Introduction to Scalable Graph Learning Models

Zhiming Xu
zx2rw@virginia.edu

School of Engineering and Applied Science
University of Virginia

July 16, 2021

Table of Contents

Messaging-Passing Graph Neural Networks

Current Graph Neural Networks are commonly following the spectral message passing strategy

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}\right)$$

- $\hat{\boldsymbol{A}}$: normalized adjacency matrix.
- $\boldsymbol{H}$: representation (attribute) matrix.
- $\boldsymbol{W}$: trainable parameter matrix.

Problems with Message-Passing

However, once applied on massive graphs, it can cause problems

- Computational costs in matrix multiplication.
- Space costs in storing matrices and intermediate results.

Scalable Graph Learning

- *Precomputation*: ignore parameter matrix $\boldsymbol{W}$'s at first and precompute $\hat{\boldsymbol{A}}^l \boldsymbol{H}$ that passes information to l-hop neighbors.
- *Sampling*: sample subsets of nodes and edges with mini-batch training.
- *Data Transformation*: transform large graph to sequences, e.g., by random walk.

Table of Contents

Simplifying Graph Convolutional Networks [1]

- Precompute up to k-th order of $\hat{\boldsymbol{A}}$ to receive message from the k-hop neighbors.
- Propagation: $\boldsymbol{H}^{(l)} = \sigma\left(\hat{\boldsymbol{A}}^{(l)}\boldsymbol{X}\boldsymbol{W}\right)$.

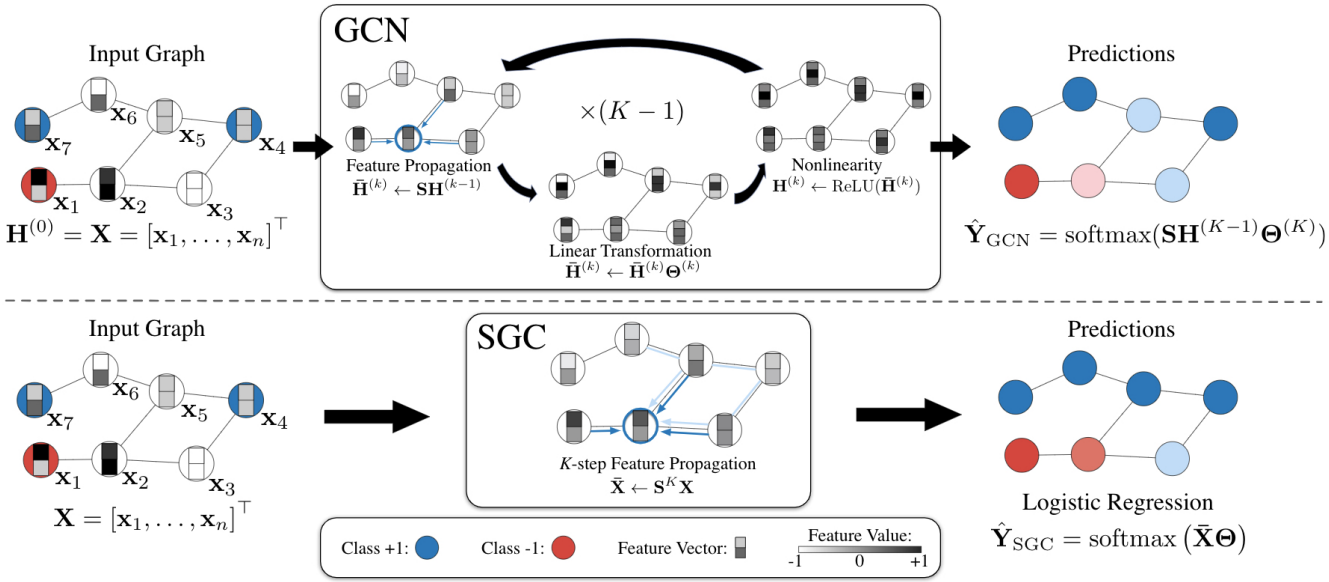

Figure 1: Comparison between GCN and SGC's message-passing.

Table of Contents

(Approximated) Personalized Page-rank of Neural Predictions [2]

- Connect GCN's limit distribution with that of page-rank.
- Separate neural networks from information propagation.
- Simplify propagation rule and reduce parameters, leading to lower computational costs.

