AC297r Project: Harvard - Inari

Predicting the Effects of Genetic Perturbation in Maize

Milestone 3

Victor Avram, Sergio Miguel Moya Jimenez, M. Eagon Meng, Wenhan Zhang

April 27, 2021

Contents

1 INTRODUCTION AND MOTIVATION

As a plant breeding technology company, Inari's work comprises three high-level stages: (1) using computational methods to build genetic knowledge; (2) genetic editing; and (3) delivery of altered genetic information to specific parts of the plant. Through seeds genetic editing, Inari seeks to increase crop diversity, increase productivity so as to make efficient use of land, water and fertilizer, and ultimately make socio-economic impact by ensuring food security and enabling small farmers.

Our project sits within the first stage of Inari's work and seeks to answer the questions "what are the effects on the rest of the maize genes when we perturb a subset of the maize genes?" This allows Inari to edit maize seeds with greater confidence and efficiency. Essentially, we hope to construct a genetic network that can act as a look-up table to inform Inari of the genetic effects and side-effects of perturbing particular maize genes, as opposed to only observing unexpected effects later in the breeding process. Specially, when we are given an unseen sample of maize expressions, we want to be able to predict the rest of the gene expressions from the expressions of a small subset of the genes.

We begin our task with the logical and straightforward approach of selecting the 1,000 genes that are likely to be the most predictive for others (we refer to these genes as predictor genes or features interchangeably) and then perform regression with these features to predict the rest of the gene expressions. Gradually, we improve and experiment with various components, including feature selection methods, regression methods and finally replacing linear approaches with non-linear prediction approaches.

Throughout our semester of exploration, two existing works that we have found to be especially inspiring for us are the Connectivity Map (CMap) project (Subramanian et al, 2017) and the graph feature autoencoder (GFAE) paper (Hasibi, R. et al, 2020). The first illuminates a sophisticated and highly effective feature selective methodology, while the second leads us on to the promising path of making use of the gene network information.

1.1 Key Insights

To tackle the problem of how to predict unobserved gene expression values, our work and the understandings we gained over the course of the semester could be summarized as follows.

1.1.1 Standard Feature Selection and Regression Methods: Lacking in performance and potential for improvement

Our baseline model comprises two stages, feature selection of the 1,000 predictor genes and predicting unobserved gene expression values using regression methods. Essentially, we divide the genes into 1,000 predictor genes (X) and the predicted genes (y), train and test to see how well the predictor genes can function as "features" that inform the target genes' values.

To select the predictor genes, we used 1. the straightforward criteria of selecting the most expressed 1000 genes, 2. GENIE3 which uses random forests to filter out the more "significant" genes, 3. Landmark 1,000 genes methodology inspired by CMap.

After trying using the most expressed 1,000 genes as the predictor and using the 1,000 most significant genes selected through GENIE3 as the predictor, we obtained mediocre prediction

accuracy. It occurred to us that we need to search for a better feature selection methodology. As for the ensuing regression task, we also sensed the limitations in the predictive power of linear methods. When using linear methods to predict gene expressions, we are essentially repeating the 1,000-to-1 (1,000 predictor genes to one predicted gene) regression tasks. This is inefficient, as well as neglectful of the potential network information among the genes. This paves the way for our shift to graph-based methods later on.

1.1.2 Landmark 1,000 Genes as Feature Selection: Unsuited for large predicting task and our non-perturbation-driven data

We found a close counterpart of our problem for human genetics in the Connectivity Map (CMap) project (Subramanian et al, 2017). With the selected landmark 1,000 genes as predictors, the CMap project achieved 81% accuracy when predicting the rest of the human genome. Discovering the Landmark 1,000 genes methodology was a helpful breakthrough as it offered a more sophisticated feature selection method, together with its proven effectiveness in conducting the same task in the human genome.

While exploring the L1000 method, we did principal components analysis and cluster analysis that shed more light on our dataset.

However, we eventually found L1000 to be unsuitable for our specific task. Initially, the difficulty rests in that, compared to the CMap project, our task involves more genes, fewer samples and a heavier prediction task, which made the computation more expensive and the prediction less effective. Our dataset of 480 tissue samples across 25 individuals is significantly smaller than the 12,031 genetic profiles CMap analyzed. CMap performed inference on 11,350 human genes, while we were initially tasked with predicting the rest of the 46,430 maize genes.

The key obstacle, however, is that our data is not perturbation-driven like in the CMap project. This meant that we only have the subtle natural variations between the 25 individuals. Therefore, in the cluster analysis which is the central process in L1000, we are made to select a representative gene for each cluster and discard the rest of the genes in the cluster. Since our expression data lacks in variation, discarding each large cluster loses significant amount of information, leaving landmark genes less effective in their predictive power.

1.1.3 Methods of Imputation: Increased performance over feature selection and regression-based methods, while still leaving room for further improvement

We implemented two imputation models, namely a k-nearest neighbors model and a variational autoencoder model. Both models exhibited similar performance and had significantly better performance relative to a naive mean imputation model. These models. also outperformed regression-based methods for prediction.

However, these imputation methods, especially the k-NN imputation method, are not sensitive to single gene perturbations. This insensitivity makes it impossible to properly model the effects of gene perturbations.

1.1.4 Graph Autoencoders: Simultaneous training on the expression data and a network representation

As our final attempt, we found the paper on graph autoencoders by Hasibi, R. et al to be very pertinent. The team makes use of graph autoencoders for a task that is essentially the same as ours, specifically the prediction of unobserved gene expression values. At the same time, our previous work and lack of success also points us towards the direction of graph autoencoders.

Graph neural networks have become prevalent tools in the context of biological and biomedical research given the natural tendency of representing different biological phenomena as networks (e.g. gene regulatory networks, protein-protein interaction networks, cell-signaling networks). This approach uses a graph autoencoder, an autoencoder designed for network-style inputs, in order to learn features of a network representation of the expression data along with the expression data itself. The latent embedding learned by the autoencoder is used to predict unknown node (gene) features (expression values). We use a maize protein-protein interaction network as opposed to a gene network derived from our expression data, allowing us to incorporate information past the transcriptional phase into the model and allowing us to use causal links between nodes. The graph autoencoder model is expensive to train, but has potentially better predictive capabilities as compared to the previous methods.

2 DATA

2.1 Data Description

The data is comprised of gene expression values derived from 26 individual maize plants. Multiple samples were taken from 10 different tissues from these 26 individuals and subsequently sequenced in order to obtain gene expression counts. The number of samples contributed varies per individual and varies per tissue. As well, samples were collected at different developmental stages. The samples collected from 1 of the individuals are set aside for validation (referred to as the test set), leaving the samples collected from the remaining 25 individuals as the data to be used for model creation and model improvement (referred to as the training set). These datasets are provided in tabular form. Each entry represents the quantification of the expression level for the given gene in the given sample. Therefore, the set of genes analyzed is consistent across all samples. The training set contains 480 samples, each with expression levels for 46,430 genes. The test set contains 21 samples, each with expression levels for the same 46430 genes found in the training set.

