Disease Protein Prediction: A Machine Learning Approach

Sabri Eyuboglu and Pierce Freeman

November 2017

1 Introduction

In human cells, proteins and the interactions between them can sometimes be disturbed or altered via viruses, genetic mutations, or other processes. For a given disease, there exists a set of proteins that when altered or otherwise disturbed are responsible for causing the disease's manifestation. The problem of disease protein discovery is that of identifying this set of proteins; such a prediction deepens our understanding of the disease and facilitates drug and treatment discovery. Disease protein discovery can be framed as a prediction problem: armed with existing data on human disease and proteins, for a particular disease compute the probability that a given protein is involved in that disease.

Statistical learning has been applied to the disease protein prediction problem before, but with limited success. In this work, we revisit the application of statistical learning to the disease protein prediction problem. We use two main data sources: the protein-protein interaction network and the Gene Ontology, and apply powerful feature extraction algorithms to each. Our features are used with three different binary classification models: logistic regression, support vector machines, and neural networks. Finally, we rigorously assess the performance of our models by performing a grid search over our feature selection algorithms, models, and hyper-parameters. TODO: One sentence on findings.

2 Problem Formulation

The disease-protein prediction problem can be framed quite naturally as a supervised binary classification problem. Let's call the set of all human proteins P. For every disease, the set of *known* disease proteins can be denoted $D \subset P$. It is important to distinguish between the set of *known* disease proteins, D and the set of *all* disease, which we'll denote $\hat{D}$. However, because we don't know the full set disease proteins, we use D as an approximation of $\hat{D}$. We then take all of the disease proteins in D and label them as *in-disease* ($y = 1$). Our binary classification now needs some *out-of-disease* examples ($y = 0$). However, because our known set of proteins D is incomplete, we can't say definitively that a node outside of D is out-of-disease. To address the lack of *out-of-disease* examples, we propose a simple approach: choose k random nodes not in D and label them as *out-of-disease*. Let's call this set of nodes of *out-of-disease* nodes N. At first glance, this may seem misguided at best or catastrophic at worst — if we follow this strategy a large majority of our labels will be randomly chosen. However, let's consider the probability that a protein j chosen at uniformly at random from P is active in the true disease pathway $\hat{D}$:

$$P(j \in D) = \frac{|\hat{D}|}{|P|} \tag{1}$$

We know that $|V| \approx 21,500$ and can estimate $|\hat{D}|$ with the maximum size of D over all our dataset of diseases. Using this approximation we get: $P(j \in D) \approx 0.2\%$. With such a low probability of error, we can view our random negative labels as accurate labels with some reasonable amount of random noise. Indeed, most supervised learning problems involve some noise in the labels and robust learning algorithms should be able to handle that noise. (In general, we set $k = |D|$, but this parameter can be tuned.) Thus, we can build a label vector $y \in \mathbb{R}^{|D|+k}$ where $m = |D| + k$.

$$y^{(i)} = \begin{cases} 1 & \text{if } i \in D \\ 0 & \text{if } i \in N \end{cases} \tag{2}$$

For each protein we can build a feature vector of length n using one of the feature extraction methods we discuss in Section 3. We can then define a feature matrix $X \in \mathbb{R}^{n \times m}$ where $m = |D| + k$.

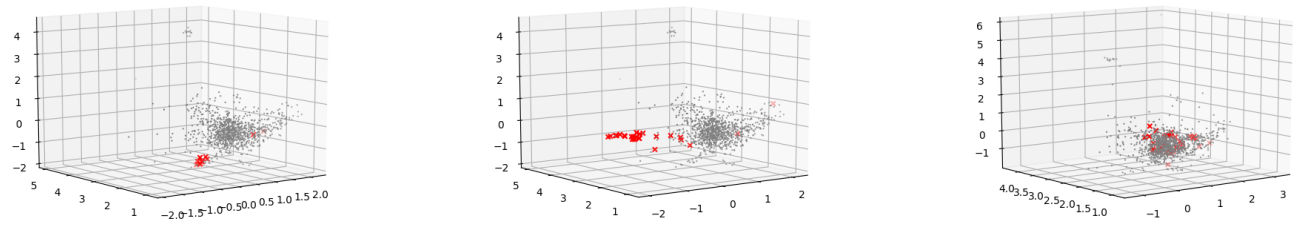
With a feature vector $X \in \mathbb{R}^{n \times m}$ and a label vector $y \in \mathbb{R}^m$, the disease protein prediction problem has been framed as a simple supervised binary classification problem. The next step is choosing how to fill the feature vector X.