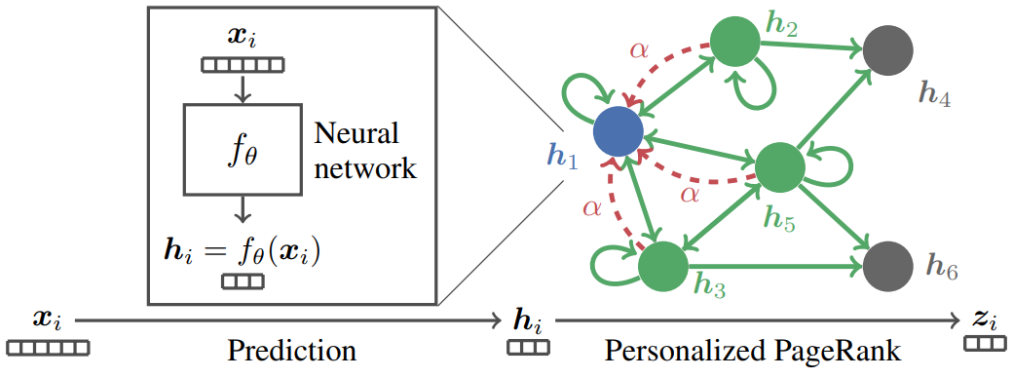

Figure 2: Neural Predictions Propagated with Page-rank.

(Approximated) Personalized Page-rank of Neural Predictions (cont.)

- *PPNP*: $\boldsymbol{Z} = \text{softmax}\left(\alpha\left(\boldsymbol{I}_n - (1-\alpha)\hat{\boldsymbol{A}}\right)^{-1}\boldsymbol{H}\right), \boldsymbol{H} = f_\theta(\boldsymbol{X}).$
- *APPNP*:

$$\begin{aligned}\boldsymbol{Z}^{(0)} &= \boldsymbol{H} = f_\theta(\boldsymbol{X}) \\ \boldsymbol{Z}^{(l+1)} &= (1-\alpha)\hat{\boldsymbol{A}}\boldsymbol{Z}^{(l)} + \alpha\boldsymbol{H} \\ \boldsymbol{Z}^{(L)} &= \text{softmax}\left((1-\alpha)\hat{\boldsymbol{A}}\boldsymbol{Z}^{(L-1)} + \alpha\boldsymbol{H}\right)\end{aligned}$$

Problem with Precomputation

- Reduce the model's representation ability and lower its performance.
- Still involve expensive multiplications between huge matrices.

Table of Contents

Graph Sample and Aggregate [3]

Sample neighbors and aggregate their information.

