Neurocomputing 312 (2018) 336–351

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Data-driven graph construction and graph learning: A review

Lishan Qiao [a,c], Limei Zhang [a,c], Songcan Chen [b,*], Dinggang Shen [c,d]

[a] School of Mathematics, Liaocheng University, Liaocheng 252000, China
[b] School of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China
[c] Department of Radiology and BRIC, University of North Carolina at Chapel Hill, NC 27599, USA
[d] Department of Brain and Cognitive Engineering, Korea University, Seoul 02841, Republic of Korea

ARTICLE INFO

Article history:
Received 14 September 2017
Revised 20 February 2018
Accepted 21 May 2018
Available online 1 June 2018

Communicated by Prof. Hanghang Tong

Keywords:
Graph construction
Machine learning
Sparse representation
Low-rank representation
Multi-graph learning
Brain network

ABSTRACT

A graph is one of important mathematical tools to describe ubiquitous relations. In the classical graph theory and some applications, graphs are generally provided in advance, or can *at least* be defined clearly. Thus, the main focus is to measure/analyze the graphs for mining informative patterns. However, for many real-world scenarios, the graph is often uncertain due to the fact that data associated with the vertices in the graph are high-dimensional, noisy, differently distributed, and even no clear definition. As a result, one needs to design or learn graphs from data prior to any analysis, which *in turn* affects the subsequent tasks. Therefore, constructing a high-quality graph has become an increasingly hot research problem, which inspired many graph construction methods being proposed in the past years. Since there has been no systematic summary on this topic, in this paper, we review the main-stream graph construction/learning methods involved in *both* general machine learning algorithms (including semi-supervised learning, clustering, manifold learning, and spectral kernel learning, etc.) *and* some specific applications (especially, the modeling and analysis of functional brain connectivity). Additionally, we introduce a matrix-regularized graph learning framework that can benefit to unify some existing graph construction models and develop new graph learning algorithms. Finally, we discuss several related topics and some promising research directions in this field.

1. Introduction

We live in an interconnected world. Graph theory provides an effective tool to model and analyze such interconnectivities. As an old mathematical branch, graph theory can be traced back to the well-known Konigsberg's bridge problem, for which Euler proposed an elegant solution in 1736. Even so, the word "graph" was not used in its current sense [1], until nearly one and a half centuries after Euler's work. From then on, graph theory is successfully applied in various fields, ranging from traditional structural mechanics, traffic scheduling, to modern electronic communication, protein function prediction [2], gene co-expression modeling [3], brain network analysis [4,5], and so on.

In typical graph theory, graphs are generally given in advance, or can be easily abstracted from practical problems. For example, the well-known traveling salesman problem [6] can be modeled as an undirected weighted graph, with its vertices as cities, edges as roads, and edge-weights as road lengths. Consequently, the main focus is to analyze valuable properties or mine informative patterns hidden in graphs, such as small-worldness, global efficiency, modularity, etc. However, for a large family of practical problems, including functional brain connectivity analysis [4], gene co-expression network modeling [3], frequent subgraph mining [7, 8], and many general machine learning methods such as spectral clustering, manifold learning and semi-supervised learning, the graphs are generally not off-the-shelf, but need well-design or learn from data.

On one hand, the graph is at the heart of the above mentioned learning methods [9–11], and even an inevitable step in some applications [3, 4]; on the other hand, the graph construction/learning is usually challenging due to the fact that data (associated with the nodes of a graph) have one or more of the following characteristics.

1) The data lie in a high-dimensional space. This is a chief culprit that makes trouble for graph construction due to the so-called "curse of dimensionality" [12].
2) The data are derived from different "distributions", such as a (multi-)subspace structure, clustering structure, (multi-)manifold structure, and even their mixtures. Undoubtedly, we need to construct graphs according to different distribution assumptions.

* Corresponding author.
E-mail addresses: s.chen@nuaa.edu.cn (S. Chen), dgshen@med.unc.edu (D. Shen).

https://doi.org/10.1016/j.neucom.2018.05.084
0925-2312/

3) The data are generally contaminated by noises. Even though we know the data are theoretically derived from a certain structure, in practice, it is almost never the case since noises may scatter around. This gap between theory and practice makes the graph construction more difficult.
4) The data, as the nodes of a graph, are sometimes uncertain. For example, in functional brain network modeling, each node generally corresponds to the data (e.g., the signals of neural activity) associated with a brain region [13]. Obviously, different parcellations of the brain region may result in different signals (nodes) and thus different graph structures.

As a consequence, one needs to construct graphs from the data carefully prior to any analysis or learning task. In this paper, we call this data-driven graph construction or graph learning for standing out from traditional graph theory. Since graphs usually have a great impact on the performance of the ensuing algorithms/applications [9, 11, 14], in the recent years various graph construction/learning methods have been developed based on different motivations, priors, assumptions, or applications.

However, to our best knowledge, there has been no systematical summary work on this topic. Therefore, in this paper, we review almost all popular graph construction (learning) methods, *not only* involved in general machine learning problems such as clustering, manifold learning and semi-supervised learning, *but also* in some specific applications. More specifically, (1) we summarize several dozen graph construction methods from a data-driven perspective, categorize them, and rewrite their models (if any) using unified mathematical notations, which benefit the understanding and comparison of the motivations behind different methods; (2) We discuss the advantages/disadvantages of each kind of methods, and provide suggestions for guiding the method selection in practice; (3) We introduce a matrix-regularized graph learning framework that can give clearer explanations for some existing graph construction methods and provide a platform to develop new graph learning models; (4) We discuss the relationship between graph learning and several related fields, including kernel learning, metric learning, and dimensionality reduction, which help us understand these topics in a unified view.

The rest of this paper is organized as follows. In Section 2, we introduce basic notations and concepts. In Section 3, we systematically review the graph construction/learning methods. In Section 4, we introduce the matrix-regularized graph learning framework. In Section 5, we discuss related topics and promising directions in this field. Finally, we conclude the paper in Section 6.

2. Basic notation and concept

In this section, we first list the notations in Table 1. Then, we introduce the basic concepts related to the graph and some popular similarity "metrics" widely used in graph construction.

A graph G(V, E) is made up of a set of vertices (or nodes) V, and a set of edges E that connect the vertices. In this paper, we restrict the vertices to vector data, i.e., $V = \{x_i \in R^d\}_{i=1}^n$, since more general data structures (*e.g.*, matrices, tensors, strings, trees and graphs) can be vectorized by concatenating or embedding into a vector space [15], and, without loss of generality, we rewrite the point set as an ordered matrix $V = X = [x_1, x_2, \ldots, x_n] \in R^{d\times n}$. Also, we do not distinguish the specific physical meaning of the nodes in a graph unless stated otherwise. For example, *a node may indicate a reshaped image, an fMRI signal, and even a parametric vector* corresponding to a "task" in multi-task learning, etc. The edges in a graph are directed or undirected, and, equivalently, the edge set E can be considered as a binary matrix whose element $E_{ij} = 1$ if existing an edge between x_i and x_j, and 0 otherwise. A weight W_{ij} is assigned to each edge for measuring the link strength, which is usually a necessary step for graph learning. Therefore, we describe a graph by a three-tuple $G(V, E, W)$.

As we know, a (dis)similarity score is important to measure the pair-wise relations between nodes, *not only* for determining the edge set E, *but also* for computing the edge weight W. Therefore, in Table 2, we summarize popular (dis)similarity measures widely used in graph construction. Note that we present dissimilarity measures separately from similarity measures only for consistency with their original expressions. For example, in statistics and machine learning communities, the similarities are generally used to describe the dependency between *variables* (features), while dissimilarities tend to be used for measuring the difference between *samples*. However, a dissimilarity measures can be easily converted to its similarity counterpart by "*similarity* = *1*−*normalized dissimilarity*". Also, it is worth emphasizing that Table 2 merely covers some simple and popular (dis)similarities for vector data. In practice, however, various "metrics", such as K-L divergence, mutual information and edit distance, have been developed to measure non-vector data. Beyond these, some studies [16-19] also designed different scores according to specific applications.

3. Graph construction and learning methods

Given a vertex set V, two main steps are generally conducted to construct a graph. (1) Determine the topological structure of the graph, i.e., the edge set E (*E-step* for short); (2) Based on the current edge set, determine the weight matrix W (*W-step* for short). For some scenarios or methods, there may be no principled boundary between these two steps. For example, we can simply set $W = E$, meaning that these two steps are merged. Another example is the sparse representation based methods that learn the edge weight W of a graph directly, but implicitly induce the graph topology E due to the sparsity constraint (see Section 3.2.5 for details). On the other hand, however, some graph construction methods actually work in two separate steps. For example, in LLE algorithm [20], the topological structure of a graph is first defined by Euclidean distance, and then the edge weight is assigned by the reconstructive coefficient in the pre-defined local topological structure (see Section 3.2.3 for details). Therefore, we use a two-step framework in this paper for covering as many methods as possible. Note that such a two-step framework is only used for the graph construction process. Once the construction is completed, the topological structure E can be completely contained in the edge weight matrix W. In Sections 3.1 and 3.2 below, we review the methods related to E-step and W-step, respectively.

3.1. Graph topology definition/learning (E-step)

For the graph topology definition or learning involved in E-step, the current methods can be classified into three groups. The first group includes (1) minimum spanning tree, (2) Delaunay triangulation, (3) Gabriel graph, and (4) relative neighborhood graph, all of which are parameter-free. The second group includes (5) beta-skeleton, (6) k nearest neighbors, (7) ε-ball neighborhood, and (8) b-matching, each of which has one parameter. The third group includes methods in (9), which generally employ more flexible parameter selection schemes towards robust and adaptive proximity graphs.

Given any similarity/dissimilarity score, as shown in Table 2, one can naturally get a *fully* connected similarity matrix C (or equivalently, a dissimilarity matrix D) whose entries measure the pair-wise similarities/dissimilarities of the vertices. In *E-step,* the graph topology learning methods essentially rely on how to sparsify such a connected matrix. Note that the sparsity of a graph topology is important since it tends to result in higher efficiency, better interpretation, and stronger robustness [21].

Table 1
Notation description.

Notation	Description	Notation	Description
G	Graph	$\sigma/\sigma_{ij}/\sigma_k$	Scale parameter of Gaussian kernel
V	Vertex (or node) set of a graph	α	Coefficient vector for combing multi-graphs
E	Edge set/matrix of a graph	E_{ij}	Entries of the edge set (or binary matrix) E
W	Weight matrix associated with the edge set E	W_{ij}	Entries in edge weight matrix W
L	Graph Laplacian	d_{ij} or D_{ij}	Dissimilarity between nodes x_i and x_j
C	Fully connected pairwise similarity matrix	d_i	Degree of the node x_i
D	Fully connected pairwise **dis**similarity matrix	$\rho_{ij\cdot V\setminus\{i,j\}}$	Partial correlation between x_i and x_j
P	Projection matrix	$1(\cdot)\rightarrow\{0,1\}$	Indicator function
I	Identity matrix	$\exp(\cdot)$	Exponent function
Σ	Covariance matrix	$L(\cdot,\cdot)$	Loss function
S	Sample covariance matrix	$d(\cdot,\cdot)$	Distance function
Θ	Inverse covariance matrix $\Theta=\Sigma^{-1}$	$N_k(x_j)$	Subset of k nearest neighbors for node x_j
A	Dictionary as basis for linear representation	$\mathrm{rank}(\cdot)$	Rank of a matrix
Z	Encoding matrix based on dictionary A	$\mathrm{tr}(\cdot)$	Trace operator of a matrix
1	A matrix or vector whose entries are all ones	$\lvert\cdot\rvert$	Determinant of matrix
X	Data matrix $X=[x_1,x_2,\cdots,x_n]\in R^{d\times n}$	$(\cdot)_p^{1/p}$	p-norm of vector
d	Dimensions of the vector associated with nodes	W_F	F-norm of matrix W, i.e., $(\sum_j\sum_i W_{ij}^2)^{1/2}$
n	The number of nodes in a graph	W_1	L_1-norm of matrix W, i.e., $\sum_j\sum_i\lvert W_{ij}\rvert$
$x_i\in R^d$	d-dimensional vector associated with the ith node	$W_{2,1}$	$L_{2,1}$-norm of matrix W, i.e., $\sum_j(\sum_i W_{ij}^2)^{1/2}$
$\bar{x}_i$	Average value of the entries in vector x_i	W_*	Trace (or nuclear) norm of matrix W
λ	Regularized parameter	$W\geq 0$	Non-negative matrix
k, ε, β, b	Para. in kNN, ε-N, β-skeleton, and b-matching	$W\succcurlyeq 0$	Matrix W is positive semi-definite

Table 2
Similarities and dissimilarities commonly used in graph construction.

