Learning Deep Representations for Graph Clustering

Fei Tian*
University of Science and Technology of China
tianfei@mail.ustc.edu.cn

Bin Gao
Microsoft Research
bingao@microsoft.com

Qing Cui*
Tsinghua University
cuiqing1989@gmail.com

Enhong Chen
University of Science and Technology of China
cheneh@ustc.edu.cn

Tie-Yan Liu
Microsoft Research
tyliu@microsoft.com

Abstract

Recently deep learning has been successfully adopted in many applications such as speech recognition and image classification. In this work, we explore the possibility of employing deep learning in graph clustering. We propose a simple method, which first learns a nonlinear embedding of the original graph by stacked autoencoder, and then runs k-means algorithm on the embedding to obtain clustering result. We show that this simple method has solid theoretical foundation, due to the similarity between autoencoder and spectral clustering in terms of what they actually optimize. Then, we demonstrate that the proposed method is more efficient and flexible than spectral clustering. First, the computational complexity of autoencoder is much lower than spectral clustering: the former can be linear to the number of nodes in a sparse graph while the latter is super quadratic due to eigenvalue decomposition. Second, when additional sparsity constraint is imposed, we can simply employ the sparse autoencoder developed in the literature of deep learning; however, it is non-straightforward to implement a sparse spectral method. The experimental results on various graph datasets show that the proposed method significantly outperforms conventional spectral clustering, which clearly indicates the effectiveness of deep learning in graph clustering.

1 Introduction

Deep learning has been a hot topic in the communities of machine learning and artificial intelligence. Many algorithms, theories, and large-scale training systems towards deep learning have been developed and successfully adopted in real tasks, such as speech recognition (Dahl et al. 2012), image classification (Krizhevsky, Sutskever, and Hinton 2012), and natural language processing (Collobert et al. 2011). However, to our knowledge, the adoption of deep learning in clustering has not been adequately investigated yet. The goal of this work is to conduct some preliminary investigations along this direction.

Clustering aims to group similar patterns among massive data points. Graph clustering is a key branch of clustering, which tries to find disjoint partitions of graph nodes such that the connections between nodes within the same partition are much denser than those across different partitions. On one hand, many real-world problems can be cast as graph clustering such as image segmentation (Shi and Malik 2000), community detection (Smyth and White 2005), and VLSI design (Chan, Schlag, and Zien 1994); on the other hand, it is easy to transform a clustering problem in the vector space to a clustering problem on the similarity graph built from the vector representations of the data points. Therefore, we choose to put our focus on graph clustering in this work, and in particular, we investigate the use of stacked sparse autoencoder to perform graph clustering.

Our proposal is motivated by the similarity between autoencoder and spectral clustering, a state-of-the-art graph clustering method, in terms of what they actually optimize. Among many existing graph clustering algorithms (Karypis and Kumar 1998)(Shi and Malik 2000) (Van Dongen 2000) (Dhillon, Guan, and Kulis 2007)(Satuluri and Parthasarathy 2009), spectral clustering has attracted people's great attention in the past decades due to its solid theoretical foundation and global optimal solution. Given an n-node graph, spectral clustering method runs an Eigenvalue Decomposition (EVD) on the normalized graph Laplacian matrix; then the eigenvectors corresponding to the k smallest non-zero eigenvalues are extracted as the representation of the graph nodes, where k is the predefined number of clusters; [1] after that, a k-means method is run on the graph representations to get the clusters results.

Note that these k eigenvectors are also the eigenvectors of the normalized graph similarity matrix, whereas corresponding to its k largest eigenvalues. Therefore these eigenvectors can be regarded as an encoding of the normalized graph similarity matrix, and according to the Eckart-Young-Mirsky theorem, this encoding can lead to the optimal rank-k reconstruction of the original normalized graph similarity matrix. People familiar with autoencoder may immediately realize that spectral clustering is very similar to autoencoder: autoencoder also attempts to find a low-dimensional encoding of the input data that can keep the most information of the original set of data through reconstruction. Actually, our the-

*This work was done when the two authors were visiting Microsoft Research Asia.

