AGS-GNN: Attribute Guided Sampling for Graph Neural Network

Siddhartha Shankar Das‡, SM Ferdous†, Mahantesh Halappanavar†, Edoardo Serra*, Alex Pothen‡
Purdue University‡, Pacific Northwest National Laboratory†, Boise State University*
Contact: das90@purdue.edu

CLSAC Chesapeake Large-Scale Analytics Conference

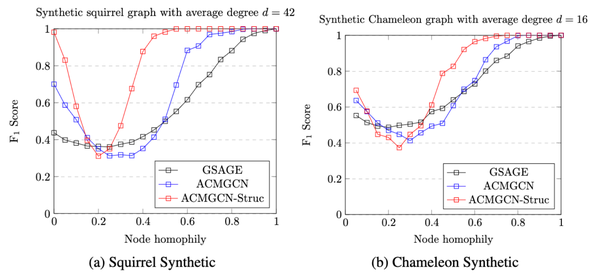
Abstract

We introduce an Attribute-Guided Sampling (AGS) framework for Graph Neural Network (GNN), which addresses the scaling challenges of graph representation learning under varying levels of homophily and heterophily. The framework leverages unsupervised and supervised sampling strategies, optimizing the subgraph selection based on features and structure. AGS utilizes the nearest neighbor to encourage homogeneity and submodular optimization to ensure diversity in the neighborhood of a selected sparse subgraph. AGS-GNN is the only inductive scalable GNN for large heterophilic graphs in the literature. Experimental results on wide ranges of benchmark datasets demonstrate the effectiveness of AGS-GNN, yielding improved performance and faster convergence compared to traditional methods and even outperforming baselines and state-of-the-art approaches in various scenarios.

Preliminaries

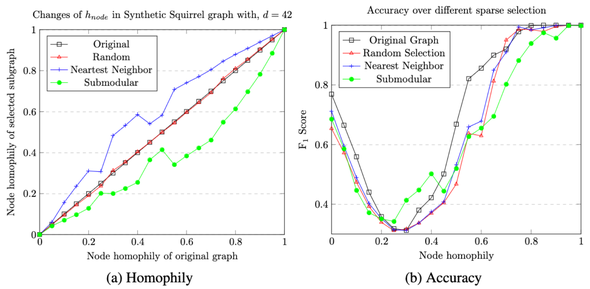
Assumption: The nodes with similar features tend to have the same label.

Node Homophily: $h_{node} = \frac{1}{|V|}\sum_{u\in V}\frac{(u,v):v\in N(u)\wedge Y_u=Y_v}{N(u)}$. Here, $N(u)$ represents the neighbors of node u. A graph with a low homophily score is termed *heterophilic*;



(a) Squirrel Synthetic (b) Chameleon Synthetic

Fig. 1 shows F_1 scores of a synthetic graphs in different ranges of homophily using homophilic and heterophilic GNNs

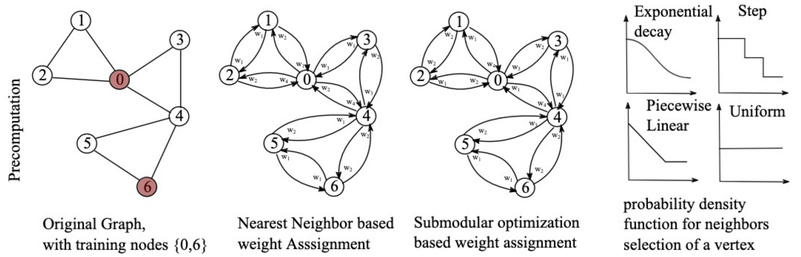


(a) Homophily (b) Accuracy

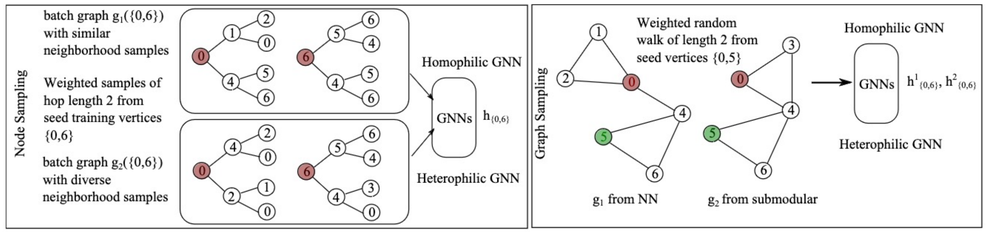
Fig. 2 shows how *k*-nearest neighbor and *k*-neighbor selection from submodular optimization-based ranks change homophily relative to random *k*-neighbor subgraph selection.

Proposed Method: AGS-GNN

Fig. 3 Outline of AGS-GNN



(a) Precomputation



(b) Node Sampling and Graph Sampling

Pre-computation

Similarity metric Learning: Without an appropriate similarity metric between node features, we can learn a straightforward edge weight prediction model from the known labeled data and given graph edges and non-edges.

K-NN weight assignment: *k*-nearest neighbor-based rankings are used to fit specific probability density function.

Submodular optimization weight assignment: Submodular functions have diminishing returns property. We rank the neighbors of a vertex based on the gains. We get diverse neighborhoods for each vertex depending on the node features and appropriate functions. . Example functions are coverage-based, feature-based, and facility location.

