Graph Learning by Dynamic Sampling

Luca Hermes
Machine Learning Group
Bielefeld University, Germany
lhermes@techfak.uni-bielefeld.de

Aleksei Liuliakov
Machine Learning Group
Bielefeld University, Germany
aliuliakov@techfak.uni-bielefeld.de

Malte Schilling
Autonomous Intelligent Systems Group
University of Münster, Germany
malte.schilling@uni-muenster.de

I. Introduction

Over the last years, graph convolutional neural networks (GCNs) [4], [11] have become a standard model to represent graph-structured data. The basic underlying principle in these models is to propagate node features along the graph edges which integrates information over the graph space. Investigations on the effectiveness of GCNs have shown that node dependencies over large distances on the graph pose one such challenge. Integrating information over large distances usually requires to stack multiple GCN layers, as a single layer usually integrates a 1-hop neighborhood and every layer added increases this receptive field by one further hop. Li et al. [5] argues that the neighborhood aggregation method is a form of Laplacian smoothing, it smoothes out the integrated signal when repeated over multiple layers. As a consequence, nodes become indistinguishable and the aggregated information is summarizing large parts of the whole graph instead. This problem is known as the *over-smoothing* problem, which is inherent to the smoothing kernel of GCNs.

A second obstacle is related to structural features of certain graphs, i.e. heterophily. Such graphs are characterized by the tendency of connected nodes to belong to different classes. This property greatly impedes the performance of GCNs. [8] and Yan et al. [12] relate the effect to the smoothing behavior of GCNs. To approach these two problems, recent research has turned toward altering the graph structure to improve the information conductance or on devising different aggregation techniques that counteract the degrading performance in more heterophilic settings.

Here, we propose a method that utilizes a sparse form of message-passing. Specifically, we investigate a sampling-based method to focus the aggregation to only a few salient neighbors. We propose a framework that can be used to learn the message function and the selection of the sparse message pathways jointly. This method is build around the concept of walks along a given graph that we use to directly aggregate features from higher-order neighbors. In order to find important graph elements, we deploy a differentiable sampling mechanism that is optimized jointly with the given downstream objective. As walks follow distinct paths, the information growth is kept linear instead of growing exponentially as in dense aggregation methods. We now introduce all the individual building blocks and finally demonstrate the effectiveness especially in heterophilic settings using common transductive graph benchmarks.

II. Dynamic Sampling Graph Neural Network

In the following, we formalize our dynamic graph sampling process and notation convention. We consider input graphs $\mathcal{G} := (\mathcal{V}, \mathcal{E}, X)$ that consist of a set of nodes $v \in \mathcal{V}$, edges $(v_i, v_j) \in \mathcal{E}$ and node features $X \in \mathbb{R}^{N \times d}$, where $N = |\mathcal{V}|$ and d is the dimensionality of the node features. We denote the adjacency matrix as A and $A_{i,j} > 0$ if $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise.

Conceptually, our approach implements a random-walk, where walkers traverse $\mathcal{G}$ iteratively. Starting from a node v_s, these walkers can visit nodes $v_j \in \mathcal{N}^T(v_s)$ over T sampling steps, where $\mathcal{N}^k(v_i)$ denotes the k-hop neighborhood around a node v_i. Usually the sampling procedure of a random-walk is purely Markovian, as in each step the transition probabilities from a node v_i to a neighbor $v_j \in \mathcal{N}(v_i)$ are similarly modeled as $p_{v_i,v_j} = D(v_i)^{-1}$, i.e. the inverse node degree of v_i. In contrast, we deploy stateful walkers $w_i \in W$ that integrate node features along the walk. The probabilities to transition from a node $v_c \in \mathcal{V}$ to one of its neighbors depend not only on v_c, but also on the state of the walker, which represents an additive Markov process. We ultimately use the walker states to update the nodes $Pos^0(w_i)$ where the trajectories of the respective walkers originated. Here, $Pos^t(w_i) \in \mathcal{V}$ denotes the position of walker w_i^t at the sampling step t.