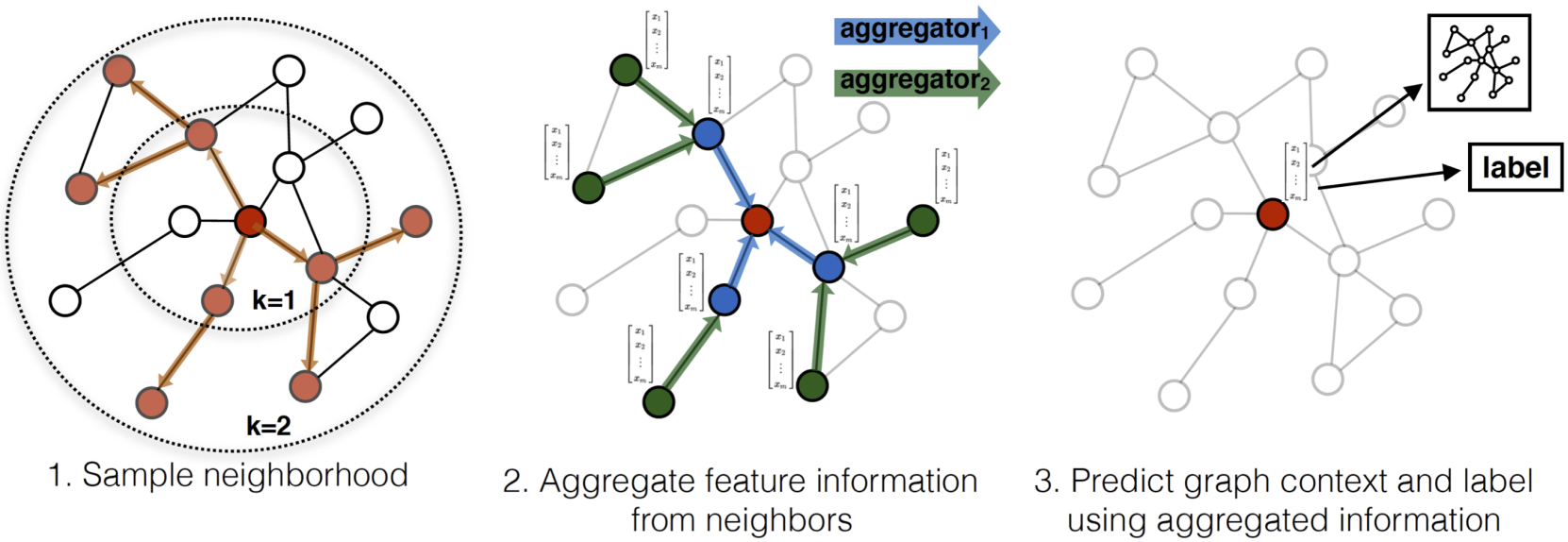

Figure 3: Illustration of GraphSAGE forward propagation.

GraphSAGE (cont.)

Forward Propagation

Result: Node i's representation z_i after K iterations

$\vec{h}_i^0 \leftarrow \vec{h}_i, \forall i \in \mathcal{V}$;

for $k = 1 \ldots K$ **do**

  for $i \in \mathcal{V}$ **do**

    $\vec{h}_{N_i}^k \leftarrow \text{AGGREGATE}_k \left(\{\vec{h}_j, \forall j \in \mathcal{N}_i\} \right)$;

    $\vec{h}_i^k \leftarrow \sigma \left(W^k \cdot \left[\vec{h}_i^{k-1} : \vec{h}_{\mathcal{N}_i}^k \right] \right)$;

  end

  $\vec{h}_i^k \leftarrow \frac{\vec{h}_i^k}{\|\vec{h}_i^k\|_2}, \forall i \in \mathcal{V}$;

end

$\vec{z}_i \leftarrow \vec{h}_i^K, \forall i \in \mathcal{V}$

GraphSAGE (cont.)

Graph-Based Loss Function

$$L_G(\vec{h}_i) = -\log\left(\sigma\left(\vec{h}_i^\top \vec{h}_j\right)\right) - Q \cdot \left(\mathbb{E}_{v_i \sim P_{n(i)}} \log\left(\sigma\left(-\vec{h}_i^\top \vec{h}_{v_i}\right)\right)\right)$$

- j is a node that co-occurs near i on fixed-length random walk.
- σ is the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$
- P_n is a negative sampling distribution, Q is # of negative samples.

Based on loss L_G, the parameters in Algorithm 1 are optimized with stochastic gradient descend.

PinSAGE [4]

Sample neighbors more wisely with the help of random walk.

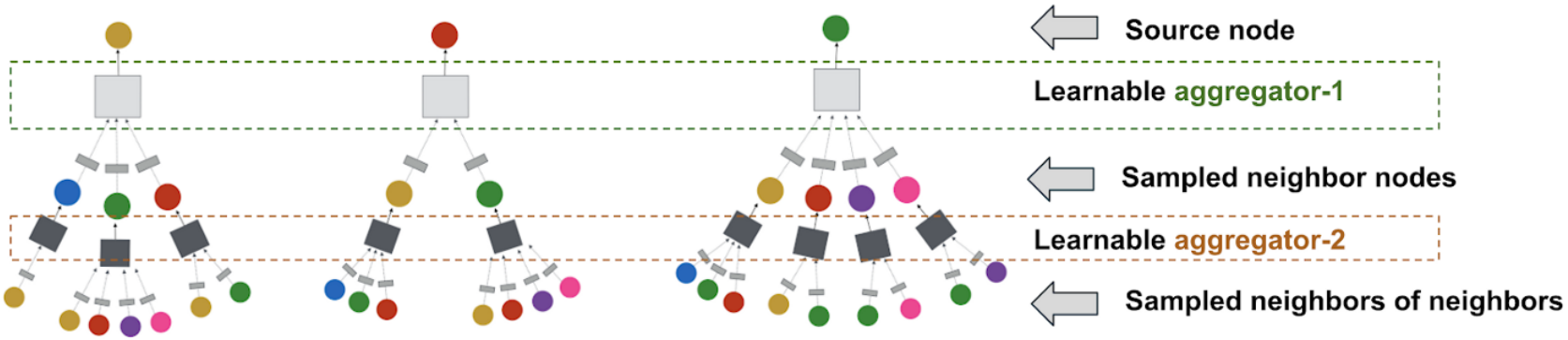

Figure 4: Illustration of PinSAGE forward propagation.

