Better Inference Scores for Chemical-Disease Relationships in the Comparative Toxicogenomics Database

Ashton Teng
ashteng@stanford.edu

Introduction and Motivation

The Comparative Toxicogenomics Database (CTD) (Davis et al., 2017) related three fundamental entities in biology: chemicals, genes, and diseases, which can be constructed as a tripartite graph with Chemical-Gene (C-G), Gene-Disease (G-D), and Chemical-Disease (C-D) relationships. Each of the three types of relationships can be manually curated via experimental studies of different natures. "In July 2016, core CTD consisted over 1.6 million manually curated interactions (including 1,379,105 chemical-gene, 202,085 chemical-disease (C-D) and 33,583 gene-disease direct interactions)… from 117 866 peer-reviewed scientific articles studied in 564 species" (Davis et al., 2017).

In particular, Chemical-Disease (C-D) relationships can have two types. **Curated relationships** that have experimental evidence, and **inferred relationships** that do not have direct experimental evidence. Inferred C-D relationships are only calculated if the chemical A has a curated interaction with gene B, and independently gene B has a curated interaction with disease C (Davis et al., 2011).

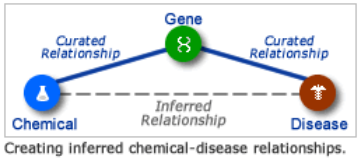
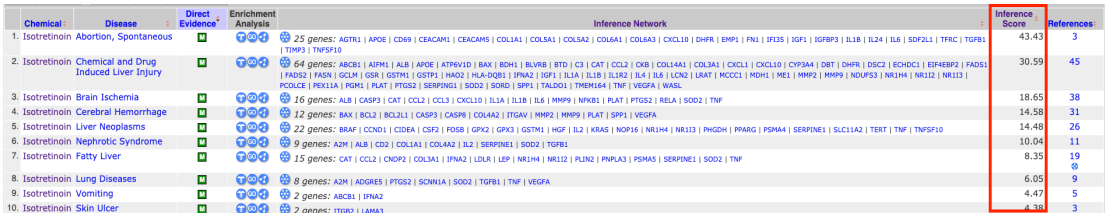


Figure 1. Infographic showing the 3 main relationships in CTD, chemicals, genes, and diseases.

The **inferred** C-D relationships are assigned an **Inference Score**, based on the topology of the network consisting of the chemical, disease, and their shared genes. The diseases associated with a certain chemical (or vice versa) can then be ranked by inference score. As the data in CTD have grown, the number of inferred edges has increased exponentially. Thus, the task is to **develop a method for ranking C-D inferences**. For a biologist that is interested in a particular C-D relationship, such ranked inference scores are extremely useful for prioritizing research targets, as experiments take a significant amount of time, energy, and money.



	Chemical	Disease	Direct Evidence	Enrichment Analysis	Inference Network	Inference Score	References
1.	Isotretinoin	Abortion, Spontaneous	M		25 genes: AGTR1 | APOE | CD69 | CEACAM1 | CEACAM5 | COL1A1 | COL5A1 | COL5A2 | COL6A1 | COL6A3 | CXCL10 | DHFR | EMP1 | FN1 | IFI35 | IGF1 | IGFBP3 | IL1B | IL24 | IL6 | SDF2L1 | TFRC | TGFB1 | TIMP3 | TNFSF10	43.43	3
2.	Isotretinoin	Chemical and Drug Induced Liver Injury	M		64 genes: ABCB1 | AIFM1 | ALB | APOE | ATP6V1D | BAX | BDH1 | BLVRB | BTD | C3 | CAT | CCL2 | CXB | COL14A1 | COL3A1 | CXCL1 | CXCL10 | CYP3A4 | DBT | DHFR | DSC2 | ECHDC1 | EIF4EBP2 | FADS1 | FADS2 | FASN | GCLM | GSR | GSTM1 | GSTP1 | HAO2 | HLA-DQB1 | IFNA2 | IGF1 | IL1A | IL1B | IL1R2 | IL4 | IL6 | LCN2 | LRAT | MCCC1 | MDH1 | ME1 | MMP2 | MMP9 | NDUFS3 | NR1H4 | NR1I2 | NR1I3 | PCOLCE | PEX11A | PGM1 | PLAT | PTGS2 | SERPING1 | SOD2 | SORD | SPP1 | TALDO1 | TMEM164 | TNF | VEGFA | WASL	30.59	45
3.	Isotretinoin	Brain Ischemia	M		16 genes: ALB | CASP3 | CAT | CCL2 | CCL3 | CXCL10 | IL1A | IL1B | IL6 | MMP9 | NFKB1 | PLAT | PTGS2 | RELA | SOD2 | TNF	18.65	38
4.	Isotretinoin	Cerebral Hemorrhage	M		12 genes: BAX | BCL2 | BCL2L1 | CASP3 | CASP8 | COL4A2 | ITGAV | MMP2 | MMP9 | PLAT | SPP1 | VEGFA	14.58	31
5.	Isotretinoin	Liver Neoplasms	M		22 genes: BRAF | CCND1 | CIDEA | CSF2 | FOSB | GPX2 | GPX3 | GSTM1 | HGF | IL2 | KRAS | NOP16 | NR1H4 | NR1I3 | PHGDH | PPARG | PSMA4 | SERPINE1 | SLC11A2 | TERT | TNF | TNFSF10	14.48	26
6.	Isotretinoin	Nephrotic Syndrome	M		9 genes: A2M | ALB | CD2 | COL1A1 | COL4A2 | IL2 | SERPINE1 | SOD2 | TGFB1	10.04	11
7.	Isotretinoin	Fatty Liver	M		15 genes: CAT | CCL2 | CNDP2 | COL3A1 | IFNA2 | LDLR | LEP | NR1H4 | NR1I2 | PLIN2 | PNPLA3 | PSMA5 | SERPINE1 | SOD2 | TNF	8.35	19
8.	Isotretinoin	Lung Diseases	M		8 genes: A2M | ADGRE5 | PTGS2 | SCNN1A | SOD2 | TGFB1 | TNF | VEGFA	6.05	9
9.	Isotretinoin	Vomiting	M		2 genes: ABCB1 | IFNA2	4.47	5
10.	Isotretinoin	Skin Ulcer	M		2 genes: ITGB2 | LAMA3	4.38	3

