An EBM view of the GNN

John Y. Shin
Department of Computer Science
New York University
New York, NY 10033
jys308@nyu.edu

Prathamesh Dharangutte
Department of Computer Science
New York University
New York, NY 10033
ptd244@nyu.edu

Abstract

Graph Neural Networks are a popular variant of neural networks that take as input graph-structured data. In this project, we have considered the experimental question of casting graph neural networks in the energy-based view of [1]. We successfully implement this framework and benchmark our method against the standard GCN architecture [2]. In addition to generative features, we add generation over the adjacency matrix itself. While further experiments are necessary, we believe we have built a starting foundation for a robust energy-based hybrid GNN. Our code is at: https://github.com/johnshin86/MDS-DS-GA1013-project.

1 Introduction

Graph neural networks (GNNs) are a generalization of neural networks that take as input graph-structured data, typically in the form of an adjacency matrix or graph laplacian, and feature vectors defined on the nodes. They have found much success in tasks such as link prediction, node classification, and graph classification. Alongside the experimental success, recent theoretical work has been moving forward, with theoretical results on their depth [3], architectural alignment with algorithms [4], and their discriminative power [5]. Recently the work of [1] proposed viewing traditional classifiers as energy-based models, adjusting the softmax transfer function to the Boltzmann distribution, and adding an inner Stochastic Langevin Gradient Descent loop over the features. We have incorporated this framework into Graph Neural Networks, as well as added generation over the adjacency matrix itself. We have benchmarked our new network using the Cora and Pubmed datasets.

2 State of the Art

CORA, Pubmed and citeseer are three standard datasets often used to benchmark node classification tasks. As of May 5th, 2020, the top 3 architectures in accuracy for node classification on Cora are SplineCNN [6], GCN-LPA [7], and AS-GCN [8], which, at their core, are modifications of the standard "GCN" [2]. SplineCNN makes use of two new ideas: having pseudo-coordinates defined over the edges, as well as a new convolution operator with B-spline functions defined over these pseudo-coordinates. GCN-LPA combines GCNs with the label propagation algorithm, which propagates labels amongst neighbors. AS-GCN makes use of adaptive sampling during forward propagation to train on large graphs.

Likewise, the top 3 architectures in accuracy for node classification for PubMed are GCN-LPA [7], GNN RH-U [9], and DFNet-ATT [10]. GCN-LPA is the same as described above in Cora. GNN RH-U identifies issues with robustness in GCNs and defines a new robust loss function. DFNet-ATT makes use of a new spectral graph filter. For citeseer, GCN-LPA [7], AdaGCN [11] and PPNP [12] are the top performing architectures. AdaGCN leverages AdaBoost to extract knowledge from higher order neighbors and integrate them. PPNP proposes a new propagation scheme based on personalized

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

PageRank which preserves local information while at the same time extracting information from large neighborhood.

We would also like to note that the papers with top accuracy for the Cora and PubMed datasets (SplineCNN and GCN-LPA) appear to use different training/test splits than the one used in the original GCN paper. We achieve comparable results by using train/test splits following [7]. To be consistent with the literature, we use the original train/test splits in the GCN paper [2].

Our technique is in a sense "architecture agnostic" and can be used with any graph neural network with a similar message-passing scheme. In our study, we use the "vanilla" GCN [2] as a baseline comparison. In theory, we could adapt our technique with any of the leading architectures.

3 Methodology

Below, we have an outline of our algorithm, which is a modification of the algorithm in [1], adjusted for graph neural networks:

Algorithm 1 EBM-GNN training: Initialize f_θ, SLGD step-size α. SLGD noise σ, replay buffer B, SGLD steps η, reinitialization frequency ρ. Energy threshold τ.