Key Novelty over GraphSAGE

- *Construct Neighborhood graph via Random Walk*. PinSAGE uses random walk to select neighbors with highest visit counts, which takes the importance of different neighbors into consideration and can control how many nodes to sample.
- *MapReduce computation to avoid overhead*. The *bottom-up* computation in Figure 4 fits in MapReduce. First *map* each node to the latent space, then *join* them to the upper-level nodes, and finally *reduce* to perform the aggregation.

Problem with Node-wise Sampling

The size of neighborhood increases exponentially with more layers. Therefore, the computational and storage costs are more and more expensive if the model goes deeper.

Table of Contents

FastGCN [5]

Key Assumptions

- The whole large graph is possibly infinite.
- Each sampled subgraph's nodes are i.i.d.
- The convolution and loss function is seen as integration and expectation.

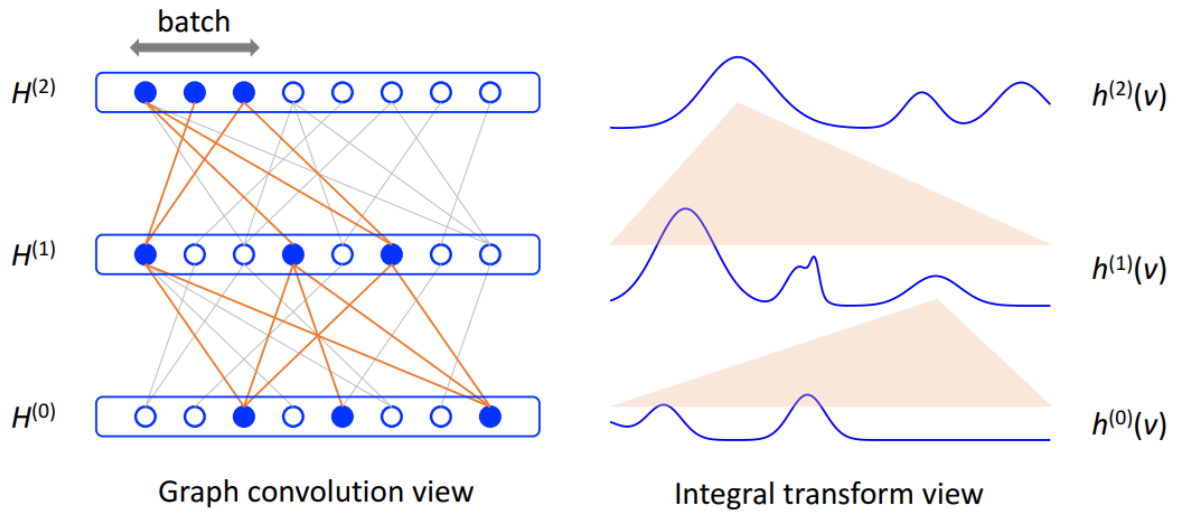

Figure 5: Comparison between mini-batch GCN and FastGCN.

FastGCN (cont.)

- Message-passing and loss as expectation.

$$\boldsymbol{h}^{(l+1)}(v) = \sigma\left(\mathbb{E}_p\left[\hat{\boldsymbol{A}}(v,u)\boldsymbol{h}^{(l)}(u)\boldsymbol{W}^{(l)}\right]\right)$$

$$\mathcal{L} = \mathbb{E}_p\left[\mathcal{L}(\boldsymbol{h}^{(M)}(v))\right]$$

- Expectation Approximation.

$$\boldsymbol{H}^{(l+1)}(v,:) = \sigma\left(\frac{N}{n_l}\sum_{j=1}^{n_l}\hat{\boldsymbol{A}}(v,u_j^{(l)})\boldsymbol{H}^{(l)}(u_j^{(l)},:)\boldsymbol{W}^{(l)}\right)$$

$$\mathcal{L} = \frac{1}{n_M}\sum_{i=1}^{n_M}\mathcal{L}(\boldsymbol{H}^{(M)}(u_i^{(M)},:))$$

n_l is # of samples in the l-th layer, N is # of all samples.

