Graph Convolutional Networks to explore Drug and Disease Relationships in Biological Networks

Payal **Bajaj**[*], Suraj **Heereguppe**[*], Chiraag **Sumanth**[*]
pabajaj, hrsuraj, csumanth {@stanford.edu}

1 Abstract

Discovering drug-disease relationships is crucial for medical diagnosis and can provide critical insights for medical treatment. Traditionally, this analysis has been based on protein-protein interaction (PPI) graphs where a few known disease-associated proteins are used to identify other proteins by exploring network properties. The aim of this project is to explore recent graph convolutional networks on drug-disease interaction in biological networks. These networks represent nodes using node embeddings (vectors of floating point numbers) which are generated by incorporating features derived from their neighborhoods in the graphs. These embeddings can be learned in an unsupervised manner (to recover the graph structure) or supervised manner (combined with a downstream task). This project could potentially be applied to predicting and uncovering previously undiscovered or incompletely identified disease-drug interactions, with the help of these learned features, bootstrapping off a relatively smaller number of labeled nodes at the start of training (inductive learning). We plan to experiment with both methods for our problem of predicting proteins associated with a particular disease.

2 Introduction

Discovering drug-disease relationships is crucial for medical diagnosis and can provide critical insights for medical treatment. Traditionally, this analysis has been based on protein-protein interaction (PPI) graphs where a few known disease-associated proteins are used to identify other proteins by exploring network properties. In this project, we plan to experiment more recent approaches to characterize proteins for predicting proteins associated with a disease. we aim to focus primarily on graph convolutional approaches such as GCNs [2] and GraphSAGE[1] where structural properties of a node can directly be incorporated into the embeddings generated by the neural network. These approaches have demonstrated better results in multiple tasks such as node label classification and link prediction compared to node2vec[3]. The embeddings of these proteins can then be used as features in the prediction task.

We will follow the paradigm of machine learning by testing the hyper-parameters for each of these techniques and compare the relative effectiveness of these approaches versus each other and the results the authors present in [5]. We will evaluate the model using the number of disease-related proteins identified by the algorithm, more specifically evaluate on the precision-recall metrics produced by the various classifiers we build. Our baseline would start off using a logistic regression approach in trying to classify nodes of proteins in our Protein-Protein Interaction Networks, characterized by the features produced by node2vec to a respective disease group. We then follow this up by exploring and evaluating the the performance of both GCN and GraphSAGE on the same task.

We would like to explore if it is possible to manually incorporate additional information while learning node embeddings similar to [8]. For instance, we want to try to incorporate information into the process of learning node embeddings based on other literature such as node2vec[3] and identify if adding non-linear variants of such features improve performance as compared to using them directly for the prediction task.

[*]Authors worked together on the entirety of this project and have each contributed equally. Therefore, each author maybe graded on the basis of equal contribution to this project, on all aspects from conceptualization, implementation, experimentation and report generation

3 Problem Statement

Given a protein-protein interaction graph $G = (V, E)$ where V represents the proteins and E represents protein-protein interactions. We formulate the problem of discovery of protein-disease relationships as a node classification where the label corresponds to the disease that the protein(node) is associated with. Nodes are featurized with the help of node embeddings, as described earlier, which is the method used to encode predictive information.

4 Related Work

4.1 Discussion

Most diseases cannot be explained by defects in a single gene. Rather, they involve the coordinated function of distinct gene groups. Guney et al. [7] argue that drug development must shift focus from individual genes to a network-based perspective of disease mechanisms. The need for this change in analysis paradigm is born out of the fact that traditional drugs offer palliative cure rather than target the genetic cause of the disease. Current approaches are based on target profile similarity, defined as either the number of common targets between two drugs or as the shortest path between the drug targets.

While these studies are biased towards already studied proteins, the method outlined in this paper is argued to be unbiased and unsupervised, based on recent evidence that the genes associated with a particular disease tend to cluster in the same neighborhood called the disease module. This is a subnetwork within the interactome rich in disease proteins. The main hypothesis made by the authors is that an effective drug must target proteins within the corresponding disease module, or in the immediate vicinity. Protein-protein interaction, drug-disease association, and drug-target association data were integrated to test this hypothesis. A new drug-disease proximity measure is proposed, that quantifies the therapeutic effect of drugs. This helps distinguish palliative and effective treatments, while also offering an unsupervised approach to discover novel uses for existing drugs.

