A report on the Praktikum 2

Deep Learning Clustering approach

Winter Semester 2017-18

Saman Paidar Nia

Nr. **01358135**

spaidarn@mtu.edu

https://github.com/saman-nia/Autoencoder_Clustering

Agenda Items:

1. Abstract

This report contains details about two kind of clustering algorithms. It mainly compares two of the clustering algorithms, spectral clustering and Autoencoder clustering and difference and similarity between them. To show accuracy of algorithms performance, I measure the result with *Normalized Mutual Information (NMI)*. It is noteworthy that, I spend more than 20 hours per week on the project.

2. Introduction

With the employing of the Deep Learning and a large amount of computing on the data, systems now can translate speech and recognize objects in real time. In this project, I scheme an Autoencoder Embedding Network for representational learning, which is more profitable and efficient for cluster analysis. K-Means and Spectral Clustering are well-known methods which focus on modeling the similarity relationship among samples but, those ignore to extract more effective representation for cluster analysis. Initially, I exploit a Deep Autoencoder to construct a high-dimensional input into a low-dimensional embedding and then uses that low-dimensional embedding to clustering codes of different classes. To make the learned representations suitable for clustering, I impose Feed-forward Autoencoder which aims to embed similarity data into manifold space. Then, comparing with Normalized and Unnormalized Spectral Clustering which extracts representations from the similarity matrix. We apply a group loss cost constraint for the learned representations, and aim to learn representations in which the non-zero groups correspond to its cluster. The K-Means clustering measures on various graph datasets produced by Autoencoder Embedding and Spectral Embedding show that the Autoencoder method significantly better

perform than Spectral Clustering. The complexity of computing in autoencoder is lower that spectral clustering.

3. Graph Theory

G (V, E) is a graph which *V* is number of vertices (node) and *E* is number of edges. The *E* is a subset of $V \times V$ ($E \subset V \times V$). The graph edges could have weights, which expression a feature between the nodes.

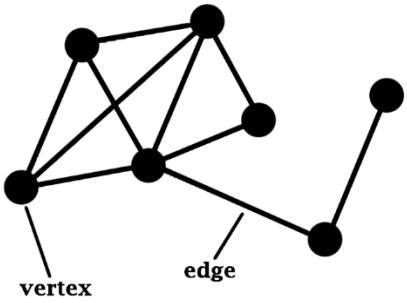

Figure 1 *A Graph.*

A graph can be directed or undirected:

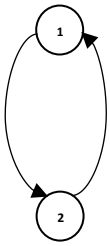

Figure 1 *Undirected Graph*

	1	2
1	0	1
2	1	0

Table 1 *Affinity Matrix for the Undirected Graph*

Figure 2 *Directed Graph*

	1	2
1	0	1
2	0	0

Table 2 *Affinity Matrix for the Directed Graph*

3.1. Graph Similarity

The technique is based on identify a mapping that is both one-to-one between nodes which are hold affinity. Here we have two directed graph which could be consider as similarity computing:

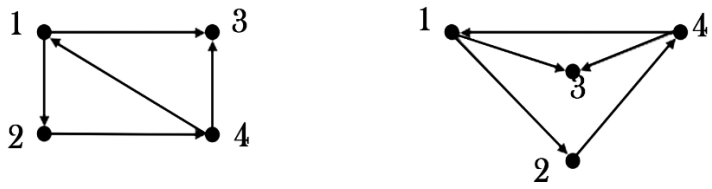

Figure 3 *Affinity between two Directed Graphs.*

3.2. Types of used Similarities

I have used two type of similarities such as: Normalized Cosine Similarity and Normalized Nearest Neighbors Graph Similarity. I have used Normalized Nearest Neighbors Graph Similarity only for the Spectral Clustering to measure that whether this a good idea to use Normalized Cosine Similarity for comparing both algorithms or not.

3.2.1. Normalized Cosine Similarity: When we make matrix standard, we could have normally distribution on our features. After that I compute cosine similarity for my data. Now we need to get diagonal elements of matrix (D^{-1}). Each column should hold a minimum value 0 and maximum 1. Diagonal creates a matrix with the elements of zero.

