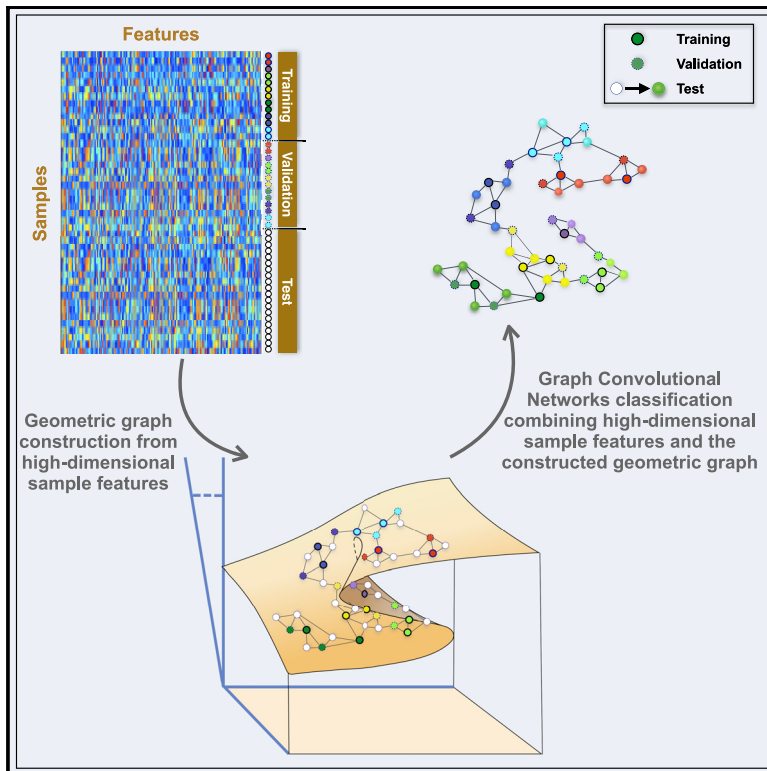# Geometric graphs from data to aid classification tasks with Graph Convolutional Networks

Graphical abstract

Authors

Yifan Qian, Paul Expert,
Pietro Panzarasa, Mauricio Barahona

Correspondence

p.panzarasa@qmul.ac.uk (P.P.),
m.barahona@imperial.ac.uk (M.B.)

In brief

We systematically explore how feature-derived graphs, constructed and optimized in a simple and problem-agnostic manner, can be used with Graph Convolutional Networks to improve classification performance on datasets where graphs are not originally available. This allows us to leverage recent graph-based deep-learning algorithms and extend the applicability of Graph Neural Networks to feature-only datasets.

Highlights

- Geometric graphs from data can be used in deep learning to improve classification

- Optimized graphs align the data to the class labels and enhance class separability

- Sparsifying the optimized graph can potentially improve classification performance

- Extensive experiments are performed on datasets from various scientific domains

CellPress

Article

Geometric graphs from data to aid classification tasks with Graph Convolutional Networks

Yifan Qian,[1] Paul Expert,[2,3] Pietro Panzarasa,[1,*] and Mauricio Barahona[4,5,*]

[1]School of Business and Management, Queen Mary University of London, London, UK
[2]Global Digital Health Unit, School of Public Health, Imperial College London, London, UK
[3]World Research Hub Initiative, Tokyo Institute of Technology, Tokyo, Japan
[4]Department of Mathematics, Imperial College London, London, UK
[5]Lead contact
*Correspondence: p.panzarasa@qmul.ac.uk (P.P.), m.barahona@imperial.ac.uk (M.B.)
https://doi.org/10.1016/j.patter.2021.100237

THE BIGGER PICTURE Supervised classification assigns unseen samples to classes based on their features by learning from examples with known class labels. We show that classification can be improved by using the sample features not only as the basis for classification, but also as a means to construct geometric graphs that encapsulate the closeness between samples. Such feature-derived graphs can be used within graph-based deep-learning models to improve classification. To understand the benefits of these graphs, we show that they align the data to the class labels and enhance class separability. We also demonstrate how to make the graphs sparser, and hence more efficient, while still potentially improving their performance. Our findings are timely given the increasing interest in combining graphs with classification and learning tasks.

1 2 3 4 5 **Development/Pre-production:** Data science output has been rolled out/validated across multiple domains/problems

---

SUMMARY

Traditional classification tasks learn to assign samples to given classes based solely on sample features. This paradigm is evolving to include other sources of information, such as known relations between samples. Here, we show that, even if additional relational information is not available in the dataset, one can improve classification by constructing geometric graphs from the features themselves, and using them within a Graph Convolutional Network. The improvement in classification accuracy is maximized by graphs that capture sample similarity with relatively low edge density. We show that such feature-derived graphs increase the alignment of the data to the ground truth while improving class separation. We also demonstrate that the graphs can be made more efficient using spectral sparsification, which reduces the number of edges while still improving classification performance. We illustrate our findings using synthetic and real-world datasets from various scientific domains.

INTRODUCTION

Classifying samples into a given set of classes is one of the fundamental tasks of data analytics.[1] In supervised machine learning, traditional methods train a classifier using a dataset in which both features and class labels are observed for each sample. Once a classifier has been learned from the training dataset, its parameters are optimized over a validation set. Then the model can be used to predict the class of unseen samples based on their features. Intuitively, a good classifier learns a representation of the data where samples belonging to different classes are well separated.

In some instances, datasets contain additional information in the form of observed relational links between samples. For example, in a dataset of scientific articles, each article will be described by features that encode its text, but we might also have information on citations between articles; in a dataset of patients, each person will be associated with a series of clinical or socio-economic features, but we might also have information about their social interactions. Such relational information could
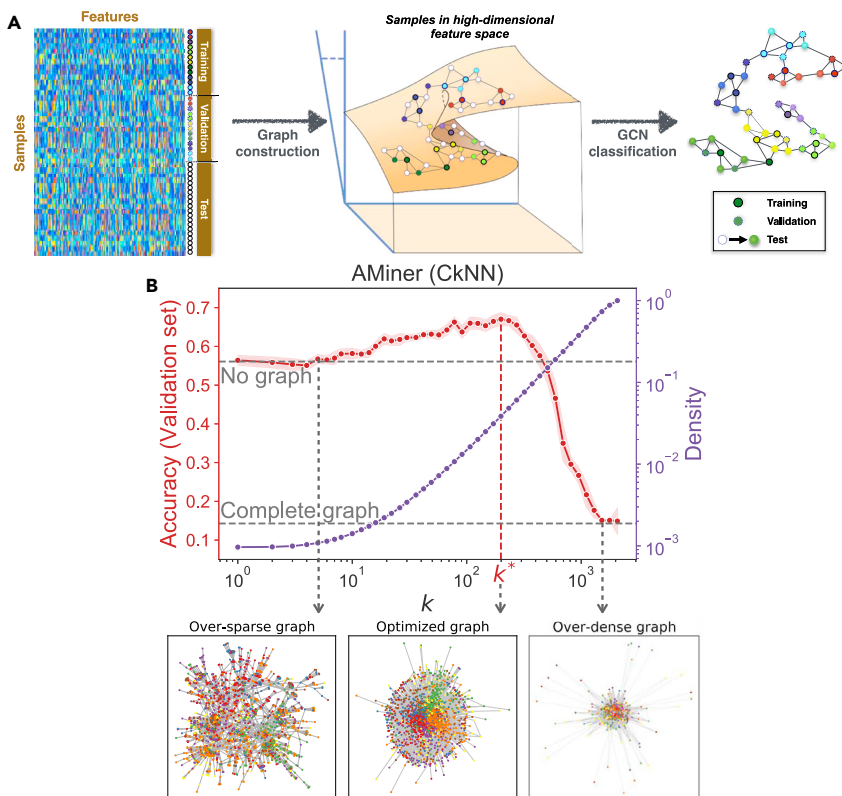
(A) Workflow for GCN classification using feature-derived graphs.