(Dis) similarity	Mathematical expression	Comments
Cosine Similarity	$\frac{x_i^T x_j}{\sqrt{x_i^T x_i}\sqrt{x_j^T x_j}}$	It is a baseline to measure the similarity of text data.
Pearson's Correlation	$\frac{(x_i-\bar{x}_i)^T(x_j-\bar{x}_j)}{\sqrt{(x_i-\bar{x}_i)^T(x_i-\bar{x}_i)}\sqrt{(x_j-\bar{x}_j)^T(x_j-\bar{x}_j)}}$ where $\bar{x}_i=1x_i/d$, and 1 is the all-one matrix.	(1) It is essentially a centralized cosine similarity; (2) it has been widely used to estimate both functional brain networks and gene co-expression networks.
Spearman's Correlation	$\frac{(x_i^r-\bar{x}_i^r)^T(x_j^r-\bar{x}_j^r)}{\sqrt{(x_i^r-\bar{x}_i^r)^T(x_i^r-\bar{x}_i^r)}\sqrt{(x_j^r-\bar{x}_j^r)^T(x_j^r-\bar{x}_j^r)}}$	Here, x_i^r is the ordinal counterpart of x_i, and so this expression is a ranked version of Pearson's correlation.
Partial Correlation	$\rho_{ij\cdot V\setminus\{i,j\}}=-\theta_{ij}/\sqrt{\theta_{ii}\theta_{jj}}$ where $(\theta_{ij})_{d\times d}=\Theta=\Sigma^{-1}$ is the inverse covariance matrix (or precision matrix).	(1) It has been used in functional brain network construction, and (2) closely related to the inverse covariance matrix (see Section 3.2.4 for details).
Simple Matching Coef.	$\frac{1}{d}\sum_{k=1}^d 1(x_{ik}=x_{jk})$ where $1(\cdot)\rightarrow\{0,1\}$ is an indicator function.	(1) It is generally used for binary data, *e.g.*, labels, and (2) closely related to hamming distance below.
Hamming Distance	$\sum_{k=1}^d 1(x_{ik}\neq x_{jk})$ where $1(\cdot)\rightarrow\{0,1\}$ is an indicator function.	(1) Widely used in multi-label learning and coding theory; (2) normalization is recommended by d.
Jaccard Index	$\sum_{k=1}^d min(x_{ik},x_{jk})/\sum_{k=1}^d max(x_{ik},x_{jk})$	A generalization for real data, but commonly used for binary data, similar to the simple matching coefficient.
Minkowski Metric	$(x_i-x_j)_p^{1/p},\ p>0$	Manhattan distance when $p=1$; Euclidean distance when $p=2$; Chebychev distance when $p=\infty$.
Mahalanobis Dist.	$(x_i-x_j)^T\Sigma^{-1}(x_i-x_j)$, Σ is covariance matrix.	It reduces to Euclidean distance when $\Sigma=I$.
Chi-square Distance	$\sum_{k=1}^d\frac{(x_{ik}-x_{jk})^2}{x_{ik}+x_{jk}}$	Generally used to measure the dissimilarity between histograms.

1) *Minimum spanning tree (MST).* As one of the simplest sparsification strategies, MST defines the graph topology E by searching a minimum sum of the pairwise scores based on the dissimilarity matrix D. Although it has been successfully applied in, for example, functional brain networks modeling and analysis [22], the traditional MST is too sparse, containing only $n-1$ edges and no cycles, where n is the number of the nodes. As a result, it is hard to capture the complex data structure involved in many machine learning tasks. To address this problem, Zemel and Carreira-Perpiñán [23] recently proposed to perturb the data randomly for generating multiple MSTs and then determine the final graph topology E by the MST ensemble. In fact, several other proximity graphs, reviewed in the following cases (2)–(4), can also handle the "oversparsity" problem, and they have an interesting inclusion relation.
2) *Delaunay triangulation (DT).* Given a set of vertices V, DT is defined as a triangulation such that no vertices in V are inside the circum-hypersphere of any simplex. It is well known that DT is a dual of the Voronoi tessellation, and plays an important role in computer graphics. However, DT is generally noise-sensitive, and not good enough at modeling the manifold structure since it may connect the vertices *non-locally* [23]. In contrast, the Gabriel graph (GG) and Relative neighborhood graph (RNG) we will discuss below can alleviate this problem to some extent.
3) *Gabriel graph (GG).* GG is a subgroup of DT, named from Gabriel and Sokal [24]. Based on the dissimilarity matrix D, the graph topology E can be defined as follows. $E_{ij}=1$ (meaning the existence of an edge between node x_i and x_j) if the pairwise distances meet $d_{ij}\leq(d_{ik}^2+d_{jk}^2)^{1/2},\ \forall k\neq i,j$ and $i\neq j$, and $E_{ij}=0$ otherwise. In other words, $E_{ij}=1$ means that there is no point falling in the hyper-sphere with the diameter d_{ij}. In [25], GG is used in an agglomerate clustering method for determining which clusters are neighboring.
4) *Relative neighborhood graph (RNG).* Similar to GG, the topological structure of RNG is defined as that $E_{ij}=1$ iff $d_{ij}\leq\max\{d_{ik},$

$d_{jk}\}, \forall k \neq i,\ j \text{ and } i \neq j$, and 0 otherwise. RNG was first proposed by Toussaint in [26], and pointed out the inclusion relation that MST⊆RNG⊆GG⊆DT, where "⊆" means "is a subgraph of". Recently, Toussaint wrote a comprehensive summary on the RNG's applications in different fields [27]. Although the methods in the first group (i.e., MST, RNG, GG and DT) can determine graph topologies with different sparsity, they are not flexible enough for capturing complex structures in data. In contrast, the methods in (5)–(8) below can address this problem effectively.

5) *Beta-skeleton*. The β-skeleton is a generalization of the GG and RNG, with one adjustable parameter $\beta > 0$ [28]. In particular, it reduces to RNG if $\beta = 1$, and GG if $\beta = 2$. So, the β-skeleton is more flexible than both GG and RNG to model the locality of the data. Recently, Correa et al. explore the β-skeleton as a tool for clustering analysis, and the empirical results show that the clustering is relatively stable for different values of β [29].
6) *k nearest neighbors (kNN)*. kNN is possibly the most commonly used approach for defining graph topology, and has been successfully applied to almost all graph-based machine learning problems. For each node x_i, based on the fully connected matrix C, kNN first searches its k nearest neighbors set denoted by $N_k(x_i)$, and then defines the graph topology E according to the following symmetrization strategy that $E_{ij} = 1$ if $x_j \in N_k(x_i)$ *or/and* $x_i \in N_k(x_j)$, and $E_{ij} = 0$ otherwise. Note that, however, kNN graph is generally sensitive to noise or outlier points, and it might not be adequately rational to use a fixed neighborhood size for all nodes without considering the data distribution or sampling density. In Case 9) below, we will discuss several schemes to alleviate this problem.
7) *ε-ball neighborhood (ε-N)*. ε-N method gets the graph topology E from the connected matrix C by simply setting $E_{ij} = 1$ if $C_{ij} \geq \varepsilon$, or 0 otherwise, where ε is a pre-defined threshold. Note that the symmetrization used in kNN is avoidable for ε-N graph *unless* the matrix C itself is *asymmetric*. ε-N is a commonly used method for sparsifying both brain networks and gene co-expression networks, where the connected matrix C is often a Pearson's correlation matrix [30]. In contrast, the machine learning community prefers to use kNN strategy, because ε-N tends to generate many disconnected components in a graph even with careful selection of the parameter ε, which is undesirable since some algorithms (*e.g.*, label propagation [31]) cannot work well on a disconnected graph.
8) *B-matching*. In graph theory, there is a class of graphs, known as regular graphs. The typical examples include ring or grid lattice graphs. In [21], Jebara et al. suggested that the regularity helps to produce more robust proximity graphs, and thus potentially improve the subsequent performance. In particular, they employed the b-matching method toward this goal as follows.

$$\begin{aligned} &\min_{E} \sum_{i,j=1}^{n} E_{ij} D_{ij} \\ &\text{s.t.} \sum_{i,j=1}^{n} E_{ij} = b,\ E_{ii} = 0,\ E_{ij} = E_{ji} \end{aligned} \tag{1}$$

where D_{ij} is the pairwise distance of the nodes and b is a "neighbor" size. In fact, the traditional kNN graph (before being symmetrized) can be formulated as a similar optimization problem by just replacing b with k and further removing the symmetric constraint $E_{ij} = E_{ji}$ [21]. The only difference is that kNN-like methods "prune" the edges E_{ij} via a greedy algorithm and produce an asymmetric graph. Although the asymmetric graph has the same row degree k for all the nodes, the column degree is uncertain. Thus, the constructed proximity graph is often irregular and imbalanced even with a post-processing of symmetrization. In contrast, the b-matching directly solves Eq. (1) and, due to the symmetric constraint, it achieves a symmetric and regular graph structure. Recently, Jebara and Huang further improved the efficiency of b-matching by an enhanced belief propagation algorithm [32].
9) *Towards robust and adaptive proximity graph.* Intuitively, it is more reasonable to *change the "neighbor" parameter values adaptively according to variant sample densities and manifold curvature.* Several recent studies also empirically support this point, and provide interesting strategies towards constructing robust and adaptive proximity graphs. Based on their previous works [33], Zhang et al. proposed an *adaptive* method to select neighbor parameter values for each node by well-designed neighborhood contraction and expansion algorithms [34]. Dominant neighbor [18] is another method which can select variable-sized neighbors for different nodes. In particular, it refines the traditional kNN neighbor set $N_k(x_i)$ by searching the maximal clique $DN(x_i) \subseteq N_k(x_i)$. By removing the possible noisy neighbors, the dominant neighbors were shown to be more robust than the traditional kNN method [18]. Also, in [35] the authors proposed to make use of consensus information from multiple k-NN procedures, aiming to construct better graphs. In brief, if two nodes x_i and x_j keep appearing among the same k nearest neighbors of a certain node, then the linking chance of the two nodes should be enhanced. Finally, a global threshold (similar to the ε-N scheme) is introduced to get a variable-sized neighborhood that is expected to represent the local neighborhood structure adaptively. In addition, a recent study [36] proposed a unified and generalized data similarity inference framework based on the unsupervised clustering random forest for selecting adaptive neighborhoods in graph construction, to name a few.

As a summary of E-step, we discuss the (dis)advantages of the graph topology learning strategies reviewed in this section from the view of parameter selection. For the methods with parameters, including β-skeleton, kNN, ε-**N** and b-matching, they generally have more flexibility to model the geometric structure of the data, but the final performances usually depend heavily on the selected parameter values. For example, a recent study [37] investigated the big influence of the thresholding parameter ε on the results of the brain network analysis. Therefore, we suggest parametric methods for supervised learning tasks (e.g., classification) with sufficient training samples, since cross validation, in spite of its time-consuming computation [38], can be used as an effective way to conduct parameter selection. In contrast, we believe that the parameter-free graph construction methods, including MST, RNG, GG and DT, are particularly valuable for unsupervised learning or semi-supervised learning with limited labels, where the parameter selection may be extremely challenging without the guidance of label information. However, the parameter-free methods always lead to deterministic topological structures, and thus fail to model the complex data distributions. In this case, the methods reviewed in the case 9) above provide a potential solution for selecting adaptive parameters by heuristic strategies.

3.2. Edge weight definition/learning (W-step)

Similar to E-step, one can also determine edge weights in W-step by direct **definition** (Section 3.2.1) or automatic **learning** from data (Sections 3.2.2–3.2.8).

3.2.1. Edge weight redefinition

The most direct way to compute the weight matrix W is based on a given (dis)similarity measure. *Although W-step can share the same (dis)similarity used in E-step, in practice it generally redefines the weight matrix W using different measures for better interpretability.*

1) *Binary weight*. The simplest method for assigning edge weight is to set $W = E$ directly. Obviously, such a scheme cannot provide any extra information beyond graph topology, and so, in the following (2)–(4), we review several kinds of informative edge weights widely used in graph construction.
2) *Gaussian kernel and its variants*. Gaussian kernel (a.k.a., heat kernel or RBF kernel) may be the most popular scheme to assign edge weights for a graph. The commonly used weight expression is.

$$W_{ij} = E_{ij} \cdot \exp\left(-d\left(x_i, x_j\right)^2 / 2\sigma^2\right) \tag{2}$$

where $d(x_i, x_j)$ is a pairwise dissimilarity measure, and σ is a scale (or bandwidth) parameter. E_{ij} is the graph topology preobtained in E-step for filtering out the useless weights. That is, $W_{ij} = \exp(-d(x_i, x_j)^2/2\sigma^2)$ if $E_{ij} = 1$, and 0 otherwise. One can employ $\exp(-\sum_k d(x_{ik}, x_{jk})^2/2\sigma_k^2)$ or $\exp(-\sum_k d(x_i, x_j)^2/2\sigma_{ij}^2)$ for defining more flexible edge weights, where the former uses different scales for different dimensions, while the latter use different scales for different node pairs.

In principle, any dissimilarity measure listed in Table 2 can be used in Eq. (2) to define edge weights for different purposes or data. For example, the χ^2 distance is generally employed to measure the dissimilarity of histogram data. Note that *Eq. (2) in fact provides a way to convert dissimilarities to similarities.* In addition, it is worth pointing out that the parameter σ, despite providing more flexibility to adapt to the local data distribution, undoubtedly leads to the intractable parameter selection problem. Thus, *different tricks have been developed for selecting a "good" σ*. In particular, Gretton et al. heuristically set σ as the median among the aggregate nodes [39]; Zelnik-Manor and Perona first assumed $\sigma_{ij} = \sigma_i\sigma_j$, and then assigned a local scale for each data point [40]; Yang et al. used an adaptive scale size σ_{ij} based on the average distance between the k-nearest neighbors of x_i and x_j [41]. Besides these heuristic methods, several studies expect to learn the optimal parameter σ from data. We will give a detailed discussion in Section 3.2.2.

