Principal Graph and Structure Learning Based on Reversed Graph Embedding

Qi Mao, Li Wang, Ivor W. Tsang, and Yijun Sun

Abstract—Many scientific datasets are of high dimension, and the analysis usually requires retaining the most important structures of data. Principal curve is a widely used approach for this purpose. However, many existing methods work only for data with structures that are mathematically formulated by curves, which is quite restrictive for real applications. A few methods can overcome the above problem, but they either require complicated human-made rules for a specific task with lack of adaption flexibility to different tasks, or cannot obtain explicit structures of data. To address these issues, we develop a novel principal graph and structure learning framework that captures the local information of the underlying graph structure based on reversed graph embedding. As showcases, models that can learn a spanning tree or a weighted undirected ℓ_1 graph are proposed, and a new learning algorithm is developed that learns a set of principal points and a graph structure from data, simultaneously. The new algorithm is simple with guaranteed convergence. We then extend the proposed framework to deal with large-scale data. Experimental results on various synthetic and six real world datasets show that the proposed method compares favorably with baselines and can uncover the underlying structure correctly.

Index Terms—Principal curve, principal graph, structure learning

1 Introduction

In many fields of science, one often encounters observations represented as high-dimensional vectors sampled from unknown distributions. It is sometimes difficult to directly analyze data in the original space, and is desirable to perform data dimensionality reduction or associate data with some structured objects for further exploratory analysis. The problem of uncovering an underlying structure from data has become more and more important in fields ranging from computer vision to computational biology. In this paper, we focus on principal graph and structure learning to uncover the underlying structure of data.

The classic approach for obtaining latent structure from data is principal curve [1], which was initially proposed as a nonlinear generalization of the first principal component line. Informally, a principal curve is an infinitely differentiable curve with a finite length that passes through the middle of data. Several principal curve approaches have been proposed by minimizing certain types of risk functions, including the quantization error [1], [2], [3], [4], [5] and the negative log-likelihood function [6], [7]. To overcome the overfitting issue, regularization is required. Kégl et al. [2], [8] bounded the total length of a principal curve, and proved that the principal curve with a bounded length always exists if the data distribution has a finite second moment. Similar results were obtained by bounding the turns of a principal curve [4]. The elastic maps and principal graph approach [9], [10] took the elastic energy of a membrane (and plate) to regularize the manifold learning problem. An alternative definition of a principal curve based on a mixture model was considered in [1], where the model parameters are learned through maximum likelihood estimation and the regularization is achieved using the smoothness of coordinate functions. Generative topographic mapping (GTM) [7] was proposed to maximize the posterior probability of the data which is generated by a low-dimensional discrete grid mapped into the original space and corrupted by additive Gaussian noise. GTM provides a principled alternative to the self-organizing map [11] for which it is impossible to define an optimality criterion [12].

Methods for learning a principal curve have been widely studied, but they cannot handle self-intersecting data. As stated in [13], the self-intersecting data structure (or principal graph) is a collection of smooth curves where these curves can intersect each other. A few methods can handle complex principal objects. Kégl and Krzyzak [14] extended their polygonal line method [2] for skeletonization of handwritten digits and fingerprint minutiae [15], where sophisticated domain-specific rules were required to generate the skeleton structure [14]. If the sophisticated rules are not available for certain tasks, this method is not directly applicable to these tasks. The principal graph grammar approach [16] extends the classical elastic maps and principal graph approach [10] to learn a graph by using graph grammar [17], [18]. A set of predefined graph grammar operations is applied to a given graph in all possible ways; an elastic functional over each permissible

- *Q. Mao is with the HERE Company, Chicago, IL 60606. E-mail: qimao.here@gmail.com.*
- *L. Wang is with the Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, IL 60607. E-mail: liwang8@uic.edu.*
- *I.W. Tsang is with the Centre for Artificial Intelligence, University of Technology Sydney, Ultimo, NSW 2007, Australia. E-mail: ivor.tsang@uts.edu.au.*
- *Y. Sun is with the Department of Microbiology and Immunology, The State University of New York at Buffalo, Buffalo, NY 14228. E-mail: yijunsun@buffalo.edu.*

Manuscript received 4 Dec. 2015; revised 4 Oct. 2016; accepted 18 Nov. 2016. Date of publication 4 Dec. 2016; date of current version 10 Oct. 2017. Recommended for acceptance by G. Chechik.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2016.2635657

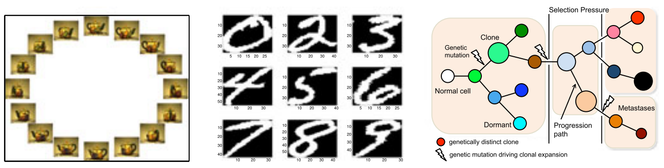
graph is minimized by the splitting algorithm; a transformed graph with the largest energy descent is found over all permissible graphs. Simple grammars worked well in some tasks as demonstrated in [5], [19], and the splitting algorithm has similar convergence guarantee as K-means. However, the convergence of the principal graph grammar approach by alternating between elastic graph construction and the splitting algorithm is not well established, and termination criterion based on the number of transformations is quite heuristic. In addition, the computation cost increases iteratively when the number of nodes in a graph increases. Simple grammars can alleviate the computational issue [5], [19], but they cannot form complicated principal objects, such as loops and disconnected components. Topological data analysis [20] is another related direction to extract simple graph representation of high dimensional datasets. An efficient algorithm [21] was proposed to construct principal objects from datasets represented by Reeb graphs [22], which were used to model skeleton graphs, but it requires a proximity graph (ϵ-neighborhood or k-nearest neighborhood) as an input. Another topological method called Mapper [23] was proposed to produce a multiresolution or multiscale image of the dataset by building a simplicial complex, and was able to achieve substantial simplifications, as well as preserve certain topological structures from the original dataset. It has been successfully applied to 3D object recognition [23] and breast cancer transcription data [24]. The topological structure constructed varies significantly if there are a small number of noise samples since the connectivity could be formed whenever two clusters have nonempty intersection. Additionally, a subspace constrained mean shift (SCMS) method [13], [25] was proposed that can obtain principal points for any given second-order differentiable density function, but it is still not trivial to transform a set of principal points to an explicit structure. In many real applications, an explicit structure uncovered from the given data is helpful for downstream analysis. This critical point will be illustrated in Sections 2 and 6 in detail with various real world datasets.

Another line of research relevant to this work is structure learning, which has had a great success in constructing or learning explicit structures from data. The graph structures that are commonly used in graph-based clustering and semi-supervised learning are the k-nearest neighbor graph and the ϵ-neighborhood graph [26]. Dramatic influences of these two graphs on clustering techniques have been studied in [27]. Since the ϵ-neighborhood graph could result in disconnected components or subgraphs in the dataset or even isolated singleton vertices, the b-matching method is applied to learn a better b-nearest neighbor graph via loopy belief propagation [28]. However, it is improper to use a fixed neighborhood size since the curvature of manifold and the density of data points may be different in different regions of the manifold [29]. In order to alleviate these issues, a method for simultaneous clustering and embedding of data lying in multiple manifolds [29] was proposed using ℓ_2 norm over the errors that measure the linear representation of every data point by using its neighborhood information. Similarly, ℓ_1 graph was learned for image analysis using ℓ_1 norm over the errors for enhancing the robustness of the learned graph [30]. Instead of learning directed graphs by using the above two methods, an integrated model for learning an undirected graph by imposing a sparsity penalty on a symmetric similarity matrix and a positive semi-definite constraint on the Laplacian matrix was proposed [31]. Most recently, probabilistic models [32], [33] for dimensionality reduction are proposed to learn general graphs in terms of kernels [34]. Although these methods have demonstrated their effectiveness on various problems, it is challenging to apply them to moderate-size data, let alone large-scale data, due to the high computational complexity. Moreover, they might yield suboptimal results by heuristically transforming a directed graph to an undirected graph for clustering and dimensionality reduction [29], [30].

This paper is an extension of our preliminary work [35], where as a showcase we have demonstrated the effectiveness of learning a minimum-cost spanning tree for datasets of tree structures in certain applications. We move forward to take into account more complicated structures that exist in real world datasets as discussed in Section 2. Moreover, we propose two strategies to specifically overcome the issue of high complexity of the proposed method for large-scale datasets. The main contributions of this paper are :

- We propose a novel regularized principal graph and structure learning framework that addresses the aforementioned limitations by learning a set of principal points and an explicit graph structure from data, simultaneously. Our work represents principal objects using an explicit graph that is learned in a systematic way with a guaranteed convergence, and the learning is practical for large-scale data.
- Our novel formulation called reversed graph embedding for the representation of a principal graph facilitates the learning of graph structures, generalizes several existing methods, and possesses many advantageous properties.
- In addition to spanning trees, a weighted undirected ℓ_1 graph is proposed for modeling various types of graph structures, including curves, bifurcations, self-intersections, loops, and multiple disconnected components. This facilitates the proposed learning framework for datasets with complicated graphs.
- We further propose two strategies to deal with large-scale data. One is to use side-information as a priori to reduce the complexity of graph learning, the other is to learn a representative graph over a set of landmarks. Both strategies can be simultaneously incorporated into our framework for efficient handling of large-scale data.
- Extensive experiments are conducted for unveiling the latent graph structures on a variety of datasets, including various synthetic datasets and six real world applications, consisting of various structures: hierarchy of facial expression images, progression path of breast cancer in microarray data, rotation circle of teapot images, smoothing skeleton of optical characters, and similarity patterns of digits on two large-scale handwritten digits databases.

The rest of the paper is organized as follows. We first illustrate the learning problem by using various real world datasets in Section 2. In Section 3, we present three key building blocks for the representation of principal graphs and structure



(a) 360° rotation (b) complicated structures (c) cancer progression

Fig. 1. Real world problems exhibiting a variety of graph structures. (a) 360 degree rotation of teapot images forming a circle. (b) Optical character templates for different digits containing loops, bifurcations, and multiple disconnected components. (c) Branching architecture of cancer evolution (modified from [40]). Selective pressures allow some tumor clones to expand while others become extinct or remain dormant.

learning. Based on these key building blocks, we propose a new regularized principal graph and structure learning framework in Section 4. In Section 5, we incorporate two strategies into the proposed framework to deal with large-scale datasets. Extensive experiments are conducted in Section 6. Finally, we conclude this work in Section 7.

2 Motivation of Structure Learning

In many fields of science, experimental data resides in a high-dimensional space. However, the distance between two data points may not directly reflect the distance measured on the intrinsic structure of data. Hence, it is desirable to uncover the intrinsic structure of data before conducting further analysis.

It has been demonstrated that many high-dimensional datasets generated from real-world problems contain special structures embedded in their intrinsic dimensional spaces. One example is a collection of teapot images viewed from different angles [36]. Each image contains 76×101 RGB pixels, so the pixel space has a dimensionality of 23,028, but the intrinsic structure has only one degree of freedom: the angle of rotation. As shown in Fig. 1a, distances computed by following an intrinsic circle are more meaningful than distances computed in the original space. Another example is given in [37] where it is demonstrated that a collection of facial expression images (217×308 RGB pixels) contains a two-layer hierarchical structure (Fig. 3b in [37]). The images from three facial expressions of one subject are grouped together to form the first layer, while all images from the three subjects form the second layer. In other words, images of one subject should be distant from images of all other subjects, but images from three expressions of the same subject should be close. More complicated structures including loops, bifurcations, and multiple disconnected components as shown in Fig. 1b can be observed in optical character templates for identifying a smoothing skeleton of each character [13], [14]. Other examples with specific intrinsic structures are also discussed in [38], [39].

We are particularly interested in studying human cancer, a dynamic disease that develops over an extended time period. Once initiated from a normal cell, the advance to malignancy can to some extent be considered a Darwinian process—a multistep evolutionary process—that responds to selective pressure [40]. The disease progresses through a series of clonal expansions that result in tumor persistence and growth, and ultimately the ability to invade surrounding tissues and metastasize to distant organs. As shown in Fig. 1c, the evolution trajectories inherent to cancer progression are complex and branching [40]. Due to the obvious necessity for timely treatment, it is not typically feasible to collect time series data to study human cancer progression [28]. However, as massive molecular profile data from excised tumor tissues (static samples) accumulates, it becomes possible to design integrative computation analyses that can approximate disease progression and provide insights into the molecular mechanisms of cancer. We have previously shown that it is indeed possible to derive evolutionary trajectories from static molecular data, and that breast cancer progression can be represented by a high-dimensional manifold with multiple branches [41].