Figure 2. CTD user interface that shows the inference score of chemical-disease associations.

The CTD authors explain that "the inference score reflects the degree of similarity between CTD chemical–gene–disease networks and a similar scale-free random network. The higher the score, the more likely the inference network has atypical connectivity" (Davis et al., 2017). However, from the definition above, it is unclear how the inference score relates to the probability that a chemical-disease relation actually exists in real life, rather than just being of "atypical connectivity" compared to a scale-free random network.

In their paper detailing how the inference score is measured (King et al., 2012), the CTD authors based their method of inferring C-D edges on existing protein-protein interaction modeling literature. These

methods are based on inferences on local network topology, defined by one chemical, one disease, and the direct set of genes that interact with the chemical and disease. Their method computationally accessible and easily interpretable, though as I will show that the C-D inferences scores produced actually does a poor job in predicting which C-D edges actually exist based on direct experimental evidence.

I demonstrate two methods that produces C-D inference scores that better models real curated edges. Instead of only using local topology of one chemical, one disease, and the genes that connect to both, I utilize both unsupervised and supervised methods that learn higher order node and edge representations from the surrounding graph, such as DeepWalk and Graph Auto-Encoder (GAE). After the nodes and edges are embedded, I then phrase this problem as a link prediction problem and take the model's predicted probability of a link as the inference score. My first model involving DeepWalk for node embeddings and Logistic Regression achieved an AUROC of 0.94, and my second GAE model achieving 0.98, both remarkably higher than the CTD inference score AUROC of 0.56.

Related Work

CTD authors Davis et al. uses local topology-based statistics to drive C-D edge inference scores and does not take into account the curated status of certain C-D relationships, and thus is completely unsupervised. One local network consists of one chemical, one disease, and the set of genes that interacts with the chemical and disease. Davis et al. references past work by Goldberg and Roth, as well as Li and Liang, in measuring reliability of edge predictions in protein-protein networks, which is a similar problem, except between two proteins instead of a chemical and a disease.

Goldberg & Roth argues that due to the small-world nature of protein-protein interaction networks (which Davis et al. argues also applies to CTD), it should suffice to base inference scores of an edge based on the neighborhood cohesiveness around them. They measured the degree to which the neighborhood cohesiveness of an edge is consistent with a small-world network to score individual edges according to their likelihood of being true. They propose four variants of a clustering coefficient, of which the best performing was the hypergeometric clustering coefficient:

$$C_{XY} = -\log \sum_{i=m}^{\min(n_X, n_Y)} \frac{\binom{n_x}{i}\binom{N-n_x}{n_y-i}}{\binom{N}{n_y}} \qquad (1)$$

In equation 1, n_x and n_y are the number of edges for nodes x and y, respectively, m is the number of mutual neighboring nodes, and N is the total number of chemicals, genes and diseases with any interaction in CTD. C_{XY} quantifies the negative log probability of obtaining a certain number of mutual neighbors between nodes x and y at or above the observed number m by chance, under the null hypothesis that neighborhoods are independent. This metric solely takes into account the connectivity of each node, as well as the number of common neighbors.

Li & Liang developed two common neighbor statistics, P_1 and P_2, combined to assess the reliability of protein-protein interactions. $\boldsymbol{P_1}$ (Equation 2) is similar to C_{XY} above, quantifying **the probability that the two proteins have their respective number of nodes and common connections compared to what is randomly expected to happen**. They then realized that this metric would lead to erroneously enhanced scores for edges connected to hub proteins, as they only take into account the average degrees of nodes.

$$P_1(m|N,n_x,n_y) = \frac{\binom{N}{m}\binom{N-m}{n_x-m}\binom{N-n_x}{n_y-m}}{\binom{N}{n_x}\binom{N}{n_y}} \quad (2)$$

$$P_2(X \text{ and } Y \text{ share } A|\kappa,N) = \prod_{i \in A} \frac{n_i(n_i-1)}{N(N-1)} \quad (3)$$

P_2 (Equation 2, which is an approximation) was then devised, where A, in CTD terms, is the set of genes that connect the chemical and disease nodes, and n_i is the number of edges for each connecting gene in A. **P_2 therefore controls for the number of connections within the connecting genes as well, effectively mitigating the effect of hub nodes which have a large number of connections**.

