TOWARDS THE INTEGRATION OF GENOMIC PROFILES AND GENE INTERACTION NETWORKS FOR MACHINE LEARNING

BY

HENRY A. LIN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Advisor:

Professor Jiawei Han

Abstract

With the advent of big data, scientists are collecting biological data faster than they have in the past, including genomic profiles which describe individuals by thousands of genes at a time. Adding to this library of knowledge are gene interaction networks, which model overarching cellular processes by describing how genes interact with each other.

When approached with genomic profile data together with gene interaction data, it becomes a question of how to integrate these two pieces of knowledge together for machine learning. Previous studies have attempted to employ some form of feature engineering process to “collapse” the network topology alongside the genomic profiles, losing the potential for global network information.

Instead, we explore a framework based upon network propagation. We explain how network propagation algorithms can enhance standalone genomic profiles, called **embeddings**, and show these enhancements lead to improved predictive accuracies on drug response classification. We next show that these embeddings contain predictive signals that are not necessarily implicated by gene ranking methods such as PageRank. Last, we apply network propagation to a dataset presented by the DREAM organization, and show we can improve a naïve linear regression that solves for a drug sensitive ranking task.

To the pursuit of knowledge,

to the quenching of curiosity,

to the fear of failure, and

to the perseverance to overcome it.

Acknowledgments

I first thank my advisor, Professor Jiawei Han, for helping me grow as a researcher. The skills and knowledge I've obtained during my Master's program in this university will prove invaluable to me in industry and future research careers I pursue. Next, I thank Professor Jian Peng, who was with me through this entire process of computational biology research. Professor Jian Peng was the one who introduced me to this entire area of computational biology in the first place.

Next, I want to thank the reviewers of this thesis, who took the time out of their day to help edit this work. This list of people contains: Professor Jiawei Han, Professor Jian Peng, Amin Emad, Yiding Hao, Sheng Wang, Joey (Zhou) Yi, and Jack Hou.

I'd also like to recognize everyone who has inspired me to dive further into research. This is a list in chronological order from when I met said person. I apologize to people if they believe they deserve to be mentioned here, but are not.

Yiding Hao	Joey (Zhou) Yi	Michael Hongyi Wu
Jeff Erickson	Sheng Wang	Jason Cho
Sailunsi Chen	Saurabh Sinha	Charles Blatti
Jiawei Han	Jack Hou	Doris Xin
Amin Emad	Jian Peng	

I also thank the scientific python community, both for bringing the world and myself such wonderful research software, but also helping me find that place where software engineers and researchers happily exist.

And of course, my father and my mother, who set me on this course of working towards my goals, and think about me everyday. My brother and my sister, who have had to put up with me for all these years. I think I turned out okay, by their standards.

Table of Contents

Chapter 1

Introduction

Over decades of research, biologists have curated and published data of heterogeneous types for use in future studies. In the context of machine learning, we wish to gain insight into these datasets by building predictive models from these datasets. For example, some studies (Friedman et al. (2009)) employ regression algorithms to understand what genes lead to a particular phenotype, such as drug response.

However, underlying interactions between genes in the data pose a problem for classical methods such as linear regressions and classifiers, which learn univariate models between the genes. In other words, most popular linear models assume each of these genes is independent of each other, and weigh their importance as such. In reality, genes are known to interact with one another, and we ideally want to construct a model which captures the interactions between features.

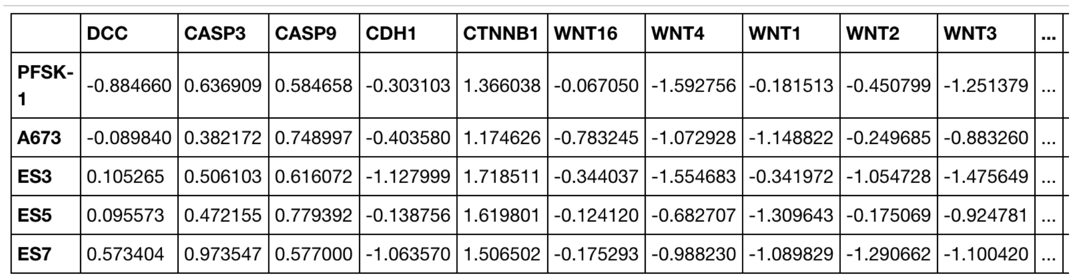
New research has now focused on the idea of **embedding**. Embedding is the concept of transforming one or several objects into a different space. For example, researchers have developed a software package called Word2Vec for natural language processing which transforms words into vector representations (Mikolov et al. (2013)). For our purpose, what we are interested in is embedding our genomic profiles, whose genes are connected by interaction networks, into some representation that can aid our linear models.

1.1 Problem Statement

Assume the standard problem setting, where we are given genomic profiles $X \in \mathbb{R}^{N \times g}$ describing N cell line samples such that each $X_{i1}, \ldots, X_{ig}$ is associated with some gene in some set $\mathcal{G}_X$. Figure 1.1, for example, shows part of a dataset which describes cell lines using its genes. Each cell line / gene pair corresponds to the gene microarray expression data ("expression data", for short).[1]

When we have a set of responses, $y \in \mathbb{R}^N$, we can fit an off-the-shelf linear model to this data. However, as mentioned before, these genes have an underlying interaction between them, which may not be captured in our

[1]Cell lines act as proxies to individuals. For example, we can associate a tumor cell line with a cancer patient of whom we would like to cure.
Because each row of our dataset X is said to be a learning *sample* (or example) in a machine algorithm, we may interchange the terms "cell line", "sample", "cell line sample", or even "patient".



	DCC	CASP3	CASP9	CDH1	CTNNB1	WNT16	WNT4	WNT1	WNT2	WNT3	...
PFSK-1	-0.884660	0.636909	0.584658	-0.303103	1.366038	-0.067050	-1.592756	-0.181513	-0.450799	-1.251379	...
A673	-0.089840	0.382172	0.748997	-0.403580	1.174626	-0.783245	-1.072928	-1.148822	-0.249685	-0.883260	...
ES3	0.105265	0.506103	0.616072	-1.127999	1.718511	-0.344037	-1.554683	-0.341972	-1.054728	-1.475649	...
ES5	0.095573	0.472155	0.779392	-0.138756	1.619801	-0.124120	-0.682707	-1.309643	-0.175069	-0.924781	...
ES7	0.573404	0.973547	0.577000	-1.063570	1.506502	-0.175293	-0.988230	-1.089829	-1.290662	-1.100420	...

Figure 1.1: Example gene expression data from the GDSC (Yang et al. (2013))
Each row i represents the microarray gene expression for a tumor cell line i, while each column j represents the microarray gene expression for a given gene j across all cell lines. The set of all genes in this data X is denoted as $\mathscr{G}_X$.

linear model. We now introduce the underlying network, G, whose genes form the set $\mathscr{G}_G$.[2] What we wish to do is capture these underlying gene interactions within our original genomic profiles.