Menche et al. [6] explore network properties and claim that disease-associated proteins interacting with each other suggests that they tend to cluster in the same neighborhood of the interactome, forming a disease module, a connected subgraph that contains all molecular determinants of a disease. The accurate identification of the corresponding disease module represents the first step toward a systematic understanding of the molecular mechanisms underlying a complex disease. In this paper, the authors present a network-based framework to identify the location of disease modules within the interactome and use the overlap between the modules to predict disease-disease relationships.

Using network science, the authors showed show that we can only uncover disease modules for diseases whose number of associated genes exceeds a critical threshold determined by the network incompleteness. They found that the higher the degree of agglomeration of the disease proteins within the interactome, the higher the biological and functional similarity of the corresponding genes. If two disease modules overlap, local perturbations causing one disease can disrupt pathways of the other disease modules as well, resulting in shared clinical and pathobiological characteristics. These findings indicate that many local neighborhoods of the interactome represent the observable part of the true, larger and denser disease modules, and open doors for further exploration in drug target identification.

Agrawal et al [5] focus on identifying disease pathways in PPI network. They define disease pathways as sets of proteins associated with a given disease. They demonstrate that such pathways often form separate disconnected components in the graph using multiple distance metrics such as size of largest pathway component, density of pathway and distance of pathway components. Then they prove that standard techniques fail in such PPI networks as they mainly focus on proximity based measures to discover proteins related to a particular disease. They implement various proximity based approaches and prove empirical correlation between performance and distance metrics listed above. Thus in PPI networks where the disease pathways are disconnected, such standard proximity based techniques will not work. The novelty of their work is exploiting higher-order network structures such as motifs for discovering disease pathways by explicitly characterizing features from these motifs: they count the number of times a protein from a disease pathway participates at specific positions in these motifs. They concatenate these features with neural embeddings followed by a logistic regression classifier to predict

disease related proteins (or pathways) from the PPI network.

4.2 Critique

The authors in [5] experimented with a lot of techniques to discover disease pathways and focused on the importance of structural properties instead of standard proximity based metrics like density, largest component, etc. and demonstrated preliminary results for the same. However, they do not specify the details of the single layer neural network [3] and random walks they are using, making it difficult to understand why neural embeddings perform worse than when combined with higher order network structure especially when [3] offers a way to capture structural properties as well. Further, they do not explore the recent advances in the field of neural embeddings, absent from the work presented in [2] and [1] as well. More specifically, the graph convolutional approaches that are constructed to incorporate structural roles of nodes into their embeddings. It would be really interesting to compare the performance of their approach versus automatically learning higher order network structure properties of nodes using these recent approaches.

The work cited in [6] has a drawback wherein the authors rely on using external datasets to extract features (such as comorbidities) beyond what is available in the interactome network to establish one of their primary objectives. This makes the approach very specific, and may not generalize well when applied to characterizing other types of biological networks with different characteristic/feature requirements. Another issue the authors faced was with the incompleteness of the interactome they were dealing with itself. To be specific, they offered quantitative evidence for the identifiability of some disease modules, while showing that for other diseases the identifiability condition is not yet satisfied at the current level of incompleteness of the interactome. Thus, there are several instances of diseases that the authors could not uncover any valuable information from, despite their novel approach to tackle the incomplete interactome.

The major theme we see here is that most methods currently include less than 20% of all potential pairwise protein interactions in the human cell, which means that we seek to discover drug and disease associations relying on interactome maps that are 80% incomplete. Additionally, the gene lists of diseases and drugs remain incomplete. Because of the incompleteness of the interactome and the limited knowledge of disease- and drug-associated genes, it is not clear if the available data have sufficient coverage to map out modules associated with each disease and each drug. It provides a clear premise for us to explore recent advancements in the field of inductive learning approaches to networks in trying to featurize all nodes in an interaction network, which are found in these large incomplete regions and hence bridge this gap in an attempt to propose new protein-drug / disease-drug interactions.

5 Datasets

Protein-Protein Interactions Network: Protein–protein interactions (PPIs) are the physical contacts of high specificity established between two or more protein molecules as a result of biochemical events steered by electrostatic forces including the hydrophobic effect. Many are physical contacts with molecular associations between chains that occur in a cell or in a living organism in a specific biomolecular context. We use BioGRID for our analysis as a Protein-Protein interaction network. It is an interaction repository with data compiled through comprehensive curation efforts. The current version has about 1,499,723 protein and genetic interactions, 27,785 chemical associations and 38,559 post translational modifications from major model organism species. For the purpose of this project, we use the latest(2017) release of the database which has 67371 distinct protein identifiers.