Coming back to Davis et al., they basically experimented with the above metrics, and also two additional weighted combination metrics S_{XYA} and W_{XYA} (Equations 4-6), which they find as being a good medium ground between P_1 and P_2, and leads to the best results on the CTD network.

$$p(\theta) = k \prod_{i=1}^{n} p_i(\theta)^{w_i} \quad (4)$$

$$S_{XYA} = -(w_1 \log_{10}(p_1) + w_2 \log_{10}(p_2)) \quad (5)$$

$$W_{XYA} = -(w_1 \log_{10}(p_1) + w_2 \log_{10}(p_2)) \quad (6)$$

where

$$w_1 = \left(1 - \frac{e}{2e^m}\right)$$

and

$$w_2 = \left(1 - \frac{e}{2e^m}\right)$$

Davis et al. evaluated these different inference score methods via many methods. One was to use the curated evidence as a guide. They evaluated which inference score ranked C-D inferences with curated evidence higher than those without. This includes seeing how many of the top tank C-D edges are from the curated set and measuring the difference between mean curated C-D edge scores and mean non-curated C-D edge scores. They also looked at which metric was able to discount hub nodes appropriately (e.g. a poor metric would rank C-D edges related to “prostatic neoplasms” highly, as it is a hub node with dense connections). They also qualitatively looked at the gene-sets associated with inferred C-D connections and saw if they formed coherent biological pathways by referencing databases such as GO and KEGG.

In the above three papers, C-D and protein-protein are completely made using local network topology, defined by one chemical, one disease, and the set of chemicals that interact with the chemical and disease. The strength of this approach is that calculations are computationally accessible, even with dense common gene structure, C-D local topologies are in the range hundreds of nodes and edges. Another strength is that the formulas for calculating inference scores are based on interpretable discrete probability theory. When new evidence is added into the CTD network, only the inference score of the chemical and disease in question needs to be updated.

However, the weakness of these approaches is the loss of information embedded in the graph outside of the local topology, such as clustering of neighboring chemicals, clustering of neighboring diseases, and structural roles. Even though both Davis et al. and Li & Liang tried to incorporate second degree connections within common neighbors, local topology probably loses some generalizable information, especially since biological pathways often have complex chain of actions. The current method is akin to doing local feature engineering on the graph and manually producing a scoring function for those

features. Furthermore, all of the above methods try to derive a probability or a score for an edge, which requires certain assumptions to be made about not only a null hypothesis of a random graph, but also a mathematical characterization of a "real" and "normal" protein-protein or chemical-gene-disease graph. For example, Davis et al.'s "inference score reflects the degree of similarity between CTD chemical–gene–disease networks and a similar scale-free random network". Similarly, Goldberg & Roth makes the assumption of a small-world network, in which "an edge is likely to be a side of more triangles than would be expected in a random graph". Khanin & Wit examined 10 published datasets of various biological interactions and determined that none of them are actually scale-free and drawn from the power-law distribution, thus rendering the 3 papers' assumptions of the graph potentially erroneous.

Methods and Results

This project attempts to derive better C-D inference scores from the CTD network by resolving the above issues. First of all, instead of only making inferences from local topology (one chemical, one disease, and the genes that connect them), I explore higher degree node representations that take advantage of larger, more organic graph structure. I try both representation learning methods based on random walks (DeepWalk), and end-to-end neural networks (Graph Auto-Encoder). Second of all, in order to improve the issue of making unrealistic assumptions about the background null hypothesis graph in order to evaluate inference scores, I do not make any assumptions of the structure of the graph and leverage the subset of C-D edges that are curated using supervised learning for evaluation purposes. The two supervised models presented are one involving regularized Logistic Regression on the concatenated node embeddings from DeepWalk, and one end-to-end solution with Graph Auto-Encoders (GAE).

Data Preparation

From the Comparative Toxicogenomics Database (CTD), I collected the relevant datasets for my problem: chemical-gene (C-G), gene-disease (G-D), and chemical-disease (C-D) associations, and mapped alphanumeric names for chemicals, genes, and diseases into unique integers to generate edge-lists. The C-G and G-D edge-lists were concatenated together (let's call this the C-G-D graph), which serves as the ground for us to learn the representations of nodes, which will then be used to predict the existence of C-D edges. The C-D dataset was used to construct the labels for supervised learning (the current CTD inference score is unsupervised – the curated edges were only used as evaluation). This dataset included C-D pairs, and also a binary indicator for whether or not this edge is curated by experimental evidence, and CTD.'s original inference score for this edge. This cleaned data consists of 16,116 distinct chemicals, 7224 distinct diseases, and 49,072 distinct genes.