These concrete examples convince us that many datasets in a high-dimensional space can be represented by certain complicated structures in low dimension. Existing methods exploit these structures of data implicitly either by learning a parametric mapping function or approximating a manifold via local geometric information of the observed data. However, these methods do not aim to directly learn an explicit form of a structure in a low-dimensional space. In contrast, our proposed method is designed for this purpose to express the complicated structure of data by an explicit graph representation, which will be discussed in the following sections.

3 Principal Graph Representation

Let $\mathcal{X} = \mathbb{R}^D$ be the input space and $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathcal{X}$ be a set of observed data points. We consider learning a projection function $h_\mathcal{G} \in \mathcal{H}$ such that latent points $\mathbb{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_K\} \subset \mathcal{Y} = \mathbb{R}^d$ in d-dimensional space can faithfully represent $\mathbb{X}$, where $\mathcal{H} = \{h_\mathcal{G} : \mathcal{Y} \to \mathcal{X}\}$ is a set of functions defined on a graph $\mathcal{G}$, and function $h_\mathcal{G}$ maps $\mathbf{y}_k$ to $h_\mathcal{G}(\mathbf{y}_k) \in \mathcal{X}, \forall k = 1, \ldots, K$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ be a weighted undirected graph, where $\mathcal{V} = \{1, \ldots, K\}$ is a set of vertices, $\mathcal{E}$ is a set of edges, and $\mathbf{W} \in \mathbb{R}^{K \times K}$ is an edge weight matrix with the (k, k')th element denoted by $w_{k,k'}$ as the weight of edge $(k, k'), \forall k, k'$. Suppose that every vertex k has a one-to-one correspondence with latent point $\mathbf{y}_k \in \mathcal{Y}$, which resides on a manifold with an intrinsic dimension d. Next, we introduce three key components that form the building blocks of the proposed framework.

3.1 Reversed Graph Embedding

The most important component of the proposed framework is to model the relationship between graph $\mathcal{G}$ and latent points $\mathbb{Y}$. Given $\mathcal{G}$, weight $w_{k,k'}$ measures the similarity (or connection indicator) between two latent points $\mathbf{y}_k$ and $\mathbf{y}_{k'}$ in an intrinsic space $\mathcal{Y}$. Since the corresponding latent points $\{\mathbf{y}_1, \ldots, \mathbf{y}_K\}$ are generally unknown in advance, we coin a new graph-based representation, namely Reversed Graph Embedding, in order to bridge the connection between graph structure $\mathcal{G}$ and corresponding data points in the input space. Specifically, our intuition is that if any two latent variables $\mathbf{y}_k$ and $\mathbf{y}_{k'}$ are neighbors on $\mathcal{G}$ with high similarity $w_{k,k'}$, two corresponding points $h_\mathcal{G}(\mathbf{y}_k)$ and $h_\mathcal{G}(\mathbf{y}_{k'})$ in the input space should be close to one another. To capture this intuition, we propose to minimize the following objective function as an explicit representation of principal graphs, given by

$$\Omega(\mathbb{Y}, h_\mathcal{G}, \mathbf{W}) = \sum_{(k,k') \in \mathcal{E}} w_{k,k'} ||h_\mathcal{G}(\mathbf{y}_k) - h_\mathcal{G}(\mathbf{y}_{k'})||_2^2, \tag{1}$$

where $h_{\mathcal{G}} \in \mathcal{H}$ and $\mathbb{Y} \in \mathcal{Y}$ are variables to be optimized for a given $\mathcal{G}$. At this moment, we cannot obtain interesting solutions by directly solving problem (1) with respect to $\{\mathbb{Y}, h_{\mathcal{G}}\}$ due to multiple trivial solutions leading to zero objective value. However, objective function (1) possesses many interesting properties and can be used as a graph-based regularization for unveiling the intrinsic structure of the given data, which is detailed below.

3.1.1 Relationship to Laplacian Eigenmap

The objective function (1) can be interpreted from a reverse perspective of the well-known Laplacian eigenmap [26]. Denote $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{d\times N}$. The Laplacian eigenmap solves the following optimization problem:

$$\min_{\mathbb{Y}} \sum_{i=1}^{N} \sum_{j=1}^{N} v_{i,j} ||\mathbf{y}_i - \mathbf{y}_j||_2^2 : \mathbf{Y}\mathbf{D}\mathbf{Y}^T = \mathbf{I},$$

where the similarity $v_{i,j}$ between $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed in the input space $\mathcal{X}$ in order to capture the locality information of the manifold modeled by neighbors of the input data points, e.g., $v_{i,j} = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||_2^2}{2\sigma^2})$ using the heat kernel with bandwidth parameter σ^2 if i is the neighbor of j and j is also the neighbor of i, and $v_{i,j} = 0$ otherwise, and $\mathbf{D} = \mathrm{diag}([\sum_{i=1}^{N} v_{i,j}]) \in \mathbb{R}^{N\times N}$ is a diagonal matrix. The Euclidean distance between $\mathbf{y}_i$ and $\mathbf{y}_j$, i.e., $||\mathbf{y}_i - \mathbf{y}_j||_2$, are computed in the latent space $\mathcal{Y}$. Consequently, two multiplication terms of the objective functions of reverse graph embedding and Laplacian eigenmap are calculated in a different space: (i) weights $w_{i,j}$ and $v_{i,j}$ are computed in $\mathcal{Y}$ and $\mathcal{X}$, respectively; (ii) distances $||h_{\mathcal{G}}(\mathbf{y}_k) - h_{\mathcal{G}}(\mathbf{y}_{k'})||_2$ and $||\mathbf{y}_i - \mathbf{y}_j||_2^2$ are computed in $\mathcal{X}$ and $\mathcal{Y}$, respectively.

Our formulation can directly model the intrinsic structure of data, while Laplacian eigenmap approximates the intrinsic structure by using neighborhood information of observed data points. In other words, our proposed reversed graph embedding representation facilitates the learning of a graph structure from data. The weight $w_{k,k'}$ encodes the similarity or connectivity between vertices k and k' on $\mathcal{G}$. For example, if $w_{k,k'} = 0$, it means that edge (k, k') is absent from $\mathcal{E}$. In most cases, a dataset is given, but graph $\mathcal{G}$ is unknown. In these cases, it is necessary to automatically learn $\mathcal{G}$ from data. The objective function (1) is linear with respect to weight matrix $\mathbf{W}$. This linearity property facilitates the learning of the graph structure. We will discuss different representations of graphs based on this linearity property in Section 3.3.

Another important difference is that the number of the latent variables $\mathbb{Y}$ is not necessarily equal to the number of input data points $\mathbb{X}$, i.e., $K \leq N$. This is useful to obtain a representative graph over a set of landmark points by compressing a large amount of input data points, and the representative graph can still faithfully represent the whole dataset. We will explore this property for large-scale principal graph learning in Section 5.2.

3.1.2 Harmonic Function of a General Graph

We have discussed the properties of the reversed graph embedding by treating variables $\mathbf{y}_k$ and $h_{\mathcal{G}}$ as a single integrated variable $h_{\mathcal{G}}(\mathbf{y}_k), \forall k$. Actually, the optimal function $h_{\mathcal{G}} \in \mathcal{H}$ obtained by solving (1) is related to harmonic or pluriharmonic functions. This can be further illustrated by the following observations. Let $\mathcal{N}_k$ be the neighbors of point $\mathbf{y}_k, \forall k$. For any given $\mathbf{y}_{k'}$ problem (1) can be rewritten as $\min_{h_{\mathcal{G}}(\mathbf{y}_k)} \sum_{k'\in\mathcal{N}_k} w_{k,k'} ||h_{\mathcal{G}}(\mathbf{y}_k) - h_{\mathcal{G}}(\mathbf{y}_{k'})||^2$, which has an analytic solution by fixing the rest of variables $\{h_{\mathcal{G}}(\mathbf{y}_k)\}_{k\neq k'}$ given by

$$h_{\mathcal{G}}(\mathbf{y}_k) = \frac{1}{\sum_{k'\in\mathcal{N}_k} w_{k,k'}} \sum_{k'\in\mathcal{N}_k} w_{k,k'} h_{\mathcal{G}}(\mathbf{y}_{k'}). \tag{2}$$

If equality (2) holds for all k, function $h_{\mathcal{G}}$ is a harmoinc function on $\mathcal{G}$ since its value in each nonterminal vertex is the mean of the values in the closest neighbors of this vertex [5]. The plurihamonic graphs defined in [5] impose penalty only on a subset of k-stars as,

$$\left|\left| h_{\mathcal{G}}(\mathbf{y}_m) - \frac{1}{\sum_{j\in\mathcal{N}_m} w_{m,j}} \sum_{j\in\mathcal{N}_m} w_{m,j} h_{\mathcal{G}}(\mathbf{y}_j) \right|\right|^2, \tag{3}$$

where $|\mathcal{N}_m| = k, \forall m$. In contrast, our formulation (1) flexibly incorporates any neighborhood structure existing in $\mathcal{G}$. The connection of $h_{\mathcal{G}}$ to harmonic or pluriharmonic functions enriches the target function, which has been previously discussed in [5].

3.1.3 Extension of Principal Curves to General Graphs

Equation (1) promotes the generalization of various existing methods. It is worth noting that the quantity $\Omega(\mathbb{Y}, h_{\mathcal{G}}, \mathbf{W})$ can be considered as the length of a principal graph in terms of the square of ℓ_2 norm. In the case where $\mathcal{G}$ is a linear chain structure, $\Omega(\mathbb{Y}, h_{\mathcal{G}}, \mathbf{W})$ is same as the length of a polygonal line defined in [2]. However, general graphs or trees are more flexible than principal curves since the graph structure allows self-intersection. For principal graph learning, elastic map [5] also defines a penalty based on a given graph. However, based on the above discussion, it is difficult to solve problem (3) with respect to both the function $h_{\mathcal{G}}$ and the graph weights $\mathbf{W}$ within the elastic-maps framework. In contrast, the proposed reversed graph embedding leads to a simple and efficient algorithm to learn a principal tree or a weighted undirected ℓ_1 graph with guaranteed convergence. This will be clarified in Section 4.

3.2 Data Grouping

The second important component of the proposed framework is to measure the fitness of latent variables $\mathbb{Y}$ to a given data $\mathbb{X}$ in terms of a given graph $\mathcal{G}$ and projection function $h_{\mathcal{G}}$. As the number of latent variables is not necessarily equal to the number of input data points, we assume that projected point $h_{\mathcal{G}}(\mathbf{y}_k)$ is a centroid of the kth cluster of $\mathbb{X}$ so that input data points with high similarity form a cluster. The empirical quantization error [3] is widely used as the fitting criterion to be minimized for the optimal cluster centroids, and it is also frequently employed in principal curve learning methods [1], [2], [3], [4], [5], given by

$$\ell(\mathbb{X}, \mathbb{Y}, h_{\mathcal{G}}) = \sum_{i=1}^{N} \min_{k=1,\ldots,K} ||\mathbf{x}_i - h_{\mathcal{G}}(\mathbf{y}_k)||_2^2. \tag{4}$$

Based on Equation (4), we have the following loss functions.

Data Reconstruction Error. If $K = N$, we can reformulate the Equation (4) by reordering $\mathbb{Y}$ as

$$\ell_N(\mathbb{X}, \mathbb{Y}, h_{\mathcal{G}}) = \sum_{i=1}^{N} ||\mathbf{x}_i - h_{\mathcal{G}}(\mathbf{y}_i)||_2^2. \tag{5}$$

This formulation can be interpreted as the negative log-likelihood of data $\mathbb{X}$ that is i.i.d drawn from a multivariate normal distribution with mean $h_{\mathcal{G}}(\mathbf{y}_i)$ and covariance matrix $\frac{1}{2}\mathbf{I}$.