The raw gene expression counts are first converted to transcripts per million (TPM). The steps of this preprocessing technique are as follows: 1) Divide the expression level by the length of the given gene in kilobases, 2) Divide by the summation of gene length normalized expression levels in the given sample, 3) Multiply by 1,000,000. The process of converting raw counts to TPM first normalized the expression levels by the gene length. More fragments are likely to map to longer genes and vice versa for shorter genes. These gene length normalized expression levels are then adjusted for sequencing depth. Sequencing depth refers to the total counts attributed to a given sample. Given the inability to sequence all samples at the same time and with the same machinery, as well as the possibility that different protocols were used for different samples, sequencing depth normalization is an important step in being able to make comparisons across samples. Lastly, the expression levels are scaled by a large factor.

2.2 Data Preprocessing

TPM expression data is normalized by gene length and by sequencing depth, making it amenable for many downstream gene expression analyses. However, further preprocessing is often done before performing these analyses. Gene expression data is typically skewed with a relatively low number of very high values. As well, it is more biologically meaningful to look at proportional discrepancies as opposed to additive discrepancies. The expression data was $\log_2$ transformed with an offset in order to handle 0 values. The log-transformed data log-TPM is derived as follows for every x_{ij} entry in the $n \times p$ expression matrix X.

$$\text{transformed } x_{ij} = \log_2(x_{ij} + 1), i = 1, \ldots, n, j = 1, \ldots, p$$

Lowly expressed genes often do not provide a substantial amount of information and do not elucidate relationships between groups or genes. For example, removing lowly expressed genes when performing differential expression analysis between groups of samples will increase the power to detect significant differences given that the correction for multiple comparisons will be less stringent. Genes were deemed as lowly expressed if they met the following criteria: 1) No expression in at least 80% of samples, 2) The maximum expression across all samples was less than or equal to 2 TPM. 7733 genes were considered as lowly expressed and subsequently removed, leaving 38697 genes.

3 EXPLORATORY DATA ANALYSIS

3.1 Gene Expression Distributions

After data preprocessing, the data was first analyzed by looking at the mean expression levels across all genes. These distributions can be found in Figure 1. Mean expression levels from all individuals follow a similar distribution.

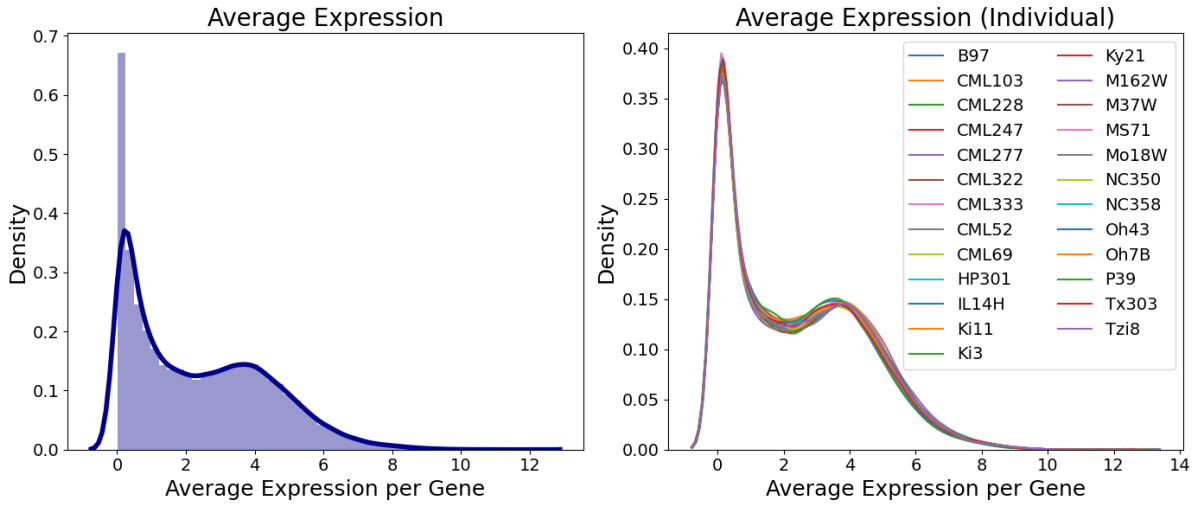

Figure 1: Distributions of the mean expression level per gene. The plot on the left shows the distribution when taking the mean expression level across all samples. The plot on the right shows 25 distributions derived from taking the mean expression level across samples from a particular individual.

Picking the top g most expressed genes to serve as a feature set for predictive modeling may

seem to be an advisable initial selection criterion. However, highly variable genes adjusted for the mean are likely to serve as better predictors. Figure 2 shows the relationship between the coefficient of variation (CV) and mean, as well as the distribution of CV values.

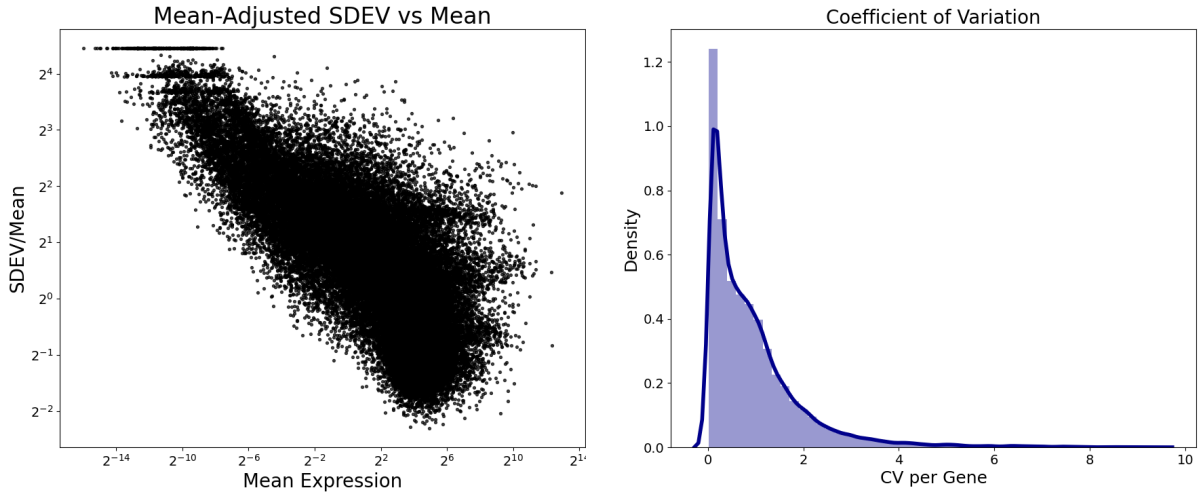

Figure 2: Left) Coefficient of variation vs mean for the expression level across all genes. Right) The distribution of the coefficient of variation values. The expression data used are unprocessed TPM values.

3.2 Dimensionality Reduction Visualizations

Dimensionality reduction techniques were used in order to quickly assess whether or not the data clustered based on certain sample metadata. Principal component analysis (PCA) was performed on the preprocessed expression data. As well, t-Distributed Stochastic Neighbor Embedding (tSNE) was performed on the preprocessed expression data. TSNE is a nonlinear dimensionality technique that is often used when analyzing single-cell data, but can be applied to any high-dimensional data. The main objective is to preserve the local structure of the data, while tSNE's main pitfall is its inability to preserve global structure and therefore its inability to draw conclusions from between-cluster distances. Explanations of these dimensionality reduction techniques can be found in the Appendix. Figures 3 and 4 show PCA and tSNE plots colored by individual and by tissue (organism part). Figure 4 indicates that there is distinct clustering by tissue. For this reason, it may be advisable to perform separate analyses and create separate predictive models per tissue.

