Structural Deep Clustering Network

Deyu Bo*
Beijing University of Posts and Telecommunications
Beijing, China
bodeyu@bupt.edu.cn

Xiao Wang*
Beijing University of Posts and Telecommunications
Beijing, China
xiaowang@bupt.edu.cn

Chuan Shi†
Beijing University of Posts and Telecommunications
Beijing, China
shichuan@bupt.edu.cn

Meiqi Zhu
Beijing University of Posts and Telecommunications
Beijing, China
zhumeiqi@bupt.edu.cn

Emiao Lu
Tencent
Shenzhen, China
emiao.lu@gmail.com

Peng Cui
Tsinghua University
Beijing, China
cuip@tsinghua.edu.cn

ABSTRACT

Clustering is a fundamental task in data analysis. Recently, deep clustering, which derives inspiration primarily from deep learning approaches, achieves state-of-the-art performance and has attracted considerable attention. Current deep clustering methods usually boost the clustering results by means of the powerful representation ability of deep learning, e.g., autoencoder, suggesting that learning an effective representation for clustering is a crucial requirement. The strength of deep clustering methods is to extract the useful representations from the data itself, rather than the structure of data, which receives scarce attention in representation learning. Motivated by the great success of Graph Convolutional Network (GCN) in encoding the graph structure, we propose a Structural Deep Clustering Network (SDCN) to integrate the structural information into deep clustering. Specifically, we design a delivery operator to transfer the representations learned by autoencoder to the corresponding GCN layer, and a dual self-supervised mechanism to unify these two different deep neural architectures and guide the update of the whole model. In this way, the multiple structures of data, from low-order to high-order, are naturally combined with the multiple representations learned by autoencoder. Furthermore, we theoretically analyze the delivery operator, i.e., with the delivery operator, GCN improves the autoencoder-specific representation as a high-order graph regularization constraint and autoencoder helps alleviate the over-smoothing problem in GCN. Through comprehensive experiments, we demonstrate that our propose model can consistently perform better over the state-of-the-art techniques.

KEYWORDS

deep clustering, graph convolutional network, neural network, self-supervised learning

*Both authors contributed equally to this research.
†Corresponding author

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380214

ACM Reference Format:
Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural Deep Clustering Network. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380214

1 INTRODUCTION

Clustering, one of the most fundamental data analysis tasks, is to group similar samples into the same category [5, 23]. Over the past decades, a large family of clustering algorithms has been developed and successfully applied to various real-world applications, such as image clustering [31] and text clustering [1]. Recently, the breakthroughs in deep learning have led to a paradigm shift in artificial intelligence and machine learning, achieving great success on many important tasks, including clustering. Therefore, the deep clustering has caught significant attention [6]. The basic idea of deep clustering is to integrate the objective of clustering into the powerful representation ability of deep learning. Hence learning an effective data representation is a crucial prerequisite for deep clustering. For example, [30] uses the representation learned by autoencoder in K-means; [4, 29] leverage a clustering loss to help autoencoder learn the data representation with high cluster cohesion [24], and [9] uses a variational autoencoder to learn better data representation for clustering. To date, deep clustering methods have achieved state-of-the-art performance and become the de facto clustering methods.

Despite the success of deep clustering, they usually focus on the characteristic of data itself, and thus seldom take the structure of data into account when learning the representation. Notably, the importance of considering the relationship among data samples has been well recognized by previous literatures and results in data representation field. Such structure reveals the latent similarity among samples, and therefore provides a valuable guide on learning the representation. One typical method is the spectral clustering [23], which treats the samples as the nodes in weighted graph and uses graph structure of data for clustering. Recently, the emerging Graph Convolutional Networks (GCN) [11] also encode both of the graph structure and node attributes for node representation. In summary, the structural information plays a crucial role in data

representation learning. However, it has seldom been applied for deep clustering.

In reality, integrating structural information into deep clustering usually needs to address the following two problems. (1) *What structural information should be considered in deep clustering?* It is well known that the structural information indicates the underlying similarity among data samples. However, the structure of data is usually very complex, i.e., there is not only the direct relationship between samples (also known as first-order structure), but also the high-order structure. The high-order structure imposes the similarity constraint from more than one-hop relationship between samples. Taking the second-order structure as an example, it implies that for two samples with no direct relationship, if they have many common neighbor samples, they should still have similar representations. When the structure of data is sparse, which always holds in practice, the high-order structure is of particular importance. Therefore, only utilizing the low-order structure in deep clustering is far from sufficient, and how to effectively consider higher-order structure is the first problem; (2) *What is the relation between the structural information and deep clustering?* The basic component of deep clustering is the Deep Neural Network (DNN), e.g. autoencoder. The network architecture of autoencoder is very complex, consisting of multiple layers. Each layer captures different latent information. And there are also various types of structural information between data. Therefore, what is the relation between different structures and different layers in autoencoder? One can use the structure to regularize the representation learned by the autoencoder in some way, however, on the other hand, one can also directly learn the representation from the structure itself. How to elegantly combine the structure of data with the autoencoder structure is another problem.

In order to capture the structural information, we first construct a K-Nearest Neighbor (KNN) graph, which is able to reveal the underlying structure of the data[16, 17]. To capture the low-order and high-order structural information from the KNN graph, we propose a GCN module, consisting of multiple graph convolutional layers, to learn the GCN-specific representation.

In order to introduce structural information into deep clustering, we introduce an autoencoder module to learn the autoencoder-specific representation from the raw data, and propose a delivery operator to combine it with the GCN-specific representation. We theoretically prove that the delivery operator is able to assist the integration between autoencoder and GCN better. In particular, we prove that GCN provides an approximate second-order graph regularization for the representation learned by autoencoder, and the representation learned by autoencoder can alleviate the over-smoothing issue in GCN.

Finally, because both of the autoencoder and GCN modules will output the representations, we propose a dual self-supervised module to uniformly guide these two modules. Through the dual self-supervised module, the whole model can be trained in an end-to-end manner for clustering task.

In summary, we highlight the main contributions as follows:

- We propose a novel Structural Deep Clustering Network (SDCN) for deep clustering. The proposed SDCN effectively combines the strengths of both autoencoder and GCN with a novel delivery operator and a dual self-supervised module. To the best of our knowledge, this is the first time to apply structural information into deep clustering explicitly.
- We give a theoretical analysis of our proposed SDCN and prove that GCN provides an approximate second-order graph regularization for the DNN representations and the data representation learned in SDCN is equivalent to the sum of the representations with different-order structural information. Based on our theoretical analysis, the over-smoothing issue of GCN module in SDCN will be effectively alleviated.
- Extensive experiments on six real-world datasets demonstrate the superiority of SDCN in comparison with the state-of-the-art techniques. Specifically, SDCN achieves significant improvements (17% on NMI, 28% on ARI) over the best baseline method.

2 RELATED WORK

In this section, we introduce the most related work: deep clustering and graph clustering with GCN.

Deep clustering methods aim to combine the deep representation learning with the clustering objective. For example, [30] proposes deep clustering network, using the loss function of K-means to help autoencoder learn a "K-means-friendly" data representation. Deep embedding clustering [29] designs a KL-divergence loss to make the representation learned by autoencoder surround the cluster centers closer, thus improving the cluster cohesion. Improved deep embedding clustering [4] adds a reconstruction loss to the objective of DEC as a constraint to help autoencoder learn a better data representation. Variational deep embedding [9] is able to model the data generation process and clusters jointly by using a deep variational autoencoder, so as to achieve better clustering results. [8] proposes deep subspace clustering networks, which uses a novel self-expressive layer between the encoder and the decoder. It is able to mimic the "self-expressiveness" property in subspace clustering, thus obtaining a more expressive representation. DeepCluster [3] treats the clustering results as pseudo labels so that it can be applied in training deep neural network with large datasets. However, all of these methods only focus on learning the representation of data from the samples themselves. Another important information in learning representation, the structure of data, is largely ignored by these methods.