1.2 Related methods which solve different tasks

As we have alluded to previously, there have been many studies trying to utilize these underlying networks and their genes. For example, Paradigm-shift (Kalia and Gupta (2005)) both try to use the network data to understand perturbations in networks due to somatic gene mutations. DCA and ClusDCA (Cho et al. (2015); Wang et al. (2015)) both try to embed each gene into some feature vector x for protein function prediction. LINE (Tang et al. (2015)) is also another gene embedding algorithm which can use the local context and similar graph substructures to predict a gene's function in the network.[3] But we emphasize that these algorithms do not solve the task at hand. Remember that we are trying to represent an individual with a genomic profile; our individuals cannot be represented by single genes.

An algorithm which more closely represents what we would like to do, but still solves a different task, is DawnRank (Hou and Ma (2014)). Dawnrank tries to find "personalized" driver genes by ranking these genes in a network. The algorithm is able to achieve personalization by using a cell line's unique gene expression profile and somatic mutation data. Though our algorithm does have a personalization component to it, we are not trying to rank genes, but to embed this data into a form of which machine learning models can use.

[2]Note that, in some cases, $\mathscr{G}_X \neq \mathscr{G}_G$.
[3]Results for LINE on protein function prediction have not yet been published.

1.3 Related network embedding works

One method that utilizes network information to improve a machine learning model is the generalized elastic net algorithm from Sokolov et al. (2016). In this work, the authors created a regularizer based upon the graph laplacian L of a gene network G. This problem formulation is different than our vision of embedding and enhancing the samples, though one could argue it seems more simple and takes into account the network topology into the machine learning tasks. Unfortunately, we did not compare our method to this method, in part because the published R package does not function correctly as expected.

More approaches that try to take advantage of the network structure into predictions are evaluated in Staiger et al. (2011). These algorithms tried to perform feature engineering to “collapse” the underlying networks into feature vectors for breast cancer survival prediction. Surprisingly, what the authors discuss disagrees with our hypothesis. The authors discuss how under strict evaluation settings, and with consistent datasets across all tests, the genomic networks actually do not improve the classifiers mentioned in the paper. In one of the tests, the authors generate a random network, and show that even while using this random network, there is no significant decrease in performance across the evaluated classifiers.

Nonetheless, the classifiers discussed in Staiger et al. (2011) do not use network propagation for integrating the genomic profiles and networks together. And we will show in this work that this method improves the results of baseline linear models. Chapter 2 discusses network propagation algorithms, and how we build our framework around them. Chapter 3 reveals some results using this method for a drug response prediction task. Our last chapter, chapter 4, introduces this method for cell line ranking using a famous dataset curated by the NCI-DREAM organization.

Chapter 2

Network propagation across samples

In this section, we discuss the PageRank algorithm, a random walk with restart (RWR) algorithm. RWR algorithms provide the foundation for network propagation. We then discuss previous uses of the PageRank algorithm in other studies, and explain why our approach is different.

2.1 The PageRank Algorithm

We define **network propagation** as the process used to diffuse heat through a network using a RWR algorithm such as PageRank. We review the network propagation algorithm, as described in Vanunu et al. (2010).

Let $G = (V, E)$ be a weighted directed graph with N nodes, and let $W \in [0, \infty)^{N \times N}$ be the corresponding **weight matrix**, such that if edge $i \to j$ has weight w, then $W_{ij} = w$. We will assume in this thesis that if $W_{ij} = 0$, edge $i \to j$ effectively does not exist in G. (Observe that we can adapt this definition to unweighted networks by setting each $W_{ij} = 1$.) Furthermore, in the case that G is undirected, we can assume that W is symmetric.

We first want to form a row stochastic matrix $\widetilde{W}$[1]. We can intuitively think of $\widetilde{W}_{ij}$ being the probability that a particle starting from node i will transition to node j. If each node in G has an outgoing edge, we can define $\widetilde{W}$ by the following.

Definition 1

Assuming W is the weight matrix of G, and each node in G has an outgoing edge (which is usually not the case), define the ***transition matrix*** *$\widetilde{W}$ as*

$$\widetilde{W}_{ij} = \frac{W_{ij}}{\sum_j W_{ij}}.$$

We now consider the case when G contains nodes with no outgoing edges, called **dangling nodes**. If G contains danging nodes, we can modify the weight matrix W of G by setting $W_{ij} = 1$ for each j, if i is a dangling node.

[1]A matrix $\widetilde{W} \in [0, 1]^{N \times N}$ is **row stochastic** if $\sum_j \widetilde{W}_{ij} = 1$ for each row i.

Applying definition 1 gets us our transition matrix of interest.[2]

With these definitions, the following is the PageRank algorithm from Page et al. (1998).

Definition 2

The ***PageRank recurrence*** *is given by*

$$p_{t+1} = (1-\alpha) \cdot p_t \cdot \widetilde{W} + \alpha \cdot u$$

where $u = (1/N, \ldots, 1/N)$ is a vector of length N, and $\odot$ represents element-wise multiplication. p_0 is set to be a vector representing any probability distribution.

As $t \to \infty$, this algorithm provably converges to a unique solution (regardless of p_0). When it does, p_∞ is the ***equilibrium distribution****, denoted as p for simplicity.*

The three components of interest in definition 2 are p, α, u.

- p, the equilibrium distribution, represents how heat is spread across the network after being introduced. Because p is a distribution, $\sum_i p_i = 1$, and $p_i \geq 0$.

- α can be thought of as a "smoothness" parameter. It is referred to as the "restart probability" in Page et al. (1998), and it describes how far heat is allowed to travel starting from its source node. It is chosen based upon the network itself (Leiserson et al. (2015); Hofree et al. (2013)).

  The authors in Leiserson et al. (2015) state that their network propagation algorithm works best if large values of α (≈ 0.5) are chosen, since this helps capture local structure in the network. However, because we distribute our heat using u, we can capture global structure as well. We believe this is one of the advantages of using network propagation over the methods described in Staiger et al. (2011).

- u is the personalization distribution granted to each node in the network. Given some fraction of heat u_i, u_i heat is transferred back to gene i when the walk restarts.

  u can be thought of as a prior knowledge vector (Hofree et al. (2013); Vanunu et al. (2010)), and the RWR moves this "prior knowedge" across the network, leading to a "smooth" distribution of node probabilities.

We describe **network propagation** as the process of how different personalization distributions u will result in different distributions of heat across the network. By *biasing* the RWR with this personalization, one can integrate outside knowledge not originally part of the genomic network.

[2]One difference between our algorithm and the formulation in Vanunu et al. (2010) is that in their work, the authors set $\widetilde{W} = D^{-1/2} W D^{-1/2}$. This product is similar to a graph laplacian for directed networks. $\widetilde{W}$ in this case is usually not row stochastic, but will still cause the algorithm in 2 to converge.

