Graph Neural Networks with Extreme Nodes Discrimination

George Panagopoulos[1] and Hamid Jalalzai[2]

[1] Ecole Polytechnique, Route de Saclay, 91120 Palaiseau, France
[2] Telecom Paris. 19 Place Marguerite Perey, Palaiseau, France
george.panagopoulos@polytechnique.edu
hamid.jalalzai@telecom-paris.com

Abstract. Graph neural networks (GNNs) are a successful example of leveraging the underlying structure between samples to perform efficient semi-supervised learning. Though the spatial correlation of the nodes is inherently taken into account by the models' architecture, structural correlations and their effects in learning remain a relatively overlooked topic. In this work, we propose a new approach to train a GNN, by separating the samples based on their structural importance, meaning discriminating for samples that belong in a higher tier in terms of a network centrality metric. Our proposed method is supported by recent theoretical findings based on Extreme Value Theory, that buttress the separation of extreme and regular samples in binary classification. Essentially we split a GNN into two parts, each trained and validated separately using extreme and regular nodes from the observed set. We perform experiments in the three most prevalent GNN models, using three well-known benchmark datasets and compare the predictions of the model with and without sample discrimination. The classification of extreme nodes is clearly benefited, validating the relevant theory. In contrast, the regular nodes are undermined, despite their significantly larger train set. Exploratory findings suggest the limited structure contained in regular samples to be a potential reason for this.

1 Introduction

As the success of machine learning skyrocketed the past decade in the academic community, an increasing number of real-world problems have been undergoing machine learning-based solutions. One major setback in this natural turn of events is the limited number of labels accompanying real-world datasets or the total lack thereof. The arduous procedure required to label a massive dataset motivated a shift of attention towards effective semi-supervised learning approaches. Moreover, given the abundance of relational data, from chemical [7] to social networks [20], and from drug-discovery [3] to fake news detection [15], semi-supervised learning based on the graph of the input samples has been exceedingly popular [26], whether the underlying graph is already defined [25], or is implicitly inferred [18].

In this work, we focus on Graph neural networks (GNNs), which have exhibited impressive results in transductive semi-supervised classification tasks using a minuscule supervision [9]. Their impressive accuracy stems from leveraging the correlation of the input samples in the underlying graph. Numerous neural architectures have been recently proposed to combine the structure of the graph and the attributes of the nodes in an effective end-to-end manner [23]. That being said, the majority of the models focus on strategies of aggregation from multiple nodes, overlooking the structural role of each specific node and its effect on the learning task. Although node centrality is taken into account inherently in message passing, its effect is unclear, since the aggregation step does not preserve the scale of the representations' combination [24]. On the one hand, this serves the learning tasks, which rely on positional information i.e. a paper's field or a products' category are similar to their neighbors. On the other, highly connected nodes in realistic networks exhibit different characteristics than regular ones, and may potentially correlate with the function classes. For example, classifying famous people in twitter based on their features and their followers may entail a different learning procedure than classifying less famous users, for the most classification tasks imaginable.

From a theoretical perspective, extreme value theory argues that the distribution of the extreme samples may differ from the bulk of the data and their dependency structure may be different, therefore extremely rare samples, *i.e.* samples located far from the bulk, should undergo learning separated from the rest. The main assumption being that a real valued random variable X with distribution F is regularly varying *i.e.* if there exists $\rho > 0$ such that $\forall x > 1$,

$$\frac{1 - F(\lambda x)}{1 - F(x)} \overset{x \to \infty}{\to} \lambda^{\rho}$$

where the event $X > x$ is less likely as x gets large and ρ is known as the *index of regular variation*. If instead of the sample's norm, we consider the degree as the "measure of rarity", EVT can be applied to samples that are connected with an underlying graph *i.e.* input nodes to a graph learning problem. The heavy-tail assumption of the degree matches the main property of the scale-free networks, which are ubiquitous for real, human-made graphs [1,19], including all the GNN node classification benchmarks. Figures (1a, 1b) and (1c) illustrate the distribution of the degree for the three common graph benchmark datasets. Each distribution follows a heavy-tail. The vertical axis is on a log scale so that the nodes with extreme degrees are visible. One can see that the distributional tail may vary from a figure to another, to this extent one way to compare the degree distribution of a graph to another would be to standardize the distribution of degrees. one common standardization being the Pareto standardization: $1/\big(1-F(x)\big)$ where F is the distribution of degrees. If one considers n nodes u_i with corresponding degree e_i, F can be approximated by its empirical counterpart $\widehat{F}(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{e_i \leq t\}$, where $\mathbb{1}\{\mathcal{E}\}$ corresponds to the indicator function of the event $\mathcal{E}$. Standardization and distributional comparison of degrees for different graphs will be subject of future work.

