Introduction to Deep Neural Network HW2 Report

2021313549 정성수

**#1 Code Implementation Details**

A screenshot of a computer

AI-generated content may be incorrect.

Import libraries, and set hyper-parameters as default (given)

CONTENT and CITES variables are the location of data.

A screenshot of a computer code

AI-generated content may be incorrect.

Here load cora dataset and make graph, adjacency matrix and normalize it with

Then return X(torch: feature matrix), adj\_norm(torch: ), edges(np: G.edges)

A computer screen shot of a code

AI-generated content may be incorrect.

Split edges to train(70%), validation(10%), and test(20%).

And sample k negative edges, where k is 2 times of positive edges. (hyper-parameter: Q\_NEG)

A computer screen shot of a computer code

AI-generated content may be incorrect.

Just for sampling dataset, separate pos and neg edges. With 1 and 0

A computer screen shot of a program code

AI-generated content may be incorrect.

Here GCN layers are defined. First, define graph convolution class that has no bias term, Kaiming(He) uniform initialize, and relu activation function. Xavier uniform initialize was also considered, but Kaiming had better result so Xavier was not selected. (Detailed comparisons in #2)

e = torch.cat() <= Here the final node embedding was **concatenated** and using MLP, go through additional **linear layers** for edge classification. Homework instruction clearly said that there is a linear layer for edge classification. So this kind of decoder layer was added.

But also the loss function described in homework instruction uses **dot product** of final node embedding, not concatenation of them. So I made two different version of code then test each of them. The result and selection is described in #2 in detail.

A screenshot of a computer program

AI-generated content may be incorrect.

Define function evaluating ROC AUC.

Define train function. First, make a list of hyper-parameters combinations to try. The first one is the default one.

Then load dataset.

A screenshot of a computer program

AI-generated content may be incorrect.

Second, get ready for model selection. For each config, prepare to train moving the data, link dataset, and sample edges.

A screenshot of a computer program

AI-generated content may be incorrect.

Third, start training – for each epoch, get model’s logit(prediction) and compute loss with binary corss entropy.

Usage of ‘binary\_cross\_entropy\_with\_logits’ in PyTorch will further explained in the next section (#2)

A screen shot of a computer code

AI-generated content may be incorrect.

Find the best model among the hyper-parameter configs using the validation set AUC score.

Load the best model and compute the final test AUC using test set.

A screenshot of a computer program

AI-generated content may be incorrect.

Build graph of the best model.

A screen shot of a computer code

AI-generated content may be incorrect.

Visualize the graph with ‘spring\_layout’, the layout selection is also later described in #2

**#2 Selecting Initialize, Loss function and Visualization method**

**1. Initialization:**

- Xavier initialize and Kaiming initialize are considered,

Kaiming:

A close-up of a calculator

AI-generated content may be incorrect.

Xavier:

A close-up of a calculator

AI-generated content may be incorrect.

So Kaiming initialize selected.

**2. Loss function:**

This loss function above is form of cross entropy loss, as far as I know. Because expectation of Q sampled negative edges are multiplied by Q again so the value is summation of all sampled negative log (of sigmoid) dot product for negative edges. The binary\_cross\_entropy\_with\_logit function compute the very same logic. However I must admit that I’m not 100% sure about this. But this was the only way to implement the loss function without any error inside the boundary of my knowledge. So cross-entropy in PyTorch is used.

Another available variation is to represent the node embedding with dot product or concatenation with a decoder layer.

First, I tried to implement the logit with the dot product as given. The result is:

A close-up of a calculator

AI-generated content may be incorrect.

Then concatenation with decoder layer. The result is:

A screenshot of a train loss

AI-generated content may be incorrect.

And this was such **a huge improvement** so this version is used for final implementation.

The two different version for model’s return format are both implemented in the final code. The concat version is activated and the other is deactivated (#).

**3. Visualization**

Another layouts experiments:

spring\_layout (seed=42):

A red and blue lines on a white background

AI-generated content may be incorrect.

shell\_layout:

A screenshot of a cell phone

AI-generated content may be incorrect.

Both were good but kamada\_kawai was the one that nodes are spreaded the most. (most readable, and analyzable)

**#3 Final Result**

Starting hyperparameter tuning for 7 configurations...

--- Training Configuration 1/7 ---

Hyperparameters: {'HIDDEN\_DIM': 16, 'DROPOUT': 0.1, 'LR': 0.01, 'WEIGHT\_DECAY': 4e-05, 'Q\_NEG': 2}

Config 1 - Epoch 01 | Train Loss 0.6401 | Val AUC 0.7097

Config 1 - Epoch 02 | Train Loss 0.5573 | Val AUC 0.7936

Config 1 - Epoch 03 | Train Loss 0.4803 | Val AUC 0.8298

Config 1 - Epoch 04 | Train Loss 0.4137 | Val AUC 0.8714

Config 1 - Epoch 05 | Train Loss 0.3606 | Val AUC 0.9115

...

