DEEP MANIFOLD GRAPH AUTO-ENCODER FOR ATTRIBUTED GRAPH EMBEDDING

Bozhen Hu[1,2]*, Zelin Zang*[2]*, Jun Xia*[2]*, Lirong Wu*[2]*, Cheng Tan*[2]*, Stan Z. Li*[2⋆]

[1] Zhejiang University, Hangzhou, 310058, China
[2]AI Division, School of Engineering, Westlake University, Hangzhou, 310030, China

ABSTRACT

Representing graph data in a low-dimensional space for subsequent tasks is the purpose of attributed graph embedding. Most existing neural network approaches learn latent representations by minimizing reconstruction errors. Rare work considers the data distribution and the topological structure of latent codes simultaneously, which often results in inferior embeddings in real-world graph data. This paper proposes a novel Deep Manifold (Variational) Graph Auto-Encoder (DMVGAE/DMGAE) method for attributed graph data to improve the stability and quality of learned representations to tackle the crowding problem. The node-to-node geodesic similarity is preserved between the original and latent space under a pre-defined distribution. The proposed method surpasses state-of-the-art baseline algorithms by a significant margin on different downstream tasks across popular datasets, which validates our solutions. We promise to release the code after acceptance.

***Index Terms*—** Manifold learning, structure information, graph embedding, crowding problem

1. INTRODUCTION

Attributed graphs are graphs with node attributes/features that emerge in various real-world applications. Such examples include social networks [1], citation networks [2], protein-protein interaction networks [3], etc. From a data representation perspective, graph embedding is to encode the high-dimensional, non-Euclidean information about the graph structure and attributes associated with the nodes and edges into a low-dimensional embedding space. The learned embeddings can then be used for various data mining tasks, including clustering [4], recommendation [4] and prediction [5].

Early graph embedding approaches are based on Laplacian eigenmaps [6], matrix factorization [7, 8], and random walks [9]. Recently, there has been a surge of approaches focusing on deep learning methods on graphs. Specifically, Graph Convolutional Network (GCN) based methods such as graph auto-encoder (GAE) and variational graph auto-encoder (VGAE) [5], AGC [10], DAEGC [11] and ARGA [12] obtain embeddings by enforcing the reconstruction constraint, have made significant progress in many graph learning tasks [13]. Moreover, GraphMAE [14] focuses on feature reconstruction with a masking strategy to increase its robustness. The task of most existing deep learning methods requires preserving information beneficial to reconstruction and fails to optimize embeddings directly to maintain the topological structure information yet gets inferior representation in many cases. Worse, when the high-dimensional data is mapped into a low-dimensional latent space, a crowding problem [15] may occur. In detail, the embeddings generated from such methods are insufficiently constrained, resulting in the appearance of curvature folds that do not exist in the original data space and eventually distorting the embeddings, which means nodes of the same class are prone to crowd together.

Manifold learning unveils low-dimensional structures from the input data based on the manifold assumption, i.e., data lie on a low-dimensional manifold immersed in the high-dimensional ambient space. Equipped with deep neural networks, deep manifold learning (DML) is transferable to graph data, e.g., MGAE [16] advances the auto-encoder to the graph domain to embed node features and adjacency matrix. Inspired by MGAE and DLME [17], DML can be applied in learning graph embeddings while keeping distances between nodes. Different from them, we relieve and even tackle the crowding problem by preserving the topological structure for latent embeddings of the graph data under a distribution efficiently. Therefore, we propose the Deep Manifold (Variational) Graph Auto-Encoder (DMVGAE/DMGAE) method for attributed graph embedding to improve the representations' stability and quality. The problem of preserving the structure information is converted into keeping inter-node similarity between non-euclidean high dimensional latent space and euclidean input space. For DMVGAE, firstly, we use a variational auto-encoder mechanism to learn the distribution and get codes from it. Secondly, we propose a graph geodesic similarity to collect graph structure information and node feature information to measure node-to-node relationships in the input and latent space. t-distribution is used as a kernel function to fit the neighborhoods between nodes to balance intra-cluster and inter-cluster relationships. Therefore, this method takes advantage of both manifold learning based and auto-encoder-based methods for attributed graph embedding, which is a successful attempt to embed them together, with the reason that we

'$\star$' denotes the corresponding author.

usually think about graphs in terms of their combinatorial properties, variational auto-encoder with the data distribution properties, whereas manifolds in terms of their topological and geometric properties. Thus, our contributions can be summarized as follows:

- Obtain the topological and geometric properties of graph data under a pre-defined distribution, improve the learned representations' stability and quality, and tackle the crowding problem.
- Propose a manifold learning loss that considers graph structure information and node feature information to preserve the observing node-to-node geodesic similarity.
- Achieve state-of-the-art performance on different tasks on benchmark through extensive experiments.

