Data Mining: Mid-term Report

Ouyang Siqi & Jiang Jiaxuan YaoClass 70 2017011385 & 2017012515

November 20, 2020

1 Introduction

This project focuses on Graph-structured Data based on AutoML.

In the project, we propose automatic solutions that can effectively and efficiently learn high-quality representation for each node based on the given features, neighborhood and structural information underlying the graph. Based on the graph structure and the node features, we will finish a classification task on 5 different data sets.

2 Data

We have 5 different data sets, where each of them contains a graph, directed or undirected, dense or sparse, along with features of nodes. Different data sets have different characteristics as shown in Table 1.

Data Set Number	a	b	c	d	e
Node Number	2708	3327	10000	10000	7521
Edge Number	5278	4552	733316	5833962	7804
Average Degree	1.949	1.368	73.3316	583.3962	1.0376
Feature Number	1414	3657	589	293	0
Directed	Undirected	Undirected	Undirected	Directed	Undirected
Class Number	7	6	41	20	3

Table 1: Characteristics of diffrent data set.

2.1 a

Here we have 7 classes, 1414 features, 10556 edges, and 2708 nodes in **a**, where the feature space is really sparse, that is, only a few feature components are 1, most of the other components are 0. Besides, the edges of the graph is also sparse. So we need to embed carefully to reduce the dimension remaining most of the information.

2.2 b

Here we have 6 classes, 3657 features, 9104 edges, and 3327 nodes in **b**. The situation of **b** is nearly the same as that of **a** but **b** is larger and has more features.

2.3 c

Here we have 41 classes, 589 features, 1466632 edges, and 10000 nodes in **c**, where the feature space is not sparse any more. However, the edges of the graph is really dense. So we need to pay more attention to the structure of the graph. Besides, the number of classes is large which means it harder to classify the node.

2.4 d

Here we have 20 classes, 293 features, about 5833962 edges, and 10000 nodes in **d**, where the features are all 0, which means there is no information in features. Moreover, the graph is directed and dense so we need to somehow 'transfer' the location information of nodes to learn the information underlying the graph.

2.5 e

Here we have 3 classes, 0 feature, 15608 edges, and 7521 nodes in **e**. Similarly to **d**, we have no information of feature. All we only have is just sparse undirected edge information. So the task is more like doing clustering rather than classification.

3 Model Design

3.1 GCN

Our core model is GCN (Graph Convolutional Network) with following layer-wise propagation rule:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

where $\tilde{A} = A + I_N$ is the adjacency matrix of the graph $\mathcal{G}$, I_N is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function, such as the $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. $H^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the l^{th} layer; $H^{(0)} = X$. This propagation rule can be motivated via a first-order approximation of localized spectral filters on graphs.[1] Experiments on a number of network data sets suggest that the proposed GCN model is capable of encoding both graph structure and node features in a way useful for semi-supervised classification.[2]

3.2 Adjacent Matrix

For graphs with few features or even no features, all we need to mine is the information underlying the structure of model. Not only do we need to pass the information to the adjacent nodes, but also remain the cluster characteristics of nodes. Here we tried to use a compressed vector of adjacent matrix to extract the structure information. Nevertheless, the performance didn't improve. Since the element of adjacent matrix is sparse and small, we might loss the information of adjacent matrix after compression. So we directly use the origin adjacent matrix $\tilde{A}$ as a latent feature of graphs, that is, each row of $\tilde{A}$ denotes features of the corresponding node. Two nodes that are close in adjacent matrix space have a short cut in graph, so that the mapping feature space quantifies the cluster information in graph.

3.3 Hyper-parameter Search

In machine learning, a hyper-parameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are derived via training. A hyper-parameter is usually of continuous or integer type, leading to mixed-type optimization problems, which makes them difficult to learn or even untrainable. Here we do hyper-parameter search including learning rate, drop rate, number of layers, and weight decay, to find better hyper-parameters in order to better learn the classification task. One thing to note is that adding over two layers does not improve the performance.

4 Performance

The classification performance of our model is shown as Table 2.

Data Set Number	a	b	c	d	e
Test Accuracy	0.869	0.733	0.934	0.936	0.88
Drop Rate	0.1	0.1	0.1	0.1	0.1
Learning Rate	0.0063	0.0016	0.0063	0.0010	0.0001
Layer Number	2	2	2	2	2
Weight Decay	1e-4	1e-4	1e-4	1e-4	1e-4

Table 2: Performance of the Model

5 Division of Labor

5.1 Ouyang Siqi

- Run the code
- Process data
- Hyper-parameter search
- Feature compression

5.2 Jiang Jiaxuan

- Write the report
- Construct basic GCN model
- Adjacent matrix

6 Future Work

Note that the accuracy of Task **b** is low. Our performance is not so good on sparse graph with sparse features. Our next Step is to optimize the performance om sparse graph using better feature embedding and other graph learning model to mine the underlying structure of the graph.

References

[1] Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering.

[2] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks, 2017.