(B) The validation set is used to search for graphs with optimized edge density—here illustrated with the AMiner dataset and CkNN graph construction. In red, the GCN classification accuracy on the validation set as a function of the density parameter, k. The results are averaged over ten runs with random weight initializations; shaded region represents standard deviation. As we sweep k from "no graph" (MLP) to complete graph (mean field, random assignment), the classification accuracy on the validation set exhibits a maximum for a CkNN graph with density parameter k^*. In purple, edge density of the CkNN graphs as k is varied. Figures for all graph constructions and datasets are provided in Figure S1. Also shown below, graph visualizations using the spring layout for over-sparse, optimized, and over-dense graphs, with nodes colored according to their ground truth class.

be used in conjunction with the sample features to achieve the best possible class separation, and hence improved classification. Graphs are a natural way to represent such relational links: the samples are viewed as the nodes of the graph, and the relationships between samples are formalized as edges. A large number of machine learning methods have been proposed to leverage the information in such graph structure. Graph Neural Networks (GNNs) is a nascent class of methods, which refers to a broad set of techniques attempting to extend deep neural models to graph-structured data.[2] GNN has witnessed success in a variety of research domains, including computer vision,[3,4] natural language processing,[5–8] traffic,[9,10] recommendation systems[11,12] chemistry[13,14] and many other areas.[15–20] For an in-depth review of GNNs, see Wu et al.[21]

Recently, work with Graph Convolutional Networks (GCNs)[5] has suggested that using a graph of samples in conjunction with sample features can improve classification performance when compared with traditional methods that use only features. Computationally, the graph allows the definition of a convolution operation that exchanges and aggregates the features of samples that are connected on the graph. If the graph and the features align well with the underlying class structure,[22] the graph convolution operation homogenizes features of neighboring nodes, which will also tend to be more similar, while making less similar samples, which will be more distant on the graph, belong to other classes.

In many instances, extra relational information in the form of a graph is not easily available. However, the intuition that nodes that are close in feature space tend to belong to the same class

can still be exploited by constructing geometric graphs directly from the data features, and in doing so creating neighborhoods of similar samples. Such feature-derived graphs can then be used to aid and potentially sharpen the classification.

Here, we explore the benefit of constructing geometric graphs from the features of the samples and using them within a GCN for sample classification (Figure 1A). Graph construction, or inference, is a problem encountered in many fields,[23] from neuroimaging to genetics, and can be based on many different types of heuristics, from simple thresholding[24] or statistically significant group-level thresholding[25] to sophisticated regularization schemes.[26] In general, the goal is to obtain graphs that concisely preserve key properties of the original dataset as sparsely as possible, i.e., with a low density of edges. In this work, we use several popular geometric graph constructions to extract graphs from data, and study how the classification performance depends on the graph construction method and the edge density. We find that there is a range of relatively low edge densities over which the constructed graphs improve the classification performance. Among the construction methods, we show that the recently proposed Continuous k-Nearest Neighbor (CkNN)[27] performs best for GCN classification. To gain further intuition about the role played by the graph in improving classification, we compute two simple measures: (1) the alignment of the convolution of graph and features with the ground truth and (2) the ratio of class separation in the output activations of the GCN. We show that the optimized geometric graphs increase the alignment and the class separation. Finally, we show that the graphs can be made more efficient using spectral graph

Table 1. Classification accuracy (in percent) on the test set (averaged over ten runs with random initializations) for seven datasets with eight classifiers (four graph-less methods; GCN with four graph constructions)

Classifier	Constructive	Cora	AMiner	Digits	FMA	Cell	Segmentation	Average improvement
MLP = GCN (no graph)	42.1	54.2	54.4	82.0	34.3	79.5	72.0	–
kNNC	31.4	38.2	28.0	88.3	30.6	58.7	68.8	(− 10.6)
SVM	40.0	55.9	51.4	87.7	35.3	81.5	87.7*	(+ 3.0)
RF	36.3	56.1	47.7	83.0	33.0	88.0*	88.8*	(+ 2.1)
GCN (kNN)	53.9*	66.4*	59.2	92.0	35.6*	83.8	83.5	(+ 8.0)
GCN (MkNN)	45.2	64.1	61.8*	93.2*	35.6*	84.0	83.0	(+ 6.9)
GCN (CkNN)	51.1*	66.6*	61.6*	93.4*	36.0*	84.0	83.9	(+ 8.3)
GCN (RMST)	45.9	64.8	61.5	89.3	35.4	84.9*	83.0	(+ 6.6)

The standard deviation is reported in Table S2. The top two results for each dataset are marked with asterisks. Overall, GCN with CkNN graphs displays the best performance. The density parameters of optimized graphs are reported in Table S3.

sparsification,[28] which reduces the edge density of the optimized CkNN graphs while improving further the classification performance.

RESULTS

Geometric graphs constructed from data features can aid sample classification

We consider geometric graph constructions that fall broadly in two groups: (1) three methods based on local neighborhoods, i.e., k-Nearest Neighbor (kNN), Mutual k-Nearest Neighbor (MkNN), and CkNN[27] graphs; and (2) a method that balances local and global distances measured on the Minimum Spanning Tree (MST), i.e., the Relaxed Minimum Spanning Tree (RMST).[29] In all cases, we start from an MST to guarantee that the resulting graph comprises a single connected component, and we add edges based on the corresponding distance heuristics. In each construction, a parameter regulates the edge density of the graph: k in kNN, MkNN, and CkNN, and γ in RMST (see Methods for a full description of the methods).

For each dataset and each graph construction, we find the edge density that maximizes the average GCN classification accuracy on the validation set by sweeping over 50 values of the edge density, from sparse to dense. For each value of the density, we run the GCN classifier 10 times starting from random weight initializations to compute the average accuracy. Note that the two limiting cases are well characterized: the "no graph" limit corresponds to the Multilayer Perceptron (MLP); the "complete graph" limit is equivalent to mean field and leads to random class assignment.[22] Figure 1B shows the classification performance of a GCN with a CkNN graph of increasing density applied to a dataset of computer science papers (AMiner), which we use as our running example throughout. We find that adding relatively sparse graphs improves the classification accuracy, reaching a maximum increase of 10.9% at an edge density of 0.039 ($k^* = 199$) on the validation set. Once the edge density parameter is optimized on the validation set, we apply the GCN classifier to the test set and the test accuracy is recorded. In this case, the GCN yields an improvement of 7.2% in classification accuracy on the test set with respect to MLP, as reported in Table 1.

