3.75

There are too few relevant details given for your model. Make sure that you fully understand and specify the method you are using (see comments.)

Flexible Variational Graph Auto-Encoders

Alex Kan[*]
Department of Computer Science
Department of Statistics and Operations Research
UNC-Chapel Hill
Chapel Hill, NC 25799
akan@unc.edu

Armaan Sethi
Department of Computer Science
Department of Physics and Astronomy
UNC-Chapel Hill
Chapel Hill, NC 27599
armaan@live.unc.edu

1 Introduction

In recent years, Graph Neural Networks (GNNs) have proved interesting for learning problems posed on non-grid structured data such as graphs, meshes, and point clouds. Recently, Kipf and Welling [12] have proposed a Graph Convolutional Network (GCN) layer that can be utilized for many learning tasks on graph represented data, including semi-supervised and unsupervised learning. These layers have the ability to learn flexible graph structure on the local level, and also can incorporate information gleaned from node features.

tepid

beef up sec

Our work is currently focused on the link prediction problem with the realm of graph-structured data. We have chosen to expand on the Variational Autoencoder (VAE) [11] framework with the goal of developing a flexible model that can learn a good latent representation of undirected graphs and accurately predict their edges.

2 Related Work

restructure para

There has been much previous work on deep learning for graph-structured data. Graphs that represent objects and their relationships are ubiquitous in the real world. Social networks, e-commerce networks, biological networks may all be represented using graphs [19]. Graphs are usually complicated structures that contain rich underlying value.

Graphs data can be extremely complicated with diverse structures. There are many different properties of a graph that are important to consider, that are not represented in other data types. For example, graphs can be heterogeneous or homogeneous, weighted or unweighted, and signed or unsigned. Additionally, tasks on graphs can be either graph-focused such as classification and generation or node focused such as node classification and link prediction. These different tasks need different model architectures. Recently many new unsupervised methods such as Graph Autoencoders (GAEs), Graph Recurrent Neural Networks (Graph RNNs) and Graph Reinforcement learning have been used in order to learn the representation of graphs [19, 20].

There are several challenges with graph data that make using deep learning techniques nontrivial. While images, text, and audio all lie in a regular domain, graphs lie in an irregular domain making it hard to generalize many mathematical operations on graphs. One key example of this is that in a Convolutional Neural Network, it is not straightforward to define convolution and pooling operations on graphs. In the paper "Semi-supervised Classification with Graph Convolutional Networks" [12] one mathematical definition of a graph convolutional operator was proposed:

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta},$$

you need to define what all of these terms are.

[*]Project Github Repository may be found at: https://github.com/akan72/comp790

Preprint. Work in progress.

Table 1: Baseline Dataset Summary

Name	# of Nodes	# of Edges	Features per Node	# of Classes
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
Pubmed	19717	44338	500	3

Where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denotes the adjacency matrix with inserted self-loops and $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ its diagonal degree matrix.

This convolutional operation can be used in Graph Convolutional Networks (GCNs) for semi-supervised learning by utilizing node attributes and node labels to train a model for a specific task such as classification. There have also been many other graph convolutional operators proposed such as the Chebyshev Spectral Graph Convolutional operator from the "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering" [5] paper, the graph neural network operator from [15], the graph attentional operator from the "Graph Attention Networks"[16] paper, and many more.

One improvement to graph convolutions is a graph attention network (GAT). As opposed to GCNs, GATs allows for implicitly assigning different importances to nodes of a same neighborhood, enabling a leap in model capacity. Analyzing the learned attentional weights may lead to benefits in interpretability, as was the case in the machine translation domain. This was achieved by leveraging masked self-attentional layers and stacking layers in order to provide attention to different neighborhoods' features.

In the paper "Variational Graph Auto-Encoders" [13] the authors create a variational graph auto-encoder (VGAE) based on a variational auto-encoder (VAE) by using a Graph Convolutional Network (GCN) in the encoder and a simple inner product decoder for unsupervised learning on graph data. This method was able achieved competitive results on a link prediction task in citation networks.

Another way of generating graphs is by decomposing the graph generation process into a sequence of node and edge formations conditioned upon the graph structure generated so far. In the paper "GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models" [18] the authors introduce a benchmark suite of datasets, baselines, and novel evaluation metrics to quantitatively evaluate the performance of graph generation.

3 Methods

Our datasets are citation networks that come from PyTorch Geometric's "Datasets" package [7] which were introduced through the Planetoid framework [17]. We have ran our initial experiments on the 'Cora', 'Citeseer,' and 'Pubmed' datasets.

