Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Predicting Gene-Disease Associations Using Heterogeneous Graph Neural Networks

Scientific Students' Association Report

Author:

Balázs Róbert Glávits

Advisor:

Dr. András Gézsi

2022

Contents

Kivonat

Gének és betegségek közti kapcsolatok (gene-disease association, GDA) ismerete felhasználható a betegségek mögött rejlő mechanizmusok mélyebb megértéséhez, ez által végső soron hozzájárulhat új fajta diagnosztikai, prevenciós és kezelési módszerek fejlesztéséhez. Azonban az ismereteink koránt sem teljesek, a valódi gén-betegség asszociációk nagy része feltehetően még felfedezetlen. Nehézséget jelent, hogy az új, hipotetikus asszociációk kísérletes igazolása költséges és időigényes feladat. Hasznos lehet egy módszer, ami képes várhatóan valóságos új asszociációkat javasolni, így lecsökkentve a negatív eredményű kísérletekre szánt erőforrásokat.

Az orvostudomány által felhalmozott genomikai mérési eredmények, illetve kollektív tudás messze túlnőtte az egyén által könnyen megérthető méretet, ezért az elmúlt években a gépi tanulás-alapú módszerek kerültek az orvosbiológiai elemzések középpontjába. Ilyen módszer a gráf neurális hálózat, mely alkalmas félig felügyelt módon megtanulni és jósolni entitás párok közti releváns kapcsolatok jelenlétét vagy hiányát.

Dolgozatomban különböző biológiai adatbázisokat integráltam egy heterogén tudáshálózattá, melyben két típust képez a gének és a betegségek halmaza. Megvizsgáltam és kiértékeltem különböző architektúrájú gráf neurális hálózatok teljesítményét a tudáshálózatomban található gén-betegség asszociációk prediktálásában.

A kapcsolódó forráskód elérhető a következő webhelyen: `https://github.com/GlavitsBalazs/GeneDiseaseGNN`.

Abstract

Knowledge of gene-disease associations (GDAs) is helpful in further understanding the mechanisms that underly diseases and syndromes. This understanding can, in turn, aid in developing novel methods for diagnosis, prevention, or treatment. However, the picture is far from complete. Supposedly, the majority of true GDAs are yet undiscovered. The fact that experimental verification of new hypothetical GDAs is a resource and time-intensive task hinders discovery. Perhaps a way of predicting GDAs that likely exist could assist in reducing resources expended on experiments with a negative outcome.

The size of genomic measurement results, along with the collective body of knowledge amassed by the entire field of medicine, has far outgrown the scale which is easily understandable by the individual. For this reason, machine learning-based methods have become central in recent biomedical analyses. One such method is the graph neural network. This model can learn from examples and so predict whether or not relevant relationships exist between pairs of entities.

In my work, I've integrated various biomedical databases to form a heterogeneous knowledge graph, where the sets of genes and diseases are two distinct types. I then used this data to train and evaluate graph neural networks with different architectures for the task of GDA prediction.

The source code is available at `https://github.com/GlavitsBalazs/GeneDiseaseGNN`.

Chapter 1

Introduction

Let me present a hypothesis. Suppose, in a simplified world view, that the human body is a complicated machine. The fundamental building blocks of every mechanism in this machine are proteins. Sometimes, when a person falls ill the disease is related to these mechanisms failing. What do the proteins that took part in this failure look like? Answers to this question could lead to better understanding of the disease. Through further research, even to better methods of diagnosis, prevention, or treatment.

Changes to protein structures are often due to variations in the genes that encode them. Perhaps a route to discovery is to find the genes where variations are associated with susceptibility to disease.[32] How to find such gene-disease associations? Experimentally, of course, but there's a limit to that. Only so much experimentation can be done with the time and resource constraints of real life.

The next best thing is inference. To look at previous results and try to identify patterns in them to predict new ones. Such a prediction of course doesn't constitute any sort of proof, more so a suggestion. Still, a potentially valuable one if chosen appropriately out of the large space of possibilities.

With the goal in mind, I'll briefly outline my plan on how to accomplish such a thing. Imagine what a giant knowledge base of gene-disease associations would look like in an abstract way. One possible picture is of a network, like in figure 1.1. The genes and diseases are nodes and associations between them are links. This view is advantageous because it becomes possible to talk about the problem in a mathematical way using the tools available from graph theory. There's also extensibility, to incorporate more information into the model. Drawing connections between the genes and the diseases. To add edge weights describing the strengths of associations. Attributes can be attached to the nodes. Even more is possible.

1.1 Graph Neural Networks

A great tool for learning patterns in networks as described previously, is the graph neural network (GNN)[44], which are artificial neural networks that include message passing (also known as graph convolution, or neighborhood aggregation) operations.

I'll give a definition for the message passing operation, but first it has to be said how the graph gets encoded as input to neural networks. For a $G = (V, E), E \subseteq V \times V$ directed graph both nodes and edges have to be given numberings (in an arbitrary manner). An edge list is constructed in the form of a $2 \times |E|$ matrix where the j-th column corresponds

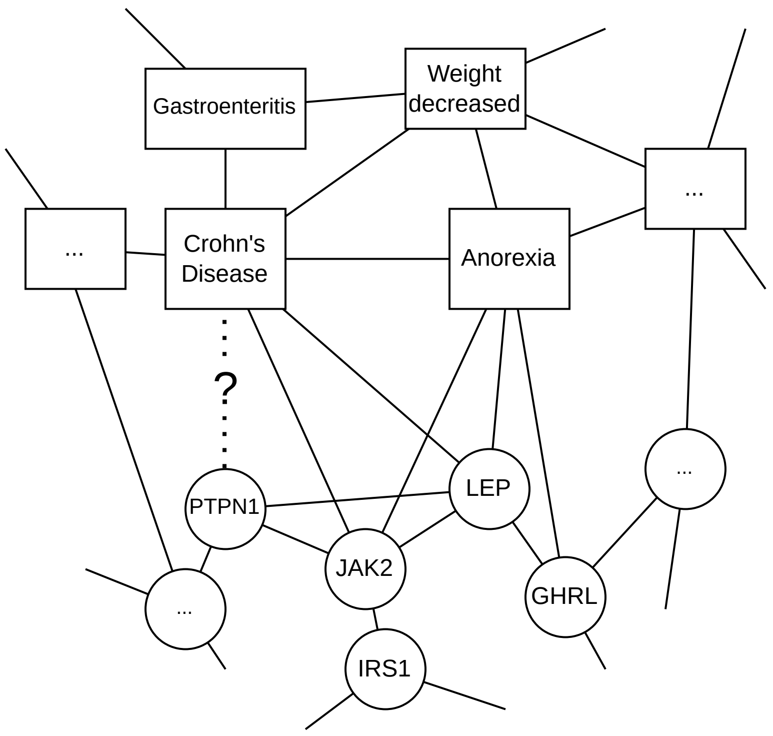

Figure 1.1: Illustration of the gene-phenotype heterogeneous network. Round nodes denote genes. Rectangle nodes show phenotypes such as diseases, symptoms or categories of diseases. The line with the question mark is a supposed, yet unknown association.

to the j-th edge. Each column consists of two node numbers, where the first corresponds to the starting point and the second the end point of an edge. Then node features are organized in a $|V| \times n$ matrix. The i-th row of which is the feature vector of the i-th node. A similar construction can be made to store edge feature data (with only one dimension if the feature is just a weight).