3.2.2. Normalized Nearest Neighbors Similarity: I compute Normalized Nearest Neighbors Similarity for my data and the same steps from the previous similarity.

4. Methods

I have employed two type of embedding such as Spectral Embedding and Autoencoder Embedding. Finally I ran the K-Means algorithm on them to get the clustering result.

4.1. K-Means Algorithm: The algorithm works in the way that corresponds to calculating the mean for each cluster. Sum component for all the points in the cluster separately, and divide it by the number of points in the cluster. The previous cluster center will not be a part of the calculation of the new cluster center. K-means minimizes the sum of Squared Euclidean and follow Euclidean distance. It minimizes the sum of squares which happens to agree with squared Euclidean distance.

4.2. Spectral Embedding: spectral clustering is including two major steps which are *Laplacian Eigenmaps* and *K-Means algorithm*. According to the *Laplacian Eigenmaps* method, we have to solve the generalized eigenvalue problem, which is $Ly=\lambda Dy$. According to the origin paper [3]:

$$x_i \rightarrow (f1(_i), \ldots, fm(_i)) xi \rightarrow (f1(_i), \ldots, f_m(_i))$$

For instance, the embedding of a point x_2 in, say, in two components is given by $(f_1\ (2),\ f_2\ (2))$ where f_1 and f_2 are the eigenvectors corresponding to the two smallest non-zero eigenvalues from the generalized eigenvalue problem $Lf=\lambda Df$. In other words, we can get the embedding of a point that was already considered when computing L. this method would be employee for non-linear dimensionality reduction.

4.3. Graph of Autoencoder: In this subsection I introduce the Autoencoder-based graph clustering model called Graph Encoder.

In general, the key component of Graph Encoder is a Deep Feed-Forward Neural Network with Autoencoder Architecture 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 $s_1, s_2, \cdots, s_n$, $s_i \in \mathbb{R}_n$, $i = 1,2,\ldots,n$. Note that $s_i = \{s_{ij}\}$, $j = 1, 2,\ldots, n$. 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.

5. Model Description

Here, I introduce my proposed autoencoder and spectral methods for graph clustering.

5.1. Motivation

I have seen the potential computational complexity and scalability issues of Spectral Clustering when applied to large Graphs. On the other we know that the approach proposed in, which is Autoencoders implemented with neural networks.

I propose to replace the eigenvector decomposition step in Spectral Clustering, as defined in, by an multi-layer Autoencoder pipeline implemented using only the K-Means clustering algorithm in a recursive way. K-Means is well known for its computational simplicity, ease of implementation, and ease of scalability in parallel distributed computing frameworks.

5.2. Principle

Here I will explain the purpose of Spectral Clustering algorithm:

1. Obtaining a Normalized Cosine Similarity.
2. Construct the Graph Laplacian from *A*; I decide on a normalization *L = D^{-1/2} A D {-1/2}.*
3. Solve an Eigenvalue problem.
4. Select *X* eigenvectors to define a k-dimensional subspace.
5. Form clusters in this subspace using, K-Means.

Here I will explain the purpose of Autoencoder Clustering Algorithm:

1. Obtaining a Normalized Cosine Similarity.
2. Design the autoencoder architecture with the three hidden layers with Adam Optimizer class from TensorFlow.
3. Set the parameters:
    a. 0.01 for learning rate which should be float value.
    b. Set a loop on *training_epochs* to train.
    c. Set a value on *n_backpropagation* to save the data production by the model into an array.
    d. Set 0.9 for beta1: The exponential decay rate for the 1st moment estimates[4].
4. Set dataset to model to get a low-dimension dataset from the Denoising Autoencoder by optimizing with backpropagation.
5. Apply the K-Means on the dataset produced.

To show the Autoencoder I have also ran the K-Means on the $N \times N$ dataset.

6. Experimental Evaluation

I took *Wine* and *Breast Cancer* and *Iris datasets* to make my model. Here I will use the datasets show the clustering performance of my proposed Autoencoder Clustering algorithm and will compare it with Spectral Clustering algorithms.