[1] Some researchers also suggest selecting the eigenvectors corresponding to the $\lfloor\sqrt{k}\rfloor$ smallest non-zero eigenvalues.

oretical study shows that the objective functions of spectral clustering and autoencoder are actually very similar when they are used to solve graph clustering problems (i.e., the reconstruction error of the normalized similarity matrix under the Frobenuis norm by the encoding), and both of them can achieve the desired solution when the optimization processes are well done.[2]

While autoencoder is similar to spectral clustering in theory, it is much more efficient and flexible in practice. First, as we know, spectral clustering is computationally expensive because it involves an eigenvalue decomposition (EVD). The complexity of a straightforward implementation of EVD is cubic to the number of nodes in the graph, while even the fastest implementation to our knowledge requires a super-quadratic computational complexity. In contrast, autoencoder is highly efficient due to its back-propagation framework, whose computational complexity can be linear to the number of nodes in the graph when the nodes are sparsely connected. Second, When dealing with very large-scale data, we usually hope to get some sparse representations so as to improve the efficiency of the data processing. However, the encoding produced by spectral clustering cannot guarantee the sparsity property because the eigenvectors of the graph Laplacian are probably very dense. In addition, it is non-straightforward to incorporate sparsity constraints into spectral clustering without destroying its nature of being a spectral method. In contrast, it is very easy to fulfill the sparsity requirement by using the autoencoder. Actually, one can simply use the sparse autoencoder developed in the literature of deep learning for this purpose, which basically introduces a L_1 regularization term to the original objective function of autoencoder. Furthermore, one can stack multiple layers of sparse autoencoders, to achieve additional benefit from deep structures.

Based on the above discussions, we propose a method called *GraphEncoder* for graph clustering. First, we feed the normalized graph similarity matrix into a deep neural network (DNN) which takes sparse autoencoder as the building block. Then through a greedy layer-wise pretraining process, we seek the best non-linear graph representations that can approximate the input matrix through reconstruction and achieve the desired sparsity properties. After stacking several layers of sparse autoencoders, we run k-means on the sparse encoding output by the final layer to obtain the clustering results. To verify the effectiveness of the proposed method, we conducted extensive experiments on various real-world graph datasets. The experimental results show that the proposed algorithm can significantly outperform the baseline algorithms like spectral clustering. The results also indicate that the stacked deep structure can boost the clustering results, i.e., the results become more and more accurate when going from the shallow layers to the deep layers.

To the best of our knowledge, this is the first work that investigates how to use deep learning for graph clustering. It enriches our understanding on the power of deep learning, by opening a door to use unsupervised pre-training techniques like stacked sparse autoencoder to deal with the clustering problems.

[2]Note that spectral clustering can obtain the global optimum; however, autoencoder usually leads to a local optimum due to the back-propagation algorithm it employs.

2 Background and Related Work

Given an undirected weighted graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the node set and $E = \{e_{ij}\}$ is the edge set, graph clustering aims to find a disjoint partition $\{V_i\}_{i=1}^k$ of V, where k is cluster number. As has been mentioned before, our proposed model is highly related to spectral clustering and deep learning, so we will briefly review these two methods.

2.1 Spectral Clustering

We use $S = \{s_{ij}\}$ to denote the similarity matrix of graph G, and thus s_{ij} $(i, j = 1, 2, \cdots, n)$ is the similarity score between node i and j. Let $d_i = \sum_j s_{ij}$ be the degree of node i, based on which we measure the capacity of a subset A of V, i.e., $vol(A) = \sum_{i \in A} d_i$. For any disjoint subsets $A, B \subset V$, we define $link(A, B) = \frac{1}{2} \sum_{i \in A, j \in B} s_{ij}$. One of the commonly-used clustering objective that spectral clustering aims to minimize is the Normalized Cut (NCut):

$$NCut(V_1, ..., V_k) = \sum_{i=1}^{k} \frac{link(V_i, \bar{V}_i)}{vol(V_i)}. \tag{1}$$

Here $V_i \cap \bar{V}_i = \emptyset$ and $V_i \cup \bar{V}_i = V$. To achieve this goal, spectral clustering converts the above objective function to the following discrete optimization problem:

$$\begin{aligned} & \min_{H \in \mathbb{R}^{n \times k}} Trace(H^T L H) \\ s.t. \quad & H^T D H = I. \\ & H_{ij} = \begin{cases} 1/\sqrt{vol(V_j)}, & if \quad v_i \in V_j \\ 0, & otherwise \end{cases}, \\ & (i = 1, 2, \cdots, n; j = 1, 2, \cdots, k); \\ & D = diag(d_1, d_2, \cdots, d_n); \\ & L = D - S. \end{aligned} \tag{2}$$

Here L is the so-called graph Laplacian matrix. It can be seen that the discrete optimization problem in (2) is NP-Hard (Wagner and Wagner 1993). Therefore, spectral clustering turns to relax the condition in (2) by allowing H_{ij} to take any real values. According to some matrix calculus, the solution yields H to consist the eigenvectors corresponding to the k smallest non-zero eigenvalues of the normalized Laplacian matrix $D^{-1}L$. The final clustering results are then obtained through running the k-means algorithm on the graph embedding matrix H.