Result: Trained network f_θ. Generative features, $\hat{x}_t$ in B. Generative graph adjacency, $\hat{A}$.

while *not converged* **do**
- $L_{\text{clf}}(\theta) = \text{xent}(f_\theta(x), y)$
- Batch sample x^i and y^i from dataset
- **for** *all* i **do**
  - Sample $\hat{x}_0^i \sim B$ with probability $1-\rho$, else $\hat{x}_0^i \sim \mathcal{U}(-1, 1)$.
  - **for** $t \in [1, 2, \dots \eta]$ **do**
    - Let $Z_t = \text{LogSumExp}_{y'} f_\theta(\hat{X}_t)[y']$ be the energy of the entire graph with the new features.
    - $\hat{x}_t^i = \hat{x}_{t-1}^i + \alpha \frac{\partial \text{LogSumExp}_{y'}(f_\theta(\hat{x}_{t-1}^i)[y'])/Z}{\partial \hat{x}_{t-1}^i} + \sigma \mathcal{N}(0, I)$
  - **end**
- **end**
- Let $Z_{\text{gen}} = \text{LogSumExp}_{y'} f_\theta(\hat{X})[y']$ be the energy of the entire graph with the new features.
- Let $Z_{\text{clf}} = \text{LogSumExp}_{y'} f_\theta(X)[y']$ be the energy of the entire graph with the original features.
- $L_{\text{gen}}(\theta) = \text{LogSumExp}_{y'}(f(x)[y'])/Z_{\text{clf}} - \text{LogSumExp}_{y'}(f(\hat{x}_t)[y'])/Z_{\text{gen}}$
- $L(\theta) = L_{\text{clf}}(\theta) + L_{\text{gen}}(\theta)$
- Compute gradients $\frac{\partial L(\theta)}{\partial \theta}$ and propagate.
- Add $\hat{x}_t$ to B.
- **for** *All the new features,* $\hat{x}_t^i$ **do**
  - **if** $\left| \textit{LogSumExp}_{y'}(f_\theta(\hat{x}_t^i)[y']) - \textit{LogSumExp}_{y'}(f_\theta(\hat{x}_t^j)[y']) \right| \leq \tau$ **then**
    - Add Link to Adjacency Matrix, $\hat{A}$.
  - **end**
- **end**

end

In words, it is a stochastic langevin gradient descent (SLGD) loop nested inside of an stochastic gradient descent (SGD) loop. The inner SLGD loop samples the generative features over some simple distribution, and minimizes the energy of the new features using SGLD. The outer SGD loop computes the cross-entropy of the classifier, and combines the loss for the classifier as well as the generative loss, and computes the gradient. Finally, within the SGD loop, we generate new links in the adjacency matrix.

The key factor for convergence when using Stochastic Langevin Gradient Descent training with graph structured data is the renormalization by the total graph energy. The intuitive justification for this is that the gradient correction for the features of a node takes into account the total energy of the graph, since the messages are passed over the entire graph in the forward pass.

In addition to the generation over features, we have incorporated generation over the adjacency matrix itself. The motivation for this stems from the fact that for traditional machine learning models, samples are thought to be independent but that need not be the case for graph structured data as nodes are connected by edges. We use the energy of a sample as a quantitative measure to incorporate edges between sampled data and existing graph. To do this, we compare the energy of two nodes, and add a link to the graph if difference between energy of two nodes are within threshold τ. For stability, we do this once every 50 epochs.

4 Results

We consider three citation network datasets CORA, Pubmed and Citeseer. Reported accuracies are average best performance on test set for 5 runs with random initialization, with train-test split followed from literature. We use dgl library [13] for implementing GCN upon which we build the energy based framework. The hyperparameters used are mentioned in 2. GCN-O and GCN-JEMO correspond to model with additional term in loss function which force the weight matrix $W^{(l)}$ for each layer l to be orthogonal. We observe better performance with this additional constraint. The additional term is:

$$\left\|(W^l)^T W^l - I\right\|_F \tag{1}$$

Where $W^{(l)}$ is the weight matrix of a given layer and the norm is the Frobenius norm.

Model	CORA	Pubmed	Citeseer
GCN	81.5	79.0	70.3
GCN-JEM	82.44	77.04	67.28
GCN-JEMO	83.66	77.6	66.72