So a message passing operation is of the form

$$\mathbf{x}_i' = \text{Update}\Big(\mathbf{x}_i\,, \text{Aggregate}(\{\text{Message}_{j\to i}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ji})\,|\, j \in N(i)\})\Big)\,,$$

where $\mathbf{x}_i$,$\mathbf{x}_j$ are the input features of the i-th and j-th nodes, e_{ji} is the edge feature of the directed edge from j to i. The graph structure is also considered as an input and appears here as the neighbor function $N(i)$, returning the set of neighbors of the i-th node. Message and Update can be any function that take and return vectors of appropriate shape. In practice these are neural networks possessing arrays of learnable parameters. Aggregate is similarly a function of vectors, but this one is typically not a neural network, and has an additional important constraint that it must be permutation invariant: it must return the same thing regardless of the order of its arguments. This is to mimic the key property of graphs that there is no preferred ordering of neighbors of a node. Common choices for Aggregate are sum, average or max.

So the output of all this is x_i', the embedding of the i-th node. With chaining more following neural layers (so called prediction heads), these embeddings can be molded during training to work in a wide variety of applications.

Perhaps this is somewhat too general in practice. Let's see a simpler, concrete example: GCN[15]. Set $\text{Message}_{j\to i} = e_{ji}\mathbf{W}\mathbf{x}_j$, $\text{Update}(\mathbf{x}_i, \mathbf{y}) = \mathbf{x}_i + \mathbf{y}/\sum_{j\in N(i)} e_{ji}$ and Aggregate is a simple sum. $\mathbf{W}$ is a trainable weight matrix, and e_{ji} is an edge weight (default value 1).

How to use graph neural networks for predicting gene-disease associations? Each association is an edge, and GNNs can learn which edges are present in the graph. Suppose we have a GNN with many layers of message passing operations, activation functions, linear layers or anything else necessary. Remember that it takes as input the graph itself and outputs node embeddings. Define a "decoder" operation that takes two of these embeddings and produces a predicted probability of an edge being present. Often times simply an inner product followed by a sigmoid activation works well: $P((i,j) \in E) \approx \sigma(\mathbf{x}_i^T \mathbf{x}_j) = p_{ij}$.

Training this GNN has to be done in a semi-supervised (self-supervised) manner. There are no ground truth labels available telling us whether an edge should be present or not. We only have the graph itself, so we must learn from that. For this a special kind of dataset split is needed. To use the terminology of Jure Leskovec, a transductive link prediction split. The graph's edges must be split in four parts: training message passing edges, training supervision edges, validation edges, and testing edges. At training time the graph with only training message passing edges is shown to the GNN and it's asked to predict scores for every training supervision edge. Over time it may learn the patterns of the training supervision edges too so during validation and testing care must be taken to avoid data leakage. When validating enter as input the union of train message passing and train supervision edges and query the validation edges. At testing time train message passing, train supervision and validation edges may all be shown to the GNN to finally predict the test edges.

In each training epoch we have have our appropriate message passing graphs and also positive supervision edges, but negative supervision edges are too required. This to make the cross entropy loss work:

$$\mathcal{L}(\mathbf{p}) = -\sum \log(\mathbf{p}_{pos}) - \sum \log(1 - \mathbf{p}_{neg}).$$

We're telling the GNN to output 1 for the positive supervision edges and 0 for negatives, so which edged should be the negatives? Simply put anything that's not in the graph. There are in practice much more edges that aren't in the graph than that are so a balance must be reached. One approach is to randomly generate such edges, with the number of them equal to the positives. The simplest option to do so is by starting from two independent uniform distributions of nodes and rejecting any node pairs that happen to be positive.

1.1.1 Heterogeneous graph neural networks

I've shown that it's theoretically possible to predict gene-disease associations with GNNs. However there is a practical opportunity for improvement left out of the discussion. The knowledge graph as described previously (see fig.1.1) has additional structure, namely that it has two entirely different kinds of nodes. A simple GNN doesn't even get to know which is which. Not to mention the fact that the edge structure between the varying types of nodes could be drastically different as well. Not just statistically but semantically.

Heterogeneous graph neural networks aim to benefit from such additional structure. To be formal, a heterogeneous (or sometimes multimodal or relational) graph[45][33] contains types assigned to every edge and node, with $\phi : V \to \mathcal{T}$ and $\psi : E \to \mathcal{R}$ label function to $\mathcal{T}, \mathcal{R}$ sets of types. For a given type, the subset of nodes belonging to it can be described as $V_i = \{v \in V : \phi(v) = i\}, i \in \mathcal{T}$, as can subsets of edges $E_r = \{e \in E : \psi(e) = r\}, r \in \mathcal{R}$. Each $(i, j, r) \in \mathcal{T} \times \mathcal{T} \times \mathcal{R}$ triple designates a subgraph, such that $G_{ijr} = (V_i \cup V_j, (u, v) \in E_r : \phi(u) = i \wedge \phi(v) = j)$.

A heterogeneous message passing operation can be defined as composing individual, "homogeneous" message passing operations, each having a type triple and acting on its designated subgraph. The outputs are then aggregated appropriately

$$\mathbf{x}'_n = \text{Aggregate}(\{\text{MessagePassing}_{(ijr)}(G_{ijr}) \,|\, i = \phi(n), j \in \mathcal{T}, r \in \mathcal{R}\}),$$

where $\text{MessagePassing}_{(ijr)}$ is the message passing operation assigned to the type triple $(i, j, r) \in \mathcal{T} \times \mathcal{T} \times \mathcal{R}$. n is the target node, for which the embedding x'_n is computed.

1.2 Previous methods

One of the first projects investigating edge predicting heterogeneous GNNs was Zitnik et al.'s Decagon[46]. They applied Schlichtkrull et al.'s architecture[30] on a knowledge graph of genes and drugs, to predict unknown side effects of polypharmacy. Following shortly after [20]'s PGCNLi et al. forked Decagon's source code for gene-disease prediction. PGCN's authors have developed a heterogeneous graph of genes and diseases based on OMIM[11] and HumanNet v1[17]. PGCN has served as a great inspiration for my work and I hope to improve upon it. Another group of researchers[31] have published developments recently, but they've relied on the original knowledge graph of PGCN. I intend to construct my own dataset that is larger and more up to date.

Another early work using GNNs for GDA prediction was Rao et al.'s GCAS[28]. Their dataset (titled HANRD) was composed from HPO, MeSH, Wiki Pathways, and Orphanet. The latter choice made it particularly specialized for rare diseases. Again this graph would be considered outdated today due to its small size.

Other research projects of interest include HGCNMDA[18], HPOFiller[23], HNEEM[39], and GCN-MF[12], each having unique specializations and algorithms.

The most recent research projects rely on DisGeNet as their main data source. As does PDGNet [42], which utilized a multi-view learning framework rather than a heterogeneous GNN. geneDRAGNN[2] is solving a more specialized task than the methods previously discussed. Rather than predicting associations for any disease, they specifically target lung adenocarcinoma, allowing them to use a node classification type of approach rather than an edge prediction based one.

Finally I'd like to mention an algorithm belonging in an entirely separate class of algorithms: RWRH[19] first published in 2010. This method is not a neural network but rather type a network propagation[7]. Recent studies[37] have applied it on more up to date datasets (the original was based on OMIM, Valdeolivas et al. used DisGeNet v4) and found considerable success.