There are works towards efficient implementation of spectral clustering such as (Chen et al. 2011), in which the authors' aim is to construct a sparse similarity graph as the input to the parallelized spectral clustering method, whereas what we aim to achieve is to replace the expensive EVD in spectral clustering, leading to the difference with former works.

2.2 Deep Learning

Recently deep learning has won great success in many applications such as image classification, speech recognition, and natural language processing (Bengio 2009) (Dahl et al. 2012) (Collobert et al. 2011) (Krizhevsky, Sutskever, and Hinton 2012). One of the strategies that make training deep architectures possible and effective is the greedy layerwised unsupervised pretraining (Hinton and Salakhutdinov 2006) (Bengio et al. 2007). This strategy aims to learn useful representations one layer at a time, and then to set the output features to be the input of the next layer. Each layer in this process involves some kind of non-linearity, such as non-linear activation function (e.g., *sigmoid* or *tanh*) or some regularization on features (e.g., sparsity constraints (Poultney et al. 2006)). By stacking these non-linear single layers together, deep learning are believed to yield better representations (Part 4 in (Bengio, Courville, and Vincent 2013)).

In the greedy layerwised pretraining process, *autoencoder* (Bourlard and Kamp 1988) (Hinton and Zemel 1994) is commonly used as a basic unit to generate new representations in each layer, and it is also the main building block in our model. In the autoencoder framework, a feature extraction function $f(\cdot;\theta_1)$ is firstly implemented on original feature vector $x_i,\ i = 1,2,\cdots,n$ (n is the number of training samples), yielding a new representation $f(x_i;\theta_1)$. Function $f(\cdot;\theta_1)$ is named as *encoder*. After that, $f(x_i;\theta_1)$ is transformed back into the input space by another function $g(\cdot;\theta_2)$, which is called as *decoder*. The aim of autoencoder is to minimize the reconstruction loss between the original data and the reconstructed data from the new representations, i.e.,

$$Loss(\theta_1,\theta_2) = \sum_{i=1}^{N} l\left(x_i, g\left(f(x_i;\theta_1);\theta_2\right)\right). \tag{3}$$

Here $l(\cdot)$ is the sample-wise loss function. Usually encoder and decoder are composed of a linear transformation, followed by an activation function. That is, $f(x;\theta_1) = f_0(Wx+b)$, $g(x;\theta_2) = g_0(Mx+d)$, where f_0 and g_0 are activation functions like the element-wise sigmoid function. In this sense, $\theta_1 = \{W,b\}$ and $\theta_2 = \{M,d\}$ are the parameters to be learned in the training process. Furthermore, there are works on sparse autoencoder, which aims to penalize the large hidden layer outputs (Olshausen and Field 1997)(Boureau, Cun, and others 2007).

3 Model Description

In this section, we introduce our proposed deep learning method for graph clustering, including the motivation, the basic building block, and the implementation details.

3.1 Motivation

As mentioned in the introduction, our proposal is motivated by the similarity between autoencoder and spectral clustering. To show this, we will first explain spectral clustering in the viewpoint of matrix reconstruction, and then demonstrate that autoencoder is a better choice than spectral clustering in the scenario of large-scale graph clustering.

We start with some notations. Let $Q = D^{-1}L$, where L is the graph Laplacian matrix and D is the diagonal matrix with the node degrees in the corresponding diagonal elements. According to the properties of the Laplacian matrix, Q is symmetric and its rank is assumed to be $n-r$, where r is the number of connected components in the graph G. We write the eigenvalue decomposition of Q as $Q = Y\Lambda Y^T$, where $Y \in R^{n\times n}$ stacks the N eigenvectors of Q in columns and $Y^TY = I$, $\Lambda = diag(\lambda_1,\lambda_2,\cdots,\lambda_n)$, $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ are the n eigenvalues of Q. Let Λ_k denote the diagonal matrix with the k smallest non-zero eigenvalues in Λ in its diagonal elements, and $Y_k \in R^{n\times k}$ be the matrix containing the k columns from Y corresponding to the k non-zero eigenvalues in Λ_k. Thus Y_k is the embedding matrix used in spectral clustering when minimizing the normalized cut in order to cluster the graph nodes into k groups.