Table 1: Performance on node classification task

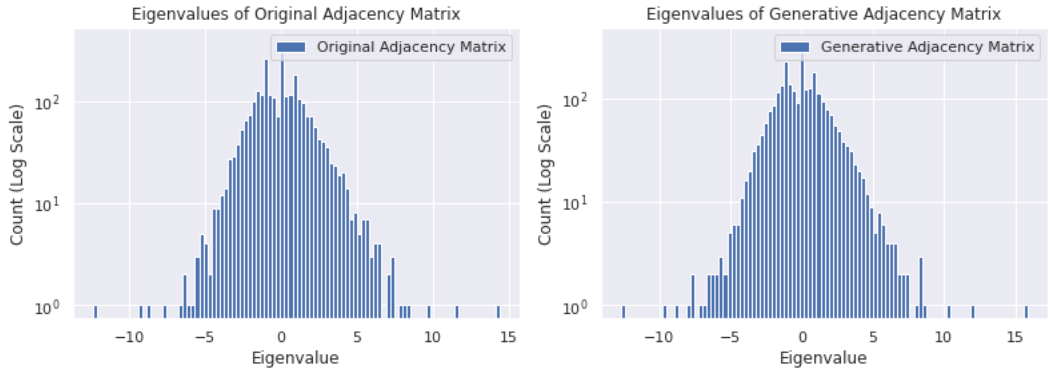
Parameter	Value
epoch	500
learning rate	0.01
SGLD lr	1
SGLD steps	20
ρ	0.05
SGLD noise	0.01
Sampling batch size	32

Table 2: Hyperparameters

5 Discussion

Our experimental results do not consistently improve the accuracy across three datasets (improvement for CORA), but it should be noted that the main focus of [1] was to improve the robustness of classifiers while having discriminative performance comparable to other top performing models. In [1], the authors mainly focus on comparing their method to other hybrid models, and in fact lose accuracy compared to a purely discriminative model. As such, we explore the generative capabilities of our model below.

In figure 1, we see the spectrum of the original adjacency matrix and the spectrum of the generative adjacency matrix for the Cora dataset. The x-axis is the eigenvalue, and the y-axis is the count with log scaling. The number of bins is set at 100. We can see that the spectrum of the adjacency matrix is quite similar. By only adding links to nodes that are close in energy, we surmise that the generative A is only adding short cycles. We can calculate the number of closed paths of length n with the following formula:



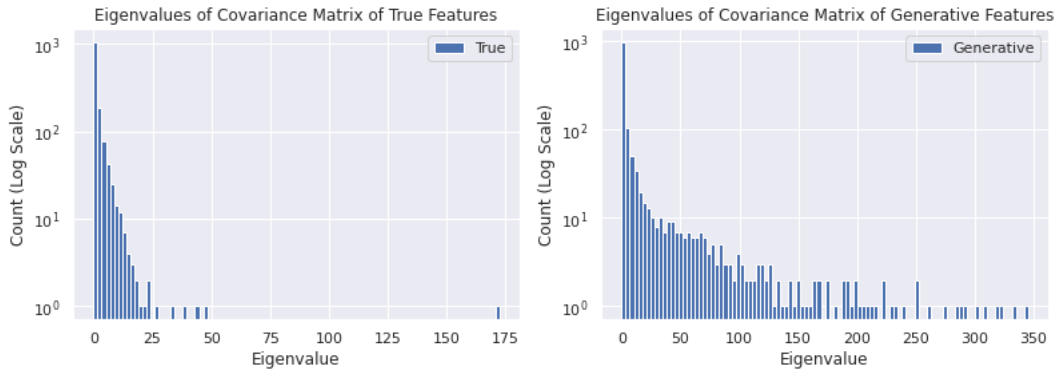
(a) Original A matrix

(b) Generative A matrix

Figure 1: Spectrum of Original Adjacency Matrix and Generative Adjacency Matrix. The x-axis is the eigenvalue, and the y-axis is the count with log scaling. While they look largely the same at eye-level, the number of short cycles in the generative A has increased.

$$\{\#\text{closed paths of length n}\} = \sum_i \lambda_i^n \tag{2}$$

Where λ_i is the spectrum of the adjacency matrix. We compute this for cycles of length $n = 3$ for both the original A matrix as well as the generative adjacency matrix. It is 9780 for the original matrix, and 10674 for the generative matrix.



