Disease Gene Association with Adversarial Graph Frameworks

Mamoon Habib
Kennedy College of Sciences
University of Massachusetts Lowell
Lowell, USA
mamoon_habib@student.uml.edu

***Abstract*—Disease-gene prediction (DGP) refers to the computational challenge of predicting associations between genes and diseases. Effective solutions to the DGP problem have the potential to accelerate the therapeutic development pipeline at early stages via efficient prioritization of candidate genes for various diseases. Graph embedding is an effective method to represent graph data in a low dimensional space for graph analytics. Most existing embedding algorithms typically focus on preserving the topological structure or minimizing the reconstruction errors of graph data, but they have mostly ignored the data distribution of the latent embeddings from the graphs, which often results in inferior embedding in real-world graph data. In this project, I apply two robust embedding methods, adversarially regularized graph autoencoder (ARGA) and adversarially regularized variational graph autoencoder (ARVGA), that take inspiration from adversarial learning methods, as a promising unsupervised approach for learning powerful latent embeddings in disease-gene networks that can be used for the DGP problem.**

***Index Terms*—gnn, adversarial learning, link-predicition, graph embeddings**

I. Introduction

Today, many well-known deep learning methods exist for data that comes in the form of sequences (i.e. words in a sentence) or images. In fact, all these types of data can be viewed as special cases of graphs: sequences are graphs that are linear in structure, and images are graphs with a structured lattice of nodes. GNNs extend prior methods and work on generalized input graphs that can have any arbitrary shape. Although there are many different variations of graph neural network layers, at the heart of each layer are three steps: message passing, aggregation, and update. GNNs convert graph data into a low-dimensional, compact, and continuous feature space. The key idea is to preserve the topological structure, vertex content, and other side information. This new learning paradigm has shifted the tasks of seeking complex models for classification, clustering, and link prediction to learning a robust representation of the graph data so that any graph analytic task can be easily performed by employing simple traditional models.

Graph embedding algorithms can be classified into three categories: probabilistic models, matrix factorization-based algorithms, and deep learning-based algorithms. These approaches are typically unregularized and focus on preserving the structure relationship (probabilistic approaches) or minimizing the reconstruction error (matrix factorization or deep learning methods). They have mostly ignored the data distribution of the latent codes. In practice, unregularized embedding approaches often teach identity mapping where the latent embeddings are free of any structure and can easily result in poor representation in dealing with real-world sparse and noisy graph data. One common way to handle this problem is to introduce some regularization to the latent codes and enforce them to follow some prior data distribution. Recently generative adversarial-based frameworks have also been developed for learning robust latent representation [2]. However, none of these frameworks is specifically for graph data, where both topological structure and content information are required to embed to a latent space. Therefore, to be able to learn robust latent embeddings while applying adversarial frameworks, [1] propose a novel adversarial framework with two variants, namely adversarially regularized graph autoencoder (ARGA) and adversarially regularized variational graph autoencoder (ARVGA), for graph embeddings. The theme of this framework is to not only minimize the reconstruction errors of the graph structure but also to enforce the latent codes to match a prior distribution. By exploiting both graph structure and node content with a GNN, ARGA and ARVGA encode the graph data in the latent space. With a decoder aiming at reconstructing the topological graph information, further an adversarial training scheme to regularize the latent codes to learn a robust graph representation is incorporated. The adversarial training module aims to discriminate if the latent codes are from a real prior distribution or from the graph encoder. The graph encoder learning, and adversarial regularization are jointly optimized in a unified framework so

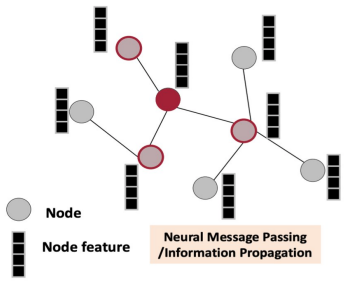

Fig. 1. Overview of a Graph Neural Network.

that each can be beneficial to the other and finally lead to a better graph embedding.

