Data Mining 21/22 Homework 4

Marco Calamo

December 2021

All work is tracked on Github at `https://github.com/IlKaiser/DM22_HW`

1 Exercise 1

The jupyter-notebook delivered with the homework is also available here for online consultation.
The version with the full graph (without graph and explainability visualization) is available here

1.1 Data Preparation

After having downloaded the dataset indicated in the homework text, I started testing the data I got.
In order to choose the disease object of the research, I opted for a disease with a great number of genes involved that grants, at the same time, a balanced dataset between nodes involved and not involved in the disease.
I choose the ***Breast Carcinoma***, the top fifth disease with most different genes involved (6775), and a positive genes to total ratio of 46.19%.
Then I went forward with extracting data from Protein-Protein Interaction dataset. First I tried with the full dataset, and while the overall classification performances of the models were slightly better than the reduced, the visualization of the graph was almost impossible and for the final report I used the **reduced dataset**.

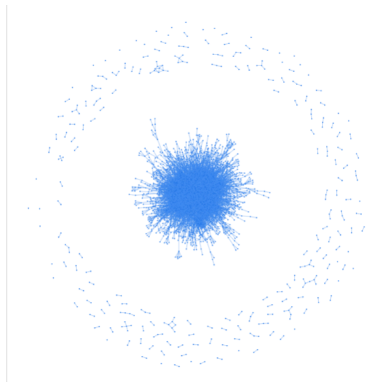


Figure 1: Visualization of the reduced graph using *pyvis* library

1.2 Model Training

1.2.1 Node2Vec

In order to have the P2P graph ready for being processed in a regular Neural Network, as requested, the nodes of the graph were encoded into 128-dimensions *embeddings* with the **Node2Vec** model from the ***pytorch-geometry library***.

1.2.2 Binary Classifier

For the classification I opted for a classic **multi-layer-perceptron (MLP)**, with one hidden layer and relu as activation function.

```
    Binary Classification(
  (layer 1): Linear(in_features=128, out_features=256, bias=True)
  (layer 2): Linear(in_features=256, out_features=256, bias=True)
  (layer out): Linear(in_features=256, out_features=1, bias=True)
  (relu): ReLU()
  (dropout): Dropout(p=0.1, inplace=False)
  (batchnorm1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True,
  track_running_stats=True)
  (batchnorm2): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True,
  track_running_stats=True)
)
```

After many different attempts, I found that 25 epochs are the best in order to obtain a nice trade-off between accuracy in train set and accuracy in the test set, avoiding overfitting.
The final accuracy with the test set is **61.0%** (this percentage goes up to **71.0%** when considering the full graph).
In the picture below it is possible to observe some scheme to sum the overall

performances of the model. We can see that the model is pretty average in its classification with nothing more than acceptable (especially the poor 0.5 in all categories when classifying the 1s) results in precision recall and f1.

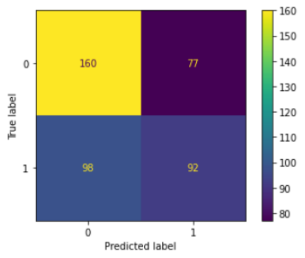


Figure 2: Confusion matrix for the test set

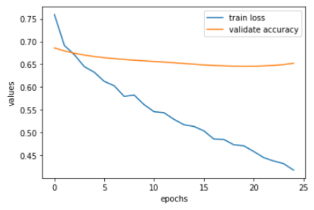


Figure 3: Training and evaluation of the MLP in 25 epochs

```
              precision    recall  f1-score   support

         0.0       0.62      0.68      0.65       237
         1.0       0.54      0.48      0.51       190

    accuracy                           0.59       427
   macro avg       0.58      0.58      0.58       427
weighted avg       0.59      0.59      0.59       427
```

Figure 4: Precision, Recall, F1-score and Support

1.2.3 Graph Neural Network

For the Graph Neural Network part I decided to adopt a simple Graph Convolutional Network as suggested. I obtained the best performances when I used the 128-dimensions embeddings computed previously as node input features. The model was also built with explainability in mind and made fully compatible with the *GNNExplainer* library.
The Network printed by the torch library is this:

```
    Net(
  (conv1): GCNConv(128, 16)
  (conv2): GCNConv(16, 2)
)
```

The overall performances of this model are far from ideal and not that better than the previous model: the accuracy in the test set is just **61.3%** (But with the full graph the prevision goes up to **71.7%**). However we can see overall better stats in precision recall and f1, making –in my opinion– this model preferable over the previous one.

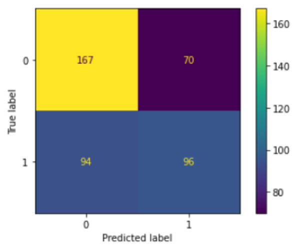


Figure 5: Confusion matrix for the test set

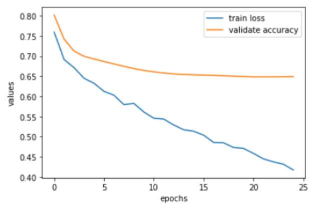
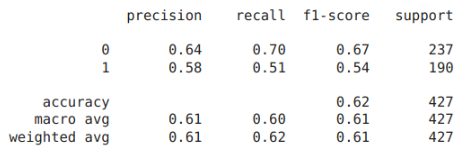


Figure 6: Training and evaluation of the GNN in 25 epochs



```
              precision    recall  f1-score   support

           0       0.64      0.70      0.67       237
           1       0.58      0.51      0.54       190

    accuracy                           0.62       427
   macro avg       0.61      0.60      0.61       427
weighted avg       0.61      0.62      0.61       427
```

Figure 7: Precision, Recall, F1-score and Support

1.3 Explainability

In order to explain and make sense to the obtained results, as mentioned before, I used the **GNNExplainer** library from *pytorch-geometric*.
In the picture we can see that the node 10 is classified correctly as 1 (involved) mostly because of the links with other correctly detected 1s nodes, like 11, 13, 14 and the others. We can say that 12 is the most important in this picture because aggregates the results of many different 1 nodes.

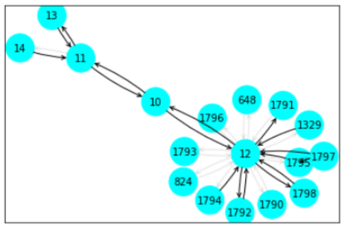


Figure 8: Classification as 1 explained for node 10 (MAP3K4 protein)

The same metrics apply for the node 114 correctly classified as 0, where the meaningful connection come from others node classified with 0 that aggregate many other nodes outcome.

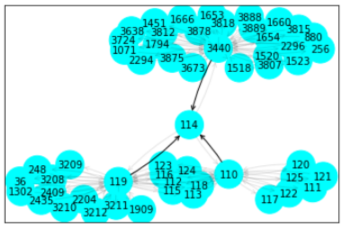


Figure 9: Classification as 1 explained for node 114 (DDX27 protein)

1.4 Conclusions

We can conclude that the proposed classification problem is really challenging and finding a neural network which produces good results is not a trivial problem: even the GNN performed about the same as the MLP which is not specifically designed for graph.
We could also state that the Node2Vec algorithm does a really good job in vectorization with meaningful parameters that actually helped while training the MLP.
Finally –as stated before– my choice for further analysis of this problem would still be the GNN network that produced the best results with the full graph maybe trying to tweak parameters or shape of the Network even further.