This needs to be explained

Each dataset contains an edgelist denoting citation links between documents. Feature vectors X, are comprised of bag-of-words representations of the documents that are cited. Although not currently utilized in our work, each dataset also includes a predicted class of document to which the node belongs.

During out initial experiments, we first applied our architecture to the MNIST Superpixels dataset [14], but were unable to learn a flexible enough model to decrease our loss to a reasonable amount (Figure 4). Although the link prediction problem that we primarily focus upon does not have an implied ordering to our nodes, this may be important for other types of graph-represented data. Without a particular ordering, the semantic meaning of a sample (such as the recognition of a handwritten MNIST digit) may be lost. We have chosen to not further explore this area at this time, because we feel that learning the "position" vectors necessary to the intrinsic meaning of MNIST digits is out of the scope of this project.

We remove all diagonal entries from our original adjacency matrix and then split all of the edges in our new matrix into train, test, and validation sets. Our training set contains 85% of the edges, validation has 5% of the edges, and test has 10%.We then normalize our adjacency matrix, and sparsify both it and our input features.

How exactly are you doing edge prediction using VAE?

Table 2: Baseline Dataset Results

Name	Test Accuracy	ROC AUC	AP
Cora	76%	91.4%	93.0%
Citeseer	75%	89.3%	91.0%
Pubmed	75%	94.3%	94.3%

Our baseline model utilizes the same architecture as in Variational Graph Auto-Encoders [13].For our encoder network, we use a 32-dimension hidden layer along with a 16-dimension latent code space. For our baselines, our decoder network consists of a simple inner product between of our latent codes and their transpose [8].

if you have latent codes per node, you have to specify this somewhere.

We chose Adam as our optimizer [10] and used a learning rate of 0.01. In order optimize our loss function we used Pyro's [2] built in Stochastic Variational Inference (SVI) Module, the "Trace_ELBO" loss function.

The "guide" portion of our GAE class is where we utilize our encoder network in order to learn $q_\phi(z)$, an approximation of $p_\theta(z|x)$, and then sample from our latent code z with the parameters learned by our encoder network. The "model" portion of the network decodes these sampled codes into a new adjacency matrix and then scores it against our observations with a Bernoulli distribution.

Because we can't directly minimize the KL divergence between $q_\phi(z)$ and $p_\theta(z|x)$, we can however, optimize an different objective function: the Evidence Lower Bound (ELBO). The minimization of the KL divergence between two distributions is the same as maximizing the ELBO [3]:

what are these in your model

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x,z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)]$$
$$\log p(x) = ELBO(\lambda) + \mathbb{KL}(q_\lambda(z \mid x) \,||\, p(z \mid x))$$

During training, we take gradients with respect to our ELBO objective function and the normalize our training loss. We then evaluate the embeddings learned by our model on the held out sets of validation and test edges; we compute the accuracy of our link prediction, along with the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve and the Average Precision (AP) of our predictions.

you need to write out the model in [13] if this is your base!

4 Preliminary Results

As in Variational Graph Auto-Encoders paper, we trained our Pyro Implementation for 200 epochs, using the Adam optimizer and a learning rate of 0.01. On the link prediction task, we believe that the current baseline model is able to learn a good representation of the undirected graphs in the Cora and Citeseer datasets (Figures 1 and 2). However, we do have concerns about our model's ability to correctly predict edges in the Pubmed dataset (Figure 4), as evidenced by the strange ROC curve produced.

5 Future Plans

Our most immediate plans for involve running experiments in the same setting, but with the usage of different distributions for our chosen prior distribution and the distribution used to score our learned embeddings. Using a Student's t-distribution with various degrees of freedom rather than a Normal distribution when sampling our latent codes may be useful if samples from a heavier tailed distribution are desirable.

Using a Gumbel-Softmax Distribution rather than than a Weighted Bernoulli distribution to score embeddings is another area of interest. It has been shown that this distribution can be used to yield better estimators in the same setting where a Bernoulli distribution is commonly used [9].

Our next extension is to run our experiments on the same datasets with a more flexible Decoder network, rather than use a simple inner product of the learned latent codes. We are currently trying decoders also comprised of the previously described GCN layers, with the aim of emulating the upconvolution operation as is usually used in CNN models for images. We plan to accomplish the above three tasks within the next two weeks.