K-means. If $K < N$, we introduce an indicator matrix $\Pi \in \{0,1\}^{N\times K}$ with the (i,k)th element $\pi_{i,k} = 1$ if $\mathbf{x}_i$ is assigned to the kth cluster with centroid $h_{\mathcal{G}}(\mathbf{y}_k)$, and $\pi_{i,k} = 0$ otherwise. Consequently, we have the following equivalent optimization problem:

$$\ell_{\Pi}(\mathbb{X}, \mathbb{Y}, h_{\mathcal{G}}) = \sum_{i=1}^{N}\sum_{k=1}^{K} \pi_{i,k}||\mathbf{x}_i - h_{\mathcal{G}}(\mathbf{y}_k)||_2^2, \tag{6}$$

where $\sum_{k=1}^{K}\pi_{i,k} = 1$ and $\pi_{i,k} \in \{0,1\}, \forall i = 1,\ldots,N$. This is the same as the optimization problem of the K-means method that minimizes the objective function (4).

Generalized Empirical Quantization Error. If $K \leq N$, a right stochastic matrix $\mathbf{P} \in [0,1]^{N\times K}$ with each row summing to one is introduced, that is, $\sum_{k=1}^{K} p_{i,k} = 1, \forall i = 1,\ldots,N$. This variant is equivalent to the above representation of indicator matrix Π if an integer solution $\mathbf{P}$ is obtained. When K is relatively large, K-means that minimizes (6) might generate empty clusters. To avoid this issue, we introduce the soft assignment strategy by adding negative entropy regularization as

$$\ell_{\mathbf{P}}(\mathbb{X}, \mathbb{Y}, h_{\mathcal{G}}) = \sum_{i=1}^{N}\sum_{k=1}^{K} p_{i,k}\Big[||\mathbf{x}_i - h_{\mathcal{G}}(\mathbf{y}_k)||_2^2 + \sigma \log p_{i,k}\Big], \tag{7}$$

where $\sigma > 0$ is a regularization parameter.

In this paper, we use $\ell_{\mathbf{P}}(\mathbb{X}, \mathbb{Y}, h_{\mathcal{G}})$ since it is a generalized version of the other two cases and also has a close relationship with the mean shift clustering method [42], Gaussian mixture model, and the K-means method. Below, we discuss these interesting properties in detail. The proofs of Propositions 1 and 2 are given in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2016.2635657.

Proposition 1. *The mean shift clustering [42] is equivalent to minimizing objective function (7) with respect to a left stochastic matrix $\mathbf{P} \in [0,1]^{N\times K}$ with each column summing to one.*

Proposition 2. *Minimizing objective function (7) with respect to a right stochastic matrix $\mathbf{P} \in [0,1]^{N\times K}$ with each row summing to one can be interpreted as the Gaussian mixture model with uniform weights.*

Corollary 1. *We have the following relationships among three loss functions:*

1) *If $\sigma \rightarrow 0$, minimizing (7) is equivalent to minimizing (6) so that $\mathbf{P} = \Pi$.*
2) *If $\sigma \rightarrow 0$ and $N = K$, minimizing (7) is equivalent to minimizing (5).*

According to the aforementioned results, we can briefly summarize the merits of function (7). First, empty clusters will never be created according to Proposition 2 for any $K \leq N$. Second, the loss function takes the density of input data $\mathbb{X}$ into account. In other words, the centroids obtained by minimizing the loss function should reside in the high density region of the data. Third, the loss function makes the learning of graph structures computationally feasible for large-scale data in the case of $K < N$, which will be discussed in Section 5.

3.3 Latent Sparse Graph Learning

The third important component of the proposed framework is to learn a latent graph from data. To achieve this goal, we investigate two special graph structures that can be learned from data by formulating the learning problems as linear programming. One is a tree structure, represented as a minimum-cost spanning tree, the other is a weighted undirected graph that is assumed to be sparse. Let $\mathbb{Z} = \{\mathbf{z}_1,\ldots,\mathbf{z}_K\} \subset \mathbb{R}^D$ be a dataset. Our task is to construct a weighted undirected graph $\mathcal{G} = (\mathcal{V},\mathcal{E},\mathbf{W})$ with a cost $\phi_{k,k'}$ associated with edge $(k,k') \in \mathcal{E}, \forall k,k'$ by optimizing similarity matrix $\mathbf{W} \in \mathbb{R}^{K\times K}$ based on the assumption of the specific graph structure.

3.3.1 Minimum Cost Spanning Tree

Let $\mathcal{T} = (\mathcal{V},\mathcal{E}_{\mathcal{T}})$ be a tree with the minimum total cost and $\mathcal{E}_{\mathcal{T}}$ be the edges forming a tree. In order to represent and learn a tree structure, we consider $\mathbf{W}$ as binary indicator matrix where $w_{k,k'} = 1$ if $(k,k') \in \mathcal{E}_{\mathcal{T}}$, and $w_{k,k'} = 0$ otherwise. The integer linear programming formulation of a minimum spanning tree (MST) can be written as, $\min_{\mathbf{W}\in\mathcal{W}_0} \sum_{k,k'} w_{k,k'}\phi_{k,k'}$, where $\mathcal{W}_0 = \{\mathbf{W} \in \{0,1\}^{N\times N}\} \cap \mathcal{W}'$ and $\mathcal{W}' = \{\mathbf{W} = \mathbf{W}^T\} \cap \{\frac{1}{2}\sum_{k,k'} w_{k,k'} = |\mathcal{V}| - 1, w_{k,k} = 0, \forall k\} \cap \{\frac{1}{2}\sum_{k\in\mathcal{S},k'\in\mathcal{S}} w_{k,k'} \leq |\mathcal{S}| - 1, \forall \mathcal{S} \subseteq \mathcal{V}\}$. The first constraint of $\mathcal{W}'$ enforces the symmetric connection of undirected graph, e.g., $w_{k,k'} = w_{k',k}$. The second constraint states that the spanning tree only contains $|\mathcal{V}| - 1$ edges. The third constraint imposes the acyclicity and connectivity properties of a tree. It is difficult to solve an integer programming problem directly. Thus, we resort to a relaxed problem by letting $w_{k,k'} \geq 0$, that is,

$$\min_{\mathbf{W}\in\mathcal{W}_{\mathcal{T}}} \sum_{k=1}^{K}\sum_{k'=1}^{K} w_{k,k'}\phi_{k,k'}, \tag{8}$$

where the set of linear constraints is given by $\mathcal{W}_{\mathcal{T}} = \{\mathbf{W} \geq 0\} \cap \mathcal{W}'$. Problem (8) can be readily solved by the Kruskal's algorithm [43], [44].

3.3.2 Weighted Undirected ℓ_1 Graph

A complete graph with imposed sparsity constraints over edge weights $\mathbf{W}$ called ℓ_1 graph is considered for learning a sparse graph. An ℓ_1 graph is based on the assumption that each data point has a small neighborhood in which the minimum number of points that span a low-dimensional affine subspace passing through that point. The affine function is defined as $\mathbf{z}_k = \sum_{k'=1}^{K} w_{k',k}\mathbf{z}_{k'}, \forall k$, where $\mathbf{z}_k$ is the vector of targeted point, $\mathbf{W}$ is the edge weight matrix, and $\mathbb{Z}$ is a set of k data points in the small neighborhood. In practice, there may exist noise in certain elements of $\mathbf{z}_k$, and a natural idea is to estimate the edge weights by tolerating these errors. We introduce error $\xi_k \in \mathbb{R}^D$ to each linear equation and the equality constraints are formulated as $\mathbf{z}_k = \sum_{k'=1,k'\neq k}^K w_{k',k}\mathbf{z}_{k'} + \xi_k, \forall k$. In order to learn a similarity measurement for an undirected graph, we impose nonnegative and symmetric constraints on $\mathbf{W}$, i.e., $\{\mathbf{W} \geq 0, \mathbf{W} = \mathbf{W}^T\}$. Generally, a sparse solution is more robust and facilitates the consequent identification of the test sample $\mathbf{z}_k$. Following the recent results on the sparse representation problem, the convex ℓ_1-norm minimization can effectively recover the sparse solution [29], [30]. Let $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_K] \in \mathbb{R}^{D\times K}$. We propose to learn $\mathbf{W}$ by solving the following linear programming problem

$$\begin{aligned} &\min_{\mathbf{W},\{\xi_k\}_{k=1}^K} \sum_{k=1}^K \sum_{k'=1}^K w_{k,k'}\phi_{k,k'} + \lambda \sum_{k=1}^K ||\xi_k||_1 \\ &\text{s.t. } \mathbf{z}_k = \sum_{k'=1,k'\neq k}^K w_{k',k}\mathbf{z}_{k'} + \xi_k, \forall k, \mathbf{W} \geq 0, \mathbf{W}{=}\mathbf{W}^T, \text{diag}(\mathbf{W}){=}0, \end{aligned} \tag{9}$$

where $\lambda > 0$ is a parameter for weighting all errors and $||\xi_k||_1 = \sum_{m=1}^D |\xi_k^m|$ with $\xi_k = [\xi_k^1, \ldots, \xi_k^D]^T$. According to (9), two points $\mathbf{z}_k$ and $\mathbf{z}_{k'}$ are similar (small $\phi_{k,k'}$) if edge weight $w_{k,k'}$ is a large positive value. In other words, the coefficient of linear combination is coincident with the edge weight on a graph. Problem (9) is different from methods [29], [30] that learn directed ℓ_1 graphs, and different from the method [31] that learns undirected graphs using the probabilistic model with Gaussian Markov random fields. Moreover, data points $\mathbb{Z}$ can be different from the data points that are used to compute costs.

3.3.3 Generalized Graph Structure Learning

For the ease of representation, we use a unified formulation $g(\mathbf{W}, \mathbb{Z}, \mathbf{\Phi})$ for learning a similarity graph, where $\mathbb{Z}$ is a given dataset, $\mathbf{W}$ is the similarity matrix of a graph over the given dataset, and $\mathbf{\Phi}$ is a cost matrix with the (k, k')th entry as $\phi_{k,k'}$. The feasible space of $\mathbf{W}$ is denoted as $\mathcal{W}$. Specifically, we solve the following generalized graph structure learning problem such that any graph learning problem that can be formulated as linear programming with constraints represented as $\mathcal{W}$ can be used in the following proposed framework, given by

$$\min_{\mathbf{W}\in\mathcal{W}} g(\mathbf{W}, \mathbb{Z}, \mathbf{\Phi}). \tag{10}$$

It is easy to identify the correspondences of problem (10) to problems (8) and (9).

4 Regularized Principal Graph and Structure Learning

By combining the three building blocks discussed in Section 3, we are ready to propose a unified framework for learning a principal graph. We use the alternate convex search algorithm to solve the proposed formulations by simultaneously learning a set of principal points and an undirected graph with guaranteed convergence.

4.1 Proposed Formulation

Given an input dataset $\mathbb{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, we aim to uncover the underlying graph structure that generates $\mathbb{X}$.

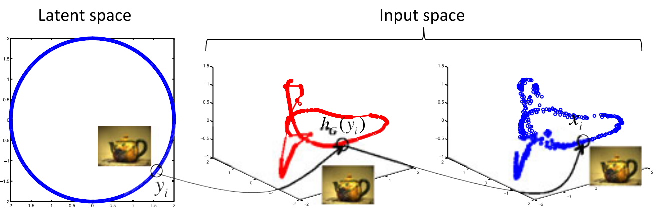


Fig. 2. A cartoon illustrating the proposed formulation on the teapot images. Each circle marker represents one teapot image. Our assumption of the data generation process is that a graph $\mathcal{G}$ exists in the latent space (e.g., a rotation circle in the left subplot), and each image $\mathbf{y}_i$ is then mapped to point $h_{\mathcal{G}}(\mathbf{y}_i)$ in the input space by maintaining the graph structure through the reversed graph embedding, and finally image $\mathbf{x}_i$ is observed conditioned on $h_{\mathcal{G}}(\mathbf{y}_i)$ according to certain noise model.