The subset of C-D edges that are curated by experimental evidence numbers 20,991. As this network does not have any definite negative edges (non-existent edges could be negatively, neutral, or just undiscovered), An imperfect decision needed to be made regarding negative labels. Taking inspiration from a past CS224W project that studied protein-protein interaction networks (Eyuboglu and Freeman, 2017), I randomly sampled 20,991 non-curated C-D edges that do have an inference score (that means C and D shares at least one curated gene) and assigned them as negative edges. This is potentially problematic because non-curated edges might simply be undiscovered relationships, since there are no labeled negative edges in the network. However, since biological networks tend to be sparse, and because there are 2,459,998 total inferred C-D edges, our random negative sample represents only 0.85% of total possible edges. It is highly likely that the majority of our sample are negative. Even though this approach adds some noise to our model, it is preferable to adding a huge bias by manually choosing edges via some metric. Now we arrive at a dataset that can be used to evaluate both my supervised models as well as the CTD inference score (by simply setting a threshold for classification of C-D edges). The final result of the data collection process is a set of 20,991*2 = 41,982 data points, with each data point being a tuple

(chemical id, disease id, class), with class being binary and balanced. This is a small fraction of the entire C-G-D graph from which the chemical and disease representations will be learned.

Graph Characteristics

Constructed within SNAP, the C-G graph contains 61,700 nodes and 1,106,384 edges, and the G-D graph contains 55,901 nodes and 26,777,114 edges. The combined C-G-D graph (joined by common genes) contains 69,584 nodes and 27,883,498 edges, with edges mostly coming from G-D connections. The entire C-G-D graph is basically one huge connected component, with 69,557 nodes within one component. Upon analysis of the diameter of our graph by running breadth-first search on 100 test nodes, the C-G-D graph has a diameter of 6, whereas an Erdos Renyi random graph with the same number of nodes and edges has a diameter of 3, and a similar Small World graph has a diameter of 72. In addition, in terms of degree distribution (Figure 3), our C-G-D graph shows a marked difference from the two random graphs as well, with a power-law like decay in the number of nodes with high degree. The degree distribution and the graph diameter support the idea of a small number of highly connected global hub nodes, with most other nodes having fewer connections.

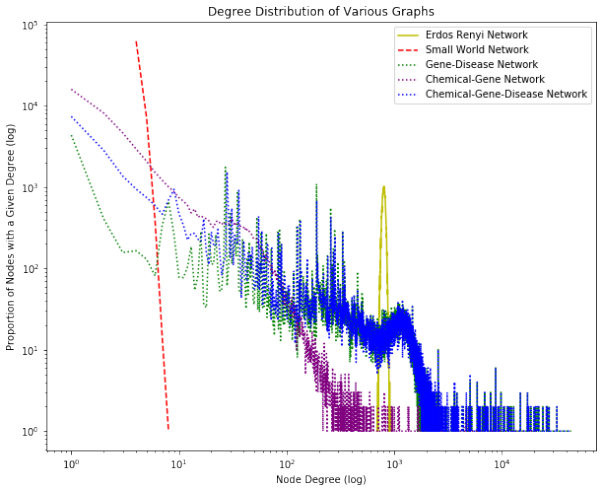


Figure 3. Degree distributions of C-G, G-D, and C-G-D networks compared with an Erdos Renyi and small world random networks. The C-G-D network is mostly similar to the G-D network due to the large number of G-D edges.

Evaluating Current CTD Inference Score

Since the goal of this project is to derive a better inference score for non-curated C-D relationships, it is important to first evaluating the existing CTD inference score for how well it does in classifying positive (experimental evidence) and negative (negative sampling) edges. The CTD inference score is a real number that is assigned to all C-D pairs that have at least one curated gene in common. Davis et al. did use curated evidence to evaluate whether the inference score ranked C-D inferences with curated evidence higher than those without but did not sample for negative edges or formulate a classification problem. The current evaluation involves simply choosing a real number threshold and classifying C-D edges with inference scores above threshold as positive, and those below as negative.

Figure 4 shows the inference for positive (1) and negative edges (0) respectively. Even though low inference score edges tended to be negative, there was a very wide range of inference scores within positive edges. To quantify this separation, I vary the threshold of binary classification to form a ROC curve (Figure 5), resulting in an AUROC of only 0.56 and a maximum classification accuracy of 0.61. Even though a poorer performance is expected as this is a supervised technique with no access to labels, it is still surprising that the current inference score in production at CTD does so poorly in predicting which edges have experimental evidence.

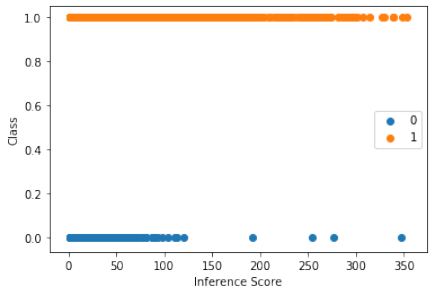


Figure 4. The CTD inference score does not provide an effective way to separate C-D edges that have a curated relationship and not.