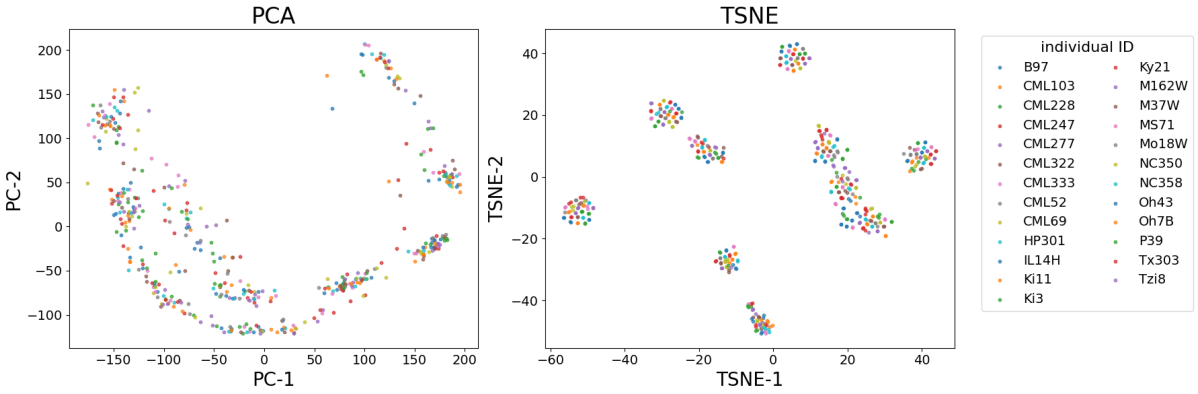

Figure 3: PCA and tSNE dimensionality reduction were used on preprocessed TPM expression values. The datapoints are colored by individual ID. There is no obvious clustering based on individual ID.

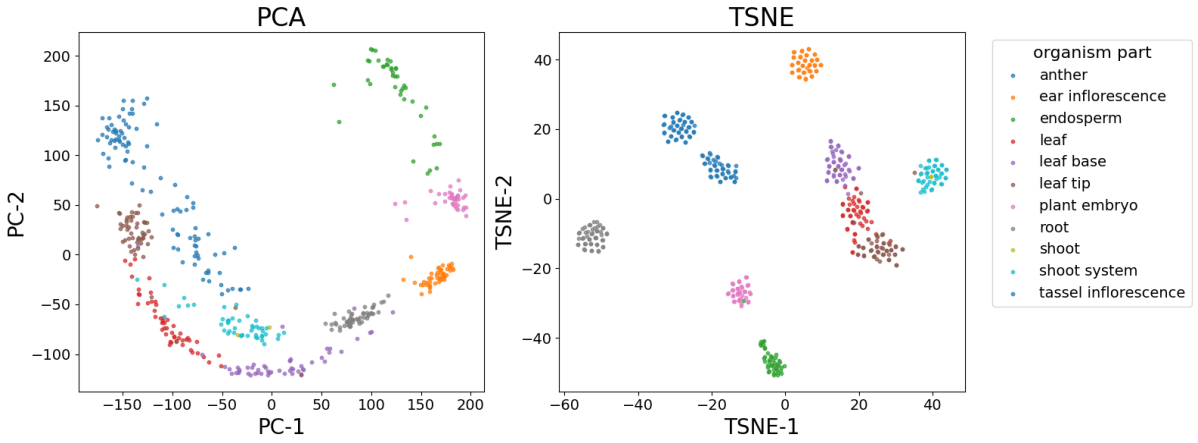

Figure 4: PCA and tSNE dimensionality reduction were used on preprocessed TPM expression values. The datapoints are colored by tissue. Distinct clustering based on tissue is present.

4 APPROACH 1: STANDARD FEATURE SELECTION & REGRESSION METHODS

In Milestone 2, we experimented with multiple combinations of feature selection and regression-based methods to predict unobserved gene expression values.

Feature Selection Methods	Prediction Methods
Most Expressed 1,000 Genes^	Various Regression Methods
GENIE3 Pairwise Correlation*	
Landmark 1000 Genes^	

Table 1: Summary of feature selection and prediction methods
*Unique set for each target gene
^Common set for all the genes

4.1 Selecting Most Expressed 1,000 Genes as Predictors

As a first pass for our baseline model, we chose a representative subset of the genes by their Coefficient of Variation, or simply the standard deviation divided by the mean. This is a metric that aims to capture the genes with the most variability in their expression profiles, while adjusting for their average expression.

The idea behind this approach is to avoid genes that are expressed at constant or near constant levels (and hence would not help in any standard regression model), while at the same time ordering by genes that have relevantly high levels of expression such that their variability is not entirely anomalous or due to noise.

4.1.1 Results

As a first step, we built Ordinary Least Square (OLS) models using the 1000 most expressed genes to predict the expression profiles of each one of the next 1000 most expressed genes. A separate OLS model was trained for each target gene. The ranking metric used was the average expression level. After eliminating a total of 2,432 genes which were not expressed in any of the samples, we sorted the genes in two ways: 1) Using the average expression level per gene across all samples and 2) Using the coefficient of variation per gene across all samples. This allowed us to account for different scales across genes, ranking by variability in a more representative way.

In total we ran 1000 models per method using a train-test split of 70-30 (i.e., 70% for the training set and 30% for the validation set). The models were fitted using the training set and R^2 of the validation sets were obtained for further analysis of accuracy.

The summary statistics for the R^2 values derived from sorting using the average expression level and sorting using the coefficient of variation of the expression level are provided in Table 2. It is evident that models derived from the mean sorting method performed substantially better than the models derived from CV sorting.

—	Mean sorting R^2	CV sorting R^2
mean	0.747	-1.8995e+16
SD	0.285	5.2427e+17

Table 2: Summary Statistics for R^2. Mean sorting: The average expression levels were used for sorting, determining the predictors, and determining the response variables. CV sorting: The coefficients of variation for the expression levels were used for sorting, determining the predictors, and determining the response variables.

In general, almost all R^2 were close to 0, allowing us to conclude that the models were no better at predicting gene expression than naively predicting the average expression levels for the response genes.

Given the the use of the same 1000 gene predictor set led to disappointing results, an alternative approach was used based on variable feature selection per gene. GENIE3 calculates pairwise correlations of genes by inferring gene regulatory networks from expression data. For each target gene that we wanted to predict, we selected as features the genes that had $|R| \geq 0.01$ with the given target gene. Given our memory limitations, we only ran GENIE3 on

4000 genes, which is about 10% of the total number of genes. We selected these 4000 by removing genes that are not expressed in 95% of the samples, or whose highest expression value is less than 300 TPM.