Config 1 - Epoch 24 | Train Loss 0.1460 | Val AUC 0.9483

Config 1 - Epoch 25 | Train Loss 0.1507 | Val AUC 0.9503

Config 1 - Epoch 26 | Train Loss 0.1510 | Val AUC 0.9500

Config 1 - Epoch 27 | Train Loss 0.1451 | Val AUC 0.9493

Config 1 - Epoch 28 | Train Loss 0.1461 | Val AUC 0.9491

Config 1 - Early stopping at epoch 28.

Config 1 - Best Val AUC: 0.9513

Config 1 became new best model! Test AUC: 0.9495

--- Training Configuration 2/7 ---

Hyperparameters: {'HIDDEN\_DIM': 32, 'DROPOUT': 0.1, 'LR': 0.01, 'WEIGHT\_DECAY': 4e-05, 'Q\_NEG': 2}

Config 2 - Epoch 01 | Train Loss 0.6083 | Val AUC 0.8557

Config 2 - Epoch 02 | Train Loss 0.4243 | Val AUC 0.8894

Config 2 - Epoch 03 | Train Loss 0.3344 | Val AUC 0.9041

Config 2 - Epoch 04 | Train Loss 0.2824 | Val AUC 0.9213

Config 2 - Epoch 05 | Train Loss 0.2428 | Val AUC 0.9286

...

Config 2 - Epoch 23 | Train Loss 0.0750 | Val AUC 0.9625

Config 2 - Epoch 24 | Train Loss 0.0750 | Val AUC 0.9620

Config 2 - Epoch 25 | Train Loss 0.0729 | Val AUC 0.9639

Config 2 - Epoch 26 | Train Loss 0.0679 | Val AUC 0.9600

Config 2 - Epoch 27 | Train Loss 0.0626 | Val AUC 0.9645

Config 2 - Early stopping at epoch 27.

Config 2 - Best Val AUC: 0.9670

Config 2 became new best model! Test AUC: 0.9568

--- Training Configuration 3/7 ---

Hyperparameters: {'HIDDEN\_DIM': 48, 'DROPOUT': 0.1, 'LR': 0.01, 'WEIGHT\_DECAY': 4e-05, 'Q\_NEG': 2}

Config 3 - Epoch 01 | Train Loss 0.6160 | Val AUC 0.8377

Config 3 - Epoch 02 | Train Loss 0.4035 | Val AUC 0.9060

Config 3 - Epoch 03 | Train Loss 0.3096 | Val AUC 0.9301

Config 3 - Epoch 04 | Train Loss 0.2540 | Val AUC 0.9377

Config 3 - Epoch 05 | Train Loss 0.2156 | Val AUC 0.9376

...

Config 3 - Epoch 23 | Train Loss 0.0623 | Val AUC 0.9576

Config 3 - Epoch 24 | Train Loss 0.0663 | Val AUC 0.9593

Config 3 - Epoch 25 | Train Loss 0.0683 | Val AUC 0.9565

Config 3 - Epoch 26 | Train Loss 0.0596 | Val AUC 0.9581

Config 3 - Epoch 27 | Train Loss 0.0558 | Val AUC 0.9566

Config 3 - Early stopping at epoch 27.

Config 3 - Best Val AUC: 0.9596

--- Training Configuration 4/7 ---

Hyperparameters: {'HIDDEN\_DIM': 32, 'DROPOUT': 0.2, 'LR': 0.01, 'WEIGHT\_DECAY': 0.0005, 'Q\_NEG': 1}

Config 4 - Epoch 01 | Train Loss 0.6679 | Val AUC 0.7507

Config 4 - Epoch 02 | Train Loss 0.5696 | Val AUC 0.8440

Config 4 - Epoch 03 | Train Loss 0.4708 | Val AUC 0.8792

Config 4 - Epoch 04 | Train Loss 0.3902 | Val AUC 0.8924

Config 4 - Epoch 05 | Train Loss 0.3451 | Val AUC 0.9002

...

Config 4 - Epoch 11 | Train Loss 0.2635 | Val AUC 0.9021

Config 4 - Epoch 12 | Train Loss 0.2546 | Val AUC 0.9015

Config 4 - Epoch 13 | Train Loss 0.2386 | Val AUC 0.8994

Config 4 - Epoch 14 | Train Loss 0.2490 | Val AUC 0.9061

Config 4 - Epoch 15 | Train Loss 0.2463 | Val AUC 0.9004

Config 4 - Early stopping at epoch 15.