3 Feature Selection

3.1 Network Embedding

Human diseases are not merely a product of our genetic constitution or an infection, but rather arise out of an intricate system of interactions between the proteins encoded by our genes. This fact has inspired a huge effort to document and understand the network of those protein-protein interactions which we'll refer to collectively as the protein-protein interaction network or the human interactome. The protein-protein interaction network can be intuitively represented as an undirected graph: the nodes are proteins and each edge represents a binary physical interaction between proteins. The mechanics behind human diseases can often be traced back to complex pathways of interacting proteins in our cells. This is to say, a disease phenotype is not so much a defect in a single gene or protein, as much as it is a consequence of a complex interaction of protein processes in human cells. Thus, disease pathways may be encoded in the human protein-protein interaction network.

The idea that diseases are caused by sets of proteins interacting in the cell has empirical support: proteins active in the same disease have a tendency to share interactions in the protein-protein interaction network. Goh *et al.*, in their paper The Human Disease Network, showed that the number of interactions between proteins of the same disease pathway was ten



(a) Diamond-Blackfan Anemia (b) Mitochondrial Complex I Deficiency (c) Celiac Disease

Figure 1: 3-dimensional neighborhood preserving embeddings of the PPI network. We used the same neighborhood preserving optimization problem to generate d-dimensional feature vectors. Proteins corresponding to each disease pathway are plotted in red, and random subset of 1,000 PPI network proteins are plotted in grey. We ran our embedding with $p = q = 1$.

times higher than random expectation[1]. This finding has been corroborated and extended by the works of others [2]. The idea that the PPI network could be leveraged for accurate disease protein prediction is in large part founded in these studies. Indeed, they suggest that proteins involved in the same or similar diseases often compose well-connected regions or neighborhoods of the PPI network.[3]

Given that disease proteins have a tendency to share neighborhoods in the protein-protein interaction network, we'd like to develop a feature embedding that can encode those neighborhoods. In their paper Scalable Feature Learning for Networks, Grover et al. proposed a method for learning a low dimensional mapping of nodes that preserves network neighborhoods.[4] Their approach maximizes an objective function that models the likelihood that the graphs networks are observed given the feature embedding. For a given graph $G = (V, E)$ and dimensionality d we can define an embedding $f \in V \rightarrow \mathbb{R}^d$ with parameters $\theta = \mathbb{R}^{|V| \times d}$. Grover et al.'s objective function can be written as the following log-conditional likelihood:

$$\sum_{u \in V} \log P(N_s(u)|\theta) \tag{3}$$

where $N_s(u)$ is the neighborhood of u. If we assume (1) that the probabilities that different nodes $v_i, v_j \in V$ belong to the same neighborhood $N_s(u)$ are conditionally independent, and (2) that the probability that a node $v_i \in V$ belongs to a neighborhood $N_s(u)$ is given by the softmax function $\sigma(\theta \cdot f(u))_i$, the conditional probability of observing a particular neighborhood $N_s(u)$ around node u given an embedding f can be written as follows:

$$P(N_s(u)|\theta) = \prod_{v_i \in N_s(u)} P(v_i \in N_s(u)|\theta) \tag{4}$$

$$= \prod_{v_i \in N_s(u)} \frac{\exp(f(v_i;\theta) \cdot f(u;\theta))}{\sum_{v_j \in V} \exp(f(v_j;\theta) \cdot d(u;\theta)} \tag{5}$$

Plugging into the objective function above, we can define the simple optimization problem:

$$\theta = \arg\max_{\theta} \sum_{u \in V} \left(-\log Z_u + \sum_{v_i \in N_s(u)} f(v_i;\theta) \cdot f(u;\theta) \right) \tag{6}$$

with the normalizing constant $Z_u = \sum_{v \in V} exp(f(u;\theta) \cdot f(v;\theta)$. The objective function above, which closely resembles that of softmax regression, can be optimized using stochastic gradient ascent to yield an optimal feature embedding $\theta \in \mathbb{R}^{|V| \times d}$.