6.1. Dataset description

6.1.1. Breast Cancer Wisconsin: The Data Set is a Multivariate dataset with 569 instances and 32 attributes and 2 classes which has made it very suitable for Deep Learning.

6.1.2. Wine: The Data Set is a Multivariate dataset with 178 instances and 13 attributes and 3 classes.

6.1.3. Iris: The Data Set is a Multivariate dataset with 150 instances and 4 attributes and 3 classes. The lower attributed would not be nice for Deep Learning models. However I took it for a testing.

6.2. Setting: Check the Table 3 for setting of model.

Table 3: *Neural Network Layers*

Dataset	Nodes in each Hidden Layers	Nodes in Input Layer	Nodes in Output Layer
Breast cancer	256-128-64	569	2
Wine	128-64-32	178	16
Iris	128-64-32	150	16

6.3. Results: All the clustering result are presented in Table 4.

Table 4: *Clustering Results*

Clustering Algorithms	Breast Censer	Wine	Iris
Spectral Clustering on CS	0.43	0.75	0.83
Spectral Clustering on NN	0.32	0.82	0.27
K-Means on CS	0.42	0.56	0.86
AE on CS	**0.51**	**0.87**	**0.87**

- *CS is shorted for Cosine Similarity and NN for Nearest Neighbors.*

6.4. Training Cost

Here I would to show the performance of Autoencoder on Breast Cancer dataset. After *2000* time of backpropagation, I have received a cost training of approximately 0.6, check the Figure 4. While the curves look promising it's difficult to, upon first glance, understand whether we have reached a plateau at a good cost.

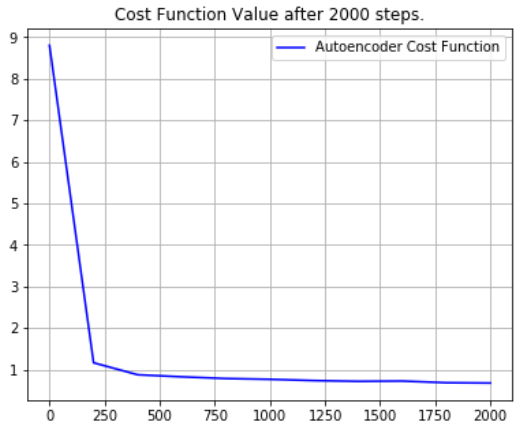

Figure 4 *Cost Training.*

6.5. NMI score in the each Backpropagation

Here I would to show the performance of NMI on embedding Autoencoder data. After 2000 time of backpropagation, I have received an accuracy of approximately 0.51, check the Figure 5.

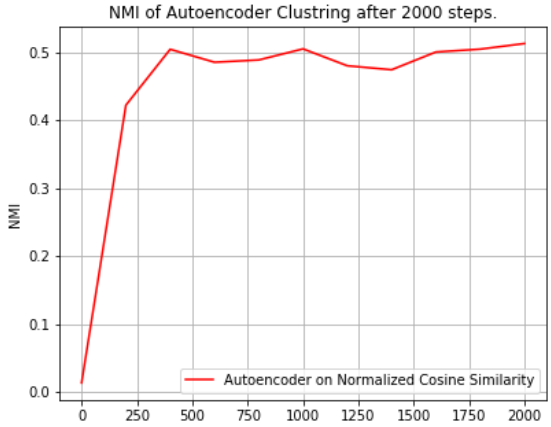

Figure 5 *NMI for each Backpropagation in the loop.*

6.6. Visualization of the Results

To show the power of Deep Learning, I visualized the embedding data produced by Spectral Embedding and Autoencoder Embedding. Check the figure 6.

