2019-1 Deep Learning Homework #4

Fabian Tan Hann Shen (ID 20185112)

May 15, 2019

1.

Graph convolutional network (GCN) passes the messages from one node to another and apply *NodeApplyModule*. In GCN, the weight in the layer, l is shared by the all edges. The GCN is built by stacking multiple convolutional layers and each layer deploys point-wise non-linearity. Cora dataset is used as the input data. Cora dataset is a scientific publication dataset, with 2708 papers belonging to 7 different machine learning sub-field. The model is then trained for 50 epochs and the loss is minimized by performing back-propagation. Table. 1 shows the losses over the epoch 10, 30, and 50.

The layer-wise propagation formula for multilayer GCN is as below:

$$H^{(l+1)} = \sigma(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}HW)$$

where H^l indicates the l^{th} layer, and σ is the activation function, whereas W is the weight matrix. D and A are the degree matrix and adjacent matrix respectively.

Table 1: Overall loss over epochs in GCN

Epochs	Loss
10	1.7856
30	1.4897
50	1.2774

2.

Relational Graph Convolutional network, known as RGCN, is able to solve the entity classification and link prediction using a common graph convolutional network extended with multi-edge, with separate downstream processing. R-GCN has edges that can represent different relations, as differs from the vanilla GCN. Different edge has different weights on them, except those edges with the

same relation will share the same weights, W_r

The formula for the hidden layer:

$$h_i^{l+1} = \sigma[W_0^{(l)} h_i^{(l)} + \sum_{r \in R} \sum_{r \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)}]$$

where N_i^r denotes the number of neighbours of node i and $c_{i,r}$ is a normalization constant.

In the experiment, the dataset used was AIFB dataset. In each layer, it computes outgoing message using node representation and weight matrix. Incoming messages are aggregated and produce the next node representation. In order to overcome overfitting issue, there are two different methods introduced - basis decomposition and block-diagonal decomposition. Basic decomposition can be considered as a form a weight-sharing between different relations, whereas block-diagonal decomposition as a sparsity constraint om the weight metrics for each relation. Table. 2 shows the training/validation accuracies and losses over the each epochs. It can be seen that the validation accuracies and losses are improved as training epochs are increased to avoid overfitting issue.

Table 2: Accuracy/Loss over epochs in R-GCN

Epochs	Train Accuracy	Train Loss	Validation Accurary	Validation loss
10	0.9732	0.8515	1.00	0.9008
30	1.00	0.7458	0.9643	0.7945
50	1.00	0.7439	0.9643	0.7952

3.

Line graph neural network can target on community detection problem. Community detection can be used as graph clustering by breaking down the graph's vertices into clusters. Instead of labelling them like previous GCN, similar vertices will be grouped as clusters. The GNN on the line graph using the non-backtracking operators can be interpreted as learning directed edge features from an undirected graph. In this experiment, the dataset used was CORA dataset. These datasets will be formulated as directed graph, with the vertices as the paper, and edges as the citation links. Here, we focus on the binary community detection, which means one pair of two classes are picked among these 7 classes in CORA dataset (7C2), 21 training pairs are made up the data from multiple hop and update them. The model is updated using the Adam optimizer with learning rate 0.004, and trained for 50 epochs, 2 feature maps and J = 2. In addition, the network's community prediction is visualized as shown in Fig. 1 together with the ground truth.