The disease-gene association prediction problem aims to find new associations between diseases and genes, based on existing data on diseases, genes, and known associations. Finding the linkage between genes and diseases could be crucial for applications in the pharmaceutical industry, where drugs are often developed to target specific proteins encoded by genes. More broadly, this knowledge could help scientists understand the functional similarities between genes and the underlying genetic similarities between diseases. Traditional processes for discovering disease-gene associations can be labor and data-intensive. For example, genome-wide association studies (GWAS) require finding a specific cohort of subjects and dedicated genetics software to analyze the data. A deep learning approach could bypass these difficulties by leveraging the model's ability to learn latent features in predicting new associations. In particular, the task can be modeled as a link prediction task on a graph where nodes represent genes and diseases, and edges represent associations between the genes and diseases.

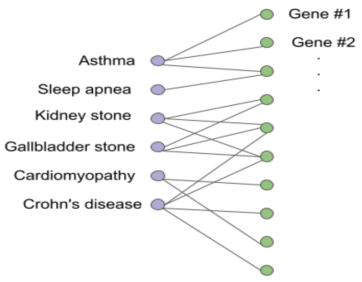

Fig. 2. Sample Image of Disease-Gene Connections.

II. Related Work

Given the clinical relevance of the DGP problem, there has been significant interest in developing computational methodologies to tackle the problem over the past several years. The approaches can be categorized into three dominant classes: linkage methods, module-based methods, and diffusion methods. Earlier, candidate genes were identified through the analysis of a protein interaction network. Candidate genes are characterized by having an interaction with another gene (via corresponding proteins) known to have an association with a particular disease. From this perspective, a gene can qualify as a candidate if and only if it interacts with another gene that is known to have an association with the disease of interest, inherently limiting the scope of the search for candidate genes. In addition to the traditional classes of approaches for tackling the DGP problem, representation learning approaches have recently emerged, which are focused on using graph-based methods to learn feature representations of nodes in relevant biological networks, often coupled with machine learning to identify candidate genes. Very recently, [3] introduced two unsupervised probabilistic approaches; Variational Graph Autoencoder (VGAE) and Constrained Variational Graph Autoencoder (C-VGAE). The work presented takes a step towards bridging the gap between graph neural network-driven methods and computational challenges in biological sciences, while proposing a more generalizable extension to the VGAE for specific link prediction in heterogeneous graphs by outperforming baseline random-walk methods.

III. Methods

A. Dataset

The disease-gene association network from the Stanford Biomedical Network Dataset Collection, which contains 7813 nodes in total (7294 genes, 519 diseases and 21375 edges) is used to evaluate methods for the DGP problem. The disease-gene associations that comprise this network are collected from the Online Mendelian Inheritance in Man (OMIM) database [4], the Comparative Toxicogenomics Database, the DisGeNET dataset [5], a comprehensive platform integrating information on human disease-associated genes and variants, and MINER, a large scale multimodal biological network developed by the Stanford SNAP group.

B. Overall Framework

The objective is to learn a robust embedding given a graph G = V, E, X. To this end, an adversarial architecture with a graph autoencoder to directly process the entire graph and learn a robust embedding. Figure 1 demonstrates the workflow of ARGA which consists of two modules: the graph autoencoder and the adversarial network.

- **Graph Convolutional Autoencoder.** The autoencoder takes in the structure of graph A and the node content X as inputs to learn a latent representation Z, and then reconstructs the graph structure A from Z.
- **Adversarial Regularization.** The adversarial network forces the latent codes to match a prior distribution by an adversarial training module, which discriminates whether the current latent code zi Z comes from the encoder or from the prior distribution.

C. Algorithm

Graph Convolutional Autoencoder. To represent both graph structure A and node content X in a unified framework, I used a novel GNN; graph convolutional network (GCN) [6] as a graph encoder:

$$Z\left(l+1\right)=f\left(Z\left(l\right),A\mid\mid W\left(l\right)\right) \tag{1}$$

A Variational Graph Encoder is defined by an inference model:

$$q(Z_i|X,A)=N(Z_i|mu_i,diag(\sigma^2)) \tag{2}$$

The decoder model is used to reconstruct the graph data. We can reconstruct either graph structure **A**, content information **X**, or both. In [1], the authors propose to reconstruct graph structure **A**, which provides more flexibility in the sense that the algorithm will still function properly even if there is no content information **A** available (e.g., X = I).

