are naturally developed for KML.

The experimental results demonstrated that the proposed KP can give a better comprehensive performance in both effectiveness and efficiency compared to the other three state-of-the-art algorithms, and the two developed out-of-sample extensions following the KP's idea also show their effectiveness in our experiments.

Finally, how to enhance the robustness in seed-kernel learning phase and why BE does not always outperform OE will be our next undertaking.

Appendix
Part-A: Experimental Evaluation for the Difference Between KP and PCP

For the two formulations of KP and PCP, there is a seeming similarity to great extent except for different utilizations *just* for unit-length constraints on the samples in $\mathcal{X}_u$, that is, PCP plugs the constraint "$K(i,i) = 1$" for each unconstrained point into its SDP optimization objective, whereas KP does not but instead takes a diagonal normalization as a postprocessing step into consideration. Now, a reader would naturally raise a question, Will KP yield a (almost) same solution as PCP? Its answer should be NO! However, currently we fail to prove it *theoretically*, thus instead, we *experimentally* provide an empirical evidence for their essential difference.

As is well known, the rank is one of matrix's algebraic invariants and is equal to the dimension number of the space spanned by the columns (or rows) of a matrix, implying that two completely identical matrices must have the same rank. Thus and so, we adopt the rank to identify and evaluate

the difference between the two full-kernel matrices K_{KP} and KPCP obtained by KP and PCP, respectively.

Thus, following the same settings as in Section IV, we compute the ranks of K_{KP} and KPCP for varying number of pairwise constraints on Wine and PIE-10-20, respectively, and tabulate the 20 trial's average ranks (rounded to the nearest integer) in Table VII, from which, we can derive the two main observations as follows.

1) The ranks of Kkp are far less than those of KPCP correspondingly, which sufficiently supports our previous conclusion that KP will lead to a far lower rank compared to PCP.
2) As an interesting result, the rank of Kkp gradually increases while that of KPCP gradually decreases as the constraint number grows.

From these two observations, we can draw a conclusion that KP indeed has a different solution from PCP. This implies that despite KP and PCP yielding almost identical classification accuracies (in NMI) in our experiments, their solution matrices, as intermediate results of the whole learning procedure, are not necessarily close to each other mathematically.

Part-B: Illustration for why the Improved-Order and Gaussian-Field Hardly Benefits from Additional Constraints

It is worthwhile to note that a SS algorithm is unnecessary to be improved as the amount of supervised information increases. For example, as shown in [10], both SL [27] and SSKK [28] algorithms are hardly improved in performance as the number of pairwise constraints grows. Below, we will give an explanation for why the "weak" eigenvectors are indeed probable to produce such a no-improvement behavior for spectral transformation such as improved-order and Gaussian-field algorithms.

In fact, Sinha and Belkin [12] have stated "it turns out that when the data has clustered, that is, when the high density regions are sufficiently separated by low density valleys, each high density area corresponds to a unique representative eigenvector of graph Laplacian." Therefore, according to this statement, when the same-class data are distributed in multiple clusters, their corresponding eigenvectors derived from graph Laplacian will fail to conform to the ground-truth class relationship, implying that these eigenvectors will be uninformative and thus can be coined as "weak."

In order to understand this viewpoint easily, we give an illustration of two-class problem in Fig. 5. Specifically, in the left-subplot of the figure, class 1 consists of the data points in lines A and C, while class 2 consists of those in lines B and D, such that the data points are unconnected relatively to class distributions. In such a situation, the four leading eigenvectors corresponding to these four lines are unrepresentative and thus uninformative since they fail to conform to the ground-truth class relationship, i.e., line A (B) has the same class as line C (D). As a result, these four uninformative eigenvectors will be still weak even though they are optimally combined according to optimization criterion as suggested in [9]. On the contrary, both KP and PCP can expectedly connect or merge line A (B) with line C (D) together in the feature space due to the imposed must-link constraints between them.

In the right-subplot of Fig. 5, the classification accuracies of kernel K-means demonstrate our intuition, that is, two algorithms of improved-order and Gaussian-field not only present poor NMI accuracies but also are hardly improved as the number of constraints grows, but in contrast, KP and PCP get better NMI accuracies and their NMI accuracies are significantly and gradually improved as the number of constraints grows.

VI. Acknowledgment

The authors would like to thank the editor and the anonymous reviewers for their valuable comments, which significantly improved the quality of this paper. Also, they wish to thank Z.-G. Li, who kindly provided the PCP [10] code to us.

