CSCE 676 600: REPORT 1

Name: Rohan Chaudhury

UIN: 432001358

10th February 2022

Paper Title: Scaling Graph Neural Networks with Approximate PageRank

Paper Citation: Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling Graph Neural Networks with Approximate PageRank. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, New York, NY, USA, 2464–2473.
DOI:https://doi.org/10.1145/3394486.3403296

1. SUMMARY:

a. Motivation:

The authors were motivated to work on scaling up Graph Neural Networks (GNN) because GNNs have proven to be highly useful in multiple network mining tasks and they show better performance than classical methods. However, GNNs have very limited scalability, especially when the sizes of the graph increase into millions, and existing solutions to this problem are similarly prohibitively expensive. The scalability issue arises from the fact that many contemporary GNN models rely on the recursive message passing technique for information propagation through the graph. The recursive message passing method computes the hidden representation of a node by aggregating information from its neighboring nodes at every layer. The neighboring nodes perform aggregation of information from their own neighboring nodes and so on. This recursive nature of the method gives rise to this neighborhood expansion issue which creates the scalability and performance bottleneck. To tackle this issue, the authors have proposed the PPPRGo model. This model achieves considerable speed increases while keeping state-of-the-art prediction performance by using an effective approximation of information diffusion in GNNs. The model is also scalable and can be easily parallelized for huge datasets.

b. Main technical contributions and previous work leading to it:

Sampling method used in previous research to solve scalability issue and its drawbacks:

To solve the scalability issue, a popular technique described in previous research is to sample the graph to reduce the neighborhood size during training, with the design of the sampling scheme being one of the fundamental differences between the different scaling techniques. For example, selecting a specified number of nodes from a given node's k-hop neighborhood to compute its prediction. But this method has 2 major drawbacks: (1) the trade-off between runtime and accuracy is exceedingly severe in these techniques as they depend on the multi-hop message passing procedure. (2) sampling does not effectively decrease the number of nodes that must be retrieved

for many of the proposed approaches, as we have to calculate the importance scores first in many of these approaches.

As a result, the authors came to the conclusion that this wasn't the ideal technique for resolving the problem and instead discovered that just a few neighbors were genuinely relevant for a node's final prediction. The primary issue now is being able to determine how to locate them efficiently.

Klicpera et al. method of detaching feature transformation from propagation and its drawbacks:

Klicpera et al. proposed another way for dealing with the issues of the message passing system. They advocated detaching feature transformation from propagation in their PPNP model where predictions are initially generated for each node using its own features followed by propagating it using a personalized PageRank adaptation. As demonstrated in recent research, personalized PageRank may be utilized to effectively include multi-hop neighborhood information of a node without the need for explicit message-passing since personalized PageRank propagation correlates to an infinite number of neighborhood aggregation layers, with node effect decreasing exponentially with each layer. Their PPNP model is given as:

$$Z = \text{softmax}(\Pi^{\text{sym}} H), \qquad H_{i,:} = f_\theta(\boldsymbol{x}_i)$$

Where Z= Matrix where the rows represent the prediction vectors for the nodes after propagation, H=Matrix where the rows represent the vectors for the node, $\Pi^{\text{sym}} = \alpha(I_n - (1-\alpha)\tilde{A})^{-1}$ is the symmetric propagation matrix, $\tilde{A} = D^{-1/2}AD^{-1/2}$ is the normalized adjacency matrix, and α is the teleport probability. A neural network f_θ generates the local per-node representations $H_{i,:}$. The authors had proposed a variant of power iteration for calculating the final predictions since the direct computation of Π^{sym} is expensive. However, since these computations need to be done during every gradient update step, for large graphs even a modest number of power iteration evaluations becomes very expensive.