We have investigated six real-world datasets from different domains, ranging from text (AMiner,[30,31] Cora[32]) to music track features (FMA[33,34]) to single-cell transcriptomics (Cell[35]) to imaging (Digits,[36] Segmentation[37]). We have also studied one constructive dataset with a well-defined ground truth based on a stochastic block model. For a detailed description of the datasets, see Note S1 and Table S1. We have compared the performance of four graph-less, feature-based classifiers (MLP, kNN classification [kNNC], Support Vector Machine [SVM], and Random Forest [RF]) to GCN classifiers with optimized feature-derived geometric graphs (Table 1). Our numerical experiments indicate that the GCNs with feature-derived graphs generally achieve better classification performance than graph-less classifiers. In particular, the CkNN graph construction achieves the highest accuracy improvement (+8.3% on average above MLP) across our seven datasets.

The role of feature-derived graphs in classification

Our results show improved classification performance of GCNs with feature-derived geometric graphs of appropriate edge density. Indeed, over-sparse graphs perform close to MLPs, the no graph limiting case, whereas over-dense graphs are detrimental, as they approach the "mean field" limit that behaves like random class assignment. Hence, there is a sweet spot of relatively low edge density where graphs improve the performance maximally. To gather further insight into the role of the constructed graphs in classification, we explore their properties from two complementary perspectives.

Over-dense graphs degrade the alignment of graph-convolved features with the ground truth

Consider the classification of N samples with F features into C classes making use of a graph with adjacency matrix A. In Qian et al.[22] it was shown that good GCN performance requires a certain degree of alignment between the linear subspaces associated with the matrix of features, $X \in R^{N \times F}$, the adjacency matrix of the graph with self-loops, $\widehat{A} \in R^{N \times N}$, and the ground truth membership matrix, $Y \in R^{N \times C}$ (see Methods for a full description of GCNs). Inspired by Qian et al.,[22] we evaluate the alignment between the ground truth Y and the graph-convolved features $X_A := \widehat{A}X$ as:
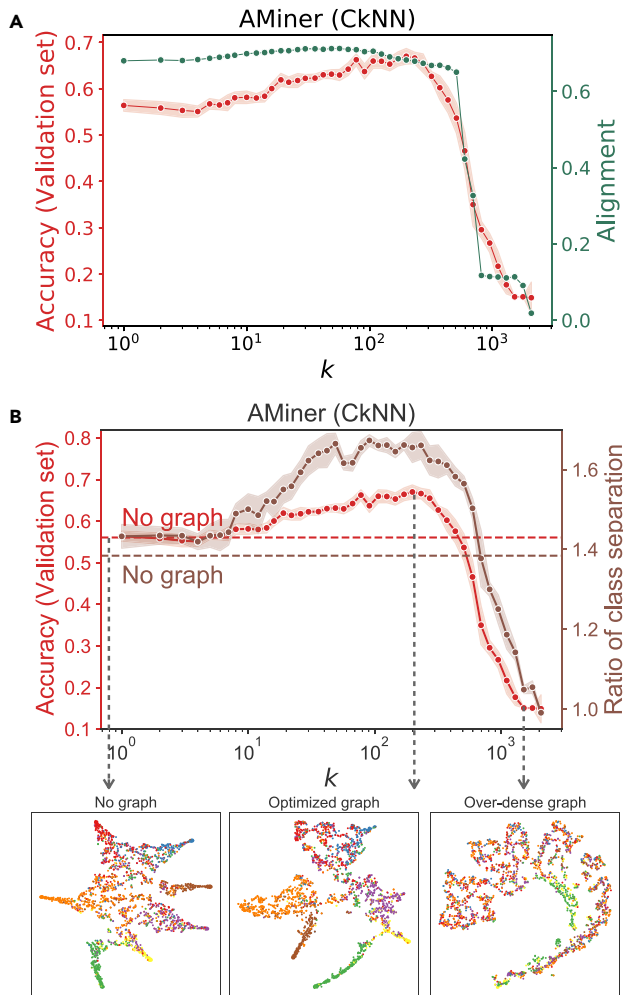
Figure 2. The role of feature-derived graphs in classification

For a Figure360 author presentation of this figure, see https://doi.org/10.1016/j.patter.2021.100237.

(A) In green, we show the alignment (Equation 1) of CkNN graphs for the AMiner dataset as a function of the density parameter k. In red, classification accuracy as in Figure 1B. The drop in classification accuracy corresponds to the drop in the subspace alignment. Results for all graph constructions and datasets are given in Figure S2.

(B) Ratio of class separation (Equation 2) computed from the output activations of the GCN with CkNN graphs for AMiner dataset as a function of the density parameter k, in brown. The results are averaged over ten runs with random weight initializations; shaded region is the standard deviation. The brown dashed line represents the RCS for the MLP, i.e., GCN with no graph. In red, classification accuracy, as in Figure 1B. Below, we show two-dimensional t-SNE projections of the output activations of GCNs with no graph, optimized graph and over-dense graph. The nodes are colored according to the ground truth class labels. The optimized graph induces higher class separability, as shown by an increased RCS and better resolved t-SNE projection. Results for all graph constructions and datasets are provided in Figure S3.

$$S\left(X, \widehat{A}, Y\right) = \cos(\theta_1(\mathcal{X}_{\mathcal{A}}, \mathcal{Y})). \qquad \text{(Equation 1)}$$

Here, $\theta_1(\mathcal{X}_{\mathcal{A}}, \mathcal{Y})$ is the *minimal principal angle*[38–40] between the column spaces of the matrices $\text{PCA}(X_A, p^*)$ and $\text{PCA}(Y, p^*)$, which contain the top principal components, as determined by

the parameter p^*, of $\widehat{A}X$ and Y, respectively. The parameter p^* is the ratio of explained variance that maximizes the Pearson correlation between the alignment (Equation 1) and the classification accuracy on the validation set.

Figure 2A shows the alignment (Equation 1) between the ground truth and the graph-convolved data for CkNN graphs of increasing density on the AMiner dataset. We find that the reduction in classification accuracy induced by over-dense graphs is linked to a strong disruption of the subspace alignment $S(X, \widehat{A}, Y)$. In the limit of the complete graph, the alignment approaches the value of 0, i.e., the minimal angle $\theta_1 = \pi/2$, indicating that the two subspaces are orthogonal. Sparse graphs, on the other hand, induce a slight increase of the subspace alignment at the same time as improving the classification accuracy. The alignment and classification accuracy show good correlation for the AMiner dataset: the Pearson correlation between alignment and accuracy (validation set) is 0.970, obtained for a value of $p^* = 0.4$. The same procedure has been carried out for all seven datasets, and the results are presented in the Figure S2. The Pearson correlation coefficient between alignment and accuracy (validation set) ranges from 0.602 (Segmentation) to 0.970 (AMiner) with an average of 0.852 over all 7 datasets, thus indicating a good correspondence between the classification accuracy and the graph-induced alignment of data and ground truth.