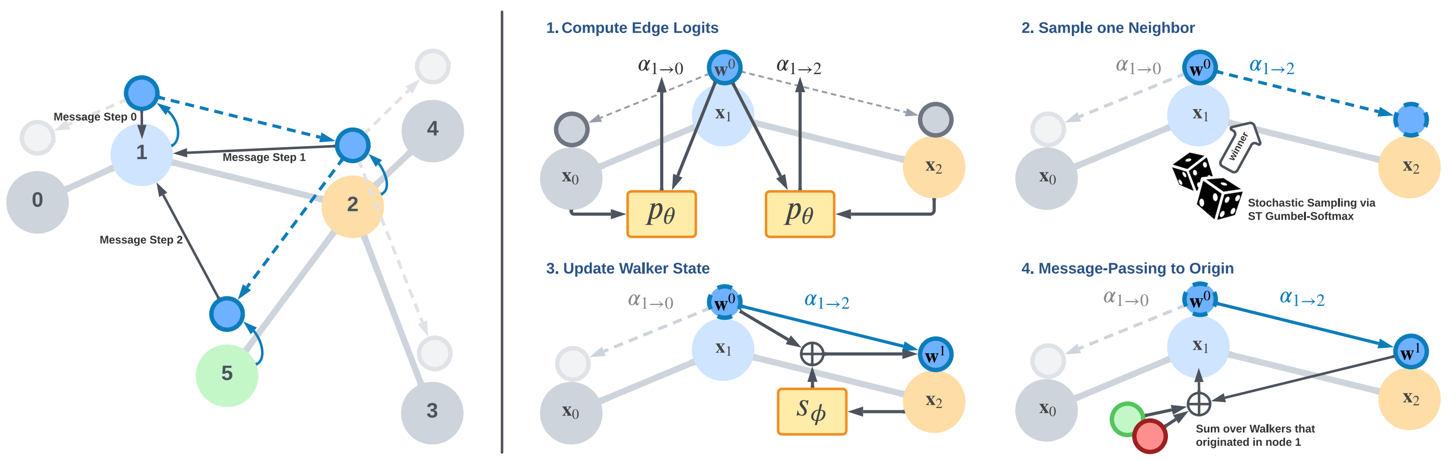

Fig. 1. Overview of the dynamic sampling GNN framework. Left: An individual walker (dark blue) sampling two steps, updating its own state (solid blue arrows) and the node features of the origin node (solid black arrows). Right: Steps performed to sample a single step – here from node 1 to node 2: Model p_theta computes edge logits (1) from which the trajectory is sampled to select a single neighbor (2). The walker traverses to the selected neighbor and updates its own state using s_ϕ (3). Finally, the walker updates its origin node (4). If multiple walkers (here red and green)—that originated in the same node 1—meet, their states contribute equally to the node update (4). Computations are denoted by solid black arrows, yellow boxes denote parameterized functions and dashed lines denote sampling trajectories.

TABLE I

RESULTS FOR TRANSDUCTIVE GRAPH BENCHMARKS. THE VALUES ARE ACCURACIES AVERAGED OVER ALL 10 DATA SPLITS AND WE INCLUDE THE STANDARD DEVIATION. THE BEST PERFORMING MODEL IS HIGHLIGHTED IN BOLD AND THE SECOND BEST IS MARKED WITH ITALIC.