2. METHOD

2.1. Problem Statement

An attributed graph is denoted as $G = (V, E, X)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the vertex set with n nodes, E is the edge set, and $X = [x_1, x_2, \cdots, x_n]^T$ is the feature matrix. Compared with original graph G, attributed graph $\bar{G} = (V, \bar{E}, X)$ is a fully connected graph with no unconnected nodes and $\bar{E}$ is the new edge set. The topology structure of graph G can be denoted by adjacency matrix A:

$$A = \{a_{ij}\} \in \mathbb{R}^{n\times n}, a_{ij} = 1 \text{ if } (v_i, v_j) \in E \text{ else } 0 \quad (1)$$

$a_{i,j} = 1$ indicates that there is an edge from node v_i to node v_j. We aim to find a set of low-dimensional embeddings $Z = [z_1, z_2, \cdots, z_n]^T$ which can relieve and even tackle the crowding problem via preserving both the local and global topological features, the resulting embeddings Z can be used to accomplish a variety of downstream tasks like node clustering, link prediction, and visualization.

2.2. Proposed method

To increase the nonlinearity of node features, we use L Fully-Connected (FC) layers to transform the input feature X into X'. And then a two-layer GCN is adopted as the graph encoder like VGAE, which takes a Gaussian prior $p(Z) = \prod_i p(z_i) = \prod_i \mathcal{N}(z_i|0, I)$. Generating an approximation $q(Z|X', A)$ of the posterior probability by the variational graph encoder, we optimize the variational lower bound:

$$L_0 = \mathbb{E}_{q(Z|(X',A))}[\log p(\hat{A}|Z)] - KL[q(Z|X', A)||p(Z)] \quad (2)$$

where $KL[q(\cdot)||p(\cdot)]$ is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$.

For the non-probabilistic graph auto-encoder, we minimize the reconstruction error of the graph data by:

$$L_1 = \mathbb{E}_{q(Z|(X',A))}[\log p(\hat{A}|Z)] \quad (3)$$

where $\hat{A}$ is the reconstructed graph.

In order to relieve and even tackle the crowding problem, we introduce DML to preserve the geometric structure of the graph G. In detail, we calculate graph geodesic distance matrices on prior graph G_X and on complete graph $\bar{G}_X$ in the input space before training, where the prior graph is the given graph or k-nearest graph $G_X = G = (V, E, X)$, and the complete graph $\bar{G}_X = (V, \bar{E}, X)$, as we want to get local (G_X) and global $(\bar{G}_X)$ structure features from different aspects. In experiments, we find that if we only use the given graph or k-nearest graph G_X, the clustered nodes of the same class still easily get together. In the latent space, we sample K latent embeddings $(Z_0, Z_1, \cdots, Z_i, \cdots, Z_K)$ from $q(Z|X', A)$ and calculate graph geodesic distance matrices only on local graphs $G_{Z_i} = (V, E, Z_i)$ in order to reduce the algorithm complexity and training time. The details are shown in Algorithm.1. These operations differ from other DML-based embedding methods, which contribute to tackling the crowding problem.

for a given graph G_X, we define the graph geodesic distance matrix $D(G_X) = \left\{d_{ij}^{G_X}|i, j = 1, 2, \cdots, n\right\}$, where $d_{ij}^{G_X}$ is calculated from Euclidean distances of (x_i, x_j) when $(v_i, v_j) \in E$, otherwise, $d_{ij}^{G_X}$ are set to be a large number. Similarly, we can get $D(\bar{G}_X)$ and $D(G_{Z_i})$.

In order to avoid the adverse effects of outliers and neighborhood inhomogeneity on the characterization of the manifold, the distance d_{ij} is transformed to $d_{i|j}$:

$$d_{i|j} = d_{ij} - \rho_i \quad (4)$$

where $\rho_i = \min([d_{i0}, d_{i1}, \cdots, d_{in}])$ is deducted from distances of all others nodes to node v_i for alleviating the influence of outliers. After getting a distance matrix, we can use a non-linear function convert it to a similarity matrix. Different from t-SNE [18] and UMAP [19], the former uses the normalized Gaussian and Cauchy functions and the latter uses the fitted polynomial function, here we adopt t-distribution as we find that the degree of freedom ν in t-distribution can be used as a tool to prevent the training from converging to bad local minima and control the separation margin between different manifolds in experiments which means that changing ν can relieve the crowding problem. The similarity is formulated as follows:

$$\begin{aligned} p_{i|j}(\sigma_i, \nu) &= g(d_{i|j}, \sigma_i, \nu) \\ &= C_\nu (1 + \frac{d_{i|j}}{\sigma_i \nu})^{-\frac{(\nu+1)}{2}} \end{aligned} \quad (5)$$

$$C_\nu = \sqrt{2\pi} \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\Gamma(\frac{\nu}{2})} \quad (6)$$

where the degree of freedom $\nu \in \mathbb{R}_+$, $\Gamma(\cdot)$ in the coefficient C_ν is the gamma function, and the data-adaptive parameter $\sigma_i > 0$, is estimated by a binary search method as in the UMAP:

$$\sum_{j\neq i} p_{i|j}(\sigma_i, \nu) = \log_2 Q_p \quad (7)$$

where Q_p is a hyperparameter that controls the compactness of neighbors.

We can get the graph geodesic similarity:

$$P = \{p_{ij}|i, j = 1, 2, \cdots, n\} \tag{8}$$

where p_{ij} is the joint probability by symmetrizing the graph geodesic similarity $p_{i|j}$, which is performed by:

$$p_{ij} = p_{i|j} + p_{j|i} - 2p_{i|j}p_{j|i} \tag{9}$$

Using above equations, we can calculate graph geodesic distance and get node similarity matrices $P(G_X), P(\bar{G}_X)$ and $P(G_{Z_i})$ on different graphs $G_X = G = (V, E, X)$, $\bar{G}_X = (V, \bar{E}, X)$ and $G_{Z_i} = (V, E, Z_i)$ to preserve local and global structure features from different perspectives by a manifold learning loss which is different from graph embedding methods that only take features or structures as input without information maintaining. Here, we adopt a logistic loss:

$$L_{\mathcal{M}}(a, b) = a \log \frac{a}{b} + (1 - a) \log \frac{1 - a}{1 - b} \tag{10}$$

therefore, the loss for structure-preserving is constructed as follows:

$$\begin{aligned} L_2 =& \frac{1}{K} \sum_{i=0}^{K} (L_{\mathcal{M}}(P(G_X), P(G_{Z_i})) + \\ & \alpha L_{\mathcal{M}}(P(\bar{G}_X), P(G_{Z_i}))) \end{aligned} \tag{11}$$

where α is a weight hyper-parameter, K is the number of samples.

Manifold learning loss L_2 has ability to tackle the crowding problem as the distance of each pair of nodes is equally calculated on the fully-connected graph $\bar{G}_X$, while distances of the given graph G_X is only performed on the neighbouring nodes, which are both expected to be preserved with distances on graph G_{Z_i} in the latent space. Because L_2 tries intra-manifold points to transform into a cluster in the latent space and pushes away inter-manifold point pairs from each other. The final loss L (DMVGAE) and L' (DMGAE) is combined as:

$$L = L_2 + \beta L_0 \tag{12}$$

and

$$L' = L_2 + \beta L_1 \tag{13}$$

where β is used to balance the reconstruction loss and the manifold learning loss.

We use the prior graph instead of the fully-connected graph in the latent space as well as a well-established mini-batch-based training method to reduce training complexity. Thus, the complexity is $O(KB_sN_n)$ with N_n neighbours ($N_n < n$), where B_s is the batch size. In experiments, we find that $K = 1$ is enough to learn a good representation without much performance degradation. The complexity is reduced to $O(B_sN_n)$.

Algorithm 1 DMVGAE/DMGAE

Input: Graph with links and features:$G[V, E, X]$, weight hyper-parameters: α, β, number of FC layers: L, number of samples : K, learning rate: l_r, batch size: B_s, epochs: T, perplexity: Q_p, degree of freedom: ν
Output: Graph Embedding Z
Initialization
Calculate $D(G_X)$ and $D(\bar{G}_X)$, transform them to $P(G_X)$ and $P^(\bar{G}_X)$ by Eq.4-Eq.9
while $t = 0, 1, \cdots, T - 1$ **do**
  Get X' from the input data X by FC layers
  Generate an approximation $q(Z|X', A)$ of the posterior probability by the variational graph encoder like VGAE
  Calculate loss L_0 by Eq.2 or loss L_1 by Eq.3 and generate K latent embeddings $Z_0, Z_1, \cdots, Z_K$ from $q(Z|X', A)$
  while $k = 0, 1, \cdots, K - 1$ **do**
    Calculate $D(G_{Z_i})$ and transform it to $P(G_{Z_i})$ by Eq.4-Eq.9
    Get $L_{\mathcal{M}}(P(G_X), P(G_{Z_i})), L_{\mathcal{M}}(P(\bar{G}_X), P(G_{Z_i}))$ by Eq.10
  end while
  Calculate Loss L_2 by Eq.11
  Calculate Loss L by Eq.12 or Loss L' by Eq.13
  Update network parameters with its stochastic gradient
end while