Using a 70-30 percent train-validation split, we ran a few baseline and ensemble models using both 1000 most expressive genes (sorted by Coefficient of Variation) to predict the next 1000 most expressive genes and with GENIE3 to select the features for the 4000 genes chosen for this analysis. The models that were ran were: a) Linear Regression, b) Lasso Regression, c) Lars Regression, d) Random Forest Regression, e) Ada Boosting Regression, f) Gradient Boosting (check the appendix for an explanation of the ensemble methods). Results of validation R^2 are shown in Table 3.

4.2 Selecting Most Significantly Correlated Genes as Predictors

GENIE3 is a tool aimed at recovering a gene regulatory network from expression data using tree-based ensemble methods. A random forest regression is run on each target to achieve the effect of feature selection (i.e. the genes that are the best predictors of the target gene). This process then outputs a list of pairwise correlations between genes. Then, for each target gene, we select the 1,000 genes that are most correlated with the target gene. The correlation values may not be statistically significant, but are indicators of the relative strengths of the pairwise relationships.

4.2.1 Results

As the reader can see in Table 2, if we use the same features (i.e., the 1000 most expressive genes by Coefficient of Variation) for all predictions, we get disappointing results of the validation R^2 for all models. On the other hand, If we use GENIE3 to select the features per gene, we get significantly better results. As expected, the ensemble models resulted to be the better models. Random Forest achieved the highest mean R^2 with 0.6483. Even though using the 1000 most expressive genes by Coefficient of Variation allow us to use the same features for all predictions, results were significantly worse than in GENIE3. Nevertheless, a major drawback of the GENIE3 method is that we need the entire database and significant computation power to calculate the pairwise correlation. For each gene that we have to predict, we have to calculate the pairwise correlation with all of the other genes to select the features that we're going to use for the respective regression.

Given these results, we decided to pursue several more feasible feature selection methods and models that will likely provide better performance and which are going to be discussed in the next sections of this report.

—	Mean of R^2 GENIE3	Mean of R^2 1000 Most Expressive Genes
Linear Regression	0.4895	-1.899e+16
Lasso Regression	0.5061	-27.0659
Lars Regression	0.4889	-7.8803
Random Forest Regression	0.6483	-1.6092
Ada Boosting Regression	0.5755	-2.3085
Gradient Boosting	0.6285	-4.0844

Table 3: Average R^2 values for regression models using both feature selection methods

5 APPROACH 2: LANDMARK 1000 GENES

5.1 L1000 for Feature Selection

In this section, we attempt to recreate the L1000 landmark gene (Subramanian et al., 2017) approach that uses aggressive clustering techniques to select a representative "landmark" set of genes. The key idea is the assumption that the landmark genes are of central importance to the genome and might be good predictors of all the rest of the genes. The steps are:

1. Dimensionality reduction of the expression data using principal components analysis (PCA)
2. Cluster analysis to cluster the genes
3. "Bootstrapping" to obtain subsamples so as to obtain 100 different clusterings
4. Iterative peel-off to remove genes far from the centroids of the clusters
5. Consensus matrix to obtain the percentage of trials where each pair of genes are in the same cluster
6. Setting a threshold of the percentage, select the pairs of genes that are always in stable clusters and further group these pairs into clusters
7. By placing the processed clusters back in the reduced dimensions of the PCA and identify a single centroid, defined as the gene that has the least euclidean distances to every other gene in the processed cluster
8. The centroids of the processed clusters are pronounced as landmark genes
9. Repeat the process until 1,000 landmark genes are identified

In particular, this approach emphasizes the need to find a highly informative subset of the gene expression data, that ideally captures the variation allowing us to predict the expression of the held-out majority group. The main benefits of this approach lie in the fact that of the existing literature, this is one of the few regression-oriented approaches that have been attempted with purely gene expression data, and that they seemed to have decent accuracy, measured by the ability of the landmark 1000 to recover 81-83% (paper cites varying numbers) of the missing gene expressions.

The main challenges are that their experiment was conducted with many orders of magnitude more effective data, in both sheer quantity (10,000+ individual samples) and data variation (164 introduced perturbations in cells), as well as inference targets (10,000 genes studied instead of 40,000). In this section, we attempt to replicate their methods and determine the efficacy of the landmark gene approach at our data scale.

5.2 L1000 Detailed Methodology

The provided code in Subramanian et al.'s (2017) paper and respective GitHub repositories were extremely difficult to navigate and reproduce, spanning multiple platforms (Matlab, Python, Java, R) where critical analytical steps had unique implementations *across* different platforms, meaning that there was no complete Python pipeline (requiring analysis to be partly done on

the other platforms). On top of this, the code when studied in detail revealed that they made huge assumptions about how the data was structured, tailored specifically to their dataset. Finally, the actual step of selecting the L1000 landmark genes and the processing required to do so was elided from their pipelines, where downstream processing referred to a hardcoded "L1000" data file instead of upstream pipeline components to do this processing.

In light of these challenges, we reproduced the critical elements of their pipeline and broke down the necessary steps into reproducible components using simple, existing library implementations such as clustering tools from Scikit Learn. The basic selection algorithm is as follows:

1. **Dimensionality reduction:**
   In order to perform Euclidean-based clustering, the original data dimensions are transformed into a lower dimensional representation. Here, like in the referred paper, we chose Principal Component Analysis and decomposed each gene into the top 50 representative principal components. We visualize the PCA decomposition using a further tSNE dimensionality reduction in Figure 5.

Figure 5: PCA decomposition of all corn genes, visualized using further tSNE dimensionality reduction. Some small clusters of similar genes, hinting at gene regulatory subnetworks, can be seen.

2. **K-Means Clustering:**
   We then cluster on the first 50 principal components, using a simple K-Means clustering. In order to speed up computation over the width of the dataset, we use Mini Batch K-Means clustering. Figure 6 is a visualization of one sample of K-Means clustering, overlaid over the tSNE representation of the PCA components.

3. **Consensus Matrix Construction**
   The next step involves repeating the K-Means sampling on "bootstrapped" samples, each representing a randomly sampled 75% of the total gene count. As done in the paper, this repeated K-Means sampling will be repeated 100 times. Across all 100 clustering runs,

Figure 6: K-Means clustering applied to the first 50 principal components, again visualized by tSNE. We chose $k = 100$ for each iteration of the iterative peel-off process described below. While cluster overlap between tSNE and and K-Means projections seem fairly weak visually, many of the larger clusters (note purple cluster at bottom) were particularly stable across multiple K-Means iterations.

a consensus matrix is then constructed, where the pairwise ratio of the number of times two genes end up clustered together across the 100 runs is represented.

The consensus matrix is then thresholded for clusters above a certain ratio, arbitrarily .8 for the original approach. Clusters that remain in the consensus matrix with genes that are highly clustered together are then the targets for further processing.

One of the primary challenges this step revealed was the lack of variation in our source data set. The greatest source of variability in our gene expression levels comes from the 10 different tissue samples. However, there are no further perturbations introduced per sample, which in comparison to the 164 experimentally induced perturbations in the original paper causes a number of issues like large cluster size.