To cope with the structural information underlying the data, some GCN-based clustering methods have been widely applied. For instance, [10] proposes graph autoencoder and graph variation autoencoder, which uses GCN as an encoder to integrate graph structure into node features to learn the nodes embedding. Deep attentional embedded graph clustering [27] uses an attention network to capture the importance of the neighboring nodes and employs the KL-divergence loss from DEC to supervise the training process of graph clustering. All GCN-based clustering methods mentioned above rely on reconstructing the adjacency matrix to update the model, and those methods can only learn data representations from the graph structure, which ignores the characteristic of the data itself. However, the performance of this type of methods might be limited to the overlapping between community structure.

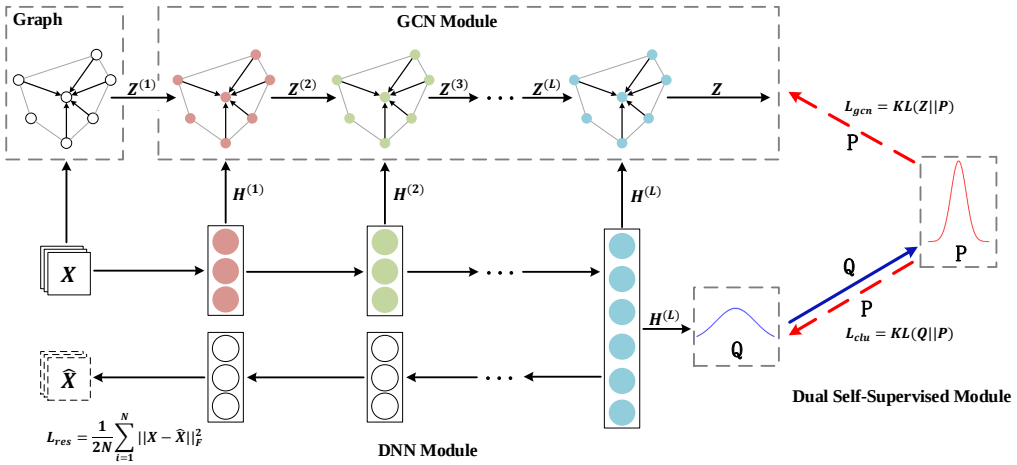


Figure 1: The framework of our proposed SDCN. X, $\hat{X}$ are the input data and the reconstructed data, respectively. $H^{(\ell)}$ and $Z^{(\ell)}$ are the representations in the ℓ-th layer in the DNN and GCN module, respectively. Different colors represent different representations $H^{(\ell)}$, learned the by DNN module. The blue solid line represents that target distribution P is calculated by the distribution Q and the two red dotted lines represent the dual self-supervised mechanism. The target distribution P to guide the update of the DNN module and the GCN module at the same time.

3 THE PROPOSED MODEL

In this section, we introduce our proposed structural deep clustering network, where the overall framework is shown in Figure 1. We first construct a KNN graph based on the raw data. Then we input the raw data and KNN graph into autoencoder and GCN, respectively. We connect each layer of autoencoder with the corresponding layer of GCN, so that we can integrate the autoencoder-specific representation into structure-aware representation by a delivery operator. Meanwhile, we propose a dual self-supervised mechanism to supervise the training progress of autoencoder and GCN. We will describe our proposed model in detail in the following.

3.1 KNN Graph

Assume that we have the raw data $\mathbf{X} \in \mathbb{R}^{N\times d}$, where each row $\mathbf{x}_i$ represents the i-th sample, and N is the number of samples and d is the dimension. For each sample, we first find its top-K similar neighbors and set edges to connect it with its neighbors. There are many ways to calculate the similarity matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ of the samples. Here we list two popular approaches we used in constructing the KNN graph:

1) **Heat Kernel**. The similarity between samples i and j is calculated by:

$$\mathbf{S}_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}}, \tag{1}$$

where t is the time parameter in heat conduction equation. For continuous data, e.g., images.

2) **Dot-product**.The similarity between samples i and j is calculated by:

$$\mathbf{S}_{ij} = \mathbf{x}_j^T \mathbf{x}_i. \tag{2}$$

For discrete data, e.g., bag-of-words, we use the dot-product similarity so that the similarity is related to the number of identical words only.

After calculating the similarity matrix S, we select the top-K similarity points of each sample as its neighbors to construct an undirected K-nearest neighbor graph. In this way, we can get the adjacency matrix A from the non-graph data.

3.2 DNN Module

As we mentioned before, learning an effective data representation is of great importance to deep clustering. There are several alternative unsupervised methods for different types of data to learn representations. For example, denoising autoencoder [26], convolutional autoencoder [21], LSTM encoder-decoder [20] and adversarial autoencoder [19]. They are variations of the basic autoencoder [7]. In this paper, for the sake of generality, we employ the basic autoencoder to learn the representations of the raw data in order to accommodate for different kinds of data characteristics. We assume that there are L layers in the autoencoder and ℓ represents the layer number. Specifically, the representation learned by the ℓ-th layer in encoder part, $\mathbf{H}^{(\ell)}$, can be obtained as follows:

$$\mathbf{H}^{(\ell)} = \phi\left(\mathbf{W}_e^{(\ell)}\mathbf{H}^{(\ell-1)} + \mathbf{b}_e^{(\ell)}\right), \tag{3}$$

where ϕ is the activation function of the fully connected layers such as Relu [22] or Sigmoid function, $\mathbf{W}_e^{(\ell)}$ and $\mathbf{b}_e^{(\ell)}$ are the weight matrix and bias of the ℓ-th layer in the encoder, respectively. Besides, we denote $\mathbf{H}^{(0)}$ as the raw data $\mathbf{X}$.

The encoder part is followed by the decoder part, which is to reconstruct the input data through several fully connected layers by the equation

$$\mathbf{H}^{(\ell)} = \phi\left(\mathbf{W}_d^{(\ell)}\mathbf{H}^{(\ell-1)} + \mathbf{b}_d^{(\ell)}\right), \tag{4}$$

where $\mathbf{W}_d^{(\ell)}$ and $\mathbf{b}_d^{(\ell)}$ are the weight matrix and bias of the ℓ-th layer in the decoder, respectively.