Disease-Protein Interactions Network: Proteins do not function in isolation; it is their interactions with one another and also with other molecules (e.g. DNA, RNA) that mediate metabolic and signaling pathways, cellular processes, and organismal systems. Due to their central role in biological function, protein interactions also control the mechanisms leading to healthy and diseased states in organisms. Diseases are often caused by mutations affecting the binding interface or leading to biochemically dysfunctional allosteric changes in proteins. Therefore, protein interaction networks can elucidate the molecular basis of disease, which in turn can inform methods for prevention, diagnosis, and treatment. Disease-protein interactions are extracted from DisGeNET[9] which is a platform that centralized the knowledge on Mendelian and complex diseases. These diseases are mapped to UMLS (Unified Medical Language System) codes. Out of 2297 diseases (with protein associations from DisGeNET), about 545

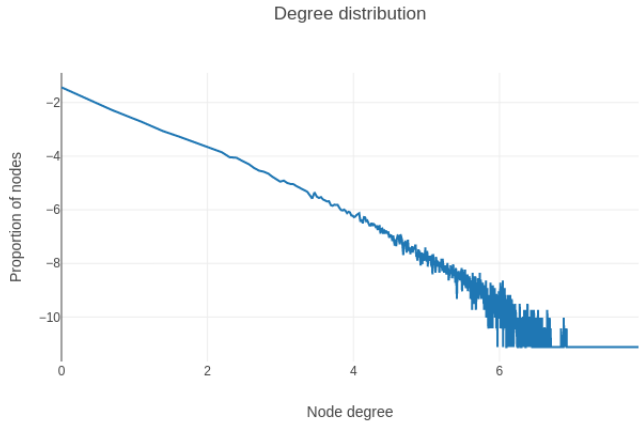


Figure 1: Degree Distribution in PPI network

UMLS Code	Class	# Training	# Test
DOID:0014667	0	609	173
DOID:14566	1	1794	478
DOID:7	2	2898	729
DOID:630	3	166	46
DOID:0080015	4	385	104
DOID:0050117	5	166	43
DOID:225	6	135	44
DOID:150	7	1264	335

Figure 2: Class Label Distribution

diseases are associated with a UMLS code. Following the authors' work in [5], we work with diseases that have at least 10 associated proteins. We get about 534 diseases after this filtering. These are mapped to 8 distinct UMLS codes, each representing a high-level disease group, which form the 8 classes in our classification task. We then mapped each of our proteins in the PPI network to their associated disease codes, crosswalked using the UMLS mappings.

Exploratory Data Analysis: The average clustering coefficient of this graph is 0.161678 and the strongest connected component contains 62036 proteins (90% of the total nodes). The degree distribution of this network is shown in the figure 1. The distribution appears to be a power law distribution where a few proteins have very high degree that appear in the tail of the power law.

We formulate disease-protein discovery as a node classification problem. Out of 67371 proteins in the PPI network, 5489 proteins are associated with at least one disease and thus have labels. Therefore, we split these proteins into 4391 nodes for training and 500 for validation and 598 for test. As proteins can be associated with multiple diseases, we can have more than one correct label for each protein. The distribution of proteins per label is presented in fig 2.

6 Approach

We use a combination of multiple techniques in our approach. In this section, we talk about such techniques and we explain how we use them in the experiments.

Node2vec: Node2vec is a framework that learns embeddings for nodes in a graph which are low-dimensional representations for nodes. The objective of the training node2vec on a graph optimizes neighborhood preserving objective. The objective is designed in a way such that it allows for various definitions of network neighborhoods by simulating biased random walks. Specifically, it provides a way of balancing the exploration-exploitation trade-off that in turn leads to representations that can be expressive enough to capture the diversity of connectivity patterns observed in networks, that is, they can be biased towards graph connectivity and structural equivalence based on the parameters provided to the algorithm.

Formally, given a source node u, we simulate a random walk of fixed length l. Let c_i denote the i^{th} node in the walk, starting with $c_0 = u$. Nodes c_i are generated by the following distribution:

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & if\ (v,x) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant.