Note that the non-zero requirement on the eigenvalues does not impact the clustering result by much when k is relatively large. The reason is as follows. The zero eigenvalue of Q corresponds to the eigenvector with all its elements equal to 1. If we put this eigenvector into Y_k, it will have no effect on the clustering result except that it will squeeze out the least informational eigenvector. For simplicity, in the following discussions we will directly talk about the k smallest eigenvalues in Λ without requiring the eigenvalues to be non-zero. Further note that $Q = D^{-1}L = D^{-1}(D-S) = I - D^{-1}S$. Therefore, the k smallest eigenvalues of Q are exactly the k largest eigenvalues of the normalized graph similarity matrix $D^{-1}S$, and accordingly Y_k contains the k eigenvectors of $D^{-1}S$ corresponding to its k largest eigenvalues. The *Eckart-Young-Mirsky Theorem* (Eckart and Young 1936) explains the reconstruction nature of spectral clustering, which is related to the low-rank approximation of a matrix.

Theorem 1 *(Eckart-Young-Mirsky). For a rank-r matrix $P \in R^{m\times n}$, with singular value decomposition (SVD) $P = U\Sigma V^T$, $U^TU = I$, $V^TV = I$; if $k < r$, we have:*

$$\arg \min_{\substack{\tilde{P}\in R^{m\times n} \\ rank(\tilde{P})=k}} ||P-\tilde{P}||_F = U\tilde{\Sigma}V^T \tag{4}$$

where $\tilde{\Sigma}$ is the same matrix as Σ except that it contains only the k largest singular values and the other singular values are replaced with 0.

The above theorem shows that the matrix reconstruction by the truncated largest k singular vectors in SVD is the best rank-k approximation of the original matrix. Furthermore, if a matrix Z is symmetric, i.e., $Z = PP^T$, there exists orthogonal decomposition $Z = U\Sigma^2U^T$=$U\Lambda U^T$, where $U^TU = I$ and Λ is diagonal matrix with the eigenvalues of Z in the diagonal elements. This is the well-known fact that the SVD of a matrix P is highly related to the EVD of the symmetric matrix PP^T. Specifically, for symmetric matrix Z, the matrix reconstruction by the truncated largest k eigenvectors in EVD is also the best rank-k approximation under the Frobenuis norm. According to the above discussions, we obtain the following corollary.

Corollary 2 *$Y_k\Lambda_kY_k^T$ is the best reconstruction of the symmetric matrix $D^{-1}S$ in terms of the Frobenuis norm among all rank-k matrices, where Y_k is the embedding matrix in spectral clustering.*

On one hand, Corollary 2 tells us that spectral clustering can be regarded as a process of matrix reconstruction for $D^{-1}S$. From the new embedding matrix Y_k obtained through eigenvalue decomposition, it can build the best rank-k matrix approximation towards the normalized graph similarity matrix $D^{-1}S$ in terms of Frobenuis norm. On the other hand, imagine that if we use the normalized graph similarity matrix $D^{-1}S$ as the input feature matrix for an autoencoder, we are actually also seeking for the best matrix reconstruction for $D^{-1}S$ in terms of Frobenuis norm by solving the autoencoder. That is, both autoencoder and spectral clustering minimize the reconstruction error for the original normalized similarity matrix $D^{-1}S$. We therefore say that autoencoder and spectral clustering are similar in terms of what they optimize.

While autoencoder is similar to spectral clustering in theory, the former is much more flexible in incorporating additional constraints. In many real large-scale graph clustering problems, it is desirable to seek some sparse representations as the graph embedding. On one aspect, this can greatly improve the efficiency of the system in terms of both storage and data processing. On another aspect, it usually also improves the clustering accuracy, since it can remove some noisy information that hurts the clustering results. However, the embedding produced by spectral clustering cannot guarantee the sparsity property since the eigenvectors of Q are probably very dense. Furthermore, it is non-straightforward to introduce sparsity constraint to spectral clustering; if we do it in a stiff way (e.g., directly add a sparsity constraint to the objective function of spectral clustering), it is very likely that we cannot get a spectral method any more. In sharp contrast, we will have a much easier life if we use autoencoder. That is, we can simply adopt the sparse autoencoder developed in the literature of deep learning. It basically introduces a sparsity regularization term to the original objective function of autoencoder and can still benefit from the efficient back-propagation algorithm for the optimization. Furthermore, one can stack multiple layers of sparse autoencoders, to achieve additional benefit from deep structures. The above discussions motivate us to adopt an autoencoder-based method for graph clustering.