3) *Inverse of distance*. Similar to Gaussian kernel, the inverse distance $d^{-1}(x_i, x_j)$ provides an alternative for converting dissimilarity measures to similarity measures, and, in principle, $d(\cdot, \cdot)$ can indicate any dissimilarity included in, but not limited to, those defined in Table 2. For example, the inverse Euclidean distance is used in [42] to calculate edge weights of a graph. Comparing with Gaussian kernel, the inverse distance is *parameter-free*, and thus avoids the difficulty of parameter selection. In contrast, a carefully selected scale parameter in Gaussian kernel can provide more *flexibility* to describe the local geometry in data.

The two most popular schemes in (2) and (3) assign edge weights based on *dissimilarity-to-similarity* transfer functions, which is widely used in the machine learning field. On the other hand, the similarity measures (*e.g.*, those in Table 2) can also be used directly to compute edge weights. In the following cases (4) and (5), we review two kinds of weighting strategies directly based on similarity measures.

4) *Full correlation*. As shown in Table 2, Pearson's correlation coefficient is defined as the covariance of two variables, x_i and x_j, with normalization by their standard deviations. It is currently the most popular scheme employed in the functional brain network construction [30], where the node (or data) x_i can be sampled from the signal associated with the neural activity of the ith brain region. Pearson's correlation takes a value between -1 and $+1$, where the sign shows different correlation tendencies. In order to distinguish it from the partial correlation that will be discussed in the next case, we refer to this as *full correlation*. Despite its simplicity, Pearson's correlation has been successfully applied to many graph construction problems, such as functional brain network and gene co-expression network modeling. Recently, it was reported that the correlation-based approaches tend to provide relatively high sensitivity to functional brain graph detection, compared with the methods based on *higher-order* statistics [43]. In practice, however, Pearson's correlation can only detect the linear relationship between variables (nodes).

Alternatively, Spearman's rank correlation coefficient [44] is another form of full correlation. From the mathematical expression of the Spearman's correlation given in Table 2, we note that it is essentially the Pearson's correlation of the rank vectors whose elements are the ordinal numbers relative to the original vector. Different from the original Pearson's correlation, Spearman's correlation is nonparametric, and can detect nonlinear relationships among the nodes. Further, Spearman's correlation is in general less sensitive to strong outliers, since it coverts these outliers to the corresponding rank values before computing correlation. Also, there are other rank correlations, such as Kendall tau correlation [45], can be used in calculating the edge weights of a graph.

5) *Partial correlation*. As mentioned previously, Pearson's correlation is the most popular approach for measuring the dependency between nodes. However, it fails to remove the confounding effect from other nodes. Here, we use a case often involved in Bayesian networks [46] to illustrate this problem. For simplicity, we consider a graph with three nodes. The nodes have a "dependency", like $x_i \rightarrow x_k \leftarrow x_j$ or $x_i \leftarrow x_k \rightarrow x_j$, where "$\rightarrow$" denotes a directed edge. Due to the influence of the x_k, Pearson's correlation will result in a fully-connected graph, even though x_i and x_j have no essential relation. In contrast, partial correlation measures the dependency between two nodes by regressing out the remaining ones. That is, the edge weight W_{ij} provides a conditional correlation between x_i and x_j given other nodes. There are several approaches to compute the partial correlation. We suggest using the inverse covariance matrix estimation [47], since it gives all the pairwise partial correlations in a batching mode. We will give more detailed discussion on this topic in Section 3.2.5.

Besides the ways discussed in (1)–(5), in practice one can employ other measures for defining edge weights. For example, recently mutual information has been applied to define edge weights for modelling brain and gene networks [48]. Additionally, some post-processing to the edge weights, *e.g.*, normalization, might improve the performance by alleviating the over-domination of some nodes with big degrees. However, different from the direct definition, most graph construction methods aim to *learn the optimal edge weights from data*. In the following Sections 3.2.2–3.2.8, we review several kinds of edge weight learning methods involved in different fields. For consistency with the traditional expression, we call these methods *graph learning*, although it in fact conducts "edge weight learning" .

3.2.2. Parametric graph learning

In the machine learning community, Gaussian kernel $\exp(-\|x_i - x_j\|^2/2\sigma^2)$ is the most commonly used strategy to compute the edge weights of a graph, where the scale parameter σ often affects the ultimate performance significantly. Therefore, some of the early studies for graph construction mainly focus on how to select suitable value for this scale parameter. In Section 3.2.1, we briefly review several works on *empirical or heuristic* parameter assignment for Gaussian kernel. Here, we discuss the studies that aim to learn the optimal value for parameter σ from data.

In [49], Zhu et al. assumed different scales are associated with different feature dimensions, and proposed to learn a parameter set $\{\sigma_k\}_{k=1}^d$ by minimizing the average label entropy on unlabeled data. With a similar assumption, authors in [50] *instead* optimized the parameter set by minimizing the leave-one-out error based on the labeled data. In [51], the authors formulated the hyper-parameter learning problem based on a Bayesian framework and Expectation Maximization algorithm. Different from these three works that focused on (semi-)supervised learning, a recent work [52] conducted *learning-task independent* graph learning in a parametric form. In particular, the parametric model, called Adaptive Edge Weighting (AEW) [52], is given as follows.

$$\min_{\{\sigma_k\}_{k=1}^d} \sum_{i=1}^n \left\| x_i - \frac{1}{d_i} \sum_{j=1}^n E_{ij} W_{ij} x_j \right\|^2 \tag{3}$$

where $W_{ij} = \exp(-\sum_k d(x_{ik}, x_{jk})^2 / 2\sigma_k^2)$ is Gaussian kernel with different scale parameters for different features; $d_i = \sum_j E_{ij} W_{ij}$ is the row degree of the node x_i, and E_{ij} is the graph topology predefined in E-step. Essentially, the objective function in Eq. (3) measures the distance between each node x_i and the weighted average of its "neighbors". In [52], the authors solved Eq. (3) based on a gradient descent algorithm, and provided some interesting properties (see Section 4 in [52] for details).

3.2.3. Locality-inducing graph learning

1) *Local linear reconstruction (LLR).* LLR is originally developed for manifold learning [20]. It learns the edge weights based on a given graph structure E (which is pre-determined in E-step by, for example, kNN as in [20]) through solving the following least square problem:

$$\min_W \sum_{i=1}^n \left\| x_i - \sum_{j=1}^n E_{ij} W_{ij} x_j \right\|^2$$
$$\text{s.t.} \sum_{j=1}^n E_{ij} W_{ij} = 1 \tag{4}$$

Different from the AEW method discussed above, LLR is non-parametric, and simply constrains the row degree $d_i = \sum_{j=1}^n E_{ij} W_{ij} = 1$. Note that minimizing the sum form in Eq. (4) is equivalent to minimizing every component separately due to the non-negativity of the components. However, the least square solution is generally noise-sensitive, and tends to over-fit the data due to insufficient samples in each neighborhood. Thus, a regularized LLR is preferable in practice [53]. In addition, an interesting study recently attempted to derive graphs from the local manifold learning (including LLR-based method), and successfully used the strategy in the hyper-spectral image classification [54].

2) *Non-negative local linear reconstruction.* Eq. (4) above may generate negative edge weights, and lead to the failure of some algorithms such as label propagation [55]. Thus, a non-negative variant was developed in [56] as follows.

$$\min_W \sum_{i=1}^n \left\| x_i - \sum_{j=1}^n E_{ij} W_{ij} x_j \right\|^2$$
$$\text{s.t.} \sum_{j=1}^n E_{ij} W_{ij} = 1,\ W_{ij} \geq 0 \tag{5}$$

Such an optimization problem can be solved by the standard quadratic programming.

3) *Locality preserving graph construction (LPGC).* Recently, Zhang et al. proposed to construct a graph based on the locality preserving principle [57]. Although we classify it into the local methods, LPGC does not use the locality constraint explicitly encoded in the graph structure E. In contrast, LPGC achieves the locality for W by a locality-preserving term, i.e., the first term shown in the following,

$$\min_{W \geq 0} \sum_{i,j=1}^n \|x_i - x_j\|^2 W_{ij} + \lambda_1 \sum_{i=1}^n (d_i - 1)^2 + \lambda_2 \sum_{i,j=1}^n W_{ij}^2 \tag{6}$$

In other words, the first term expects that the edge weights change smoothly according to the local geometry of the data. The second term plays a similar role to the degree constraint, for avoiding a degenerated solution as in Eqs. (3)–(5). The third term aims to prevent some edges dominating the degree of a certain node. Based on a symmetric assumption on W, Eq. (6) can be reformulated as a quadratic programming, and a cutting plane algorithm can then be used to solve such a problem [57].

3.2.4. Graph learning based on global reconstruction

In [58], Daitch et al. conducted graph learning by *Fitting a Graph to Vector data (FGV)*, as follows.

$$\min_W \sum_{i=1}^n \left\| d_i x_i - \sum_{j \neq i} W_{ij} x_j \right\| 2 \tag{7}$$

Since Eq. (7) has a trivial solution (namely, $W = 0$), Daitch et al. introduced two optional constraints, (1) $d_i \geq 1$, and (2) $\sum_i (\max(0, 1 - d_i))^2 \leq \eta n$, for handling this issue. According to Daitch et al. [58], with any of these two constraints, Eq. (7) can be formulated as a convex quadratic programming. Note that this approach has a similar model to LLR. However, the weights in LLR are computed in a predefined local neighborhood, thus resulting in a difficulty of locality parameter selection. In contrast, Daitch et al.'s method does not necessarily work locally. Even so, FGV can interestingly achieve a *sparse* solution with at most $2(d+1)n$ edges (see Theorem 3.1 in [58]). Of course, the sparse solution may be non-local. That is, some edges may be assigned between the nodes far away from each other, which was illustrated by a toy example in a recent work [57].

3.2.5. Sparsity-inducing graph learning

Sparsity is a ubiquitous characteristic, existing in physical, biological, and social systems. Especially for graphs, sparsity plays an important role in providing robustness, efficiency, and interpretability.

1) *L_1graph.* In recent years, several methods were independently proposed to construct graphs based on sparse representation. According to the terminology used in [59], we call them *L_1-graphs* because they, in general, conduct graph construction by solving an L_1 norm-regularized problem. These methods share similar graph learning models, yet for different purposes. For example, Elhamifar and Vidal construct L_1 graph for subspace clustering [60]; Yan and Wang aim at semi-supervised learning [61]; Cheng et al. design a sparsity induced similarity measure also for semi-supervised learning [62]; Qiao et al. developed the Sparsity Preserving Projection algorithm to preserve the L_1 graph in a low-dimensional space for dimensionality reduction [63]. Compared with the popular kNN graphs, the L_1 graphs are generally more suitable for modeling the data derived from a multi-subspace structure [60], and have greater robustness with adaptive "neighborhood" size for different nodes [62].

Besides the application backgrounds, there are also some differences among several seminal works [60–63] for constructing L_1 graphs. For example, our model [63] introduced a sum-to-one constraint on each row of the edge weight W to obtain the translation

invariance for data embedding. However, the core of these works is based on the following mathematical model.

$$\min_{W} \sum_{i=1}^{n} \left(\left\| x_i - \sum_{j \neq i} W_{ij} x_j \right\|^2 + \lambda \sum_{j \neq i} |W_{ij}| \right) \tag{8}$$

Clearly, we can get its optimal solution when all the n components in the sum expression reach their minimum points. As a result, we only need to conduct each sub-problem, respectively, which is exactly a traditional L_1-minimization problem. Recently, various software packages, such as Liu et al.'s SLEP[1] have been developed to solve the L_1 regularized (or equivalently L_1 constrained) problems. For convenience of comparison with the models of low-rank graph learning in the next section, we recast Eq. (8) as the following matrix form.

$$\begin{aligned} &\min_{W} \|X - XW\|_F^2 + \lambda \|W\|_1 \\ &\text{s.t.} \quad W_{ii} = 0, \ \forall i = 1, \ \ldots, \ n \end{aligned} \tag{9}$$

That is, we can treat the L_1-graph learning problem in a batch processing way. Such a formulation can be easily solved by, for example, the SLEP toolbox with slight modification.

Due to its simplicity and empirical success (especially for the data from the multi-subspace structure), many researchers extend the L_1-graph based on different considerations. To name a few, He et al. [64] presented a non-negative sparse presentation algorithm for semi-supervised learning, and Gui et al. [65] proposed a supervised version of the L_1-graph based on the sparsity preserving projections [63].

2) *Sparse inverse covariance estimation (SICE).* Inverse covariance matrix (a.k.a., precision matrix or concentration matrix) estimation is a hot research topic in the past few years, *not only* in statistics and machine learning fields [47, 66], *but also* in some specific applications [67, 68]. It is well known that the entries in the inverse covariance matrix can induce the *partial correlation* among variables (see Table 2 in Section 2 for the corresponding formulas). Thus, the inverse covariance matrix estimation *in fact* provides a scheme for constructing a graph with its edge weight W_{ij} describing the partial correlation (or conditional dependency) between the nodes (variables, here) x_i and x_j.

However, it is an ill-posed problem to estimate the partial correlation when the sample size is smaller than the variable dimensionality. To address this problem, a sparsity-oriented L_1 regularizer is generally introduced, which results in the so-called Sparse Inverse Covariance Estimation (SICE). The most widely used method is based on the penalized *maximum likelihood* estimation, and the corresponding model is shown as follows [47],

$$\min_{W \succcurlyeq 0} \operatorname{tr}(SW) - \log(|W|) + \lambda \|W\|_1 \tag{10}$$

where S is the sample covariance matrix, and $W \succcurlyeq 0$ means a positive semi-definite constraint on W. In recent years, many extended works have been proposed to improve the problem described in Eq. (10). For example, in [69] the authors constructed functional brain networks by incorporating a population prior which constrains all subjects sharing the same graph topology E.