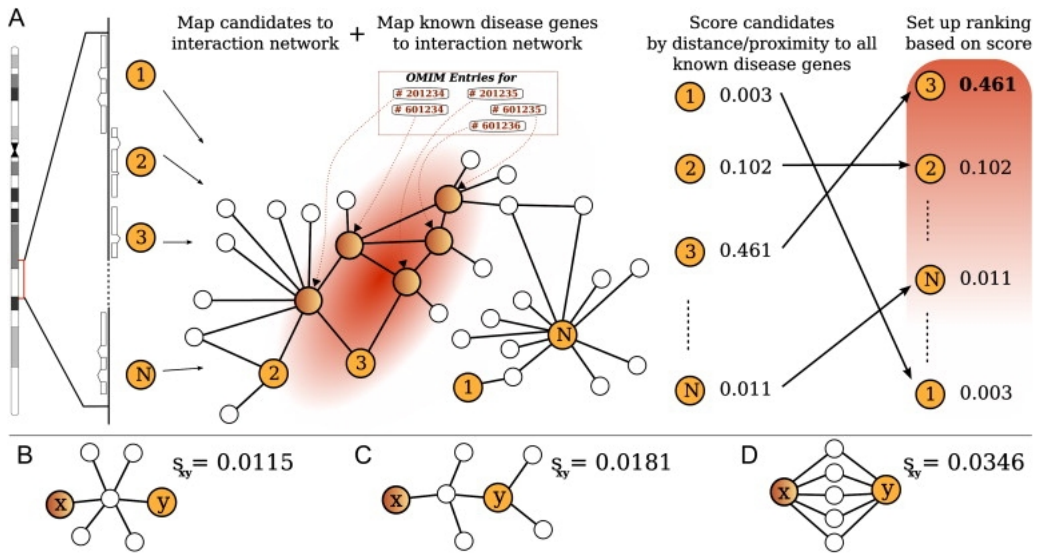

A
Map candidates to interaction network
+
Map known disease genes to interaction network
OMIM Entries for
# 201234
# 201235
# 601234
# 601235
# 601236
Score candidates by distance/proximity to all known disease genes
Set up ranking based on score
0.003
0.102
0.461
0.011
0.461
0.102
0.011
0.003
B
s_{xy} = 0.0115
C
s_{xy} = 0.0181
D
s_{xy} = 0.0346

Figure 2.1: Mapping genes to diseases.
Köhler et al. (2008) demonstrates how the results of a RWR algorithm can rank genes in a network.

For the rest of this work, we refer to the equilibrium distribution p as an **embedding**. We will provide our explicit framework in 2.3, and show results in section 3.2 and chapter 4.

2.2 Previous uses of network propagation

In bioinformatics, one of the challenges is annotating genes with their functions, and understanding how genes cause certain diseases. Many scientific studies (Weston et al. (2004); Köhler et al. (2008); Vanunu et al. (2010); Leiserson et al. (2015); Hou and Ma (2014), and others) have previously used network propagation to rank genes in a network. The most basic of ranking schemes is simply by biasing the RWR with genomic data by changing u. The work by Köhler et al. (2008) in figure 2.1 demonstrates this process.

However, we are not interested in ranking genes in this work. Though we are interested in using network propagation, we want to use network propagation to embed a cell line's information alongside the genomic network. The work algorithm "network based stratification" (NBS) most similarly demonstrates this process (Hofree et al. (2013)). In Hofree et al. (2013), the authors clustered cancer subtypes using the mutation profiles of patients. For each of the N cell lines, the authors performed a RWR by using the patient's mutation profile to modify u, creating N unique embeddings. Using the N embeddings, the authors used non-negative matrix factorization to cluster these cell lines into cancer subtypes.

As far as we know, this may be one of the first instances where a sample's embedding was used as input into a machine learning model. And so we build off of NBS, and try generalizing this framework for other machine learning tasks.

2.3 The embedding framework

Our framework for embedding network information in our genomic profiles is shown in algorithm 1. We also depict our framework using figure 2.2 For brevity, we omit the details about preprocessing and postprocessing, but these can be found in the supplements of this work. (See section 6.1.)

The first preprocessing step removes genes from X that are absent from the network G. It also imputes values for genes not in X which are present in G. From here, any RWR algorithm (such as PageRank) can be used to embed the samples.

The postprocessing step on line 4 is crucial for the use of a machine learning algorithm on line 5. Since the output of the embeddings are probability values, we can think of each feature (a probability on a gene) to be used in classification as dependent on each other; we have the constraint that $\sum_j P_{ij} = 1$. One of the main functions of the postprocessings step is to decouple each feature for use in a predictive learning model.

Furthermore, the postprocessing step selects ideal genes for use in classification. Some studies in the past (see the supplements of Costello et al. (2014)) have attempted to use the top k genes chosen from a RWR as the canonical features to use in a machine learning algorithm. Instead, we use a supervised feature selection procedure (χ^2 feature selection) to select the best features for use in the model. We will show that this produces better results than choosing the top genes with the greatest ranks (see section 3.3.2).

We clarify that this framework simply uses network propagation algorithms in a different way than used previously for sample level machine learning. It is not meant to be complex, but simply an outline of how we envision embedding frameworks to act using network propagation.

We can generalize and state that NBS follows this pipeline, except instead of classification the authors perform clustering. The authors use gene mutation profiles to generate their personalizations, U, and postprocess their output features using quartile discretization along each feature.

DawnRank also follows this pipeline, up to the postprocessing step. The preprocessed inputs include the absolute values of the z-scored gene expression values, scaled to sum to one.

Algorithm 1 Network Enhanced Model

```
1: procedure TrainNetworkModel($G, X, y$)
2:     $\mathscr{G}_G \leftarrow$ GetGenes($G$)                          ▷ Genes in the network
3:     $\mathscr{G}_X \leftarrow$ GetGenes($X$)                          ▷ Genes in the profile
4:     $U \leftarrow$ Preprocess($X, \mathscr{G}_G, \mathscr{X}_G$)
5:     $P \leftarrow$ EmbedSamples($G, U$)         ▷ Pagerank, or any other RWR algorithm
6:     $X' \leftarrow$ Postprocess($P$)
7:     return TrainModel($X', y$)                  ▷ $y$ might not exist for clustering
8: end procedure
```

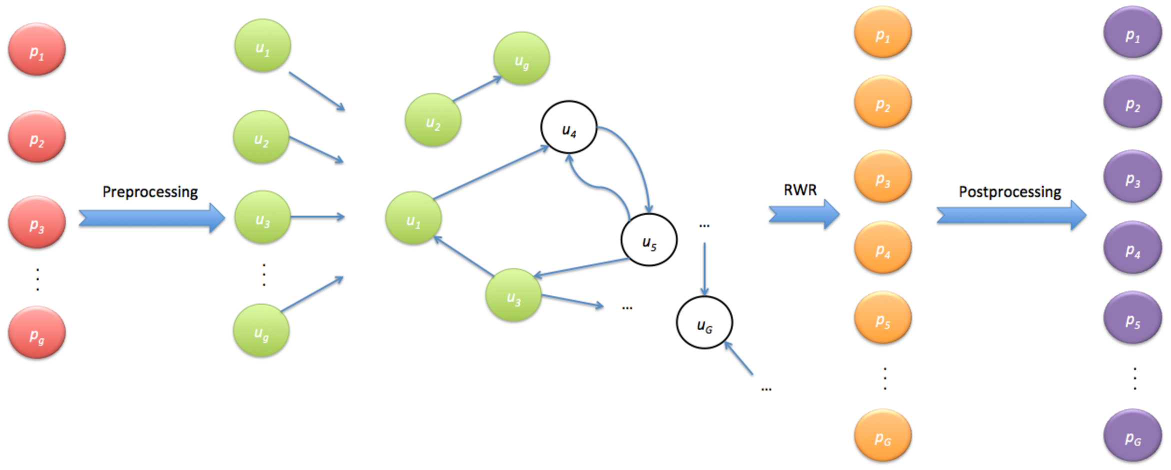


Figure 2.2: Framework outline
Red: Original genomic data (X). Green: Preprocessed data (U). White nodes in network: genes introduced by network data without genomic data. Orange nodes: The network embeddings produced by a RWR (P). Purple: postprocessed features ready to be used for predictive modeling (X').