$$\pi_{vx} = \alpha_{pq}(t,x) w_{vx} \tag{2}$$

where

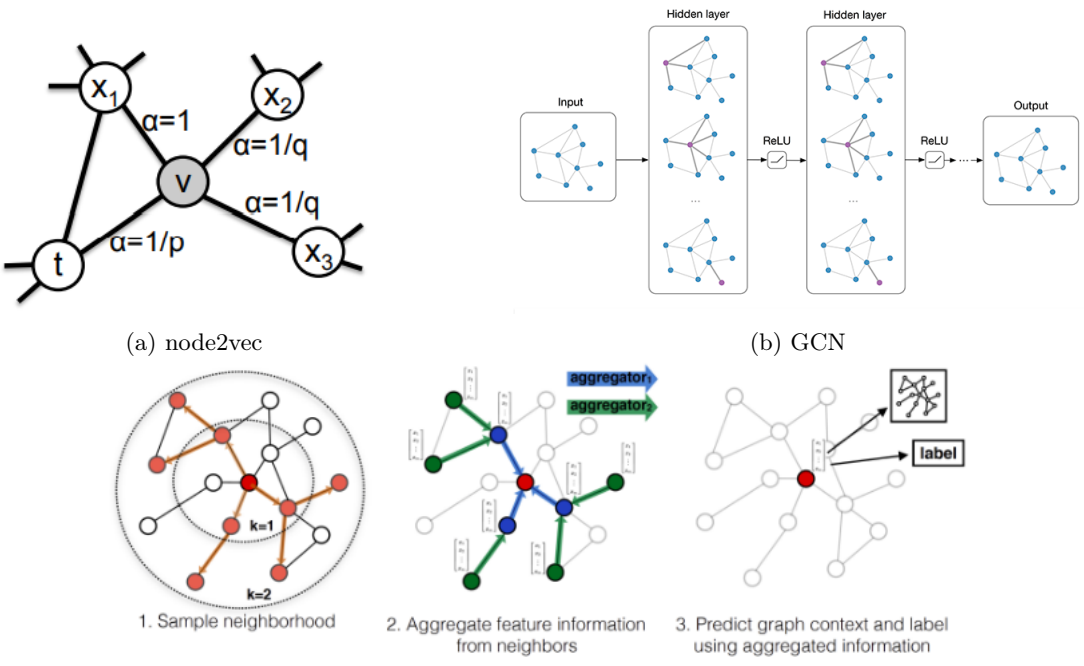
$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & if\ d_{tx} = 0 \\ 1 & if\ d_{tx} = 1 \\ \frac{1}{q} & if\ d_{tx} = 2 \end{cases} \tag{3}$$

α_{pq} provides a way to bias the random walk algorithm in a way to sample the next node based on static edge weights w_{vx}. There are several advantages of using this type of biased random walk. Not only is it computationally efficient in term of space and time complexity, biasing according to the edge weights actually helps encode the structural characteristics of the graph into the node embedding, which lends itself well to prediction tasks involving pairs of nodes instead of individual nodes, i.e., learn edge features as well, which proves very useful in our specific interactome context.

We used Node2Vec as our method of choice for unsupervised featurization of the nodes in our Protein-Protein interactome network. These node embeddings were then used in various downstream classification tasks, starting with our baseline logistic regression, as well as GCN and GraphSAGE implementations.

Logistic Regression Baseline: Logistic regression was directly applied on the obtained node-embedding vectors using Node2Vec, to perform classification based on the featurized representation of such networks. Our multinomial logistic regression model maps a 200 dimensional input feature vector into a prediction of 1 of 8 potential disease-group classes it could be associated with. This method was established as a a baslien to copare further downstream techniques such as GCN and GraphSAGE.

It is however, interesting to note that we did not ignore the very skewed distribution of training data across classes, and hence decided to use a weighted loss function for our logistic regression encouraging the network to not just learn from more commonly occurring classes, but also relatively incentivized the network to learn from under-represented classes as well.



(a) node2vec

(b) GCN

(c) GraphSAGE

Figure 3: Framework

Graph Convolutional Networks: GCNs [2] provide a semi-supervised learning framework based on node embeddings which is an efficient variant of convolutional neural networks to operate directly on graphs. This is ideal in our setting where only a few proteins have disease associated with them. For

a given graph, GCN trains embeddings for each node using its neighborhood to predict its class label. Learning node representations from local neighborhood for each node makes the model scale linearly in the number of graph edges. Embeddings are updated using the following equation:

$$H^{(l+1)} = \sigma(D^{-1/2}AD^{-1/2}H^{(l)}W^{(l)}) \tag{4}$$

where $H^{(l)}$ represents the embeddings when labels at l hops are taken into consideration, $W^{(l)}$ is the weight matrix for l^{th} hop, A is the adjacency matrix after adding self-loops and D is the degree matrix creating using the updated adjacency matrix (with self-loops).

We use the GCN code provided by the authors [10]. We started with Node2Vec embeddings and perform convolutions using the GCN architecture to produce classifications on protein node-disease association. Additionally, similar to Logistic Regression, we changed the original loss function with a weighted loss function to encourage better learning of the GCN neural network from under-represented classes as well, reducing the polarizing effect in training seen from highly over-represented classes.