Besides the *maximum likelihood* strategy, the sparse inverse covariance matrix can be estimated by *linear regression* models like LASSO [66], which shares a similar expression to Eq. (9) used for constructing an L_1-graph, yet with a different motivation. More specifically, the L_1-graph models the relations between samples, while the SICE (or approximately, sparse partial correlation estimation) encodes the relations between variables (or features). Currently, more powerful models have been developed to extend the traditional LASSO for estimating partial correlations. For example, in [68] Peng et al. conducted a partial correlation estimation by a coupling sparse regression model, and used the proposed method for genetic regulatory network construction; Lee et al. modeled partial correlation for constructing functional brain graphs in the view of compressive sensing [70]; more recently, Wee et al. [71] developed a multi-task version of LASSO to estimate functional brain networks by incorporating a population prior as in [69].

3.2.6. Low-rank graph learning

Unlike sparsity-inducing graph learning that aims to compute the sparsest reconstructive coefficients for each node individually, low-rank graph learning aims to find the *jointly* lowest-rank representation of the whole node set. Consequently, the low-rank methods are expected to better capture the *global structure* of the data [72]. In theory and practice, the effectiveness of the low-rank property has been verified, especially for matrix complement and robust subspace recovery [73]. Here, we review the main low-rank graph learning models chronologically.

1) *Low-rank graph learning.* In [72] and its extended version [73], Liu et al. proposed a robust subspace segmentation algorithm based on low-rank graph learning. The model is defined as follows.

$$\min_{W} \|X - AW\|_{2,1} + \lambda \|W\|_* \tag{11}$$

where the $L_{2,1}$ norm in the first term implicitly assumes that the reconstructive errors (or noises) are node-specific, i.e., only partial node vectors are corrupted. The trace norm (or nuclear norm) in the second term is a convex relaxation of the rank(W). The matrix A is a "dictionary" for representing the data X. Since the goal here is graph learning, one can simply set $A = X$ as Eq. (9) used for constructing L_1-graphs. Similar to sparse representation, many efficient algorithms have been developed to solve the low-rank problem. For example, the SLEP [74] include a solver for low-rank representation.

2) *(Non-negative) Low-rank and sparse graph learning.* Sparse representation tends to capture the local structure, while the low-rank representation to capture the global structure [72]. By considering both the local and global properties in data, Zhuang et al. conducted graph learning by combining the low-rank with sparse constraints in a single objective function [75]. The model is shown below:

$$\min_{W} \|X - XW\|_{2,1} + \lambda_1 \|W\|_* + \lambda_2 \|W\|_1 \tag{12}$$

which was solved by an inexact augmented Lagrange multiplier method [75]. Since the optimal edge weight W_{ij}^* may be negative, a post-processing, $(|W_{ij}^*| + |W_{ji}^*|)/2$, is required in order to conduct the subsequent semi-supervised learning algorithm. In contrast, the authors introduced a *non-negative* constraint directly into the optimization problem in their other paper [76], and gave rise to a non-negative low-rank and sparse graph learning mode.

3) *Non-negative low-rank and positive semidefinite graph learning.* Unlike the low-rank graph learning models mentioned above, which construct graphs based on original data (i.e., node set X), Luo et al. proposed to learn a low-rank graph "around" a prespecific one [77]. Meanwhile, the authors pointed out that an "ideal" graph should be non-negative, symmetric, low-rank, and positive semi-definite, which results in the following model.

$$\min_{W} \left\| W - W^0 \right\|_F^2 + \lambda \|W\|_*$$

[1] http://yelab.net/software/SLEP/

$$\text{s.t. } W \succcurlyeq 0,\ W \geq 0,\ W = W^T \tag{13}$$

where W^0 is a pre-defined graph, and the symbols, "$\succcurlyeq$" and "$\geq$", denote positive semi-definition and non-negativity, respectively. A variant of the augmented Lagrangian multiplier was used to solve this problem, and the effectiveness was verified under the scenario of semi-supervised learning.

4) *Low-rank graph learning with local constraint*. In [78], a local regularizer was incorporated into the typical low-rank graph learning, and the resulted model is shown as follows.

$$\min_W \|X - XW\|_{2,1} + \lambda_1 \|W\|_* + \lambda_2 \mathrm{tr}\left(WLW^T\right) \tag{14}$$

where L in the last term denotes a neighbor graph Laplacian, by which the local regularizer $\mathrm{tr}(WLW^T)$ provides a "constraint" that near node pairs (x_i, x_j) tend to have similar representation coefficients/weights W_{ij}. Such a constructed graph was used in hyperspectral image destriping in [78]. Almost at the same time, another local low-rank model [79] was proposed by introducing a different regularizer, $\sum_{i,j} \|x_i - x_j\|^2 |W_{ij}|$, into the original low-rank model in Eq. (11). Such a regularizer is essentially a generalized L_1-norm with a locality-sensitive weight, $\|x_i - x_j\|^2$. Also, a more recent work [80] used a similar local regularizer for low-rank graph learning, yet with deeper theoretical analysis.

5) *Low-rank Markov random walk for graph learning*. In [81], the authors proposed to learn low-rank graphs under a regularized Markov random walk (MRW) framework. In particular, the graph W is considered as a probabilistic transition matrix that needs to be non-negative, symmetric and with sum-to-one rows, as shown in the constraints of the following model.

$$\begin{aligned} &\min_W \sum_{i,j=1}^{n} \left\|x_i - x_j\right\|^2 W_{ij} + \lambda \|W\|_* \\ &\text{s.t.} \sum_{j=1}^{n} W_{ij} = 1,\ W \geq 0,\ W = W^T \end{aligned} \tag{15}$$

Note that the first term in the objective function is based on the locality preserving principle, which has been used in many graph learning methods, such as LPGC in Section 3.2.3 and GoLP in Section 3.2.8, aiming to model the local geometry of data and make the edge weights of the graph change smoothly along the data. In [81], the authors proved that such a model can recover the edge weight matrix with a block-diagonal structure, which is very important to subspace clustering and estimation. In fact, the block-diagonal structure of a graph can be modeled by several different approaches under mild conditions. See [72, 73, 82, 83] for more theoretical results.

6) *Low-rank graph learning with b-matching constraint.* Motivated by the advantages of low-rank representation and b-matching, Li and Fu presented a joint optimization algorithm to learn both low-rank and balanced graph simultaneously in a single model [84, 85].

$$\begin{aligned} &\min_{Z,\ W} \|X - AZ\|_0 + \lambda_1 \mathrm{rank}(Z) - \lambda_2 \sum_{i,j=1}^{n} W_{ij} \left(Z^T Z\right)_{ij} \\ &\text{s.t.} \sum_{j=1}^{n} W_{ij} = b,\ W = W^T,\quad 1^T Z = 1^T \end{aligned} \tag{16}$$

where A is a dictionary for encoding the data X in the low rank matrix Z, and 1 is a vector with all ones. Besides the combination with the "b-matching", there are several differences between this method and the other low-rank methods discussed above. (1) The previous works directly use the low-rank reconstructive coefficients as the edge weights of a graph. In contrast, this model re-learns the edge weight matrix W in a new space spanned by the columns of the low-rank encoding matrix Z. (2) Instead of using the L_1 norm and trace norm, this method employs a new trick [86] to relax the zero "norm" and rank operator, respectively. Then, Eq. (16) can be solved by the alternating optimization between Z and W. In fact, with a fixed Z, Eq. (16) reduces to a b-matching problem, thus yielding a balanced graph. It is worth pointing out that W here is a binary matrix, and plays the same role as the graph topology E in E-step. Thus, Eq. (16) essentially learns the graph topology instead of edge weights. We put the model in this section (for W-step) due to two considerations: i) the model shares a similar form to the other low-rank models in this section, and ii) it can be naturally extended for edge weight learning.

Besides the works reviewed in this section, some improved low-rank methods have been developed to learn high-quality graphs. The work [87] combined the low-rank graph learning with a manifold fitted graph [88] for semi-supervised learning. We do not plan to review the details of these methods one by one, but, as a summary of this subsection, suggest that it may be a potentially effective way to *combine* the advantages from different graph construction methods for modeling complex data distributions. In Section 3.2.7 below, we will review another kind of combination scheme, namely multi-graph learning.

3.2.7. Multi-graph learning

In practice, data may derive from different domains, representations, views, or modalities. For example, functional brain networks can be constructed from the fMRI and PET signals, respectively. A natural issue is figuring out how to combine these graphs for approximating the "ground truth" or improving the subsequent learning tasks. The related techniques are generally called multi-graph learning. Note that, however, in some literatures [89, 90], the term "multi-graph learning" is used to indicate the methods that aim at classifying multiple graphs, which goes beyond the focus of this paper.

Early works, to our best knowledge, in this topic can be found in the papers [91] and [92], both of which linearly combined multiple graphs in the form of the graph Laplacian. For example, [92] fused multiple graphs based on the linear combination, i.e., $W = \sum_k \alpha_k W^{(k)}$, with $\sum_k \alpha_k = 1,\ \alpha_k \geq 0$, and then the graph learning problem is converted to find combination coefficients $\{\alpha_k\}_{k=1}^m$. Without loss of generality, we put them into a coefficient vector $\alpha = [\alpha_1,\ \alpha_2,\ \ldots\cdots, \alpha_m]$. To optimize these coefficients, many methods have been developed according to different applications, but most of them share a similar objective function. Thus, we only give several representative examples, and simplify the objective function by ignoring the specific type of loss functions for clearer comparison and explanation.

1) Unified Video Annotation via Multi-graph Learning [93]. A simplified model is

$$\begin{aligned} &\min_{\alpha, f} \sum_{i=1}^{l} L(y_i, f_i) + \lambda \sum_{k=1}^{m} \alpha_k^r \left(\sum_{i,j=1}^{n} \left(f_i - f_j\right)^2 W_{ij}^{(k)} \right) \\ &\text{s.t.} \sum_{k=1}^{m} \alpha_k = 1,\ \alpha_k \geq \end{aligned} \tag{17}$$

where $L:\ R \times R \to [0, +\infty]$ is a loss function. In this work, the authors used it to measure the differences between the predicted label f_i and the ground truth y_i based on the l labeled samples. It is well-known that different loss functions will lead to different models and optimization algorithms. However, we do not care about the specific form of the loss function here since we mainly focus on graph learning. For multi-graph learning, the central part of Eq. (17) is the second term, which constrains the predicted labels changing smoothly along the combination of multiple graphs,

and, in fact, such a term can naturally be converted to a form of *combined graph Laplacian*, $\sum_k \alpha_k f^T L_k f$, where $f = [f_1, f_2, \ldots, f_n]$ is the predicted label vector and L_k is the kth graph Laplacian. Note that, the parameter r in the power of α_k is a constant greater than 1, for avoiding a trivial solution. A more recent work [94] conducted multi-graph learning based on the same model as Eq. (17), but for a different application of affective image retrieval.

2) *Multiple graph label propagation by sparse integration* [95]. In this work, the related multi-graph learning model was given as follows.

$$\min_{\alpha,f} \sum L(y_i, f_i) + \lambda_1 \sum_k \alpha_k \left(\sum_{i,j=1}^{n} \left(f_i - f_j\right)^2 W_{ij}^{(k)} \right) + \lambda_2 \|\alpha\|^2$$
$$\text{s.t.} \sum_{k=1}^{m} \alpha_k = 1, \ \alpha_k \geq 0 \tag{18}$$

Similarly, the first term in Eq. (18) is the loss function, and the second term aims to constrain the label changing smoothly along multiple graphs. The last term $\|\alpha\|^2$ plays the same role as the r in Eq. (17) for *non-linearizing* the combination coefficient vector $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_m]$, since the pure linear "constraint" on it will lead to a trivial solution. Based on such a regularizer, the authors proved that a sparse solution of α can be achieved by assigning suitable values for the hyper-parameter λ_2. Recently, Gao et al. applied a similar multi-graph learning model to medical image retrieval [96], and Yang et al. employed it for robust visual tracking [97].

3) *Multimodal graph-based re-ranking for web image search* [98]. Compared with two previous works, we rewrite its model as follows,

$$\min_{\alpha,f,P_k} \sum L(y_i, f_i) + \lambda_1 \sum_k \alpha_k \left(\sum_{i,j=1}^{n} \left(f_i - f_j\right)^2 W_{ij}^{(k)} \right) + \lambda_2 \|\alpha\|^2$$
$$\text{s.t.} \sum_{k=1}^{m} \alpha_k = 1, \ \alpha_k \geq 0 \tag{19}$$

Despite sharing the similar expression with Eq. (18), the model here assumes that the edge weight matrix has a *parametric form*, $W_{ij}^{(k)} = \exp(-\|P_k^T x_i - P_k^T x_j\|^2)$, which is computed in a new space spanned by the projection matrix P_k. As a result, Eq. (19) *not only* aims to achieve an optimal graph combination coefficient α, *but also* expects to learn a group of optimal data projections $\{P_k\}_{k=1}^m$ for defining "better" graphs. In Section 3.2.8, we will give a more detailed review on graph learning in *transformed spaces*.