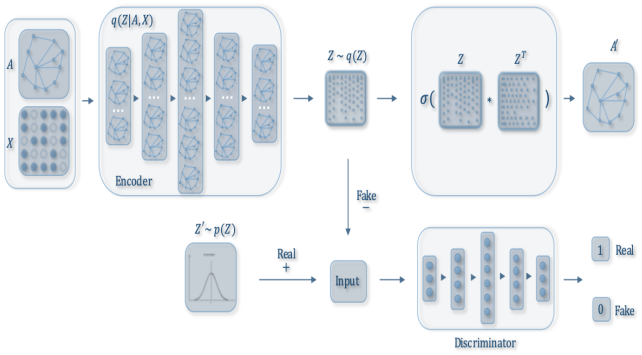

Fig. 3. The architecture of the adversarially regularized graph autoencoder (ARGA). The upper tier is a graph convolutional autoencoder that reconstructs a graph A from an embedding Z which is generated by the encoder which exploits graph structure A and the node content matrix X. The lower tier is an adversarial network trained to discriminate if a sample is generated from the embedding or from a prior distribution. The adversarially regularized variational graph autoencoder (ARVGA) is similar to ARGA except that it employs a variational graph autoencoder in the upper tier

The decoder $P(A'|Z)$ predicts whether there is a link between two nodes. More specifically, we train a link prediction layer based on the graph embedding:

$$p(\hat{A}_{ij} = 1|z_i, z_j) = sigmoid(z_i^T, z_j) \quad (3)$$

For the variational graph encoder, we try to optimize the variational lower bound as follows:

$$L = E_{q(Z|(X,A))}[logp(\hat{A}|Z)] - KL[q(Z|X,A)||p(Z)] \quad (4)$$

Adversarial Model. The key idea of the model is to enforce latent representation Z to match a prior distribution, which is achieved by an adversarial training model. The adversarial model is built on a standard multi-layer perceptron (MLP) where the output layer only has one dimension with a sigmoid function. The adversarial model acts as a discriminator to distinguish whether a latent code is from the prior p_z or from the Graph Encoder $G(X, A)$.

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))] \quad (5)$$

Algorithm Explanation. Given a graph G, the graph convolutional encoder gets the latent variables matrix Z and the real data distribution p_z respectively, to update the discriminator with the cross-entropy cost. After K runs of training the discriminator, the graph encoder will try to confuse the trained discriminator and update itself with generated gradient. We can update the graph autoencoder loss to train the adversarially regularized graph autoencoder (ARGA) or to train the adversarially regularized variational graph autoencoder (ARVGA), respectively. Finally, we will return the graph embedding $Z \in R^{nxd}$.

IV. Experiments

Two sets of experiments are performed, with one involving the standard ARGA, and the other ARVGA for disease-gene link prediction. Because the dataset only contains the structural information (links between disease and gene nodes), a uniform set of features is added to represent the features of the diseases and genes. To run the experiments on the dataset, PyTorch and PyTorch Geometric were used to train the GNN model. To create the train and test sets, one can split the dataset by partitioning the edges of the graph. This is done by hiding some edges from the model during each phase. As shown in Fig. 4., in the training phase, solid edges are used to predict the hidden edges (dotted). Similarly, during the test phase, a few more edges can be uncovered, and the model is allowed to use all the edges from training and validation. To set up the standard neural network pipeline I used the Adam optimizer, with a dropout rate of 0.5, and a learning rate of 0.001.

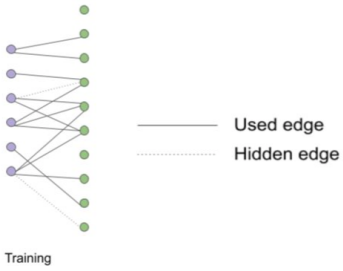

Fig. 4. Splitting the edges for training

Performance Metrics. Since the disease-gene association prediction problem is essentially a classification problem, where the model outputs whether there exists an edge or not, I analyzed the model's performance during both training and testing with two standard metrics: area under the ROC curve (AUC-ROC), and average precision (AP). **AUC-ROC** provides an aggregate measure of performance across all possible classification thresholds. For the link prediction problem, it represents how well the model can predict if a positive edge is really a positive edge (and vice versa if the negative edge is really a negative edge). A AUC-ROC closer to 1 means the model has good separability of the positive and negative edges. **AP** summarizes the precision-recall curve as the weighted mean of precisions at each threshold n. Intuitively, it is the area under the Precision-Recall curve. This performance metric indicates whether a model can identify all positive edges without accidentally marking too many negative edges as positive.