References

[1] X. J. Zhu, "Semi-supervised learning literature survey," Dept. Comput. Sci., Univ. Wisconsin, Madison, Tech. Rep. 1530, 2005.
[2] F. Aiolli, G. Da San Martino, M. Hagenbuchner, and A. Sperduti, "Learning nonsparse kernels by self-organizing maps for structured data," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1938–1949, Dec. 2009.
[3] K. Wagsaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained K-means clustering with background knowledge," in *Proc. 18th Int. Conf. Mach. Learn.*, Williamstown, MA, Jun. 2001, pp. 577–584.
[4] A. Smola and R. Kondor, "Kernels and regularization on graphs," in *Proc. 16th Annu. Conf. Comput. Learn. Theory 7th Kernel Workshop*, Washington D.C., Aug. 2003, pp. 144–158.
[5] T. Zhang and R. K. Ando, "Analysis of spectral kernel design based semi-supervised learning," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, pp. 1601–1608.
[6] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster kernels for semi-supervised learning," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, pp. 585–592.
[7] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete input spaces," in *Proc. 19th Int. Conf. Mach. Learn.*, Sydney, Australia, Jul. 2002, pp. 315–322.
[8] X. J. Zhu, J. Lafferty, and Z. Ghahramani, "Semi-supervised learning: From Gaussian fields to Gaussian processes," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-03-175, 2003.
[9] X. J. Zhu, J. Kandola, J. Lafferty, and Z. Ghahramani, "Graph kernels by spectral transforms," in *Semi-Supervised Learning*, O. Chapelle, B. Schölkopf, and A. Zien, Eds. Cambridge, MA: MIT Press, 2006, pp. 277–289.
[10] Z. Li, J. Liu, and X. Tang, "Pairwise constraint propagation by semi-definite programming for semi-supervised classification," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jun. 2008, pp. 576–583.
[11] P. Sun and X. Yao, "Sparse approximation through boosting for learning large scale kernel machines," *IEEE Trans. Neural Netw.*, vol. 21, no. 6, pp. 883–894, Jun. 2010.
[12] K. Sinha and M. Belkin, "Semi-supervised learning using sparse eigen-function bases," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Cambridge, MA: MIT Press, 2009, pp. 1687–1695.
[13] 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: MIT Press, 2006, pp. 193–215.
[14] X. J. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington D.C., Aug. 2003, pp. 912–919.
[15] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, Jun. 2006, pp. 985–992.

[16] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf, "Ranking on data manifold," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004, pp. 169–176.
[17] Y. Bengio, O. Delalleau, and N. L. Roux, "A discussion of semi-supervised learning and transductive," in *Semi-Supervised Learning*, O. Chapelle, B. Schölkopf, and A. Zien, Eds. Cambridge, MA: MIT Press, 2006, pp. 473–478.
[18] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet, "Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004, pp. 177–184.
[19] D. Zhang, S. Chen, Z.-H. Zhou, and Q. Yang, "Constraint projections for ensemble learning," in *Proc. 23rd AAAI Conf. Artif. Intell.*, Chicago, IL, Jul. 2008, pp. 758–763.
[20] W. Tang, H. Xiong, S. Zhong, and J. Wu, "Enhancing semi-supervised clustering: A feature projection perspective," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Jose, CA, Aug. 2007, pp. 707–716.
[21] K. Q. Weinberger, B. D. Packer, and L. K. Saul, "Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, Savannah, GA, Jan. 2005, pp. 381–388.
[22] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang, "Generalized manifold-ranking-based image retrieval," *IEEE Trans. Image Process.*, vol. 15, no. 10, pp. 3170–3177, Oct. 2006.
[23] M. Breitenbach and G. Grudic, "Clustering through ranking on manifolds," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, Aug. 2005, pp. 73–80.
[24] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, no. 3, pp. 583–617, Mar. 2003.
[25] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang, "Manifold-ranking based image retrieval," in *Proc. 12th ACM Int. Conf. Multimedia*, New York, Oct. 2004, pp. 9–16.
[26] B. Borchers, "CSDP, a C library for semidefinite programming," *Optim. Methods Softw.*, vol. 11, nos. 1–4, pp. 613–623, 1999.
[27] S. Kamvar, D. Klein, and C. Manning, "Spectral learning," in *Proc. 18th Int. Joint Conf. Artif. Intell.*, Acapulco, Mexico, Aug. 2003, pp. 561–566.
[28] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: A kernel approach," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, Aug. 2005, pp. 457–464.

Enliang Hu received the M.Sc. degree in mathematics from Yunnan Normal University, Kunming, China, in 2003, and the Ph.D. degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2010.

He is currently a Lecturer in the Department of Mathematics, Yunnan Normal University. His current research interests include machine learning, pattern recognition, and data mining.

Songchan Chen received the B.Sc. degree in mathematics from Hangzhou University, Hangzhou, China, in 1983, the M.Sc. degree in computer applications from Shanghai Jiaotong University, Shanghai, China, in 1985, and the Ph.D. degree in communication and information systems from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 1997.

He was an Assistant Lecturer at NUAA in January 1986. Since 1998, he has been with the Department of Computer Science and Engineering at NUAA as a full Professor. He has authored or co-authored over 130 scientific papers. His current research interests include pattern recognition, machine learning, and neural computing.

Daoqiang Zhang received the B.Sc. and Ph.D. degrees in computer science from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 1999 and 2004, respectively.

He was a Post-Doctoral Fellow in the Department of Computer Science and Technology at Nanjing University. He joined the Department of Computer Science and Engineering, NUAA, in 2004, as a Lecturer, and is currently an Associate Professor. He has published over 30 technical papers in refereed international journals or conference proceedings. His current research interests include machine learning, pattern recognition, data mining, and image processing.

Dr. Zhang is a member of the Chinese Association of Artificial Intelligence Machine Learning Society. He was nominated for the National Excellent Doctoral Dissertation Award of China in 2006, and won the Best Paper Award in the 9th Pacific Rim International Conference on Artificial Intelligence. He served as a program committee member for several international and national conferences.

Xuesong Yin received the Ph.D. degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2010.

He is currently an Associate Professor in the School of Information and Engineering, Zhejiang Radio and TV University, Hangzhou, China. His current research interests include machine learning, data mining, and image processing.