In Figure 7, we see the average of cluster sizes across all 100 K-Means cluster attempts. The variability was low, and each individual cluster attempt demonstrated the visualized behavior of very large clusters at the top-end with an average of 4000 members. Although this is the largest such cluster and the next clusters range in the hundreds, the ideal cluster size for a data set of 40,000 genes and 1000 landmark genes is at least an order of magnitude smaller (roughly 40000/1000 = 40).

4. **Iterative Peel-Off**
   In the next step of the pipeline, the top clusters identified are utilized by selecting the single gene closest to their center. The center is defined as the centroid of the cluster in PCA space, with a sample visualization in Figure 8.

   The iterative peel-off methodology then removes all other cluster genes from consideration, and only keeps the gene closest to the centroid as a landmark gene. Once a few landmark genes are identified, the entire process of 100 K-Means clustering attempts is repeated, and another consensus matrix is created. The hope is that without the genes of

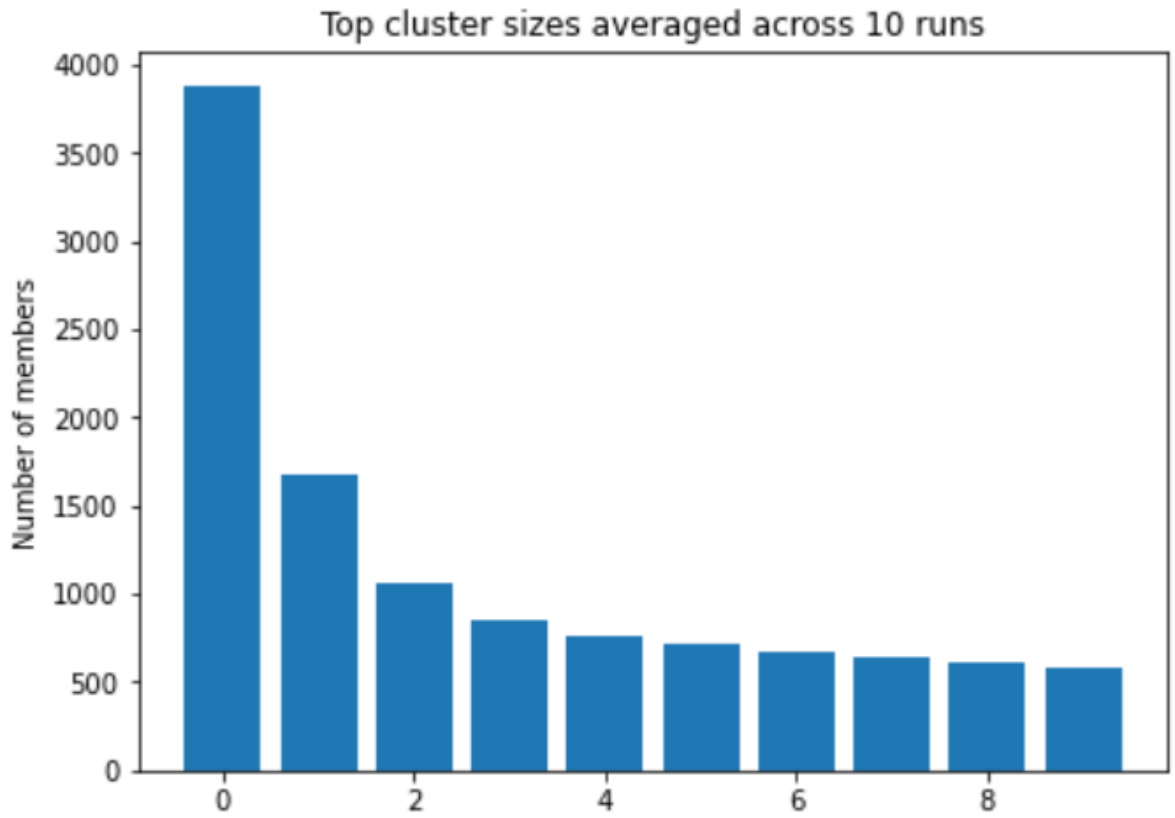

Figure 7: A chart of the largest cluster sizes, averaged across 100 iterations of K-Means with k=100. The largest clusters are disproportionately large, on average around 4000 gene members, easily eclipsing the target of 1000 genes as a representative sample for the entire set. These clustering averages remain relatively consistent even with k=1000, as the granularity is developed at the tail end.

the previously identified clusters for landmark selection, a new highly related cluster will surface as candidates for the next few landmark genes.

In practice, this approach on our problem ran into a number of significant challenges. The first of which is already explored above, and primarily stems from the lack of data variability to form tight enough clusters such that we do not remove all of our data before we get a chance to form a large enough representative set. This is ameliorated by the random iterative sampling, and in truth with tight thresholds (pairs of genes show up in >0.8 of clustering attempts) we can see small clusters that might suggest appropriate landmark genes.

The main issue however, is one purely of computation. We have an exponential data scaling issue, even more so than the original paper's 10,000 human gene set under analysis. Consider a single consensus matrix shaped 40000×40000: if we use standard floating point values of 8 bytes, we are looking at $40000 \times 40000 \times 8 = 25600000000$, or 25.6 GB of data for a single instantiation. This can be reduced with lower precision representations depending on what we need, but the sheer computation of multiple arrays requires machines with larger than conventional memory sets. Note that at 10,000 genes, these sizes are manageable on standard 16GB machines. We are currently considering methods of scaling our computation and working on cloud computing with larger memory resources.