Config 4 - Best Val AUC: 0.9073

--- Training Configuration 5/7 ---

Hyperparameters: {'HIDDEN\_DIM': 48, 'DROPOUT': 0.3, 'LR': 0.005, 'WEIGHT\_DECAY': 4e-05, 'Q\_NEG': 3}

Config 5 - Epoch 01 | Train Loss 0.5787 | Val AUC 0.6824

Config 5 - Epoch 02 | Train Loss 0.5043 | Val AUC 0.7932

Config 5 - Epoch 03 | Train Loss 0.4243 | Val AUC 0.8868

Config 5 - Epoch 04 | Train Loss 0.3232 | Val AUC 0.9354

Config 5 - Epoch 05 | Train Loss 0.2736 | Val AUC 0.9465

...

Config 5 - Epoch 26 | Train Loss 0.0885 | Val AUC 0.9719

Config 5 - Epoch 27 | Train Loss 0.0829 | Val AUC 0.9735

Config 5 - Epoch 28 | Train Loss 0.0850 | Val AUC 0.9721

Config 5 - Epoch 29 | Train Loss 0.0738 | Val AUC 0.9720

Config 5 - Epoch 30 | Train Loss 0.0762 | Val AUC 0.9739

Config 5 - Best Val AUC: 0.9739

Config 5 became new best model! Test AUC: 0.9657

--- Training Configuration 6/7 ---

Hyperparameters: {'HIDDEN\_DIM': 64, 'DROPOUT': 0.15, 'LR': 0.01, 'WEIGHT\_DECAY': 0.0001, 'Q\_NEG': 2}

Config 6 - Epoch 01 | Train Loss 0.5753 | Val AUC 0.8519

Config 6 - Epoch 02 | Train Loss 0.3649 | Val AUC 0.9368

Config 6 - Epoch 03 | Train Loss 0.2690 | Val AUC 0.9504

Config 6 - Epoch 04 | Train Loss 0.2259 | Val AUC 0.9469

Config 6 - Epoch 05 | Train Loss 0.1908 | Val AUC 0.9579

Config 6 - Epoch 06 | Train Loss 0.1819 | Val AUC 0.9487

Config 6 - Epoch 07 | Train Loss 0.1616 | Val AUC 0.9564

Config 6 - Epoch 08 | Train Loss 0.1615 | Val AUC 0.9553

Config 6 - Epoch 09 | Train Loss 0.1420 | Val AUC 0.9570

Config 6 - Epoch 10 | Train Loss 0.1308 | Val AUC 0.9519

Config 6 - Early stopping at epoch 10.

Config 6 - Best Val AUC: 0.9579

--- Training Configuration 7/7 ---

Hyperparameters: {'HIDDEN\_DIM': 16, 'DROPOUT': 0.25, 'LR': 0.001, 'WEIGHT\_DECAY': 0.0005, 'Q\_NEG': 1}

Config 7 - Epoch 01 | Train Loss 0.6872 | Val AUC 0.7351

Config 7 - Epoch 02 | Train Loss 0.6754 | Val AUC 0.7131

Config 7 - Epoch 03 | Train Loss 0.6656 | Val AUC 0.6916

Config 7 - Epoch 04 | Train Loss 0.6503 | Val AUC 0.6826

Config 7 - Epoch 05 | Train Loss 0.6302 | Val AUC 0.7128

Config 7 - Epoch 06 | Train Loss 0.6058 | Val AUC 0.7271

Config 7 - Early stopping at epoch 6.