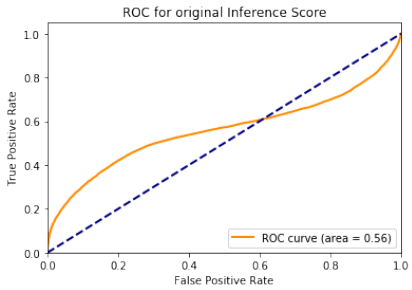


Figure 5. ROC curve for differing thresholds of the inference score to classify C-D edges.

Baseline Model (DeepWalk + Logistic Regression)

To improve upon the current CTD inference score, I propose a supervised learning approach in which a model learns graph representations and makes a binary link prediction for every C-D pair. The probability of the binary prediction can then be translated to a continuous inference score. Under this framework, there are three things that the model needs to address: 1) how to generate node embeddings, 2) how to combine two node embeddings in a meaningful to produce an edge embedding, and 3) how to utilize the noisy and limited curated labels to train the model effectively to do link prediction from the edge embeddings. In my baseline model, I generate unsupervised node embeddings from the C-G-D graph using DeepWalk, represent C-D edges by the concatenation of C and D node embeddings, and predict edges using Logistic Regression.

DeepWalk is an unsupervised method of learning node embeddings. It uses local "neighbors" information obtained from truncated random walks to learn node representations by predicting a node's neighbors (Perozzi et al., 2014). This is akin to the natural language processing method Word2Vec, where the task of predicting neighboring words is used to learn word embeddings. DeepWalk operates in two steps: for each node, generate a random walk of length *t* to form a path, and training node embeddings to maximize the probability of seeing the neighbors on this path. DeepWalk is trained using stochastic gradient descent (SGD). DeepWalk was originally devised to capture node representations in social networks but has since been utilized widely across many problem domains, thanks to the generalizability of the method. No labels are required as the co-occurrence of nodes within a random walk is seen as a natural metric of similarity between nodes. However, one potential pitfall with our tripartite graph is whether DeepWalk will be able to converge while simultaneously learning representations for chemicals, genes, and diseases, since the loss function optimizes for the similarity between node representations that co-occur on a random walk. I ran DeepWalk (Perozzi et al., 2014) on the C-G-D graph with 40 walks on each node and a walk length of 40, producing node embeddings with 64 dimensions. PCA analysis of chemical and disease embeddings mapped onto the same space show distinct node embedding patterns for chemicals and diseases (Figure 6). The chemical and disease embeddings are then concatenated into a 128-dimension vector that will be used to represent the embedding of C-D edges. Figure 7 shows PCA analysis of these concatenated embeddings, labelled with class (with or without curated edge).

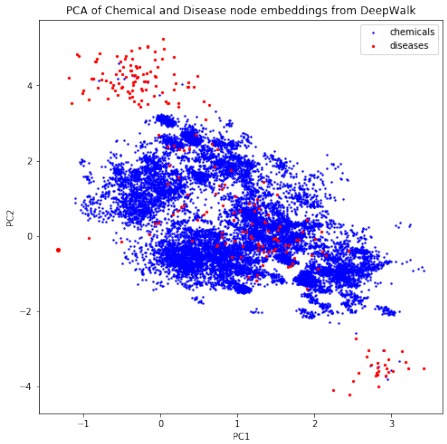


Figure 6. PCA analysis of chemical and disease node embeddings (each 64 dimensions).

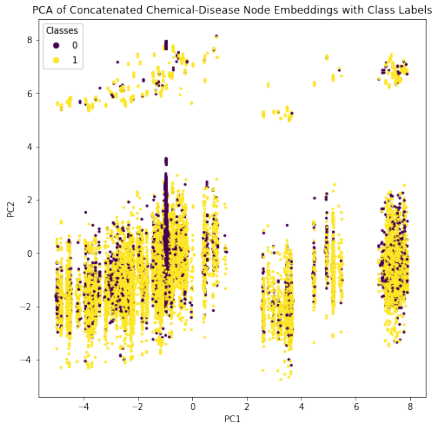


Figure 7. PCA analysis of concatenated chemical-disease node embeddings (128 dimensions

With the concatenated node embeddings forming the 128-dimensional embedding for each C-D edge, I applied regularized Logistic Regression to generate the classification in this binary link prediction problem. As this model was meant to be a baseline, I did not do hyperparameter tuning. After extracting the balanced positive and negative labels through negative sampling, I did a 67-33 train-test split (28,127 training points and 13,855 testing points), and used Scikit-learn's Logistic Regression classifier with L2 regularization to fit the training data, using default parameters (C=1.0, intercept=True) (Pedregosa et al., 2011). The results were surprisingly good, with accuracy on the training set reaching 0.9186 (AUROC 0.95) and testing set reaching 0.9171 (AUROC 0.94) (Figure 8).