Chapter 2

Materials and methods

2.1 Database integration

Here I'll describe may way of constructing a large heterogeneous knowledge graph of genes and diseases, which is suitable for training graph neural networks.

The state of the art in cataloging gene-disease associations is DisGeNet v7.0[26]. Over a million associations are recorded between 20000+ genes and 30000+ diseases or phenotypes. Its large size and clean labeling make it a natural choice for training neural networks. Not just diseases but symptoms, categories of diseases, or even body parts are present as well. In general I'll refer to such things as phenotypes, with the connotation that these phenotypes are relevant in discussion of diseases, by merit of being included in DisGeNet.

There is just one downside that comes with such size, not all of it is equally reliable. The majority of the data comes from text mining sources such as BeFree[25], which were generated algorithmically and not always validated by experts. For this reason I chose to partition DisGeNet into three qualities by the score. I've filtered out any phenotypes with insufficient evidence: if the sum of all scores remains below 0.6. Even this low threshold leaves only 8070 terms out of 30293.

In DisGeNet each association is annotated by a numeric score, which in essence describes the reliability of its evidence. I'll call high quality any association with score > 0.1, meaning that it either comes from a manually curated source or more than ten papers mention it. It should be mentioned that this notion of high quality may not be high enough for other kinds of applications. Associations that are only supported by text mining evidence, but more than more one, I'll deem medium quality. This is encoded by scores that are < 0.1 and > 0.01. Finally, score $= 0.01$ associations are only mentioned in one text mined publication, therefore should be treated as low quality.

For connections between genes I've adopted the STRING database[35] v11.5. Except, of course STRING is a network between proteins not genes. In my application these will be treated as equivalents. I'll assume that whenever a phenotype is associated with a gene it is also associated with any coded proteins. Ensembl BioMart[14] was used to translate between the HGNC gene IDs in DisGeNet to the Ensembl version 77 protein IDs (or ENSPs for short) in use by STRING. All 5071682 records were inserted as weighted directed[1] edges in my graph verbatim. Though most of them (85%) have low scores < 0.4.

DisGeNet identifies diseases by their UMLS Metathesaurus[4] version 2019AA concept unique identifiers (CUIs). The Metathesaurus authors have created a database which subsumes several other biomedical thesauruses (collections of terms and their definitions), linking equivalent or synonymous spellings of terms to appropriate abstract concepts. Relationships between concepts have been adopted too. I'll draw from these to form my phenotype-phenotype network. Some source vocabularies, such as OMIM[11] and SNOMED CT[8] include relationships information such as disease symptoms and comorbidity. It turns out that a large number of interesting comorbidity relationships were adopted from an unexpected lesser known source titled Clinically Useful Problem Statement Systems[5].

2.1.1 Gene annotation

For the success of training a graph neural network it is critical to choose node feature vectors appropriately. After all, the only thing a graph convolution does is transformation of node feature vectors. Care should be taken that the vectors are not too similar, otherwise we run the risk of the neural network confusing different nodes for one another[41]. The dimensionalities of the vectors can't grow arbitrarily large either, because the graph convolutions entail linear transformations on these vectors. The number of trainable parameters in the linear transformations would grow quadratically with more added dimensions, which in turn may lead to overfitting and memorization.

Recall that the probability of an edge as predicted by the model depends on the inner product of the embedding of the end point nodes. Perhaps it would be a good idea to construct feature vectors with a similar property of having larger inner products for nearby nodes. It's not immediately possible to construct vectors that behave this way between genes and diseases – that's the end goal of the whole endeavor –, but inside the gene-gene network, we can attempt to. For a given $\mathbf{K}$ kernel on the graph, which is positive-definite and k_{ij} measures the similarity between the i-th and j-th node, the rows of its factorization $\mathbf{A}\mathbf{A}^T = \mathbf{K}$ would suffice.

STRING includes hierarchical clustering information[34], computed using HPC-CLUST[24]. Consider treating the dendrogram of this clustering as an ontology, where the objects are the proteins, the classes are the clusters, and every time a cluster or protein is included in another cluster draw a directed edge from it, representing an "inverse is a" relation. In this ontology it's possible to compute Resnik's information content based similarity[29] between any pair of proteins.

So here's the procedure in practice. Take the hierarchical clustering data from STRING and construct an ontology graph. Compute the intrinsic information content of each node, find the most informative common ancestor of each protein pair and construct a kernel matrix on the graph $\mathbf{K}'$. Technically this won't be a kernel yet because it might not be positive-definite. A quick fix for this issue is to find the smallest (negative) eigenvalue $\lambda_{\min}$ and subtract it from the diagonal: $\mathbf{K} = \mathbf{K}' - (\lambda_{\min} - \varepsilon)\mathbf{I}$, where $\varepsilon > 0$ is a small value needed to make it positive-definite and not just positive semi-definite. This step introduces a bias favoring self-similarity of nodes, so care must be taken not to rely on these measurements blindly. Now, this may be factored using Cholesky decomposition. One issue remains

[1]In STRING every protein-protein interaction is encoded as two directed edges between the proteins, one having orientation opposite of the other. This way essentially undirected relationships may be represented in a directed graph.

still, that the dimension of these vectors is too large. Of all the dimensionality reduction methods, truncated SVD is useful here, because $\|\mathbf{A}'\mathbf{A}'^T - \mathbf{K}\|$, where $\mathbf{A}'$ is the reduction of $\mathbf{A}$, remains small.

What are the strengths and the weaknesses of this approach? Most important is the fact that every node receives a proper feature vector, none are missing. It would be hard to find a database of genetic data that includes complete and consistent annotations for every gene. The numerical range of the data is not too large, mostly between -1 and 1. The mean is close to zero the standard deviation is 0.1, which make it easy for neural networks to learn from. A handy property of the truncated SVD dimensionality reduction is that it's possible to first compute, say, 1024 leading principal components. Afterwards, whenever lower dimensioned vectors are required, it suffices to throw away the last few items of these 1024.

The Resnik similarity kernel appears to contain information relevant for edge prediction inside the gene-gene graph. To demonstrate this I set up the following simple test scenario with conditions somewhat similar as those yet to be faced by the graph neural network. The inner product of feature vectors is computed for each pair of nodes with an edge between them, and also for an equal number of randomly sampled negative edges. For calibration purposes, a one dimensional logistic regression model is trained with 1 labels given for positive edges and 0 labels for negatives. The calibrated scores should now serve as predicted probabilities for discriminating the presences of edges. Indeed they do, showing $\approx 70\%$ average precision and receiver operator characteristic scores.

However, by definition, any kernel must describe similarities between all pairs of items in the dataset. This could pose an issue in inductive settings, where new nodes would have to be processed by the model which were never part of the training dataset. By choosing to use a kernel I constrain the applicability of this model to transductive settings only. Every time a never before seen new gene or phenotype should be evaluated, it is necessary to reconstruct the dataset and retrain the model all over again. This limitation is acceptable in the task at hand, simply because in practice we rarely expect to see such brand new genes or phenotypes.

