Data Augmentation for Few-Shot Learning on Graphs

November 9, 2022

Riccardo La Marca

Abstract

The goal of this project is to compare different techniques of training when dealing with the so-called *Few-Shot Learning*. The code is available https://github.com/lmriccardo/few-shot-graph-classification

1. Introduction

When training a classifier, we need as much labeled data as possible in order to build an accurate and better generalizing model. However, most of the time we don't have this huge amount of informations. To overcome this problem, *Few-Shot Learning*, and with it also *Meta-Learning*, is started being used. In general we can group FSL problems into *Few-Shot Classification* and *Few-Shot Regression*. In this project I'm going to focus on the former, specifically referred to graph classification problems.

Few-Shot Classification. Let's say we have the train, validation and test set where train/validation and test set do not share any classes. In FSC, first we sample N classes from those of the train/validation set and then for each class we sample $K + Q$ data, for a total of $N \times (K + Q)$ graphs. The first $N \times K$ samples are called **support data**, while the latter $N \times Q$ are the **query data**. Given labeled support data, the goal is to predict the labels of query data. Note that in a single task, support and query data share the same class space. This is also called **N-way-K-shot** learning. At test stage when performing classification tasks on unseen test classes, we firstly fine tune the learner on the support data, then report classification performances on the query set. You can find more about FSL in (Wang & Yao, 2019).

2. Techniques

In this section I would like to give a brief description of all the techniques that I found being most interesting of GDA for few-shot graph classification, given also AS-MAML. I choose this GDA techniques since I decided to deal with graph-classification tasks, and because they overall cover all the possible augmentation methods for graphs.

AS-MAML. Adaptive-Step MAML (Ma et al., 2020) is a meta-learning techniques based on the most famous MAML (or Model-Agnostic Meta-Learning) (Finn et al., 2017). It works exactly like the original MAML implementation, but with the addition of the so-called *step-adaptation* using an Adaptation Controller. It was introduced to alleviate the hard problem of finding the optimal learning rate and step size in MAML techniques. To this end they decided to use an RL-based Controller to decide the optimal step size for the adaptation given the training loss and the embdedding quality, computed using the ANI (Average Node Information). Moreover, based on the same two measures the controller should also know when to stop the adaptation. The stop probability is computed using a StopController model that uses LSTM layers and a last sigmoid. Finally, the controller is updated using gradient ascent with a computed reward, as it is classic in Reinforcement Learning techniques.

Model Evolution. M-Evolve (Zhou et al., 2020) is a graph data augmentation technique based on adding/dropping edges. The way we drop or add more edge to a graph is driven by heuristics. In this case we have the so-called *motifs-similarity mapping*. Motifs-similarity mapping uses the concept of motifs (Chen et al., 2021), and in this particular case of a precise motif called *open-triad*. The idea is to randomly remove one edge for each open-triad in the graph via weighted random sampling, and add other edges always via weighted random sampling to these motifs in order to preserve structural properties. All the weight are computed using vertex similarity score with the *Resource Allocation Index* (Li et al., 2017/04). Finally, to choose which of the new generated data should be added to the training set, we compute the matching degree between the new sample and their label via *label reliability*. Essentially, we compute the label reliability score and if it is bigger than a threshold we put this new data into the train set.

FLAG. Free Large-scale Adversarial Augmentation on Graphs (Kong et al., 2020) is another GDA technique, but in this case based on adversarial perturbation of node fea-

Email: <lamarca.1795030@studenti.uniroma1.it>.

Deep Learning and Applied AI 2022, Sapienza University of Rome, 2nd semester a.y. 2021/2022.

tures of a graph. The overall idea is to iteratively craft gradient-based adversarial perturbations and apply them to node features during the training phase. In this way, by making the model invariant to small fluctuations in input data, this method help models to generalize out-of-distribution samples and boost model performance at test time. Moreover, to fully exploit the generalizing ability and enhance the diversity and quality of adversarial perturbations, they proposed to craft multi-scale augmentations using the so-called *free-training*: while computing the gradient for the perturbation, we simultaneously produces the model parameter on the same backward pass. Finally, since we are going to perturb only node features, the entire structure of the new graph remains the same.

$\mathcal{G}$-Mixup. $\mathcal{G}$-Mixup (Han et al., 2022) is another GDA technique. In this case instead of changing node features or drop/add edges, we want to generate new graphs by applying Mixup (Zhang et al., 2017) procedure to two randomly sampled graphs. However, Mixup is defined for data that lives in the Euclidean space, and we know that graphs don't. This is quite challenging, so *what can we do?* We use *graphons*, i.e., continuous and symmetric functions such that given two nodes it returns the probability that there exists an edge connecting those nodes. It represents a kind of edge distributions from which we can sample graphs. Graphons are matrices, so they lives in the Euclidean space. For this reason, instead of mixing up two graphs we can mixup two graphons, and from the resulting graphon we sample the new graph data. One principal difference with respect to the other GDA techniques, is that in this case we are going to use one-hot encoded labels.