Graphs with optimized density increase the ratio of class separation

Another way of assessing the effect of the constructed graphs on classification is to study the inherent separability of the probabilistic GCN assignment matrix, i.e., the row-stochastic matrix $Z \in R^{N \times C}$ of output activations in Equation 9. The effect of the graph on Z reflects the quality of the classifier: a good graph should enhance the separation of samples from different classes while clustering together samples from the same class in C-dimensional space. We quantify the separability of the GCN mapping using $Z' \in R^{N \times 2}$, the two-dimensional t-SNE[41] embedding of Z, on which we compute the ratio between the average inter-class and intra-class distances, denoted ratio of class separation (RCS):

$$\text{RCS} = \frac{\left(\mathbf{1}^T\left(D^{(Z')} \circ M^{\text{inter}}\right)\mathbf{1}\right) \big/ \left(\mathbf{1}^\mathsf{T} M^{\text{inter}}\mathbf{1}\right)}{\left(\mathbf{1}^T\left(D^{(Z')} \circ M^{\text{intra}}\right)\mathbf{1}\right) \big/ \left(\mathbf{1}^T M^{\text{intra}}\mathbf{1}\right)}. \qquad \text{(Equation 2)}$$

Here, $D^{(Z')}$ is the Euclidean distance matrix for the t-SNE embedding Z', i.e., $D_{ij}^{(Z')} = \left\|Z_i' - Z_j'\right\|_2$; the notation $\circ$ represents the Hadamard, element-wise, matrix product; $M^{\text{inter}} \in R^{N \times N}$ is the inter-class indicator matrix, i.e., $M_{ij}^{\text{inter}} = 1$ if samples i and j belong to different classes and $M_{ij}^{\text{inter}} = 0$; otherwise, and conversely, $M^{\text{intra}} \in R^{N \times N}$ is the intra-class indicator matrix. Compactly, we have

$$M^{\text{inter}} = \mathbf{1}\mathbf{1}^T - YY^T$$
$$M^{\text{intra}} = YY^T - I_N,$$

where $I_N \in R^{N \times N}$ is the identity matrix and $\mathbf{1}$ is the N-dimensional vector of ones.

Figure 2B shows the RCS (Equation 2) computed from the output activation of GCNs with CkNN graphs of increasing density (AMiner dataset). We observe a high correlation between RCS and the classification accuracy (validation set): the Pearson correlation coefficient for AMiner is 0.953. Similar figures for all datasets are shown in the Figure S3. The Pearson correlation coefficient between RCS and accuracy (validation set) is high for all datasets, ranging from 0.876 (Segmentation) to 0.976 (Cora), with an average Pearson correlation coefficient of 0.938 across all seven datasets. These results indicate that sparse graphs unfold the data and facilitate class separation, as illustrated by the t-SNE plots and the increased RCS; on the other hand, overdense graphs reduce separability and eventually converge to the mean field limiting value of RCS $= 1$, i.e., when there is no distinction between inter- and intra-class separation.

Spectral sparsification of optimized geometric graphs can further improve classification

Sparse graphs are generally favored over-dense graphs, in particular for large datasets, as they are more efficient for both numerical computation and data storage. We investigate whether it is possible to sparsify the optimized geometric graphs obtained above, while preserving, or even improving, GCN classification performance. Motivated by the key importance of spectral properties in graph partitioning,[42,43] we apply the Spielman-Srivastava sparsification algorithm (SSSA)[28] to our optimized CkNN graphs. The SSSA reduces the number of edges of a graph while preserving the spectral content of the graph Laplacian given by Equation 8 (see Methods for a full description of the method).

We apply the SSSA to the optimized CkNN and select the sparsification that maximizes the classification accuracy on the validation set. Figure 3A shows that for the AMiner dataset it is possible to improve the classification accuracy using sparser graphs obtained with SSSA. This procedure was repeated for all seven datasets (see Figure S4). For several of our datasets, the sparsified graphs perform better on the test data with reduced edge density (see Figure 3B). The results of the sparsification are robust: starting the sparsification from three different highly optimized CkNN graphs leads to similar results (see Figure S4 and Table S4). Furthermore, the sparsification induces increased alignment and RCS, which correlates with the improved classification accuracy on the validation set (see Figures S5 and S6).

DISCUSSION

Our empirical study used datasets from different domains to show that sparse geometric graphs constructed from data features can aid classification tasks when used within the framework of GNNs. It is worth noting that although here we have used the widely popular GCN framework to perform the classification task, other advanced GNN architectures (e.g., Deep Graph Infomax)[44] could be incorporated in our pipeline as an alternative to GCN for this purpose. In our numerics, GCN with CkNN geometric graphs display the largest improvement in classification accuracy (Table 1). This result is in line with recent work on geometric graph construction for data clustering,[45] which showed improved behavior of CkNN over other neighborhood methods, such as kNN. CkNN graphs have been recently proposed as a consistent discrete approximation of the Lap-
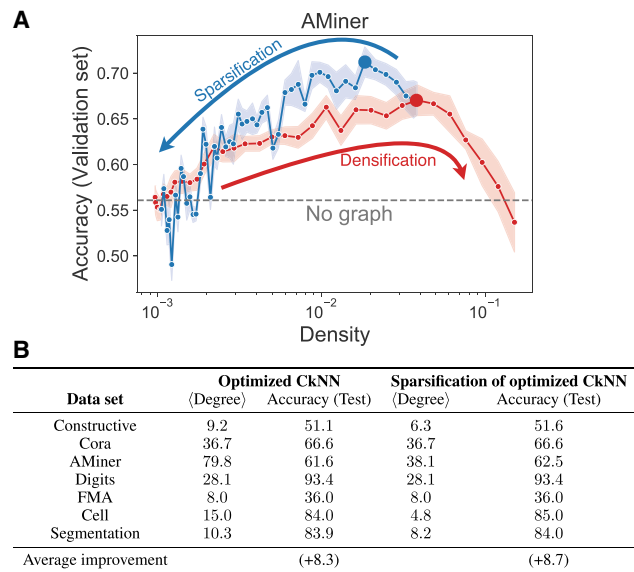

A

B

Data set	Optimized CkNN		Sparsification of optimized CkNN	
	⟨Degree⟩	Accuracy (Test)	⟨Degree⟩	Accuracy (Test)
Constructive	9.2	51.1	6.3	51.6
Cora	36.7	66.6	36.7	66.6
AMiner	79.8	61.6	38.1	62.5
Digits	28.1	93.4	28.1	93.4
FMA	8.0	36.0	8.0	36.0
Cell	15.0	84.0	4.8	85.0
Segmentation	10.3	83.9	8.2	84.0
Average improvement		(+8.3)		(+8.7)

Figure 3. Spectral sparsification of optimized geometric graphs can further improve classification

For a Figure360 author presentation of this figure, see https://doi.org/10.1016/j.patter.2021.100237. [Figure360▷]

(A) In red, the same data as in Figure 1B, i.e., classification accuracy of GCN with CkNN graphs on AMiner dataset for increasing edge density; ten runs with random weight initializations, shaded area is standard deviation. The large red dot indicates the optimized graph found as edges are added (densification). Starting from this optimized graph, we reduce the number of edges using the SSSA (sparsification) and record the classification accuracy on the validation set, in blue; ten runs with random weight initializations, shaded region is standard deviation. The large blue dot indicates the optimized sparsified graph. The gray dashed line corresponds to the classification accuracy of the MLP (no graph) on the validation set. Results for all datasets are provided in Figure S4.