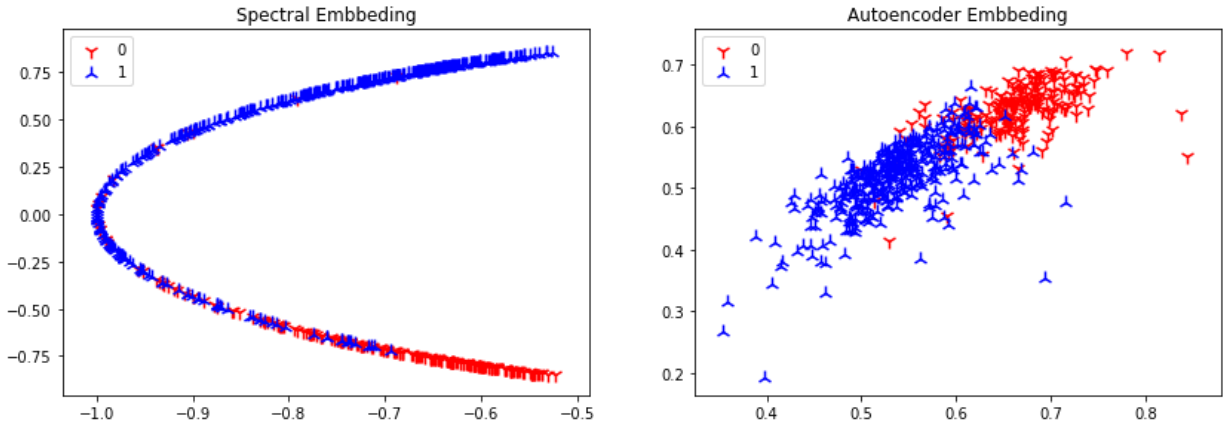

Figure 6 *Two dimension data produced by Spectral Embedding and Autoencoder with the original labels.*

Notice that the autoencoder would to have a better performance of K-Means clustering.

7. Conclusion

In this Praktikum, I have proposed a novel graph clustering method based on Deep Feed-forward Neural Network, which takes the Autoencoder as its building block. Experimental results on several real world datasets have shown that the proposed method outperforms several has better performance baselines including Spectral Clustering.

8. References

1. ***A Notion of Graph Homeomorphism*** *by Oliver Knill, Arxiv: 1401.2819v1 [Math.Gn] 13 Jan 2014.*
2. ***Graph Similarity*** *by Laura Zager and George Verghese at Eecs, Mit, March 2005.*
3. ***Laplacian Eigenmaps for Dimensionality Reduction and Data Representation*** *by Mikhaol Belkin and Partha Niyogi, December 8, 2002.*
4. ***Normalized Cuts and Image Segmentation****, 2000 Jianbo Shi, Jitendra Malik* *http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.160.2324*
5. ***A Tutorial on Spectral Clustering****, 2007 Ulrike von Luxburg* *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.165.9323*
6. ***Multiclass Spectral Clustering****, 2003 Stella X. Yu, Jianbo Shi* *http://www1.icsi.berkeley.edu/~stellayu/publication/doc/2003kwayICCV.pdf*
7. ***Sparse Autoencoder*** *by Andrew Ng, CS294A Lecture notes,* *https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf*
8. ***Learning Deep Representations for Graph Clustering*** *by Fei Tian, Bin Gao, Qing Cui, Enhong Chen, Tie-Yan Liu.*
9. ***Fundamental of Deep Learning*** *by Nikhil Buduma, OReilly.*
10. ***UCI Machine Learning Repository****, Asuncion, A., and Newman, D. 2007.*
11. ***Reducing the Dimensionality of Data with Neural Networks****, Hinton, G. E., and Salakhutdinov, R. R. 2006.*
12. ***Autoencoders, Minimum Description Length, and Helmholtz Free Energy,*** *Hinton, G. E., and Zemel, R. S. 1994.*
13. ***Textual Spatial Cosine Similarity*** *by Giancarlo Crocetti, Pace University Seidenberg School of CSIS, White Plains, NY 10606, USA*
14. ***Semantic Cosine Similarity,*** *Faisal Rahutomo*, Teruaki Kitasuka, and Masayoshi Aritsugi, Graduate School of Science and Technology, Kumamoto University.*
15. ***Similarity Learning for Nearest Neighbor Classification*** *by Ali Mustafa Qamar and Eric Gaussier.*
16. ***For TensorFlow library: https://www.tensorflow.org***
17. ***For Scikit-learn library: http://scikit-learn.org***
18. ***For Numpy library: http://www.numpy.org/***
19. ***For Matrix transformation: https://docs.scipy.org/***
20. ***For expression ambiguity: https://stackoverflow.com***