FastGCN (cont.)

- Unbiased Approximation via Importance Sampling.[1]

$$\boldsymbol{h}^{(l+1)}(v) = \sigma\left(\mathbb{E}_q\left[\hat{\boldsymbol{A}}(v,u)\boldsymbol{h}^{(l)}(u)\boldsymbol{W}^{(l)} \cdot \frac{p(u)}{q(u)}\right]\right)$$

Analytically, $q^* \propto \sqrt{\mathbb{E}_v\left[\hat{\boldsymbol{A}}(v,u)^2\right]} \cdot \left|\boldsymbol{h}^{(l)}(u)\boldsymbol{W}^{(l)}\right| \cdot p(u)$ when the deviation of approximation is minimized.

However, to reduce computational costs and ensure stability, it is approximated by $\hat{q} \propto \sqrt{\mathbb{E}_v\left[\hat{\boldsymbol{A}}(v,u)^2\right]} = \left\|\hat{\boldsymbol{A}}(:,u)\right\|^2$.

[1] If r.v. $\bar{X}$ is used to estimate X, then $\bar{X}$ is unbiased if $\mathbb{E}[\bar{X}] = X$. The deviation of such estimation is $\mathbb{E}[\bar{X}^2] - \mathbb{E}^2[\bar{X}]$.

Problems with Layer-wise Sampling

- To ensure unbiased approximation, the sampling algorithm itself is expensive.
- If the i.i.d assumption does not hold, e.g., the graph is large and sparse, the model might be unable to learn useful representations.

Table of Contents

GraphSAINT [6]

Key Novelty

- Sample subgraphs before training, then draw different subgraphs to run a full GCN on them and aggregate information.
- Address the aggregation bias incurred by subgraph sampling.

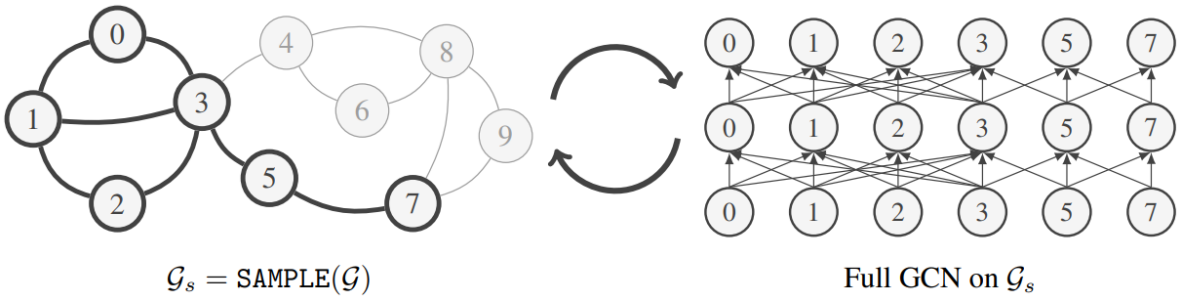

Figure 6: Illustration of GraphSAINT's "batch" and "unbatch".

GraphSAINT (cont.)

For a subgraph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, the aggregation process of node $v \in \mathcal{G}_s$ is

$$\boldsymbol{H}_v^{(l+1)} = \sum_{u \in \mathcal{V}} \frac{\hat{\boldsymbol{A}}_{v,u}}{\alpha_{u,v}} (\boldsymbol{W}^{(l)})^\top \boldsymbol{H}_u^{(l)} \mathbb{I}_{u|v}$$

- $\alpha_{u,v}$ is *aggregator normalization*.
- $\hat{\boldsymbol{A}}$ is the normalized adjacency matrix.
- $\mathbb{I}_{u|v} = \begin{cases} 0, & \text{if } u \in \mathcal{V}_s \wedge (u,v) \notin \mathcal{E}_s \\ 1, & \text{if } (u,v) \in \mathcal{E}_s \\ \text{undef.}, & \text{if} v \notin \mathcal{V}_s \end{cases}$

Debias and Minimize Deviation

To debias and minimize deviation in the approximation of $\boldsymbol{H}_v^{(l+1)}$, it is proved that

- When $\alpha_{u,v} = p_{u,v}/p_v$, then $\boldsymbol{H}_v^{(l+1)}$ is the unbiased approximation of node v's aggregation.
- When $|\mathcal{E}_s| = m$, $p_{u,v} = \dfrac{m}{\sum_{e'} \left\| \sum_l b_{e'}^{(l)} \right\|} \left\| \sum_l b_e^{(l)} \right\|$, the deviation induced from approximation is minimized.