Surprised at how well this simple model did, I investigated the DeepWalk embeddings. As a sanity check, I randomized the embeddings entered into the classifier (for each C-D pair, I concatenated a random C embedding and D embedding), thankfully getting a 0.5062 classification accuracy, which is expected by random given the balanced class labels. I then fit another Logistic Regression model on only chemical embeddings (64 dimensions), and got an accuracy of 0.5621. Finally, I when using only disease embeddings (64 dimensions), I found that the classifier achieves an accuracy of 0.9159. Looking back and visualizing my Logistic Regression weights on the original model, I found that indeed, the classifier is almost only looking at the embeddings from the disease node to do the classification (Figure 9). This is concerning, as in all my effort in finding representations for C-D edges, my classifier would output the same result for a disease regardless of chemical (of course, the chemical must share a curated gene with this disease to be included in this dataset, so it's not just any chemical). Nevertheless, even if we are only using the disease embeddings, the embedding itself includes information from all of its gene and chemical neighbors through DeepWalk.

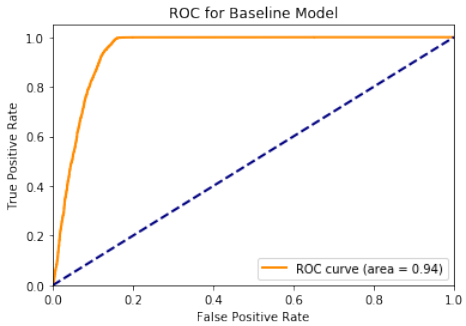


Figure 8. ROC curve for the baseline model testing set, AUROC=0.94. The training set has an AUROC=0.95.

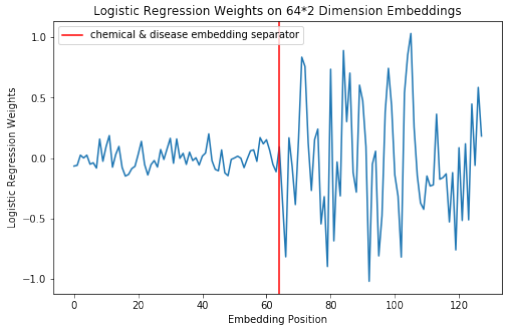


Figure 9. Weights from the classifier showing high magnitudes only for the disease portion of the concatenated edge embeddings.

Method	AUROC (test set)	Max Classification Accuracy (test set)
DeepWalk + Logistic Regression (random embeddings)	0.50	0.5062
DeepWalk + Logistic Regression (chemical embeddings)	0.57	0.5621
DeepWalk + Logistic Regression (disease embeddings)	0.94	0.9159
DeepWalk + Logistic Regression (chemical + disease)	**0.94**	**0.9171**
CTD Inference Score	0.56	0.6106

Table 1. Summary of performance of baseline model experiments vs. CTD inference score

From these preliminary results it is evident that my basic supervised learning approach is vastly superior to the CTD inference score in predicting which edges have a real curation (Table 1). Of course, this is not an entirely fair comparison, as my model was trained on such labels, and the CTD inference score was devised from domain knowledge about the local topology of the graph. Clearly the model captures obvious features present in the graph that are very different between the positive and negative edges in the training set, easily captured via DeepWalk, but not captured by the local topology definitions in the original CTD inference score.

End-to-End Model (Graph Auto-Encoders)

Even though the simple supervised approach produced impressive results, it still made some misclassifications. Furthermore, the use of DeepWalk to produce node embeddings, which are then concatenated to produce an edge representation, followed by fitting a logistic regression model, seems partly contrived. The DeepWalk loss function of predicting which nodes are on the same random walk also does not seem that relevant in the case of the C-G-D tripartite relationships. I therefore explored end-to-end models that directly do link prediction, instead of first building explicit node embeddings.

One attractive family of methods that can perform end-to-end edge prediction is the Graph Auto-Encoder (GAE). Much like non-graph autoencoders, GAEs learn latent node and edge representations by first using an encoder to reduce the graph to a reduced representation, and then via a decoder, reconstructing the graph structure, such as the graph adjacency matrix. It is considered an unsupervised method as it only relies on the nodes and edges already present in the graph (however since the edges are our labels it can be considered supervised). The encoders and decoders vary widely depending on the application. Encoders can range from multi-layer perceptrons to convolutional graph layers, LSTM networks, etc, and decoders can be multi-layer perceptrons as well, or a simple similarity measure between two node representations. As I am interested in the problem of link prediction, I will utilize the methods described in the paper "Variational Graph Auto-Encoders" which produced state-of-the-art results in link production tasks with datasets such as Cora, Citeseer, and Pubmed (Kipf and Welling, 2016).