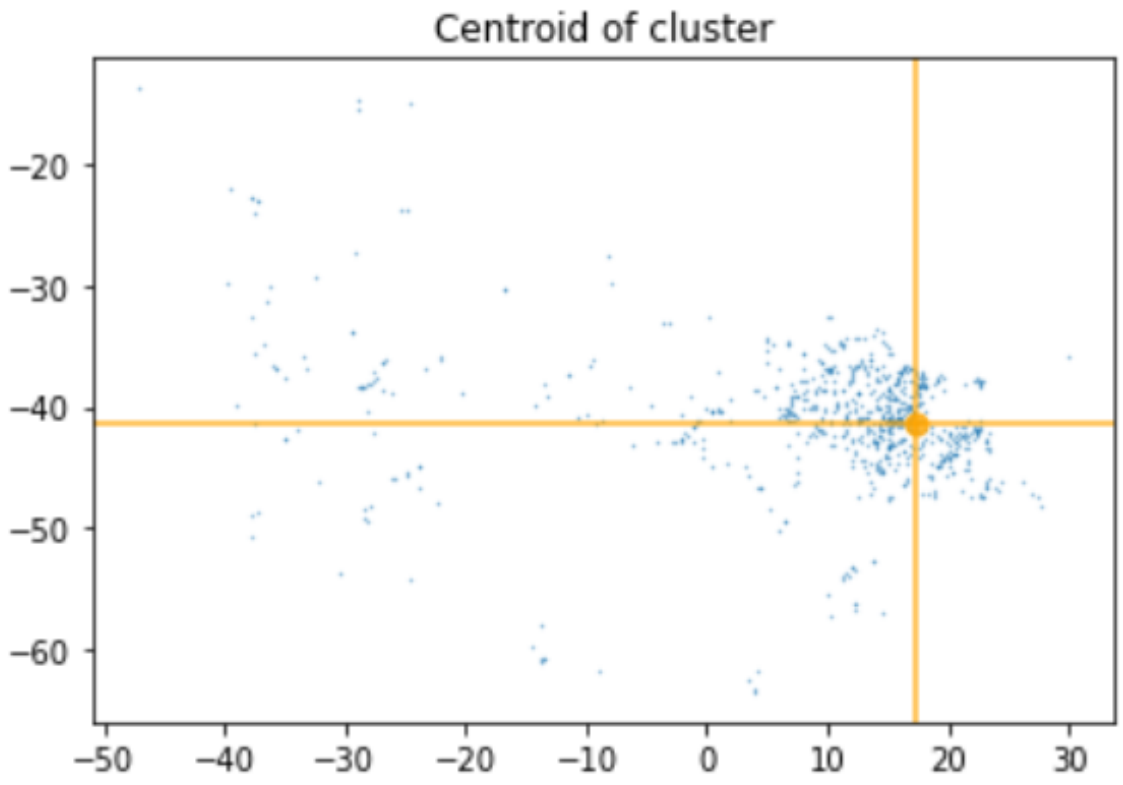

Figure 8: A sample cluster centroid, of which we then choose the single landmark gene closest in Euclidean distance to this centroid. As part of the iterative peel-off process, all other genes in this cluster will be removed except for the chosen landmark gene.

6 APPROACH 3: IMPUTATION OF MISSING VALUES

Missing value imputation is widely used when analyzing gene expression data due to a phenomenon known as dropout. Either due to the detection threshold or inherent technical mishaps of the sequencer, the expression levels for certain genes for certain samples will not be registered. Instead of removing observations that contain missing values, often times missing value imputation is used in order to reconstruct these expression levels from the available information (i.e. the non-missing entries). Given the abundant documentation on the performance of imputation methods in the context of gene expression data, several imputation algorithms were re-purposed for predictive modeling.

Mean, k-nearest neighbors (kNN), and variational autoencoder (VAE) imputation models were built and compared. Each model was trained on the training set consisting of samples from 25 individual maize plants (n = 480). The training set did not contain any missing values. Trained models were used to impute missing values in the test set with samples from 1 individual maize plant (n = 21). The imputation models are described as follows.

6.1 Naive Mean Imputation

The naive mean imputation model predicts the mean expression level for a given gene. First, the mean expression level for every gene in the full training set is calculated. The mean expres-

sion levels are used to impute missing values in the test set for those genes that contain missing values. A gene's missing value(s) are imputed with the mean expression level of the same gene derived from the training set. While the unprocessed expression data is right-skewed, the log-transformation performed during data preprocessing removes, to an extent, the skewness of the data. For this reason, we use a naive mean imputation model instead of a naive median imputation that predicts the median expression level for a given gene.

6.2 k-Nearest Neighbors Imputation

The kNN imputation model constructs a kNN graph by deriving the k nearest neighbors for every gene. The nearest neighbors are defined as those genes among the k^{th} closest genes based on the euclidean distance. The distance measure $d(p, q)$ between genes p and q is defined as follows.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (x_{ip} - x_{iq})^2}$$

A missing value x_{ip} found in the test set is imputed with the average expression level of the k nearest genes at the given observation.

$$x_{ip} = \frac{1}{k} \sum_{j=1}^{k} x_{ij}$$

6.3 Variational Autoencoder Imputation

Autoencoders are a family of neural networks with a 2-part architecture comprised of an encoder and a decoder. The encoder takes an input X and transforms the input into a latent representation Z. The decoder takes as input Z and outputs a reconstructed version of the input $\hat{X}$. The neural network aims to minimize the reconstruction error or reconstruction loss $L(X, \hat{X})$ that quantifies the discrepancy between the input and the output. In most cases, the Z is constrained to a lower-dimensional space as compared to X. Variational autoencoders, instead of using an encoder to map observations of the input to points in a latent space, use an encoder to map observations of the input to their respective latent distributions. In practice, the latent distribution corresponding to each observation is normal. The decoder samples from the latent distributions before reconstructing the input. The sampling step effectively regularizes the autoencoder, making the model less likely to overfit to training data. Variational autoencoders have been shown to perform well for imputing missing values in gene expression data (Qui, Y.L. et al 2020).

6.4 Results

The 3 imputation models were tested under two distinct scenarios. In scenario 1, a specified proportion of the entries in the expression matrix are selected and subsequently masked. In scenario 2, a specified subset of genes are selected and all entries corresponding to the selected subset of genes are subsequently masked. Scenario 2 is analogous to the case where a subset of genes are used as predictors in order to determine the expression levels for the response genes for which no data exists. Representations for scenario 1 and scenario 2 are given in Figure 9. Each model was tested under varying levels of missing data severity. For

both scenarios, 5%, 10%, and 50% of the entries in the test set were masked, these entries posing as missing values. For every scenario-missing data severity pair, missing values were randomly generated 5 times. The average R^2 values are reported in Table 4. The computation for the R^2 values is based on the true values and imputed values for the missing data and does not take into account all entries in the test set as this would lead to an artificial inflation of the given model's performance.

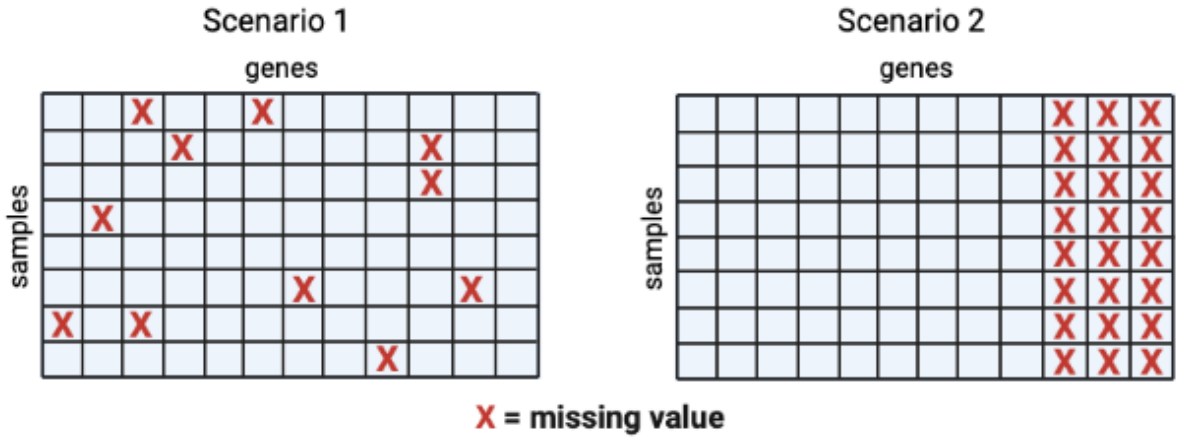

Figure 9: Scenario 1: A proportion of entries are randomly selected and masked. Scenario 2: A subset of genes are randomly selected and all entries corresponding to the selected genes are masked.