Another open possibility is to extend the feature vectors further. With every node having a solid foundation of basic data describing it, additional dimensions may be concatenated that contain values coming from less reliable or less complete sources. Even if a gene happens to be missing from such a database, placing zeros or averages in its vector should not cause substantial problems because having the synthetically generated feature information present should orient it differently in latent space than other genes with information similarly missing. It's possible to avoid repeatedly annotating every missing gene with the same placeholder vector. However I opted not to explore these possibilities in the current experiments. I don't believe that altering the current values of the feature vectors could significantly improve the end results.

2.1.2 Disease annotation

Finding good vector representations for disease nodes poses a challenge. I've elected not to use the common way of previous studies: to describe diseases with their characteristic symptoms. This is because in my network symptoms are represented as nodes, just as diseases are. Valuable network structure and gene-symptom association data would be lost if symptoms were removed from the graph to become parts of vector representations.

Instead I'll use the same same approach as I did with the gene nodes. Finding an ontology of diseases, computing the Resnik similarity kernel then factoring it. The UMLS Metathesaurus again proves useful here. DisGeNet contains many phenotype terms which have no describing records anywhere but in the UMLS Metathesaurus. Even obscure terms are given at least a categorization in SNOMED CT[8] or in other ontologies.

So the goal is to construct an ontology (rooted directed acyclic graph, where edges are "inverse is a" relations) made of smaller ontologies found in the UMLS Metathesaurus, while covering as many DisGeNet phenotypes as possible. This becomes possible with the help of the UMLS Metathesaurus, but it is perhaps a bit too helpful. Many commonly used terms are contained in multiple ontologies (for example MeSH[21], HPO[16] and SNOMED CT) at the same time. This is a problem because wrong categorizations can appear between terms originating from different semantic frameworks.

Unfortunately there is no way to avoid introducing such incorrect relations when forcefully integrating so many different systems of categories. Though I did try in the following way. I took every single parent-child relationship in UMLS and placed them in different ontology graphs based on the source vocabulary from which it came, all while filtering out data that's unrelated to the set of terms contained in DisGeNet. So I found 76 individual ontologies of various sources that contained some DisGeNet terms. Each one should be consistent, meaning that no foreign terms or relations were added to them by me, potentially causing corruption. Now to assembling them, I went with a greedy approach inspired by Chvatal[6]. Always choosing the largest UMLS ontology to add to my union of ontologies, then removing its edges from all others and re-measuring the size after removal. I hope that this way fewer different source ontologies and fewer cross edges between them were incorporated. Afterwards only a minor pruning is required to make the graph acyclic.

This automated method of constructing a large disease ontology certainly has its flaws. Just a quick glance at the results shows many inconsistencies, mislabelings, and sometimes too much branching. Perfection is not the goal however. The fact that similar diseases are placed somewhat closely together leads to a good enough approximation. The neural network can learn to work around any possible issues here.

Now with an ontology in hand the feature vectors can be computed as discussed. Some terms in DisGeNet unfortunately had no relationships records in UMLS so were discarded. The situation is not a total loss though. Remember that we now have an ontology full of relevant terms and interesting connections. The Resnik kernel can include these as well, thereby giving them feature vectors. So I've added all this to the disease-disease graph, naturally without any associations to genes.

2.1.3 Summary

Finally, the heterogeneous network includes 63906 edges connecting 9399 phenotypes. Though these edges can potentially mean 209 different classes of things based on their relationship type attribute from the UMLS Metathesaurus. [2] 19385 genes were taken from STRING with 5969249 edges between them. DisGeNet had 449945 records of associations between 5880 of these genes and 7914 of the phenotypes. 142827, 100900 and 206218 of these are high, medium, and low quality respectively. The dimensionality of the feature vectors can be any number desired by discarding a few principal components.

[2]A missed opportunity in development was to encode these types as edge feature vectors.

2.2 Architecture

Here are some details about the neural network architecture I've implemented using the PyTorch Geometric framework[9].

Figure 2.1 illustrates the anatomy of a single heterogeneous layer in. GCNConv[15] was chosen for the gene-gene network, because it naturally handles graphs that have edge weights, as does STRING db. SAGEConv[10] works well on bipartite graphs. Contrary to the gene graph, the edges in the disease-disease graph have variable semantics and there is no available way of assigning weights to them in a consistent manner. Therefore by using GATConv[38] I give the neural network a chance to learn by itself which edges have more importance than others.

The layers use dropout, batch normalization as regularization and PReLU[13] activation.

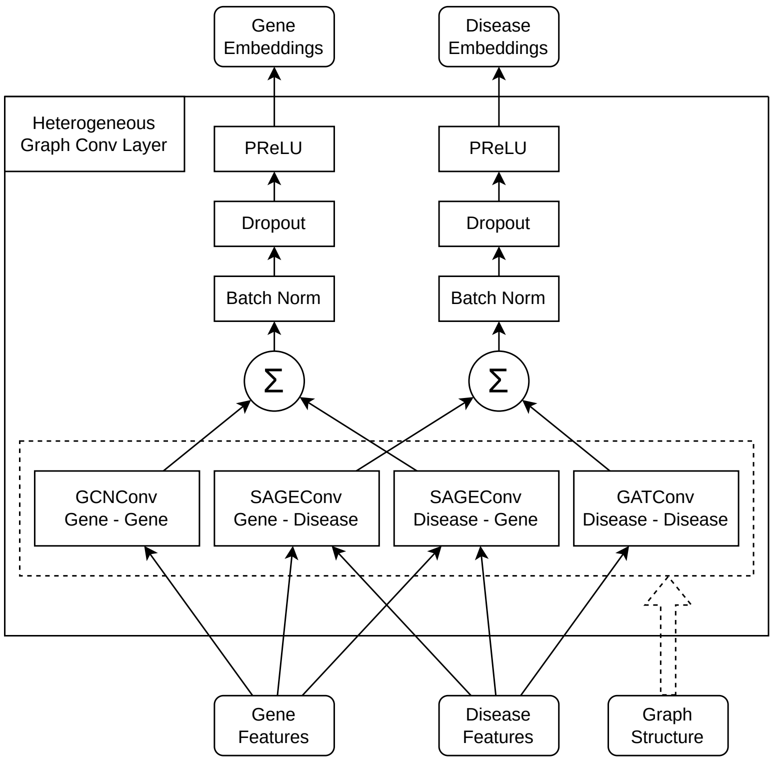

Figure 2.1: Heterogeneous graph convolution layer.

In 2.2 The decoder takes a list of target edges and all the node embeddings as input. For each target edge it "decodes" the corresponding pair of node embeddings into a probability. In this case a simple inner product followed by a sigmoid activation suffices.

2.2.1 Training

The model was trained using the Adam optimizer for 76 epochs. I have not employed any mini-batching mechanism, as the entire dataset comfortably fits in memory. In each epoch the entire train message passing dataset and the entire train supervision edge set is passed to the model along with a new set of randomly sampled negative supervision edge set.