	Texas	**Wisconsin**	**Actor**	**Squirrel**	**Chameleon**	**Cornell**	**Citeseer**	**Pubmed**	**Cora**
homophily	0.11	0.20	0.22	0.22	0.24	0.13	0.74	0.80	0.81
# Nodes	183	251	7600	5201	2277	183	3327	19717	2708
# Edges	325	515	33391	217073	36101	298	4614	44325	5278
# Classes	5	5	5	5	5	5	6	3	7
DSGNN-DP	**86.25**$\pm$3.32	84.27$\pm$3.12	*37.62*$\pm$0.60	49.50$\pm$0.98	*68.07*$\pm$1.54	74.58$\pm$4.56	76.57$\pm$0.85	88.38$\pm$0.41	85.87$\pm$0.65
DSGNN-GAT	85.42$\pm$3.93	83.66$\pm$2.50	37.43$\pm$0.82	47.87$\pm$1.30	63.77$\pm$1.31	72.19$\pm$3.37	76.25$\pm$0.86	88.32$\pm$0.80	85.80$\pm$0.87
$O(d)$-NSD	*85.95*$\pm$5.51	**89.41**$\pm$4.74	**37.81**$\pm$1.15	**56.34**$\pm$1.32	68.04$\pm$1.58	84.86$\pm$4.71	76.70$\pm$1.57	89.49$\pm$0.40	86.90$\pm$1.13
GGCN	84.86$\pm$4.55	86.86$\pm$3.29	37.54$\pm$1.56	*55.17*$\pm$1.58	**71.14**$\pm$1.84	**85.68**$\pm$6.63	77.11$\pm$1.45	89.15$\pm$0.37	*87.95*$\pm$1.05
H2GCN	84.86$\pm$7.23	*87.65*$\pm$4.98	35.70$\pm$1.00	36.48$\pm$1.86	60.11$\pm$2.15	82.70$\pm$5.28	77.11$\pm$1.57	89.49$\pm$0.38	87.87$\pm$1.20
GCNII	77.57$\pm$3.83	80.39$\pm$3.40	37.44$\pm$1.30	38.47$\pm$1.58	63.86$\pm$3.04	77.86$\pm$3.79	*77.33*$\pm$1.48	**90.15**$\pm$0.43	**88.37**$\pm$1.25
Geom-GCN	66.76$\pm$2.72	64.51$\pm$3.66	31.59$\pm$1.15	38.15$\pm$0.92	60.00$\pm$2.81	60.54$\pm$3.67	**78.02**$\pm$1.15	*89.95*$\pm$0.47	85.35$\pm$1.57
GCN	55.14$\pm$5.16	51.76$\pm$3.06	27.32$\pm$1.10	53.43$\pm$2.01	64.82$\pm$2.24	60.54$\pm$5.30	76.50$\pm$1.36	88.42$\pm$0.50	86.98$\pm$1.27
GAT	52.16$\pm$6.63	49.41$\pm$4.09	27.44$\pm$0.89	40.72$\pm$1.55	60.26$\pm$2.50	61.89$\pm$5.05	76.55$\pm$1.23	87.30$\pm$1.10	86.33$\pm$0.48
MLP	80.81$\pm$4.75	85.29$\pm$3.31	36.53$\pm$0.70	28.77$\pm$1.56	46.21$\pm$2.99	81.89$\pm$6.40	74.02$\pm$1.90	87.16$\pm$0.37	75.69$\pm$2.00

Following this procedure, nodes accumulate higher order neighborhood information at every sampling step. Figure 1 (left) visualizes two such sampling steps as an example for a walker (dark blue circle) starting at node v_1 at sampling step $t = 0$ and traversing two steps to node v_5. After each sampling step, a message is send to the origin node, i.e. node v_1 is updated. As shown, our model consists of two main components: A sampling model p_θ for neighbor selection, and a state model s_ϕ to integrate node features along a trajectory. We consider two different versions of the sampling model p_θ. The first one replicates the attention mechanism of the graph attention network (GAT) [11] and the second one implemented dot-product attention. Both implementations leverage parameterized 2-layer MLPs with a single non-linearity between the two layers to not limit the numerical range of the logits. In order to sample just a single edge from the attention distribution and maintain differentiability, we apply the Straight-Through Gumbel-Softmax Estimator (ST Gumbel-Softmax) [3]. To integrate node features along a sampled trajectory we simply use a simple permutation invariant model. Specifically, to update the state w_i^t of a walker i at sampling step t, we use a residual block $\mathbf{w}_i^{t+1} = s_\phi(\mathbf{w}_i^t, \mathbf{x}_j^t) = \text{MLP}_\phi(\mathbf{x}_j^t) + \mathbf{w}_i^t$. Lastly, we integrate the walker states $\mathbf{w}_i^t$ into the representation of the origin node $Pos^0(w_i)$ at every sampling step $t \in [0, T]$.