Though there are architectures that capitalize on the node's centrality by analogous attention weights ([22]), assuming different learning for each possible level of degree increases severely the danger of overfitting. Moreover, our approach is not a new model, but rather a different type of training that can be coupled with any GNN architecture. To this end, we propose to split a GNN model into two submodels, each being trained, validated, and tested using respectively the "extreme" and "regular" nodes of the initial train, validation and test set. The extreme and regular nodes are defined based on the degree distribution and a threshold, which is treated as a hyperparameter and is optimized using the validation set. We demonstrate the effectiveness of this approach using the three most well-known GNN architectures (GAT,GCN and GraphSAGE), and three prevalent benchmark datasets (**CORA**,**PubMed**,**Amason-Photo**). For each model, we use as a baseline the respective GNN model with double the number of hidden parameters. Some findings of our experiments include:

- A sufficient increase of accuracy for the extreme nodes, throughout most of the models and datasets.
- An exploration of the relationship between the number of samples and the samples' edges, that potentially explains why datasets with few extreme samples but rich structure achieve better results then ones with more extreme samples but lower degree.

The code to reproduce our analysis can be found online [3].

2 Background and Related work

2.1 Extreme Value Theory

Extreme value theory is the branch of statistics which focuses on the deviations from the median (or any other centrality measure) of probability distributions. Models based on extremes tend to learn the unusual rather than the usual. The field of application of these models range from risk management like finance, insurance, telecommunication or environmental science to teletraffic data and large graph analysis. EVT provides insights on rare events.

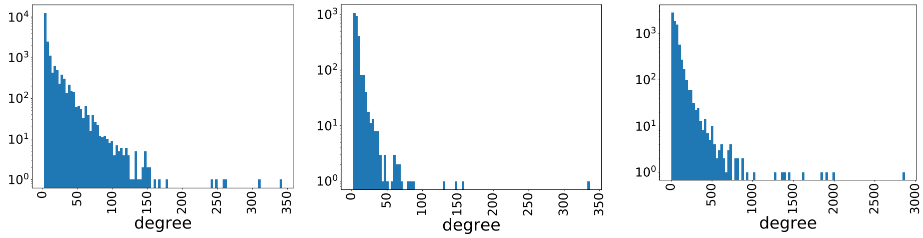
In the univariate setting, the empirical quantity corresponding to $(1-p)^{th}$ quantile of F, the distribution of a random variable X, for a given probability p of exceedance, is $\hat{x}_{p,n} = \inf\{x \in \mathbb{R}, \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{X_i \geq x\} \leq p\}$ for large up to moderate values of p. Nonetheless, as p gets small the finite dataset $\mathcal{D}_n = \{X_i\}_{i=1}^{n}$ is not guaranteed to provide valid and non degenerate solution $\hat{x}_{p,n}$, unless one relies on EVT to estimate large quantiles of a distribution. In this way, EVT boils down to studying the distribution of maxima as a Generalized Extreme Value (GEV) distribution, that is to say an element of the Gumbel, Fréchet or Weibull parametric families. The main assumption is the existence of two sequences $\{a_n, n \geq 1\}$ with $a_n > 0$, $\{b_n, n \geq 1\}$ and a non-degenerate cumulative distribution function G such that

$$\lim_{n\to\infty} n\mathbb{P}\left(\frac{X - b_n}{a_n} \geq x\right) = -\log G(x) \tag{1}$$

[3] https://github.com/GiorgosPanagopoulos/Extreme-GNNs

where x is any continuity point of the domain of G.

