Application of sparse optimization algorithm in gene regulation network reasoning

Abstract

In this paper, we tested several sparse optimization algorithms based on the public dataset of the DREAM5 Gene Regulatory Network Inference Challenge. And we find that introducing 20% of the regulatory network as a priori known data can provide a basis for parameter selection of inference algorithms, thus improving prediction efficiency and accuracy. In addition to testing common sparse optimization methods, we also developed voting algorithms by bagging them. Experiments on the DREAM5 dataset show that the sparse optimization-based inference of the moderation relation works well, achieving better results than the official DREAM5 results on three datasets. However, the performance of traditional independent algorithms varies greatly in the face of different datasets, while our voting algorithm achieves the best results on three of the four datasets.

To explore the adaptability of different algorithms for different size networks, we conducted experiments based on simulated data and found that the coordinate descent-based Lasso algorithm and OMP algorithm are better for small-scale network reasoning; while the improved algorithms such as SP and CosaMP have higher accuracy for large-scale networks. Meanwhile, we focus on the noise in the gene regulation process, and we find that in a low-noise situation, algorithms such as OMP with good theoretical convergence support have higher prediction accuracy. And for the realistic gene regulation with high noise, using voting algorithm can give a good result. The voting algorithm always gives a better prediction result and has good accuracy variance and stability for repeating the experiment many times.

Finally, we make the code used in the experiments into a python package and open source it on Pypi and Github.

Contents

1 Introduction

For certain diseases or traits, genes play a crucial role in their expression. And the interactions between genes, i.e., gene regulatory networks (GRNs), have become a recent research hotspot. With the availability of high-throughput gene expression data and the substantial increase in arithmetic power, it is possible to reconstruct large-scale gene regulatory networks[1]. In gene regulatory networks, genes can be divided into two categories. Transcription factors (TF), also known as trans-acting factors, are DNA-binding proteins that specifically interact with the cis-acting elements of eukaryotic genes and have an activating or inhibiting effect on gene transcription. The gene that receives this activation or repression is referred to as the target gene. It is important to note that transcription factors themselves may also be target genes, i.e., there may be mutual regulation in the regulatory network.

In a biological system consisting of n transcription factors and m target genes, there are at most $m * n$ groups of regulatory relationships. However, the fact is that the number of groups of regulatory relationships will be much smaller than $m * n$, i.e. such a regulatory network is sparse. Current research is mainly based on differential equations[11][12], Boolean networks[13], mutual information[16], and Bayesian networks[14][15].However, these methods do not make good use of the sparse nature of gene regulatory networks, resulting in unsatisfactory accuracy or excessive time consumption.

We assume that the regulation between genes is linear. Since the total number of genes in this problem is much larger than the given sample size. Therefore, the equation(1)

$$Ax = b + \varepsilon \tag{1}$$

is a ill-posed problem. The mathematical description of the problem is as equation(2)

$$A \in \mathbb{R}^{m \times n}, \quad x \in \mathbb{R}^{n}, \quad b \in \mathbb{R}^{m} \tag{2}$$

Since $rank(A) < n$, the equation $Ax = b + \varepsilon$ has an infinite set of solutions. At this point, if when x is sparse, the equation has a unique solution. We call this class of problems as compression-aware or sparse reconstruction problems.

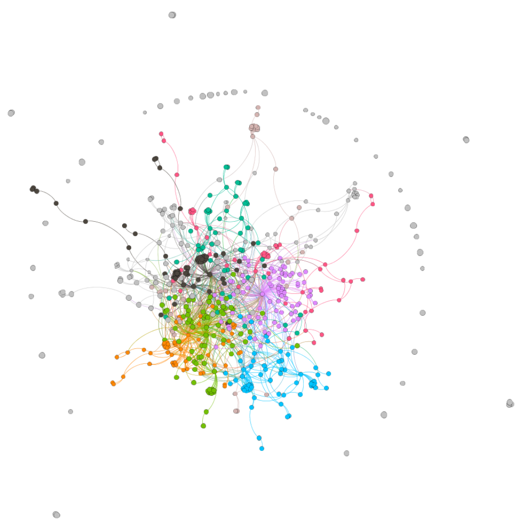


Figure 1: **Gene regulatory relationships:** Playing the role of x in the above set of linear equations, the colored dots are the factors involved in the regulatory relationship and the gray dots represent the factors that are not involved in the regulation. From the figure, we can find that this regulatory relationship is sparse.

In this paper, we will use the DREAM5[3][26] challenge dataset as experimental data. First, we designed the workflow for reasoning about the regulatory network from the expression matrix and tested various sparse optimization algorithms. We then bagged these algorithms to achieve better results for any dataset. Due to the large cost of real data acquisition, we then synthesized simulated data based on linear stochastic synthesis algorithms and varied their sparsity, several genes, sample size, and other factors to explore the relationship between accuracy and them. Finally, we integrated the above algorithms into an easy-to-use package and open-sourced it.