3.2 GraphEncoder

In this subsection we introduce the autoencoder-based graph clustering model called *GraphEncoder*. In general, the key component of GraphEncoder is a deep neural network with sparse autoencoder as its building block. As stated in the previous sections, given an n-node graph G with its similarity matrix S, we can treat S as the training set containing n instances $s_1, s_2, \cdots, s_n$, $s_i \in \mathbb{R}^n$, $i = 1, 2, \cdots, n$. Note that $s_i = \{s_{ij}\}$, $j = 1, 2, \cdots, n$. Then we feed the normalized training set $D^{-1}S$ into the deep neural network and use the output features in the deepest layer of DNN as the graph embedding. At last, k-means clustering is implemented on the graph embedding of G to produce the final clustering result.

Specifically, we consider the autoencoder in each layer of the deep neural network. Let x_i be the i-th input vector of this layer, and f_0 and g_0 be the activations of the hidden layer and the output layer respectively. We have $h_i = f_0(Wx_i + b)$ and $y_i = g_0(Mh_i + d)$, where $\Theta = \{\theta_1, \theta_2\} = \{W, b, M, d\}$ are the parameters to be learned, f_0 and g_0 are the non-linear operators such as the sigmoid function ($sigmoid(z) = 1/(1+exp(-z))$) or tanh function ($tanh(z) = (e^z - e^{-z})/(e^z + e^{-z})$). Then the optimization goal is to minimize the reconstruction error between the original data x_i and the reconstructed data y_i from the new representation h_i,

$$arg \min_{\theta \in \Theta} \sum_{i=1}^{n} ||y_i - x_i||_2. \tag{5}$$

We also impose the sparsity constraints to the activation in the hidden layer. That is, we add a regularization term to the reconstruction error in (5),

$$Loss(\theta) = \sum_{i=1}^{n} ||y_i - x_i||_2 + \beta KL(\rho||\hat{\rho}), \tag{6}$$

where β controls the weight of the sparsity penalty, ρ is set to be a small constant such as 0.01, $\hat{\rho} = \frac{1}{n}\sum_{j=1}^{n} h_j$ is the average of the hidden layer activations, and $KL(\rho||\hat{\rho})$ is defined as:

$$KL(\rho||\hat{\rho}) = \sum_{j=1}^{|\hat{\rho}|} \rho log \frac{\rho}{\hat{\rho}_j} + (1-\rho) log \frac{1-\rho}{1-\hat{\rho}_j}. \tag{7}$$

Note that (6) can be solved by standard back-propagation algorithms. After the training of the current layer is completed, we use the hidden layer activations as the inputs to train the next layer. This greedy layer-wise training process forms the model of the Stacked Sparse Autoencoder (Vincent et al. 2010). When all the layers are trained in this manner, we use the output of the final layer as the new graph representation and run k-means on it to get the clustering results. The whole procedure is summarized as the algorithm in Table 1.

Table 1: Clustering with GraphEncoder

Input

n-node graph G, with similarity matrix $S \in \mathbb{R}^{n \times n}$ and degree matrix $D = diag\{d_1, d_2, \cdots, d_n\}$, where d_i is the degree of node i; DNN layers number Γ, with number of nodes $n^{(j)}$ in layer j; $n^{(1)} = n$; $X^{(j)} \in \mathbb{R}^{n \times n^{(j)}}$ is the input to layer j. $X^{(1)} = D^{-1}S$.

For $j = 1$ to Γ

1. Build a three layer sparse autoencoder with input data $X^{(j)}$.
2. Train the sparse autoencoder by optimizing (6) with back-propagation. Obtain the hidden layer activations $h^{(j)}$.
3. Set $X^{(j+1)} = h^{(j)}$.

End

Run k-means on $X^{\Gamma} \in \mathbb{R}^{n \times n^{(\Gamma)}}$.

Output

Final clustering result.

The proposed model GraphEncoder has the following advantages:

- Usually in autoencoder, the dimension of the hidden layer is lower than that of the input layer. This captures the intuition that not all edges in the original graph are necessary in clustering. For a certain node, it is possible that only its relationships with part of the other nodes (e.g., the nodes with the highest degrees) determine which cluster it should belong to. The optimization of (5) captures this useful low dimensional information.