(B) Comparison of optimized graphs obtained through the densification and sparsification processes. The average degree of the graph (⟨Degree⟩) and classification accuracy in percent on the test set are reported; averaged over ten runs with random weight initializations. Overall, sparsified graphs exhibit improved accuracy on the test set with lower edge density.

lace-Beltrami operator governing the diffusion on an underlying manifold.[27] Since GCN uses the graph to guide the diffusion of features to neighboring nodes, this offers a natural explanation for the good performance of CkNN under the GCN framework. Within our graph construction methods, RMST graphs use a criterion that balances neighborhood distances with non-local distances in the dataset. While RMST outperforms graph-less methods, it does not outperform neighborhood-based methods in the examples considered here. However, RMST graphs could be appropriate for datasets where similarities based on longer paths are important.

Intuitively, geometric graphs capture the closeness (i.e., similarity) between samples in feature space, and can thus be helpful to learn and channel class labels from known samples to unseen similar samples. To gain further insight into why geometric graphs can improve GCN classification performance, we showed that the graph induces an increased alignment of features and ground truth, as measured by the simple measure (Equation 1). The alignment correlates well with classification

performance, specifically capturing the deleterious effect that over-dense graphs have on classification performance (Figure 2A). When the graphs are over-dense, they lead to a mean field averaging over the whole dataset, which breaks the alignment—an analogous problem to the over-smoothing observed when there are too many layers in GCNs.[46,47] We also showed that graphs with appropriate density induce increased class separability, as measured by the RCS (Equation 2) derived from the GCN output activations, whereas over-sparse and over-dense graphs lead to lower class separability (Figure 2B). Deviating from strictly geometric graphs, we demonstrated that spectral sparsification (SSSA) applied to the optimized CkNN graphs can be used to reduce the number of edges while still improving the classification performance (Figure 3). Our choice of a spectral criterion for sparsification stems from the fact that the preserved Laplacian quadratic form (Equation 8) is strongly related to graph partitioning and community detection.[42,43] The resulting efficient graphs are thus the product of a mixed process: a geometric graph provides a local similarity neighborhood, which is further sharpened using global graph properties captured by the Laplacian spectrum.

Methods that leverage graphs in data analysis have a long history,[48] and have been recently considered in conjunction with deep-learning algorithms. Franceschi et al.[33] proposed a novel method that jointly learns graph structure and the parameters of a GNN by solving a bilevel program to obtain a discrete probability distribution on the edges of the graph. We have compared our method with the one proposed in Franceschi et al.[33] Our results are summarized in Table S5 and indicate that our proposed method achieves, on average, classification accuracy comparable to Franceschi et al.,[33] yet with a significantly smaller number of parameters, thus simplifying the training and reducing the inclination to overfitting. Table S5 also includes the results[22] obtained by applying GCN to datasets that contain a graph as an additional source of information (i.e., the citation networks for Cora and AMiner). The improved accuracy of GCN with these original graphs stems from the additional information the graphs contain, beyond what is present in the features alone. Specifically, the original graphs for Cora and AMiner collate citations between scientific articles, which encode additional information about the similarity between articles not captured by the features (i.e., the text embedding vectors) of the articles themselves. Another recent method constructed a local neighborhood graph as part of convolution-based classifiers.[49] Our work, in contrast, focuses on graph-theoretical measures,[45] by exploring different graph constructions and the importance of edge density and spectral content for classification, and characterizing the effect of graphs through geometric notions of separability and subspace alignment.[22]

In our numerical experiments, feature-derived geometric graphs appear to be most useful when the data are high-dimensional, noisy, and co-linearity is present in the features. In particular, GCN with optimized graphs outperforms the graph-less methods in all our datasets except "Segmentation." All our datasets are high-dimensional without feature engineering except Segmentation, a dataset with 19 engineered features specifically optimized for classification—this is the setup where SVM and RF are expected to work well. However, even in that case, we note that the featured-derived graphs still improve the classification perfor-

mance with respect to MLP, indicating that the graphs help filter out feature similarities that can obscure the action of MLP.

Beyond the potential to improve performance, using graphs to aid classification changes the paradigm from supervised to semi-supervised learning. Supervised methods, e.g., MLP, perform inductive learning, whereas graph-based semi-supervised learning can be either transductive or inductive. GCNs belong to transductive learning, since the graph of the *full* dataset is used for the training. Therefore, while potentially advantageous, the use of GCNs can also restrict the generalizability to new samples. In many applications such a requirement does not impose severe restrictions, but graph-based methods can still be adapted to classify new data without the need to recompute the model. For instance, one could predict the class label of a new sample directly from the output activations Z of the closest samples in the original set, or using more elaborate diffusion-based schemes.[8]

Our proposed pipeline also shares common ground with some of the most successful clustering methods developed for single-cell genomics datasets. For example, Seurat[50] uses Louvain modularity optimization to perform community detection on a kNN graph constructed from the top principal components of data. Similarly, other methods for graph-based clustering have been introduced using multiscale extensions of the Louvain algorithm in the framework of Markov stability.[45] Although classification and clustering are different learning tasks, we have carried out a comparison between our proposed method (CkNN+GCN) and two Louvain-based clustering methods (Seurat and a straightforward kNN+Louvain clustering). After optimizing each method using the training and validation sets, we computed the assignment it produces on the test set, and compared it with the ground truth classes (see Note S2). The quality of the assignments (evaluated with the Adjusted Rand Index and Normalized Mutual Information) presented in Table S6 indicates that, on average across our datasets, our proposed method performs better than Seurat's approach.

Our study opens several avenues for future work. Here, we explored graph construction based on geometry; it will be interesting to consider graph construction paradigms that incorporate other criteria, e.g., small-worlds,[51] graph expanders,[52] or entangled networks,[53] among others. Similarly, although we showed that spectral sparsification[54] is a good choice to improve efficiency, other graph sparsification paradigms, e.g., cut sparsification,[55] might also be useful to achieve efficient graphs for classification. Here, we have adopted the Euclidean distance as a simple metric to base our geometric graph construction. However, other metrics could be used in our pipeline and could be indeed more appropriate for different types of data. Investigating the effect of different distance metrics (such as the Manhattan distance, cosine similarity, or distances in transformed spaces, such as PCA or other projections), would be an important question for future research.

EXPERIMENTAL PROCEDURES

Resource availability
Lead contact
Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Mauricio Barahona (m.barahona@imperial.ac.uk).

Materials availability

This study did not generate any unique reagents.

Data and code availability

The datasets for graph construction can be found at https://github.com/haczqyf/ggc/tree/master/ggc/data. The code for graph construction can be found at https://github.com/haczqyf/ggc. The sources for other code (e.g., GCN and graph sparsification) are described in Note S3. The algorithm complexity of our proposed pipeline has been discussed in Note S4. The run time and memory requirements have been described in Note S5.