Further work done by the authors in this paper by utilizing previous research and solving their drawbacks:

So the PPNP model deals with the issues of message passing technique by separating feature transformation from propagation but calculating the dense propagation matrix using power iterations becomes a bottleneck for large graphs. To deal with this issue, the authors in this paper have proposed a method that eliminates the need for doing the power iterations during training. In their approach instead of calculating Π^{sym}, the authors are approximating the personalized PageRank matrix Π^{ppr} which is given as:

$$\Pi^{\text{ppr}} = \alpha(I_n - (1-\alpha)D^{-1}A)^{-1}$$

The rows of this matrix $\pi(i) := \Pi^{\text{ppr}}_{i,:}$ correspond to the personalized PageRank vector for the nodes. Because most of the probability mass in these vectors is localized on a limited number of nodes, the authors hypothesized that we can approximate $\pi(i)$ with a sparse vector by truncating small elements to zero and that can, in turn, allow us to approximate Π^{ppr} with a sparse matrix $\Pi^{(\epsilon)}$, which can be directly used for information propagation or can be renormalized to an approximation of Π^{sym}. Thus the final predictions of the PPRGo model can be defined with the following equations:

$$Z = \text{softmax}(\Pi^{(\epsilon)} H), \qquad H_{i,:} = f_\theta(\boldsymbol{x}_i)$$

For obtaining the rows of the matrix $\Pi^{(\epsilon)}$, the authors had used the push-flow algorithm described in **Andersen et al**. Then additional truncation of this matrix is carried out by the authors to consider the nodes with top k highest scores according to $\pi(i)$, for each node i. So the predictions for each node i is given as:

$$z_i = \text{softmax}\left(\sum_{j \in \mathcal{N}^k(i)} \pi^{(\epsilon)}(i)_j H_j \right)$$

where $\mathcal{N}^k(i)$ counts the indices of the top k non-zero elements in $\pi^{(\epsilon)}(i)$. To compute the final prediction for a given node, the above equation shows that we only need to evaluate a minimal number of nodes. Moreover, by raising or reducing the number of neighbors k, we may effectively trade-off scalability and performance. Also, by changing the value of α the amount of information that is being incorporated from the neighborhood of a node can be changed as: (1) values of α closer to 1 indicates placing greater emphasis on the node's immediate surroundings as the random walks would teleport to that particular node i more frequently and, (2) values of α closer to 0 indicates placing a greater emphasis on the node's extended neighborhood. Also, it can be seen that the k-hop neighborhood's significance is proportional to $(1-\alpha)^k$. So, by changing the values of α and k we can change the effective neighborhood size whereas, in message-passing frameworks, additional layers were required to include information from the extended neighborhood which dramatically increased the computational cost.

Distributed training and efficient inference:

The model training consists of 2 stages which are carried out in a distributed fashion. The first stage involves pre-computing the approximated personalized PageRank vectors using an efficient batch data processing pipeline similar to MapReduce and the second stage involves using stochastic gradient descent to train the model parameters. Since these 2 stages are implemented in a distributed fashion, it dramatically decreases the training time.

During Inference time, since the runtime required for calculating the personalized PageRank vectors can be large when the number of test nodes is large, the authors have argued that it is more efficient to use power iteration in this case. The authors are using only a few power iteration steps during inference since after that there is no significant increase in accuracy with increasing the number of power iteration steps. The final bottleneck during inferencing is calculating the output of the neural network f_θ for the nodes, which can be computationally expensive in certain cases. To avoid this, the authors are taking advantage of the graph's homophily to reduce the number of nodes that are being analyzed during inferencing. Utilizing these above techniques the authors were also able to reduce the inference runtime significantly without sacrificing accuracy.

c. Interesting experimental result/theoretical finding:

The authors had compared the performance of the PPRGo model with 2-hop and 3-hop GNN models and the FastGCN models. The authors made certain that the models were all trained in the same distributed manner and on the same infrastructure and that the multi-hop models and the PPRGo models use the same number of neighbors. Under these conditions, the authors had observed there is a huge reduction in the runtime of the PPRGo model without any significant drop in predictive performance with an accuracy of around 61% for all the models. The authors show that the PPRGo model is faster, takes fewer steps to converge, and with an increase in the number of workers the model becomes even faster than the baseline models. Also, the speed decrease in the PPRGo model is the least as compared to the multi-hop models when the authors had increased the value of k which

controls the number of top-k neighbors to be considered for prediction. This is because, in contrast to multi-hop models, which must recursively update the hidden representations, PPRGo can process all top-k neighbors at once.