One of the strengths of Grover et al.'s approach lies in the flexibility of the neighborhood function N_s. In order to generate the neighborhood for a node u, the neighborhood function $N_s(u)$ samples nodes around u in the network via random walks. Each random walk is parameterized by p and q: the return and in-out parameters, respectively. A higher p value will produce random walks that resemble depth first searches, while a lower p will produce random walks that more closely resemble breadth-first searches. Conversely, a high q value will encourage a breadth-first approach, while a low q will produce depth-first behavior. As Grover *etal.* show, using a neighborhood sampling function that more resembles breadth-first search will produce network embeddings that emphasize the structural roles of nodes in the network, whereas using a depth-first neighborhood sampling technique will produce embeddings that reflect more directly neighborhood locality. Our network feature embeddings will, thus, take three parameters: p, q, and d.

If we presume the disease module hypothesis – i.e. that disease proteins are localized in PPI network neighborhoods – then it would make sense to set p high and q low. However, Recent analysis of the PPI network by Agrawal *et al.* indicates that the disease module hypothesis is hardly a sure bet. Rather, while disease pathways do show high internal edge connectivity – by other network metrics, some disease pathways lack the characteristics of tightly-knit neighborhoods. Indeed, when we visualize our feature embeddings in 3-dimensional space, we find that many diseases show clear neighborhood localization similar to Diamond-Blackfan Anemia in Figure 2a and Mitochondrial Complex I Deficiency in Figure 2b. However, many diseases also resemble Celiac Disease, which shows no obvious

[1] Goh, Kwang-Il et al. "The Human Disease Network." Proceedings of the National Academy of Sciences of the United States of America 104.21 (2007): 8685–8690. PMC. Web. 20 Oct. 2017.

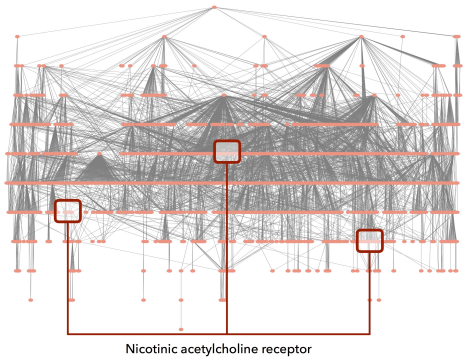
[2] Gandhi T, et al. Analysis of the human protein interactome and comparison with yeast, worm and fly interaction datasets. Nature Genetics. 2006; 38:285–293.

[3] Barabási, Albert-László, Natali Gulbahce, and Joseph Loscalzo. "Network Medicine: A Network-Based Approach to Human Disease."

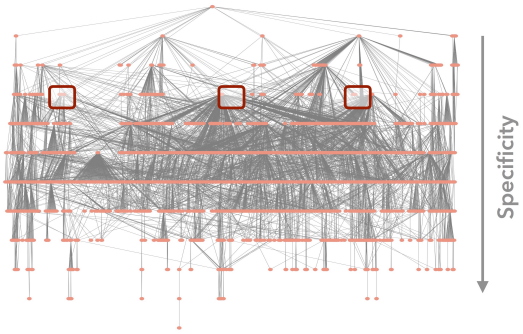
[4] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2016.

neighborhood localization in 3-dimensional space. Agrawal *et al.* also found that for many diseases, active proteins tend to take on the same structural role in network motifs of the PPI network – even if they appear in different neighborhoods. For 60% of diseases studied, proteins in the disease pathway occupy specific positions within certain motifs at a much higher rate than proteins drawn at random. [5] This suggests that rather than emphasizing neighborhood locality with a high p and low q, it would be promising to emphasize structural equivalence with a low p an high q. Thus, rather than over-theorizing and picking one set of parameters p and q, we include the parameters p and q as hyperparameters in our grid search.

3.2 Gene Ontology



(a) Direct Protein to Ontology Mapping



(b) Protein to Fixed Specificity

Figure 2: Ontology Trees

Due to their unique folding pattern, each protein performs a set of quantized "actions." These units of work may be used for different ends within a cell, but the capabilities of the protein itself doesn't change.