Any aggregation function used in GNNs could be used here and we opt for mean aggregation. After traversing the graph over T sampling steps, we apply a 2-layer MLP that generates the final prediction from the node encodings $n^T \in N^T$. Applying this to graph-level tasks is possible by aggregating the nodes globally prior to the final output MLP.

III. Results

We evaluate our approach on common transductive graph benchmarks [2], [7], [9], [10]. The respective graphs are diverse in size and exhibit varying homophily rates. The benchmarks are cross-validated on 10 different splits with 45%, 32%, and 20% nodes per class for training, testing and validation, respectively. These are the exact same data splits that have been used by [1], [8], [12]. For each split an individual model is trained. During training—after every epoch—models are stored. For evaluation on the test data the best performing model is restored and used (for details see [1], [8], [12]).

Table I shows the mean accuracies and standard deviations for the transductive graph benchmarks. We include multiple baselines and recent approaches that have been specifically designed for heterophilic (low homophily value) graph datasets. The models provided for comparison have been evaluated on the exact same data splits, i.e. the splits provided by [8]. The accuracy values are taken from [12], FAGCN and MixHop are taken from [6] and results of the $O(d)$-NSD model are taken from [1].

IV. Discussion, Remarks and Conclusion

We proposed a sampling-based graph neural network that learns to find salient trajectories along an underlying graph structure jointly with a given downstream task. We evaluated this approach on commonly used transductive benchmarks and found that the sampling is especially useful for heterophilic graphs.

While we introduced the general framework and presented first promising results, there are open questions for future work: First, this model can be applied to inductive tasks as well which has been confirmed by preliminary experiments in the molecular domain. This is to be investigated further in different domains as well to showcase a general effectiveness in inductive tasks. Second, we provided an exemplary implementation of the components of the general framework here. But there are many possibilities for variations and different choices which we want to investigate in the future. As one example, currently we simply use a permutation invariant state model which offers multiple routes for improvement. We consider an iterative graph encoder as a promising replacement. As one further direction, the selection process of trajectories of the walkers in itself is an interesting source of information. We are aiming to generate local explanations on this selection process comparable to [13] and analysing the sampling behavior globally could yield insight into properties of graph datasets as a whole.

References

[1] C. Bodnar, F. Di Giovanni, B. P. Chamberlain, and et al. Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs, Oct. 2022. arXiv:2202.04579 [cs, math].

[2] M. Craven, D. DiPasquo, D. Freitag, and et al. Learning to extract symbolic knowledge from the world wide web. AAAI'98/IAAI'98, page 509–516.

[3] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *ICLR 2017, Conference Track Proceedings*, 2017.

[4] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[5] Q. Li, Z. Han, and X. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *CoRR*, abs/1801.07606, 2018.

[6] V. Lingam, R. Ragesh, A. Iyer, and S. Sellamanickam. Simple truncated SVD based model for node classification on heterophilic graphs. *CoRR*, abs/2106.12807, 2021.

[7] G. M. Namata, B. London, L. Getoor, and B. Huang. Query-driven active surveying for collective classification. In *Workshop on Mining and Learning with Graphs*, 2012.

[8] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

[9] P. Sen, G. Namata, M. Bilgic, and et al. Collective classification in network data articles. *AI Magazine*, 29:93–106, 09 2008.

[10] J. Tang, J. Sun, C. Wang, and et al. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD*, KDD '09, page 807–816, New York, NY, USA, 2009.

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2017.

[12] Y. Yan, M. Hashemi, K. Swersky, and et al. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. page arXiv:2102.06462, Feb. 2021.

[13] R. Ying, D. Bourgeois, J. You, and et al. GNN explainer: A tool for post-hoc explanation of graph neural networks. *CoRR*, abs/1903.03894, 2019.