The output of the decoder part is the reconstruction of the raw data $\hat{\mathbf{X}} = \mathbf{H}^{(L)}$, which results in the following objective function:

$$\mathcal{L}_{res} = \frac{1}{2N}\sum_{i=1}^{N}\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 = \frac{1}{2N}||\mathbf{X} - \hat{\mathbf{X}}||_F^2. \tag{5}$$

3.3 GCN Module

Autoencoder is able to learn the useful representations from the data itself, e.g. $\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \cdots, \mathbf{H}^{(L)}$, while ignoring the relationship between samples. In the section, we will introduce how to use the GCN module to propagate these representations generated by the DNN module. Once all the representations learned by DNN module are integrated into GCN, then the GCN-learnable representation will be able to accommodate for two different kinds of information, i.e., data itself and relationship between data. In particular, with the weight matrix $\mathbf{W}$, the representation learned by the ℓ-th layer of GCN, $\mathbf{Z}^{(\ell)}$, can be obtained by the following convolutional operation:

$$\mathbf{Z}^{(\ell)} = \phi(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(\ell-1)}\mathbf{W}^{(\ell-1)}), \tag{6}$$

where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{\mathbf{ij}}$. $\mathbf{I}$ is the identity diagonal matrix of the adjacent matrix $\mathbf{A}$ for the self-loop in each node. As can be seen from Eq. 6, the representation $\mathbf{Z}^{(\ell-1)}$ will propagate through the normalized adjacency matrix $\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$ to obtain the new representation $\mathbf{Z}^{(\ell)}$. Considering that the representation learned by autoencoder $\mathbf{H}^{(\ell-1)}$ is able to reconstruct the data itself and contain different valuable information, we combine the two representations $\mathbf{Z}^{(\ell-1)}$ and $\mathbf{H}^{(\ell-1)}$ together to get a more complete and powerful representation as follows:

$$\widetilde{\mathbf{Z}}^{(\ell-1)} = (1-\epsilon)\mathbf{Z}^{(\ell-1)} + \epsilon\mathbf{H}^{(\ell-1)}, \tag{7}$$

where ϵ is a balance coefficient, and we uniformly set it to 0.5 here. In this way, we connect the autoencoder and GCN layer by layer.

Then we use $\widetilde{\mathbf{Z}}^{(\ell-1)}$ as the input of the l-th layer in GCN to generate the representation $\mathbf{Z}^{(\ell)}$:

$$\mathbf{Z}^{(\ell)} = \phi\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{Z}}^{(\ell-1)}\mathbf{W}^{(\ell-1)}\right). \tag{8}$$

As we can see in Eq. 8, the autoencoder-specific representation $\mathbf{H}^{(\ell-1)}$ will be propagated through the normailized adjacency matrix $\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$. Because the representations learned by each DNN layer are different, to preserve information as much as possible, we transfer the representations learned from each DNN layer into a corresponding GCN layer for information propagation, as in Figure 1. The delivery operator works L times in the whole model. We will theoretically analyze the advantages of this delivery operator in Section 3.5.

Note that, the input of the first layer GCN is the raw data $\mathbf{X}$:

$$\mathbf{Z}^{(1)} = \phi(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}^{(1)}). \tag{9}$$

The last layer of the GCN module is a multiple classification layer with a softmax function:

$$Z = softmax\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(L)}\mathbf{W}^{(L)}\right). \tag{10}$$

The result $z_{ij} \in Z$ indicates the probability sample i belongs to cluster center j, and we can treat Z as a probability distribution.

3.4 Dual Self-Supervised Module

Now, we have connected the autoencoder with GCN in the neural network architecture. However, they are not designed for the deep clustering. Basically, autoencoder is mainly used for data representation learning, which is an unsupervised learning scenario, while the traditional GCN is in the semi-supervised learning scenario. Both of them cannot be directly applied to the clustering problem. Here, we propose a dual self-supervised module, which unifies the autoencoder and GCN modules in a uniform framework and effectively trains the two modules end-to-end for clustering.

In particular, for the i-th sample and j-th cluster, we use the Student's t-distribution [18] as a kernel to measure the similarity between the data representation $\mathbf{h}_i$ and the cluster center vector $\boldsymbol{\mu}_j$ as follows:

$$q_{ij} = \frac{(1 + \left\|\mathbf{h}_i - \boldsymbol{\mu}_j\right\|^2/v)^{-\frac{v+1}{2}}}{\sum_{j'}(1 + \left\|\mathbf{h}_i - \boldsymbol{\mu}_{j'}\right\|^2/v)^{-\frac{v+1}{2}}}, \tag{11}$$

where $\mathbf{h}_i$ is the i-th row of $\mathbf{H}^{(L)}$, $\boldsymbol{\mu}_j$ is initialized by K-means on representations learned by pre-train autoencoder and v are the degrees of freedom of the Student's t-distribution. q_{ij} can be considered as the probability of assigning sample i to cluster j, i.e., a soft assignment. We treat $Q = [q_{ij}]$ as the distribution of the assignments of all samples and let α=1 for all experiments.

After obtaining the clustering result distribution Q, we aim to optimize the data representation by learning from the high confidence assignments. Specifically, we want to make data representation closer to cluster centers, thus improving the cluster cohesion. Hence, we calculate a target distribution P as follows:

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}, \tag{12}$$

where $f_j = \sum_i q_{ij}$ are soft cluster frequencies. In the target distribution P, each assignment in Q is squared and normalized so that the assignments will have higher confidence, leading to the following objective function:

$$\mathcal{L}_{clu} = KL(P||Q) = \sum_i\sum_j p_{ij} log\frac{p_{ij}}{q_{ij}}. \tag{13}$$

By minimizing the KL divergence loss between Q and P distributions, the target distribution P can help the DNN module learn a better representation for clustering task, i.e., making the data representation surround the cluster centers closer. This is regarded as a self-supervised mechanism[1], because the target distribution P is calculated by the distribution Q, and the P distribution supervises the updating of the distribution Q in turn.

As for training the GCN module, one possible way is to treat the clustering assignments as the truth labels [3]. However, this strategy will bring noise and trivial solutions, and lead to the collapse of the whole model. As mentioned before, the GCN module will also

[1] Although some previous work tend to call this mechanism self-training, we prefer to use the term "self-supervised" to be consistent with the GCN training method.

provide a clustering assignment distribution Z. Therefore, we can use distribution P to supervise distribution Z as follows:

$$\mathcal{L}_{gcn} = KL(P||Z) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{z_{ij}}. \tag{14}$$

There are two advantages of the objective function: (1) compared with the traditional multi-classification loss function, KL divergence updates the entire model in a more "gentle" way to prevent the data representations from severe disturbances; (2) both GCN and DNN modules are unified in the same optimization target, making their results tend to be consistent in the training process. Because the goal of the DNN module and GCN module is to approximate the target distribution P, which has a strong connection between the two modules, we call it a dual self-supervised mechanism.

Algorithm 1: Training process of SDCN

Input: Input data: X, Graph: $\mathcal{G}$, Number of clusters: K, Maximum iterations: $MaxIter$;
Output: Clustering results $\mathbf{R}$;

Initialize $\mathbf{W}_e^{(\ell)}, \mathbf{b}_e^{(\ell)}, \mathbf{W}_d^{(\ell)}, \mathbf{b}_d^{(\ell)}$ with pre-train autoencoder;
Initialize $\boldsymbol{\mu}$ with K-means on the representations learned by pre-train autoencoder;
Initialize $\mathbf{W}^{(\ell)}$ randomly;
for $iter \in 0, 1, \cdots, MaxIter$ **do**
  Generate DNN representations $\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \cdots, \mathbf{H}^{(L)}$;
  Use $\mathbf{H}^{(L)}$ to compute distribution Q via Eq. 11;
  Calculate target distribution P via Eq. 12;
  for $\ell \in 1, \cdots, L$ **do**
    Use the delivery operator with ϵ=0.5 $\widetilde{\mathbf{Z}}^{(\ell)} = \frac{1}{2}\mathbf{Z}^{(\ell)} + \frac{1}{2}\mathbf{H}^{(\ell)}$;
    Generate the next GCN layer representation $\mathbf{Z}^{(\ell+1)} = \phi\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{Z}}^{(\ell)}\mathbf{W}_g^{(\ell)}\right)$;
  end
  Calculate the distribution Z via Eq. 10;
  Feed $\mathbf{H}^{(L)}$ to the decoder to construct the raw data $\mathbf{X}$;
  Calculate $\mathcal{L}_{res}$, $\mathcal{L}_{clu}$, $\mathcal{L}_{gcn}$, respectively;
  Calculate the loss function via Eq. 15;
  Back propagation and update parameters in SDCN;
end
Calculate the clustering results based on distribution Z;
return $\mathbf{R}$;

Through this mechanism, SDCN can directly concentrate two different objectives, i.e. clustering objective and classification objective, in one loss function. And thus, the overall loss function of the our proposed SDCN is:

$$\mathcal{L} = \mathcal{L}_{res} + \alpha\mathcal{L}_{clu} + \beta\mathcal{L}_{gcn}, \tag{15}$$

where $\alpha > 0$ is a hyper-parameter that balances the clustering optimization and local structure preservation of raw data and $\beta > 0$ is a coefficient that controls the disturbance of GCN module to the embedding space.