Since the input data may be drawn from a noise model, it is improper to learn $\mathcal{G}$ directly from $\mathbb{X}$. To unveil the hidden structure, we assume that (i) the underlying graph can be recovered from the learning model represented in the form of (10), e.g., the graph is an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with specific structures; (ii) graph $\mathcal{G}$ satisfies the reversed graph embedding assumption where the cost is defined by $\phi_{i,j} = ||h_{\mathcal{G}}(\mathbf{y}_i) - h_{\mathcal{G}}(\mathbf{y}_j)||_2^2$, that is, if two vertices i and j are close on $\mathcal{G}$ with high similarity $w_{i,j}$, points $h_{\mathcal{G}}(\mathbf{y}_i)$ and $h_{\mathcal{G}}(\mathbf{y}_j)$ should be close to each other; (iii) data points $\{h_{\mathcal{G}}(\mathbf{y}_i)\}_{i=1}^N$ are considered as the denoised points of $\mathbb{X}$ so that the movement of $\mathbf{x}_i$ from the denoised point $h_{\mathcal{G}}(\mathbf{y}_i)$ can be measured by data grouping properly by letting $K = N$.

For clarity, we illustrate the relationships of various variables in Fig. 2. Based on the above assumptions, we propose the novel regularized principal graph and structure learning framework by simultaneously optimizing the graph weight matrix $\mathbf{W}$, latent variables $\mathbb{Y}$, projection function $h_{\mathcal{G}} \in \mathcal{H}$, and soft-assignment matrix $\mathbf{P}$ as

$$\begin{aligned} &\min_{h_{\mathcal{G}}\in\mathcal{H},\mathbb{Y}} \min_{\mathbf{W}\in\mathcal{W},\mathbf{P}\in\mathcal{P}_r} g(\mathbf{W}, \mathbb{X}, \mathbf{\Phi}) + \gamma \ell_{\mathbf{P}}(\mathbb{X}, \mathbb{Y}, h_{\mathcal{G}}) \\ &\qquad \text{s.t. } \phi_{i,j} = ||h_{\mathcal{G}}(\mathbf{y}_i) - h_{\mathcal{G}}(\mathbf{y}_j)||_2^2, \forall i, j, \end{aligned} \tag{11}$$

where $\gamma > 0$ is a tradeoff parameter between the graph-based regularizer and the total noise of the input data. It is worth noting that (10) is the key component of (11) for learning a graph given $h_{\mathcal{G}}$ and $\mathbb{Y}$, and also contains the reversed graph embedding objective (1) as a crucially important element for two special graph learning (8) and (9). The second term in the objective function (11) is the loss function (7).

In the following of this section, we propose a simple method for solving problem (11) and present its convergence and complexity analysis.

4.2 Optimization Algorithm

We focus on learning the underlying graph structure from the data. Instead of learning $h_{\mathcal{G}} \in \mathcal{H}$ and $\mathbb{Y}$ separately,[1] we optimize $h_{\mathcal{G}}(\mathbf{y}_i)$ as a joint variable $\mathbf{c}_i, \forall i$. Let $\mathbb{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_N\}$ where $\mathbf{c}_i = h_{\mathcal{G}}(\mathbf{y}_i)$, and $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_N] \in \mathbb{R}^{D\times N}$. In the case of weighted undirected ℓ_1 graph learning, the optimization problem (11) with respect to variables $\{\mathbb{C}, \mathbf{W}, \mathbf{P}\}$ are reformulated as

1. To learn $h_{\mathcal{G}}$, we should define $\mathcal{H}$ properly, e.g., a linear projection function might be possible as studied in our previous work [45].

$$\begin{aligned}\min_{\mathbb{C},\mathbf{W},\mathbf{P}} \; & \sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}||\mathbf{c}_i-\mathbf{c}_j||_2^2 + \lambda\sum_{i=1}^{N}\Big|\Big|\mathbf{x}_i-\sum_{j\neq i} w_{j,i}\mathbf{x}_j\Big|\Big|_1 \\ & + \gamma\sum_{i=1}^{N}\sum_{j=1}^{N} p_{i,j}\Big[||\mathbf{x}_i-\mathbf{c}_j||_2^2+\sigma\log p_{i,j}\Big] \\ \text{s.t. } & \mathbf{W}\geq 0, \mathbf{W}=\mathbf{W}^T, \mathrm{diag}(\mathbf{W})=0, \sum_{j=1}^{N} p_{i,j}=1, p_{i,j}\geq 0, \forall i,j.\end{aligned} \tag{12}$$

In the case of learning an MST, the feasible space $\mathcal{W}$ is $\mathcal{W}_{\mathcal{T}}$, and the objective function is similar to (12) by removing the second term from the objective function. For simplicity, we do not explicitly show the formulation of learning a spanning tree.

Problem (12) is a biconvex optimization problem [46]: for fixed $\mathbf{P}$ and $\mathbf{W}$, optimizing $\mathbb{C}$ is a convex optimization problem; for fixed $\mathbb{C}$, optimizing $\mathbf{P}$ and $\mathbf{W}$ is also convex. We propose to solve problem (12) by using the alternate convex search, a minimization method to solve a biconvex problem where the variable set can be divided into disjoint blocks [46]. The blocks of variables defined by convex subproblems are solved cyclically by optimizing the variables of one block while fixing the variables of all other blocks. In this way, each convex subproblem can be solved efficiently by using a convex minimization method. Below, we discuss each subproblem in details and the pseudo-code of the proposed method is given in Algorithm 1.

Algorithm 1. Principal Graph & Structure Learning

```
1: Input: Data $\mathbf{X}\in\mathbb{R}^{D\times N}$, $\lambda$, $\sigma$, and $\gamma$
2: Initialize $\mathbf{C}=\mathbf{X}$
3: repeat
4:     $\phi_{i,j}=||\mathbf{c}_i-\mathbf{c}_j||_2^2, \forall i,j$
5:     $p_{i,j}=\frac{\exp(-||\mathbf{x}_i-\mathbf{c}_j||^2/\sigma)}{\sum_{j=1}^{N}\exp(-||\mathbf{x}_i-\mathbf{c}_j||^2/\sigma)}, \forall i,j$
6:     Solve the following linear programming problem:
       - (14) for a weighted undirected $\ell_1$ graph
       - (15) for a spanning tree
7:     $\mathbf{C}=\mathbf{X}\mathbf{P}(2\gamma^{-1}\mathbf{L}+\mathbf{\Lambda})^{-1}$
8: until Convergence
```

4.2.1 *Fix $\mathbf{C}$ and Solve $\mathbf{P},\mathbf{W}$*

Given $\mathbf{C}$, problem (12) with respect to $\mathbf{P}$ and $\mathbf{W}$ can be solved independently. Due to the decoupled variables of $\mathbf{P}$ in rows, we can solve each row independently. According to Proposition 2, we have the analytic solution of $\mathbf{P}$ given by

$$p_{i,j}=\frac{\exp\Big(-||\mathbf{x}_i-\mathbf{c}_j||^2/\sigma\Big)}{\sum_{j=1}^{N}\exp\Big(-||\mathbf{x}_i-\mathbf{c}_j||^2/\sigma\Big)}, \forall i,j. \tag{13}$$

For problem (12) with respect to $\mathbf{W}$, we have to solve the following optimization problem:

$$\begin{aligned}\min_{\mathbf{W}} \; & trace(\mathbf{\Phi}^T\mathbf{W})+\lambda\sum_{i=1}^{N}\Big|\Big|\mathbf{x}_i-\sum_{j\neq i}w_{j,i}\mathbf{x}_j\Big|\Big|_1 \\ \text{s.t. } & \mathbf{W}\geq 0, \mathbf{W}=\mathbf{W}^T, \mathrm{diag}(\mathbf{W})=0,\end{aligned} \tag{14}$$

where $\mathbf{\Phi}\in\mathbb{R}^{N\times N}$ with the (i,j)th element $\phi_{i,j}=||\mathbf{c}_i-\mathbf{c}_j||_2^2$. For problem (11) to learn a spanning tree structure, we solve the following problem:

$$\min\nolimits_{\mathbf{W}\in\mathcal{W}_{\mathcal{T}}} trace(\mathbf{\Phi}^T\mathbf{W}). \tag{15}$$

As discussed in Section 3.3, Problem (15) can be solved by the Kruskal's algorithm. Problem (14) is a linear programming problem, which can be solved efficiently by off-the-shelf linear programming solver for small- or moderate-sized datasets, such as Mosek [47].

4.2.2 *Fix $\mathbf{P},\mathbf{W}$ and Solve $\mathbf{C}$*

Given $\mathbf{P}$ and $\mathbf{W}$, problem (12) with respect to $\mathbf{C}$ can be rewritten as

$$\min_{\mathbf{C}} \; trace\big(\mathbf{C}(2\mathbf{L}+\gamma\mathbf{\Lambda})\mathbf{C}^T-2\gamma\mathbf{C}^T\mathbf{X}\mathbf{P}\big), \tag{16}$$

where the graph Laplacian matrix $\mathbf{L}=\mathrm{diag}(\mathbf{W}\mathbf{1})-\mathbf{W}$ and diagonal matrix $\mathbf{\Lambda}=\mathrm{diag}(\mathbf{1}^T\mathbf{P})$. Problem (16) is an unconstrained quadratic programming. We have the analytic solution given by

$$\mathbf{C}=\mathbf{X}\mathbf{P}(2\gamma^{-1}\mathbf{L}+\mathbf{\Lambda})^{-1}, \tag{17}$$

where the inverse of matrix $2\gamma^{-1}\mathbf{L}+\mathbf{\Lambda}$ always exists since $\mathbf{L}$ is a positive semi-definite matrix and diagonal matrix $\mathbf{\Lambda}$ is always positive definite according to (13).

4.3 Convergence and Complexity Analysis

Since problem (12) is non-convex, there may exist many local optimal solutions. Following the initialization strategy of the mean shift clustering, we initialize $\mathbf{C}$ to be the original input data as shown in Algorithm 1. The theoretical convergence analysis of Algorithm 1 is presented in the following theorem and its proof is given in the supplementary material, available online.

Theorem 1. *Let $\{\mathbf{W}_\ell,\mathbf{C}_\ell,\mathbf{P}_\ell\}$ be the solution of problem (12) in the ℓth iteration, and $\varrho_\ell=\varrho(\mathbf{W}_\ell,\mathbf{C}_\ell,\mathbf{P}_\ell)$ be the corresponding objective function value, then we have:*

(i) $\{\varrho_\ell\}$ is monotonically decreasing;
(ii) Sequences $\{\mathbf{W}_\ell,\mathbf{C}_\ell,\mathbf{P}_\ell\}$ and $\{\varrho_\ell\}$ converge.

According to Theorem 1, we define the stopping criterion of Algorithm 1 in terms of the relative increase in the function value compared to the last iteration, and fix it as 10^{-5} in all experiments. The empirical convergence results are shown in Section 6.1.

The computational complexity of Algorithm 1 for learning a tree structure is determined by three individual parts. The first part is the complexity of running Kruskal's algorithm to construct a minimum spanning tree. It requires $\mathcal{O}(N^2D)$ for computing a fully connected graph and $\mathcal{O}(N^2\log N)$ for finding a spanning tree. The second part is dominated by computing the soft assignments of samples, which has a complexity of $\mathcal{O}(N^2D)$. The third part is dominated by the inverse of a matrix of size $N\times N$ that takes $\mathcal{O}(N^3)$ operations and matrix multiplication that takes $\mathcal{O}(DN^2)$ operations. Thus, the total complexity for each iteration is $\mathcal{O}(N^3+DN^2)$. For learning an undirected weighted

ℓ_1 graph by solving problem (14), the only difference is the complexity of linear programming, which can be solved efficiently by Mosek [47]. In Section 5, we will extend the proposed algorithm for handling large-scale data.

5 Regularized Principal Graph and Structure Learning on Large-Scale Data

In order to reduce the high computational complexity of Algorithm 1 for large-scale data, we propose to incorporate two strategies into the proposed model for fast learning by using landmarks and side information.