The GAE in this case involves an encoder with two graph convolutional layers, which take the form:

$$\boldsymbol{Z} = enc(\boldsymbol{X}, \boldsymbol{A}) = G_{conv}(ReLU\big(G_{conv}(\boldsymbol{A}, \boldsymbol{X};\ \boldsymbol{\Theta}_1)\big);\ \boldsymbol{\Theta}_2)$$

Where $\boldsymbol{Z}$ denotes the network embedding matrix of a graph with nodes feature matrix $\boldsymbol{X}$ and adjacency matrix $\boldsymbol{A}$. G_{conv} is a standard graph convolutional layer, and $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_1$ are the weight matrices learned in the two graph convolutional layers. The GAE decoder attempts to decode the latent relationships between nodes into the graph adjacency matrix, upon which the reconstruction loss can be calculated. A natural way to model pairwise similarities between nodes is the inner product of the node embeddings:

$$\widehat{\boldsymbol{A}}_{v,u} = dec(\boldsymbol{z}_v, \boldsymbol{z}_u) = \sigma(\boldsymbol{z}_v^T \boldsymbol{z}_u)$$

$$L = CrossEntropy(\boldsymbol{A}, \widehat{\boldsymbol{A}})$$

Where $\boldsymbol{z}_v$ is the embedding of node *v*. GAE is then trained by minimizing the negative cross entropy given the real adjacency matrix $\boldsymbol{A}$ and the reconstructed adjacency matrix $\widehat{\boldsymbol{A}}$.

In the same paper Kipf and Welling also introduced the Variational Graph Auto-Encoder (VGAE), which aims to reduce overfitting in GAE by designing a loss function to allow the autoencoder to learn the distribution of the data instead of the data itself, by learning the parameters μ and σ from a $N(0,1)$ distribution, making the model more generative. However, for the CTD dataset my experiments with VGAE actually yielded worse test results than GAE.

However, the GAE cannot be directly used on the CTD dataset. In all of the canonical link prediction datasets, there is only one node type and one edge type, therefore the model is trained on the same type of edges that it is tested on. However, CTD is a tripartite graph with three different types of edges. Furthermore, we would ideally like to train on the C-G and G-D edges and predict the C-D edges. Therefore, we want the loss function to only penalize wrongly predicted C-D edges, and not the two other edge types. Therefore, I devised an alternative formulation of the GAE in which the encoder only sees the C-G-D graph without any C-D edges, and the decoder, when reconstructing the adjacency matrix, is only scored on how well it reconstructed the C-D edges.

$$\boldsymbol{Z} = enc(\boldsymbol{X}, \boldsymbol{A}_{CG,GD}) = G_{conv}(ReLU\left(G_{conv}(\boldsymbol{A}_{CG,GD}, \boldsymbol{X}; \boldsymbol{\Theta}_1)\right); \boldsymbol{\Theta}_2)$$

$$\widehat{\boldsymbol{A}}_{v,u} = dec(\boldsymbol{z}_v, \boldsymbol{z}_u) = \sigma(\boldsymbol{z}_v^T \boldsymbol{z}_u)$$

$$L = CrossEntropy(\boldsymbol{A}_{CD}, \widehat{\boldsymbol{A}_{CD}})$$

The formulation is the same, except the encoder is only concerned with C-G and G-D edges, while the loss function is only concerned with the C-D edges. I trained this GAE formulation by splitting the C-D edges into train, validation, and test, with accompanying negatively sampled C-D pairs as negative edges. The entire C-G and G-D edges are fed into the model without splits. Unfortunately, this model formulation was not able to converge. A reason for this is that as C-D edges are a relative minority in the entire graph (one sixth of all edges), a loss function that only considers C-D edges will not allow the autoencoder to learn meaningful representations of the graph as a whole, making it impossible to reconstruct the adjacency matrix.

Nevertheless, I tried a second approach by combining all C-G, G-D, and C-D edges into one big graph, and trained GAE on the entire graph. Now, the train, validation and test splits are done across all three edge types, and the loss function will penalize reconstruction differences in all three edge types. Even though this model is no longer making C-D specific predictions, it is able to make edge predictions on all edges in the graph, arguably more powerful. The model well in this task, successfully learning embeddings for all three relationships, and improved performance on the prediction of the subset of C-D edges in the test set. With GAE it achieved 0.9567 maximum accuracy on the C-D edges (Table 2).

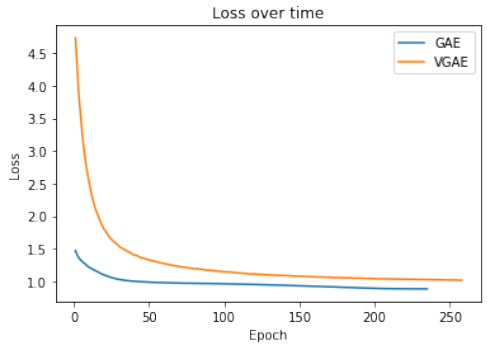


Figure 10. Training loss over time for GAE and VGAE.

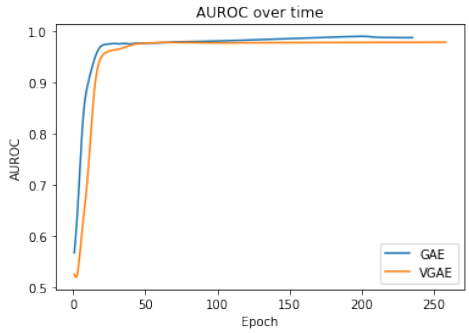


Figure 11. Training AUROC over time for GAE and VGAE.