The Gene Ontology Consortium is an effort to standardize the definitions of these capabilities. The project starts by laying out a schema used to classify the different types of molecular functions. It then annotates individual proteins with this predefined schema. By unifying the two efforts, we're able to describe each protein in the PPI through its functional properties.

[5]Monica Agrawal, Marinka Zitnik, Jure Leskovec; Large-Scale Analysis of Disease Pathways in the Human Interactome; bioRxiv 189787; doi: https://doi.org/10.1101/189787

Prior work has established that these annotations can reveal latent attributes of proteins and serve as a compliment to the protein-protein interaction network.[6] Intuitively, interacting proteins need to perform complimentary actions. As such, a model should be able to learn what is missing within an existing chain and therefore predict additional involved proteins.

Parsing through the Gene Ontology is a significant undertaking, due to their use of proprietary formats and the magnitude of the data available. We developed an extensible parser for these file types and devised a mapping from our GeneIDs (within the PPI network) to the UniProtKB ID (used with the gene ontology).

The ontology itself roughly resembles a tree - where general terms lead to more specific terms. We make a few novel observations about this graph in the context of our machine learning problem. Namely, there's a to-many mapping between proteins and the terms that define its molecular function. Depending on the confidence of the annotator, this annotation location can be at wildly different depths within the overall ontology tree (see the measure of specificity in Figure 2). Additionally, using overly specific features may balloon the size of our feature space and lead to over-fitting.

This led us to develop a method for traversing this tree to recover a specificity level given any term. By retrieving the same level of specificity across the board, we both limit the feature space and better equalize the information describing each protein. This also lets us experiment with different feature sizes without throwing out information from our rich dataset. The adjustable parameter for the ontology is the level of specificity from which we want our features to be produced.

4 Models

We applied three different supervised binary classification models to the disease protein prediction problem: logistic regression, support vector machines, and neural networks.

Logistic Regression fits a linear separator in our feature space by maximizing the likelihood of observing the training set data. We chose this model because it of its simplicity and also because it outputs probabilities very naturally. Computing a predicted probability for each protein is important in the disease protein prediction problem, because it is often most useful to rank proteins and return the top-k, rather than simply returning the classification of the model. Our logistic regression takes as parameters a regularization type (i.e. L1 or L2) and a regularization strength.

The second model we chose was **Support Vector Machines** the reason being that SVMs can learn separators nonlinear in our feature space via the use of kernel's. Specifically, we applied the Radial Basis Function kernel as well as linear kernel for comparison. The hyperparameter we tune is the regularization strength. Support Vector Machines do, however, present two major drawbacks when compared to logistic regression: (1) SVMs take substantially longer to train and (2) SVMs do not

[6]Wu, Xiaomei, et al. "Prediction of yeast protein–protein interaction network: insights from the Gene Ontology and annotations." Nucleic acids research 34.7 (2006): 2137-2150. APA

output probabilities, so computing a ranking of proteins is more challenging.

Finally, we also applied a simple **Neural Network** to the disease protein prediction problem. Our neural network is fully connected with ReLU activation and a sigmoid output layer. In a way, this model presents a sort of best-of-worlds between SVMs and Logistic Regression: it is able to learn non-linear separators of data and also outputs probabilities useful for protein ranking. Of course, the central drawback of Neural Networks is their costly training process via backpropagation. The hyperparameters we tuned for our neural networks were the number of hidden layers, the number of hidden units per layer, and number of training epochs.

Algorithm 1 Grid Search

```
let p = parameters
for feature ∈ features do
    for model ∈ models do
        params = permutations(p_feature, p_model)
        let m as metrics
        for param in params do
            m_param = recall(feature, model, param)
        end
        best := argmax(m)
    end
end
```

5 Experimental Setup

In this section we describe the data and methods used for tuning and evaluating or disease protein prediction models.

5.1 Data

Human Protein-Protein Interaction Network Our first dataset is the protein-protein interaction (PPI) network compiled by Menche *et al.*[7] and Chatr-Aryamontyi *et al.* [8] The PPI network is encoded in a graph where each node represents a protein and each edge records a known binary physical interaction. Within this model, there are $|V| = 21,557$ nodes with $|E| = 340,000$ edges.