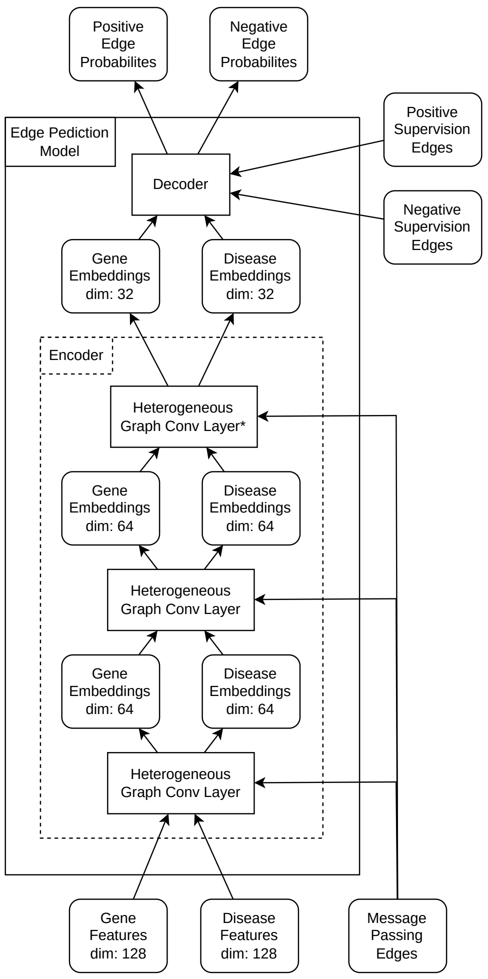
Positive Edge Probabilites
Negative Edge Probabilites
Edge Pediction Model
Positive Supervision Edges
Decoder
Negative Supervision Edges
Gene Embeddings dim: 32
Disease Embeddings dim: 32
Encoder
Heterogeneous Graph Conv Layer*
Gene Embeddings dim: 64
Disease Embeddings dim: 64
Heterogeneous Graph Conv Layer
Gene Embeddings dim: 64
Disease Embeddings dim: 64
Heterogeneous Graph Conv Layer
Gene Features dim: 128
Disease Features dim: 128
Message Passing Edges

Figure 2.2: Edge prediction model. Note: the last convolution layer marked with an asterisk (*) has had its regularization (dropout, batch norm) and activation operations omitted.

2.2.2 Hyperparameter optimization

Inspired by the work of You et al.[43] I explored various choices for altering the neural network architecture and training. In summary: the number of layers, embedding dimensions of each layer, the inclusion of linear layers between graph convolutions, the choice of regularization, and others. I have not experimented with any large models deeper than three layers or using embedding dimensions higher than 256 due to rising computational costs and no apparent signs pointing to better performance in this area. My tool of choice for carrying out these experiments was the Optuna framework[1]. The current best architecture was decided upon after running ≈ 200 trials.

Chapter 3

Results

From now on I'll treat each method as a black box. A gene-disease association predictor can be any algorithm that takes some biological databases as input and outputs a matrix of real numbers. Each row corresponds to a gene, and each column to a disease. Higher values indicate a higher likelihood of association being present, but I'm not assuming that the numbers would form a valid probability distribution.

The methods compared are the following:

- GNN: As described in section 2.2.
- MLP: It's a simple multi-layer perceptron that takes a pair of feature vectors concatenated as input, has one hidden layer of dimension 64 and outputs the predicted probability of an edge being present. Trained the exact same way as was the GNN.
- RWRH: Running the random walk with restart algorithm[19][37] on the heterogeneous graph with parameters $\lambda = 0.5, \alpha = 0.7$. Each gene-phenotype association score is the stationary probability received by the gene when the network propagation is started from the phenotype.
- Uniform random baseline: The prediction for each gene-phenotype association is a random number sampled from independent uniform distributions on $[0, 1]$.
- PGCN: See section 3.4.

3.1 Binary evaluation

Here I'll assume that each prediction in the result matrix is an independent binary classifier. Every association in DisGeNet is treated as a positive sample. Negative samples are generated in a uniformly random manner, same way as in 2.2.1. The number of negatives is chosen to equal the positives. These assumptions mirror those used in the neural network training process.

The metrics I chose to use are the following:

- ROC: Receiver Operator Characteristic.
- AveP: Average Precision, or equivalently, Area Under the Precision-Recall Curve (AUPRC).

- BEDROC: Boltzmann Enhanced Descrimination of the ROC, introduced by Truchon and Bayly[36], used by Li et al.
- AP@K: Average precision at the top $K = 200$ predictions.
- BinROC: Binarized ROC. Every prediction where the score is > 0.5 gets replaced by 1 or 0 otherwise before computing the ROC.
- BinAveP: Average precision similarly binarized.

The formula for average precision at the top K is the following[22]:

$$AP@K = \frac{1}{min(|R|, K)} \sum_{i=1}^{K} \delta(i \in R) \frac{|\{r \in R : r \leq i\}|}{i},$$

where R is the set of ranks of true positive items. $\delta(i \in R) = 1$, if the i-th predicted item is positive, 0 otherwise.

The findings of my tests are summarized in table 3.1.

The total number of positive samples in the dataset is 100900 for medium and 206218 for low quality. In the case of the 142827 high quality edges originally present in the graph, 90% was used as training data, leaving only 14283 for this test. Not shown is the fact that RWRH produces $\approx$ 99% ROC and AveP on high quality DisGeNet data because those edges were part of the propagation graph. Note that the baseline (as produced by an uniform random classifier) for these metrics is 0.5, except for the case of AP@200, where it is approximately 0.25.

Applying such a binary evaluation framework for RWRH is perhaps an objectionable choice. The main cause for concern is that the individual probabilities assigned to each gene-phenotype association are not independent of each other, rather, they sum up to a probability distribution. They will almost never exceed 0.5, explaining the complete lack of binarized classification performance. An unfair advantage is given to RWRH over the neural networks when evaluated under the ROC and AveP metrics.

In conclusion, these results show that the GNN model was successfully able to learn the patterns in high quality DisGeNet data and generalize this knowledge to the remaining samples with high quantity but lower quality. The MLP results indicate that the node feature vectors were adequately chosen during database construction, they do hold information that's relevant for solving the task at hand, but utilizing graph structure remains necessary.

However, seeing the excellent performance of the much simpler RWRH algorithm suggests a flaw in this binary approach. Perhaps it is less important or even counterproductive to discriminate rather than prioritize potential gene-phenotype associations.

	DisGeNet quality	ROC	AveP	BEDROC	AP@200	BinROC	BinAveP
GNN	high	0.976	0.967	0.987	0.983	0.936	0.899
	medium	0.953	0.934	0.954	0.751	0.894	0.858
	low	0.922	0.893	0.917	0.694	0.831	0.796
MLP	high	0.720	0.699	0.803	0.666	0.661	0.606
	medium	0.712	0.698	0.807	0.889	0.653	0.600
	low	0.668	0.656	0.757	0.800	0.622	0.576
RWRH	medium	0.909	0.901	0.967	0.987	0.500	0.500
	low	0.865	0.848	0.927	0.913	0.500	0.500

Table 3.1: Binary evaluation results.

3.2 Ranked evaluation

Using a different approach to testing, I'll consider each phenotype independently, taking the corresponding column from the prediction matrix as the list of its candidate genes. Again relying on DisGeNet as the source of truth, it gives a subset of these genes that are positive. The question I'm asking now is not whether those genes are predicted or not, but rather what their rank is in the ordered list of candidates.

In this application DisGeNet seems somewhat sparser. Out of 9393 only 3847 phenotypes had any matching high quality associations, likewise 3243 had medium, and 4485 had low quality matches. The average number of genes matched per phenotype is 19, 31, and 46 for the different qualities. So overall not too sparse, but maybe more matches would be expected based on the advertised number of a million plus total associations in the database.