Chapter 3

Experimental results

In this chapter, we demonstrate the usefulness of our method to a drug responses task (section 3.2). We then analyze the features produced by the embeddings, and show that they have more signal than the original genomic profile (3.3).

3.1 Data and task introduction

We used curated data collected from the Genomics of Drug Sensitivity in Cancer (GDSC) website (Yang et al. (2013)). This data contains inhibitory concentration[1] across cell lines and drug samples.

We chose to classify whether a drug d's IC_{50} value for cell line c exceeded the recommended dosage of d. Furthermore, we limited our classification task to one drug (midostaurin) for consistency. At the end, our drug response data contains 2283 genes, 592 samples, 309 negative examples, and 283 positive samples. Each gene is associated with a corresponding z-scored gene expression value. For brevity, we refer to this dataset X as the **drug response** data.

We conducted experiments across two different network datasets, both originating from the Kyoto Encyclopedia of Genes and Genomes (KEGG) database. (See table 3.1.) Network 1 was published with Hou and Ma (2014), and Network 3 has not yet been published. Last, we formed networks 2, 4, and 6 ourselves. The idea behind networks 2 and 6 was mainly from Leiserson et al. (2015), where the authors only considered one connected component to find gene subnetworks.

It's important to note that it's possible we won't have gene expression values for each gene in the network, and conversely genes in the network might not have associating gene expression values. See chapter 6.1 to see the full details of how we handle these missing values.

Last, we evaluated each of our models using F1 score, accuracy score, and precision. With the consideration that in the field of precision medicine, it's important that we minimize the number of false negatives as possible. Hence, precision and F1 score are ideal metrics for this task.

[1]A cell line is said to have an inhibitory concentration x value (IC_x) of c for some drug d if a concentration of c is needed to kill x percent of the cell line using d.

Network ID	**Description**	**Directed**	**# Genes**	**# Self Loops**	**# Edges**	**# Components**	**# Dangling**	**# Solitary**
1	Aggregated Kegg Network	Yes	8726	1178	155900	185	1896	70
2	Largest Component of 1	Yes	8377	1175	149542	1	1698	0
3	Kegg Network 2	Yes	2283	55	14916	487	782	461
4	Largest Component of 3	Yes	1700	42	14688	1	284	0
5	Merged Network	Yes	8892	1218	165707	271	1907	157
6	Largest Component of 5	Yes	8458	1213	165516	1	1626	0

Table 3.1: Network data information
Information about the network data in question. “Solitary” nodes are nodes that lack both incoming and outgoing edges.

Network ID	**# Overlapping Genes**	$\lvert \mathscr{G}_G \setminus \mathscr{G}_X \rvert$	$\lvert \mathscr{G}_X \setminus \mathscr{G}_G \rvert$
1	2117	6609	166
2	2091	6286	192
3	2283	0	0
4	1700	0	583
5	2283	6609	0
6	2171	6287	112

Table 3.2: Overlapping network and drug response information
$\lvert \mathscr{G}_G \setminus \mathscr{G}_X \rvert$ is the number of genes the network has that the drug response information does not have, and $\lvert \mathscr{G}_X \setminus \mathscr{G}_G \rvert$ is the number of genes the drug response information has that the network does not have.

Table 3.3: Drug response prediction accuracy

F1 score	Accuracy	Precision	# Genes selected
59.43	63.18	63.25	2283 (all)

Network ID	Restart probability	F1 score	Accuracy	Precision	# Genes selected
1	0.60	**60.04**	**63.35**	**63.58**	6980
2	0.55	**60.27**	**63.68**	**63.78**	6701
3	0.70	**59.93**	**63.35**	**63.23**	1598
4	0.60	58.90	61.83	61.60	510
5	0.55	**60.26**	**63.35**	**63.65**	3556
6	0.60	59.39	63.00	63.15	6776

3.2 Results for drug response classification

We evaluated an L2 regularized logistic regression using only drug response data. See section 6.2 for full details about hyperparameter tuning this baseline model. Just like we do for our network propagation tasks, we also applied feature selection before training the logistic regression. Figure 3.1 contains a plot of the feature selection search on the gene drug response data.

Next, using the same regularization value as our baseline logistic regression, as well as the same regularization (L2), we conducted our experiments across all six of our networks. Figure 3.2 contains the feature selection searchs on the embeddings.

Last, we display the obtained accuracies in table 3.3.

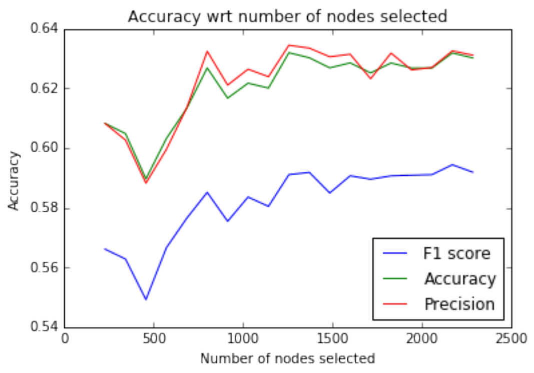


Figure 3.1: Feature selection search on the number of genes selected from the data, step size 0.05% of the genes.