3. Experimental results

To compare the four techniques I decided to use the same three datasets from the AS-MAML paper: TRIANGLES (TR), COIL-DEL (CD) and Letter-High (LH). Note that only a subset of TRIANGLES graphs have been used. Each dataset has been trained, validated and tested using the Graph SAGE model: 3 SAGE convolutional layers, 3 SAGPool layers and 3 final FC layer, all of them using the LeakyRELU activation function except for the last linear layer for which the softmax is used. Each model has been trained using 200 epochs each of them running 200 training and validation episodes, then tested using only 200 testing episodes. Finally, the configuration for the few-shot sampling is the same for all dataset: 3 classes (way) for train, test and validation, 10 samples (shot) for support train and test/validation and 15 samples (query) for query train and test/validation. At the end, each episode had 75 graphs. These are the otained results

A%	**TR**	**CD**	**LH**
AS-MAML	82.47	86.51	59.86
+MEvolve	**85.48**	**92.12**	**62.39**
+FLAG	84.91	90.73	61.07
+$\mathcal{G}$-Mixup	83.73	89.98	60.68

Table 1. Test accuracies

DATASET	$\lvert\mathcal{G}\rvert$	**Avg.**$\lvert\mathcal{V}\rvert$	**Avg.**$\lvert\mathcal{E}\rvert$
TRIANGLES	45000	28.85	35.50
Letter-High	2250	4.67	4.50
COIL-DEL	3900	21.54	54.24

Table 2. Dataset Statistics.

4. Conclusion

The goal of this project was to compare different graph data augmentation techniques for few-shot learning, used to improve generalization by augmenting the train set with new samples. From (Zhao et al., 2022) I chose this three techniques because, among all those present in the paper, they were the ones that in my opinion seemed to be the most promising based on the results of the respective papers. Among the three, based on the above results, the most promising is *MEvolve*. This is not a case of course. In *MEvolve* we combine three important steps: data augmentation, data filtration and retraining of a pre-trained model. What effectively increase the performances of this GDA technique is the data filtration phase. During this step only a portion of the newly produced data will belong to the augmented training set, i.e. those data which "score" is less than a threshold. Moreover, the threshold is dynamically optimized using validation results. This means that better is the model, more precise will be filtration. The only limitations of *MEvolve* is that it does not scale on very large graphs (real social network graphs for example), becoming extremely inefficient. Immediately after MEvolve, there is FLAG which performances differ just by 1%. Differently from the previous one, in FLAG we create new data by adversarial perturb node features in order to make the model more robust to small fluctuations. All of this is additionally boosted by the so-called *free training*. While MEvolve aim to graph classification tasks, FLAG can be used to improve both graph classification and node classification tasks. Moreover, it scale with the size of real-world graphs. Finally, we have $\mathcal{G}$-Mixup and its three step data augmentation: (1) estimate graphons; (2) Mixing up two graphons; (3) Sample new data from the resulting graphons. Like FLAG it is model-agnostic and its performance scale with the dimension of the graphs. Moreover, differently from the previous GDA techniques, it combines the topology of more than one graph to create new data. Maybe the greatest limitation derives from the randomness of the new data

which is it not "driven", differently respect to MEvolve that instead implements the data filtration step.

References

Chen, X., Cai, R., Fang, Y., Wu, M., Li, Z., and Hao, Z. Motif graph neural network. *CoRR*, abs/2112.14900, 2021. URL https://arxiv.org/abs/2112.14900.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

Han, X., Jiang, Z., Liu, N., and Hu, X. G-mixup: Graph data augmentation for graph classification, 2022. URL https://arxiv.org/abs/2202.07179.

Kong, K., Li, G., Ding, M., Wu, Z., Zhu, C., Ghanem, B., Taylor, G., and Goldstein, T. FLAG: adversarial data augmentation for graph neural networks. *CoRR*, abs/2010.09891, 2020. URL https://arxiv.org/abs/2010.09891.

Li, L., Bai, S., Yang, S., Qu, L., and Yang, Y. Link prediction via extended resource allocation index. In *Proceedings of the 2017 5th International Conference on Frontiers of Manufacturing Science and Measuring Technology (FMSMT 2017)*, pp. 455–460. Atlantis Press, 2017/04. ISBN 978-94-6252-331-9. doi: https://doi.org/10.2991/fmsmt-17.2017.96. URL https://doi.org/10.2991/fmsmt-17.2017.96.

Ma, N., Bu, J., Yang, J., Zhang, Z., Yao, C., and Yu, Z. Few-shot graph classification with model agnostic meta-learning. *CoRR*, abs/2003.08246, 2020. URL https://arxiv.org/abs/2003.08246.

Wang, Y. and Yao, Q. Few-shot learning: A survey. *CoRR*, abs/1904.05046, 2019. URL http://arxiv.org/abs/1904.05046.

Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. URL http://arxiv.org/abs/1710.09412.

Zhao, T., Liu, G., Günnemann, S., and Jiang, M. Graph data augmentation for graph machine learning: A survey. 2022. doi: 10.48550/ARXIV.2202.08871. URL https://arxiv.org/abs/2202.08871.

Zhou, J., Shen, J., Yu, S., Chen, G., and Xuan, Q. M-evolve: Structural-mapping-based data augmentation for graph classification. *CoRR*, abs/2007.05700, 2020. URL https://arxiv.org/abs/2007.05700.