CS-502 Deep Learning in Biomedicine

Homework 2 Report

Cyril Achard
SCIPER - 310048

Note

Due to the 2-pages limit, more detailed information may be found in the provided Jupyter Book. Please refer to it for an interactive view of all figures and more detailed description.

1 Introduction

In this homework, we are concerned with the classification of molecules. We have molecules as graph data, and we seek to predict if the molecule is mutagenic or not, using the MUTAG dataset[1] as provided by HuggingFace. To this end, we will make use of several Graph Neural Networks architectures. We organize the data as follows:

- The adjacency matrix, recovered from the edge indices and padded up to the maximum number of nodes in the dataset. It will therefore have shape $N \times N$, where N is the maximum number of nodes in the dataset.
- The node features, padded up to the maximum number of nodes in the data, of shape $N \times F_{nodes}$, where F_{nodes} is the number of features per node.
- Finally, we will use the edge features, reshaped to $N \times N \times E_{feat}$, with E_{feat} being the edge feature dimension, serving as both adjacency matrix and edge features.

To incorporate edge features, we take inspiration from Gong et al.[2], to define a custom Attention-based Edge-Enhanced GNN, termed EGNN(A), in the following way:

Algorithm 1 Custom EGNN(A) layer

procedure EGNNA($\mathbf{X^{l-1}}, \mathbf{E}^{l-1}$)
  `Compute` $\alpha^l_{ijp} = f^l(X^{l-1}_i, X^{l-1}_j) \cdot E^{l-1}_{ijp}$
`for each` p
  `Normalize` α^l_{ijp} `on dimensions` i, j
  `Compute` $A^l_{ij} = a_{ij}||^P_{p=1}\alpha^l_{ijp}$
  `Update` $X^l = \sigma(A^l X^{l-1} W^l)$
  `Update` $\mathbf{E^l} = \alpha^\mathbf{l}$
  return $\mathbf{X^l}$, $\mathbf{E^l}$
end procedure

where i,j are nodes dimensions, p the edge feature dimension, $||$ the concatenation operator, f^l the attention score formula suggested in the homework, a_{ij} a weight matrix that brings the concatenation $||^P_{p=1}\alpha_{ijp}$ to the output shape of the layer ($N \times F_{out}$) by outputting A^l_{ij} (as in multi-head attention), and α is the edge-enhanced attention score. For simplicity, we use Group normalization in step 3.

2 Results

Four models weere defined : GCN, GraphSAGE, AttentionGCN and EGNN(A). Performance was compared under different layer numbers and dimensions, activation functions, pooling layers, weight initialization and the use of loss weighting. Due to time constraints and the scope of the homework, no large-scale cross validation across all possible parameters was performed, however the accuracy of the chosen "best" runs was validated by changing the training/validation split (not shown) and ensuring reproducibility. All models were trained with Binary Cross Entropy loss (with logits) for 800-2000 epochs, a 70/15/15 train/validation/test split and a batch size of 20 unless specified otherwise.

2.1 GCN

The GCN was the most heavily tuned, in order to find good baselines for each hyperparameter. For example, max pooling consistently scored higher on GCN, and this trend was observed across other models as well, therefore max pooling was kept for most of the later tuning.

Best hyperparameters for the GCN were :

- SELU or LeakyReLU : Negative features contributed to performance when used, therefore ReLU usage was discontinued.

- Max pooling: Instead of smoothing (mean) which consistently hindered performance, pooling was done according to the most salient features (max) before the classifier head.
- Kaiming initialization : An initialization scheme that does take into account non-linearities enabled better performance than Glorot or default initialization by a small margin.
- Loss weigthing, to account for the moderate class imbalance, tended to improve performance slightly (not shown due to reproducibility issues).

Model - **GCN**	**ROC-AUC**	**AP**
Mean pooling	0.738	0.863
ReLU	0.744	0.898
LeakyReLU	0.766	0.891
SELU	0.783	0.912
Glorot init.	0.827	0.928
Kaiming init.	**0.866**	**0.953**

For the layers, best performance was obtained by using four layers of size 256,256,128 and 64.

Model - **GCN**	**ROC-AUC**	**AP**
3x256 layers	0.772	0.881
256 to 8 layers	0.7	0.837
256, 64 layers	0.66	0.761
128,128,64,32	0.722	0.826
256,256,128,64	**0.866**	**0.953**

2.2 GraphSAGE

For GRAPHSage, tuning was first achieved by changing aggregation types. Tested aggregations included Sum, Mean, Square Root Mean and LSTM (see notebook). Best performance

Model - **SAGE**	**ROC-AUC**	**AP**
Sum Agg.	0..859	0.794
Mean Agg.	0.913	0.8
SqrtMean Agg.	0.858	0.794
LSTM Agg.	**0.953**	**0.872**

was achieved by the LSTM Aggregation, on which further tuning was performed due to overfitting. LSTM converged much faster for the loss and train accuracy, at the cost of higher variance. Attempts to reduce overfitting included increasing the dropout and changing the batch size/learning rate to slow convergence on training accuracy. Performance remained fairly stable despite these changes; with the best run at **0.877 ROC-AUC and 0.953 AP**, using a batch size of 30 and 14k epochs. These runs all had very close AP however so it is possible this particular one is overfitting the test set slightly.

2.3 Attention GCN

The Attention GCN was implemented as described, with the addition of multiplying the attention scores with the adjacency matrix, in order to obtain the score for first-order neighbors only as originally defined by [3]. It was also found that appending an extra layer of output size 32 increased performance. This architecture overall performed less well than the GCN, with good validation/test results but lower train accuracy convergence speed.

Model - **AttGCN**	**AUC**	**AP**
No adjacency	0..811	0.85
LR 10^{-3}	0.808	0.924
LR 10^{-4}	0.811	0.85
Extra 32 layer	**0.872**	**0.937**

2.4 Edge-Enhanced Attention GCN

This model was the best overall, with vastly increased convergence speed and performance, highlighting the usefuleness of edge features as additional predictors. Trained for 200 epochs only, it still outperforms all other architectures proposed here while being far more stable. Several layer architecture variants were tested, but the original 256,256,128,64 was still the best. The learning rate also had to be lowered to 10^{-4} to allow better descent.

Model - **EGNN**	**AUC**	**AP**
128,128,64 layers	0..811	0.909
256,128,64 layers	0.788	0.909
Extra 32 layer	0.738	0.882
Original layers	**0.911**	**0.964**

References

[1] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity", eng, Journal of Medicinal Chemistry **34**, 786–797, ISSN: 0022-2623 (1991) 10.1021/jm00106a046.

[2] L. Gong and Q. Cheng, "Exploiting Edge Features for Graph Neural Networks", en, in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019), pp. 9203–9211, ISBN: 978-1-72813-293-8, 10.1109/CVPR.2019.00943.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph Attention Networks*, en, arXiv:1710.10903 [cs, stat], Feb. 2018.