GraphSAGE: Much like Graph Convolutional Networks, GraphSAGE is an efficient inductive learning algorithm that generates node embeddings through a semi-supervised learning approach. Instead of training individual embeddings for each node, GraphSAGE learns a function that generates embeddings by sampling and aggregating features from a node's local neighborhood.

The novel embedding generation technique used in this algorithm follows that in each iteration, or search depth, nodes aggregate information from their local neighbors, and as this process iterates, nodes incrementally gain more and more information from further reaches of the graph. After aggregating the neighboring feature vectors, GraphSAGE then concatenates the node's current representation, with the aggregated neighborhood vector, and this concatenated vector is fed through a fully connected layer with nonlinear activation function σ, which transforms the representations to be used at the next step of the algorithm.

Another interesting proposal in this work shows that the authors uniformly sample a fixed-size set of neighbors, instead of using full neighborhood sets in the algorithm, in order to improve the overall computational efficiency, and keep the computational footprint of each batch fixed.

7 Experiments and Results

Baseline: To recreate the results from the disease pathways literature [5], we have trained node embeddings in an unsupervised manner using node2vec [3, 4]. We simulated 50 walks, each of length 80 with parameters p and q to be 1. We repeated the experiments with different values of these parameters but didn't observe much difference. Here we report the values for the experiment where p and q are set to be 1.

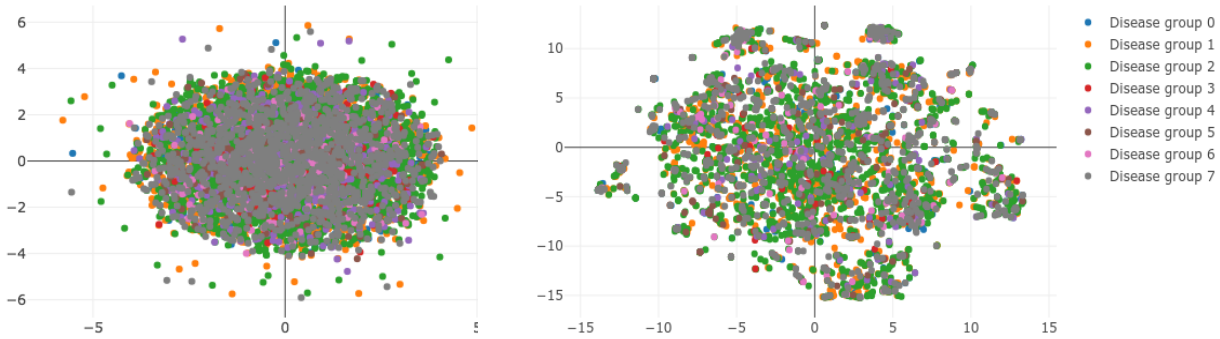
The embeddings generated using node2vec are visualized in fig 4. We also visualize random embeddings of the same nodes in 4a in order to demonstrate the results of node2vec. We can see that node2vec tries to group nodes in the same disease together as we can see small clusters of nodes corresponding to disease group 7 and how nodes are comparatively more spread out in 4b. However, we can see that most of the nodes corresponding to the same disease group are still pretty spread out an not necessarily form clean clusters for separate disease groups.

We then used multinomial logistic regression to predict the disease group of the proteins. The embeddings generated using node2vec form the node features for this algorithm. The results are summarized in table 1. Vanilla LR refers to the case where the loss weighs all data points equally and balanced LR refers to the case where loss weights data points differently based on the probability of the class they belong to.

As an extension to the baseline we experimented with a two-hidden layer deep neural network, trained on node2vec embeddings. We did notice a significant improvement in recall over our baseline logistic regression, but also a drop in precision across the outcome classes as the neural network tends to over-fit the training data, and even with regularization and other experimentation with the choice of non-linear

functions at the output layer, did not generalize well to unseen test data. Therefore, we decided to continue our efforts by using recent graph-convolution approaches GCN and GraphSAGE, outlined below.

GCN: Using GCN with pre-trained node-embeddings implies we are treating the node-embeddings as node features. We performed multiple experiments with different hyper-parameter values. The results of this search are summarized in fig 5. Overall, we found the training process to be pretty unstable as we can see from the jittery behavior of the loss function. We found that learning rate of 0.001 and dropout of 0.1 exhibited stable behavior than others as we can see that blue line is still comparatively consistent.