In the case where assumption (1) is fulfilled, F is said to be *in the domain of attraction* of G. The tail behavior of F boils down to the distribution of G. Up to rescalling, $G(x) = \exp\big(-(1+\gamma x)^{-\frac{1}{\gamma}}\big)$ where $1+\gamma x > 0$, $\gamma \in \mathbb{R}$. By convention, $(1+\gamma x)^{-\frac{1}{\gamma x}} = \exp(-x)$ for $\gamma = 0$. The shape of the tail is controled by the sign of γ.



(a) Degree distribution in Pubmed graph

(b) Degree distribution in Cora graph

(c) Degree distribution in Amazon graph

Fig. 1: Degree distribution in the examined datasets.

2.2 Graph Neural Networks

Graph Neural Networks (GNN) are a neural architecture that utilizes the underlying relations between the input samples of the data, which form a graph, to perform semi-supervised learning with minuscule training labels. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with a set of nodes $\mathcal{V}$ with $|\mathcal{V}| = n \geq 2$ and edges $\mathcal{E}$ with $|\mathcal{E}| = m \leq n(n-1)$ describing the set of edges. An edge between node u and v are depicted as $e_{u,v}$. Our setting is binary node classification, in which a label $Y_u \in \{-1, 1\}$ is associated to each node u, who is associated with a feature vector $X_u \in \mathbb{R}^{1\times d}$ where d is the number of features of each node, forming a feature matrix $X \in \mathbb{R}^{n\times d}$ and a label vector $Y \in \mathbb{R}^{n\times 1}$. The graph can be described in the form of its adjacency matrix $A \in \mathbb{R}^{n\times n}$ and normalized adjacency $\hat{A} = D^{-1/2}(A - I_n)D^{-1/2}$ where I_n is the identity matrix and D is the degree matrix of the graph [4]. Overall, a GNN architecture consists of an initial aggregation step, where the features of a node's neighbors are aggregated and combined with a weighted non-linear formula. In one of the earliest GNNs, Graph Convolutional Network [12], the aggregation was performed using the adjacency of the graph $H_1 = ReLU(\hat{A}XW_0))$. Here $W_0 \in \mathbb{R}^{fxd}$ is a matrix of learnable parameters. This combination is one hidden layer of the neural network. $H_1 \in \mathbb{R}^{n\times d}$ can then be passed into another hidden layer $\hat{A}H_1W_1$, combining information from two-hop neighbors of the node using a new set of parameters W_1. Once the layers have achieved the desired depth, the representation of the final hidden layer can be inserted into an output layer such as a softmax function to derive a node's probability to belong in a certain category. This message passing framework is more clear in GraphSAGE [8], where the hidden representation of each node u is given by a simple aggregation of its neighbors' $N(u)$ representations, such as the mean. Similar to GCN, GraphSAGE can built deeper models with more parameters. The final model we will use in our experiments is the Graph Attention Network [21], where the aggregation step is similar to GraphSAGE, but apart from the parameters of the hidden layer, the aggregation of the neighbors are weighted by an attention parameter α, different for each edge, learnt through multiple epochs.

These are the most well-known GNN architectures, all of whom serve as obligatory benchmarks of comparison for a newly proposed method. That being said, numerous architectures have been proposed in recent years that excel in node classification. Specifically jumping knowledge networks [25] expands the message passing by including nodes that may not be in structural vicinity of the node but whose representations affect the node in terms of statistical influence. A GNN with ARMA layers [2] passes the input features through an ARMA filter of a certain depth, simultaneously with the graph convolution, in order come up with more robust representations. PPNP [13] uses a personalized page rank to identify nodes that should effect a node during the message passing, and is trained end to end by parameterizing the power iteration of pagerank.

3 Methodology

Overall, our methodology follows a specific framework, depicted in Figure (2). Initially, if the dataset contains predefined train-validation-test splits i.e. the **CORA** dataset, we use them. If not, we split the dataset in half at random, resembling an observed and a test set, similar to the literature [17]. The visible set is broken into the train set comprising of 20 nodes per class as is common for the datasets we employ, and the rest of the samples are kept for the validation set. The nodes in the train, validation, and test set are separated to extreme and regular, based on their degree and a hyperparameter p that defines the threshold above which a node is considered an extreme sample. These serve as input to three different instances of the same GNN model. The *Extreme-GNN* (*EGNN*) instance is trained and validated in the extreme nodes of the observed set while the *Regular-GNN* (*RGNN*) in the respective regular nodes, as described in figure (2).