- The sparsity penalty in (6) not only strengthens the requirements of deleting edges as stated above, but also makes the computation more efficient given the sparsity target ρ is a small value and the activations approach zero.
- The stacked structures provide a *smooth* way of eliminating edges. Graph representations are expected to become clearer and clearer for clustering as the training goes from shallow layers to deep layers. We will use an example in the experimental session to demonstrate this effect of the deep structure.
- GraphEncoder is much more efficient than spectral clustering. Spectral clustering relies on EVD. To our knowledge, the computational complexity of the fastest EVD solver is $O(n^{2.367})$ where n is the number of nodes in the graph, when the graph has some special sparse and dense structure like Toeplitz matrix (Pan and Chen 1999). In contrast, it is not difficult to see that the complexity of GraphEncoder is $O(ncd)$, where d is the maximum number of hidden layer nodes in DNN, and c is the average degree of the graph. Usually c can be regarded as a fixed value: for example, in the top-k similarity graph, $c = k$; and in a social network graph, c, the average friend number of people, is also bounded. Parameter d is related to the predefined number of clusters (more clusters lead to larger d), but not related to n. Therefore we can regard cd as some constant that does not increase with n, and the overall complexity of GraphEncoder is linear to the number of nodes. In addition, whereas EVD is hard to parallel, the stochastic gradient descent (SGD) (Bottou 1998) training of DNN is comparatively easy to parallel, as have been well explored in the literature of DNN.

4 Experimental Evaluation

We report the experimental results in this section. We first introduce the datasets and the benchmark algorithms used in the experiments, and then give the experiment settings. After that, we show the clustering performance of our proposed clustering algorithm compared with benchmark algorithms. In addition, we give an example to show the power of deep structures in clustering.

4.1 Datasets

To test the performance of our DNN-based model, we evaluated its performance on several real world datasets.

1. **Wine**. This is a dataset from UCI Machine Learning Repository (Asuncion and Newman 2007), consisting of 178 instances with 13 attributes. Every instance corresponds to a certain wine with its chemical analysis information as the attributes. All instances are labeled with 3 wine categories. We built a cosine similarity graph using these instances and used the labels as the groundtruth.
2. **20-Newsgroup**. This dataset is a collection of about 20,000 newsgroup documents. The documents are partitioned into 20 different groups according to their topics. We represented every document as a vector of *tf-idf* scores of each word and built the cosine similarity graph based on the *tf-idf* scores. To demonstrate the robustness of our algorithms with different targeting cluster numbers, we constructed three graphs built from 3, 6, and 9 different newsgroups respectively. The newsgroup names in each graph are listed as the following, where the abbreviation *NG* used in graph names is short for *Newsgroup*.
   - 3-NG: *corp.graphics*, *rec.sport.baseball*, *talk.politics.guns*.
   - 6-NG: *alt.atheism*, *comp.sys.mac.hardware*, *rec.motorcycles*, *rec.sport.hockey*, *soc.religion.christian*, *talk.religion.misc*.
   - 9-NG: *talk.politics.mideast*, *talk.politics.misc*, *comp.os.ms-windows.misc*, *comp.sys.ibm.pc.hardware*, *sci.electronics*, *sci.crypt*, *sci.med*, *sci.space*, *misc.forsale*

   For each chosen group, we randomly selected 200 documents from it, and thus the three graphs contain 600, 1,200, and 1,800 nodes respectively. The document labels are used as the groundtruth.
3. **DIP**. This is an unweighted protein-protein interaction (PPI) network from the Database of Interacting Proteins (Salwinski et al. 2004). The average degree of nodes is about 6.
4. **BioGrid**. The last dataset is another PPI network obtained from the BioGrid Database (Stark et al. 2011). We removed the nodes without any connections to other nodes from the original graph and got a final graph of 5,964 nodes. The average degree is approximately 65, which is much higher than that of DIP. We used the protein complex data in CYC2008 (Pu et al. 2009) as the groundtruth for the two PPI networks.

The detailed information of all these graphs are summarized in Table 2. Since all these datasets have groundtruth, we evaluated the performance of a clustering algorithm by the Normalized Mutual Information (NMI) of its clustering results. The range of NMI values is $[0,1]$. The higher the NMI is, the better the corresponding clustering results are.

Table 2: Datasets Information.

Dataset	#Nodes	Weighted or Not	AvgDegree
Wine	178	Weighted	Fully Connected
3-NG	600	Weighted	Fully Connected
6-NG	1,200	Weighted	Fully Connected
9-NG	1,800	Weighted	Fully Connected
DIP	4,741	Unweighted	6.4
BioGrid	5,964	Unweighted	64.7

4.2 Benchmark Algorithms

We used the following graph clustering methods as the benchmark algorithms.