In practice, after training until the maximum epochs, SDCN will get a stable result. Then we can set labels to samples. We choose the soft assignments in distribution Z as the final clustering results. Because the representations learned by GCN contains two different kinds of information. The label assigned to sample i is:

$$r_i = \arg\max_j \ z_{ij}, \tag{16}$$

where z_{ij} is calculated in Eq. 10.

The algorithm of the whole model is shown in Algorithm 1.

3.5 Theory Analysis

In this section, we will analyze how SDCN introduces structural information into the autoencoder. Before that, we give the definition of graph regularization and second-order graph regularization.

Definition 1. *Graph regularization [2]. Given a weighted graph $\mathcal{G}$, the objective of graph regularization is to minimize the following equation:*

$$\sum_{ij} \frac{1}{2} \left\|\mathbf{h}_i - \mathbf{h}_j\right\|_2^2 w_{ij}, \tag{17}$$

where w_{ij} means the weight of the edge between node i and node j, and $\mathbf{h}_i$ is the representation of node i.

Based on Definition 1, we can find that the graph regularization indicates that if there is a larger weight between nodes i and j, their representations should be more similar.

Definition 2. *Second-order similarity. We assume that $\mathbf{A}$ is the adjacency matrix of graph $\mathcal{G}$ and $\mathbf{a}_i$ is the i-th column of $\mathbf{A}$. The second-order similarity between node i and node j is*

$$s_{ij} = \frac{\mathbf{a}_i^T \mathbf{a}_j}{\|\mathbf{a}_i\| \left\|\mathbf{a}_j\right\|} = \frac{\mathbf{a}_i^T \mathbf{a}_j}{\sqrt{d_i}\sqrt{d_j}} = \frac{C}{\sqrt{d_i}\sqrt{d_j}}, \tag{18}$$

where C is the number of common neighbors between node i and node j and d_i is the degree of node i.

Definition 3. *Second-order graph regularization. The objective of second-order graph regularization is to minimize the equation*

$$\sum_{i,j} \frac{1}{2} \left\|\mathbf{h}_i - \mathbf{h}_j\right\|_2^2 s_{ij}, \tag{19}$$

where s_{ij} is the second-order similarity.

Compared with Definition 1, Definition 3 imposes a high-order constraint, i.e., if two nodes have many common neighbors, their representations should also be more similar.

Theorem 1. *GCN provides an approximate second-order graph regularization for the DNN representations.*

Proof. Here we focus on the ℓ-th layer of SDCN. $\mathbf{h}_i$ is the i-th row of $\mathbf{H}^{(\ell)}$, representing the data representation of sample i learned by autoencoder and $\hat{\mathbf{h}}_i = \phi\left(\sum_{j\in\mathcal{N}_i} \frac{\mathbf{h}_j}{\sqrt{d_i}\sqrt{d_j}}\mathbf{W}\right)$ is the representation $\mathbf{h}_i$ passing through the GCN layer. Here we assume that $\phi(x) = x$ and $\mathbf{W} = \mathbf{I}$, and $\hat{\mathbf{h}}_i$ can be seen as the average of neighbor representations. Hence we can divide $\hat{\mathbf{h}}_i$ into three parts: the node representations $\frac{\mathbf{h}_i}{d_i}$, the sum of common neighbor

representations $\mathcal{S} = \sum_{p \in \mathcal{N}_i \cap \mathcal{N}_j} \frac{\mathbf{h}_p}{\sqrt{d_p}}$ and the sum of non-common neighbor representations $\mathcal{D}_i = \sum_{q \in \mathcal{N}_i - \mathcal{N}_i \cap \mathcal{N}_j} \frac{\mathbf{h}_q}{\sqrt{d_q}}$, where $\mathcal{N}_i$ is the neighbors of node i. The distance between the representations $\hat{\mathbf{h}}_i$ and $\hat{\mathbf{h}}_j$ is:

$$\begin{aligned}
&\left\|\hat{\mathbf{h}}_i - \hat{\mathbf{h}}_j\right\|_2 \\
&= \left\|\left(\frac{\mathbf{h}_i}{d_i} - \frac{\mathbf{h}_j}{d_j}\right) + \left(\frac{\mathcal{S}}{\sqrt{d_i}} - \frac{\mathcal{S}}{\sqrt{d_j}}+\right) + \left(\frac{\mathcal{D}_i}{\sqrt{d_i}} - \frac{\mathcal{D}_j}{\sqrt{d_j}}\right)\right\|_2 \\
&\leq \left\|\frac{\mathbf{h}_i}{d_i} - \frac{\mathbf{h}_j}{d_j}\right\|_2 + \left|\frac{\sqrt{d_i} - \sqrt{d_j}}{\sqrt{d_i}\sqrt{d_j}}\right| \|\mathcal{S}\|_2 + \left\|\frac{\mathcal{D}_i}{\sqrt{d_i}} - \frac{\mathcal{D}_j}{\sqrt{d_j}}\right\|_2 \\
&\leq \left\|\frac{\mathbf{h}_i}{d_i} - \frac{\mathbf{h}_j}{d_j}\right\|_2 + \left|\frac{\sqrt{d_i} - \sqrt{d_j}}{\sqrt{d_i}\sqrt{d_j}}\right| \|\mathcal{S}\|_2 + \left(\left\|\frac{\mathcal{D}_i}{\sqrt{d_i}}\right\|_2 + \left\|\frac{\mathcal{D}_j}{\sqrt{d_j}}\right\|_2\right).
\end{aligned} \tag{20}$$

We can find that the first term of Eq. 20 is independent of the second-order similarity. Hence the upper bound of the distance between two node representations is only related to the second and third terms. For the second item of Eq. 20, if $d_i \ll d_j$, $w_{ij} \leq \sqrt{\frac{d_i}{d_j}}$, which is very small and not consistent with the precondition. If $d_i \approx d_j$, the effect of the second item is paltry and can be ignored. For the third item of Eq. 20, if two nodes have many common neighbors, their non-common neighbors will be very few, and the values of $\left\|\frac{\mathcal{D}_i}{\sqrt{d_i}}\right\|_2$ and $\left\|\frac{\mathcal{D}_j}{\sqrt{d_j}}\right\|_2$ are positively correlated with non-common neighbors. If the second-order similarity s_{ij} is large, the upper bound of $\left\|\hat{\mathbf{h}}_i - \hat{\mathbf{h}}_j\right\|_2$ will drop. In an extreme case, i.e. $w_{ij} = 1$, $\left\|\hat{\mathbf{h}}_i - \hat{\mathbf{h}}_j\right\|_2 = \frac{1}{d}\left\|\mathbf{h}_i - \mathbf{h}_j\right\|_2$. □

This shows that after the DNN representations pass through the GCN layer, if the nodes with large second-order similarity, GCN will force the representations of nodes to be close to each other, which is same to the idea of second-order graph regularization.