3. EXPERIMENTS

3.1. Datasets and settings

Our experiments are conducted on four popular benchmark datasets: Cora, CiteSeer, PubMed, and Wiki. For all datasets, the degree of freedom ν in the input space is set to 100. We directly set $\sigma_i = 1$ and $\rho_i = 0$ in the latent space as a trade-off between the speed and performance, the necessity of calculating them additionally becomes not so much in the latent space as the layer goes deeper after some nonlinear manifold unfolding. All codes are implemented using the PyTorch library and run on NVIDIA v100 GPU. The best results for each indicator are shown in bold.

Baselines and metrics. We compare our model with several prevalent and concurrent algorithms: DeepWalk [9], K-means [20], various DML and graph embedding methods, GAE and VGAE [5], DGI [21], ARGA [12], AGE [22], GIC [23], MGAE [16], PANE [24]. We employ three metrics for node clustering: Accuracy (ACC), Normalized Mutual Information (NMI), and balanced F1-score (F1) [25]. For link prediction, we partition datasets following AGE [22], and report Area Under Curve (AUC), and Average Precision (AP).

3.2. Results on Node clustering

In the node clustering task, the generated embeddings are clustered into several clusters by the K-means algorithm in

Table 1. Experimental results of node clustering.

Methods	Input	Cora			Citeseer			PubMed			Wiki		
		ACC	NMI	F1	ACC	NMI	F1	ACC	NMI	F1	ACC	NMI	F1
K-means	Feature	0.347	0.167	0.254	0.385	0.170	0.305	0.573	0.291	0.574	0.334	0.302	0.245
DeepWalk	Graph	0.467	0.318	0.381	0.362	0.970	0.267	0.619	0.167	0.471	0.385	0.324	0.257
GAE	Both	0.533	0.407	0.420	0.413	0.183	0.291	0.641	0.230	0.493	0.173	0.119	0.154
VGAE	Both	0.560	0.385	0.415	0.444	0.227	0.319	0.655	0.251	0.510	0.287	0.303	0.205
MGAE	Both	0.634	0.456	0.380	0.636	0.398	0.395	0.439	0.082	0.420	0.501	0.480	0.392
DGI	Both	0.713	0.564	0.682	0.688	0.444	0.657	0.533	0.181	0.186	-	-	-
ARGA	Both	0.640	0.449	0.619	0.573	0.350	0.546	0.591	0.232	0.584	0.414	0.395	0.383
AGE	Both	0.712	0.559	0.682	0.569	0.348	0.544	-	-	-	0.519	0.494	0.408
GIC	Both	0.725	0.537	0.694	0.696	0.453	0.654	0.673	0.319	0.704	0.505	0.486	0.438
Ours(DMGAE)	Both	0.741	0.578	**0.703**	0.698	0.452	0.666	0.752	**0.384**	0.760	0.534	0.493	**0.479**
Ours(DMVGAE)	Both	**0.745**	**0.584**	0.702	**0.701**	**0.459**	**0.668**	**0.758**	0.382	**0.765**	**0.538**	**0.511**	0.476

an unsupervised manner, which are then evaluated by true external labels. Results are shown in Table 1. In order to get a fair comparison, we used K-means for the embeddings of AGE. Algorithms, whether they use feature and graph information or not, are presented. We can see that DMVGAE outperforms almost all of these state-of-the-art methods on the four datasets. The results of DMVGAE are much better than those of VGAE, which indicates that the manifold learning loss L_2 is essential to get better graph embeddings for node clustering.

3.3. Link Prediction Results

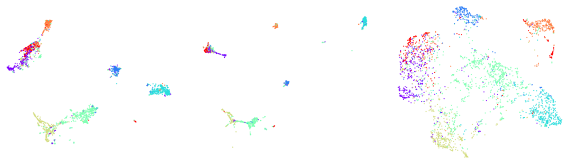
In the link prediction task, some edges are hidden randomly in the input graph and the goal is to predict the existence of hidden edges based on the computed embeddings. We follow the setup in [5]. Results are shown in Table 2. Our proposed method achieves the highest average values on these datasets.

Table 2. Link prediction performance on Cora, Citeseer, and PubMed.