With the rankings of the positive associations in hand, it's time to take measurements. The metrics I chose were the AveP and AP@K, which it turns out, are applicable both for binary and ranked types of classification. The mean reciprocal rank (MRR) was also measured.

My results are summarized in Figure 3.1, which shows the distribution of ranked evaluation measurements over all possible phenotypes. It seems that the neural networks pay little attention to the ordering of their positive results. Not surprising, since that's not part of the loss function. By a coincidence GNN slightly outperforms MLP but both pale in comparison to RWRH.

The baseline value for AveP was ≈ 0.01. AP@200 and MRR were ≈ 0.001, as found by measuring the performance of the uniform random classifier and taking the 95-th percentile (regardless of quality).

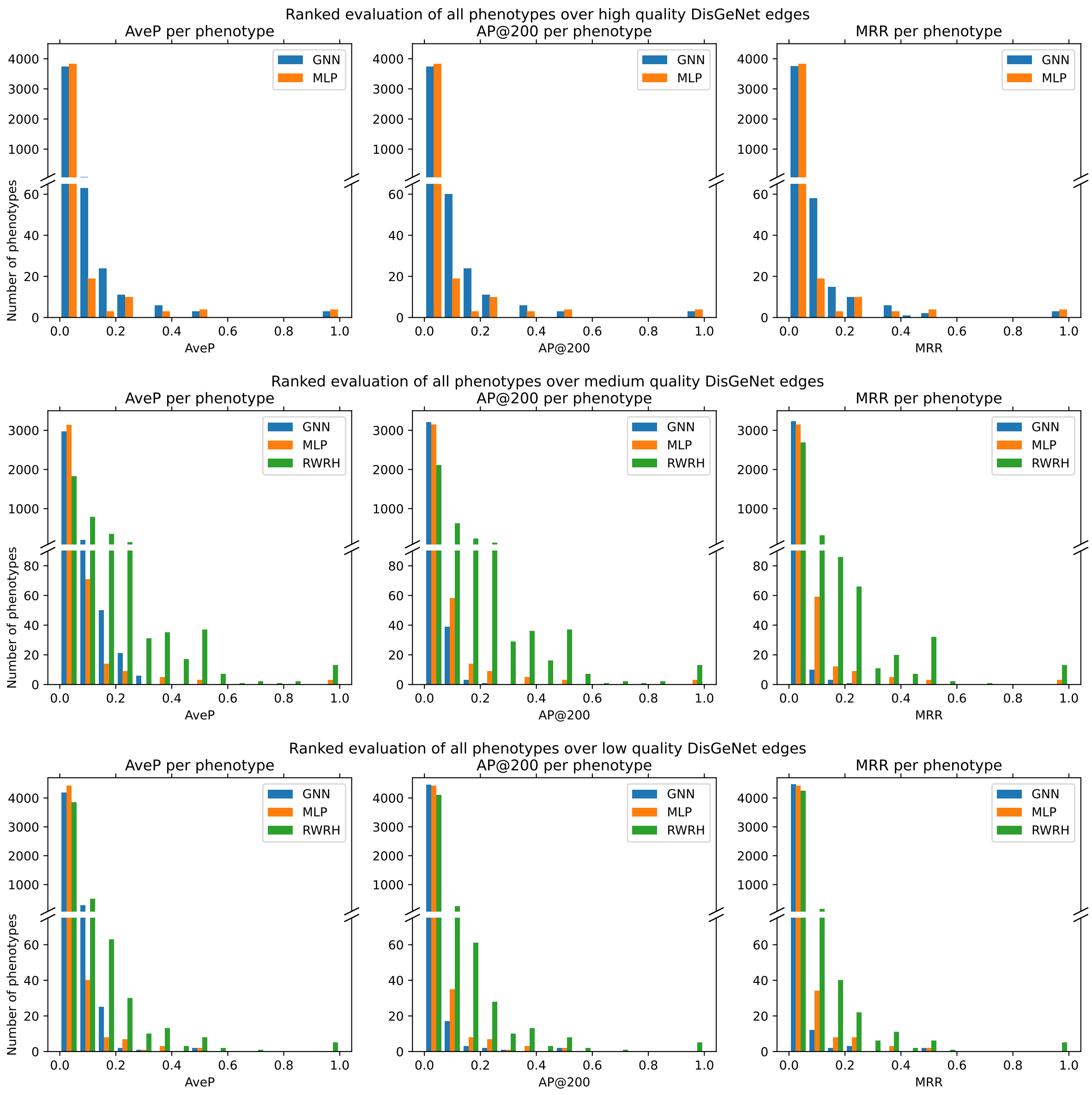

Figure 3.1

3.3 Comparison with GWA studies

So far, in my experiments, I've only used DisGeNet as the single source of gene-disease associations. While, of course, DisGeNet itself composes a wide range of sources, there remains an insufficiency, namely that it only contains a few associations per phenotype. There is no information of how a given phenotype relates to the remaining 19000 genes for which no records are present. Even knowledge of supposed negative associations would be helpful.

I turn to genome-wide association studies (GWAS) as the solution. Such a study measures the genetic variants throughout the whole genome over a population exhibiting a given phenotype. Based on this data, the study presents hypothesized links between genes and the phenotype, but negative results, genes from which no association is provable, are recorded as well.

GWAS Atlas[40] is a database cataloging over 4000 GWA studies. Importantly, it is independent from DisGeNet so an ideal choice for external validation. The experimental results are presented in the following form. The name of the phenotype or trait is given in English, along with a list of genes by their Ensembl IDs considered in the study. The strengths of the associations between the phenotype and the genes are quantified as p-values.

I chose not to treat these p-values in the conventional manner of evidences for rejecting null hypotheses. That would give me a list of positive associations, which I already have enough of from DisGeNet. Rather, I assume that lower p-value genes have a higher likelihood of association than ones with higher p-values, therefore an ordering of genes is formed. So my method for testing is to match phenotypes in my database to GWA studies and compare the predicted ordering of genes to those presented in the GWAS.

There stands the practical issue of matching the phenotypes in my dataset – which come from DisGeNet and are defined by UMLS CUIs – to the GWA Studies, that have only English language textual descriptions. The basis for solving this is to look in the UMLS Metathesaurus records, which list all canonical ways of naming a given concept, including different spellings, variations, or synonyms. Unfortunately the GWAS authors rarely use these UMLS accepted exact phrasings as their phenotype descriptors, but sometimes they do come close. So to account for inexact, approximate matches between GWAS descriptions and UMLS terms, I opted to use the 2-gram occurrence vector cosine similarity metric to find nearly equivalent pairs of strings.

After normalizing for special characters, every occurrence of every pair of letters is counted in the phrase to form an occurrence vector. Occurrence vectors of all the GWAS trait names can be stored in a sparse matrix with each column normalized. I then ran through the database of UMLS terms, computing the normed 2-gram occurrence vector and multiplying it with the sparse matrix, resulting in the cosine similarities of a term to each GWAS name. I accepted any match with score ≥ 0.8, which is empirically quite a low threshold, admitting many false matches. These wrong matches however should not cause any problems, because down the line it will become evident that the genetic backgrounds of two such traits are unrelated.

