Mini-Project Report: Inductive Node Classification with Attention

Fabian Beringer

August 2020

1 Introduction

Graphs are one of the most important concepts in computer science. Due to their ability to model network-like data, they've successfully been applied in a vast amount of applications, ranging from social networks, over knowledge graphs to biological networks. Because of its representational power, it is reasonable to assume that graph structured data can also be exploited in a machine learning setting.
For example, learning representations which explicitly model graph dependencies could encode valuable information that might be missed otherwise. This information may then be leveraged for downstream tasks, e.g. classification.
A popular approach to representation learning that has been shown to be effective is deep learning. However, the standard formulation of artificial neural networks is not designed to operate on graphs. Hence, various proposals for neural network architectures that are explicitly built to work on graph structured data have been made.
This project explores the Graph Attention Network architecture proposed by [Vel+18] which allows for the learning of graph representations by modeling node interactions based on an underlying graph structure. While the original work focuses on aggregating information from the local neighbourhood of each node, this project aims to explore the effects of incorporating information from higher order neighbours.

1.1 Background

1.1.1 Graph Neural Networks

In a common formulation for neural networks that operate on graphs, each node representation is learned by aggregating information passed from its neighbouring nodes. By applying this aggregation over multiple steps, information from higher order neighbours will also effect the learned node representation. In the most simple case, the neighbor information is aggregated by averaging, then a neural network is applied to compute an updated representation:

$$\mathbf{h}_v^0 = \mathbf{x}_v \tag{1}$$

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right) \tag{2}$$

Where $\mathbf{x}_v$ is the input representation of node v and $\mathbf{h}_v^k$ the nodes hidden representation after k aggregation steps over its neighbouring nodes $N(v)$. Both $\mathbf{W}_k$ and $\mathbf{B}_k$ are learned parameters of the neural network and $\sigma(\cdot)$ is a non-linear activation function.
This formulation comes with the advantage that weights can be shared across nodes which makes it more efficient to compute.

1.1.2 Graph Attention Networks

In Graph Attention Networks, the aggregation of neighbours is performed using attention. Attention is a mechanism that allows a neural network to weight its inputs based on their interactions. In this manner,

the network can learn to focus on more important inputs and to discard less important information.
In the case of graphs, each node may attend over the nodes in its neighbourhood. The node's updated representation then becomes a sum, weighted by attention scores that are computed using a compatibility function between the node and each of its neighbours.

$$\mathbf{h}'_v = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}\mathbf{h}_v\right) \tag{3}$$

As compatibility function, [Vel+18] chose a single layer Neural Network with LeakyReLU activation. Then, for each node, its attention scores are being normalized using a softmax function.

$$\alpha_{vu} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_v; \mathbf{W}\mathbf{h}_u])}{\sum_{k \in N(v)} \exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_v; \mathbf{W}\mathbf{h}_k])} \tag{4}$$

Here ; denotes concatenation and $\mathbf{a}$ is a learned parameter vector. Furthermore, they utilize multihead-attention, meaning multplie attention mechanisms are applied in parallel over the same input and then concatenated.

1.2 Problem Setting

This project deals with the supervised learning task of node classification. Given a graph as a set of vertices and edges (V, E) the goal is to predict the ground truth labels y for each node in V based on itself and its neighbourhood defined through E.
After training on a set of nodes we want to be able to generalize to new nodes which have not been trained on.
Because we're dealing with the inductive classification task, only a subset of the graph can be observed during training. The algorithm needs to generalize to previously unknown nodes that are added to the graph only during test time.

1.3 Task

We run experiments on the cora dataset. The dataset consists of 2708 scientific publications and 5429 links between them. Each publication is assigned to one of 7 classes where a class corresponds to its topic. Publication are represented as 1433 dimensional binary vectors where each entry of the vector corresponds to a word from the 1433 vocabulary being present in it or not. In order to run make the node classification an inductive tasks, during training, only a subset of the nodes and adjacency matrix can be observed.

2 Experiments

2.1 Research Question

In the original formulation of the GAT, its attention mechanism only considers its first order neighbours. However, higher order neighbours might also provide information about the graphs structure that encourages useful node representations. In order to assess whether attending over higher order neighbours helps with the node classification task, we try different schemes for extending the attention mechanism, to attend over higher order neighbours.