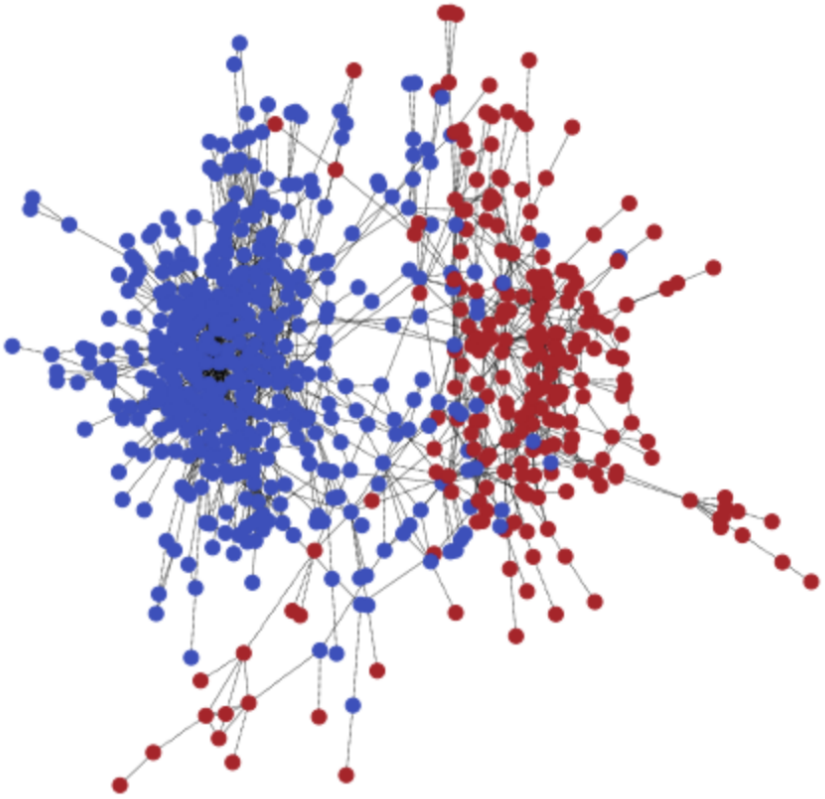

Figure 1: shows the network's community prediction graph

Table 3: Accuracy/Loss over epochs in L-GCN

Epochs	Loss	Accurary
10	0.4574	0.7933
30	0.4170	0.8226
50	0.3463	0.8568

4.

Many graph analytic problems can be characterized by a set of steady-state conditions. Flood-filled algorithm was used, which first marks the root node and then iteratively marks the nodes with one or more marked neighbours till

there is no node left.
In the experiment, there are 2 chains to be broken down, and each chain can have up to a length of 500 nodes. The model was configured in batch size of 64 and trained with 35 epochs.
The decision rule for binary classification:

$$\hat{y} = argmax_{i \in 0,1}[g_\phi h_v^T]_i$$

Table 4: Training/Testing accuracies over epochs in SSE

Epochs	Training Accuracy	Testing Accuracy
10	0.525	0.4892
35	1.00	0.9853

5.

How Graph Attention Network differs from vanilla GCN is owing to the aggregation of the information from the one-hop neighbourhood. It creates a weighting neighbor features with feature dependent and structure free normalization. GAT also introduced attention mechanism for the normalized convolution operation. The attention mechanism that is applied in a shared manner to all edges in the graph, making it does not depend on upfront access to the global graph structure or all of its nodes.

In this experiment, the dataset used was CORA dataset. The model is trained for 30 epochs and optimized using the Adam optimizer with the learning rate 1x10^{-3}. There are 4 main formulas to compute the node embedding for each layer.

$$z_i^{(l)} = W^{(l)} h_i^{(l)}, (1)$$

where Equation (1) is a linear transformation of the lower layer embedding and weight matrix

$$e_{ij}^{(l)} = \text{LeakyReLU}(\vec{a}^{(l)^T}(z_i^{(l)} || z_j^{(l)})), (2)$$

where Equation (2) computes the pair-wise unnormalized attention score between two neighbours.

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik}^{(l)})}, (3)$$

where Equation (3) applies the softmax function to normalize the attention scores.

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)}\right), (4)$$

where Equation (4) aggregates the embeddings and scaled by the attention scores.

Table 5: Losses over epochs in GAT

Epochs	Loss
10	1.9392
20	1.94304
30	1.92

The graph attentional layer utilized throughout these networks is computationally efficient, allowing assigning different importance to different nodes within a neighbourhood while interacting with different sized neighbourhoods.