1. **Spectral Clustering**. We used two versions of spectral clustering: *unnormalized spectral clustering* (Von Luxburg 2007), which aims to minimize ratio cut, and *normalized spectral clustering* (Shi and Malik 2000), which aims to minimize normalized cut. The better NMI value output by the two versions was recorded as the benchmark results.
2. **k-means**. Our algorithm runs k-means on the new graph embedding in the deep layers. To show the power of the deep structures, we also run the k-means algorithm directly on the original graph (i.e., k-means on the $n \times n$ normalized similarity matrix) as a benchmark method.

4.3 Experiment Settings

We implemented GraphEncoder based on Sparse Autoencoder (**SAE**), which is the autoencoder model that penalizes both the reconstruction error and the sparsity error in the hidden layer. In our experiments, we tuned two parameters of SAE: *sparsity penalty*, which tradeoffs the weight of the two errors in the optimization, and *sparsity target*, which is the target value that the hidden layer activations aim to reach.

The neural networks for *Wine* and *3-NG* have 3 layers and the neural networks for all the other graphs have 5 layers. The number of nodes in each layer are listed in Table 3. All the neural networks use element-wise sigmoid function as activations of each layer.

Table 3: Neural Network Structures

Dataset	#nodes in each layer
Wine	178-128-64
3-NG	600-512-256
6-NG	1,200-1,024-512-256-128
9-NG	1,800-1,024-512-256-128
DIP	4,741-2,048-1,024-512-256
BioGrid	5,964-2,048-1,024-256-128

For *Wine* and the three *20-Newsgroup* graphs, since their underlying cluster numbers by the groundtruth labels are small, we do not vary the target cluster numbers in the experiments. We just set the cluster numbers to their real cluster numbers, i.e. *Wine* with 3, the three *20-Newsgroup* graphs with 3, 6, and 9 respectively.

For the two PPI networks, we checked the algorithm performance with various predefined cluster numbers. That is, we varied the target cluster number from 5 to 400 (with average interval between two consecutive clustering numbers to be 50), and plotted the NMI values varying with the target cluster number for each algorithm.

4.4 Experimental Results

Clustering Performance. The clustering results on *Wine* and *20-Newsgroup* datasets are summarized in Table 4. It can be observed that the best performances of each dataset, listed in bolded digits, are all obtained by GraphEncoder built on sparse autoencoder.

Table 4: Clustering Results on Wine and 20-Newsgroup.

(Measured by NMI)	Wine	3-NG	6-NG	9-NG
Spectral Clustering	0.71	0.60	0.55	0.39
k-means	0.70	0.72	0.26	0.22
SAE	**0.84**	**0.81**	**0.60**	**0.41**

The experimental results on the two PPI networks are shown in Figure 1 and Figure 2 respectively, with horizontal axis to be predefined clusters number, and vertical axis to be the corresponding *NMI* value.

We can see that for all the 6 graphs: (i) GraphEncoder based on sparse autoencoder beats spectral clustering. This is exactly the empirical justification of our claim that sparsity on graph embedding can help to improve clustering results. (ii) GraphEncoder based on sparse autoencoder beats *k*-means directly running on original normalized similarity graph, showing that deep structures can help to get even better graph representations.

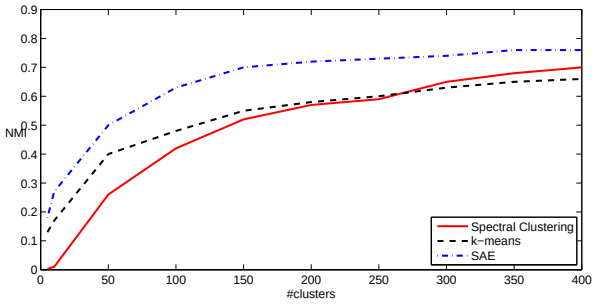

Figure 1: Clustering Results on DIP.

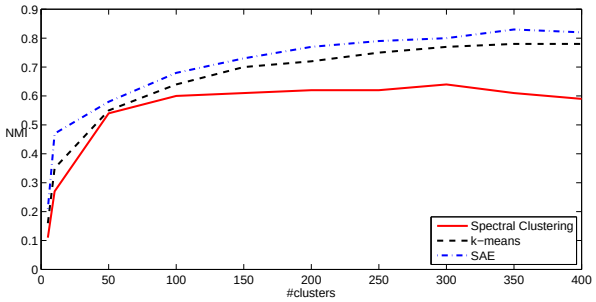

Figure 2: Clustering Results on BioGrid.