Besides the linear combination, there are two other strategies, matrix factorization [99] and co-diffusion [100], for fusing multiple graphs. In (4) and (5) below, we review these works, respectively.

4) *Multi-graph learning via matrix factorization.* In [99], Tang et al. proposed a novel method, namely Linked Matrix Factorization (LMF), to fuse information from multiple graphs. LMF assumes that each edge weight matrix can be decomposed into a graph-specific factor and a common factor shared by multiple graphs, and the model is given as follows.

$$\min_{Q,\Lambda} \sum_{k=1}^{m} \left\| W^{(k)} - Q\Lambda^{(k)}Q^T \right\|_F^2 + \lambda \left(\sum_{k=1}^{m} \left\| \Lambda^{(k)} \right\|_F^2 + \|Q\|_F^2 \right) \tag{20}$$

where Q is the common factor shared by all graphs, and $\Lambda^{(k)}$ is the specific factor corresponding to the kth graph. Two regularizers in Eq. (20) are introduced to improve the numerical stability and alleviate the overfitting problem. According to [99], the shared factor Q is regarded as a fused "graph" which extracts the common information among multiple sources and filters out the irrelevant information or noise.

5) *Multi-graph fusion via co-diffusion*. In [100], Wang et al. proposed to fuse multiple similarity matrices (graphs) based on a diffusion process. First, a basic co-diffusion process is given as follows.

$$W_{t+1}^{(1)} = S^{(1)} W_t^{(2)} \left(S^{(1)}\right)^T$$
$$W_{t+1}^{(2)} = S^{(2)} W_t^{(1)} \left(S^{(2)}\right)^T \tag{21}$$

where the initial $W_0^{(k)}$ is a fully connected similarity matrix (via normalization) corresponding to the edge weight of a graph, and $S^{(k)}$ is a sparsified counterpart (via normalization) of the corresponding edge weight matrix. After t iterations, the finally fused matrix is given by $(W_t^{(1)} + W_t^{(2)})/2$. Then, the authors proposed a multi-graph fusion strategy based on the two-graph co-diffusion, and used it in metric learning for shape image retrieval [100]. More recently, Wang et al. further extend their algorithm to fuse multiple similarity networks involved in biomedical data analysis [101].

As discussed in this section, multi-graph learning provides a natural way to fuse multiple information sources. Also, in our view, the multi-graph learning in fact *provides an alternative to treat the parameter selection problem* involved in many existing graph construction/learning strategies. In particular, we can construct multiple graphs based on different parameters (*e.g.*, different "neighbor" size in proximity graph construction or regularized parameters in sparse/low-rank graph learning), and then learn a set of optimal combination coefficients for obtaining the final graph. We can even build multiple graphs using different similarities, hyper-parameters or algorithms, and consider graph fusion based on multi-graph learning.

3.2.8. Graph learning in an adaptively transformed space

Thus far, most of the reviewed graph construction/learning methods can only work on the original data space. In contrast, a certain transformed space may be low-noisy and more informative. For example, (i) the transform based on principal component analysis [102] tends to reduce Gaussian noises, and (ii) the face images mapped into an LBP [103] or Gabor wavelet space [104] can generally produce more discriminating features. However, it is difficult to find a good mapping for a general graph learning problem without guidance from a specific application background. Recently, researchers proposed to learn the mapping and graph simultaneously in a single objective function, and empirically verify its effectiveness on some publicly available data sets.

More specifically, Zhang et al. developed a graph-optimized locality preserving (GoLP) model in [10] as follows.

$$\min_{P,W} \sum_{i,j=1}^{n} \left\| Px_i - Px_j \right\|^2 W_{ij} + \lambda \sum_{i,j=1}^{n} W_{ij} \ln W_{ij}$$
$$\text{s.t.} \sum_{i=1}^{n} \|Px_i\|^2 = 1, \sum_{j=1}^{n} W_{ij} = 1, \ W_{ij} \geq 0 \tag{22}$$

where P is a projection matrix mapping each node x_i into a new space, in which a locality preserving principle is used to guide the graph learning. That is, the closer are the node pairs (Px_i, Px_j), the bigger is the weight W_{ij}, and vice versa. The second term in the objective function is the negative entropy, which is minimized to avoid a trivial solution and make the weights change as smooth as possible along the data. Note that $\|Px_i - Px_j\|^2 = (x_i - x_j)^T P^T P (x_i - x_j)$, and, thus, the effect of the projection is equivalent to learning a Mahalanobis distance in an unsupervised form. According to Zhang et al. [10], Eq. (22) can be solved by Alternating Optimization (AO) between P and W. Since the finally obtained graph is not sparse in this locality preserving model, the authors further presented a transform-guided L_1-graph learning

method [105], as follows.

$$\min_{P,W}\sum_{i=1}^{n}\|P(x_i - XW_i)\|^2 + \lambda\sum_{i=1}^{n}\|W_i\|_1$$
$$\text{s.t.}\sum_{i=1}^{n}\|Px_i\|^2 = 1,\ PP^T = I \tag{23}$$

Despite their empirical effectiveness on some data sets, the graph in transformed space can sometimes get worse with each iteration, and, thus, a recent work [106] attempted to learn a graph by fusing the information from both the original and transformed space. Unfortunately, such a model incurs an intractable parameter selection problem [106]. In fact, as discussed previously, selecting an optimal hyper-parameter is currently an open problem for unsupervised graph construction or learning.

Besides, Xie et al. proposed to learn a graph in a transformed space [107]. However, they directly parameterized edge weight $W_{ij} = \exp(-\|Px_i - Px_j\|^2)$, and optimized the projection matrix P with pairwise must-link and cannot-link constraints. Therefore, this method is more similar to dimensionality reduction or distance metric learning. In Section 5, we will further discuss the relationship between graph learning and these two topics.

As a summary of this section, we discuss how to select a suitable graph construction/learning strategy in practice. In general, however, this is a challenging problem without a universal solution, since it depends on many factors, including the data distribution, sample size, domain knowledge, and even empirical "trial and error". Here, we just provide several suggestions.

1) For a general graph-based learning problem, a preferred graph construction scheme is the classical "kNN + Gaussian kernel" (Sections 3.1 and 3.2.1). A recent empirical study [11], under the semi-supervised scenario, also supports this point. On the other hand, for some specific applications (e.g., functional brain network modeling), the "Pearson's correlation $+\varepsilon$-N" (Sections 3.1 and 3.2.1) is the most popular strategy with more sensitivity to informative pattern than higher-order statistics [43].
2) For data with manifold structure, the "locality-inducing" methods (Section 3.2.3) are often suggested to capture local geometry in the data. Of course, all the proximity graphs discussed in Section 3.1 can be employed to model such local structures with different granularities. In contrast, the "sparse or low-rank" methods (Sections 3.2.2 and 3.2.6) have been empirically and theoretically verified to be effective for capturing the subspace or multi-subspace structure in the data [60, 73, 82].
3) With sufficient samples, we can learn a graph relatively easily in a parametric (Section 3.2.2) or non-parametric way. However, it is difficult to capture local geometry based on limited samples, and, thus, in this case we suggest using the global reconstruction method (Section 3.2.4) to learn a graph, since it can make full use of the data and does not involve any free parameter. Additionally, side label information can help address the small-sample-size problem in graph learning [108]. In practice, the label information can be naturally encoded in a graph by, for example, converting these information into pairwise must-/cannot-link constraints [109].
4) If the data are derived from different sources (modalities, or views), multi-graph learning (Section 3.2.7) is a potential choice for fusing the information from multiple sources in a single graph. In addition, multi-graph learning provides a way for model selection by combining graphs constructed using different methods or parameters.
5) Usually, the original data are noisy and lie in a high-dimensional space. In this case, the methods reviewed in Section 3.2.8 can be used to learn a graph in an adaptively transformed space. Of course, we cannot conclude that these algorithms always work well even after transformation. So are the other graph construction/learning methods. Therefore, sometimes one needs to test different methods by trial and error. This is one of our main motivations to summarize as many methods as possible in this paper.

4. Matrix-regularized graph learning framework

Improving generalization is the central goal of machine learning [110], and generalization usually depends on "data + knowledge" [111]. Here, the "knowledge" generally denotes a prior or assumption that *in practice* can be modeled by a regularizer. In this section, we unify some existing graph learning algorithms into a matrix-regularized framework, thus giving a new explanation to some graph construction models (*e.g.*, graphs with edge weights of Gaussian kernel) and also providing a platform to design new graph learning algorithms. Similar to the traditional regularized frameworks widely used in machine learning, the *matrix-regularized graph learning framework* can be described as follows.

$$\min_{W} F(X, W) + \lambda\Omega(W)$$
$$\text{s.t. } W \in \Delta \tag{24}$$

where $F(X, W)$ is a *data fitting term*, $\Omega(W)$ is a *matrix-regularized term,* and Δ is a set of constraints on the graph. Although $F(X, W)$ plays a role of "data fitting" that is similar to the "loss function" used in some classification/regression models, this term in the above equation has a different meaning. In particular, loss functions usually measure the distance between predicted values and true values, while $F(X, W)$ measures **w***hich aspects of the data a graph aim to capture.* For example, minimizing $F(X, W) = \sum_{i,j=1}^{n}\|x_i - x_j\|^2 W_{ij}$ means to capture the local structure of the data, while minimizing $F(X, W) = \|X - XW\|_F^2$ means to capture the "globally" reconstructive relationship.

As we know, regularizers are generally used to control the complexity of the models in the machine learning field. Without regularizers, models may over-fit the data, especially in the case of the small sample size. In Eq. (24), the matrix-regularized term $\Omega(W)$ is included for the similar purpose. In addition, $\Omega(W)$ provides a way to encode priors or assumptions into the graph. For example, the L_1 matrix-norm $\Omega(W) = \|W\|_1$ can be employed to model functional brain networks, since the brain has been verified to be sparsely connected.

Sometimes, priors for graphs may not naturally be expressed in the form of regularizers [112]. Therefore, an extra set Δ is introduced in Eq. (24) to include constraints imposed on W. For example, we can assume that a functional brain network is positive semi-definite (or, at least, symmetric), since the underlying graph is undirected and generally used to capture the (inverse) covariance structure in data. Another example is the label propogation algorithm which needs to constrain the W to be non-negative. As a result, the matrix-regularized framework, Eq. (24) covers most of the existing graph learning algorithms reviewed in Sections 3.2. In Table 3, we summarize the main methods for instantiating the general regularized graph learning framework.

More interestingly, *some traditional edge weight assignment formulas widely used in graph construction can also be derived from such a framework.*

Example 1. Let the data-fitting term $F(X, W)$ be $\sum_{i,j=1}^{n}\|x_i - x_j\|^2 W_{ij}$ for preserving the locality of data, and $\Omega(W)$ be $\sum_{i,j=1}^{n} W_{ij}\ln(W_{ij}/e)$, a negative entropy-liked regularizer, for keeping the weight changing as uniformly as possible, where e is the natural constant only for the purpose of "normalization". Then, by simply setting the derivative w.r.t. W_{ij} to be zero, we can derive the optimal solution of Eq. (24) as $W_{ij}^* = \exp(-\|x_i - x_j\|^2/\lambda)$,

Table 3
Instantiation of the matrix-regularized graph learning framework.

	Expressions	Explanation	Related works
Data-fitting term	$\sum_{i=1}^{n} \|x_i - \sum_{j=1}^{n} E_{ij} W_{ij} x_j\|^2$	Local representation[a]	[20,53,56]
	$\sum_{i,j=1}^{n} \|x_i - x_j\|^2 W_{ij}$	Locality preserving	[57,81]
	$\sum_{i=1}^{n} \|x_i - \sum_{j\neq i} W_{ij} x_j\|^2$ or $\|X - XW\|_F^2$	Global representation	[60–63,68,70,71]
	$\sum_{i=1}^{n} \|d_i x_i - \sum_{j\neq i} W_{ij} x_j\|^2$	Weighted global representation	[58]
	$\mathrm{tr}(SW) - \log(\|W\|)$	Graph as an inverse covariance matrix	[47,66-69]
	$\|X - XW\|_{2,1}$	Global representation with node-specific noises	[72,73,75,78–80,83]
	$\|W - W^0\|_F^2$	Learning a graph near a pre-specific one	[77]
Regularized term	$\|W\|_F^2$	F-norm for stability of the solution	[57,82,83]
	$\|W\|_1$	L_1-norm for sparsity	[47,60-63,66-68,70]
	$\|W\|_{2,1}$	$L_{2,1}$-norm for group sparsity	[69] [71]
	$\|W\|_*$	Trace norm for low-rank solution	[72,73,75-78,80,81]
	$\|W\|_1 + \lambda\|W\|_*$	Combining L1- and trace norms for block sparsity	[75,76]
	$\mathrm{tr}(WLW^T)$	Data-dependent regularizerb	[78]
	$\sum_{i,j} u_{ij}\|W_{ij}\|$	Weighted L1-norm	[79,80]
	$\sum_{i,j=1}^{n} W_{ij} \ln W_{ij}$	Entropy for smooth solution	[10]
Constraint term	$W \geq 0$	Non-negativity	[56,76,77,81]
	$W \succcurlyeq 0$	Positive semi-definition	[47,66–69,77]
	$W = W^T$	Symmetry	[77,81]
	$d_i = \sum_{j=1}^{n} W_{ij} = 1$	Hard degree constraint for normalization	[10,20,53,56,81]
	$d_i = \sum_{j=1}^{n} W_{ij} \geq 1$	Relaxed degree constraint	[58]
	$\sum_i (\max(0, 1 - d_i))^2 \leq \eta n$	Soft degree constraint	[58]

[a] Locality is described using graph topology E; [b] L is a graph Laplacian derived from a pre-specific graph.

which is exactly the *Gaussian kernel*, and, interestingly, the regularized parameter λ in Eq. (24) now plays the role of kernel scale. As formulated in our previous work [10], a normalized Gaussian kernel can also be derived by including an additional simplex constraint $W \in \Delta = \{W_{ij} | \sum_{j=1}^{n} W_{ij} = 1,\ W_{ij} \geq 0\}$, based on Lagrangian multiplier.