—	Mean Imputation R^2	kNN Imputation R^2	VAE Imputation R^2
scenario 1: 5% missing value rate	0.7231	0.9073	0.8985
scenario 1: 10% missing value rate	0.7235	0.9075	0.9052
scenario 1: 50% missing value rate	0.7223	0.8801	0.8936
scenario 2: 5% missing value rate	0.7135	0.9070	0.9034
scenario 2: 10% missing value rate	0.7206	0.9068	0.8981
scenario 2: 50% missing value rate	0.7217	0.8923	0.8941

Table 4: The average R^2 values for several imputation methods applied to different scenarios of missing values.

7 APPROACH 4: GRAPH AUTOENCODERS

As biological network representations are increasingly used in the context of genomics, proteomics, and other -omics type fields, methods have been developed based on graph neural networks (GNN) in order to draw insights from these network representations. Graph autoencoders (GAE) are autoencoders that work on network inputs. These models are trained to reconstruct a network representation of the given data (e.g. adjacency matrix, similarity matrix). It has been shown that graph autoencoders can be used to simultaneously learn features of a network representation of expression data along with features of the expression data itself (Hasibi R. et al 2020). These models can be trained in an end-to-end fashion, where missing feature values (i.e. expression values) for a subset of nodes (i.e. genes) in the given network can be derived from the feature values of surrounding nodes. By running different experimental setups, Ramin et al showed that the prediction results are the best when using both the expression data and a graph network of the genes, as opposed to using only of them.

The GAE used is adapted from Hasibi R. et al 2020. The network representation in the format of an adjacency matrix A and the matrix of node features X serve as inputs to the model. The model consists of 2 graph convolutional layers that produce an embedding matrix Z. The layers are formed by the activation functions ReLU and Sigmoid.

$$Z = GCN(X, A)$$

$$GCN(X, A) = AReLU(A, X, W_0)$$

The reconstructed adjacency matrix is defined as follows.

$$\hat{A} = \sigma(ZZ^T)$$

The model is trained to minimize the cross-entropy error between the adjacency matrix A and the reconstructed adjacency matrix $\hat{A}$. The loss function is given as follows.

$$\mathcal{L} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} \ln \hat{a}_{ij}, \text{ where } n = \text{ the number of nodes}$$

A message-passing neural network (MPNN) is used to predict node features from the latent embedding. Ramin et al also developed their own message passing network in order to predict node feature values (i.e. gene expression values) based on network neighborhood information.

Instead of deriving a gene network from the expression data, we use a maize protein-protein interaction (PPI) network taken from http://comp-sysbio.org/ppim/ (Zhu G. et al 2016). Using a PPI network in conjunction with the expression data allows us to incorporate information unseen at the transcriptional level and to establish causal links between nodes in the network. The feature matrix is filtered to strictly include genes also found in the given PPI and vice-versa. A total of 11399 genes were left after genes not found in the PPI were removed.

There are two different approaches of predicting the expression value: a) Imputation, a single model is used to predict all values of the feature matrix. The train-test values were chosen randomly within the data b) Prediction: each dimension of Y is predicted using a separate model which was trained on X. These approaches are analogous to the scenarios described in Figure 9. Given expression data from a set of training samples X_{train} and expression data from a set of test samples X_{test} we would like to learn the relationships between node features. Specifically, we would like to learn the relationships between a subset of n nodes and a subset of p nodes in X_{train} so that the same subset of n nodes in X_{test} can be used to predict the feature values for the remaining p nodes in X_{test}. However, for the purpose of this milestone, we followed the imputation approach, and are currently performing and hyperparameter-tuning the prediction approach for the final delivery.

7.1 Graph Autoencoder Imputation Approach

First, the edges of each graph were split into three folds, two used for training and the third for testing. A regression model is used on the embedding of the training data for each one of the features and is used to measure the Variance Explained on the feature for the test nodes. This procedure was done using a 3-fold cross validation scheme: it was done using each one of the fold as test set and the other two as train set, then getting the average of the test errors.

7.2 Results

8 APPENDIX (EXPLANATION OF METHODS)

8.1 Reconstructing Gene Regulatory Networks

One of the central motivations behind our problem is the attempt to understand the full relationship graph of all expressed genes, or more specifically, reconstructing some kind of Gene Regulatory Network. GRN's in the literature have been well studied for more than a decade, but the area remains largely an "unsolved problem" (Saint-Antoine et al., 2019) and is complicated by a bevy of computational challenges.

More formally, work on GRNs fully encompasses our current problem setting as follows: we consider N genes and represent their expression levels (the entirety of the data we have) as random variables $\{X_1, X_2, ..., X_N\}$. Each X_i thus represents a node in our network, and any relation $r(X_i, X_j)$ describes an edge where r models the regulatory relationship between the two genes. These relations can be modeled as directed, implying causality, or with signs, hence representing downregulation or upregulation.

This formal description of the problem leads to the natural formation of any model that wishes to describe a GRN as some subset of all possible relations $R = \{r \mid r : (X_i, X_j) \mapsto s\}$, where s is typically some score representing the importance of that relation. Indeed, the vast majority of methods in reconstructing GRNs use precisely such a metric, and one of the most popular competitions Dream Challenges ([homepage - Dream Challenges](http://dreamchallenges.org/)) provides gold-standard evaluation data where the target labels are exactly these edge importance values, as verified experimentally or through other modalities.

One key discrepancy between our current problem formulation (i.e. regression on subset of most highly expressed to find expression levels of the rest) and the existing literature is that our evaluation metric is directly based on expression levels, while all other existing methods try to evaluate based on real biological understanding of gene regulatory pathways. There are a number of clear downsides to metrics based purely on expression levels:

- Consider genes with low variability (almost always expressed at a constant level, or almost always 0, etc.). All aggregate metrics of accuracy over any test set Y would be arbitrarily inflated in accuracy by inclusion of any low-variability genes as all regression methods would simply set feature coefficients to 0 and be a constant predictor.
- Even with a reasonably accurate regressor, it is not trivial to then extrapolate that information and establish which genes are related, what sub-networks exist, what kind of motifs of regulatory action are revealed, etc. Physiologically speaking, we have no experimental variation in our data, and cannot expect to capture consistent patterns tested by perturbation, and rely solely on random individual variation.
- As follows, it will also be hard to generalize the utility of results based on cross-validation of these metrics, given the two-fold difficulty of lacking experimental variation as well as any basis of interpretability.

8.1.1 Existing Approaches

There are a number of classes of approaches in reconstructing Gene Regulatory Networks, each of which begin with the same kind of data that we have (purely gene expression levels, without other modalities of data). Each of these are potentially interesting avenues to pursue, and are worth further experimentation and consideration in our attempt to derive genuinely interesting insight from our data. Figure 5 from Saint-Antoine et al. (2019) describes each class with their respective qualities and advantages/disadvantages.