Theorem 2. *The representation $Z^{(\ell)}$ learned by SDCN is equivalent to the sum of the representations with different order structural information.*

Proof. For the simplicity of the proof, let us assume that $\phi(x) = x$, $\mathbf{b}_e^{(\ell)} = \mathbf{0}$ and $\mathbf{W}_g^{(\ell)} = \mathbf{I}$, $\forall \ell \in [1, 2, \cdots, L]$. We can rewrite Eq. 8 as

$$\begin{aligned}
\mathbf{Z}^{(\ell+1)} &= \widehat{\mathbf{A}}\widetilde{\mathbf{Z}}^{(\ell)} \\
\mathbf{Z}^{(\ell+1)} &= (1-\epsilon)\widehat{\mathbf{A}}\mathbf{Z}^{(\ell)} + \epsilon\widehat{\mathbf{A}}\mathbf{H}^{(\ell)},
\end{aligned} \tag{21}$$

where $\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$. After L-th propagation step, the result is

$$\mathbf{Z}^{(L)} = (1-\epsilon)^L \widehat{\mathbf{A}}^L\mathbf{X} + \epsilon \sum_{\ell=1}^{L} (1-\epsilon)^{\ell-1} \widehat{\mathbf{A}}^\ell \mathbf{H}^{(\ell)}. \tag{22}$$

Note that $\widehat{\mathbf{A}}^L\mathbf{X}$ is the output of standard GCN, which may suffer from the over-smoothing problem. Moreover, if $L \to \infty$, the left term tends to 0 and the right term dominants the data representation. We can clearly see that the right term is the sum of different representations, i.e. $\mathbf{H}^{(\ell)}$, with different order structural information. □

Table 1: The statistics of the datasets.

Dataset	Type	Samples	Classes	Dimension
USPS	Image	9298	10	256
HHAR	Record	10299	6	561
Reuters	Text	10000	4	2000
ACM	Graph	3025	3	1870
DBLP	Graph	4058	4	334
Citeseer	Graph	3327	6	3703

The advantages of the delivery operator in Eq. 7 are two-folds: one is that the data representation $Z^{(\ell)}$ learn by SDCN contains different structural information. Another is that it can alleviate the over-smoothing phenomenon in GCN. Because multilayer GCNs focus on higher-order information, the GCN module in SDCN is the sum of the representations with different order structural information. Similar to [12], our method also uses the fusion of different order information to alleviate the over-smoothing phenomenon in GCN. However, different from [12] treating different order adjacency matrices with the same representations, our SDCN gives different representations to different order adjacency matrices. This makes our model incorporate more information.

3.6 Complexity Analysis

In this work, we denote d as the dimension of the input data and the dimension of each layer of the autoencoder is $d_1, d_2, \cdots, d_L$. The size of weight matrix in the first layer of the encoder is $\mathbf{W}_e^{(1)} \in \mathbb{R}^{d \times d_1}$. N is the number of the input data. The time complexity of the autoencoder is $O(Nd^2d_1^2 \ldots d_L^2)$. As for the GCN module, because the operation of GCN can be efficiently implemented using sparse matrix, the time complexity is linear with the number of edges $|\mathcal{E}|$. The time complexity is $O(|\mathcal{E}|dd_1 \ldots d_L)$. In addition, we suppose that there are K classes in the clustering task, so the time complexity of the Eq. 11 is $O(NK + N\log N)$ corresponding to [29]. The overall time complexity of our model is $O(Nd^2d_1^2 \ldots d_L^2 + |\mathcal{E}|dd_1 \ldots d_L + NK + N\log N)$, which is linearly related to the number of samples and edges.

4 EXPERIMENTS

4.1 Datasets

Our proposed SDCN is evaluated on six datasets. The statistics of these datasets are shown in Table 1 and the detailed descriptions are the followings:

- **USPS**[13]: The USPS dataset contains 9298 gray-scale handwritten digit images with size of 16x16 pixels. The features are the gray value of pixel points in images and all features are normalized to [0, 2].
- **HHAR**[25]: The Heterogeneity Human Activity Recognition (HHAR) dataset contains 10299 sensor records from smart phones and smart watches. All samples are partitioned into 6 categories of human activities, including biking, sitting, standing, walking, stair up and stair down.
- **Reuters**[14]: It is a text dataset containing around 810000 English news stories labeled with a category tree. We use

Table 2: Clustering results on six datasets (mean±std). The bold numbers represent the best results and the numbers with asterisk are the best results of the baselines.

Dataset	Metric	K-means	AE	DEC	IDEC	GAE	VGAE	DAEGC	SDCN$_Q$	SDCN
USPS	ACC	66.82±0.04	71.04±0.03	73.31±0.17	76.22±0.12*	63.10±0.33	56.19±0.72	73.55±0.40	77.09±0.21	**78.08±0.19**
	NMI	62.63±0.05	67.53±0.03	70.58±0.25	75.56±0.06*	60.69±0.58	51.08±0.37	71.12±0.24	77.71±0.21	**79.51±0.27**
	ARI	54.55±0.06	58.83±0.05	63.70±0.27	67.86±0.12*	50.30±0.55	40.96±0.59	63.33±0.34	70.18±0.22	**71.84±0.24**
	F1	64.78±0.03	69.74±0.03	71.82±0.21	74.63±0.10*	61.84±0.43	53.63±1.05	72.45±0.49	75.88±0.17	**76.98±0.18**
HHAR	ACC	59.98±0.02	68.69±0.31	69.39±0.25	71.05±0.36	62.33±1.01	71.30±0.36	76.51±2.19*	83.46±0.23	**84.26±0.17**
	NMI	58.86±0.01	71.42±0.97	72.91±0.39	74.19±0.39*	55.06±1.39	62.95±0.36	69.10±2.28	78.82±0.28	**79.90±0.09**
	ARI	46.09±0.02	60.36±0.88	61.25±0.51	62.83±0.45*	42.63±1.63	51.47±0.73	60.38±2.15	71.75±0.23	**72.84±0.09**
	F1	58.33±0.03	66.36±0.34	67.29±0.29	68.63±0.33	62.64±0.97	71.55±0.29	76.89±2.18*	81.45±0.14	**82.58±0.08**
Reuters	ACC	54.04±0.01	74.90±0.21	73.58±0.13	75.43±0.14*	54.40±0.27	60.85±0.23	65.50±0.13	**79.30±0.11**	77.15±0.21
	NMI	41.54±0.51	49.69±0.29	47.50±0.34	50.28±0.17*	25.92±0.41	25.51±0.22	30.55±0.29	**56.89±0.27**	50.82±0.21
	ARI	27.95±0.38	49.55±0.37	48.44±0.14	51.26±0.21*	19.61±0.22	26.18±0.36	31.12±0.18	**59.58±0.32**	55.36±0.37
	F1	41.28±2.43	60.96±0.22	64.25±0.22*	63.21±0.12	43.53±0.42	57.14±0.17	61.82±0.13	**66.15±0.15**	65.48±0.08
ACM	ACC	67.31±0.71	81.83±0.08	84.33±0.76	85.12±0.52	84.52±1.44	84.13±0.22	86.94±2.83*	86.95±0.08	**90.45±0.18**
	NMI	32.44±0.46	49.30±0.16	54.54±1.51	56.61±1.16*	55.38±1.92	53.20±0.52	56.18±4.15	58.90±0.17	**68.31±0.25**
	ARI	30.60±0.69	54.64±0.16	60.64±1.87	62.16±1.50*	59.46±3.10	57.72±0.67	59.35±3.89	65.25±0.19	**73.91±0.40**
	F1	67.57±0.74	82.01±0.08	84.51±0.74	85.11±0.48	84.65±1.33	84.17±0.23	87.07±2.79*	86.84±0.09	**90.42±0.19**
DBLP	ACC	38.65±0.65	51.43±0.35	58.16±0.56	60.31±0.62	61.21±1.22	58.59±0.06	62.05±0.48*	65.74±1.34	**68.05±1.81**
	NMI	11.45±0.38	25.40±0.16	29.51±0.28	31.17±0.50	30.80±0.91	26.92±0.06	32.49±0.45*	35.11±1.05	**39.50±1.34**
	ARI	6.97±0.39	12.21±0.43	23.92±0.39	25.37±0.60*	22.02±1.40	17.92±0.07	21.03±0.52	34.00±1.76	**39.15±2.01**
	F1	31.92±0.27	52.53±0.36	59.38±0.51	61.33±0.56	61.41±2.23	58.69±0.07	61.75±0.67*	65.78±1.22	**67.71±1.51**
Citeseer	ACC	39.32±3.17	57.08±0.13	55.89±0.20	60.49±1.42	61.35±0.80	60.97±0.36	64.54±1.39*	61.67±1.05	**65.96±0.31**
	NMI	16.94±3.22	27.64±0.08	28.34±0.30	27.17±2.40	34.63±0.65	32.69±0.27	36.41±0.86*	34.39±1.22	**38.71±0.32**
	ARI	13.43±3.02	29.31±0.14	28.12±0.36	25.70±2.65	33.55±1.18	33.13±0.53	37.78±1.24*	35.50±1.49	**40.17±0.43**
	F1	36.08±3.53	53.80±0.11	52.62±0.17	61.62±1.39	57.36±0.82	57.70±0.49	62.20±1.32*	57.82±0.98	**63.62±0.24**