Methods

Graph construction

Let X_i be the F-dimensional feature vector (L1-normalized) of the i-th sample of our dataset with N samples. The pairwise dissimilarity between samples i and j is taken to be the Euclidean distance:

$$d(i,j) = \|X_i - X_j\|_2. \qquad \text{(Equation 3)}$$

The distance matrix of all samples $D \in R^{N \times N}$, where $D_{ij} = d(i,j)$, is then used to construct unweighted and undirected graphs based on different heuristics. To guarantee connectedness over the dataset, we first construct the MST. The MST is obtained from the Euclidean distance matrix D using the Kruskal algorithm, and contains the $N-1$ edges that connect all the nodes (samples) in the graph with minimal sum of edge weights (distances). Once the weighted MST is obtained, we ignore the edge weights, as is also done for all other graphs in the paper. Thus the resulting graphs are undirected and unweighted. The 0-1 adjacency matrix of the MST is denoted by A^{MST}. We then add edges to the MST based on two types of criteria: (1) local neighborhoods or (2) balancing local and global distances.

Methods based on local neighborhoods: Nearest neighbors. The objective of neighborhood-based methods is to construct a sparse graph by connecting two samples if they are local neighbors, as determined by $d(i,j)$.

The simplest such algorithm is kNN. A kNN graph has an edge between two samples i and j if *one* of them belongs to the k-nearest neighbors of the other. The adjacency matrix $A^{kNN} \in R^{N \times N}$ of a kNN graph is defined by:

$$A_{i,j}^{kNN} = \begin{cases} 1 & \text{if } d(i,j) \leq d(i,i_k) \text{ or } d(i,j) \leq d(j,j_k) \\ 0 & \text{otherwise} \end{cases}, \qquad \text{(Equation 4)}$$

where i_k and j_k represent the k-th nearest neighbors of samples i and j, respectively.

Although widely used, kNN has limitations. Perhaps most importantly, kNN graphs can have highly heterogeneous degree distributions and often contain hubs, i.e., samples with high number of connections, since kNN greedily connects two samples as long as one of them belongs to the other's k-nearest neighbors. It has been suggested that the presence of hubs in kNN graphs is particularly severe when the samples are high-dimensional.[56] It has been observed that hubs tend to deteriorate the classification accuracy of semi-supervised learning.[57]

To overcome this limitation, the MkNN algorithm, a variant of kNN, was proposed[57]. In an MkNN graph an edge is established between samples i and j if *each* of them belongs to the other's k-nearest neighbors. The adjacency matrix $A^{MkNN} \in R^{N \times N}$ of the MkNN graph is defined by:

$$A_{i,j}^{MkNN} = \begin{cases} 1 & \text{if } d(i,j) \leq d(i,i_k) \text{ and } d(i,j) \leq d(j,j_k) \\ 0 & \text{otherwise} \end{cases}. \qquad \text{(Equation 5)}$$

Note that the MkNN algorithm guarantees that the degrees of all samples are bounded by k. Therefore, MkNN reduces the presence of hubs when k is adequately small.

Another limitation of kNN is its lack of flexibility to provide a useful, stable graph when the data are not uniformly sampled over the underlying space, which is often the case in practice.[45] In such situations, it is difficult to find a single value of k that can accommodate the disparate levels of sampling density across the data, since the kNN graph will connect samples with very disparate levels of similarity depending on the region of the sample space (i.e., in densely sampled regions, the graph only connects data points that are very similar, whereas in poorly sampled regions, the graph connects data samples that can be quite dissimilar). This disparity biases the training of the GCN. The

non-uniformity of the data distribution thus makes it difficult to tune a unique k parameter that is appropriate across the whole dataset. If the value of k is too small, the graph is dominated by local noise, and fails to provide consistent information to improve the GCN training. If the value of k is large, the resulting graph is over-connected and leads GCN to degraded accuracy, close to mean field classification. Hence, when the sampling is not homogeneous, standard kNN graphs can be sub-optimal in capturing the underlying data structure with a view to improved learning.

CkNN[27] has recently been introduced to address this limitation by allowing an adjusted local density. The adjacency matrix $A^{CkNN} \in R^{N \times N}$ associated with a CkNN graph is defined by:

$$A_{i,j}^{CkNN} = \begin{cases} 1 & \text{if } d(i,j) < \delta\sqrt{d(i,i_k)d(j,j_k)} \\ 0 & \text{otherwise} \end{cases}, \qquad \text{(Equation 6)}$$

where the parameter $\delta > 0$ regulates the density of the graph. For a fixed k, the larger δ is, the denser the CkNN graph becomes. Berry and Sauer[27] show that the CkNN graph captures the geometric features of the dataset with the additional consistency that the unnormalized Laplacian of the CkNN graph converges spectrally to the Laplace-Beltrami operator in the limit of large data. In this work, we fix $\delta = 1$ and vary k so that CkNN can be compared with kNN and MkNN, as suggested in Liu and Barahona.[45]

All these three methods capture the geometry of local neighborhoods, with global connectivity guaranteed by the MST.

Balancing local and global distances: RMST. Alternatively, other graph constructions attempt to balance the local geometry with a measure of global geometry extracted from the full dataset. In recent years, several algorithms have been introduced to explore global properties of the data using the MST.[29,45] Here, we focus on the RMST,[29] which considers the largest distance $d_{MST-path(i,j)}^{max}$ encountered along the unique MST path between i and j. If $d_{MST-path(i,j)}^{max}$ is substantially smaller than $d(i,j)$, RMST discards the direct link between i and j, recognizing the multi-step MST path as a good model to capture the similarity between them. If, on the other hand, $d(i,j)$ is comparable with $d_{MST-path(i,j)}^{max}$, the MST path does not provide a good model, and RMST adds the direct link between i and j. The adjacency matrix $A^{RMST} \in R^{N \times N}$ associated with an RMST graph is defined by:

$$A_{i,j}^{RMST} = \begin{cases} 1 & \text{if } d(i,j) < d_{MST-path(i,j)}^{max} + \gamma(d(i,i_k) + d(j,j_k)) \\ 0 & \text{otherwise} \end{cases}, \qquad \text{(Equation 7)}$$

where $\gamma \geq 0$ is the density parameter, and $d(i,i_k)$ and $d(j,j_k)$ approximate the local distribution of samples around i and j, respectively, as the distance to their kth nearest neighbor.[58] Here, we fix $k = 1$ and vary γ to change the edge density, as in Liu and Barahona.[45]

Spectral graph sparsification

The graph construction methods above can be thought of as a *graph densification*, in which the starting point is the MST over the N samples and an edge is added between two samples i and j if the distance $d(i,j)$ meets a defined criterion. Graph sparsification operates in the opposite direction: starting from a given graph, the goal is to obtain a sparser graph that approximates the original graph so that it can be used, e.g., in numerical computations, without introducing too much error. Sparsified graphs are more efficient for both numerical computation and data storage.[54] Here, we focus on spectral graph sparsification,[54] and apply the seminal SSSA proposed in Spielman and Srivastava.[28] SSSA obtains a spectral approximation of the given graph that satisfies the following criterion:

$$(1 - \sigma)\, x^T L x \leq x^T \tilde{L} x \leq (1 + \sigma)\, x^T L x, \qquad \text{(Equation 8)}$$

where $x \in R^{N \times 1}$ is a node vector, and L and $\tilde{L}$ are the Laplacian matrices of the original and sparsified graphs, respectively. For each dataset, we obtain increasingly sparse versions of the optimized geometric graph computed above by scanning over 50 values of the sparsity parameter σ between $1/N$ and 1. At each value of σ, we run the GCN classifier 10 times starting from random weight initializations and compute the average accuracy over the validation set. We then select the graph with highest accuracy and maximum sparsity. If sparsification does not improve performance on the test set, we report the unsparsified graph as optimal (e.g., in Cora, Digits, and FMA in Figure 3B).