The baseline GNN consists of a hidden layer that is *double* the size of *EGNN*'s and *RGNN*'s, hence the number of parameters in the baseline GNN and in its Extreme-Regular version are equal. The baseline is trained and validated in the observed part of the network, in the same manner as in the literature.

Independent to the GNN employed, we can formally define this split as two separate risk minimization tasks, run in parallel. In the general setting, (u, Y) is a random pair with unknown joint distribution where u is a random node in the graph with corresponding label $Y \in \{-1, 1\}$. The goal is to obtain a classifier $g \to \{-1, 1\}$ which minimizes the classification risk $R(g) \overset{\text{def}}{=} \min \mathbb{P}\left(g(u) \neq Y\right)$.

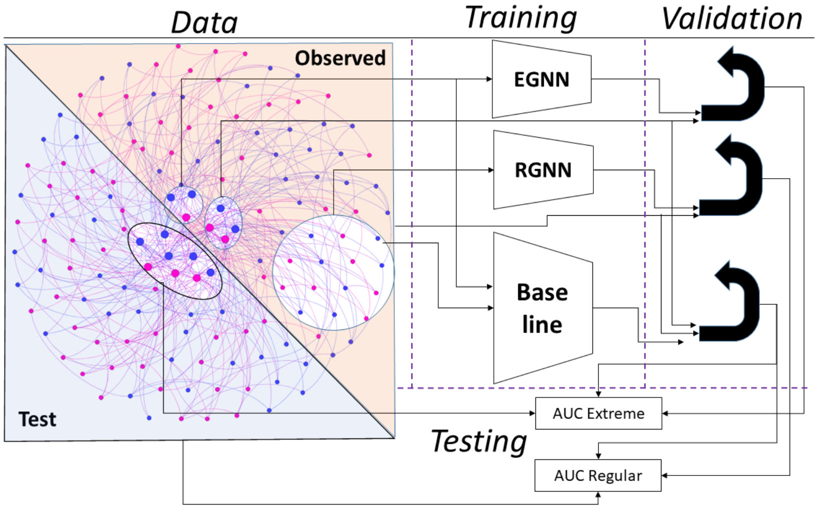


Fig. 2: The scheme of the methodology. The network as binary-class labels (color of the node) and is split in observed (red background) and test set (blue background) and in extreme-regular (big-small node). Few of the extreme nodes in the observed set are passed as training input to EGNN, the rest of the extremes are used for validation. The same happens with the regular nodes for the RGNN. The BaseLine receives the combined train set of EGNN and RGNN, and is validated with the rest of the observed set. The AUC of EGNN is computed over the extreme nodes of the Test set and the AUC of RGNN respectively with the regular nodes. The baseline model is evaluated in both.

Using the law of total probability, the classification risk can rewrite as:

$$\begin{aligned}&\mathbb{P}\left(g(u) \neq Y \mid e(u) > t\right) \mathbb{P}\left(e(u) > t\right) + \\ &\mathbb{P}\left(g(u) \neq Y \mid e(u) \leq t\right) \mathbb{P}\left(e(u) \leq t\right),\end{aligned} \tag{2}$$

where $t > 0$ is a threshold arbitrarily large.

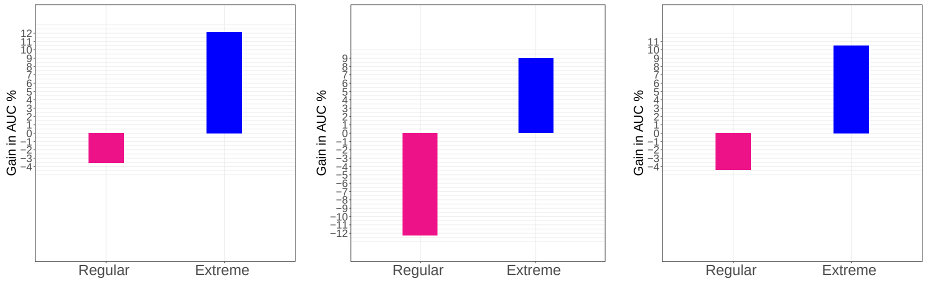
Although, because of the extremely small order of magnitude of $\mathbb{P}\left(e(u) > t\right)$ and of its empirical counterpart, nothing guarantees that the minimizer of the empirical risk on all nodes will be optimal on nodes $\{u : e(u) > t\}$. Therefore we follow the urge of [10,11] to minimize two risks following the decomposition given by Equation (2): one dedicated to the extreme nodes $R_{>t} = \mathbb{P}\left(g(u) \neq Y \mid e(u) > t\right)$ and one dedicated to the common nodes : $R_{\leq t} = \mathbb{P}\left(g(u) \neq Y \mid e(u) \leq t\right)$. Sorting the training nodes by decreasing order of number of edges, we introduce the order statistics $u_{(1)} > \ldots > u_{(n)}$ and we denote by $Y_{(i)}$ the corresponding sorted labels. Let $\tau > 0$ represent a small fraction corresponding to the proportion of considered extreme nodes, and set $k = \lfloor n\tau \rfloor$ such that $0 < k \ll n$. We define two risks namely $\widehat{R}_{>k}$ and $\widehat{R}_{\leq k}$,