5.1 Graph Learning with Side Information

Instance-level constraints are useful to express a priori knowledge about which instances should or should not be grouped together, which have been successfully applied to many learning tasks, such as clustering [48], metric learning [49], and kernel learning [50]. In the area of clustering, the most prevalent form of advice are conjunctions of pairwise instance level-constraints of the form must-link (ML) and cannot-link (CL) which state that pairs of instances should be in the same or different clusters respectively [48]. Given a set of points to cluster and a set of constraints, the aim of clustering with constraints is to use the constraints to improve the clustering results.

We consider certain structure-specific side information for guidance in learning the similarity matrix $\mathbf{W}$ and computational reduction instead of optimizing the full matrix. For this purpose, we take into account the CL constraints. Let $\mathcal{N}_i$ be a set of data points which might be linked to data point $\mathbf{x}_i$. On the contrary, data points that are not in $\mathcal{N}_i$ will belong to CL set. By incorporating the CL side information into the proposed framework, we can derive the following optimization problem for learning the ℓ_1 graph representation given by

$$\begin{aligned}\min_{\mathbf{W}} \; & \sum_{i=1}^N \sum_{j \in \mathcal{N}_i} w_{i,j}\phi_{i,j} + \lambda \sum_{i=1}^N \Big|\Big| \mathbf{x}_i - \sum_{j\in\mathcal{N}_i} w_{j,i}\mathbf{x}_j \Big|\Big|_1 \\ \text{s.t. } & w_{i,j} \geq 0, w_{i,j} = w_{j,i}, \forall i,j \\ & w_{i,j} = 0, \forall i, j \notin \mathcal{N}_i,\end{aligned} \tag{18}$$

which can be equivalently transformed into linear programming optimization problem and can be solved accurately for large-scale data by using the off-the-shelf toolbox.

5.2 Landmark-Based Regularized Principal Graph

In order to handle large-scale data, we extend the proposed regularized principal graph learning framework by using a landmark-based technique. Landmarks have been widely used in clustering methods to deal with large-scale data [51], [52]. Random landmark selection method is widely used for spectral clustering [51], [52]. It is well known that selecting landmarks can further improve clustering performance with a certain adaptive strategy, such as K-means [53], a variety of fixed and adaptive sampling schemes, and a family of ensemble-based sampling algorithms [54].

Our regularized principal graph and structure learning framework can inherently take landmarks into account through the data grouping. Taking the weighted undirected ℓ_1 graph based framework as an example, the optimization problem by considering landmarks is given by

$$\begin{aligned}\min_{\mathbb{C},\mathbf{W},\mathbf{P}} \; & \sum_{k=1}^K \sum_{k'=1}^K w_{k,k'} ||\mathbf{c}_k - \mathbf{c}_{k'}||_2^2 + \lambda \sum_{k=1}^K \Big|\Big| \mathbf{z}_k - \sum_{k' \neq K} w_{k',k}\mathbf{z}_{k'} \Big|\Big|_1 \\ & + \gamma \sum_{i=1}^N \sum_{k=1}^K p_{i,k}\Big[||\mathbf{x}_i - \mathbf{c}_k||_2^2 + \sigma \log p_{i,k} \Big] \\ \text{s.t. } & \mathbf{W} \geq 0, \mathbf{W} = \mathbf{W}^T, \operatorname{diag}(\mathbf{W}) = 0 \\ & \sum_{k=1}^K p_{i,k} = 1, p_{i,k} \geq 0, \forall i = 1,\ldots,N, k = 1,\ldots,K,\end{aligned} \tag{19}$$

where $\mathbb{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ is a set of landmarks of size K. Moreover, our proposed methods can automatically adjust centroids of landmarks during the optimization procedure. Due to the non-convexity of above objective function, we have to properly initialize these landmarks. As shown in the literature [52], [53], K-means can generally obtain better clustering performance for spectral clustering methods. In this paper, we follow this idea and use centroids obtained from K-means to initialize $\mathbf{Z}$. The pseudo-code is shown in Algorithm 2. By following the same analysis procedure as Section 4.3, the computational complexity of Algorithm 2 is $\mathcal{O}(K^3 + DKN + DK^2)$ which is significantly smaller than $\mathcal{O}(N^3 + DN^2)$ of Algorithm 1 if $K \ll N$. Hence, Algorithm 2 is practical for large-scale data with a large N.

Algorithm 2. Large-Scale Principal Graph & Structure Learning

```
1: Input: Data $\mathbf{X} \in \mathbb{R}^{D\times N}$, $\lambda$, $\sigma$, $\gamma$ and $K \ll N$
2: Obtain $\mathbf{Z} \in \mathbb{R}^{D\times K}$ by calling the $K$-means method
3: Initialize $\mathbf{C} = \mathbf{Z}$
4: repeat
5:     $\phi_{k,k'} = ||\mathbf{c}_k - \mathbf{c}_{k'}||_2^2, \forall k, k'$
6:     $p_{i,k} = \frac{\exp(-||\mathbf{x}_i - \mathbf{c}_k||^2/\sigma)}{\sum_{k=1}^K \exp(-||\mathbf{x}_i - \mathbf{c}_k||^2/\sigma)}, \forall i, k$
7:     Solve the following linear programming problem:
       - (14) for a weighted undirected $\ell_1$ graph
       - (15) for a spanning tree
8:     $\mathbf{C} = \mathbf{X}\mathbf{P}(2\gamma^{-1}\mathbf{L} + \mathbf{\Lambda})^{-1}$
9: until Convergence
```

6 Experiments

We conduct extensive experiments to evaluate the proposed models for learning either spanning trees or weighted undirected ℓ_1 graphs on various synthetic datasets and six real world applications. The source code of the proposed methods is freely available from http://liwang8.people.uic.edu/TPAMI2016-PGSL.html.

6.1 Convergence and Sensitivity Analysis

We perform a convergence analysis of Algorithm 1 using a synthetic tree dataset. Fig. 3 shows the empirical convergence results obtained by learning two different graph structures as well as their intermediate results. The top panel of Fig. 3 shows the empirical convergence results by illustrating the relative difference of the objective function

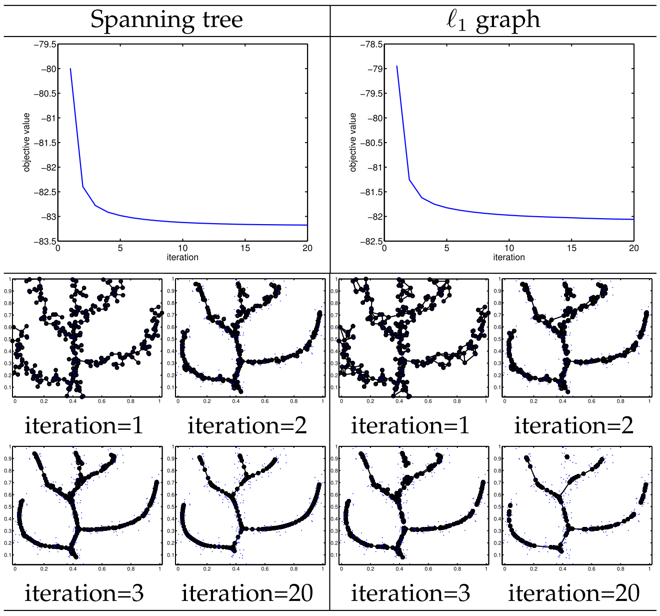


Fig. 3. Convergence analyses and intermediate results of proposed methods on the Tree dataset by learning two different graph structures.

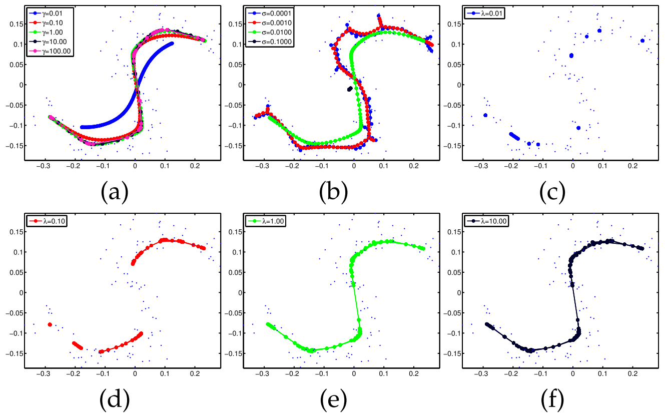


Fig. 4. Results of parameter sensitivity analysis of Algorithm 1 by learning ℓ_1 graph performed on the distorted S-shape dataset by varying different parameters: (a) varying γ; (b) varying σ; (c)-(f) varying λ.

value in terms of the number of iterations in Algorithm 1. We observe that the proposed algorithm converges in less than 20 iterations. This is consistent with the result of the theoretical analysis in Section 4.3. From the bottom panel of Fig. 3, we can see that when Algorithm 1 continues with more iterations, the tree structure becomes smoother. This empirically verifies the intuition of the reverse graph embedding.

We then perform a parameter sensitivity analysis by using the distorted S-shape dataset to demonstrate how the algorithm behaves with respect to parameters σ, γ and λ. As λ only appears in Algorithm 1 for learning an ℓ_1 graph, we investigate the sensitivity of parameters in terms of learning an ℓ_1 graph. Results obtained from learning an ℓ_1 graph can be similarly applied to Algorithm 1 for learning a spanning tree. Fig. 4 shows the ℓ_1 graphs constructed by varying one parameter and fixing the others. We have the following observations. First, according to Fig. 4a, it is clear that the smaller γ is, the shorter the length of the S-shape curve is. In contrast, the larger γ is, the more faithful the curve passes through the middle of the data. Therefore, γ is an important parameter that controls the trade-off between the curve fitting error and the length of a principal graph. Second, as shown in Fig. 4b, the graph structure becomes smoother when increasing σ. This is also stated in Propositions 1 and 2 that explore the relationships between the proposed formulation and the mean-shift clustering. In other words, a large σ encourages more data points to merge together. The data movement represented by variable matrix $\mathbf{C}$ in Algorithm 1 is also restricted on a graph and the length of the graph should be minimized so that data points in $\mathbf{C}$ will be smoother so as to reach the goal. The choice of bandwidth parameter in Algorithm 1 is then similar to that in the mean-shift clustering. As suggested in [55], it is best to explore a range of bandwidths instead of estimating one from data by minimizing a loss function or more heuristic rules since clustering is by nature exploratory, so is the structure learned in this paper. Third, the larger λ is, the more edges the learned graph contains as shown in Figs. 4c, 4d, 4e, and 4f). This is the reason why the noise term becomes large and less data points are required to represent the current data point if λ is a large value.

Due to the exploratory nature of learning a graph structure from data and the lack of evaluation metrics for graph structure learning in an unsupervised setting, the automation of setting parameters by using the provided data only is generally difficult. However, the aforementioned parameter sensitivity analysis suggests a pragmatic way to tune parameters for learning an expected graph structure. The recommended procedure is given as follows: first, setting σ to be a small value, and then tuning γ from large value to small so as to fit the data properly, finally increasing σ to obtain a reasonable structure. In order to learn an ℓ_1 graph from data, the initial graph structure should be constructed by solving problem (14) using a proper λ such that the weight matrix $\mathbf{W}$ captures the key structure of the input data. And then, the above recommended procedure for tuning σ and γ is applied. For tuning σ and γ automatically from data, we have studied in our previous work [35] by using leave-one-out-maximum likelihood criterion described in [13] and gap statistics [56] for σ and γ respectively. As discussed in [55], it is natural to tune parameter σ in a different range. In the following experiments, we will take the above tuning strategy for setting all parameters in a large range, and all parameters used will be reported for the reproducibility of the experiments.