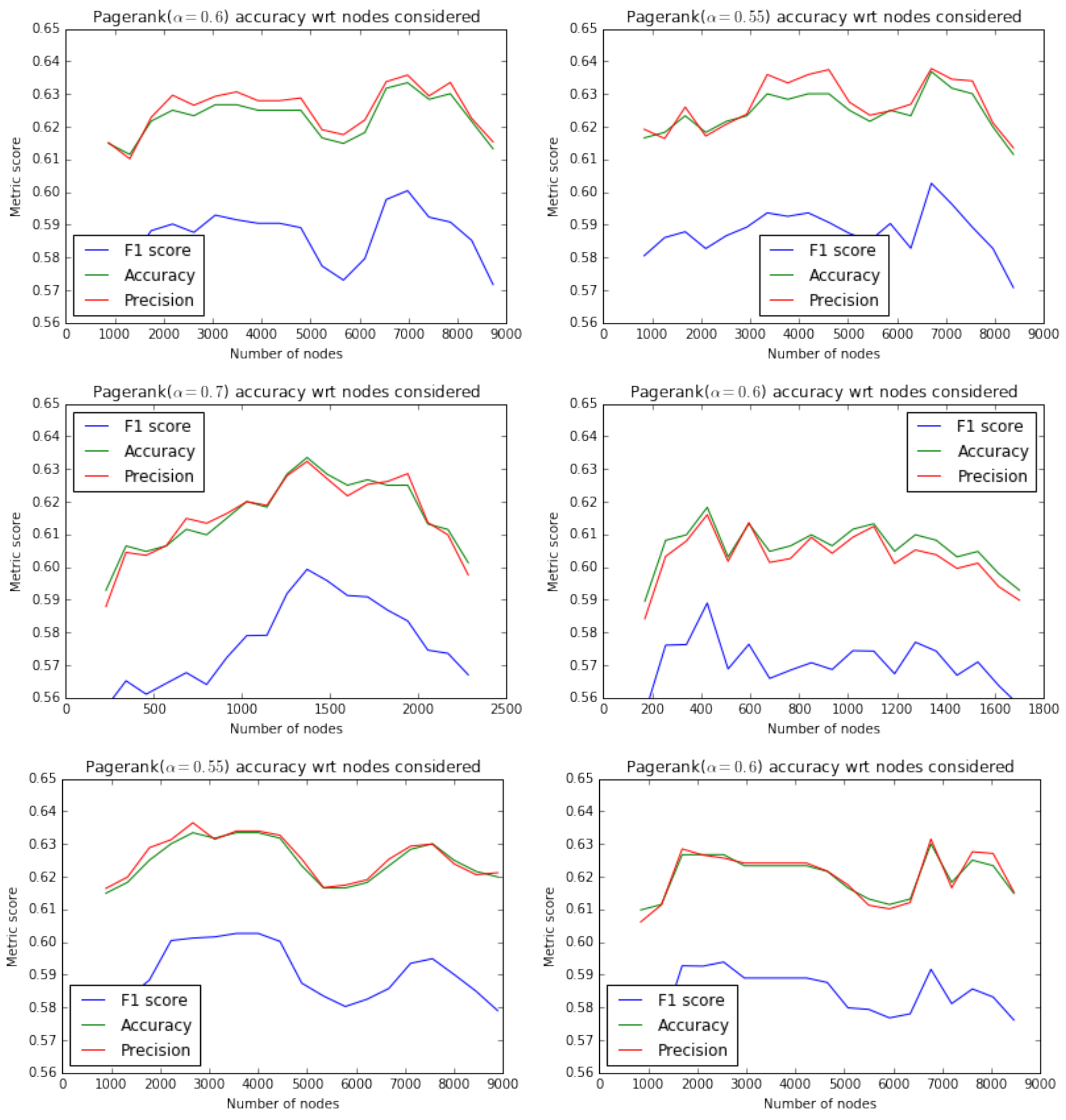


Figure 3.2: Feature selection search on the number of genes selected from the embeddings. Row one: using networks 1 and 2. Row two: using networks 3 and 4. Row three: using networks 5 and 6

As we can see from table 3.3, using networks 1, 2, 3, and 5 yields an improvement over baseline accuracies.

We also observe that there was an increase of accuracy between networks 1 and 2, but a decrease in accuracies between 3 and 4, and 5 and 6. Referring back to table 3.1, it seems we pruned 349 genes when producing Network 2 from Network 1, 434 genes from Network 3 to Network 4, and 583 genes from Network 5 to Network 6. So perhaps there is some threshold where overpruning the network does damage to our prediction accuracies.

3.2.1 Improving the embeddings using DirichletRank

One problem that Larry Page and Sergey Brin identified in their original formulation of PageRank (Page et al. (1998)) is the existance of dangling nodes. Recall a dangling node is a node that does not have any outgoing edges. The solution we discussed in section 2.1 is to allow this node to have outgoing edges to each other node in the graph.

However, as identified in Wang et al. (2008), this dramatically reduces the importances of these dangling nodes. In fact, the authors show that a node with one outgoing edge has much greater importance than one with none.

With some modifications to PageRank, one can produce DirichletRank, which is a very similar RWR algorithm. It more accurately weighs each node's restart probability with a "dynamic damping factor" μ, rather than a constant probability of restart α. Our results for the drug response task from section 3.2 are compared with those using DirichletRank instead of PageRank.

In the results, Network 6 now has accuracies that exceed the baseline. However, only Network 5's F1 score exceeds the baseline now.

3.3 Analyzing the embeddings

We now analyze the features. In this section, we show that the features produced by network propagation have more signal than the original gene expression features.

We limit this analysis to using only the embeddings produced by network 1, with $\alpha = 0.55$.[2] We also apply χ^2 feature selection on the entire data set, rather than 5 fold cross validation. This makes it easier for us to discuss the representative genes from both of the datasets.

[2]We should have used $\alpha = 0.6$ in our analysis, since this produced the best results from section 3.2.

Table 3.4: Drug response prediction accuracy comparison of PageRank and DirichletRank
Top: Baseline results. Middle: PageRank. Bottom: DirichletRank

F1 score	Accuracy	Precision	# Genes selected
59.43	63.18	63.25	2283 (all)

Network ID	Restart probability	F1 score	Accuracy	Precision	# Genes selected
1	0.60	60.04	63.35	63.58	6980
2	0.55	60.27	63.68	63.78	6701
3	0.70	59.93	63.35	63.23	1598
4	0.60	58.90	61.83	61.60	510
5	0.55	60.26	63.35	63.65	3556
6	0.60	59.39	63.00	63.15	6776

Network ID	Dynamic Damping Factor	F1 score	Accuracy	Precision	# Genes selected
1	18	**60.73**	**63.86**	**64.47**	5671
2	19	**60.51**	**64.03**	**64.49**	3769
3	16	**60.05**	**63.52**	**63.61**	1369
4	5	58.72	61.49	61.13	425
5	14	59.96	63.01	63.16	5335
6	12	**59.63**	**63.18**	**64.01**	5074

3.3.1 Gene level analysis

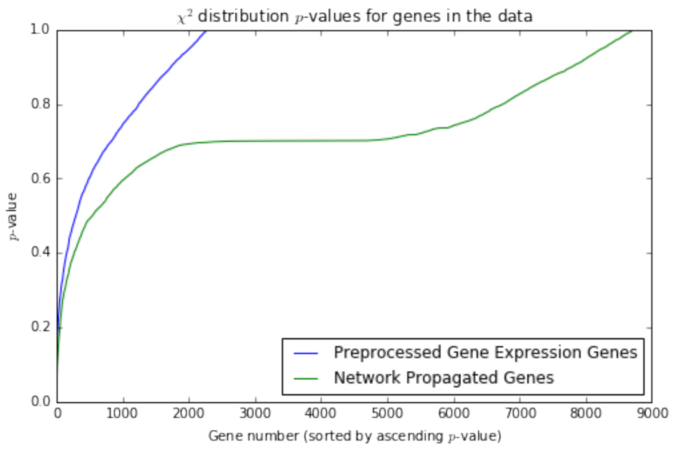


Figure 3.3: p-value distribution

Figure 3.3 shows us the distribution of each gene's p-values generated by a χ^2 distribution. Observe that the network embeddings add more features (genes) to our dataset. Therefore, one reason the classification accuracies of the network propagated classifier are higher than the baseline classifier is because we have augmented our feature set with more genes. Furthermore, some of these genes have more signal than the genes in the original drug response dataset.