Example 2. Letting the objective function be $F(X, W) = \sum_{i,j=1}^{n} \|x_i - x_j\|^2 W_{ij}^2$ and the constraint set $\Delta = \{W_{ij} |\ W_{ij} \geq 0\}$, with a simple formulation based on the Lagrangian multiplier, we can derive the optimal solution of Eq. (24) as $W_{ij}^* = \frac{\lambda}{2} \|x_i - x_j\|^{-2}$, which exactly corresponds to the *inverse Euclidean distance* discussed as in the strategy 3) of Section 3.2.1. Alternatively, we can also use the regularizer $\Omega(W) = -\sum_{i,j=1}^{n} W_{ij}$, which is equivalent to imposing a non-negative constraint on W_{ij} and thus lead to the same solution. In addition, the constraint set $\Delta = \{W_{ij} | \sum_{j=1}^{n} W_{ij} = 1,\ W_{ij} \geq 0\}$ can be employed to achieve a normalized inverse Euclidean edge weight formula, to just name a few.

Besides these two examples, Lee et al. [70] also pointed out that the *partial correlation* can be formulated as an L_1- regularized model, which can be used for constructing functional brain networks. These examples illustrate that the matrix-regularized framework *not only* covers many current graph learning methods, *but also bridges the traditional graph definition (Section 3.2.1) and graph learning (Sections 3.2.2 to 3.2.8).* We argue that it is quite helpful to *unify the traditional graph definition into a learning-based framework,* since the latter provides a more principled way to develop new models by introducing the domain knowledge into predefined graphs in the form of regularizers or constraints. For example, one can achieve a scale-free graph by reweighting different nodes of a graph [113]; In a recent study, Qiao et al. proposed to estimate functional brain networks (graphs) by incorporating a modularity prior [30], to name a few.

5. Discussion

In this section, we will briefly discuss several related topics to graph learning (Section 5.1), and then discuss promising research directions in this field (Section 5.2).

5.1. Related topics to graph learning

As reviewed in the previous sections, a graph is a mathematical tool widely used *not only* in many general learning algorithms, *but also* in some specific application fields. To achieve a "good" graph, abundant algorithms have been developed by optimizing a well-designed objective function with certain constraints. As a special kind of machine learning methods, graph learning is closely related to *at least* three machine learning topics, namely 1) kernel learning, 2) distance metric learning, and 3) dimensionality reduction, as detailed below.

1) *Kernel learning*. Kernel is a popular concept in several branches of mathematics and statistics [114]. It can be defined from a *continuous view* indicating kernel function, or a *discrete view* indicating kernel matrix. In contrast, the graph is a discrete concept with a clearer definition as given in Section 2.

Kernel learning mainly aims to learn an optimized kernel from a large kernel family, and makes the kernel machines, such as SVM, work better. There exists a close relationship between graph learning and kernel learning. On one hand, kernels (especially the Gaussian kernel) provide a natural way to define the edge weights of a graph, which has been widely and successfully used in many practical problems. On the other hand, given a graph, we can define a kernel based on the pseudoinverse of the graph Laplacian [92].

Also, we argue that a graph may be a more flexible concept/tool than a kernel for the learning problems. Besides the point that the pseudoinverse of a graph can be used directly as a kernel, one category of kernel learning methods, namely spectral kernel learning [115], depends heavily on the construction of a high-quality graph. Furthermore, a graph generally includes two parts, E and W, for describing its topological structure and edge weight strength, respectively; in contrast, a kernel just plays the role of the edge weight, and thus cannot effectively model the topological structure behind data. In addition, *without the requirement of a positive semi-definite constraint,* graph learning algorithms can generally work more efficiently than the kernel counterparts. As we know, the standard kernel learning algorithms based on semi-definitive programming (SDP) lead to a high computation complexity of $O(n^{6.5})$ [116].

However, compared with the kernel learning which has been intensively studied, especially in the machine learning field, most graph learning algorithms are just developed based on the intuitive

motivations and are still short of theoretical foundations. Therefore, it is interesting to study graph learning by borrowing experiences in the kernel learning. A typical example is multi-kernel learning [117] which shares some common goals with multi-graph learning, and, thus, may benefit from each other.

2) *Distance metric learning.* It is well known that many graph construction methods, such as the most popular *k*NN graph with Gaussian weights, rely heavily on the employed (dis)similarity measures, and, hence, a reliable distance metric might help improve the quality of the graphs.

Different from the predefined (dis)similarity measures, metric learning aims to learn a metric from data to fit the data distribution or subsequent learning tasks. The popular metric learning algorithms include Xing's global convex optimization method [118], Neighborhood Components Analysis [119] and Large Margin Nearest Neighbor algorithm [120], etc. Refer to the review paper [121] and references therein for more details.

Intuitively, distance metric learning is a more basic problem than graph learning, since a learned metric can *in principle* be used in any subsequent tasks, including classification, clustering, and also graph construction. However, from another point of view, graph learning is *instead* more general than metric learning, since graphs can be constructed based on any (dis)similarity which does not need to meet all the conditions (*e.g.*, triangle inequality) of distance metrics. For example, as mentioned in Section 3.2.5, Cheng et al. design a sparsity induced similarity measure based on the learning of an L_1-graph [62]. In addition, some algorithms attempted to incorporate graph information in the form of regularization into metric learning for better performance [122].

3) *Dimensionality reduction.* Dimensionality reduction (DR) is the most direct way to alleviate the "curse of dimensionality" by mapping data linearly or nonlinearly into a low-dimensional space. The principal component analysis (PCA) and linear discriminant analysis (LDA) are two representatives of linear DR methods, while various kernelized versions and manifold learning methods provide solutions for nonlinear DR. According to a popular research [123] and its subclass extension [124], most of the existing DR methods and some subclass variants, such as SDA [125] and CDA [126], can be unified under a graph embedding framework. That is, a large family of DR methods can be converted to a graph construction problem, or, simply speaking, graphs determine DR methods to some extent.

As a summary of this part, we use three conclusive sentences to give brief comments on the relationship between graph learning with (1) kernel learning, (2) metric learning, and (3) dimensionality reduction, respectively. For (1) kernel learning, a kernel can naturally play the role of the edge weight matrix in a graph, and the pseudo-inverse of a graph Laplacian can conversely be used as a kernel; for (2) metric learning, we can construct a graph given any learned metric, and some metrics can also be induced from a given graph; for (3) dimensionality reduction, a certain projection can exactly correspond to a Mahalanobis distance, which can be used to construct graphs, and, on the contrary, given a graph, we can learn a projection by linearly graph embedding. In our opinion, these four related topics share some intrinsic characteristics, in spite of their different focuses. On a higher-level, all of them aim to learn a new representation from the original data, which is expected to be more adaptive to the subsequent tasks, such as object recognition, disease diagnosis, and pattern mining. This partially supports the essential role of representation learning, which is considered as a critical factor for the success of deep learning [127] in some application fields.

In addition, it is worth pointing out that some other topics are also conceptually related to graphs. For example, a hyper-graph [128] is a natural generalization of traditional graphs as discussed in this paper. Unlike traditional graphs in which each edge links two nodes, the "edge" in hyper-graphs can "link" more than two nodes. As a result, the hyper-graph provides a more powerful tool to model "higher-order" relationships in data. In recent years, hyper-graphs have been successfully used in many machine learning algorithms [128] and some specific applications, such as image classification [129], disease diagnosis [130], protein function prediction [131], and functional brain network modeling [132], etc. Given the complex structures in data, we argue that hypergraph construction and learning will become an increasingly promising study topic in the future. Besides hyper-graphs, in the following Section 5.2, we further suggest several interesting research directions related to graph learning.

5.2. Promising research directions

1) *Application-oriented graph learning.* As reviewed above in this paper, various objective functions have been proposed to conduct graph learning. Now a natural problem is what constitutes a good graph. However, there is no unified answer for this question, and whether a graph is good or not is really dependent on the specific learning tasks or application fields. For example, clustering algorithms generally want the graph to be disconnected, while, for manifold learning, the graph is expected to be connected [23]. Also, the popular *b*-matching graph learning paper [21] verified that the regular or balanced graphs can help improve the performance of semi-supervised learning. However, for some other applications, such as functional brain connectivity modeling, the corresponding graph is not necessarily regular, due to the proved small-world characteristic of brain networks.

As a result, researchers need to design different graph learning algorithms based on different priors, motivations, or applications. Similar to other machine learning algorithms, graph learning also obeys the "no free lunch [133]" rule. In brief, we argue that the application-oriented graph learning will become an increasingly important field, since practical problems provide a starting point for conducting graph learning researches and *meanwhile* give a terminal point to tell whether a graph is good or not.

2) *Big graph learning.* As we know, big data is currently a hot topic in computer science and its related communities. Based on big data and machine learning, the performance of many practical systems, such as computer vision and natural language processing, has received significant improvement. For graph learning (or, more accurately, data-driven graph learning as discussed in this paper), it naturally faces the opportunities and challenges of big data. Here, we just take the brain network as an example and give a brief discussion of the challenge to construct/learn a big graph. It has been estimated that one human brain comprises 8.6×10^{11} neurons and nearly 10^{14} synapses [134]. Just considering a medium-scale brain MRI image with $64 \times 64 \times 30$ voxels, if we aim to construct a brain network for modeling the detailed connectivity relationship among voxels without any post-processing, it would theoretically generate a graph with more than 10^{10} edges. Obviously, it is unrealistic, at least extremely challenging, to handle a so large graph. Even at the macro-scale of brain regions, the order of nodes is generally from $\sim 10^2$ to $\sim 10^4$, which then results in $\sim$5000 to $\sim 5 \times 10^6$ edges [134]. A recent study [135] reported that approximately 11 weeks were spent for modeling a group of brain networks with 40,000 nodes by parallel processing on 12 cores of a high-performance cluster with 128 G RAM.

In machine learning and some related communities, researchers have currently developed many strategies for scalable graph con-

struction, in which the hashing trick, for example, is a widely-used scheme [136-139]. However, most of the existing works mainly focus on the construction of the simplest *k*NN graph, and, thus, it is an interesting topic for the future researches to explore scalable algorithms for the other graph learning models reviewed in this paper.

3) *Complex-structured graph learning.* In the classical graph theory, the graphs are generally assumed to have relatively simple structures, like the widely studied regular lattice graph. However, in practice, the structures among a system tend to be considerably complicated, and, thus, the corresponding graphs generate some new properties such as a scale-free pattern [140] and small-worldness [141]. Again, we can take a brain network as an example. In addition to the popular small worldness and scale-free degree distribution, studies have shown that the brain networks are sparse and hierarchical, and include hubs and modules. As a result, we need to explore new graph learning algorithms to model brain networks by well considering these prior characteristics. So far, to our best knowledge, only limited priors (*e.g.*, sparsity) have been used in brain network construction. However, fortunately, in the statistics and machine learning fields, researchers have been attempting to learn a better graph following this line. For example, Liu and Ihler in [113] proposed to learn a scale-free graph by reweighted sparse representation, and Tan et al. in [142] conducted graph learning for modeling hub structure in the data. In our view, an effective graph learning model in the future should cover as many of these priors as possible, which may ultimately give rise to a series of complex optimizations with multiple constraints. And, we argue that the matrix-regularized graph learning framework as reviewed in Section 4 may be one of the options to model this problem.

6. Conclusion

Graphs play an increasingly important role in modeling the connected world, ranging from microscopic protein molecule structures to macroscopic ecological and social networks. It is well known that a good graph may lead to good clustering results, good low-dimensional embedding, good spectral kernels, good classification accuracy, good brain network expression, etc. In a word, graphs lie at the heart of a large family of algorithms and applications. Then, what is a “good” graph? It is difficult to give a unified answer to this problem. However, there are more appropriate and less appropriate graph learning methods depending on the datasets and the research questions. Therefore, in this paper, we have systematically reviewed a wide range of graph construction/learning methods, and provided suggestions for selection suitable methods in practice. Beyond this, we have also unified many of the current graph construction/learning methods into a matrix-regularized framework, and discussed several topics and promising directions related to graph learning.

A characteristic of this paper is summarizing graph construction methods by putting two fields, machine learning and brain connectome [143], together, due to the two-fold considerations. (1) The graph construction problem has been studied more widely and deeply in the machine learning field than that in the new-rising brain connectome, and also various construction methods have been proposed for different tasks, which may provide some beneficial techniques and insights into brain network modeling. (2) After a long-term evolution of human being, the brain may have a nearly optimal connectivity network, whose properties, such as small worldness, scale-free, hierarchy, modularity, etc., are expected to guide the graph construction involved in the machine learning field. Thus, we expect that the readers from both machine learning and neuroscience fields will benefit from this paper.