$$\widehat{R}_{>k}(g) = \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}\{g(u_{(i)}) \neq Y_{(i)}\},$$

$$\widehat{R}_{\leq k}(g) = \frac{1}{n-k}\sum_{i=k+1}^{n} \mathbb{1}\{g(u_{(i)}) \neq Y_{(i)}\}.$$

to learn a suitable representation to perform node classification on each dedicated set of nodes.

4 Experiments

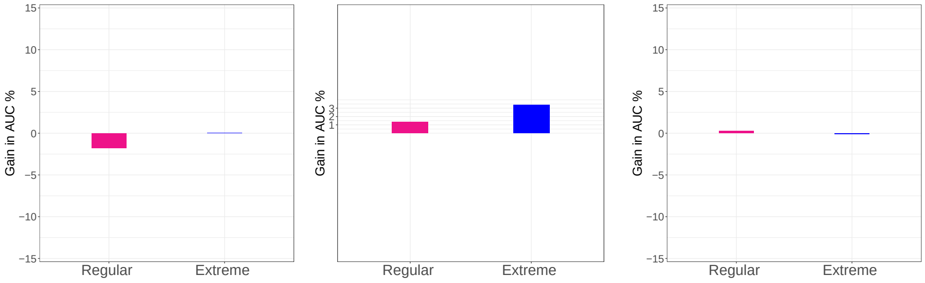


(a) GAT (b) GCN (c) GraphSAGE

Fig. 3: AUC gain difference between the baseline and extreme-regular version for **CORA**.

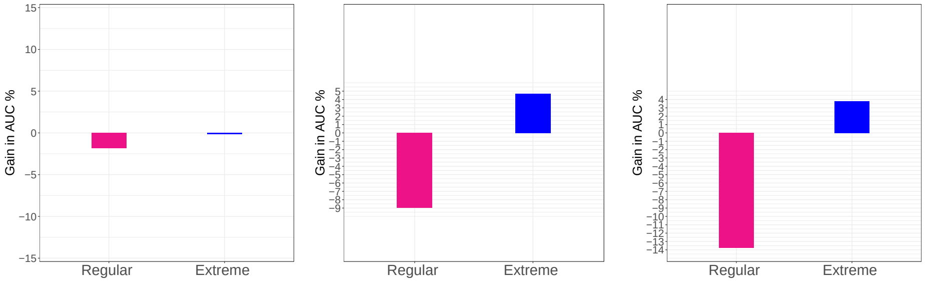
To evaluate the advantage of the proposed approach, we measure the Area Under the ROC Curve [5] of the baseline model's predictions for the extreme and the regular nodes of the test set separately. Subsequently, we compute the AUC of the EGNN in the extreme nodes of the test set and compute the difference with the aforementioned AUC of the baseline in the same nodes. The same difference is evaluated for RGNN and the regular nodes in the test set, as is noted in Figure (2). The aforementioned hyperparameter p is optimized using the same differences in the validation set.

This procedure is applied in three of the most common GNN benchmark datasets, the **CORA** and **PubMed** citation networks [6] and the **Amazon-Photo** co-purchase network [14]. In terms of methods, we test the three most well-known GNN architectures: GCN [12], GAT [21] and GraphSAGE [8]. The size of the hidden layers is set to 64 and the learning rate to 0.01, similar to [17]. As mentioned above, the theoretical properties of *EVT* have been analyzed



(a) GAT (b) GCN (c) GraphSAGE

Fig. 4: AUC gain difference between the baseline and extreme-regular version for **PubMed**.