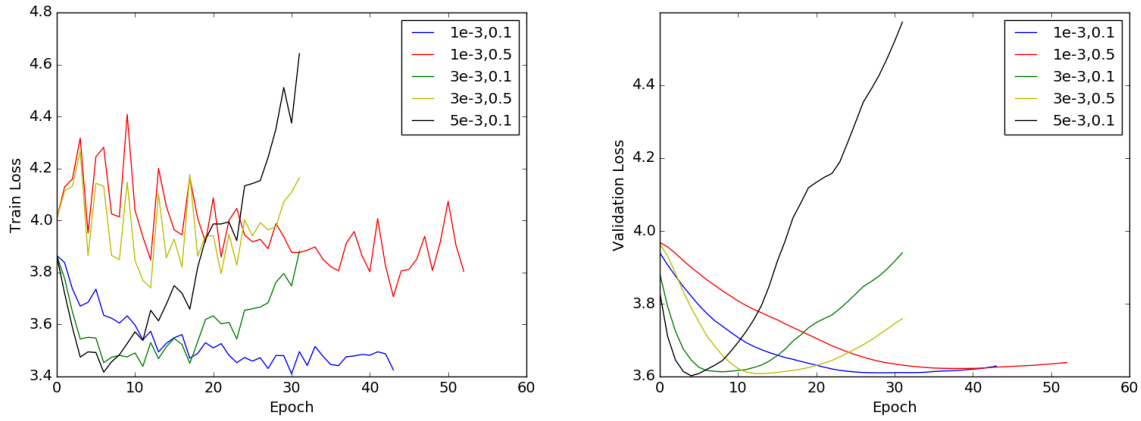


(a) Random (b) Node2vec

Figure 4: Node Embeddings

Class	0	1	2	3	4	5	6	7	Average
Vanilla LR - Node2vec Features									
Recall	0.0000	0.1695	0.82035	0.0000	0.0000	0.0000	0.0000	0.0239	0.1267
Precision	0.0000	0.4500	0.67342342	0.000	0.0000	0.0000	0.0000	0.2963	0.1775
Balanced LR - Node2vec Features									
Recall	0.1445	0.1548	0.2675	0.1304	0.1539	0.2093	0.0455	0.1373	0.2711
Precision	0.2841	0.5175	0.6964	0.0500	0.1567	0.0769	0.0161	0.3710	0.1554

Table 1: Baseline: Logistic Regression



(a) Training Loss (b) Validation Loss

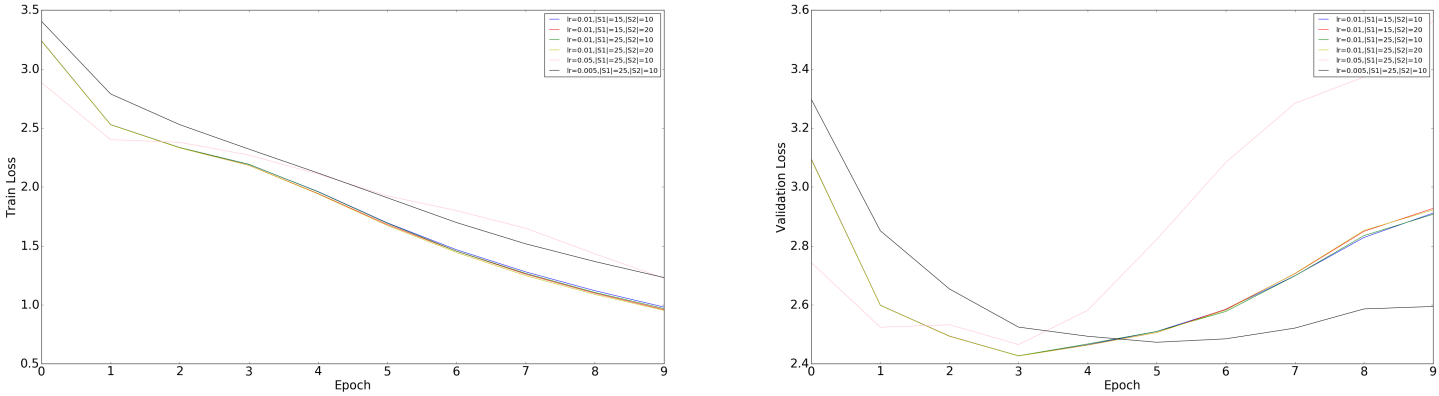
Figure 5: Graph Convolutional Networks

Table 2 presents the best recall values that could be achieved by GCN. We present the results for hyper-parameters which provide the best recall values. We present results for the case where loss is biased by the number of examples, that is unweighted loss, which we call Vanilla GCN and the case where we modify the loss function so that the weight of loss of sample is inversely proportional to number of examples in the class that sample belongs to. The latter case essentially forces the network to learn different class instead of predicting the most probable class (which is a problem we faced during milestone). We can see that per class recall values are more balanced/distributed in the latter case compared to Vanilla GCN where recall for class 2 dominates. Also, recall and precision for Vanilla GCN is very similar to logistic regression which implies that once we have node2vec features, we have all the neighborhood information we need and trying to gather more information from neighbors (which GCN does) is not necessary and a simple algorithm like logistic regression suffices.