We next want to compare the top twenty genes chosen by our feature selection. Figure 3.4 presents the top twenty genes chosen by the feature selection process. In general, we are quite pleased by the results in figure 3.4.

For example, some of the genes chosen by χ^2 feature selection did not have gene expression in the drug response dataset. This agrees with our intuition that information is passed from gene to gene, which could potentially make other genes in the network informative.

We also included information about the network itself to try explaining what topological features made these predictive genes. We observe that SPON1 is ranked as a predictive gene, but it only has 2 in-edges, and no gene expression data. One may assume that since SPON1 is a dangling node that it collected much of the heat in the network. However, our feature selection process would have not chosen SPON1 if its heat distribution remained the same across all cell line samples. Hence, SPON1 must be receiving heat from its neighbors, which causes SPON1 to be a predictive gene. More research must be performed to truly assess the predictive power of SPON1.

We also observe that MYL9, CTGF, THBS1, DCC, CAV1, COL6A2, CDKN2A are in the top selected genes for both the original drug response dataset as well as the embeddings. This implies is that our network propagations can not only capture genes implicated by network topology, but also by the original gene expression data.

Figure 3.4: Top 20 genes for both the original drug response dataset and network propagated dataset.

	p-value
Gene	
EHD2	0.023628
CTGF	0.054348
MYL9	0.055054
DDC	0.101544
COL6A2	0.110558
CAV1	0.119234
CDKN2A	0.123892
THBS1	0.127684
SERPINE1	0.127718
EPHA2	0.128812
CXCR4	0.131838
MME	0.149755
FN1	0.156744
COL18A1	0.172817
IL6	0.174192
CAV2	0.178251
GJA1	0.180613
PDGFC	0.181015
DAB2	0.186097
MRC2	0.196533

	p-value	Has Expressions	In degree	In degree percentile	Out degree	Out degree percentile	Self Loop	Average Neighbor Degree
Gene								
MYL9	0.035455	True	7	63	8	66	False	12
CTGF	0.052890	True	4	52	17	76	True	30
ITGBL1	0.054333	False	67	92	24	81	False	28
TAGLN	0.058682	False	6	60	2	40	False	1
LIMS1	0.066907	False	87	95	13	73	False	17
LIMS2	0.066907	False	87	95	10	69	False	18
NTN4	0.067044	False	79	94	7	63	False	23
CYR61	0.067967	False	9	68	10	69	True	54
MEGF9	0.068399	False	78	94	8	66	False	21
MICALL1	0.069012	False	85	94	8	66	False	21
ITGA11	0.080741	False	57	91	48	89	False	29
THBS1	0.082296	True	62	92	10	69	True	23
DDC	0.095170	True	0	7	1	29	False	6
SPON1	0.095997	False	2	38	0	10	False	0
POU5F1	0.102942	False	14	75	6	61	False	27
TNFAIP6	0.105424	False	7	63	1	29	False	0
CAV1	0.110187	True	36	87	159	97	True	59
COL6A2	0.111652	True	11	71	67	93	False	46
NOV	0.116715	False	6	60	1	29	False	3
CDKN2A	0.117393	True	16	77	37	86	False	77

Table 3.5: Subtask 1 results

	F1 score	Accuracy	Precision
Drug response gene expressions	59.53	63.35	63.56
Embeddings	**65.13**	**66.90**	**67.25**

Table 3.6: Subtask 2 results

	F1 score	Accuracy	Precision
Drug response gene expressions (see table 3.5)	59.53	63.35	63.56
Embedding induced feature selection	**62.08**	**65.38**	**66.00**
Embeddings (see table 3.5)	65.13	66.90	67.25

3.3.2 Using the χ^2 ranked features

The purpose of this section is to show that the features found by using χ^2 feature selection on the embeddings, X', are more informative than using χ^2 feature selection on the original drug response dataset, X.[3]

We perform three additional subtasks:

1. *Take the first 2283 ranked genes from both X and the X', and evaluate a logistic regression using these features.*

   Task one seeks to prove that having more genes (in quantity) in the network propagated dataset does not necessarily cause the classifier to be better. Rather, the improved accuracies are a result of, in part by, the *new genes* introduced in X'. Table 3.5 displays these results.

2. *Consider the first 2283 ranked genes from X'. Filter out the genes from X' that do not exist in X. Using these filtered genes, take the corresponding features from X (not X'), and evaluate a logistic regression on this subset of X.*

   Task two shows that network propagation isn't simply a data augmentation method. Rather, the genes chosen by network propagation are also informative in our original dataset X. Our results for this task are in table 3.6.

   These results actually help us understand how much signal the extra genes included from the network dataset provide for us.

3. *Consider the original probabilities P, and take the average of these probabilities for each gene. Use these averaged probabilities as a **rank** for a gene and evaluate a logistic regression on these rank selected features.*

   Task three simply shows that we cannot use RWR alone for feature selection. Our results in table 3.7 shows the supervised feature selection outperforms this unsupervised approach. These results are to be expected.

[3]Remember that we are performing feature selection before breaking the dataset up into folds. As a result, the accuracy scores in this section are not representative of the scores in the original task in section 3.2.

Table 3.7: Subtask 3 results

	F1 score	Accuracy	Precision	# Genes
Drug response gene expressions	59.53	63.35	63.56	2283 (all)
PageRank feature selection on embeddings	53.82	56.27	54.38	6108
PageRank feature selection on drug response	55.30	57.45	55.69	1397

Chapter 4

The NCI-DREAM 7 Challange

The NCI-DREAM 7 Challange (Costello et al. (2014)) was a predictive modeling challenge hosted in 2012 by the National Cancer Institute and the DREAM organization.[1] Their objective was to crowdsource a model that can predict how sensitive a breast cancer cell line is to a drug. To this end, the authors were trying to push towards *precision medicine*, building *in silico* models which are inexpensive to run in contrast with conventional wet lab tests.

4.1 DREAM 7 Data

Figure 4.1 outlines the data given to contestants, as well as the evaluation procedure. Contestants were given

- Gene expression data, gene mutation profiles, SNP[2] data, and other genomic data.
- A table of cell lines and 31 drugs, where each cell line / drug pair represented a cell line's GI_{50} value for the drug. A lower GI_{50} value represents a higher sensitivity to a drug.

The output of the algorithms were judged based upon 28 ranked lists of cell lines, each list induced by one of the 28 drugs used for testing. (3 from the original 31 were not included in evaluation.) The evaluation metric the authors used was the *weighted probabilistic concordance* index, which calculates a score between a "true" ranking and a "predicted" ranking. (See the supplements of Costello et al. (2014) for more information.) Because the GI_{50} values are real values, one way to view this problem is by multi-task regression.