Power of Deep Structures. To show the power of the stacked deep structures, we list the NMI values of *k*-means clustering on each layer's embedding by GraphEncoder in Table 5, in which we aim to cluster *DIP* and *BioGrid* datasets into 50 groups. Note that DNN for both datasets has five layers, with the shallowest layer to be layer 1 and deepest layer to be layer 5. The NMI value on layer 1 is exactly the result of directly running *k*-means on the input normalized similarity graph. From Table 5 we can observe that the NMI values become larger when the layer goes from shallower to deeper, showing that the deep structures play an important role in generating good graph representations for clustering.

Table 5: Layer-wise NMI Values on Two PPI Networks with #cluster= 50.

Layer	1	2	3	4	5
DIP	0.40	0.42	0.45	0.50	0.51
BioGrid	0.55	0.55	0.56	0.58	0.59

5 Conclusion

In this paper, we have proposed a novel graph clustering method based on deep neural network, which takes the sparse autoencoder as its building block. We introduced a layer-wise pretraining scheme to map the input graph similarity matrix to the output graph embedding. Experimental results on several real world datasets have shown that the proposed method outperforms several state-of-the-art baselines including spectral clustering.

References

Asuncion, A., and Newman, D. 2007. Uci machine learning repository.

Bengio, Y.; Lamblin, P.; Popovici, D.; and Larochelle, H. 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing systems* 19:153.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8):1798–1828.

Bengio, Y. 2009. Learning deep architectures for ai. *Foundations and trends in Machine Learning* 2(1):1–127.

Bottou, L. 1998. Online learning and stochastic approximations. *Online learning in neural networks* 17:9.

Boureau, Y.-l.; Cun, Y. L.; et al. 2007. Sparse feature learning for deep belief networks. In *Advances in neural information processing systems*, 1185–1192.

Bourlard, H., and Kamp, Y. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics* 59(4-5):291–294.

Chan, P. K.; Schlag, M. D.; and Zien, J. Y. 1994. Spectral k-way ratio-cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 13(9):1088–1096.

Chen, W.-Y.; Song, Y.; Bai, H.; Lin, C.-J.; and Chang, E. Y. 2011. Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33(3):568–586.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12:2493–2537.

Dahl, G. E.; Yu, D.; Deng, L.; and Acero, A. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on* 20(1):30–42.

Dhillon, I. S.; Guan, Y.; and Kulis, B. 2007. Weighted graph cuts without eigenvectors a multilevel approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 29(11):1944–1957.

Eckart, C., and Young, G. 1936. The approximation of one matrix by another of lower rank. *Psychometrika* 1(3):211–218.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Hinton, G. E., and Zemel, R. S. 1994. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems* 3–3.

Karypis, G., and Kumar, V. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20(1):359–392.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, 1106–1114.

Olshausen, B. A., and Field, D. J. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research* 37(23):3311–3325.

Pan, V. Y., and Chen, Z. Q. 1999. The complexity of the matrix eigenproblem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 507–516. ACM.

Poultney, C.; Chopra, S.; Cun, Y. L.; et al. 2006. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, 1137–1144.

Pu, S.; Wong, J.; Turner, B.; Cho, E.; and Wodak, S. J. 2009. Up-to-date catalogues of yeast protein complexes. *Nucleic acids research* 37(3):825–831.

Salwinski, L.; Miller, C. S.; Smith, A. J.; Pettit, F. K.; Bowie, J. U.; and Eisenberg, D. 2004. The database of interacting proteins: 2004 update. *Nucleic acids research* 32(suppl 1):D449–D451.

Satuluri, V., and Parthasarathy, S. 2009. Scalable graph clustering using stochastic flows: applications to community discovery. In *Proceedings of the 15th ACM SIGKDD conference*, 737–746. ACM.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22(8):888–905.

Smyth, S., and White, S. 2005. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 5th SIAM International Conference on Data Mining*, 76–84.

Stark, C.; Breitkreutz, B.-J.; Chatr-Aryamontri, A.; Boucher, L.; Oughtred, R.; Livstone, M. S.; Nixon, J.; Van Auken, K.; Wang, X.; Shi, X.; et al. 2011. The biogrid interaction database: 2011 update. *Nucleic acids research* 39(suppl 1):D698–D704.

Van Dongen, S. M. 2000. Graph clustering by flow simulation.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research* 9999:3371–3408.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and computing* 17(4):395–416.

Wagner, D., and Wagner, F. 1993. Between min cut and graph bisection. In *Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, MFCS '93, 744–750. London, UK, UK: Springer-Verlag.