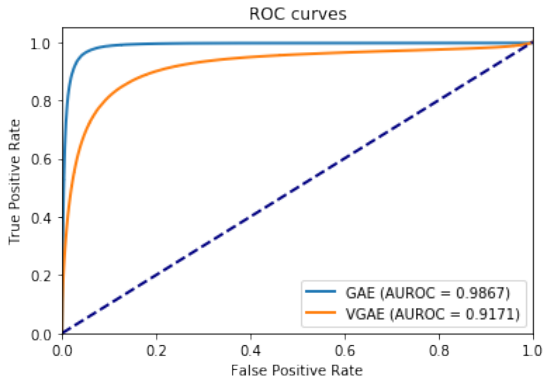


Figure 12. ROC curves for final GAE and VGAE.

In general, GAE achieved better results than the combination of DeepWalk embeddings and the Logistic Regression classifier, albeit the baseline had good results already. Both are superior to the CTD Inference Score (Table 2). GAE may be more successful for several reasons. First of all, GAE, unlike DeepWalk, allowed the inclusion of node features, for which I added only one feature – the type of the node (Chemical, Gene, or Disease). The model can do much more if it understands the tripartite nature of the graph. Second of all, the GAE encoder relies on graph convolutions, which is a natural way to aggregate neighborhood information in our C-G-D graph, since most relationships are 1-2 levels away, whereas random walks seem less suitable for our context. Third of all, the inner product decoder of the GAE trained on adjacency matrix reconstruction can learn to combine node embeddings in a more task-dependent way than the simple concatenation of node embeddings in our baseline. Table 2 summarizes the results of the second set of experiments compared with baseline and CTD Inference Score.

Method	**AUROC (test set)**	**Max Classification Accuracy (test set)**
GAE with CG, GD encoder, CD loss function	N/A	Failed to converge
GAE with CG, GD, CD encoder, CG, GD, CD loss	**0.99**	**0.9567**
VGAE with CG, GD, CD encoder, CG, GD, CD loss	0.92	0.8571
DeepWalk + Logistic Regression (chemical + disease)	0.94	0.9171
CTD Inference Score	0.56	0.6106

Table 2. Summary of performance of all classifiers.

Conclusion

The goal of the project was to derive better inference scores for C-D relationships in the Comparitive Toxicogenomics Database (CTD). Compared with the current unsupervised CTD inference score based on feature engineering on local topology, this project was successful in producing better inference scores (via binary classification probabilities on edges) on two fronts. First, both DeepWalk in the baseline model and the Graph Auto-Encoder allowed organic incorporation of higher order graph structure larger than one chemical, one disease, and the genes that connect them. This embedded each chemical, disease, and gene in the context of the entire graph. Second, this project allowed the incorporation of curated C-D edges with direct experimental evidence as supervision in training the models. The test set from supervised learning allowed for an objective measure to evaluate all models, including the original CTD Inference Score, removing the need to construct null hypothesis graphs to compare against. The simple Logistic Regression classifier on concatenated node representations generated via DeepWalk delivered greatly improved results over the CTD Inference Score, with the end-to-end Graph Auto-Encoder (GAE) improving results even further, though at a higher computational cost. There are many areas of improvement – GAE allows for features vectors to be attached to nodes (the only featured used was node type), which means rich contextual information about chemicals, genes, and diseases from other data sources can be encoded into the graph, potentially delivering even richer representations. Furthermore, more work could be done to investigate how best to sample negative edges within a tripartite graph such as CTD, as having better negative labels would greatly increase the validity of the classifier.

References

Davis, A.P., Grondin, C.J., Johnson, R.J., Sciaky, D., King, B.L., McMorran, R., Wiegers, J., Wiegers, T.C., Mattingly, C.J., 2017. The Comparative Toxicogenomics Database: update 2017. Nucleic Acids Res.

Davis, A.P., King, B.L., Mockus, S., Murphy, C.G., Saraceni-Richards, C., Rosenstein, M., Wiegers, T., Mattingly, C.J., 2011. The comparative toxicogenomics database: Update 2011. Nucleic Acids Res. https://doi.org/10.1093/nar/gkq813

Eyuboglu, E.S., Freeman, P.B., 2017. Disease Protein Prediction with Graph Convolutional Networks.

King, B.L., Davis, A.P., Rosenstein, M.C., Wiegers, T.C., Mattingly, C.J., 2012. Ranking Transitive Chemical-Disease Inferences Using Local Network Topology in the Comparative Toxicogenomics Database. PLoS One. https://doi.org/10.1371/journal.pone.0046524

Kipf, T.N., Welling, M., 2016. Variational Graph Auto-Encoders 1 A latent variable model for graph-structured data. NIPS Work.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res.

Perozzi, B., Al-Rfou, R., Skiena, S., 2014. DeepWalk: Online learning of social representations, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. https://doi.org/10.1145/2623330.2623732