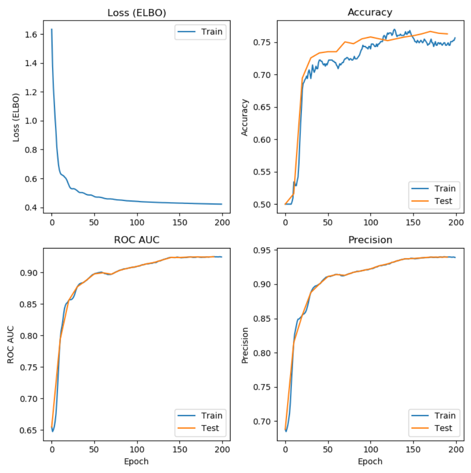

Figure 1: Preliminary Results on the Cora Citation dataset.

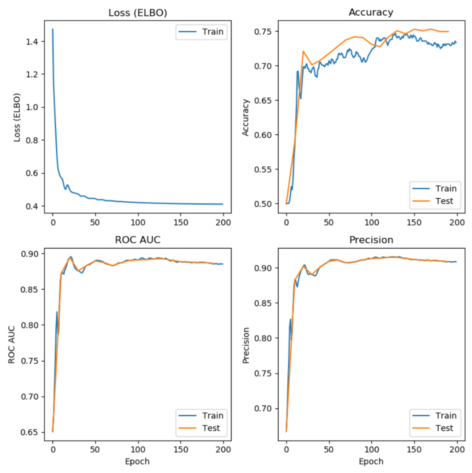

Figure 2: Preliminary Results on the Citeseer Citation dataset.

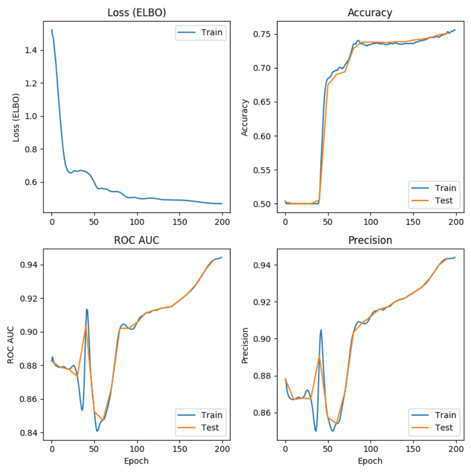

Figure 3: Preliminary Results on the Pubmed Citation dataset.

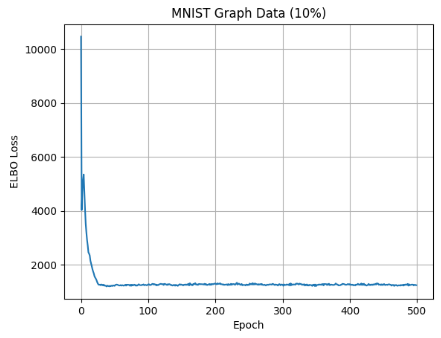

Figure 4: Preliminary Results on the MNIST Superpixels dataset.

We also intend to look into present work done on graph similarity metrics in order to try out alternative loss functions. To accomplish, we have to overload the current Pyro loss class currently implemented as "Trace_ELBO()." We are interested to see how the L2 loss and Wasserstein/Earth Mover's Distance perform in this setting. Implementing a new loss function with the Pyro backend is possible, but will take more time to accomplish.

Recently there has been work on Hierarchicial Autoencoders with Autoregressive posterior distributions [4]. This also seems to be a highly interesting area of research but applied to problems within the domain of graph-represented data.

If we desire additional experiments on new synthetic datasets, we plan to use longstanding graph generation benchmarks (Erdos-Renyi [6], Barabasi-Albert [1]) model as additional benchmarking datasets. We believe that the Erdos-Renyi model would be another good benchmark because link generation is also parameterized by a Bernoulli Distribution.

References

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[2] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 2018.

[3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[4] Jeffrey De Fauw, Sander Dieleman, and Karen Simonyan. Hierarchical autoregressive image models with auxiliary decoders. *arXiv preprint arXiv:1903.04933*, 2019.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[6] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[7] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[8] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[9] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[13] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[14] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *CoRR*, abs/1611.08402, 2016.

[15] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *arXiv preprint arXiv:1810.02244*, 2018.

[16] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[17] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

[18] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: A deep generative model for graphs. *CoRR*, abs/1802.08773, 2018.

[19] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *CoRR*, abs/1812.04202, 2018.

[20] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.