(a) Covariance of Original Features

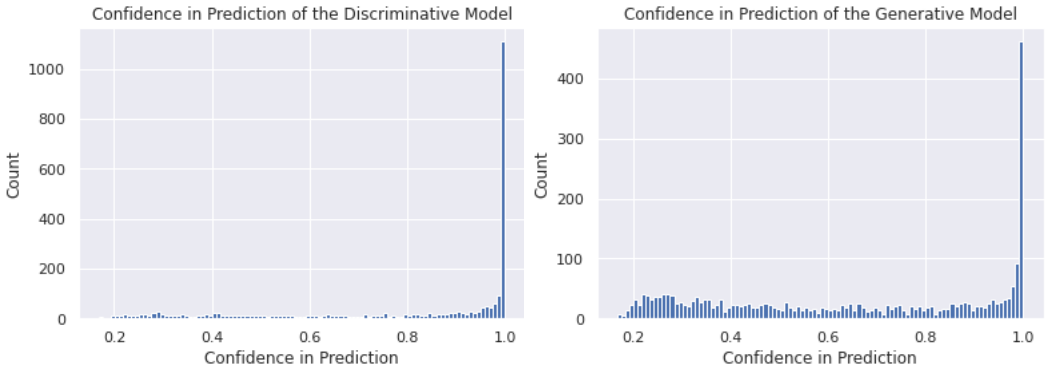
(b) Covariance of Generative Features

Figure 2: Spectrum of Original Features and Generative Features. The x-axis is the eigenvalue, and the y-axis is the count with log scaling. The generative features has many more high eigenvalues.

In figure 2, we see the spectrum of the covariance of the original features as well as the covariance of the generative features for the Cora dataset. The x-axis is the eigenvalue, and the y-axis is the count with log scaling. The number of bins is set at 100. The generative features add high covariance eigenvalues, which may add robustness against adversarial examples. We conjecture that if the features have more variance, then the classifier is less likely to be overtuned to the dataset.

In figure 3, we see the confidence of the prediction over the training dataset, where confidence is the max of the softmax of the output. The x-axis is the confidence in the prediction, and the y-axis is the count at regular scaling. The number of bins is set to 100. We can see that the generative model overall is less confident in it's predictions.

One metric to measure the calibration of a classifier is the Expected Calibration Error (ECE). First, one computes the confidence in the predictions of the dataset. Then, one bins these into equally spaced buckets. The ECE measures the absolute difference between the average accuracy over that



(a) Confidence of Discriminative Model

(b) Confidence of Generative Model

Figure 3: Histogram of Confidence in Predictions in Training Data. The x-axis is the confidence of the model on an example, where the confidence is the max of the softmax output. The y-axis is the count on a normal scale. Overall, the generative model is less confident.

bucket and the average confidence, weighed by the cardinality of the bucket. For a perfectly calibrated classifier, this value will be 0 for any choice of M.

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{3}$$

We compute an ECE of 0.52 for the generative model and 0.67 for the discriminative model over the test set (lower is better).

For future work, one point of further exploration is exploring the limitations of depth as set out in [3]. The paper posits that GNNs are limited in depth due to the action of repeated application of the graph operator. We would like to further explore if we can bypass this by perturbing A differently at every layer within our generative framework, and to see whether we can create deeper networks. One could potentially fix the adjacency matrix at the "top layer" (the input layer), and compute the energy in progressive layers using that layer's representation of the features, rather than at the softmax layer. This layer energy can then be used to adjust the adjacency matrix at a given layer. Each successive adjacency matrix represents information at different representation scales.

Another direction for future work would be using our generative model for tasks in computational chemistry and bioinformatics tasks, where graph structure is abound, and where many tasks are inverse problems–given some chemical property that one requires, what is the structure that gives that property? Such tasks would presumably be able to be tackled by a generative graph model.

Finally, we feel that the connection between GNNs and statistical mechanical models is strong, and yet this connection has yet to be fully explored in the current literature. We would like to expand what we have started here into exploring that connection more deeply.

References

[1] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

[2] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[3] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint cs.LG/1905.10947*, 2019.

[4] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

[5] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[6] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[7] Hongwei Wang and Jure Leskovec. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*, 2020.

[8] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph representation learning. In *Advances in neural information processing systems*, pages 4558–4567, 2018.

[9] Daniel Zügner and Stephan Günnemann. Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 246–256, 2019.

[10] WOK Asiri Suranga Wijesinghe and Qing Wang. Dfnets: Spectral cnns for graphs with feedback-looped filters. In *Advances in Neural Information Processing Systems*, pages 6007–6018, 2019.

[11] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. Adagcn: Adaboosting graph convolutional networks into deep models. *arXiv preprint arXiv:1908.05081*, 2019.

[12] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[13] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.