6.2 Synthetic Data

We evaluate the performance of Algorithm 1 for learning either a spanning tree or an ℓ_1 graph by comparing with three baseline methods, including the polygonal line method [2], SCMS [13], and Mapper [23], on six synthetic datasets. Among them, the first two datasets are also used in [2], [13]. In the case of learning an ℓ_1 graph, we incorporate neighbors of each data point as the side information, i.e., data points not in the neighborhood of a data point are considered as a set of cannot-links for the data point. In this paper, we take nn-nearest neighbor as a showcase for side-information. The experiments are conducted in two settings. The first setting is to evaluate the five methods for learning curves, while the second setting is to investigate other structures of the datasets, including loops, self-intersection and disconnected

TABLE 1
Parameters Used in the Proposed Algorithm for Synthetic Datasets

Dataset	N	σ	γ	λ	nn	SCMS σ
Distorted S-shape	100	0.01	0.5	1.0	5	0.07
Spiral	200	0.01	0.5	1.0	10	0.1
Circle	100	0.1	0.5	1.0	10	0.4
Two-moon	200	0.01	3	0.1	5	0.1
Tree	300	0.01	10	1.0	5	0.05
Three-clusters	300	0.01	0.5	0.1	5	0.1

The proposed method for learning an ℓ_1 graph and a spanning tree share the same set of parameters, while SCMS are tuned in fine-grid to achieve reasonable principal points.

components. For all experiments in this section, the parameters of compared methods are set as shown in Table 1. In addition, Mapper is carefully tuned by trying different filter functions and covers where Gaussian kernel function and balanced 1-d cover are chosen for the best visualization results. We take single linkage clustering as the default clustering method [23], and the rest of parameters are shown in the bottom-left side of resulting figures. It is worth noting that the tuning of parameters in the proposed method is tuning easier than tuning σ in SCMS, which needs finer tuning in order to achieve comparable results.

The first two columns of Fig. 5 show the results from the principal curve learning setting. We have the following observations: 1) The proposed methods can obtain smoother curves than the other tested methods. 2) The polygonal line method fails on the Spiral data. We also see that SCMS cannot obtain a curve structure because many projected points do not have ordering information, and some points are scattered as shown in the Spiral data. This leaves a non-trivial problem to learn the underlying structure by using SCMS. Although Mapper outputs clusters and their connectivity structure for the overlapping of any two clusters, there are several observations: i) many isolated clusters are formed due to the noisy data points; ii) the structure of data points in each cluster is undefined; and iii) the global structure of each dataset is not well captured by comparing to the underlying graph structure. Our proposed method does not have these problems.

The last four columns of Fig. 5 show the results obtained from the datasets containing loop, self-intersection, and disconnected components. The polygonal line method fails on all four datasets due to the improper assumption of principal curves. We should point out that the spanning tree is naturally used for learning a tree structure, so the proposed method for learning a tree structure outperforms the one proposed for learning an ℓ_1 graph as shown in the results

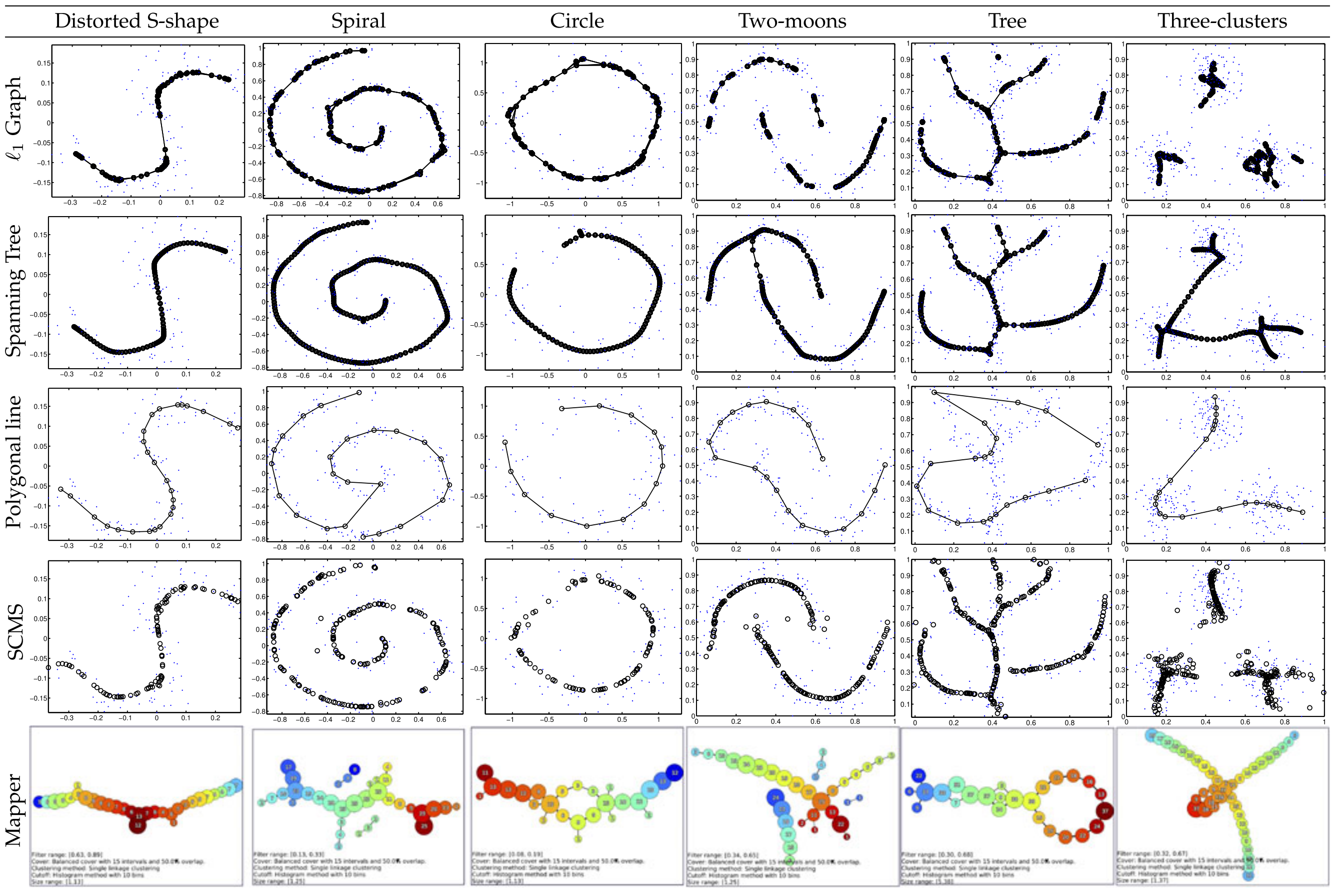


Fig. 5. Results of five compared methods performed on six synthetic datasets containing various situations including curves, loops, self-intersections, and multiple disconnected components. The first and second rows show the results of our proposed methods for learning either an ℓ_1 graph or a spanning tree. The last three columns report the results generated by the polygonal line method, SCMS, and Mapper, respectively. The results of Mapper can be interpreted as: the size of a node indicates the number of points in the set represented by the node; the color of a node indicates the value of filter function (red being high and blue being low) by a suitable average taken over the corresponding set, and the parameters used to generate figures are shown in the bottom-left side of each figure.

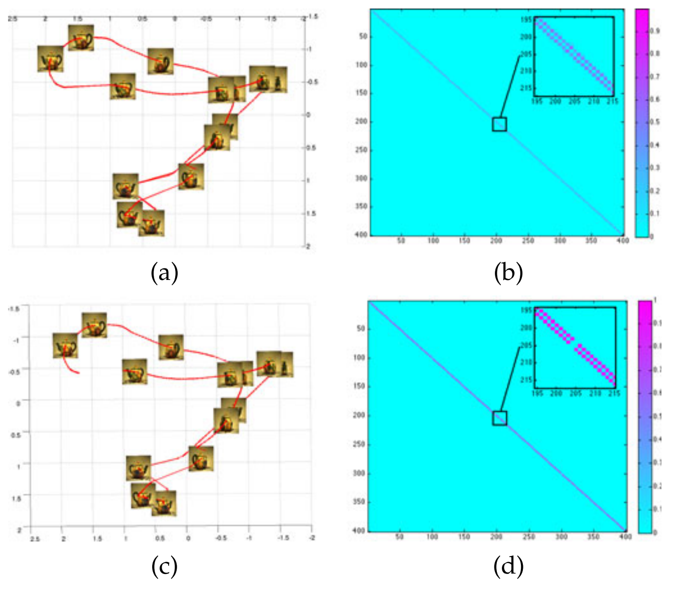


Fig. 6. Experimental results of the proposed methods applied to Teapot images. (a) Principal circle generated by the proposed method for learning a ℓ_1 graph. Each dot represents one teapot image. Images following the principal curve are plotted at intervals of 30 for visualization. (b) The adjacency matrix of the circle follows the ordering of the 400 consecutive teapot images with 360 degree rotation. (c)-(d) The principle curve and the adjacency matrix are obtained by the proposed method for learning a spanning tree.

on the tree data. However, the latter outperforms the former for datasets with multiple disconnected components and loops. All results in the second setting are consistent with those obtained in the first setting.

6.3 Rotation of Teapot Images

A collection of 400 teapot images from [36] are used.[2] These images were taken successively as a teapot was rotated 360 degree. Our goal is to construct a principal graph that organizes the 400 images. Each image consists of 76×101 pixels and is represented as a vector. The data in each dimension is normalized to have zero mean and unit standard deviation. Similar to [37], a kernel matrix $\mathbf{X}$ is generated where $X(i,j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/D)$.

We run our proposed method using the kernel matrix as the input. We set $\sigma = 0.5, \gamma = 100, \lambda = 1$ and $nn = 10$. We first perform PCA on the kernel matrix and project it to a 36-dimensional space that keeps 95 percent of total energy. The experimental results of the proposed method for learning a spanning tree and an ℓ_1 graph are shown in Fig. 6. The principal curves (Figs. 6a and 6c) are shown in terms of the first three columns of the learned projection matrix $\mathbf{W}$ as the coordinates where each dot $\mathbf{y}_i$ represents the ith image and the sampled images at intervals of 30 are plotted for the purpose of visualization. Figa. 6b and 6d show the linear chain dependency among the teapot images following the consecutive rotation process. We can see that the curve generated by our method is in agreement with the rotating process of the 400 consecutive teapot images. For this data, ℓ_1 graph is more reasonable than spanning tree since the underlying manifold forms a loop. This is clearly demonstrated in Fig. 6.

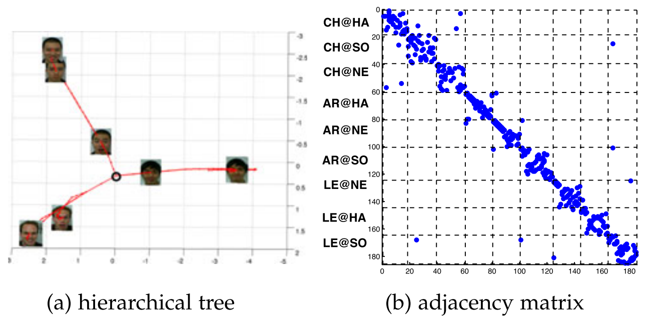


Fig. 7. Experimental results of our proposed method for learning a spanning tree performed on facial expression images. (a) The generated hierarchical tree. Each dot represents one face image. Images of three types of facial expressions from three subjects are plotted for visualization. The black circle can be considered as the root of the hierarchical structure for achieving two layers hierarchy over nine subjects; (b) the adjacency matrix of the tree on nine blocks indicates that each block corresponds to one facial expression of one subject.

A similar result is also recovered by CLUHSIC, which assumes that the label kernel matrix is a ring structure [37]. However, there are two main differences. First, the principal curves generated by our methods are much smoother than that obtained by CLUHSIC (see Fig. 4 in [37]). Second, our method learns the adjacency matrix from the given dataset, but CLUHSIC requires a label matrix as *a priori*. A two-dimensional representation of the same set of teapot images is given in [36], where Maximum Variance Unfolding (MVU) is used to arrange the images in a circle (see Fig. 3 in [36]). We attempted to run MVU by keeping 95 percent energy, i.e., $d = 36$. However, storage allocation fails due to the large memory requirement of solving a semi-definite programming problem in MVU. Hence, MVU cannot be applied to learn a relatively large intrinsic dimensionality. However, our method does not suffer from this issue.