Class	0	1	2	3	4	5	6	7	Average
Vanilla GCN - Node2vec Features; Hyper-parameters - lr=0.001,dr=0.1									
Recall	0.0121	0.2569	0.6836	0.0300	0.1000	0.0000	0.0000	0.0483	0.1251
Precision	0.25	0.4179	0.6982	0.4000	1.0000	0.0000	0.0000	0.2059	0.1965
Balanced GCN - Node2vec Features; Hyper-parameters - lr=0.001,dr=0.1									
Recall	0.01205	0.6009	0.1343	0.0615	0.0185	0.1000	0.5312	0.0690	0.1909
Precision	1.0000	0.4781	0.6429	0.1740	0.3333	0.0253	0.0370	0.1695	0.3859

Table 2: Results: Graph Convolutional Networks

GraphSAGE: Fig 6 summarizes the values of training and validation loss while training in GraphSAGE. We can see that training is more stable compared to GCN as the loss goes down in a consistent manner unlike GCN. We also see validation loss increases epoch 3, which is expected as the model starts to over-fit to training data. We also see that we get best results for learning rate 0.01 and no dropout. For these hyper-parameters values, neighborhood sample size doesn't affect loss much as can be seen in the plots. Further, we also show results for higher(0.05) and lower(0.005) learning rates to emphasize that both training and validation losses are lower for chosen hyper-parameters.



(a) Training Loss

(b) Validation Loss

Figure 6: Vanilla GraphSAGE

The results for GraphSAGE are summarized in table 3. We perform experiments with identity features - where the node embeddings are initialized randomly and node2vec features - where the node embeddings are initialized using node2vec results, like in LR and GCN. Note that we can't work with identity features in LR (because LR doesn't back-propagates into features so we need descriptive features) and GCN (because GCN works with all nodes in the graph in a batch mode and not in a mini-batch mode like GraphSAGE which forces using sparse feature representation which can't be back-propagated into as a limitation of tensorflow which in-turn implies that GCN needs representative features for training as well). As GraphSAGE uses sampling techniques, it can be made to work for mini-batches and thus allows us to back-propagate to input features thus making possible the use of identity features. Further,

we explored weighted and unweighted loss with GraphSAGE as well.

Vanilla GraphSAGE outperforms LR and GCN with node2vec features with and achieves the best average recall of 0.8059 with weighted loss.We believe that one the advantage of using GraphSAGE is that it has capability to predict multiple labels as we use *sigmoid* activation at the last layer and predict positive over threshold of 0.5. Detecting multiple labels in difficult in GCN due to reason discussed earlier. However, we tried alleviating this situation by drawing inspiration from GraphSAGE and investigated with a sigmoid non-linearity with thresholding. However, though we did manage to be able to significantly increase recall, we could not accept the drop in overall precision we obtained as a result.

Class	0	1	2	3	4	5	6	7	Average
Vanilla GraphSAGE - Identity Features; Hyper-parameters - lr=0.01,d=0,|S1|=25,|S2|=10									
Recall	0.0000	0.0550	0.8030	0.0000	0.0000	0.0000	0.0000	0.0966	0.1193
Precision	0.0000	0.2667	0.6642	0.0000	0.0000	0.0000	0.0000	0.4516	0.1728
Balanced GraphSAGE - Identity Features; Hyper-parameters - lr=0.1,d=0,|S1|=25,|S2|=10									
Recall	0.0964	0.2431	0.6687	0.0000	0.0000	0.0000	0.0000	0.3172	0.1657
Precision	0.1455	0.4309	0.6588	0.0000	0.0000	0.0000	0.0000	0.3217	0.1947
Vanilla GraphSAGE - Node2vec Features; Hyper-parameters - lr=0.01,d=0,|S1|=25,|S2|=10									
Recall	0.3470	0.4128	0.8508	0.1800	0.4928	0.6667	0.2745	0.1172	0.4177
Precision	0.5882	0.5056	0.6582	0.1300	0.1000	0.5327	0.2285	0.3954	0.3923
Balanced GraphSAGE - Node2vec Features; Hyper-parameters - lr=0.01,d=0,|S1|=25,|S2|=10									
Recall	1.0000	1.0000	1.0000	0.1846	0.9815	0.9671	0.3135	1.0000	0.8059
Precision	0.1660	0.4360	0.6700	0.2069	0.1513	0.5782	0.3124	0.29000	0.3513

