Applying GraphSAGE to Large-Scale Node Classification

Programming Machine Learning Models for HPC
Final Report

Thomas Gantz Alberto Finardi Tommaso Crippa Jan Marxen

December 2025

Abstract

This report documents our study and application of GraphSAGE (Graph SAmple and aggreGatE) [2], an inductive framework for learning node embeddings on large-scale graphs. We provide background on graph neural networks and the message passing paradigm, then describe how GraphSAGE addresses the scalability limitations of earlier approaches by learning aggregator functions rather than fixed node embeddings. We apply GraphSAGE to the ogbn-products benchmark dataset for node classification, detailing our implementation choices, hyperparameter tuning process, and experimental results.

Contents

1 Introduction

Graph-structured data appears in many domains, including social networks, citation graphs, biological networks, and recommendation systems. A central challenge is learning node representations (embeddings) that capture both node features and structural context.

Graph Prediction Tasks. Machine learning on graphs encompasses node-level prediction (classifying individual nodes), edge-level prediction (link prediction or edge classification), and graph-level prediction (predicting properties of entire graphs). This report focuses on **node classification**, the task addressed by GraphSAGE.

Graph Neural Networks. A Graph Neural Network (GNN) is a learnable transformation that updates node features using neural networks, respects graph structure by aggregating neighbor information, and is permutation invariant. GNNs build node representations through iterative **message passing**: at each layer, nodes gather neighbor embeddings, aggregate them with a permutation-invariant function, and apply a learned transformation.

Limitations of Prior Methods. Before GraphSAGE, methods like DeepWalk and node2vec [4, 1] learned embeddings via random walks but were *transductive*—adding new nodes required retraining. GCNs [3] use message passing but operate on the full adjacency matrix, limiting scalability. The fundamental problem: **no method provided a parametric function to generate embeddings for unseen nodes.**

The GraphSAGE Insight. GraphSAGE's innovation is to learn a **function** that generates embeddings rather than learning fixed embeddings per node. By learning aggregator functions that sample and combine neighborhood features, GraphSAGE becomes **inductive**: the model generalizes to new nodes or graphs without retraining.

2 GraphSAGE Framework

GraphSAGE (Graph SAmple and aggreGatE) implements inductive learning through neighbor sampling and learned aggregation. At each layer k, node v updates its representation by sampling neighbors, aggregating their embeddings, concatenating with its own embedding, and applying a learned transformation with non-linearity.

2.1 Neighbor Sampling

For scalability, GraphSAGE uniformly samples a fixed-size set of neighbors at each layer rather than using all neighbors. For a K-layer model with sample sizes $S_1, \ldots, S_K$, per-node complexity is $O(\prod_{k=1}^{K} S_k)$, independent of actual node degrees.

2.2 Aggregator Functions

GraphSAGE supports several aggregators: **mean** (element-wise average), **LSTM** (sequential processing of a random neighbor permutation), and **pooling** (neural network followed by max-pooling). The mean aggregator is simplest and often effective; pooling provides more expressiveness.

3 Implementation

Implementation details will be provided here, including our PyTorch/DGL code structure, data loading pipeline, and distributed training setup.

4 Hyperparameters

Hyperparameter tuning methodology and final configurations will be documented here, including learning rate schedules, hidden dimensions, number of layers, sample sizes, dropout rates, and batch sizes.

5 Results

Experimental results on the ogbn-products dataset will be presented here, including accuracy metrics, training curves, and comparisons with baseline methods.

6 Conclusion

Summary of findings and conclusions will be provided here.

References

[1] Aditya Grover and Jure Leskovec. "node2vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 855–864.

[2] William L. Hamilton, Rex Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, pp. 1024–1034. URL: `https://arxiv.org/abs/1706.02216`.

[3] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[4] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014, pp. 701–710.