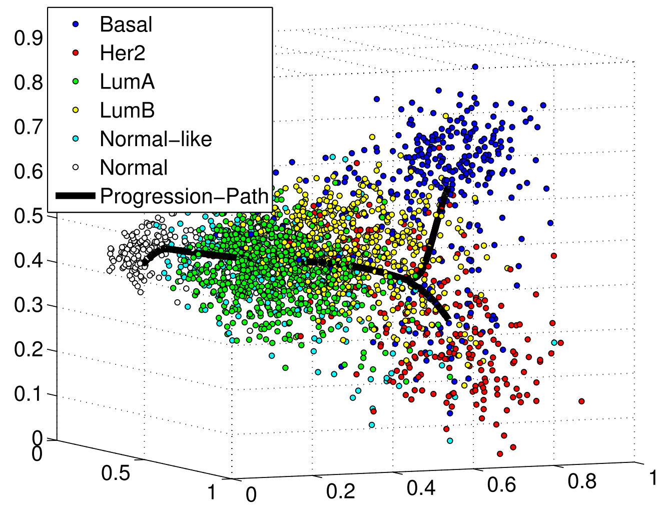
6.4 Hierarchical Tree of Facial Images

Facial expression data[3] is used for hierarchical clustering, which takes into account both the identities of individuals and the emotion being expressed [37]. This data contains 185 face images (308×217 RGB pixels) with three types of facial expressions (NE: neutral, HA: happy, SO: shock) taken from three subjects (CH, AR, LE) in an alternating order, with around 20 repetitions each. Eyes of these facial images have been aligned, and the average pixel intensities have been adjusted. As with the teapot data, each image is represented as a vector, and is normalized in each dimension to have zero mean and unit standard deviation.

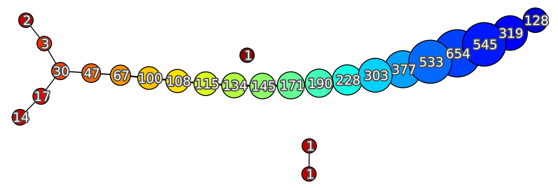
A kernel matrix is used as the input to Algorithm 1 for learning a spanning tree in the setting of $\sigma = 0.01$ and $\gamma = 1$. The experimental results are shown in Fig. 7. We can clearly see that three subjects are connected through different branches of a tree. If we take the black circle in Fig. 7a as the root of a hierarchy, the tree forms a two-level hierarchical structure. As shown in Fig. 7b, the three facial expressions from three subjects are also clearly separated. A similar two-level hierarchy is also recovered by CLUHSIC (Fig. 3b in [37]). However, the advantages of using the proposed method discussed above for the teapot images also apply here. In addition, we can observe more detailed information

2. http://www.cc.gatech.edu/~lsong/data/teapotdata.zip

3. http://www.cc.gatech.edu/~lsong/data/facedata.zip



(a) Proposed method



Filter range: [0.43, 0.88]
Cover: Hypercube cover. Intervals: (20,). Overlap: (50.0,)
Clustering method: Single linkage clustering
Cutoff: Biggest gap, exponent 0.0, max. clusters 50
Size range: [1,654]

(b) Mapper

Fig. 8. Progression path of breast cancer data. (a) The tree structure learned by the proposed method for learning a spanning tree; (b) the topological structure obtained by mapper.

from the tree structure. For example, LE@SO is at the junction with other two subjects, i.e., AR@SO and CH@SO, which can be observed from the 9th row of the adjacency matrix as shown in Fig. 7b. This observation suggests that the shock is the most similar facial expression among three subjects. However, CLUHSIC is not able to obtain this information.

6.5 Breast Cancer Progression Path

We demonstrate the ability of the proposed method to infer a putative cancer progression path using static tumor samples. To this end, we interrogate a large-scale, publicly available breast cancer dataset [57]. It contains the gene expression and copy number data of over 25,000 genes obtained from 144 normal breast tissue and 1,989 tumor tissue samples. By using a non-linear regression method, a total of 1,140 genes were identified to be associated with cancer development [41]. To visualize the cancer progression path, the learned principal points are projected onto a three-dimensional space spanned by the first three principal components of the data, and the tree structure remains as that learned in the original high-dimensional space.

In Fig. 8, we visualize the underlying data structures obtained by the proposed method, SCMS and Mapper, which may represent the progression path of breast cancer towards malignancy. As the polygonal line method does not work well on this data, we do not report the result. To assist in visualization, we color-code each tumor sample with its PAM50 molecular subtype label [58]. The PAM50 system broadly categorizes tumor samples into five subtypes, including normal-like, luminal A, luminal B, HER2+ and basal, based on the expression of 50 gene transcripts. We also use the 144 normal breast tissue samples in the dataset as a baseline. The revealed data trend supports a linear bifurcating progression path for breast cancer progression [59]. It is evident that the linear progression path tracks logically from normal tissue samples through luminal subtypes and then splits to either the HER2+ or the basal subtypes. The latter two are known to be the most aggressive breast tumor types. Our findings are consistent with previous studies [24], [41]. The comparisons of baseline methods based on the results shown in Fig. 8 are the same as those discussed in Section 6.2. We stress that the hypothesis of the interpretation of cancer development from the progression path requires additional biological verification. In this paper, we consider it as a plausible hypothesis only, and also show some possible analysis of the learned progression path in the supplementary material, available online.

6.6 Skeletons of Optical Characters

We use the proposed algorithm for learning an ℓ_1 graph to find smooth skeletons of handwritten character templates, which can be used to recover the trajectory of the penstroke. The optical characters generally contains loops, self intersections, and bifurcation points. Principal curves have been applied for similar purposes [13], [14]. Kegl et al. [14] extends their polygonal line algorithm [2] to handle loops and self intersections by modifying it with a table of rules and adding preprocessing and postprocessing steps. However, the table of rules has to be specially designed for the self intersections and it is difficult to extend to data sets with dimensionality larger than two. Ozertem [13] proposed KDE-SCMS, which can give satisfactory results without any rule or model based special treatment of the self intersections. KDE-SCMS can obtain a set of principal points to describe the skeletons of handwritten characters, but there is no explicit structures of these templates. Our proposed method for learning an ℓ_1 graph can effectively handle loops, self intersections, and disconnected components (possible noises). Moreover, we do not require a set of rules and simultaneously return the smooth skeletons of templates in the form of graphs.

The handwritten digits data set provided by Kegl, which can be downloaded from his website,[4] is the same dataset reported in [13], [14]. Following the preprocessing in work [14], we first transform black-and-white character templates into two-dimensional data sets where a unit length of the coordinate system is set to the width and height of a pixel, so that each pixel has integer coordinates. Then, we add pixels with value 1 into the data set. Finally, we scale the data set into a square area with unit width and height. We ran our proposed method for learning a weighted undirected ℓ_1 graph with one set of parameters for all handwritten characters, that is $\sigma = 0.008, \gamma = 100, \lambda = 0.4$ and $nn = 4$. Fig. 9 shows the results of sampled smooth skeletons on different character

4. https://www.lri.fr/~kegl/researchUdeM/research/pcurves/implementations/SkeletonizationTemplates/

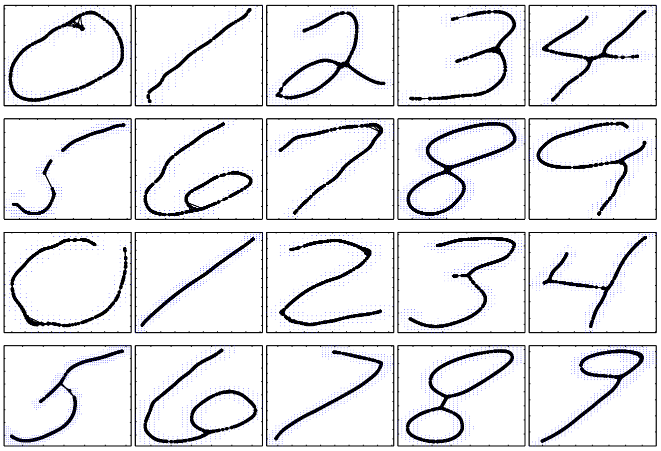


Fig. 9. Sampled results of the proposed RPG-ℓ_1 in finding the smoothing skeletons of optical characters.

templates with various written styles. More examples are shown in the supplementary material, available online.

6.7 Experiments on Large-Scale Datasets

We tested our proposed Algorithm 2 on two large-scale datasets: PenDigits and MNIST.[5] The PenDigits database is a handwritten digit dataset of 250 samples from 44 writers, which consists a total of 10,992 images represented by 16 features using sampled coordination information. The MNIST database of handwritten digits has a total of 70,000 samples. The digits have been size-normalized and centered in a fixed-size image. Each image of size 28×28 is represented as 784 features. We scale each gray pixel to $[0, 1]$ by dividing by 255. Both handwritten digit datasets have labels from 0 to 9. In order to visualize the learned graph, we apply PCA to keep 95 percent of total energy and obtain $D = 9$ and $D = 154$ for PenDigits and MNIST, respectively. We further rescale the projected low-dimensional sample by dividing by the maximum absolute value of each dimension independently.

We test Algorithm 2 for learning a tree structure. The same set of parameters, i.e., $\sigma = 0.1$ and $\lambda = 0.1$, is fixed for both datasets. Moreover, we apply the K-means method to obtain 1,000 centroids and use these centroids to initialize $\mathbf{Z}$ in Algorithm 2. The label of each centroid is obtained by using majority voting of labels of data points which are assigned to this cluster. Fig. 10 shows the results obtained by the proposed method on the two datasets. The learned adjacency matrix $\mathbf{W}$ is reported in Figs. 10a and 10b for PenDigits and MNIST respectively, where data points are reshuffled according to their labels from 0 to 9. We implemented the method in MATLAB and the empirical CPU times spent on learning $\mathbf{W}$ are 53.51 and 222.75 seconds for PenDigits and MNIST, respectively. We have the following observations: First, the learned adjacency matrices demonstrate a good shape of diagonal matrix. In other words, data points with the same labels are grouped together, which is useful for clustering. In addition, we have an explicit tree structure as shown in Fig. 10c drawn in the first three principal components, which further demonstrate the relationships among different labels. From Fig. 10c, it is clear to see

5. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

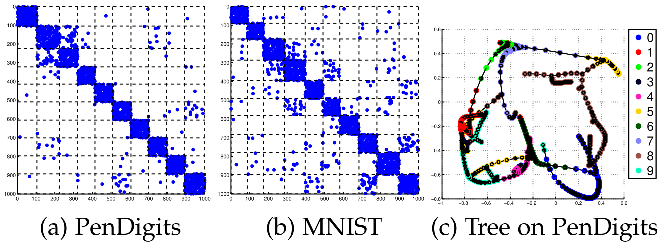


Fig. 10. Experiments on two large-scale datasets. (a)-(b) Adjacency matrices of 1,000 centroids after reshuffle according to labels from 0 to 9 for PenDigits and MNIST, respectively. (c) The learned tree structure of PenDigits over centroids with each label colored (best viewed in color) is visualized in the first three principal components.

that there is a path that starts from digit 0, passes through 6, 5, 9, and finally goes to 4. This is interesting since digits 0, 6, 5, and 9 are similar in the bottom halves of the images, while digits 9 and 4 are similar in terms of the top half of the images. That is, each path on the tree represents a certain type of manifold over digits. Furthermore, our proposed Algorithm 2 can be used for exploratory analysis on a large amount of data points.

7 Conclusion

In this paper, we propose a simple principal graph and structure learning framework, which can be used to obtain a set of principal points and a graph structure, simultaneously. The experimental results demonstrate the effectiveness of the proposed method on various real world datasets of different graph structures. Since our principal graph model is formulated for a general graph, the development of principal graph methods for other specific structures are also possible. As a future work, we will explore principal graph learning on other graphs such as K-nearest neighbor graphs and apply it to other real-world datasets.

Acknowledgments

Ivor W. Tsang is supported by the ARC Future Fellowship FT130100746 and ARC grant LP150100671.