Table 3: Results: GraphSAGE

8 Qualitative Analysis

In fig 7, we plot t-SNE on last hidden-layer activations of a few training examples in GraphSAGE for the best model observed. Comparing this figure to 4b, we can see that even though we start from embeddings which are pretty mixed up, GraphSAGE tries to cluster the nodes that belong to same disease groups - we see that most of the green nodes (corresponding to group 2) are to the left and most of the orange nodes (corresponding to group 1) are towards the right of the plot. Further, nodes corresponding to group 7 try to maintain small clusters spread throughout the plot same as node2vec in fig 4b.

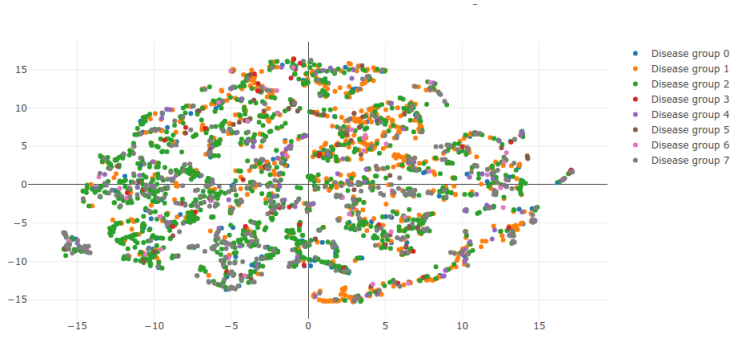


Figure 7: Activations at last hidden layer in GraphSAGE training

Additionally, it maybe noted that though we have very low for some classes, typically the ones that were the least represented during training, using balanced / weighted loss functions significantly improved performance overall, as well as for these under-represented classes.

9 Conclusion and Future Work

For all our models, we trained using vanilla unweighted loss functions without compensating for skewed outcome class representations, and also re-ran the same weighing the loss function to allow for under-represented classes to also influence the learning process.

Starting with the logistic regression baseline, operated on the Node2Vec features, we can see an improvement across both precision and recall measures when moving from the vanilla loss function to the weighted (balanced) version. It is easy to see how we have managed to considerably increase recall scores of some classes from almost 0 to as high as 0.53.

We also observe a similar improvement in moving to a balanced loss for GCN and GraphSAGE, where we are able to increase recall significantly, without compromising in the overall precision metric. It is interesting to note that GCN, uses a softmax non-linearity at the final output layer, and therefore predicting only a single label for a disease-group associated with a protein node, while GraphSAGE, with its sigmoid activation is able to appropriately threshold and output multi-class labels, which we see outperforming GCN when using node2vec features.

In future we would want to explore network motifs. Literature has shown a characteristic of PPI network structures around disease proteins,, indicating that disease proteins display significance in terms of the orbit positions they tend to inhabit. Therefore, we plan to investigate these network motifs and incorporate these motifs into our current feature set. Additonally, we would want to explore different multi-task learning strategies to be able to better predict the number of disease classes associated with each protein in a multi-class setting.

References

[1] Hamilton, William L., Rex Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs." arXiv preprint arXiv:1706.02216 (2017).

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

[3] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2016.

[4] https://github.com/aditya-grover/node2vec

[5] Agrawal, Monica, Marinka Zitnik, and Jure Leskovec. "Large-Scale Analysis of Disease Pathways in the Human Interactome." bioRxiv (2017): 189787.

[6] Menche, J., Sharma, A., Kitsak, M., Ghiassian, S. D., Vidal, M., Loscalzo, J., & Barabási, A. L. (2015). Uncovering disease-disease relationships through the incomplete interactome. Science, 347(6224), 1257601.

[7] Guney, Emre, Jörg Menche, Marc Vidal, and Albert-László Barábasi. "Network-based in silico drug efficacy screening." Nature communications 7 (2016): 10331.

[8] Cui, Qing, et al. "Knet: A general framework for learning word embedding using morphological knowledge." ACM Transactions on Information Systems (TOIS) 34.1 (2015): 4.

[9] J. Pinero et al., Database 2015 (2015).

[10] https://github.com/tkipf/gcn