Algorithm Class	Temporal Data Required?	Directionality	Advantages	Disadvantages	Examples
Correlation	No	Undirected	• Fast, scalable • Detection of feed-forward loops, fan-ins, and fan-outs	• Possibly over-simplistic • False positives for cascades	WGCNA [13] PGCNA [14]
Regression	No	Directed	• Good overall accuracy	• Bad detection of feed-forward loops, fan-ins, and fan-outs	TIGRESS [15], GENIE3 [16], bLARS [17]
Bayesian - Simple	No	Directed	• Performance on small networks	• Performance on large networks. • Inability to detect cycles	[19,20]
Bayesian - Dynamic	Yes	Directed	• Performance on small networks • Detection of cycles and self-edges	• Performance on large networks.	[21]
Information Theory	No	Undirected (at least in simplest form)	• Detection of feed-forward loops, fan-ins, and fan-outs • Similar to correlation methods, with better accuracy	• False positives for cascades	ARACNE [25], CLR [26], MRNET [27], PIDC [28]
Phixer	No	Directed	• Parsimonious output due to pruning step.	• Possible loss of overal accuracy due to pruning step (this can be removed if the user chooses)	[31]

Figure 10: The classes of models used throughout literature in attempting to reconstruct Gene Regulatory Networks.

- **Correlation:** This class of network inference works purely off correlation matrices calculated from the data, and uses varying threshold strategies to try to establish network edges. Baseline models involving WGCNA fall into this category, though their use in predictive settings will devolve equivalently into basic linear regression.
- **Regression:** The setting that most resembles our problem statement. Models in this class like TIGRESS and GENIE3 all work the same way by predicting the expression of a leave-one-out gene from the rest of the expression levels, using some kind of regressor (Least Angle Regression for TIGRESS, and random forest for GENIE3). This is a potential approach for the most direct next step of our modeling.
- **Bayesian:** Either classified simple or dynamic, this class of analysis is computationally expensive and thus well suited for smaller sized networks, but perform quite well in the

literature. In the case of dynamic Bayesian networks that incorporate temporal or pseudo-temporal features, self-regulating genes (modeled by self-edges) can also be discovered.

- **Information theory:** Similar most to correlation approach, these models use mutual information or Shannon information to understand distributional characteristics and test the importance of gene relations under this framework. A popular approach that can deal with multi-step relations, unlike regression approaches.

8.2 Graph Neural Networks

We have done extensive research into many classes of potential Graph Neural Networks to use, and will look into compatibility for the next milestone: StellarGraph, PyTorch Geometric, Deep Graph Library to name a few complete code libraries that exist with good candidate selections. The primary difficulty of usage based on current experimentation is formation of the data and in particular evaluation strategy: prediction on single-attribute nodes is not a very common task in generic graph neural networks currently, and will require an effort of coadapting the models.

To be more specific, graph neural networks aim to solve the following classes of problems:

- **Node classification:** where we infer the class of nodes, e.g. tissue type. We can adapt this to a regression setting, but given that each node's data is essentially one-dimensional (just expression value), this has not been the easiest fit as many networks in this class expect wide nodes with a fair amount of information in each.
- **Link prediction:** inference on missing edges and the relationships of nodes. This is much more prevalent in work in research and pertinent for describing GRNs, but our predictive setting cannot leverage this class of models unless we redefine objectives.
- **Community detection:** this would be an interesting avenue to pursue to do unsupervised clustering of related genes, but does not directly contribute to predictive efforts. Worth keeping on the horizon as potentially a preprocessing step.
- **Graph classification:** classification on the level of an entire graph (i.e. some phenotype given full expression), less relevant as we have no labeled data to this effect.

Combining the power of graph neural networks with the existing approaches in literature in reconstructing GRNs is one of our immediate future directions, and a potential area for novel research and genuine contribution given that we can resolve formulations of evaluation metrics in a way that can adapt to network architectures.

8.3 Regression Methods

Several ensemble methods used for the regressions were Bootstrap Aggregating through Random Forest and Boosting. Bootstrap Aggregating is an ensemble technique in which we build many independent predictors/models/learners and combine them using model averaging techniques (e.g., weighted average, majority vote or normal average). The essence is to select N bootstrap samples, fit a classifier on each of these samples, and train the models in parallel. The results of all classifiers/regression are then averaged into a Bootstrap Aggregating classifier/regression. One example of a Bootstrap Aggregating model is Random Forest, where decision trees are trained in parallel.

On the other hand, Boosting allow us to train models sequentially, instead of modelling them parallel. Each model focuses on where the previous classifier performed poorly.It makes N decision trees during the training of data. As the first decision tree/model is made, the records incorrectly classified are given more priority. Only these records are sent as input for the second model. The process will go on until the number of base learners we specified initially . One important thing to note is that the repetition of records is allowed with all boosting techniques, this guarantees independence across them.

The two Boosting techniques used were: 1) Adaptative Boosting and 2) Gradient Boosting. Adaptive Boosting (AdaBoost) uses multiple weak models to build a strong one. The base model could be any from Decision Trees (often the default) to Logistic Regression, etc.

When decision trees are used, instead of using trees with no fixed depth such as in random forest, AdaBoost only uses nodes with two leaves, Decision Stumps (decision trees with a single split). These are the weak learners in AdaBoost. They work by putting more weight on difficult to classify instances and less on those already handled well. The weights are re-assigned to each instance, with higher weights to incorrectly classified instances.

Gradient boosting relies on the idea to repetitively leverage the patterns of residuals and strengthen a model with weak predictions and make it better. Once we reach a stage that residuals do not have any pattern that could be modeled, we can stop modeling them (otherwise it might lead to overfitting); we use residuals as target variables in the sequential models. We are minimizing our loss function, such that test loss reach its minima.

ReLU (Rectified Linear Activation Unit). The function returns 0 if it receives any negative input, but for any positive value, it returns that value back. Its main advantage is sparsity.

Sigmoid: Is a bounded, differentiable, real function (S-shaped) that is defined for all real input values and has a non-negative derivative at each point and exactly one inflection point.

Qui, Y.L., Zheng, H., Gevaert, O. (2020). Genomic data imputation with variational auto-encoders. GigaScience, 9(8). https://doi.org/10.1093/gigascience/giaa082

Saint-Antoine, M. M., Singh, A. (2020). Network inference in systems biology: recent developments, challenges, and applications. Current Opinion in Biotechnology, 63, 89–98. https://doi.org/10.1016/j.cop

Subramanian, A., Narayan, R., Corsello, S. M., Peck, D. D., Natoli, T. E., Lu, X., Gould, J., Davis, J. F., Tubelli, A. A., Asiedu, J. K., Lahr, D. L., Hirschman, J. E., Liu, Z., Donahue, M., Julian, B., Khan, M., Wadden, D., Smith, I. C., Lam, D., Liberzon, A., ... Golub, T. R. (2017). A Next Generation Connectivity Map: L1000 Platform and the First 1,000,000 Profiles. Cell, 171(6), 1437–1452.e17. https://doi.org/10.1016/j.cell.2017.10.049

Hasibi, R., Michoel, Tom. (2020). A GRAPH FEATURE AUTO-ENCODER FOR THE PREDICTION OF UNOBSERVED NODE FEATURES ON BIOLOGICAL NETWORKS. arXiv

Zhu, G., Wu, A., Xu, X.J., Xiao, P.P., Lu, L., Liu, J., Cao, Y., Chen, L., Wu, J., Zhao, X.M. (2016). PPIM: A Protein-Protein Interaction Database for Maize. Plant Physiol, 170(2), 618–626. https://doi.org/10.1104/pp.15.01821