Disease Pathway Dataset The second dataset is a listing of 519 diseases and the proteins known to be active in each. These disease-protein associations come from DisGeNET, a centralized database containing information on disease-protein relationships. Across all diseases in the dataset, the median number of associated proteins is 21 and the minimum is 10. Some more complex diseases have over one hundred known associations.

Gene Ontology The Gene Ontology Consortium provides a framework for accessing data on protein function. We fill the ith index of our feature vector $x_i^{(i)}$ with a one-hot flag for whether the protein is described by molecular function i.

5.2 Hyperparameter Tuning

Unlike most implementation of grid-search, we want to optimize over both the hyperparameters specified by the features and the models. In order to evaluate our search space of possible parameters in a methodological way, we created an implementation of Grid Search that could run on distributed computation. With this, we considered each combination of feature sets and algorithms jointly. Our grid-search process is outlined in the following pseudocode:

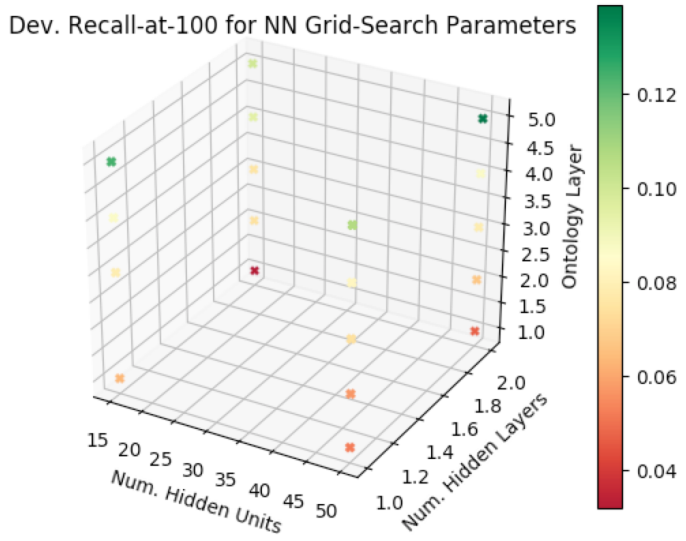


Figure 3: Grid search over selected neural network parameters. A points position in euclidean space specifies a particular hyperparameter set and its color indicates dev recall-at-100 for that configuration.

We chose 80 diseases at random for our development set and ran our grid search approach over those diseases. We took all possible permutations of manually-generated hyperparameters and performed k-fold cross validation within each disease pathway — that is, we split the disease's proteins into k-folds and ran k iterations where we trained on k-1 folds and withheld one as a dev set. We then chose the parameters that optimized average dev recall-at-100 across the iterations of k-fold cross validation. We then took 20 other diseases and used these to calculate test recall-at-100, which we report below.

In Figure 3, we visualize our grid-search results for a neural network model with gene ontology features.

5.3 Model Testing

After running hyper-parameter tuning, we ran our optimal models on test sets of size 20 diseases in our test set, and evaluated their performance using k-fold cross validation.

[7] Menche, J, Sharma, Amitabh, Kitsak, Maksim, Ghiassian, Susan Dina, Vidal, Marc, Loscalzo, Joseph Barabasi, Albert-Laszlo (2015). Uncovering disease-disease relationships through the incomplete interactome. Science, 347.

[8] Chatr-aryamontri,Andrew et al."The BioGRID Interaction Database: 2015 Update." Nucleic Acids Research 43.Database issue (2015): D470–D478. PMC. Web. 17 Nov. 2017.

6 Results

After performing grid-search to find the optimal hyperparameters for each model, we analyzed the performance on the test set of 20 diseases. The most apt performance metric for the disease protein prediction problem is **recall-at-100**. Recall-at-100 measures the fraction of disease proteins that appear in the first 100 proteins ranked by a disease protein prediction model. This is a more relevant metric than accuracy, because accuracy is generally inflated by correctly identifying non-disease proteins as non-disease. Based on the numbers we computed

Table 1: Recall-at-100 for Ontology features

	Dev. Recall-at-100	Test Recall
Logistic	0.230	0.247
SVM	0.193	0.183
Neural Network	0.138	0.151