2 Materials & Methods

2.1 Data sets from DREAM5

The data is a publicly available dataset from the Dream Challenge, with four sets of data. The first of these sets is computer-simulated data. The last three data sets are derived from microbial and real pathogen sequencing results, and the specific data sources are shown in the following table.

Network names	**Network1**	**Network2**	**Network3**	**Network4**
source	Computer simulations	Staphylococcus aureus	Escherichia coli	Brewer's yeast

Table 1: Data exploration

For the golden standard regulatory network, the gene regulation data of E. coli and Saccharomyces cerevisiae are experimentally validated[4][5]. In contrast, for computer simulated data, the network is known. The S. aureus network cannot be measured accurately because there are relatively few experimentally supported interactions available. Nevertheless, DREAM confirmed the accuracy of the predictions by integrating the predictions of all teams using an average ranking approach and by evaluating the network using a large number of computationally derived interactions from the RegPrecise[6] database. Since the data are derived from real experiments, the size and sparsity of the data are very irregular, which are as follows.

Network names	**Network1**	**Network2**	**Network3**	**Network4**
TF numbers	195	99	334	333
Gene numbers	1643	2810	4511	5950
Sample numbers	805	161	805	536
Regulating network sparsity	0.01252	0.001581	0.001371	0.001927

Table 2: Network sharp information

The sparsity of all four networks is between 0.001 and 0.002. In the next experiments, we will first make predictions for these four modulated networks and compare them with the public rankings of the DREAM community, and then we will try to generate the data by ourselves using linear synthesis and explore the relationship between accuracy and factors such as data size and sparsity.

2.2 Scoring Metrics

Since the reference network matrix in this dataset is a matrix consisting of only 0 and 1, where 1 represents the presence of a relationship and 0 represents the absence of a relationship. That is, our prediction target is an undirected and unweighted network. For this matrix, we use the AUC (area under the ROC curve) to measure the accuracy of the prediction, which is often used to evaluate the accuracy of classification models, and the problem can also be regarded as classifying the relationships between genes into ”conductive” and ”non-conductive” categories. The AUC can also be used to evaluate the relationship between genes as ”conductive” and ”not conductive”. The calculation formula is as follows.

$$AUC = \frac{\sum pred_{pos} > pred_{neg}}{positiveNum * negativeNum} \tag{3}$$

Notice that in this problem positive and negative cases are uneven (due to the sparsity of the network), while the AUC calculation method considers the classification ability for both positive and negative cases, and is still able to make a reasonable evaluation of the classifier in the case of unbalanced samples. Therefore, AUC is not sensitive to whether the sample categories are balanced or not, and is suitable as a scoring criterion for this task.

2.3 Sparse optimization model

This class of algorithms always accepts an $m * n$ matrix $\mathbf{A}$ with a $1 * m$ vector $\mathbf{b}$ and outputs a sparse $1 * n$ dimensional vector $\mathbf{x}$. In this problem, the non-zero elements in the vector $\mathbf{x}$ represent the presence of a regulatory relationship between the transcription factors in the matrix $\mathbf{A}$ and the target genes in $\mathbf{b}$. And based on how to select out the non-zero elements in $\mathbf{x}$ the algorithm can be divided into two categories: greedy algorithms based on matching tracking and compressed perception algorithms based on convex optimization. These two methods are described separately in the following. To distinguish our voting algorithm introduced later, we call the sparse optimization algorithms in this section independent algorithms.

2.3.1 Convex optimization

The essence of the algorithm based on convex optimization is to solve the regularized regression problem with $L1$ parametrization showed as equation (4):

$$\min_{w} \frac{1}{2n_{samples}} \|Xw - y\|_2^2 + \alpha \|w\|_1 \tag{4}$$

Since this loss function is a non-convex function, we need to find an optimization algorithm to solve it. Commonly used algorithms are coordinate descent method, ISTA algorithm, etc.

Coordinate descent is a non-gradient optimization algorithm that performs a one-dimensional search along a coordinate direction at the current point in each iteration to find the local minima of a function. Different coordinate directions are recycled throughout the process.It has been proved by Jerome Friedman(2010)[7] that the coordinate descent method is able to converge to a sparse solution in solving the Lasso problem.

The ISTA algorithm continues to solve the least squares problem with L1 parametrization with the iterative formulation:

$$\mathbf{x}_{k+1} = \mathrm{soft}_{\lambda t} \left(\mathbf{x}_k - 2t\mathbf{A}^{\mathrm{T}} \left(\mathbf{A}\mathbf{x}_k - \mathbf{y} \right) \right) \tag{5}$$

where $soft$ is the soft threshold operator function:

$$soft_T(\mathbf{x}) = sign\left(x_i\right)\left(|x_i| - T\right) \tag{6}$$

Beck(2009) had proved that when the iteration step is taken as the inverse of the Lipshitz constant[8], the time complexity of the algorithm is $O(\frac{1}{k})$.