On comparing the performance of Cluster-GCN and SGC models with the PPRGo model on the Reddit dataset, the authors found that both the models are far slower, utilize more memory, and have worse accuracy than the PPRGo model.

As shown by the authors in their experiments, the PPRGo model inference time can also be reduced by reducing the number of nodes considered during inference. They found that when the number of nodes considered is reduced by a factor of 10 (which reduces the inference time by 50%), there is a drop of only 0.6% accuracy. This particular result fascinates me the most as an increase in inference speed by such a significant amount (with not much decrease in accuracy) is a huge benefit for production environments especially the ones where online inferences are required.

2. MY THOUGHTS:

a. Pros and challenges they address:

i. The PPRGo model proposed by the authors has better performance than baselines on both single-machine and distributed systems.

ii. To show the efficiency of their approach the authors had introduced a new benchmark graph (MAG-Scholar) with millions of nodes, edges, and features. This is an important contribution as the number of large baseline graph datasets is less and this benchmark graph can also be used in further research.

iii. The authors also show that training their model from scratch on their benchmark graph and then predicting all the labels of the nodes in that graph takes less than two minutes on a single machine which is much better than the other baseline results on that graph. In the real-world scenario, speed is a very important factor, and their method does a very good job at optimizing it.

iv. Previous work on GNN scalability has been shown on graphs with very few nodes they mostly focus on improving single machine scalability whereas many fascinating network mining issues include graphs having billions of nodes and edges, which necessitate distributed processing over multiple machines.

v. In this method, the effective neighborhood size can be altered by changing the values of α and k, but in message-passing frameworks, extra layers were necessary to include information from the extended neighborhood, which increased the computational cost substantially.

b. Limitations:

i. For information propagation, the personalized PageRank matrix Π^{ppr} is approximated with a sparse matrix $\Pi^{(\epsilon)}$, and then additionally it is truncated to consider only the nodes with top k highest scores for each node prediction calculation. This approximation and truncation may or may not lead to desirable accuracies in all the situations.

ii. The authors are falling back to using power iterations during inference time since calculating the personalized PageRank vectors for a large number of test nodes can be time-consuming. Moreover, they are only using a few power iteration steps during inference which may result in a decrease in accuracy.

iii. To avoid the bottleneck which may arise from calculating the output of the neural network f_θ for the nodes during inferencing, the authors are reducing the number of nodes that are being analyzed by taking advantage of the graph's

homophily. This can have adverse effects on the accuracy as the authors are considering only a few nodes during inferencing.

3. **FOLLOW UP/NEXT STEPS:**
    a. I am excited to see the application of this technique to solve large graph-based problems in real and online situations. The results indicate that there is a significant increase in speed with almost the same accuracy in comparison to the other methods available to solve similar problems. That would mean we would get similar results but at lesser runtime.
    b. Instead of falling back to using a few power iteration steps during inference time and reducing the number of nodes to be analyzed to avoid the bottleneck of calculating the output of the neural network f_θ for many nodes, research can be undertaken to find out if some other method can avoid these limitations instead of using these approximations.
    c. I feel like the approximation to calculate the personalized PageRank matrix can be decreased to better improve the accuracy with this method. Further research can be pursued in that direction to find better approximation techniques.
4. **REFERENCES:**

    a. https://www.youtube.com/watch?v=nsngoAycymE
    b. https://figshare.com/articles/dataset/mag_scholar/12696653
    c. https://github.com/TUM-DAML/pprgo_pytorch
    d. https://www.in.tum.de/daml/pprgo/
    e. Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling Graph Neural Networks with Approximate PageRank. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, New York, NY, USA, 2464–2473. DOI:https://doi.org/10.1145/3394486.3403296