V. Results and Discussions

To compare the performances of ARGA and ARVGA model, I took the results from [3]. Previously used models included Node2Vec and DeepWalk as baselines and VGAE and C-VGAE as models I tried to improve from. **Table 1** shows results for both metrics; AUC-ROC and AP.

In both sets of experiments as seen in Table 1 **ARGA** and **ARVGA** outperforms previous random walk and Variational GNN methods. These results demonstrate the potential for a generative graph neural network driven methodology to

TABLE I
PERFORMANCE OF DGP PROBLEM

Methods	Performance Metrics	
	AUC	***AP***
NODE2VEC	79.6	81.4
DEEPWALK	79.5	79.2
VGAE	84.4	86.4
C-VGAE	90.8	91.3
ARGA	**93.4**	91.8
ARVGA	92.4	**92.1**

capture powerful latent structure that can be used for link prediction tasks in disease-gene networks.

Next, we can visualize the node embeddings of the latent representations Z before and after training the **ARVGA** models.**PCA** is employed on the embeddings to reduce the dimensionality to a 2D space. As shown in Fig. 5., initially the nodes are somewhat randomly scattered in the two-dimensional PCA projections. As shown in Fig. 6., by the end of training, the node embeddings have shifted to form more visible clusters differentiating gene and disease nodes. The model learns on its own from the graph connectivity information that it is a bipartite graph of diseases and nodes.

Fig. 5. PCA projection of the embeddings when ARVGA model is untrained. Blue colors are disease nodes and red colors are gene nodes.

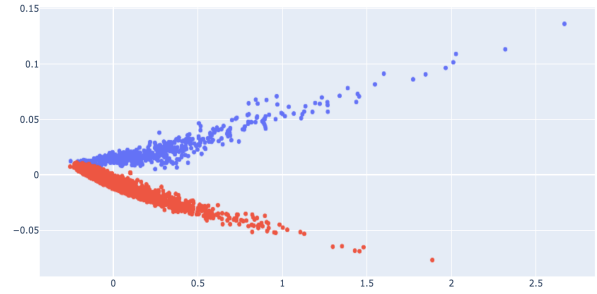

Fig. 6. PCA projection of the embeddings when ARVGA model is trained. Blue colors are disease nodes and red colors are gene nodes.

VI. CONCLUSION

In conclusion, I applied a novel adversarial graph embedding framework for link prediction on disease-gene networks for graph data. The adversarial module is jointly learned with a graph convolutional autoencoder to produce a robust representation. Experiment results demonstrated that ARGA and ARVGA outperform baselines in link prediction. As graph neural network approaches continue to develop, it is clear their application in the domain of biological sciences will be critical. The work presented shows how GNNs can bridge the gaps between network-driven methods and computational challenges in biological sciences, while proposing a more generalizable extension for specific link prediction in heterogeneous graphs.

Finally, there are additional ways to extend the models presented in the project for robust embeddings. For this project, I used a Graph Convolutional Network as the encoder for the input. This can be extended to using other novel GNNs such as GraphSage and Graph Attention Networks (GAT). Also, as mentioned before the dataset used uniform features for the feature matrix, one can try to add additional features by exploiting the random-walk methods that leverage neighborhood node connections and generate embeddings for each node. Lastly, one can also create try to create a new dataset by combining information from other datasets such as disease-disease/gene-gene interaction networks to add edges between disease and gene nodes.

ACKNOWLEDGMENT

I would like to thank Dr. Yu Cao for teaching the Deep Learning course and assigning us projects for the course to learn how to improve and apply Deep Learning algorithms to real-world datasets.

REFERENCES

[1] Pan, Shirui, et al. "Adversarially regularized graph autoencoder for graph embedding." arXiv preprint arXiv:1802.04407 (2018).
[2] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
[3] Singh, Vikash, and Pietro Lio. "Towards probabilistic generative models harnessing graph neural networks for disease-gene prediction." arXiv preprint arXiv:1907.05628 (2019).
[4] Amberger, Joanna S., et al. "OMIM. org: Online Mendelian Inheritance in Man (OMIM®), an online catalog of human genes and genetic disorders." Nucleic acids research 43.D1 (2015): D789-D798.
[5] Piñero, Janet, et al. "DisGeNET: a comprehensive platform integrating information on human disease-associated genes and variants." Nucleic acids research (2016): gkw943.
[6] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).