Methods	Cora		Citeseer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
DeepWalk	0.831	0.850	0.805	0.836	0.844	0.841
VGAE	0.914	0.926	0.980	0.920	0.964	0.965
GIC	0.935	0.933	0.970	0.968	0.937	0.935
AGE	0.924	0.932	0.924	0.930	0.968	0.971
PANE	0.933	0.918	0.932	0.919	**0.985**	**0.977**
Ours(DMGAE)	0.966	0.961	0.979	**0.981**	0.965	0.947
Ours(DMVGAE)	**0.968**	**0.977**	**0.981**	0.978	0.968	0.966

3.4. Visualization

To evaluate our proposed method, we visualize the distribution of the learned latent representations on Cora compared to each node's input features in two-dimensional space using UMAP. As shown in Fig. 1, GIC and AGE suffer from a more severe crowding problem, and our proposed method tackles this problem, performing much better.



(a) GIC (b) AGE (c) Proposed

Fig. 1. Comparison of visualization results on Cora using UMAP. Colors represent label categories.

4. CONCLUSION

This paper proposed a deep manifold (variational) graph auto-encoder method (DMVGAE/DMGAE) for attributed graph embedding. Assuming the observed data lies on a low-dimensional manifold and Considering the limitations of current auto-encoder-based methods, we introduce DML to graphs and try to preserve the structure information to tackle the crowding problem for the learned embeddings. Experiments on standard benchmarks demonstrate our solution. For future work, we plan to introduce different types of noise in the given graph, which is significant in real life to prevent attacks and improve model robustness.

5. ACKNOWLEDGMENTS

This work is supported by the Science and Technology Innovation 2030- Major Project (No. 2021ZD0150100) and National Natural Science Foundation of China (No. U21A20427).

6. REFERENCES

[1] Matthew B Hastings, "Community detection as an inference problem," *Physical Review E*, vol. 74, no. 3, pp. 035102, 2006.

[2] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[3] Behnam Neyshabur, Ahmadreza Khadem, Somaye Hashemifar, and Seyed Shahriar Arab, "Netal: a new graph-based method for global alignment of protein–protein interaction networks," *Bioinformatics*, vol. 29, no. 13, pp. 1654–1662, 2013.

[4] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 889–898.

[5] Thomas N Kipf and Max Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[6] Mark EJ Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical review E*, vol. 74, no. 3, pp. 036104, 2006.

[7] Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang, "Community detection in attributed graphs: An embedding approach," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[8] Xiao Wang, Di Jin, Xiaochun Cao, Liang Yang, and Weixiong Zhang, "Semantic community identification in large attribute networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, number 1.

[9] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[10] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu, "Attributed graph clustering via adaptive graph convolution," *arXiv preprint arXiv:1906.01210*, 2019.

[11] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang, "Attributed graph clustering: A deep attentional embedding approach," *arXiv preprint arXiv:1906.06532*, 2019.

[12] Shirui Pan, Ruiqi Hu, Sai-fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang, "Learning graph embedding with adversarial training methods," *IEEE transactions on cybernetics*, vol. 50, no. 6, pp. 2475–2487, 2019.

[13] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[14] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang, "Graphmae: Self-supervised masked graph autoencoders," *knowledge discovery and data mining*, 2022.

[15] Laurens Van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[16] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," *conference on information and knowledge management*, 2017.

[17] Zelin Zang, Siyuan Li, Di Wu, Ge Wang, Lei Shang, Baigui Sun, Hao Li, and Stan Z. Li, "Dlme: Deep local-flatness manifold embedding," 2022.

[18] Laurens van der Maaten and Geoffrey E. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, 2008.

[19] Leland McInnes, John Healy, and James Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[20] Wan-Lei Zhao, Cheng-Hao Deng, and Chong-Wah Ngo, "k-means: A revisit," *Neurocomputing*, 2018.

[21] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm, "Deep graph infomax.," *ICLR (Poster)*, vol. 2, no. 3, pp. 4, 2019.

[22] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu, "Adaptive graph encoder for attributed graph embedding," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 976–985.

[23] Costas Mavromatis and G. Karypis, "Graph infoclust: Maximizing coarse-grain mutual information in graphs," in *PAKDD*, 2021.

[24] Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, Juncheng Liu, and Sourav S. Bhowmick, "Scaling attributed network embedding to massive graphs," *very large data bases*, 2020.

[25] Guojun Gan, Chaoqun Ma, and Jianhong Wu, *Data clustering: theory, algorithms, and applications*, SIAM, 2020.