It is now possible to take an ordered list of gene predictions for a phenotype and compare it to a GWAS from GWAS Atlas. The method of comparison I went with was Spearman's rank correlation. If the genes are numbered $1, 2, 3 \ldots$ based on their ascending order of p-values as reported by the GWAS, then this is reduced to the empirical Pearson correlation coefficient between the list of ranks of the predicted genes and the list $1, 2, 3 \ldots$ The significance of such a correlation can be validated using Student's t-test[27]. With all matches tested for significance, I've used Benjamini and Hochberg's procedure[3] to select positives, targeting an expected false discovery rate of 5%.

Table 3.2 shows the summary of matching gene predictions to GWA studies. The first row includes the number of matches found by the cosine similarity matching strategy. The low fraction of GWA studies found (out of 4756) is not surprising because GWAS Atlas does not specialize in disease related studies. However for studied diseases there are often multiple instances of GWA studies present, explaining the even lower number of phenotypes found. Note that the baseline random classifier found no significant correlations between any GWAS and phenotype prediction.

The results shed light on the fact that RWRH is superior in terms of the ordering of results over the neural networks. The GNN correctly predicts important gene-disease associations but it may give them lower scores than to less important ones.

	Matches	Unique GWAS	Unique Phenotypes
All Potential	3737	1494	712
GNN	321	170	159
MLP	41	24	32
RWRH	874	376	343

Table 3.2: Comparison with GWA studies.

3.4 Comparison with PGCN

As I've based my GNN on similar principles to PGCN[20], it's possible to compare the two methods on the basis that both of them give as output a matrix of scores between genes and phenotypes. The exact genes and phenotypes differ somewhat, but a translation can be made. Originally PGCN operates with phenotype IDs from OMIM and NCBI Entrez gene IDs. So I've utilize Ensembl BioMart to translate HGNC gene IDs of DisGeNet to NCBI. The UMLS Metathesaurus entails mappings between OMIM phenotypes and its own CUIs.

Having cross-compatibility between the training set of PGCN and DisGeNet I've proceeded to validate both neural networks with both datasets using the binary evaluation method as described in section 3.1. The results in table 3.3 show nearly complete independence. The neural networks perform well on their own training sets, but fail to generalize out-of-distribution. Though in this case preference has to be given to my method for demonstrating adequate prediction ability on a dataset of size on the order of a few hundreds of thousands instead of just a few thousands.

The disparity in edge counts between methods is an artifact caused by the many-to-many nature of mapping between the different ID schemes of genes and phenotypes. For any record in DisGeNet that had a matching prediction in PGCN there may be multiple synonyms in my network. Some edges in the PGCN training network aren't included in mine.

		PGCN Training Set	DisGeNet High Quality
PGCN	Edge Count	3954	12014
	AUROC	0.968*	0.456
	AUPRC	0.973*	0.479
	AP@200	1.000*	0.282
	BEDROC	1.000*	0.508
My Method	Edge Count	2485	19105
	AUROC	0.681	0.966*
	AUPRC	0.555	0.972*
	AP@200	0.522	0.994*
	BEDROC	0.643	0.989*

		DisGeNet Medium Quality	DisGeNet Low Quality
PGCN	Edge Count	16734	31013
	AUROC	0.464	0.458
	AUPRC	0.483	0.480
	AP@200	0.228	0.236
	BEDROC	0.509	0.506
My Method	Edge Count	29854	49382
	AUROC	0.944	0.911
	AUPRC	0.957	0.923
	AP@200	0.934	0.821
	BEDROC	0.978	0.950

Table 3.3: Binary evaluation results of PGCN compared with my method using different data sets. Note: numbers marked with an asterisk (*) are taken from tests which include training data.

Chapter 4

Conclusion

Instead of relying on evaluation metrics, just take a look at the output of the GNN. Two things immediately become obvious. First, that it keeps repeating the same genes over and over again as top candidates for pretty much every disease. This is explained by the binary nature of the training of this GNN. Yes those genes may indeed have associations, but they shouldn't come first all the time. The GNN was never told to pay attention to such ordering. The second thing you'll see when disregarding the order and simply counting the number of predicted positives. There is almost always more than a thousand. So with no ordering to them, which ones out of these thousands should we pay attention to? The GNN has no answer.

In a way I've achieved success. I've implemented the edge predicting GNN as envisioned by the PGCN authors and others. Yet discovered that it was a failed premise from the start. The task of gene-disease prediction is not a binary but an ordinal one. It's the job of real life experiments to prove yes or no answers. There is little practical value in receiving such a judgment from a fallible neural network.

It seems though that RWRH is a superior solution and I've happened to find a setting where it gives promising results. Maybe more research effort should be spent on that algorithm instead.

Acknowledgments

I would like to thank my advisor András Gézsi for his continued guidance, encouragement, and patience. My thanks also go out to Bence Bruncsis for his valuable input.

Thanks to Stanford University and Jure Leskovec for publishing their Machine Learning with Graphs course freely available.

The study was supported by the National Research, Development and Innovation Office (NKFIH, Grant No. OTKA PD 134449).

Bibliography

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] Awni Altabaa, David Huang, Ciaran Byles-Ho, Hani Khatib, Fabian Sosa, and Ting Hu. geneDRAGNN: Gene disease prioritization using graph neural networks. In *2022 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–10, 2022. DOI: `10.1109/CIBCB55180.2022.9863043`.

[3] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. 57(1):289–300, 1995. ISSN 0035-9246. URL `https://www.jstor.org/stable/2346101`. Publisher: [Royal Statistical Society, Wiley].

[4] Olivier Bodenreider. The unified medical language system (UMLS): integrating biomedical terminology. 32:D267–D270, 2004. ISSN 0305-1048. DOI: `10.1093/nar/gkh061`. URL `https://doi.org/10.1093/nar/gkh061`.

[5] Steven H Brown, Randolph A Miller, Henry N Camp, Dario A Guise, and H Kenneth Walker. Empirical derivation of an electronic clinically useful problem statement system. *Annals of internal medicine*, 131(2):117–126, 1999.

[6] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[7] Lenore Cowen, Trey Ideker, Benjamin J Raphael, and Roded Sharan. Network propagation: a universal amplifier of genetic associations. *Nature Reviews Genetics*, 18 (9):551–562, 2017.

[8] Kevin Donnelly et al. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279, 2006.

[9] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html`.

[11] Ada Hamosh, Alan F Scott, Joanna S Amberger, Carol A Bocchini, and Victor A McKusick. Online mendelian inheritance in man (omim), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33(suppl_1):D514–D517, 2005.

[12] Peng Han, Peng Yang, Peilin Zhao, Shuo Shang, Yong Liu, Jiayu Zhou, Xin Gao, and Panos Kalnis. GCN-MF: Disease-gene association identification by graph convolutional networks and matrix factorization. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 705–713. Association for Computing Machinery, 2019. ISBN 978-1-4503-6201-6. DOI: `10.1145/3292500.3330912`. URL `https://doi.org/10.1145/3292500.3330912`.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[14] Kevin L Howe, Premanand Achuthan, James Allen, et al. Ensembl 2021. *Nucleic Acids Research*, 49(D1):D884–D891, 2021.

[15] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. URL `http://arxiv.org/abs/1609.02907`.

[16] Sebastian Köhler, Michael Gargano, Nicolas Matentzoglu, Leigh C Carmody, David Lewis-Smith, Nicole A Vasilevsky, Daniel Danis, Ganna Balagura, Gareth Baynam, Amy M Brower, et al. The human phenotype ontology in 2021. *Nucleic acids research*, 49(D1):D1207–D1217, 2021.