GCNs

GNNs, a new class of deep-learning algorithms, have been recently proposed to analyze graph-structured data. Here, we focus on GCNs[5] and their application to semi-supervised learning. Each sample i is characterized by an F-dimensional feature vector $X_i \in R^{1 \times F}$, which is arranged as a row of the feature matrix $X \in R^{N \times F}$. In addition, the N samples are associated with a graph $\mathcal{G}$ where the samples are the nodes and edges represent relational (symmetric) information. The graph is described by an adjacency matrix $A \in R^{N \times N}$. Each sample is also associated with one of C classes, which is encoded into a 0-1 membership matrix $Y \in R^{N \times C}$. GCNs train a model using the full feature matrix X, the adjacency matrix A of the full graph, and a small subset of ground truth labels, i.e., a few rows of Y. The obtained model is then used to predict the class of unlabeled nodes and evaluate the classification performance by comparing inferred labels with their ground truth labels.

Our study applies the two-layer GCN proposed in Kipf and Welling.[5] Given the feature matrix X and the (undirected) adjacency matrix A of the graph $\mathcal{G}$, the propagation rule is given by:

$$Z = f(X, A) = \text{softmax}\left(\hat{A}\ \text{ReLU}\left(\hat{A}XW^0\right)W^1\right), \quad \text{(Equation 9)}$$

where W^0 and W^1 are the weights connecting the layers of the GCN. The graph is encoded in $\hat{A} = \bar{D}^{-1/2}(A + I_N)\bar{D}^{-1/2}$, where I_N is the identity matrix, and $\bar{D}$ is a diagonal matrix with $\bar{D}_{ii} = 1 + \sum_j A_{ij}$. The softmax and ReLU are threshold activation functions:

$$\text{ReLU}(x)_i = \max(x_i, 0) \quad \text{(Equation 10)}$$

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}, \quad \text{(Equation 11)}$$

where x is a vector. The cross-entropy error over all labeled samples is:

$$\mathcal{L} = -\sum_{l \in \mathbb{Y}_L} \sum_{c=1}^{C} Y_{lc} \ln Z_{lc}, \quad \text{(Equation 12)}$$

where $\mathbb{Y}_L$ is the set of nodes that have labels (i.e., the training set). The weights of the neural network (W^0 and W^1) are trained using gradient descent to minimize the loss $\mathcal{L}$.

In our case, the classification is based solely on information obtained from the features since the graphs are also feature derived.

GCN architecture, hyperparameters, and implementation. We use the GCN implementation provided by the authors of Kipf and Welling,[5] and follow closely the experimental setup in Kipf and co-workers.[5,22] We use a two-layer GCN with $2,000$ epochs (training iterations); learning rate of 0.01; and early stopping with a window size of 200. Other hyperparameters are: dropout rate, 0.5; L2 regularization, 5×10^{-4}; number of hidden units, 16. The weights are initialized as described in Glorot and Bengio,[59] and the input feature vectors are L1 row normalized. We choose the same dataset splits as in Qian et al.,[22] with 5% of samples as training set, 10% of samples as validation set, and the remaining 85% as test set (see Table S1). The samples in the training set are evenly distributed across classes.

Graph-less classification methods. For comparison, we consider four graph-less classification methods: (1) MLP, which is equivalent to GCN with no graph[5,22]; (2) kNNC based on the plurality vote of its k-nearest neighbors; (3) SVM with Radial Basis Function kernel; and (4) RF. We use the Scikit-learn[36] implementation for kNNC, SVM, and RF. For each method and each dataset, we use the validation set to optimize the following hyperparameters: number of neighbors (kNNC); regularization parameter (SVM); maximum depth (RF). All other hyperparameters are set as default in Scikit-learn. We compare the graph-less methods against the MLP = GCN (no graph), which is used as the reference baseline.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.patter.2021.100237.

AUTHOR CONTRIBUTIONS

Y.Q., P.E., P.P., and M.B. designed the research. Y.Q., P.E., P.P., and M.B. performed the research. Y.Q. analyzed the data. Y.Q., P.E., P.P., and M.B. wrote the paper.

REFERENCES

1. LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature *521*, 436–444, https://doi.org/10.1038/NATURE14539.

2. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. IEEE Signal. Process. Mag. *34*, 18–42, https://doi.org/10.1109/MSP.2017.2693418.

3. Xu, D., Zhu, Y., Choy, C.B., and Fei-Fei, L. (2017) Scene graph generation by iterative message passing. In IEEE Conference on Computer Vision and Pattern Recognition.

4. Landrieu, L. and Simonovsky, M. (2018) Large-scale point cloud semantic segmentation with superpoint graphs. In IEEE Conference on Computer Vision and Pattern Recognition.

5. Kipf, T.N. and Welling, M. (2017) Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations.

6. Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems.

7. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018) Graph attention networks. In International Conference on Learning Representations.

8. Peach, R.L., Arnaudon, A., and Barahona, M. (2020). Semi-supervised classification on graphs using explicit diffusion dynamics. Foundations Data Sci. *2*, 19, https://doi.org/10.3934/FODS.2020002.

9. Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018) Diffusion convolutional recurrent neural network: data-driven traffic forecasting. In International Conference on Learning Representations.

10. Yu, B., Yin, H., and Zhu, Z. (2018) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In International Joint Conference on Artificial Intelligence.

11. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., and Leskovec, J. (2018) Graph convolutional neural networks for web-scale recommender systems. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

12. Monti, F., Bronstein, M., and Bresson, X. (2017). Geometric matrix completion with recurrent multi-graph neural networks. In Advances in Neural Information Processing Systems.

13. Duvenaud, D.K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R.P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In Advances in Neural Information Processing Systems.

14. Gainza, P., Sverrisson, F., Monti, F., Rodolà, E., Boscaini, D., Bronstein, M., and Correia, B. (2020). Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. Nat. Methods 17, 184–192, https://doi.org/10.1038/S41592-019-0666-6.

15. Allamanis, M., Brockschmidt, M., and Khademi, M. (2018) Learning to represent programs with graphs. In International Conference on Learning Representations.

16. Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K., and Tang, J. (2018) Deepinf: social influence prediction with deep learning. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

17. Zügner, D., Akbarnejad, A., and Günnemann, S. (2018) Adversarial attacks on neural networks for graph data. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

18. Choi, E., Bahadori, M.T., Song, L., Stewart, W.F., and Sun, J. (2017) GRAM: graph-based attention model for healthcare representation learning. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

19. Choi, E., Xiao, C., Stewart, W., and Sun, J. (2018). Mime: multilevel medical embedding of electronic health records for predictive healthcare. In Advances in Neural Information Processing Systems.

20. Li, Z., Chen, Q., and Koltun, V. (2018). Combinatorial optimization with graph convolutional networks and guided tree search. In Advances in Neural Information Processing Systems.

21. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S.Y. (2020). A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst. https://doi.org/10.1109/TNNLS.2020.2978386.

22. Qian, Y., Expert, P., Rieu, T., Panzarasa, P., and Barahona, M. (2021). Quantifying the alignment of graph and features in deep learning. IEEE Trans. Neural Networks Learn. Syst. https://doi.org/10.1109/TNNLS.2020.3043196.