4 root categories: corporate/industrial, government/social, markets and economics as labels and sample a random subset of 10000 examples for clustering.

- **ACM**[2][28]: This is a paper network from the ACM dataset. There is an edge between two papers if they are written by same author. Paper features are the bag-of-words of the keywords. We select papers published in KDD, SIGMOD, SIGCOMM, MobiCOMM and divide the papers into three classes (database, wireless communication, data mining) by their research area.
- **DBLP**[3][28]: This is an author network from the DBLP dataset. There is an edge between two authors if they are the co-author relationship. The authors are divided into four areas: database, data mining, machine learning and information retrieval. We label each author's research area according to the conferences they submitted. Author features are the elements of a bag-of-words represented of keywords.
- **Citeseer**[4]: It is a citation network which contains sparse bag-of-words feature vectors for each document and a list of citation links between documents. The labels contain six area: agents, artificial intelligence, database, information retrieval, machine language, and HCI.

[2]http://dl.acm.org/
[3]https://dblp.uni-trier.de
[4]http://citeseerx.ist.psu.edu/index

4.2 Baselines

We compare our proposed method SDCN with three types of methods, including clustering methods on raw data, DNN-based clustering methods and GCN-based graph clustering methods.

- K-**means** [5]: A classical clustering method based on the raw data.
- **AE** [7]: It is a two-stage deep clustering algorithm which performs K-means on the representations learned by autoencoder.
- **DEC** [29]: It is a deep clustering method which designs a clustering objective to guide the learning of the data representations.
- **IDEC** [4]: This method adds a reconstruction loss to DEC, so as to learn better representation.
- **GAE & VGAE** [10]: It is an unsupervised graph embedding method using GCN to learn data representations.
- **DAEGC** [27]: It uses an attention network to learn the node representations and employs a clustering loss to supervise the process of graph clustering.
- **SDCN**$_Q$: The variant of SDCN with distribution Q.
- **SDCN**: The proposed method.

Metrics. We employ four popular metrics: Accuracy (ACC), Normalized Mutual Information (NMI), Average Rand Index (ARI) and macro F1-score (F1). For each metric, a larger value implies a better clustering result.

Parameter Setting. We use the pre-trained autoencoder for all DNN-based clustering methods (AE+K-means, DEC, IDEC) and

SDCN. We train the autoencoder end-to-end using all data points with 30 epochs and the learning rate is 10^{-3}. In order to be consistent with previous methods [4, 29], we set the dimension of the autoencoder to d-500-500-2000-10, where d is the dimension of the input data. The dimension of the layers in GCN module is the same to the autoencoder. As for the GCN-based methods, we set the dimension of GAE and VAGE to d-256-16 and train them with 30 epochs for all datasets. For DAEGC, we use the setting of [27]. In hyperparameter search, we try $\{1, 3, 5\}$ for the update interval in DEC and IDEC, $\{1, 0.1, 0.01, 0.001\}$ for the hyperparameter γ in IDEC and report the best results. For our SDCN, we uniformly set $\alpha = 0.1$ and $\beta = 0.01$ for all the datasets because our method is not sensitive to hyperparameters. For the non-graph data, we train the SDCN with 200 epochs, and for graph data, we train it with 50 epochs. Because the graph structure with prior knowledge, i.e. citation network, contains more information than KNN graph, which can accelerate convergence speed. The batch size is set to 256 and learning rate is set to 10^{-3} for USPS, HHAR, ACM, DBLP and 10^{-4} for Reuters, Citeseer. For all methods using K-means algorithm to generate clustering assignments, we initialize 20 times and select the best solution. We run all methods 10 times and report the average results to prevent extreme cases.

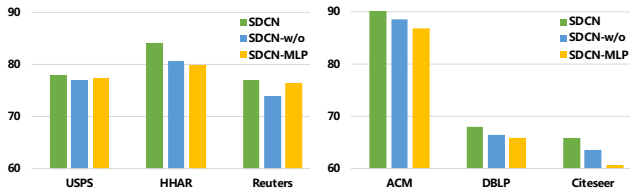
4.3 Analysis of Clustering Results

Table 2 shows the clustering results on six datasets. Note that in USPS, HHAR and Reuters, we use the KNN graph as the input of the GCN module, while for ACM, DBLP and Citeseer, we use the original graph. We have the following observations:

- For each metric, our methods SDCN and SDCN$_Q$ achieve the best results in all the six datasets. In particular, compared with the best results of the baselines, our approach achieves a significant improvement of 6% on ACC, 17% on NMI and 28% on ARI averagely. The reason is that SDCN successfully integrates the structural information into deep clustering and the dual self-supervised module guides the update of autoencoder and GCN, making them enhance each other.
- SDCN generally achieves better cluster results than SDCN$_Q$. The reason is that SDCN uses the representations containing the structural information learned by GCN, while SDCN$_Q$ mainly uses the representations learned by the autoencoder. However, in Reuters, the result of SDCN$_Q$ is much better than SDCN. Because in the KNN graph of Reuters, many different classes of nodes are connected together, which contains much wrong structural information. Therefore, an important prerequisite for the application of GCN is to construct a KNN graph with less noise.
- Clustering results of autoencoder based methods (AE, DEC, IDEC) are generally better than those of GCN-based methods (GAE, VAGE, DAEGC) on the data with KNN graph, while GCN-based methods usually perform better on the data with graph structure. The reason is that GCN-based methods only use structural information to learn the data representation. When the structural information in the graph is not clear enough, e.g. KNN graph, the performance of the GCN-based methods will decline. Besides, SDCN integrates structural information into deep clustering, so its clustering performance is better than these two methods.
- Comparing the results of AE with DEC and the results of GAE with DAEGC, we can find that the clustering loss function, defined in Eq. 13, plays an important role in improving the deep clustering performance. Because IDEC and DAEGC can be seen as the combination of the clustering loss with AE and GAE, respectively. It improves the cluster cohesion by making the data representation closer to the cluster centers, thus improving the clustering results.