(a) GAT (b) GCN (c) GraphSAGE

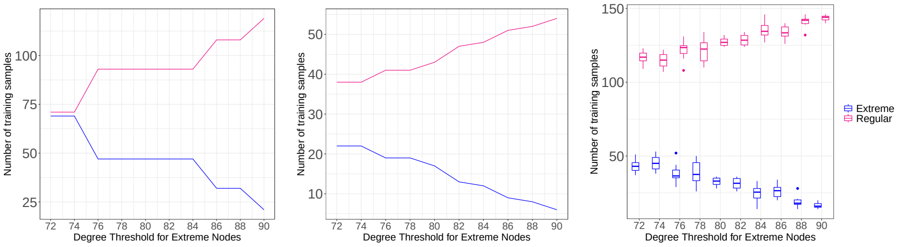
Fig. 5: AUC gain difference between the baseline and extreme-regular version for **Amazon-Photo**.

in the context of binary classification. Since the aforementioned datasets are multi-class problems, we transform them in binary classification using a one-vs-all classification , in a similar way as in [10]. We use AUC instead of plain accuracy for evaluation, because the final class distribution is imbalanced.

The **CORA** dataset consists of 2485 nodes and 5069 edges, with 5% of the dataset available for training. The results in 3 showcase the difference in AUC between the EGNN and the baseline (red bar) and the RGNN and baseline (blue bar). All models indicate a positive AUC gain over the extreme nodes and a negative over the regular ones. EGAT and ESAGE surpass the 10% AUC gain in the extreme region, a possible reason being that the baseline could not generalize to the extreme nodes in the test set due to insufficient training samples. We also see a tradeoff between AUC in the two regions, as the one increases when the other diminishes. In contrast, **PubMed**, which is larger (19717 nodes and 44324 edges), indicates a significantly smaller gain. This contradiction takes place due to the **PubMed**'s scale and minuscule label rate. More specifically, Figure (6a) shows the number of train samples for **CORA** and (6b) for **PubMed** respectively, broken in extreme and regular samples for all examined thresholds p. One can see that though **PubMed** is almost 10 times larger, it contains half the train samples of **CORA**, which renders the extreme nodes of the train set as few as 6. A train set of such scale does not suffice for the extreme models to learn. On the other hand, this affects the distribution of the regular nodes as well, as their test set is diminished and hence the prediction becomes easier. Most notably, GCN achieves a positive gain in both types of nodes (1% and 3%) (4b).

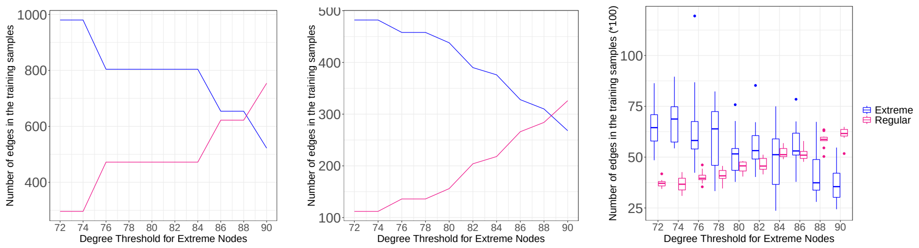
Similar minuscule differences can be seen in the **Amazon-Photo** dataset for the regular and extreme GAT in 5a (-1.5% and 0.7% respectively). The differences become more prevalent for the next two models, where the difference between the regular GAT and baseline, surpasses the respective difference for the extreme GNN. In an attempt to explain this difference, we dive deeper into the structure of the input samples.

GNNs rely on the structure of the graph as much as they rely on supervision. This is also prevalent in our results, as the extreme nodes are classified correctly more frequently than the regular ones. Specifically, throughout all experiments, RGNN surpasses EGNN only in one case in **PubMed** (Figure (4c) being a mere 0.3% difference. This happens despite EGNN being trained in a much smaller training set, in terms of the number of samples, as indicated by Figure (6). A possible reason is that the models trained on extreme nodes have access to more structural information, in terms of number of edges, as shown in Figure (7). Note that since the **Amazon-Photo** train/test split is not predefined, we perform 10 times the sampling mentioned above and showcase all values with boxplots. The total size of the train set is predetermined and stable, hence the number of nodes of the extreme and regular is complementary throughout all figures in (6).