4.2 Methods and results for DREAM 7

We used Network 1 from table 3.1 and only the gene expression profiles of the cell lines in our experiments. Between the genes in our network, $\mathcal{G}_G$ and the genes in the drug response data, $\mathcal{X}_G$, the two sets had 8628

[1] See http://dreamchallenges.org/project/closed/dream-7-nci-dream-drug-sensitivity-prediction-challenge/.
[2] Single nucliotide polymorphism

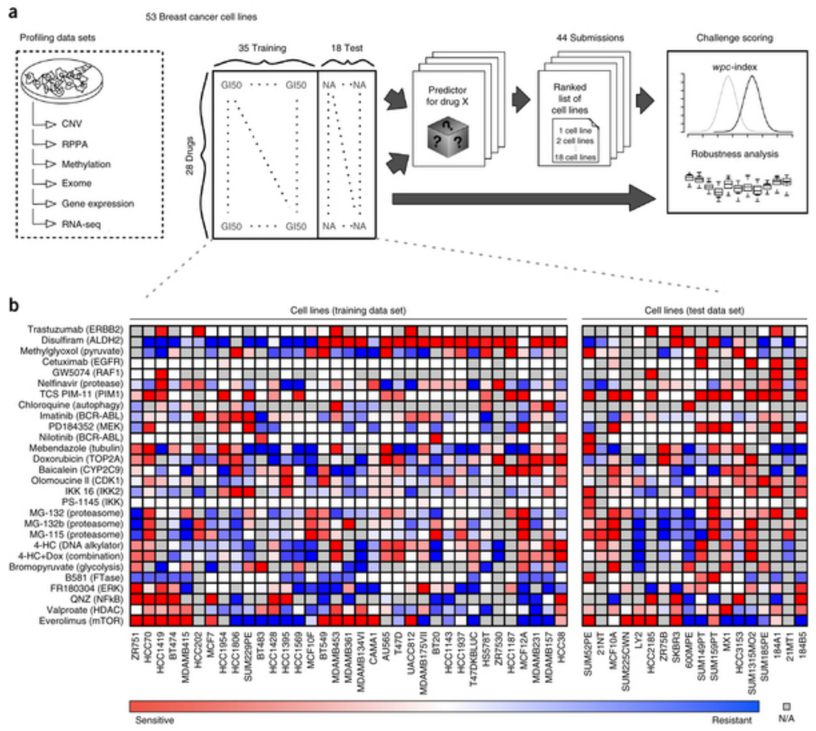


Figure 4.1: Dataset information and evaluation procedure outline for the DREAM 7 challenge

overlapping genes, but the network lacked 10004 genes which the drug response data contained. The drug response data X was only missing 98 genes from the network; we dropped these 98 genes from X altogether.

Next, we had to perform data imputation. Some of the cell lines in the testing set did not have gene expression values altogether. We set these expression values to the mean of the genes in question.

The drug response data was also lacking GI_{50} values which needed to be predicted. For this, we found that taking the max GI_{50} for a given cell line, and replacing each missing value for this cell line produced the best results.[3]

Next, we generated our embeddings using DirichletRank with a dynamic damping factor of $\mu = 14$. Our postprocessing consisted of min-max scaling each feature, however we did not perform feature selection in our postprocessing step.

In our modeling process, we used a simple linear regression to predict the GI_{50} values. Our results in 4.1 show that we were successfully able to improve the score of the baseline regressor simply by using network embeddings in our model.

Our baseline regressor would have scored 35th out of 44 in the competition, while our network propagated model would have scored 22nd. Choosing a different model, as well as including more than just gene expression

[3]We believe that imputation using univariate statistics per drug (as opposed to per sample) yielded a loss in performance because of severe overfitting.

Table 4.1: Our results on the DREAM 7 Challenge

	Data	WPC index
Baseline	Gene expression data	0.530
Network Propagation	Gene expression data, Network 1	0.549

data, may help our predictions in the future.

Chapter 5

Conclusion

In this thesis, we outline a method to enhance sample level gene profiles through use of network propagation. We showed in this work that by utilizing network structure, we can enhance the features used in machine learning methods to produce better predictive results.

We do believe, however, that there is more room to grow in this research area. For example, in our experiments, we chose to only use gene expression data, while there are many other kinds of genomic data available to biologists. Furthermore, we limited our analysis to KEGG pathway interactions, while there are many other networks available for analysis. Other networks may contain more data and interaction strengths on their edges.

Nonetheless, we hope that this framework be adopted for further research in computational biology. We hope that researchers will divert their attention from feature engineering methods like those mentioned in Staiger et al. (2011), and explore random walks to improve the results of their very own experiments.

Chapter 6

Supplements

6.1 The full network enhanced model pipeline

We omitted the entire pipleine for brevity in the main text. Observe that in PREPROCESS, we remove genes from X that are not in the network. Furthermore, we set the genes that are in the network but not X to zero. In most cases, X will represent z-scored gene expression values. Therefore, these padded zeros are actually converted to some other positive constant feature. We found that this imputation procedure performed well on our datasets.

For the χ^2 feature selection process, to select the ideal number of genes to use, we considered 5%, 10%, . . . , 100% percents of the number of genes in the datasets. Smaller step sizes provide better granularity in the number of genes chosen.

6.2 Hyperparameters for baseline logistic regression

Our baseline classifier for drug response prediction was an L2 regularized logistic regression, trained on only drug response data (microarray gene expression data). We chose an L2 regularizer over an L1 regularizer because we found that L2 yielded higher classification accuracies. Furthermore, our task was not feature selection, which L1 regularization is known for. Note that this accuracy comparison agrees with Andrew Ng's postuation in Ng (2004) that L2 regularization generally outperforms L1 regularization in accuracy when there are many features.

Next, we set our regularization constant C to $5/592$. Over grid search $(1/592, 5/592, 0.1, 0.2, \ldots, 1.0, 1.1, \ldots, 2.0)$, we found that this regularization constant yielded the best accuracy.

Note that this fractional value chosen is not arbitrary. The implementation of the logsitic regression we used is Liblinear's logistic regression Fan et al. (2008). This regularization value is equivalent to setting $\lambda = 1$ in Jerome Friedman and Trevor Hastie's formulation in Friedman et al. (2009).