$$b_e^{(l)} = \hat{\boldsymbol{A}}_{v,u} \boldsymbol{H}_u^{(l)} + \hat{\boldsymbol{A}}_{u,v} \boldsymbol{H}_v^{(l)} \approx \hat{\boldsymbol{A}}_{v,u} + \hat{\boldsymbol{A}}_{u,v}.$$

Problems with Graph-wise Sampling

The nodes in all subgraphs are connected, leading to intrinsic bias within each mini-batch.

Table of Contents

Node2Vec [7]

- Generate paths via random walks over the entire graph to obtain "corpus".
- Apply word2vec algorithm on such corpus to get node representations.

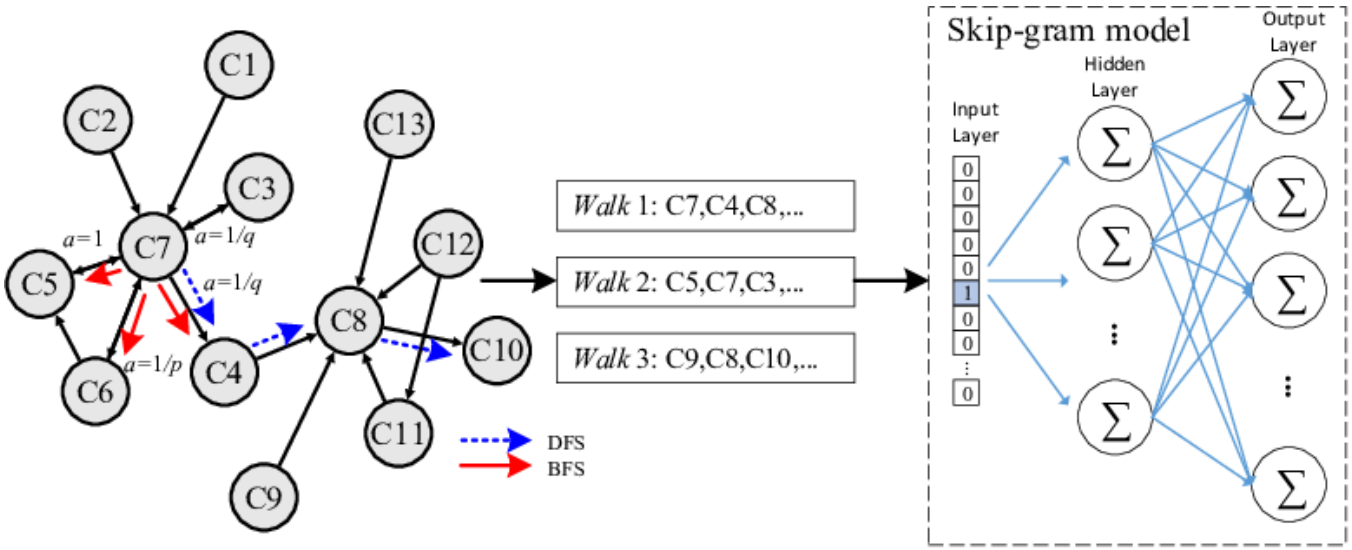

Figure 7: Illustration of node2vec.

Problems with Data Transformation

Graph structure is not well preserved and collapses to a collection of paths and useful information can be lost when generating "corpus". Therefore, it can prevent the model from learning expressive representations and hinder its performance.

Thank You for Your Attention

Q & A

References

[1] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 6861–6871. [Online]. Available: http://proceedings.mlr.press/v97/wu19e.html.

[2] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=H1gL-2A9Ym.

References (cont.)

[3] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 1024–1034. [Online]. Available: http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.

[4] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Y. Guo and F. Farooq, Eds., ACM, 2018, pp. 974–983. DOI: 10.1145/3219819.3219890. [Online]. Available: https://doi.org/10.1145/3219819.3219890.

References (cont.)

[5] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=rytstxWAW.

[6] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. K. Prasanna, "Graphsaint: Graph sampling based inductive learning method," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=BJe8pkHFwS.

References (cont.)

[7] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds., ACM, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754. [Online]. Available: https://doi.org/10.1145/2939672.2939754.