4.4 Analysis of Variants

We compare our model with two variants to verify the ability of GCN in learning structural information and the effectiveness of the delivery operator. Specifically, we define the following variants:



(a) Datasets with KNN graph (b) Datasets with original graph

Figure 2: Clustering accuracy with different variants

- **SDCN-w/o:** This variant is SDCN without delivery operator, which is used to validate the effectiveness of our proposed delivery operator.
- **SDCN-MLP:** This variant is SDCN replacing the GCN module with the same number of layers of multilayer perceptron (MLP), which is used to validate the advantages of GCN in learning structural information.

From Figure 2, we have the following observations:

- In Figure 2(a), we can find that the clustering accuracy of SDCN-MLP is better than SDCN-w/o in Reuters and achieves similar results in USPS and HHAR. This shows that in the KNN graph, without delivery operator, the ability of GCN in learning structural information is severely limited. The reason is that multilayer GCN will produce a serious over-smoothing problem, leading to the decrease of the clustering results. On the other hand, SDCN is better than SDCN-MLP. This proves that the delivery operator can help GCN alleviate the over-smoothing problem and learn better data representation.
- In Figure 2(b), we can find that the clustering accuracy of SDCN-w/o is better than SDCN-MLP in all three datasets containing original graph. This shows that GCN has the powerful ability in learning data representation with structural information. Besides, SDCN performs better than SDCN-w/o in the three datasets. This proves that there still exists over-smoothing problem in the SDCN-w/o, but the good graph structure still makes SDCN-w/o achieve not bad clustering results.

Table 3: Effect of different propagation layers (L)

		ACC	NMI	ARI	F1
ACM	SDCN-4	**90.45**	**68.31**	**73.91**	**90.42**
	SDCN-3	89.06	64.86	70.51	89.03
	SDCN-2	89.12	66.48	70.94	89.04
	SDCN-1	77.69	51.59	50.13	74.62
DBLP	SDCN-4	**68.05**	**39.51**	**39.15**	**67.71**
	SDCN-3	65.11	36.81	36.03	64.98
	SDCN-2	66.72	37.19	37.58	65.37
	SDCN-1	64.19	30.69	33.62	60.44
Citeseer	SDCN-4	**65.96**	**38.71**	**40.17**	**61.62**
	SDCN-3	59.18	32.11	32.16	55.92
	SDCN-2	60.96	33.69	34.49	57.31
	SDCN-1	58.58	32.91	32.31	52.38

- Comparing the results in Figure 2(a) and Figure 2(b), we can find no matter on which types of datasets, SDCN achieves the best performance, compared with SDCN-w/o and SDCN-MLP. This proves that both the delivery operator and GCN play an important role in improving clustering quality.

4.5 Analysis of Different Propagation Layers

To investigate whether SDCN benefits from multilayer GCN, we vary the depth of the GCN module while keeping the DNN module unchanged. In particular, we search the number of layers in the range of $\{1, 2, 3, 4\}$. There are a total of four layers in the encoder part of the DNN module in SDCN, generating the representation $\mathbf{H}^{(1)}$, $\mathbf{H}^{(2)}$, $\mathbf{H}^{(3)}$, $\mathbf{H}^{(4)}$, respectively. SDCN-L represents that there is a total of L layers in the GCN module. For example, SDCN-2 means that $\mathbf{H}^{(3)}$, $\mathbf{H}^{(4)}$ will be transferred to the corresponding GCN layers for propagating. We choose the datasets with original graph to verify the effect of the number of the propagation layers on the clustering effect because they have the nature structural information. From Table 3, we have the following observations:

- Increasing the depth of SDCN substantially enhances the clustering performance. It is clear that SDCN-2, SDCN-3 and SDCN-4 achieve consistent improvement over SDCN-1 in all the across. Besides, SDCN-4 performs better than other methods in all three datasets. Because the representations learned by each layer in the autoencoder are different, to preserve information as much as possible, we need to put all the representations learned from autoencoder into corresponding GCN layers.
- There is an interesting phenomenon that the performance of SDCN-3 is not as good as SDCN-2 in all the datasets. The reason is that SDCN-3 uses the representation $\mathbf{H}^{(2)}$, which is a middle layer of the encoder. The representation generated by this layer is in the transitional stage from raw data to semantic representation, which inevitably loses some underlying information and lacks of semantic information. Another reason is that GCN with two layers does not cause serious over-smoothing problems, proved in [15]. For SDCN-3, due to the number of layers is not enough, the over-smoothing term in Eq. 22 is not small enough so that it is still plagued by the over-smoothing problems.

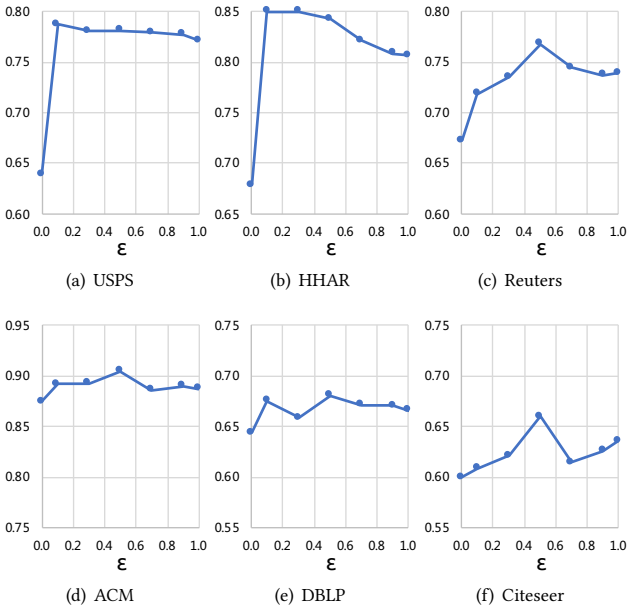
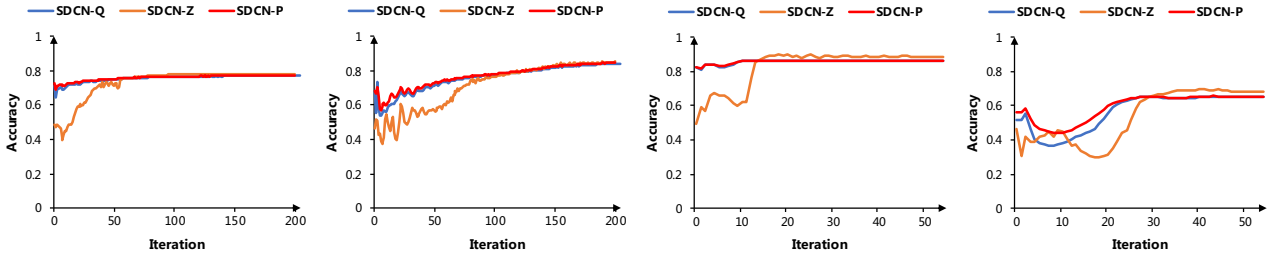