Algorithm 2 Network Enhanced Model

```
1: procedure TrainNetworkModel($G, X, y$)
2:     $\mathcal{G}_G \leftarrow$ GetGenes($G$)                         ▷ Genes in the network
3:     $\mathcal{G}_X \leftarrow$ GetGenes($X$)                         ▷ Genes in the profile
4:     $U \leftarrow$ Preprocess($X, \mathcal{G}_G, \mathcal{X}_G$)
5:     $P \leftarrow$ EmbedSamples($G, U$)          ▷ Pagerank, or any other RWR algorithm
6:     $X' \leftarrow$ Postprocess($P$)
7:     return TrainModel($X', y$)                 ▷ $y$ might not exist for clustering
8: end procedure
```

Algorithm 3 Normalize data so each sample is a probability distribution u

```
1: procedure Preprocess($X, \mathcal{G}_G, \mathcal{X}_G$)
2:     Remove columns in $X$ that correspond to genes in $\mathcal{G}_X \setminus \mathcal{G}_G$.
3:     Add columns of zeros to $X$ that correspond to genes in $\mathcal{G}_G \setminus \mathcal{G}_X$
4:     Initialize $U$ as empty
5:     for sample $x \in X$ do
6:         Scale $x$ between $[0, 1]$
7:         $u \leftarrow x / \sum_i x_i$                              ▷ $\sum_i u_i = 1$
8:         Add $u$ to $U$
9:     end for
10:    return $U$
11: end procedure
```

Algorithm 4 Perform rescaling and feature selection on data

```
1: procedure Postprocess($X, y$)
2:     for $x_{ij} \in X$ do                             ▷ Min-max scaling per feature
3:         $M \leftarrow \max_{1 \leq i \leq N} x_{ij}$
4:         $m \leftarrow \min_{1 \leq i \leq N} x_{ij}$
5:         $x_{ij} \leftarrow \frac{x_{ij} - m}{M - m}$
6:     end for
7:     $X' \leftarrow$ FeatureSelection($X, y$)
8:     $X' \leftarrow 2 \cdot X' - 1$                     ▷ Spreads the data between [-1, 1]
9:     return X'
10: end procedure
```

References

Hyunghoon Cho, Bonnie Berger, and Jian Peng. *Research in Computational Molecular Biology: 19th Annual International Conference, RECOMB 2015, Warsaw, Poland, April 12-15, 2015, Proceedings*, chapter Diffusion Component Analysis: Unraveling Functional Topology in Biological Networks, pages 62–64. Springer International Publishing, Cham, 2015. ISBN 978-3-319-16706-0. doi: 10.1007/978-3-319-16706-0_9. URL `http://dx.doi.org/10.1007/978-3-319-16706-0_9`.

James C Costello, Laura M Heiser, Elisabeth Georgii, Mehmet Gonen, Michael P Menden, Nicholas J Wang, Mukesh Bansal, Muhammad Ammad-ud din, Petteri Hintsanen, Suleiman A Khan, John-Patrick Mpindi, Olli Kallioniemi, Antti Honkela, Tero Aittokallio, Krister Wennerberg, NCI DREAM Community, James J Collins, Dan Gallahan, Dinah Singer, Julio Saez-Rodriguez, Samuel Kaski, Joe W Gray, and Gustavo Stolovitzky. A community effort to assess and improve drug sensitivity prediction algorithms. *Nat Biotech*, 32(12):1202–1212, 12 2014. URL `http://dx.doi.org/10.1038/nbt.2877`.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent, 2009.

Matan Hofree, John P Shen, Hannah Carter, Andrew Gross, and Trey Ideker. Network-based stratification of tumor mutations. *Nat Meth*, 10(11):1108–1115, 11 2013. URL `http://dx.doi.org/10.1038/nmeth.2651`.

Jack P. Hou and Jian Ma. Dawnrank: discovering personalized driver genes in cancer. *Genome Medicine*, 6 (7):1–16, 2014. ISSN 1756-994X. doi: 10.1186/s13073-014-0056-8. URL `http://dx.doi.org/10.1186/s13073-014-0056-8`.

Anu Kalia and R. P. Gupta. Proteomics: A paradigm shift. *Critical Reviews in Biotechnology*, 25(4):173–198, 2005. doi: 10.1080/07388550500365102. URL `http://dx.doi.org/10.1080/07388550500365102`.

Sebastian Köhler, Sebastian Bauer, Denise Horn, and Peter N Robinson. Walking the interactome for prioritization of candidate disease genes. *American Journal of Human Genetics*, 82(4):949–958, 04 2008. doi: 10.1016/j.ajhg.2008.02.013. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2427257/`.

Mark D M Leiserson, Fabio Vandin, Hsin-Ta Wu, Jason R Dobson, Jonathan V Eldridge, Jacob L Thomas, Alexandra Papoutsaki, Younhun Kim, Beifang Niu, Michael McLellan, Michael S Lawrence, Abel Gonzalez-Perez, David Tamborero, Yuwei Cheng, Gregory A Ryslik, Nuria Lopez-Bigas, Gad Getz, Li Ding, and Benjamin J Raphael. Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. *Nat Genet*, 47(2):106–114, 02 2015. URL `http://dx.doi.org/10.1038/ng.3168`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013. URL `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`.

Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *In ICML*, 2004.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998. URL `citeseer.nj.nec.com/page98pagerank.html`.

Artem Sokolov, Daniel E Carlin, Evan O Paull, Robert Baertsch, and Joshua M Stuart. Pathway-based genomics prediction using generalized elastic net. *PLoS Computational Biology*, 12(3):e1004790, 03 2016. doi: 10.1371/journal.pcbi.1004790. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4784899/`.

C. Staiger, Sidney Cadot, R. Kooter, Marcus T. Dittrich, Tobias Müller, Gunnar W. Klau, and Lodewyk F. A. Wessels. A critical evaluation of network and pathway based classifiers for outcome prediction in breast cancer. *CoRR*, abs/1110.3717, 2011. URL `http://arxiv.org/abs/1110.3717`.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*. ACM, 2015.

Oron Vanunu, Oded Magger, Eytan Ruppin, Tomer Shlomi, and Roded Sharan. Associating genes and protein complexes with disease via network propagation. *PLoS Computational Biology*, 6(1):e1000641, 01 2010. doi: 10.1371/journal.pcbi.1000641. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2797085/`.

Sheng Wang, Hyunghoon Cho, ChengXiang Zhai, Bonnie Berger, and Jian Peng. Exploiting ontology graph for predicting sparsely annotated gene function. *Bioinformatics*, 31(12):i357–i364, 6 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btv260.

Xuanhui Wang, Tao Tao, Jian-Tao Sun, Azadeh Shakery, and Chengxiang Zhai. Dirichletrank: Solving the zero-one gap problem of pagerank. *ACM Trans. Inf. Syst.*, 26(2):10:1–10:29, April 2008. ISSN 1046-8188. doi: 10.1145/1344411.1344416. URL `http://doi.acm.org/10.1145/1344411.1344416`.

Jason Weston, Andre Elisseeff, Dengyong Zhou, Christina S Leslie, and William Stafford Noble. Protein ranking: From local to global structure in the protein similarity network. *Proceedings of the National Academy of Sciences of the United States of America*, 101(17):6559–6563, 04 2004. doi: 10.1073/pnas.0308067101. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC404084/`.

Wanjuan Yang, Jorge Soares, Patricia Greninger, Elena J Edelman, Howard Lightfoot, Simon Forbes, Nidhi Bindal, Dave Beare, James A Smith, I Richard Thompson, Sridhar Ramaswamy, P Andrew Futreal, Daniel A Haber, Michael R Stratton, Cyril Benes, Ultan McDermott, and Mathew J Garnett. Genomics of drug sensitivity in cancer (gdsc): a resource for therapeutic biomarker discovery in cancer cells. *Nucleic Acids Research*, 41 (Database issue):D955–D961, 01 2013. doi: 10.1093/nar/gks1111. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3531057/`.