References

[1] T. Hastie and W. Stuetzle, "Principal curves," *J. Amer. Statistical Assoc.*, vol. 84, pp. 502–516, 1989.
[2] B. Kégl, A. Kryzak, T. Linder, and K. Zeger, "Learning and design of principal curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 3, pp. 281–297, Mar. 2000.
[3] A. J. Smola, S. Mika, B. Schölkopf, and R. C. Williamson, "Regularized principal manifolds," *J. Mach. Learn. Res.*, vol. 1, pp. 179–209, 2001.
[4] S. Sandilya and S. R. Kulkami, "Principal curves with bounded turn," *IEEE Trans. Inf. Theory*, vol. 48, no. 10, pp. 2789–2793, Oct. 2002.
[5] A. Gorban and A. Y. Zinovyev, "Principal graphs and manifolds," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques*, vol. 31. Hershey, PA, USA: IGI Global, 2009, ch. 2, pp. 28–59.
[6] R. Tibshirani, "Principal curves revisited," *Statist. Comput.*, vol. 2, pp. 183–190, 1992.
[7] C. M. Bishop, M. Svensén, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Comput.*, vol. 10, no. 1, pp. 215–234, 1998.
[8] B. Kégl, "Principal curve: Learning, design, and applications," Ph.D. dissertation, Dept. Comput. Sci., Concordia Univ., Montreal, Quebec, Canada, 1999.

[9] A. Gorban, A. Pitenko, A. Zinovyev, and D. Wunsch, "Vizualization of any data using elastic map method," *Smart Eng. Syst. Des.*, vol. 11, pp. 363–368, 2001.

[10] A. Gorban and A. Y. Zinovyev, "Elastic principal graphs and manifold and their practical applications," *Computing*, vol. 75, no. 4, pp. 359–379, 2006.

[11] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer, 1997.

[12] E. Erwin, K. Obermayer, and K. Schulten, "Self-organizing maps: Ordering, convergence properties and energy functions," *Biol. Cybern.*, vol. 67, pp. 47–55, 1992.

[13] U. Ozertem and D. Erdogmus, "Locally defined principal curves and surfaces," *J. Mach. Learn. Res.*, vol. 12, pp. 1249–1286, 2011.

[14] B. Kégl and A. Kryzak, "Piecewise linear skeletonization using principal curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 59–74, Jan. 2002.

[15] D. Miao, Q. Tang, and W. Fu, "Fingerprint minutiae extraction based on principal curves," *Pattern Recognit. Lett.*, vol. 28, no. 16, pp. 2184–2189, 2007.

[16] A. Gorban, N. Sumner, and A. Zinovyev, "Topological grammars for data approximation," *Appl. Math. Lett.*, vol. 20, no. 4, pp. 382–386, 2007.

[17] M. Nagl, "Formal languages of labelled graphs," *Computing*, vol. 16, no. 1/2, pp. 113–137, 1976.

[18] M. Löwe, "Algebraic approach to single-pushout graph transformation," *Theoretical Comput. Sci.*, vol. 109, no. 1/2, pp. 181–224, 1993.

[19] A. N. Gorban and A. Zinovyev, "Principal manifolds and graphs in practice: From molecular biology to dynamical systems," *Int. J. Neural Syst.*, vol. 20, no. 3, pp. 219–232, 2010.

[20] G. Carlsson, "Topology and data," *Bulletin Amer. Math. Soc.*, vol. 46, no. 2, pp. 255–308, 2009.

[21] X. Ge, I. I. Safa, M. Belkin, and Y. Wang, "Data skeletonization via Reeb graphs," in *Proc. Advances Neural Inf. Process. Syst. 24*, 2011, pp. 837–845.

[22] T. K. Dey and Y. Wang, "Reeb graphs: Approximation and persistence," *Discr. Comput. Geom.*, vol. 49, no. 1, pp. 46–73, 2013.

[23] G. Singh, F. Mémoli, and G. E. Carlsson, "Topological methods for the analysis of high dimensional data sets and 3D object recognition," presented at the Eurographics Symp. Point-Based Graph., Prague, Czech Republic, 2007.

[24] M. Nicolau, A. J. Levine, and G. Carlsson, "Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival," *Proc. Nat. Academy Sci. United States America*, vol. 108, no. 17, pp. 7265–7270, 2011.

[25] U. Ozertem and D. Erdogmus, "Local conditions for critical and principal manifolds," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2008, pp. 1893–1896.

[26] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. 14th Int. Conf. Neural Inf. Process. Syst: Natural Synthetic*, 2001, pp. 585–591.

[27] M. Maier and U. Luxburg, "Influence of graph construction on graph-based clustering measures," in *Proc. 21st Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1025–1032.

[28] T. Jebara, J. Wang, and S. Chang, "Graph construction and b-matching for semi-supervised learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 441–448.

[29] E. Elhamifar and R. Vidal, "Sparse manifold clustering and embedding," in *Proc. 24th Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 55–63.

[30] B. Cheng, J. Yang, S. Yan, Y. Fu, and T. Huang, "Learning with ℓ_1-graph for image analysis," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 858–866, Apr. 2010.

[31] B. Lake and J. Tenenbaum, "Discovering structure by learning sparse graph," in *Proc. 33rd Annu. Cogn. Sci. Conf.*, 2010, pp. 778–783.

[32] Q. Mao, L. Wang, and I. W. Tsang, "A unified probabilistic framework for robust manifold learning and embedding," *Mach. Learn.*, 2016.

[33] L. Wang, Q. Mao, and I. W. Tsang, "Latent smooth skeleton embedding," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017.

[34] Q. Mao and I. W. Tsang, "Parameter-free spectral kernel learning," in *Proc. 26th Conf. Uncertainty Artif. Intell.*, 2010, pp. 350–357.

[35] Q. Mao, L. Yang, L. Wang, S. Goodison, and Y. Sun, "SimplePPT: A simple principal tree algorithm," in *Proc. 15th SIAM Int. Conf. Data Mining*, 2015, pp. 792–800.

[36] K. Q. Weinberger and L. K. Saul, "An introduction to nonlinear dimensionality reduction by maximum variance unfolding," in *Proc. 21st Nat. Conf. Artif. Intell.—Vol. 2*, 2006, pp. 1683–1686.

[37] L. Song, A. Smola, A. Gretton, and K. Borgwardt, "A dependence maximization view of clustering," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 815–822.

[38] J. B. Tenenbaum, V. deSilva, and J. C. Landford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[39] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[40] M. Greaves and C. C. Maley, "Clonal evolution in cancer," *Nature*, vol. 481, no. 7381, pp. 306–313, 2012.

[41] Y. Sun, J. Yao, N. Nowak, and S. Goodison, "Cancer progression modeling using static sample data," *Genome Biol.*, vol. 15, no. 8, 2014, Art. no. 440.

[42] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, Aug. 1995.

[43] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Amer. Math. Soc.*, vol. 7, pp. 48–50, 1958.

[44] M. Cheung, "Minimum-cost spanning trees," (2008). [Online]. Available: http://people.orie.cornell.edu/dpw/orie6300/fall2008/Recitations/rec09.pdf

[45] Q. Mao, L. Wang, S. Goodison, and Y. Sun, "Dimensionality reduction via graph structure learning," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 765–774.

[46] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions—A survey and extensions," *Math. Methods Operations Res.*, vol. 66, pp. 373–407, 2007.

[47] E. D. Andersen and K. D. Andersen, "The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm," in *High Performance Optimization*. Berlin, Germany: Springer, 2000, pp. 197–232.

[48] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 577–584.

[49] E. P. Xing, M. I. Jordan, S. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Proc. Advances Neural Inf. Process. Syst.*, 2002, pp. 505–512.

[50] J. Zhuang, I. W. Tsang, and S. C. Hoi, "A family of simple non-parametric kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 12, pp. 1313–1347, 2011.

[51] C. Fowikes, S. Belongle, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, Feb. 2004.

[52] X. Chen and D. Cai, "Large scale spectral clustering with landmark-based representation," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 313–318.

[53] D. Yan, L. Huang, and M. Jordan, "Fast approximation spectral clustering," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 907–916.

[54] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling method for the Nyström method," *J. Mach. Learn. Res.*, vol. 13, pp. 981–1006, 2012.

[55] M. Á. Carreira-Perpiñán, "A review of mean-shift algorithms for clustering," *arXiv preprint arXiv:1503.00687*, 2015.

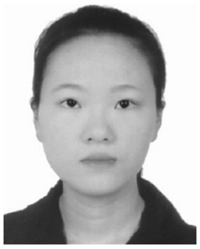
[56] D. M. Witten and R. Tibshirani, "A framework for feature selection in clustering," *J. Royal Statistical Soc.*, vol. 105, no. 490, pp. 713–726, 2010.

[57] C. Curtis, et al., "The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups," *Nature*, vol. 486, no. 7403, pp. 346–352, 2012.

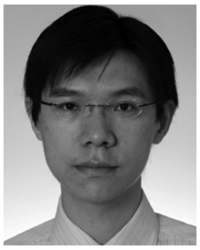
[58] J. Parker, et al., "Supervised risk predictor of breast cancer based on intrinsic subtypes," *J. Clinical Oncology*, vol. 27, no. 8, pp. 1160–1167, 2009.

[59] C. J. Creighton, et al., "The molecular profile of luminal B breast cancer," *Biologics*, vol. 6, no. 2, pp. 289–297, 2012.

Qi Mao received the bachelor's degree in computer science from Anhui University, Hefei, in 2005, the master's degree in computer science from Nanjing University, Nanjing, in 2009, and the PhD degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014. He is currently a senior researcher with HERE company, Chicago, Illinois. His research interests include machine learning, data mining, and large-scale data analysis.



Li Wang received the bachelor's degree in information and computing science from China University of Mining and Technology, Jiangsu, China, in 2006, the master's degree in computational mathematics from Xi'an Jiaotong University, Shaanxi, China, in 2009, and the PhD degree from the Department of Mathematics, University of California, San Diego, California, in 2014. She is currently a research assistant professor in the Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago. She worked as the postdoctoral fellow with the University of Victoria, BC, Canada, in 2015, and Brown University, Providence, RI, in 2014. Her research interests include large-scale optimization, semi-infinite programming, and machine learning.



Ivor W. Tsang is an ARC future fellow and a professor of artificial intelligence with the University of Technology Sydney. He is also the research director of the UTS Priority Research Centre for Artificial Intelligence. His research focuses on transfer learning, feature selection, big data analytics for data with trillions of dimensions, and their applications to computer vision and pattern recognition. He has more than 140 research papers published in top-tier journal and conference papers, including the *Journal of Machine Learning Research*, the *Madras Law Journal*, the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, the *IEEE Transactions on Neural Networks and Learning Systems*, NIPS, ICML, etc. In 2009, he was conferred the 2008 Natural Science Award (Class II) by Ministry of Education, China, which recognized his contributions to kernel methods. In 2013, he received his prestigious Australian Research Council Future Fellowship for his research regarding Machine Learning on Big Data. In addition, he had received the prestigious IEEE Transactions on Neural Networks Outstanding 2004 Paper Award in 2007, the 2014 IEEE Transactions on Multimedia Prize Paper Award, and a number of best paper awards and honors from reputable international conferences, including the Best Student Paper Award at CVPR 2010, and the Best Paper Award at ICTAI 2011. He was also awarded the ECCV 2012 Outstanding Reviewer Award.

Yijun Sun received two BS degrees in electrical and mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1995, and the MS and PhD degrees in electrical engineering from the University of Florida, Gainesville, Florida, in 2003 and 2004, respectively. From 2005 to 2012, he was an assistant scientist in the Interdisciplinary Center for Biotechnology Research and an affiliated faculty member in the Department of Electrical and Computer Engineering, University of Florida. He is now an assistant professor of bioinformatics in the Department of Microbiology and Immunology and the Center of Excellence in Bioinformatics and Life Sciences, and an adjunct faculty member in the Departments of Computer Science and Engineering, Biostatistics, and Biomedical Informatics, University at Buffalo, State University of New York, Buffalo. His research interests include machine learning, bioinformatics, and their applications to cancer informatics and metagenomics. He is a co-recipient of the 2005 IEEE M. Barry Carlton Best Transactions Paper Award. One of his papers is selected as the Spotlight Paper in the September 2010 issue of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* journal. His research is supported by NSF, NIH, Bankhead-Coley Cancer Research Program, and Susan Komen Breast Cancer Foundation, and his work on metagenomics and feature selection has been used by more than 200 research institutes worldwide.