Figure 3: Clustering accuracy with different ϵ

4.6 Analysis of balance coefficient ϵ

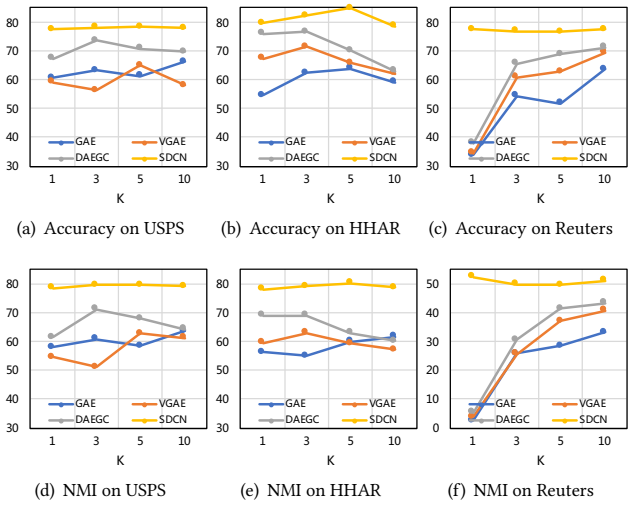
In previous experiments, in order to reduce hyperparameter search, we uniformly set the balance coefficient ϵ to 0.5. In this experiment, we will explore how SDCN is affected by different ϵ on different datasets. In detail, we set $\epsilon = \{0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$. Note that $\epsilon = 0.0$ means the representations in GCN module do not contain the representation from autoencoder and $\epsilon = 1.0$ represents that GCN only use the representation $\mathbf{H}^{(L)}$ learned by DNN. From Figure 4, we can find:

- Clustering accuracy with parameter $\epsilon = 0.5$ in four datasets (Reuters, ACM, DBLP, Citeseer) achieve the best performance, which shows that the representations of GCN module and DNN module are equally important and the improvement of SDCN depends on the mutual enhancement of the two modules.
- Clustering accuracy with parameter $\epsilon = 0.0$ in all datasets performs the worst. Clearly, when $\epsilon = 0.0$, the GCN module is equivalent to the standard multilayer GCN, which will produce very serious over-smoothing problem [15], leading to the decline of the clustering quality. Compared with the accuracy when $\epsilon = 0.1$, we can find that even injecting a small amount of representations learned by autoencoder into GCN can help alleviate the over-smoothing problem.
- Another interesting observation is that SDCN with parameter $\epsilon = 1.0$ still gets a higher clustering accuracy. The reason is that although SDCN with parameter $\epsilon = 1.0$ only use the representation $\mathbf{H}^{(L)}$, it contains the most important information of the raw data. After passing through one GCN layer,



(a) USPS (b) HHAR (c) ACM (d) DBLP

Figure 4: Training process on different datasets



(a) Accuracy on USPS (b) Accuracy on HHAR (c) Accuracy on Reuters

(d) NMI on USPS (e) NMI on HHAR (f) NMI on Reuters

Figure 5: Clustering results with different K

it can still achieve some structural information to improve clustering performance. However, due to the limitation of the number of layers, the results are not the best.

4.7 K-sensitivity Analysis

Since the number of the nearest neighbors K is an important parameter in the construction of the KNN graph, we design a K-sensitivity experiment on the datasets with KNN graph. This experiment is mainly to prove that our model is K-insensitive. Hence we compare SDCN with the clustering methods focusing on the graph data (GAE, VGAE, DAEGC). From Figure 5, we can find that with K={1, 3, 5, 10}, our proposed SDCN is much better than GAE, VGAE and DAEGC, which proves that our method can learn useful structural information even in the graphs containing noise. Another finding is that these four methods can achieve good performance when K = 3 or K = 5, but in the case of K = 1 and K = 10, the performance will drop significantly. The reason is that when K = 1, the KNN graph contains less structural information and when K = 10, the communities in KNN graph are over-lapping. In summary, SDCN can achieve stable results compared with other baseline methods on the KNN graphs with different number of nearest neighbors.

4.8 Analysis of Training Process

In this section, we analyze the training progress in different datasets. Specifically, we want to explore how the cluster accuracy of the three sample assignments distributions in SDCN varies with the number of iterations. In Figure 4, the red line SDCN-P, the blue line SDCN-Q and the orange line SDCN-Z represent the accuracy of the target distribution P, distribution Q and distribution Z, respectively. In most cases, the accuracy of SDCN-P is higher than that of SDCN-Q, which shows that the target distribution P is able to guide the update of the whole model. At the beginning, the results of the accuracy of three distributions all decrease in different ranges. Because the information learned by autoencoder and GCN is different, it may rise a conflict between the results of the two modules, making the clustering results decline. Then the accuracy of SDCN-Q and SDCN-Z quickly increase to a high level, because the target distribution SDCN-P eases the conflict between the two modules, making their results tend to be consistent. In addition, we can see that with the increase of training epochs, the clustering results of SDCN tend to be stable and there is no significant fluctuation, indicating the good robustness of our proposed model.

5 CONCLUSION

In this paper, we make the first attempt to integrate the structural information into deep clustering. We propose a novel structural deep clustering network, consisting of DNN module, GCN module, and dual self-supervised module. Our model is able to effectively combine the autoencoder-specific representation with GCN-specific representation by a delivery operator. Theoretical analysis is provided to demonstrate the strength of the delivery operator. We show that our proposed model consistently outperforms the state-of-the-art deep clustering methods in various open datasets. The code is available at https://github.com/461054993/SDCN.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2018YFB1402600) and the National Natural Science Foundation of China (No. 61772082, 61702296, 61806020, 61972442, U1936104). It is also supported by 2018 Tencent Marketing Solution Rhino-Bird Focused Research Program.

REFERENCES

[1] Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining text data*. Springer, 77–128.

[2] Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15, 6 (2003), 1373–1396.

[3] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *ECCV*. 132–149.

[4] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved deep embedded clustering with local structure preservation.. In *IJCAI*. 1753–1759.

[5] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, 1 (1979), 100–108.

[6] William Grant Hatcher and Wei Yu. 2018. A survey of deep learning: platforms, applications and emerging research trends. *IEEE Access* 6 (2018), 24411–24432.

[7] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

[8] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. 2017. Deep subspace clustering networks. In *NIPS*. 24–33.

[9] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. In *IJCAI*. 1965–1972.

[10] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[11] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[12] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

[13] Yann Le Cun, Ofer Matan, Bernhard Boser, John S Denker, Don Henderson, Richard E Howard, Wayne Hubbard, LD Jacket, and Henry S Baird. 1990. Handwritten zip code recognition with multilayer networks. In *ICPR*, Vol. 2. IEEE, 35–40.

[14] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5, Apr (2004), 361–397.

[15] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

[16] Weiwei Liu and Ivor W Tsang. 2017. Making decision trees feasible in ultrahigh feature and label dimensions. *Journal of Machine Learning Research* 18, 1 (2017), 2814–2849.

[17] Weiwei Liu, Donna Xu, Ivor W Tsang, and Wenjie Zhang. 2018. Metric learning for multi-output tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 2 (2018), 408–422.

[18] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[19] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).

[20] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148* (2016).

[21] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In *ICANN*. Springer, 52–59.

[22] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*. 807–814.

[23] Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *NIPS*. 849–856.

[24] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[25] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *SenSys*. 127–140.

[26] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*. 1096–1103.

[27] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *IJCAI*. 3670–3676.

[28] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[29] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*. 478–487.

[30] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*. 3861–3870.

[31] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. 2010. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing* 19, 10 (2010), 2761–2773.