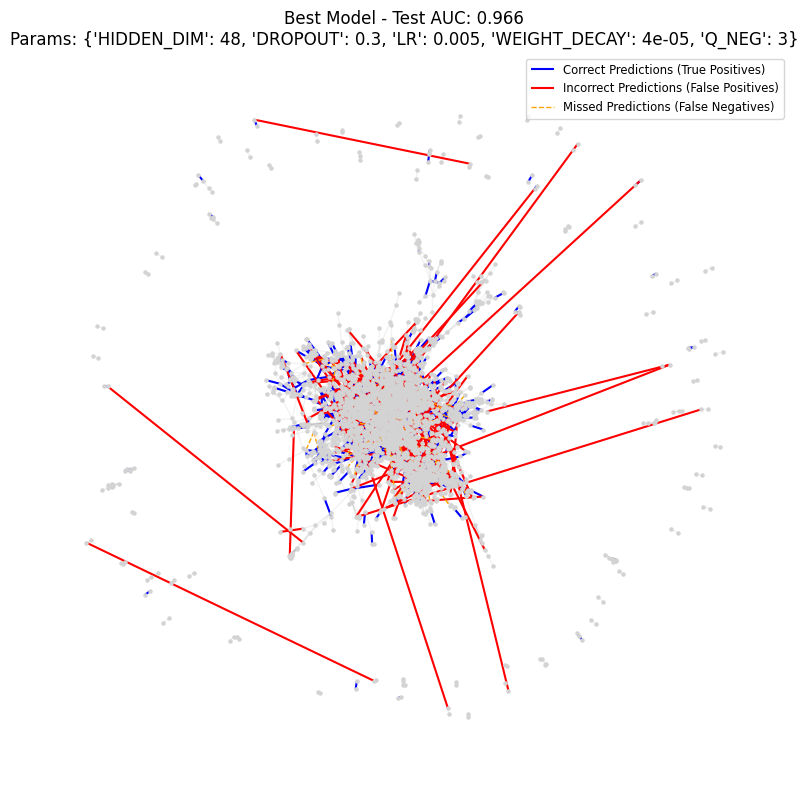
Config 7 - Best Val AUC: 0.7351

--- Hyperparameter Tuning Complete ---

Best Hyperparameters: {'HIDDEN\_DIM': 48, 'DROPOUT': 0.3, 'LR': 0.005, 'WEIGHT\_DECAY': 4e-05, 'Q\_NEG': 3}

Best Validation AUC: 0.9739

Corresponding Test AUC for the best model: 0.9657



**#4 Analysis / Discuss**

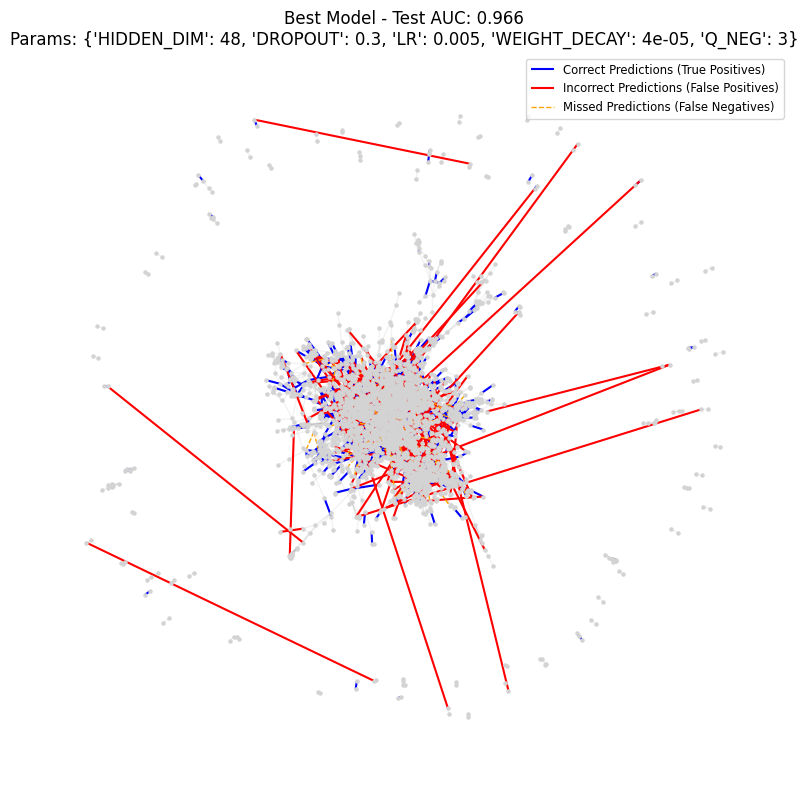
From the visualization result, we can find blue and grey edged clustered around center. And some red edges are pointed out from center to outside. Nodes around center are ones that connected with many other nodes, and nodes lying outside are ones that are hardly connected to other nodes (true connection).

* Without hyper parameter tuning. (Default)

A screenshot of a computer screen

AI-generated content may be incorrect.

* With hyper parameter tuning. (Final)



Result of visualization can be used for finding the prediction tendency of GCN model. The model predicts the connections of nodes around the center well but the wrong prediction tends to appear around boundary or sparse areas.

The improvement when hyperparameters are changed made wrong-predicted connections between nodes that are located far apart in sparse areas significantly reduced.

The most significant improvement came from the change of representation of final node embedding. My anaysis is that the concatenation of two nodes can preserve the information more than dot producting. But there is one potential problem, does the order of concatenation matter? Concatenation and can be lead to different result inside MLP. Maybe or maybe not, but I cannot ensure there won’t be. So positional invariance may be harmed.

Further potential improvements can be made by increasing epochs and ES patience since there are a few not early stopped configs.