(a) **CORA**($n = 2485$) (b) **PubMed**($n = 19717$) (c) **Amazon-Ph**($n = 7487$)

Fig. 6: The number of extreme and regular nodes that the train set contains in every degree threshold p.



(a) **CORA**($e = 5069$) (b) **PubMed**($e = 44324$) (c) **Amazon-Ph**($e = 119043$)

Fig. 7: The total number of edges belonging to extreme and regular nodes in the train set, for all degree thresholds p.

5 Conclusion

In this work, we examined the use of node discrimination in the context of GNN. This method has been examined for binary classification of non-structured input and is theoretically grounded from the Extreme Value Theory. We proposed a similar approach to separate the observed and test samples of a GNN based on their degree in extreme and regular. Our results indicate that the split can benefit the extreme samples. While we observed a case where this separation was beneficial for both types of nodes e.g. GCN at **PubMed**, the regular samples tend to suffer a loss, despite having relatively bigger train set.We examined this pattern further and attributed the increase in the extreme samples' accuracy to the number of edges associated with each sample. This means that when regular and extreme nodes do not have substantial difference in the number of edges i.e. the dataset is dense such as **Amazon-Photo**, the separation is not beneficial. We hope that our study, though inaugural, encourages future works to leverage further the structural role of the samples towards improving semi-supervised graph learning. In the future, we plan to examine automatic adjustment of the number of hidden parameters based on the ratio of extremes to regular nodes, as well as a node's influence on other nodes as a structural criterion [16,**?**].

References

1. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. science **286**(5439), 509–512 (1999)
2. Bianchi, F.M., Grattarola, D., Alippi, C., Livi, L.: Graph neural networks with convolutional arma filters. arXiv preprint arXiv:1901.01343 (2019)
3. Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., Blaschke, T.: The rise of deep learning in drug discovery. Drug discovery today **23**(6), 1241–1250 (2018)
4. Chung, F.R., Graham, F.C.: Spectral graph theory. No. 92, American Mathematical Soc. (1997)
5. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd international conference on Machine learning. pp. 233–240 (2006)

6. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
7. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1263–1272. JMLR. org (2017)
8. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in neural information processing systems. pp. 1024–1034 (2017)
9. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584 (2017)
10. Jalalzai, H., Clémençon, S., Sabourin, A.: On binary classification in extreme regions. In: Advances in Neural Information Processing Systems. pp. 3092–3100 (2018)
11. Jalalzai, H., Colombo, P., Clavel, C., Gaussier, E., Varni, G., Vignon, E., Sabourin, A.: Heavy-tailed representations, text polarity classification & data augmentation. arXiv preprint arXiv:2003.11593 (2020)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
13. Klicpera, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997 (2018)
14. Klicpera, J., Weißenberger, S., Günnemann, S.: Diffusion improves graph learning. In: Advances in Neural Information Processing Systems. pp. 13333–13345 (2019)
15. Monti, F., Frasca, F., Eynard, D., Mannion, D., Bronstein, M.M.: Fake news detection on social media using geometric deep learning. arXiv preprint arXiv:1902.06673 (2019)
16. Panagopoulos, G., Malliaros, F.D., Vazirgiannis, M.: Multi-task learning for influence estimation and maximization. arXiv preprint arXiv:1904.08804 (2019)
17. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
18. Stretcu, O., Viswanathan, K., Movshovitz-Attias, D., Platanios, E., Ravi, S., Tomkins, A.: Graph agreement models for semi-supervised learning. In: Advances in Neural Information Processing Systems. pp. 8710–8720 (2019)
19. Strogatz, S.H.: Exploring complex networks. nature **410**(6825), 268–276 (2001)
20. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 817–826 (2009)
21. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
22. Wu, J., He, J., Xu, J.: Net: Degree-specific graph neural networks for node and graph classification. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 406–415 (2019)
23. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems (2020)
24. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
25. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. arXiv preprint arXiv:1806.03536 (2018)
26. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. arXiv preprint arXiv:1603.08861 (2016)