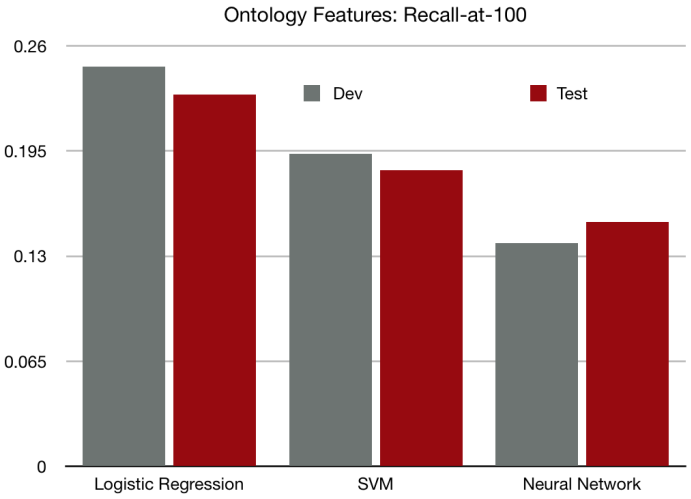


Figure 4: Recall-at-100 for different optimized models with ontology features..

below, the ontology features showed very strong performance, especially with the logistic regression model. Across models, optimal performance was achieved with a layer specificity parameter set to **5**. Also, test accuracy dipped slightly below dev accuracy for logistic regression and SVM, but actually rose for neural networks.

The ontology features performed substantially better than the network embeddings and within both feature sets, Logistic Regression outperformed SVM which itself outperformed Neural Networks.

Table 2: Recall-at-100 for Embedding Features

	Dev. Recall-at-100	Test Recall
Logistic	0.132	0.153
SVM	0.111	0.091
Neural Network	0.104	0.0616

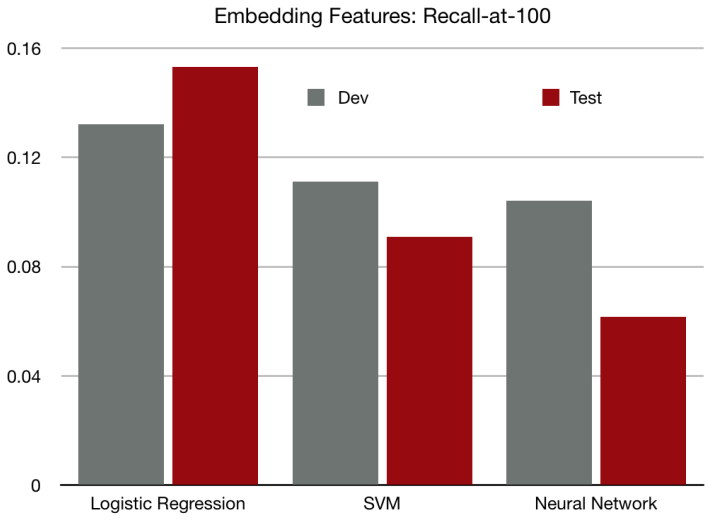


Figure 5: Recall-at-100 for different optimized models with ontology features.

7 Future Work

Going forward, we'd like to further dive into the differences in performance between the various models we used. There's also evident room for the use of ensemble methods, since both graph embeddings and the ontology provide different reference frames for each proteins. Combining these results with the use of network specific methods (either network algorithms or deep learning) is also an area for future study.

Furthermore, while this study focused on the evaluation of the two feature extractors *in isolation*, it is worth assessing the performance of a feature extractor that combines the two feature approaches into one larger feature vector.

9

[9]Contributions: Both of us spoke at length about the general strategies for this machine learning tasks, as well as investigating sources of feature-sets. During the implementation phase, Pierce focused more heavily on parsing the gene ontology network, creating feature vectors, and visualizing the dataset. Sabri focused on implementing the network embeddings and creating a testing harness for the evaluation.

Both members of our group, Pierce Freeman and Sabri Eyuboglu, are working on a CS224W project tackling the Disease Protein Prediction problem from another angle. For the 224W project, we are using Graph Convolutional Networks (GCN) to perform node classification on the PPI network. Although the approach we use for our 229 project is distinct from the approach for 224W, there a some components of the project that are or may be shared, namely: 1) Some code for running prediction experiments and parsing input data. 2) Background research into the problem and methods for the problem. 3) Gene-Ontology based feature selection may be used as input for the Graph Convolutional Network.