23. Newman, M. (2018). Network structure from rich but noisy data. Nat. Phys. 14, 542–545, https://doi.org/10.1038/S41567-018-0076-1.

24. Zalesky, A., Fornito, A., and Bullmore, E. (2012). On the use of correlation as a measure of network connectivity. NeuroImage 60 (4), 2096–2106, https://doi.org/10.1016/J.NEUROIMAGE.2012.02.001.

25. Lord, L.-D., Allen, P., Expert, P., Howes, O., Broome, M., Lambiotte, R., Fusar-Poli, P., Valli, I., McGuire, P., and Turkheimer, F.E. (2012). Functional brain networks before the onset of psychosis: a prospective fMRI study with graph theoretical analysis. NeuroImage: Clin. 1, 91–98, https://doi.org/10.1016/J.NICL.2012.09.008.

26. Omranian, N., Eloundou-Mbebi, J.M., Mueller-Roeber, B., and Nikoloski, Z. (2016). Gene regulatory network inference using fused lasso on multiple data sets. Scientific Rep. 6, 20533, https://doi.org/10.1038/SREP20533.

27. Berry, T., and Sauer, T. (2019). Consistent manifold representation for topological data analysis. Foundations Data Sci. 1 (1), 1–38, https://doi.org/10.3934/fods.2019001.

28. Spielman, D.A., and Srivastava, N. (2011). Graph sparsification by effective resistances. SIAM J. Comput. 40, 1913–1926, https://doi.org/10.1137/080734029.

29. Beguerisse-Díaz, M., Vangelov, B., and Barahona, M. (2013). Finding role communities in directed networks using role-based similarity, Markov stability and the relaxed minimum spanning tree. In IEEE Global Conference on Signal and Information Processing.

30. Qian, Y., Rong, W., Jiang, N., Tang, J., and Xiong, Z. (2017). Citation regression analysis of computer science publications in different ranking categories and subfields. Scientometrics 110, 1351–1374, https://doi.org/10.1007/S11192-016-2235-4.

31. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. (2008) ArnetMiner: extraction and mining of academic social networks. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

32. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. AI Mag. 29, 93, https://doi.org/10.1609/AIMAG.V29I3.2157.

33. Franceschi, L., Niepert, M., Pontil, M., and He, X. (2019) Learning discrete structures for Graph Neural Networks. In International Conference on Machine Learning.

34. Defferrard, M., Benzi, K., Vandergheynst, P., and Bresson, X. (2017) FMA: a dataset for music analysis. In International Symposium/Conference on Music Information Retrieval.

35. Velmeshev, D., Schirmer, L., Jung, D., Haeussler, M., Perez, Y., Mayer, S., Bhaduri, A., Goyal, N., Rowitch, D.H., and Kriegstein, A.R. (2019). Single-cell genomics identifies cell type-specific molecular changes in autism. Science 364, 685–689, https://doi.org/10.1126/SCIENCE.AAV8130.

36. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: machine learning in Python. J. Machine Learn. Res. 12, 2825–2830.

37. Dua, D., and Graff, C. (2019). UCI Machine Learning Repository.

38. Björck, A., and Golub, G.H. (1973). Numerical methods for computing angles between linear subspaces. Mathematics Comput. 27, 579–594, https://doi.org/10.1090/S0025-5718-1973-0348991-3.

39. Golub, G.H., and Van Loan, C.F. (2013). Matrix Computations, Vol. 3 (JHU Press).

40. Knyazev, A.V., and Argentati, M.E. (2002). Principal angles between subspaces in an A-based scalar product: algorithms and perturbation estimates. SIAM J. Scientific Comput. 23, 2008–2040, https://doi.org/10.1137/S1064827500377332.

41. Maaten, L.v. d., and Hinton, G. (2008). Visualizing data using t-SNE. J. Machine Learn. Res. 9, 2579–2605.

42. Delvenne, J.-C., Yaliraki, S.N., and Barahona, M. (2010). Stability of graph communities across time scales. Proc. Natl. Acad. Sci. U S A 107, 12755–12760, https://doi.org/10.1073/PNAS.0903215107.

43. Lambiotte, R., Delvenne, J.-C., and Barahona, M. (2014). Random walks, Markov processes and the multiscale modular organization of complex networks. IEEE Trans. Netw. Sci. Eng. 1, 76–90, https://doi.org/10.1109/TNSE.2015.2391998.

44. Velickovic, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., and Hjelm, R.D. (2019) Deep graph infomax. In International Conference on Learning Representations.

45. Liu, Z., and Barahona, M. (2020). Graph-based data clustering via multiscale community detection. Appl. Netw. Sci. 5, 3, https://doi.org/10.1007/S41109-019-0248-7.

46. Li, Q., Han, Z., and Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In AAAI Conference on Artificial Intelligence.

47. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In AAAI Conference on Artificial Intelligence.

48. Lauritzen, S.L. (1996). Graphical Models, Vol. 17 (Clarendon Press).

49. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., and Solomon, J.M. (2019). Dynamic graph CNN for learning on point clouds. ACM Trans. Graphics 38, 1–12, https://doi.org/10.1145/3326362.

50. Satija, R., Farrell, J.A., Gennert, D., Schier, A.F., and Regev, A. (2015). Spatial reconstruction of single-cell gene expression data. Nat. Biotechnol. 33, 495–502, https://doi.org/10.1038/nbt.3192.

51. Watts, D.J., and Strogatz, S.H. (1998). Collective dynamics of 'small-world' networks. Nature 393, 440–442, https://doi.org/10.1038/30918.

52. Hoory, S., Linial, N., and Wigderson, A. (2006). Expander graphs and their applications. Bull. Am. Math. Soc. 43, 439–561, https://doi.org/10.1090/S0273-0979-06-01126-8.

53. Donetti, L., Hurtado, P.I., and Munoz, M.A. (2005). Entangled networks, synchronization, and optimal network topology. Phys. Rev. Lett. 95, 188701, https://doi.org/10.1103/PhysRevLett.95.188701.

54. Spielman, D.A., and Teng, S.-H. (2011). Spectral sparsification of graphs. SIAM J. Comput. *40*, 981–1025, https://doi.org/10.1137/08074489X.

55. Fung, W.-S., Hariharan, R., Harvey, N.J., and Panigrahi, D. (2019). A general framework for graph sparsification. SIAM J. Comput. *48*, 1196–1223, https://doi.org/10.1137/16M1091666.

56. Radovanović, M., Nanopoulos, A., and Ivanović, M. (2010). Hubs in space: popular nearest neighbors in high-dimensional data. J. Machine Learn. Res. *11*, 2487–2531.

57. Ozaki, K., Shimbo, M., Komachi, M., and Matsumoto, Y. (2011) Using the mutual k-nearest neighbor graphs for semi-supervised classification of natural language data. In Conference on Computational Natural Language Learning.

58. Zemel, R.S., and Carreira-Perpiñán, M.Á. (2005). Proximity graphs for clustering and manifold learning. In Advances in Neural Information Processing Systems.

59. Glorot, X. and Bengio, Y. (2010) Understanding the difficulty of training deep feedforward neural networks. In International Conference on Artificial Intelligence and Statistics.