Node Sampling

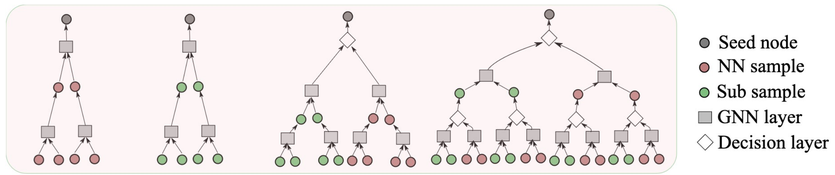


Fig. 4 different ways weighted sampling can be used in conjunction to the GNNs

Algorithm 1: AGS-GNN-NS(G,X)

Input : Graph $G(V, E)$, Feature matrix $X \in R^{N\times F}$
Parameter: Heuristic or learnable similarity measure

R_1 = Nearest-Neighbor(G, X)
R_2 = Submodular(G, X)
for *epoch* in *epochs* **do**
  $nodes$ = BatchOfTrainingNodes(G)
  /* random nodes from training vertices */
  g_1 = NodeSample($R_1, nodes, neighbors$)
  /* for homophily */
  g_2 = NodeSample($R_2, nodes, neighbors$)
  /* for heterophily */
  $output = model_\theta(g_1, g_2)$
  compute loss for $nodes$ and update θ
end for

For homophilic graphs, only nearest neighbor (NN) samples (Fig. 4a) can be good enough to get high prediction accuracy. For certain very large dense homophilic graphs, only diverse samples from submodular (Sub) (Fig. 4b) can be beneficial as it selects better representative samples. For heterophilic graphs, nodes can be either homophilic or heterophilic and thus it's better to consider both samples and left the decision to a MLP (Fig. 4c, 4d) to adaptively learn.

Graph Sampling

We can incorporate heuristics in the sampling process using edge-disjoint subgraphs. Consider the heuristic of ensuring connectivity among sampled nodes. Compute k-edge-disjoint Maximum Spanning Tree (MSTs) for a graph and combine them to get a sparse representation of the original graph. Sample a few and merge them to obtain the sparse graph and sample from the left-over graph using a weighted random walk to ensure non-zero probability for each edge. Subgraph selection examples choice include k-NN, b-matching, and Spectral Sparsifier.

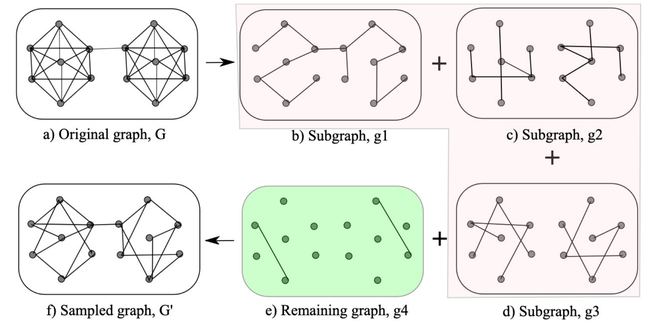


Fig. 5 Graph G is split into a collection of disjoint edges (spanning tree is used), and then we sample such subgraphs to get a sparse graph, G′ = g1 + g2 for an epoch.

Results

Heterophilic Graphs	GSAGE		GSAINT		LINKX†		ACMGCN		AGS-NS		AGS-GS	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Cornell	71.35	6.07	67.03	3.15	**76.76**	**4.32**	74.59	1.32	74.59	2.16	70.27	4.83
Texas	77.30	5.57	79.46	6.53	81.62	2.02	84.32	7.13	**84.86**	**6.19**	80.00	5.57
Wisconsin	79.61	3.64	79.61	6.86	83.53	4.74	84.31	3.28	81.96	4.71	**85.10**	**6.49**
reed98	61.87	0.53	64.15	0.69	66.63	1.37	66.11	1.25	**66.74**	**1.37**	64.66	0.89
amherst41	66.62	0.33	69.57	0.71	78.64	0.35	78.12	0.30	**79.19**	**0.47**	77.14	0.90
penn94	75.65	0.42	75.11	0.33	**85.92**	**0.32**	85.38	0.53	76.06	0.41	81.56	0.45
Roman-empire	79.52	0.42	77.51	0.47	59.14	0.45	71.42	0.39	**80.49**	**0.48**	75.38	0.25
cornell5	69.22	0.12	68.10	0.15	80.10	0.27	78.43	0.50	**82.84**	**0.01**	74.84	0.35
Squirrel	38.66	1.24	39.14	1.45	35.91	1.09	**72.06**	**2.21**	68.24	0.97	51.73	1.30
johnshopkins55	67.37	0.54	67.43	0.20	**79.63**	**0.16**	77.37	0.61	78.13	0.42	75.93	0.48
Actor	34.82	0.55	35.24	0.81	33.93	0.82	34.42	1.08	**36.55**	**0.93**	34.88	0.63
Minesweeper	**85.74**	**0.25**	85.46	0.49	80.02	0.03	80.33	0.23	85.56	0.28	85.25	0.71
Questions	97.13	0.01	97.18	0.04	97.06	0.03	97.02	0.00	**97.27**	**0.04**	97.23	0.04
Chameleon	51.18	2.70	52.32	2.47	50.18	2.01	**75.81**	**1.67**	73.46	2.29	66.67	1.65
Tolokers	79.15	0.32	78.89	0.37	80.07	0.53	80.45	0.54	**80.52**	**0.41**	80.50	0.61
Amazon-ratings	48.08	0.38	52.21	0.27	52.68	0.26	52.94	0.23	**53.21**	**0.46**	52.25	0.34

Table 1 shows the F1 measure of the node classification task for heterophilic graphs. Here, AGS-GNN is compared against homophilic (GraphSAGE, GraphSaint) and heterophilic (LINKX, ACM-GCN) works.

Acknowledgements: We used the institutional computing resources at the PNNL and Purdue University. We highlight only the core part of the work; for more details, contact the authors.