[17] Insuk Lee, U Martin Blom, Peggy I Wang, Jung Eun Shim, and Edward M Marcotte. Prioritizing candidate disease genes by network-based boosting of genome-wide association data. *Genome research*, 21(7):1109–1121, 2011.

[18] Chunyan Li, Hongju Liu, Qian Hu, Jinlong Que, and Junfeng Yao. A novel computational model for predicting microRNA–disease associations based on heterogeneous graph convolutional networks. 8(9):977, 2019. ISSN 2073-4409. DOI: `10.3390/cells8090977`. URL `https://www.mdpi.com/2073-4409/8/9/977`. Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[19] Yongjin Li and Jagdish C. Patra. Genome-wide inferring gene–phenotype relationship by walking on the heterogeneous network. 26(9):1219–1224, 2010. ISSN 1367-4803. DOI: `10.1093/bioinformatics/btq108`. URL `https://doi.org/10.1093/bioinformatics/btq108`.

[20] Yu Li, Hiroyuki Kuwahara, Peng Yang, Le Song, and Xin Gao. PGCN: Disease gene prioritization by disease and gene embedding through graph convolutional neural networks, 2019. URL `https://www.biorxiv.org/content/10.1101/532226v1`. Pages: 532226 Section: New Results.

[21] Carolyn E Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265, 2000.

[22] Lizhi Liu, Hiroshi Mamitsuka, and Shanfeng Zhu. Hpofiller: identifying missing protein–phenotype associations by graph convolutional network. *Bioinformatics*, 37 (19):3328–3336, 2021.

[23] Renming Liu, Christopher A Mancuso, Anna Yannakopoulos, Kayla A Johnson, and Arjun Krishnan. Supervised learning is an accurate method for

network-based gene classification. 36(11):3457–3465, 2020. ISSN 1367-4803. DOI: `10.1093/bioinformatics/btaa150`. URL `https://doi.org/10.1093/bioinformatics/btaa150`.

[24] João F. Matias Rodrigues and Christian von Mering. HPC-CLUST: distributed hierarchical clustering for large sets of nucleotide sequences. 30(2):287–288, 2014. ISSN 1367-4803. DOI: `10.1093/bioinformatics/btt657`. URL `https://doi.org/10.1093/bioinformatics/btt657`.

[25] Janet Piñero, Núria Queralt-Rosinach, Alex Bravo, Jordi Deu-Pons, Anna Bauer-Mehren, Martin Baron, Ferran Sanz, and Laura I Furlong. Disgenet: a discovery platform for the dynamical exploration of human diseases and their genes. *Database*, 2015, 2015.

[26] Janet Piñero, Juan Manuel Ramírez-Anguita, Josep Saüch-Pitarch, Francesco Ronzano, Emilio Centeno, Ferran Sanz, and Laura I Furlong. The DisGeNET knowledge platform for disease genomics: 2019 update. *Nucleic Acids Research*, 48(D1):D845–D855, 11 2019. ISSN 0305-1048. DOI: `10.1093/nar/gkz1021`.

[27] Press, Vettering, Teukolsky, and Flannery. *Numerical Recipes in C: The Art of Scientific Computing*, page 640. Cambridge University Press, 2 edition, 1992.

[28] Aditya Rao, Saipradeep Vg, Thomas Joseph, Sujatha Kotte, Naveen Sivadasan, and Rajgopal Srinivasan. Phenotype-driven gene prioritization for rare diseases using graph convolution on heterogeneous networks. 11(1):1–12, 2018. ISSN 1755-8794. DOI: `10.1186/s12920-018-0372-8`. URL `https://bmcmedgenomics.biomedcentral.com/articles/10.1186/s12920-018-0372-8`. Number: 1 Publisher: BioMed Central.

[29] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy, 1995. URL `http://arxiv.org/abs/cmp-lg/9511007`.

[30] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks, 2017. URL `http://arxiv.org/abs/1703.06103`.

[31] Juan Shu, Yu Li, Sheng Wang, Bowei Xi, and Jianzhu Ma. Disease gene prediction with privileged information and heteroscedastic dropout. 37:i410–i417, 2021. ISSN 1367-4803. DOI: `10.1093/bioinformatics/btab310`. URL `https://doi.org/10.1093/bioinformatics/btab310`.

[32] U Martin Singh-Blom, Nagarajan Natarajan, Ambuj Tewari, John O Woods, Inderjit S Dhillon, and Edward M Marcotte. Prediction and validation of gene-disease associations using methods inspired by social network analyses. *PloS one*, 8(5):e58977, 2013.

[33] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

[34] Damian Szklarczyk, Annika L. Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T. Doncheva, John H. Morris, Peer Bork, Lars J. Jensen, and Christian von Mering. STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. 47:D607–D613, 2019. ISSN 1362-4962. DOI: `10.1093/nar/gky1131`.

[35] Damian Szklarczyk, Annika L Gable, Katerina C Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T Doncheva, Marc Legeay, Tao Fang, Peer Bork, et al. The string database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic Acids Research*, 49(D1):D605–D612, 2021. DOI: 10.1093/nar/gkaa1074.

[36] Jean-François Truchon and Christopher I Bayly. Evaluating virtual screening methods: good and bad metrics for the "early recognition" problem. *Journal of chemical information and modeling*, 47(2):488–508, 2007.

[37] Alberto Valdeolivas, Laurent Tichit, Claire Navarro, Sophie Perrin, Gaëlle Odelin, Nicolas Levy, Pierre Cau, Elisabeth Remy, and Anaïs Baudot. Random walk with restart on multiplex and heterogeneous biological networks. 35(3):497–505, 2019. ISSN 1367-4803. DOI: 10.1093/bioinformatics/bty637. URL https://doi.org/10.1093/bioinformatics/bty637.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. URL http://arxiv.org/abs/1710.10903.

[39] Xiaochan Wang, Yuchong Gong, Jing Yi, and Wen Zhang. Predicting gene-disease associations from the heterogeneous network using graph embedding. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 504–511, 2019. DOI: 10.1109/BIBM47256.2019.8983134.

[40] Kyoko Watanabe, Sven Stringer, Oleksandr Frei, Maša Umićević Mirkov, Christiaan de Leeuw, Tinca JC Polderman, Sophie van der Sluis, Ole A Andreassen, Benjamin M Neale, and Danielle Posthuma. A global overview of pleiotropy and genetic architecture in complex traits. *Nature genetics*, 51(9):1339–1348, 2019.

[41] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019. URL http://arxiv.org/abs/1810.00826.

[42] Kuo Yang, Yi Zheng, Kezhi Lu, Kai Chang, Ning Wang, Zixin Shu, Jian Yu, Baoyan Liu, Zhuye Gao, and Xuezhong Zhou. PDGNet: Predicting disease genes using a deep neural network with multi-view features. 19(1):575–584, 2022. ISSN 1557-9964. DOI: 10.1109/TCBB.2020.3002771. Conference Name: IEEE/ACM Transactions on Computational Biology and Bioinformatics.

[43] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[44] Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. 12, 2021. Publisher: Frontiers Media SA.

[45] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. Heterogeneous graph structure learning for graph neural networks. In *35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[46] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. 34(13):i457–i466, 2018. ISSN 1367-4803. DOI: 10.1093/bioinformatics/bty294. URL https://www-cs.stanford.edu/~marinka/papers/decagon-ismb18.pdf.