Finally, it is worth pointing out that a graph is a ubiquitous conception, and the graph construction (or learning) problem may be involved in a large number of fields. Despite collecting as many methods as possible, we have mainly reviewed the methods for constructing undirected weighted graphs, in consideration of their simplicity, robustness, and efficiency. However, directed and causal graphs may be more powerful tools to model rich structures or relationships in data. Learning the directed causal graphs is an interesting topic, related to probabilistic graphical models [46], statistical relational learning [144] and some specific applications. A comprehensive review on this topic is expected in the future.

Acknowledgement

This work was partly supported by National Natural Science Foundation of China (61300154, 61402215), Natural Science Foundation of Shandong Province (ZR2018MF020), and NIH (EB008374, EB009634, MH100217, AG041721, AG049371, AG042599).

References

[1] J.J. Sylvester, Chemistry and algebra, Nature 17 (432) (1878) 284.
[2] H. Shin, A.M. Lisewski, O. Lichtarge, Graph sharpening plus graph integration: a synergy that improves protein functional classification, Bioinformatics 23 (23) (2007) 3217–3224.
[3] J.M. Stuart, E. Segal, D. Koller, S.K. Kim, A gene-coexpression network for global discovery of conserved genetic modules, Science 302 (5643) (2003) 249–255.
[4] O. Sporns, Networks of the Brain, MIT press, 2011.
[5] A. Fornito, A. Zalesky, M. Breakspear, Graph analysis of the human connectome: promise, progress, and pitfalls, Neuroimage 80 (2013) 426–444.
[6] K. Rosen, Discrete Mathematics and Its Applications, Seventh ed., McGraw-Hill, 2011.
[7] X. Yan, J. Han, gSpan: graph-based substructure pattern mining, in: Proceedings of the IEEE International Conference on Data Mining (ICDM), 2002, pp. 721–724.
[8] J. Wu, Z. Hong, S. Pan, X. Zhu, Z. Cai, C. Zhang, Multi-graph-view subgraph mining for graph classification, Knowl. Inf. Syst. 48 (1) (2016) 29–54.
[9] M. Maier, U. Luxburg, Influence of graph construction on graph-based clustering measures, in: Proceedings of the Neural Information Processing Systems (NIPS), 21, 2008, pp. 1025–1032.
[10] L. Zhang, L. Qiao, S. Chen, Graph-optimized locality preserving projections, Pattern Recognit. 43 (6) (2010) 1993–2002.
[11] C.A.R. de Sousa, S.O. Rezende, G.E. Batista, Influence of graph construction on semi-supervised learning, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2013, pp. 160–175.
[12] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical learning: Data mining, inference, and Prediction, Second ed., Springer, New York, 2009.
[13] L. Fan, et al., The human brainnetome atlas: a new brain atlas based on connectional architecture, Cereb. Cortex 26 (8) (2016) 3508–3526.
[14] P.P. Talukdar, Topics in graph construction for semi-supervised learning, Technical Report, 2009.
[15] K. Riesen, H. Bunke, Graph Classification and Clustering Based On Vector Space Embedding, World Scientific Publishing Co., Inc, 2010.
[16] M. Wang, X.-S. Hua, J. Tang, R. Hong, Beyond distance measurement: constructing neighborhood similarity for video annotation, IEEE Trans. Multimed. 11 (3) (2009) 465–476.
[17] X. Bai, X. Yang, L.J. Latecki, W. Liu, Z. Tu, Learning context-sensitive shape similarity by graph transduction, IEEE Trans. Pattern Anal. Mach. Intell. 32 (5) (2010) 861–874.
[18] X. Yang, L. Prasad, L.J. Latecki, Affinity learning with diffusion on tensor product graph, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 28–38.
[19] A. Khoreva, F. Galasso, M. Hein, B. Schiele, Classifier based graph construction for video segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, pp. 951–960.
[20] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[21] T. Jebara, J. Wang, S. Chang, Graph construction and b-matching for semi-supervised learning, in: Proceedings of the International Conference on Machine Learning (ICML), 2009.
[22] C.J. Stam, Modern network science of neurological disorders, Nat. Rev. Neurosci. 15 (10) (2014) 683–695.
[23] R.S. Zemel, M.Á. Carreira-Perpiñán, Proximity graphs for clustering and manifold learning, in: Proceedings of the Neural Information Processing Systems, 2004, pp. 225–232.
[24] K.R. Gabriel, R.R. Sokal, A new statistical approach to geographic variation analysis, Syst. Biol. 18 (3) (1969) 259–278.

[25] J. Choo, R. Jiamthapthaksin, C.-S. Chen, O.U. Celepcikay, C. Giusti, C.F. Eick, MOSAIC: a proximity graph approach for agglomerative clustering, in: Data Warehousing and Knowledge Discovery, Springer, 2007, pp. 231–240.
[26] G.T. Toussaint, The relative neighbourhood graph of a finite planar set, Pattern Recognit. 12 (4) (1980) 261–268.
[27] G.T. Toussaint, Applications of the relative neighbourhood graph, Int. J. Adv. Comput. Sci. Appl. 4 (3) (2014) 77–85.
[28] D. Kirkpatrick, J.D. Radke, A framework for computational morphology, in: G. Toussaint (Ed.), Computational Geometry, 1985, pp. 217–248.
[29] C.D. Correa, P. Lindstrom, Locally-scaled spectral clustering using empty region graphs, in: Proceedings of the Eighteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 1330–1338.
[30] L. Qiao, H. Zhang, M. Kim, S. Teng, L. Zhang, D. Shen, Estimating functional brain networks by incorporating a modularity prior, NeuroImage 141 (2016) 399–407.
[31] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University," Citeseer 2002.
[32] B.C. Huang, T. Jebara, Fast b-matching via sufficient selection belief propagation, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2011, pp. 361–369.
[33] J. Wang, Z. Zhang, H. Zha, Adaptive manifold learning, in: Proceedings of the Advances in Neural Information Processing Systems, 2005, pp. 1473–1480.
[34] Z. Zhang, J. Wang, H. Zha, Adaptive manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2) (2012) 253–265.
[35] V. Premachandran, R. Kakarala, Consensus of k-NNs for robust neighborhood selection on graph-based manifolds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2013, pp. 1594–1601.
[36] X. Zhu, C.C. Loy, S. Gong, Constructing robust affinity graphs for spectral clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2014, pp. 1450–1457.
[37] K.A. Garrison, D. Scheinost, E.S. Finn, X. Shen, R.T. Constable, The (in) stability of functional brain network measures across thresholds, NeuroImage (2015).
[38] M.H. Rohban, H.R. Rabiee, Supervised neighborhood graph construction for semi-supervised classification, Pattern Recognit. 45 (4) (2012) 1363–1372.
[39] A. Gretton, K.M. Borgwardt, M. Rasch, B. Schölkopf, A.J. Smola, A kernel method for the two-sample-problem, in: Proceedings of the Advances in Neural Information Processing Systems, 2006, pp. 513–520.
[40] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, in: Proceedings of the Advances in Neural Information Processing Systems, 2004, pp. 1601–1608.
[41] X. Yang, X. Bai, L.J. Latecki, Z. Tu, Improving shape retrieval by learning graph transduction, in: Computer Vision–ECCV 2008, Springer, 2008, pp. 788–801.
[42] C. Cortes, M. Mohri, On transductive regression, in: Proceedings of the Neural Information Processing Systems (NIPS), 2007.
[43] S.M. Smith, et al., Network modelling methods for FMRI, Neuroimage 54 (2) (2011) 875–891.
[44] J.S. Maritz, Distribution-free Statistical Methods, CRC Press, 1995.
[45] M.G. Kendall, A new measure of rank correlation, Biometrika (1938) 81–93.
[46] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, MIT Press, 2009.
[47] J. Friedman, T. Hastie, R. Tibshirani, Sparse inverse covariance estimation with the graphical lasso, Biostatistics 9 (3) (2008) 432–441.
[48] S.M. Plis, et al., High-order interactions observed in multi-task intrinsic networks are dominant indicators of aberrant brain function in schizophrenia, NeuroImage 102 (2014) 35–48.
[49] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in: Proceedings of the International Conference on Machine Learning, 3, 2003, pp. 912–919.
[50] X. Zhang, W.S. Lee, Hyperparameter learning for graph based semi-supervised learning algorithms, in: Proceedings of the Advances in Neural Information Processing Systems, 2006, pp. 1585–1592.
[51] A. Kapoor, H. Ahn, Y. Qi, R.W. Picard, Hyperparameter and kernel learning for graph based semi-supervised classification, in: Proceedings of the Advances in Neural Information Processing Systems, 2005, pp. 627–634.
[52] M. Karasuyama, H. Mamitsuka, Manifold-based similarity adaptation for label propagation, in: Proceedings of the Neural Information Processing Systems, 2013, pp. 1547–1555.
[53] L.K. Saul, S.T. Roweis, Think globally, fit locally: unsupervised learning of low dimensional manifolds, J. Mach. Learn. Res. 4 (2003) 119–155.
[54] L. Ma, M.M. Crawford, X. Yang, Y. Guo, Local-manifold-learning-based graph construction for semisupervised hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 53 (5) (2015) 2832–2844.
[55] X. Zhu, J. Lafferty, R. Rosenfeld, Semi-supervised Learning With Graphs, Carnegie Mellon University, Language Technologies Institute, School Of Computer Science, 2005.
[56] F. Wang, C.S. Zhang, Label propagation through linear Neighborhoods, IEEE Trans. Knowl. Data Eng. 20 (1) (2008) 55–67.
[57] Y. Zhang, K. Huang, X. Hou, C. Liu, Learning Locality preserving graph from data, IEEE Trans. Cybern. 44 (11) (2014) 2088–2098.
[58] S.I. Daitch, J.A. Kelner, D.A. Spielman, Fitting a graph to vector data, in: Proceedings of the International Conference on Machine Learning (ICML), 2009.
[59] B. Cheng, J. Yang, S. Yan, Y. Fu, T. Huang, Learning with L1-graph for image analysis, IEEE Trans. Image Process. 19 (4) (2010) 858–866.
[60] E. Elhamifar, R. Vidal, Sparse subspace clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.
[61] S. Yan, H. Wang, Semi-supervised learning by sparse representation, in: Proceedings of the SIAM International Conference on Data Mining (SDM), 2009.
[62] H. Cheng, Z. Liu, J. Yang, Sparsity induced similarity measure for label propagation, in: Proceedings of the IEEE Twelfth International Conference on Computer Vision, IEEE, 2009, pp. 317–324.
[63] L.S. Qiao, S.C. Chen, X.Y. Tan, Sparsity preserving projections with applications to face recognition, Pattern Recognit. 43 (1) (Jan 2010) 331–341.
[64] R. He, W.-S. Zheng, B.-G. Hu, X.-W. Kong, Nonnegative sparse coding for discriminative semi-supervised learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2011, pp. 2849–2856.
[65] J. Gui, Z. Sun, W. Jia, R. Hu, Y. Lei, S. Ji, Discriminant sparse neighborhood preserving embedding for face recognition, Pattern Recognit. 45 (8) (2012) 2884–2893.
[66] N. Meinshausen, P. Bühlmann, High-dimensional graphs and variable selection with the lasso, Ann. Stat. (2006) 1436–1462.
[67] S. Huang, et al., Learning brain connectivity of Alzheimer's disease from neuroimaging data, in: Proceedings of the Advances in Neural Information Processing Systems, 2009, pp. 808–816.
[68] J. Peng, P. Wang, N. Zhou, J. Zhu, Partial correlation estimation by joint sparse regression models, J. Am. Stat. Assoc. 104 (486) (2009).
[69] G. Varoquaux, A. Gramfort, J.-B. Poline, B. Thirion, Brain covariance selection: better individual functional connectivity models using population prior, in: Proceedings of the Advances in Neural Information Processing Systems, 2010, pp. 2334–2342.
[70] H. Lee, D.S. Lee, H. Kang, B.-N. Kim, M.K. Chung, Sparse brain network recovery under compressed sensing, IEEE Trans. Med. Imaging 30 (5) (2011) 1154–1165.
[71] C.-Y. Wee, P.-T. Yap, D. Zhang, L. Wang, D. Shen, Group-constrained sparse fMRI connectivity modeling for mild cognitive impairment identification, Brain Struct. Funct. 219 (2) (2014) 641–656.
[72] G. Liu, Z. Lin, Y. Yu, Robust subspace segmentation by low-rank representation, in: Proceedings of the International Conference on Machine Learning (ICML), 2010.
[73] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, Y. Ma, Robust recovery of subspace structures by low-rank representation, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 171–184.
[74] J. Liu, S. Ji, J. Ye, SLEP: sparse learning with efficient projections, Arizona State University 6 (2009) 491.
[75] L. Zhuang, H. Gao, J. Huang, N. Yu, Semi-supervised classification via low rank graph, in: Proceedings of the Sixth International Conference on Image and Graphics (ICIG), IEEE, 2011, pp. 511–516.
[76] L. Zhuang, H. Gao, Z. Lin, Y. Ma, X. Zhang, N. Yu, Non-negative low rank and sparse graph for semi-supervised learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 2328–2335.
[77] D. Luo, H. Huang, F. Nie, C.H. Ding, Forging the graphs: a low rank and positive semidefinite graph learning approach, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 2960–2968.
[78] X. Lu, Y. Wang, Y. Yuan, Graph-regularized low-rank representation for destriping of hyperspectral images, IEEE Trans. Geosci. Remote Sens. 51 (7) (2013) 4009–4018.
[79] Y. Zheng, X. Zhang, S. Yang, L. Jiao, Low-rank representation with local constraint for graph construction, Neurocomputing 122 (2013) 398–405.
[80] K. Tang, R. Liu, Z. Su, J. Zhang, Structure-constrained low-rank representation, IEEE Trans. Neural Netw. Learn. Syst. 25 (12) (2014) 2167–2179.
[81] R. Liu, Z. Lin, Z. Su, Learning Markov random walks for robust subspace clustering and estimation, Neural Netw. 59 (2014) 1–15.
[82] C.-Y. Lu, H. Min, Z.-Q. Zhao, L. Zhu, D.-S. Huang, S. Yan, Robust and efficient subspace segmentation via least squares regression, in: Computer Vision–ECCV 2012, Springer, 2012, pp. 347–360.
[83] H. Zhao, Z. Ding, Y. Fu, Block-wise constrained sparse graph for face image representation, in: Proceedings of the Eleventh IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 1, IEEE, 2015, pp. 1–6.
[84] S. Li, Y. Fu, Low-rank coding with b-matching constraint for semi-supervised classification, in: Proceedings of the Twenty Third International Joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 1472–1478.
[85] S. Li, Y. Fu, Learning balanced and unbalanced graphs via low-rank coding, IEEE Trans. Knowl. Data Eng. 27 (5) (2015) 1274–1287.
[86] S. Wang, D. Liu, Z. Zhang, Nonconvex relaxation approaches to robust matrix recovery, in: Proceedings of the Twenty Third International Joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 1764–1770.
[87] Y. Peng, B.-L. Lu, S. Wang, Enhanced low-rank representation via sparse manifold adaption for semi-supervised learning, Neural Netw. 65 (2015) 1–17.
[88] T. Zhang, R. Ji, W. Liu, D. Tao, G. Hua, Semi-supervised learning with manifold fitted graphs, in: Proceedings of the Twenty Third International Joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 1896–1902.
[89] J. Wu, S. Pan, X. Zhu, C. Zhang, X. Wu, Positive and unlabeled multi-graph learning, IEEE Trans. Cybern. PP (99) (2016) 1–12.
[90] J. Wu, X. Zhu, C. Zhang, P.S. Yu, Bag constrained structure pattern mining for multi-graph classification, IEEE Trans. Knowl. Data Eng. 26 (10) (2014) 2382–2396.
[91] K. Tsuda, H. Shin, B. Schölkopf, Fast protein classification with multiple networks, Bioinformatics 21 (Suppl 2) (2005) ii59–ii65.
[92] A. Argyriou, M. Herbster, M. Pontil, Combining graph Laplacians for semi su-