2.2 Proposed Methods

The original implementation of the graph attention layer uses the graph's adjacency matrix $A \in \mathbb{R}^{M \times M}$ to select the neighbours of each node that should be attended to. A straight forward approach to attend over nodes from up to k-th order neighbours, can be achieved by replacing A with a reachability matrix $A'_k = \sum_{i=0}^{k} A^i$, replacing all entries that are not zero with 1 and selecting the neighbours to attend to based on it. However because A'_k is only used for (hard-) selecting which neighbours are attended to, any

information about their relative position to each other in the graph is lost.
In order to tackle this problem, we evaluate the following approaches:

1. Apply a graph convolutional network (GCN) GCN like [KW16] layer in parallel to the attention layer, with the goal of infusing information about global graph structure. We retrieve the combined hidden combination of GCN and attention simply by summation.

$$h_{gcn} = \sigma(H' + A'_k H \mathbf{W}_{gcn} + \mathbf{b}_{gcn}) \tag{5}$$

   With $\mathbf{W}_{gcn}$ and $\mathbf{b}_{gcn}$ being learnable parameters, σ a non-linear activation function, H the stacked input nodes and H' the attention weighted node representations as computed in Eq. 3 (without applying the non-linearity).
   In addition we found that L_1 normalizing the rows of the reachability matrices helped with training stability.

2. Only assign a subset of attention heads to each higher order neighbourhood, the idea being that lower order neighbourhood information will be preserved, while higher order neighbours can also be considered.
   E.g. if we set the number of orders to k, we will assign $\frac{\#heads}{k}$ heads to attend over each of the k neighbourhoods. We refer to this as the *distributed* approach. This is in contrast with the *single* approach where all heads attend over the full expanded neighbourhood described by A'_k.

3. A combination of 1. and 2.

2.3 Experimental Setup

For our experiments we use a pre-defined train, validation, test split, each containing nodes from the cora citation graph. We follow the exact hyperparameter settings as [Vel+18] for cora: All models use a hidden size of 8 and 8 attention heads. For optimization we use the Adam optimizer with a learning rate of $5e-3$ and weight decay set to $5e-4$. All training instances are used for a weight update step and we train with early stopping based on validation loss, with patience set to 100 steps/epochs. For each run we test the model with the lowest loss on the validation set and report the highest accuracy achieved over 5 different runs. The implementation in pytorch can be found on github [1] for reproduction. All experiments were run on a single gtx 1070 gpu with 8 GB of memory.

2.4 Results

Our experimental results are summarized in table 1, where we report accuracy for each method on the validation and test set.
Firstly, if we look at the distributed approach, we can observe a slight improvement in accuracy, when assigning a subset of attention heads to a higher order neighbourhood. At $\#orders = 2$, half of the attention heads are attending over first order neighbours, while the other half is attending over second order neighbours. The increase suggests that interactions with second order neighbours might provide useful information for prediction. However, when increasing $\#orders$ further, we can observe a case of diminishing returns.
An explanation for this might be that not enough model capacity, in terms of number of heads, is being spend on the first order neighbourhood. However, we couldn't resolve the issue by simply increasing the total number of heads, so further experiments would be necessary to test this hypothesis.
Next, we can see rather large improvements being achieved by introducing the GCN layer in parallel to the attention mechanism. It is also worth noting that even though neighbourhoods of order two are still achieving the highest accuracy, the accuracy drop from further increasing the number od orders is being mitigated. Both, the single and distributed approach seem to benefit from the additional layer, with the distributed version performing slightly better.
These findings might indicate that the GCN layer can capture node interactions that the GAT is not able to model and vice versa. On the downside, we observed that even though training loss decreased smoothly over time, the validation performance suffered from large fluctuations. Therefore the approach seems to be more likely to miss maxima in terms of generalization ability, depending on initialization.

[1] https://github.com/yolomeus/DL2020Miniproject

method	#orders	accuracy validation	accuracy test
vanilla GAT	-	0.757	0.716
distributed			
	1	0.757	0.716
	2	0.770	0.734
	3	0.733	0.717
	4	0.713	0.665
single + GCN			
	1	0.790	0.778
	2	**0.803**	0.785
	3	0.773	0.781
	4	0.757	0.763
distributed + GCN			
	1	0.790	0.778
	2	**0.803**	**0.792**
	3	**0.803**	0.791
	4	0.773	0.784

Table 1: Accuracy of the different methods in the inductive node classification setting on the cora dataset. The highest values are shown in bold.

3 Conclusion

In this project, we explored different possibilities for expanding the Graph Attention Network for attending over higher order neighbours. We found that attending over second order neighbours can provide value for inductive node prediction on the particular dataset that we tested. We also found that GCN learned representations can further support the GAT's inductive node prediction ability, perhaps by additionally providing more detailed information about global graph structure.
While the combination of GCN and GAT looks like a promising direction to investigate, for now these results only apply to a single dataset. Further experiments need to be conducted in order to asses whether they still hold true on other datasets.
One might also want to explore different schemes for assigning the attention heads than in our distributed approach, to put more weight on close neighbours. It is still open whether the extension of attention over neighbours with order larger than two can be beneficial, e.g. with more carefully crafted approaches, higher model capacity or in different scenarios.

References

[KW16] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[Vel+18] Petar Veličković et al. "Graph Attention Networks". In: *International Conference on Learning Representations* (2018). accepted as poster. URL: `https://openreview.net/forum?id=rJXMpikCZ`.