Reproducing Graph Convolutional and Attention Network Architectures

Vlad Marchidanu, Supervisor: Iulia Duta

May 2020

1 Introduction

The aim of my project was to familiarise myself with graph convolutional networks in their most basic form and then proceed to implement the idea of incorporating an attention mechanism into such networks as described in [3].

2 Dataset

I worked with the Protein-Protein Interaction Dataset [1] (PPI), which describes the interactions of protein types by means of an undirected graph. Proteins also have certain binary features. The goal of my task was to classify the protein types into a number of non-disjoint classes (a multilabeling problem).

Statistics on the dataset reveal that belonging to classes is relatively infrequent, but substantially more problematic is that features are highly scarce. Therefore an approach that takes into account neighboring nodes' features is highly needed.

The connected component analysis of the graph shows the feasibility of working with connected components for the implementation of the attention mechanism.

3 Architectures

Apart from a control MLP which takes into account only node features, two architectures were tried:

3.1 GCN

Graph convolutional network with mean as aggregation function
Forward pass:

$$\mathrm{h}_v^0 = \text{initial features of node v}, \forall v \text{ node in graph} \tag{1}$$

$$\mathrm{h}_v^k = \sigma\left(W_k \frac{1}{|N(v)|} \sum_{u \in N(v)} \mathrm{h}_u^{k-1} + B_k \mathrm{h}_v^{k-1}\right), \quad \forall k \in \{1, \ldots, n\} \tag{2}$$

, where n is depth of forward pass.
In implementation, sparse matrix multiplication was used, after [2]:

$$\mathbf{h}^k = W_k D^{-1} A \mathbf{h}^{k-1} + B_k D^{-1} \mathbf{h}^{k-1} \tag{3}$$

where $D = \mathrm{diag}(\mathrm{linavg}(A))$, a normalisation matrix, and W_k and B_k can be linear transformations or MLPs.

3.2 GAT

This architecture learns attention coefficients for pairs of neighboring nodes at each iteration of the forward pass. In implementation, adjacency matrices for each component were used for masking the matrix of all possible attention coefficients in order to faster, parallelised calculations of at each forward pass. For each component Comp:

$$C_k = \text{RowSoftmax} \circ \text{LeakyReLU}_{\alpha=0.2} \left(A_{\text{Comp}} * \left(a_1(W_k \mathbf{h}_i^{k-1}) + a_2(W_k \mathbf{h}_j^{k-1}) \right)_{i,j=1,\ldots,\dim(\text{Comp})} \right), \quad (4)$$

$$h^k = \sigma \left(C_k W_k \mathbf{h}^{k-1} \right), \quad (5)$$

, where A_{Comp} is the adjacency matrix of the component, $*$ is component-wise matrix multiplication and $a_1(\text{node}_i) + a_2(\text{node}_j)$ determines the attention between node i and node j

4 Results

In the following results table, a hybrid architecture is one in which the weights W_k and B_k are linear transformations of the same size for $k <$ # iterations and an MLP of 3 itermediary layers for the last iteration.

This architecture was adopted with the idea in mind that the depth of an MLP on the last iteration would be beneficial for obtaining a more representative embedding, while using an MLP for intermediary transformations might alter the intermediary representations of node neighbors too much.

A HybridAsc architecture is similar to a Hybrid one, with the exception that the transformations W_k and B_k double in output size with each iteration.

#	model type	MLP depth / # iters	weights type	embedding size	test score (%)
1	MLP	4	-	# feats	46
2	GCN	4	Hybrid	150	54
3	GCN	3	MLP	100	60
4	GCN	3	Hybrid	250	70
5	GCN	3	Hybrid	200	62
6	GCN	3	HybridAsc	150	77
7	GAT	3	Linear	200	45
8	GAT	2	Linear	150	46

Table 1: Results

5 Conclusion

The HybridAsc model showed best performance on this task - an improvement by 17 % compared to the referenced accuracy of GraphSAGE-mean in [3]. Unfortunately, the implemented GAT model showed no better performance than the MLP random control. Possible causes for this might be numeric instability or not using multiple independent attention functions at each iteration instead of one.

References

[1] URL `https://downloads.thebiogrid.org/BioGRID`.

[2] J. Leskovec. URL `https://snap-stanford.github.io/cs224w-notes/machine-learning-with-networks/graph-neur`

[3] P. Velickovi, G. C. A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *ICLR*, 2018. URL `https://arxiv.org/pdf/1710.10903.pdf`.