pervised learning, in: Proceedings of the Advances in Neural Information Processing Systems, 2006, pp. 67–74.
[93] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, Y. Song, Unified video annotation via multigraph learning, IEEE Trans. Circuits Syst. Video Technol. 19 (5) (2009) 733–746.
[94] S. Zhao, H. Yao, Y. Yang, Y. Zhang, Affective image retrieval via multi-graph learning, in: Proceedings of the ACM International Conference on Multimedia, ACM, 2014, pp. 1025–1028.
[95] M. Karasuyama, H. Mamitsuka, Multiple graph label propagation by sparse integration, IEEE Trans. Neural Netw. Learn. Syst. 24 (12) (2013) 1999–2012.
[96] Y. Gao, E. Adeli-M, M. Kim, P. Giannakopoulos, S. Haller, D. Shen, Medical image retrieval using multi-graph learning for MCI diagnostic assistance, in: Proceedings of the Medical Image Computing and Computer-Assisted Intervention–MICCAI, Springer, 2015, pp. 86–93.
[97] X. Yang, M. Wang, D. Tao, Robust visual tracking via multi-graph ranking, Neurocomputing 159 (2015) 35–43 7/2/.
[98] M. Wang, H. Li, D. Tao, K. Lu, X. Wu, Multimodal graph-based reranking for web image search, IEEE Trans. Image Process. 21 (11) (2012) 4649–4661.
[99] W. Tang, Z. Lu, I.S. Dhillon, Clustering with multiple graphs, in: Proceedings of the IEEE International Conference on Data Mining (ICDM), IEEE, 2009, pp. 1016–1021.
[100] B. Wang, J. Jiang, W. Wang, Z.-H. Zhou, Z. Tu, Unsupervised metric fusion by cross diffusion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 2997–3004.
[101] B. Wang, et al., Similarity network fusion for aggregating data types on a genomic scale, Nat. Methods 11 (3) (2014) 333–337.
[102] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Educ. Psychol. 24 (6) (1933) 417.
[103] T. Ahonen, A. Hadid, M. Pietikäinen, Face recognition with local binary patterns, in: Computer Vision-ECCV, Springer, 2004, pp. 469–481.
[104] L. Wiskott, J.-M. Fellous, N. Kuiger, C. Von Der Malsburg, Face recognition by elastic bunch graph matching, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 775–779.
[105] L. Zhang, S. Chen, L. Qiao, Graph optimization for dimensionality reduction with sparsity constraints, Pattern Recognit. 45 (3) (2012) 1205–1210, doi:10.1016/j.patcog.2011.08.015.
[106] L. Qiao, L. Zhang, S. Chen, Dimensionality reduction with adaptive graph, Front. Comput. Sci. 7 (5) (2013) 745–753.
[107] B. Xie, M. Wang, D. Tao, Toward the optimization of normalized graph Laplacian, IEEE Trans. Neural Netw. 22 (4) (2011) 660–666.
[108] W.K. Wong, H.T. Zhao, Supervised optimal locality preserving projection, Pattern Recognit. 1 (45) (2012) 186–197.
[109] D. Zhang, Z.-H. Zhou, S. Chen, Semi-supervised dimensionality reduction, in: Proceedings of the International Conference on Data Mining, SIAM, 2007, pp. 629–634.
[110] C.M. Bishop, in: Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, New York, 2006, p. 738. xxp.
[111] O. Bousquet, S. Boucheron, G. Lugosi, Introduction to statistical learning theory, in: Advanced Lectures on Machine Learning, Springer, 2004, pp. 169–207.
[112] M. Liu, D. Zhang, Pairwise constraint-guided sparse learning for feature selection, IEEE Trans. Cybern. 46 (1) (2016) 298–310.
[113] Q. Liu, A.T. Ihler, Learning scale free networks by reweighted l1 regularization, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2011, pp. 40–48.
[114] J. Shawe-Taylor, N. Cristianini, Kernel Methods For Pattern Analysis, Cambridge university press, 2004.
[115] R. Johnson, T. Zhang, Graph-based semi-supervised learning and spectral kernel design, IEEE Trans. Inf. Theory 54 (1) (2008) 275–288.
[116] E.-L. Hu, B. Wang, S. Chen, BCDNPKL: scalable non-parametric kernel learning using block coordinate descent, in: Proceedings of the Twenty Eighth International Conference on Machine Learning (ICML), 2011, pp. 209–216.
[117] M. Gönen, E. Alpaydın, Multiple kernel learning algorithms, J. Mach. Learn. Res. 12 (2011) 2211–2268.
[118] E.P. Xing, M.I. Jordan, S. Russell, A.Y. Ng, Distance metric learning with application to clustering with side-information, in: Proceedings of the Advances in Neural Information Processing Systems, 2002, pp. 505–512.
[119] J. Goldberger, G.E. Hinton, S.T. Roweis, R. Salakhutdinov, Neighbourhood components analysis, in: Proceedings of the Advances in Neural Information Processing Systems, 2004, pp. 513–520.
[120] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (2009) 207–244.
[121] L. Yang, "Distance Metric Learning: A Comprehensive Survey (Techinical Report)," 2006.
[122] B. Shaw, B. Huang, T. Jebara, Learning a distance metric from a network, in: Proceedings of the Advances in Neural Information Processing Systems, 2011, pp. 1899–1907.
[123] S.C. Yan, D. Xu, B.Y. Zhang, H.J. Zhang, Q. Yang, S. Lin, Graph embedding and extensions: a general framework for dimensionality reduction, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (Jan 2007) 40–51.
[124] A. Maronidis, A. Tefas, I. Pitas, Subclass graph embedding and a marginal fisher analysis paradigm, Pattern Recognit. 48 (12) (2015) 4024–4035.
[125] Z. Manli, A.M. Martinez, Subclass discriminant analysis, IEEE Trans. Pattern Anal. Mach. Intell. 28 (8) (2006) 1274–1286.
[126] X.-W. Chen, T. Huang, Facial expression recognition: a clustering-based approach, Pattern Recognit. Lett. 24 (9–10) (2003) 1295–1302 6//.
[127] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[128] D. Zhou, J. Huang, B. Schölkopf, Learning with hypergraphs: clustering, classification, and embedding, in: Advances in Neural Information Processing Systems, 2006, pp. 1601–1608.
[129] J. Yu, D. Tao, M. Wang, Adaptive hypergraph learning and its application in image classification, IEEE Trans. Image Process. 21 (7) (2012) 3262–3272.
[130] Y. Gao, et al., MCI identification by joint learning on multiple MRI data, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015, Springer, 2015, pp. 78–85.
[131] S.R. Gallagher, D.S. Goldberg, Clustering coefficients in protein interaction hypernetworks, in: Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics, ACM, 2013, p. 552.
[132] B. Jie, D. Shen, D. Zhang, Brain connectivity hyper-network for MCI classification, in: Proceedings of the Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, 2014, pp. 724–732.
[133] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Second ed., Wiley, New York, 2001.
[134] A. Fornito, A. Zalesky, M. Breakspear, The connectomics of brain disorders, Nat. Rev. Neurosci. 16 (3) (2015) 159–172.
[135] A. Horn, D. Ostwald, M. Reisert, F. Blankenburg, The structural-functional connectome and the default mode network of the human brain, Neuroimage 102 (2014) 142–151.
[136] W. Liu, J. He, S.-F. Chang, Large graph construction for scalable semi-supervised learning, in: Proceedings of the International Conference on Machine Learning (ICML), 2010.
[137] W. Liu, J. Wang, S. Kumar, S.-F. Chang, Hashing with graphs, in: Proceedings of the International Conference on Machine Learning (ICML), 2011.
[138] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, S. Li, Scalable k-NN graph construction for visual descriptors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 1106–1113.
[139] D.C. Anastasiu, G. Karypis, L2Knng: fast exact k-nearest neighbor graph construction with L2-norm pruning, in: Proceedings of the Twenty Fourth ACM International on Conference on Information and Knowledge Management, ACM, 2015, pp. 791–800.
[140] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[141] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[142] K.M. Tan, P. London, K. Mohan, S.-I. Lee, M. Fazel, D. Witten, Learning graphical models with hubs, J. Mach. Learn. Res. 15 (1) (2014) 3297–3331.
[143] S.M. Smith, et al., Functional connectomics from resting-state fMRI, Trends Cognit. Sci. 17 (12) (2013) 666–682.
[144] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, Proc. IEEE 104 (1) (2016) 11–33.

Lishan Qiao received the B.Sc. degree in mathematics from Liaocheng University (LU) in 2001. He received the M.Sc. degree in applied mathematics from Chengdu University of Technology in 2004, and then worked in LU to this day. In 2010, he received the Ph.D. degree in computer science, at the Department of Computer Science & Engineering, Nanjing University of Aeronautics and Astronautics (NUAA). Currently, he is an associate professor at School of Mathematics Science, LU, and his research interests pattern recognition, machine learning, and functional brain network analysis. In these fields, he has published more than 20 papers, and one of the papers won the Biennial Pattern Recognition Journal 2010 Best Paper Award Honorable Mention.

Limei Zhang received his B.S. and M.S. degree in mathematics from Liaocheng University in 2001 and 2007, respectively. In 2012, She received her Ph.D. degree from Department of Computer Science & Engineering, Nanjing university of Aeronautics & Astronautics (NUAA). Currently she is an associate professor at Liaocheng University. Her research interests focus on pattern recognition and machine learning.

Songcan Chen received the B.Sc. degree in Mathematics from Hangzhou University (now merged into Zhejiang University) in 1983. In December 1985, he completed the M.Sc. degree in Computer Applications at Shanghai Jiaotong University and then worked at Nanjing university of Aeronautics & Astronautics (NUAA) in January 1986 as an Assistant Lecturer. There he received a Ph.D. degree in Communication and Information Systems in 1997. Since 1998, as a full Professor, he has been with the Department of Computer Science and Engineering at NUAA. His research interests include pattern recognition, machine learning and neural computing. In these fields, he has authored or coauthored over 130 scientific journal papers.

Dinggang Shen is a Professor of Radiology, Biomedical Research Imaging Center (BRIC), Computer Science, and Biomedical Engineering in the University of North Carolina at Chapel Hill (UNC-CH). He is currently directing the Center for Image Analysis and Informatics, the Image Display, Enhancement, and Analysis (IDEA) Lab in the Department of Radiology, and also the medical image analysis core in the BRIC. He was a tenure-track assistant professor in the University of Pennsylvanian (UPenn), and a faculty member in the Johns Hopkins University. Dr. Shen's research interests include medical image analysis, computer vision, and pattern recognition. He has published more than 800 papers in the international journals and conference proceedings. He serves as an editorial board member for six international journals. He is Fellow of IEEE, and also Fellow of The American Institute for Medical and Biological Engineering (AIMBE).