Graph Neural Networks Assignment 2

Szigetközi Zsolt

October 31, 2024

1 Implementations of GAT

1.1 Adjacency matrix version

For the adjacency matrix version, I implemented the batching mechanism by myself, the idea is that I put all the adjacency matrices in one that is block diagonal by the original adjacency matrices. For the GAT itself my idea was that pairing all the possibles nodes with each other, with the PyTorch repeat method, doing the dot-product with the attention weights, then masking the non-existing edges with the adjacency matrix, putting a very small negative number to the edges which are don't exist in the original graph. I did the multihead version aswell, here I just did the same but with another dimension and got the mean of the heads.

1.2 Edge index version

For this version, I also implemented the batching mechanism by myself, the idea is that I offset the edge indices in every batches to handle them separately. I used the Message Passing class, which has a pre defined message passing and aggregation mechanism. In the message method, it is possible to separate the source nodes and the target nodes, it is also pre defined by the class. I paired all the edges, here I did not need a masking, because here there are no non-exist edges. To handle the multiple heads, I had the idea to expand the dimension of the edge index tensor, flatten it, and offset the edge indices to handle the different heads separately. There is a softmax function in the PyG library that can be used for multihead mechanism since it can handle the heads separately. Here I also used the averaging between the heads.

2 Datasets

I used 2 datasets to solve this assignment, a graph-level classification and a graph-level regression.

2.1 Proteins

During my testing the self implemented GAT versions, I used the proteins dataset, because it was a simpler dataset since it has predefined node features and easier to test my implementation on it.

2.2 QM7B

This dataset is for graph-level regression, and it has no predefined node features, so I used random features from normal distribution with 0 mean and 1 standard deviation along the feature dimensions, as it is discussed in the [1] paper. I did this because the model should learn the structure of the graph. The dataset has only fully connected graphs in it, so the diameter of all the graphs is 1.

3 Comparison

The edge index version seemed to be more efficient in accuracy and in time, in my implementation. So for the further experiments, I used this version. Table 1 shows my experiments with different type of mechanisms, and the combinations of them. Since the diameter of the graphs is 1, I did not want to use too many layers, it can be seemed that with 3 layers and without skip connections the model is oversmoothing the graphs. Although with skip connections it had the second best performance. However the best performance was with the 2 layers, 4 heads per layers, pre-MLP and with skip connections. For the experiments I disabled the randomness in my code to get a deterministic behavior.

Layers	Heads	Pre-MLP	Layernorms	Dropout	Self-loops	Skip connection	MSE	MAE	RMSE
1	1	-	-	0.2	x	-	0.9288	0.7476	0.9638
1	4	x	-	0.2	x	-	0.7834	0.6682	0.8851
1	4	x	x	0.2	x	-	**0.7490**	**0.6495**	**0.8655**
1	4	x	x	0.2	x	x	0.7481	0.6487	0.8649
2	1	-	-	0.2	x	-	0.9345	0.7543	0.9667
2	4	x	-	0.2	x	-	0.7968	0.6814	0.8927
2	4	x	x	0.2	x	-	0.7515	0.6499	0.8669
2	4	x	x	0.2	x	x	**0.7360**	**0.6425**	**0.8579**
3	1	-	-	0.2	x	-	1.0566	0.7624	1.0279
3	4	x	x	0.2	x	x	**0.7401**	**0.6448**	**0.8603**

Table 1: Experimental Results Table

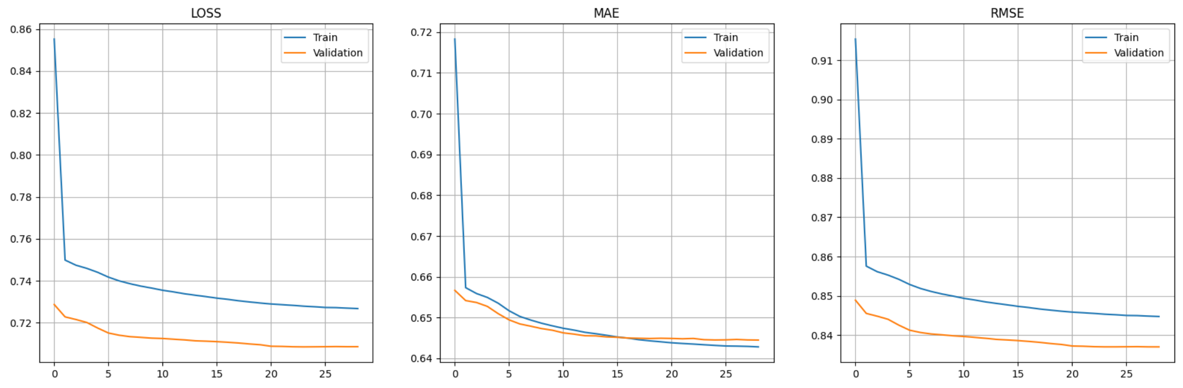

Figure 1: Caption

In Figure 1, the metrics during training with the best performed architecture can be seemed. I visualized the predictions of the best model on 5 samples in the test dataset, Figure 2 shows the results. It can be seemed that some features are matching very well, but some of them are just following the trend of the labels, but not overlapping smoothly.

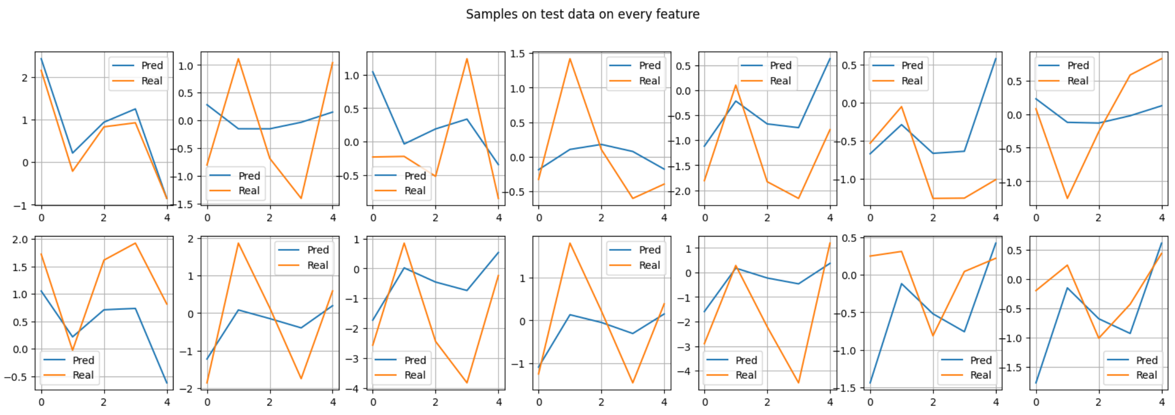

Figure 2: Caption

References

[1] Julio J. Valdés and Alain B. Tchagang. Understanding the structure of qm7b and qm9 quantum mechanical datasets using unsupervised learning, 2023.