2.3.2 greedy algorithms

The core idea of the greedy algorithm is to select the atom that is closest to the residual of the current sample in each iteration. The resulting algorithms are OMP, CoSaMP, SP, IHT, etc.The core idea of this type of algorithm is to select one or more atoms with the highest correlation in each iteration to be added to the solution set as non-zero variables.

Take OMP as an example. For matrix A and residuals r, first:

$$i = \arg\min_i \left| A_i^T r_k \right| \tag{7}$$

Then:

$$P_k = A_S \left(A_S^T A_S \right)^{-1} A_S^T, \quad r_k = (I - P_k)\, b \tag{8}$$

After repeating the above steps k times, return:

$$x_S = \left(A_S^T A_S \right)^{-1} A_S^T b \tag{9}$$

Now x contains the k most relevant elements. Joel A Tropp(2004) had proved OMP algorithm can converge in the absence of noise[9]. T. T. Cai(2011) had proved the convergence in the presence of noise[10].

The CosaMP algorithm, on the other hand, modifies the approach of selecting elements based on OMP. It calculates the subscript of θ by the equation(10) and selects the first K larger as the final output solution.

$$\hat{\theta}_t = \arg\min_{\theta_i} \|y - A_t \theta_t\| = \left(A_t^T A_t \right)^{-1} A_t^T y \tag{10}$$

Needell, D(2009) pointed out that CoSaMP delivers the same guarantees as the best optimization-based approaches. Moreover, this algorithm offers rigorous bounds on computational cost and storage. It is likely to be extremely efficient for practical problems because it requires only matrix-vector multiplies with the sampling matrix[17].

The SP algorithm is very similar to the CosaMP algorithm, and their most important differences are, in each iteration, in the SP algorithm, only K new candidates are added, while the CoSAMP algorithm adds $2K$ vectors.Dai W(2009) recognized that the SP algorithm has similar properties and advantages and disadvantages to the CoSaMP algorithm, but the SP algorithm is computationally more efficient[18].

IHT is not a convex optimization algorithm, it is similar to OMP, which is an iterative algorithm, but it is a greedy algorithm derived from an optimization problem. Therefore, it has the characteristics of two types of algorithms.IHT algorithm for solving least squares problems with L1 parametrization:

$$C_{\ell_0}(\mathbf{y}) = \|\mathbf{x} - \mathbf{\Phi y}\|_2^2 + \lambda \|\mathbf{y}\|_0 \tag{11}$$

This is not an easy problem to optimize, so the following equation is introduced as an alternative objective function.

$$C_{\ell_0}^S(\mathbf{y}, \mathbf{z}) = \|\mathbf{x} - \mathbf{\Phi y}\|_2^2 + \lambda \|\mathbf{y}\|_0 - \|\mathbf{\Phi y} - \mathbf{\Phi z}\|_2^2 + \|\mathbf{y} - \mathbf{z}\|_2^2 \tag{12}$$

It is easy to prove[19] the equations(13) and (14)

$$C_{\ell_0}^S \geq C_{\ell_0} \quad \forall z \tag{13}$$

$$\exists z = y \quad s.t. \quad C_{\ell_0}^S = C_{\ell_0} \tag{14}$$

According to the Majorization-Minimization[20] idea, we can optimize $C_{\ell_0}^S$ as an alternative function. The final obtained iterative formula is as follows.

$$\mathbf{y}^{n+1} = H_M\left(\mathbf{y}^n + \Phi^H\left(\mathbf{x} - \Phi\mathbf{y}^n\right)\right) \tag{15}$$

$$H_M\left(y_i\right) = \begin{cases} 0, & if\ |y_i| < \lambda_M^{0.5}(\mathbf{y}) \\ y_i, & if\ |y_i| \geq \lambda_M^{0.5}(\mathbf{y}) \end{cases} \tag{16}$$

2.3.3 Voting algorithm

For the above methods, there is no study to prove which is the optimal method for gene regulatory network reconstruction. We also found that the optimal method varies for different data through subsequent experiments. Therefore, we will develop a voting algorithm based on the above methods to obtain better prediction results for any data set.

The idea of the voting algorithm is simple: since every algorithm has some possibility of making mistakes, the result that is approved by most algorithms should have relatively high accuracy. For the modulation relation at the (i, j) position, we have the following algorithm:

$$N_{i,j} = \begin{cases} 1, & if\ \sum_{k=1}^{n} N_{k,i,j} > a \\ 0, & if\ \sum_{k=1}^{n} N_{k,i,j} \leq a \end{cases} \tag{17}$$

The meaning of the parameter a is that when a of the algorithms participating in the voting believe that a moderation relation exists here, it is recognized as such. The setting of a also affects the sparsity of the final result. A smaller value of a yields a denser matrix - the result is more likely to contain the correct moderation relation but is also more likely to treat a non-existent moderation relation as the present.

Take a system containing five transcription factors as an example: Transcription factors A, B, C, D, and E may have potential regulatory relationships for the target gene T. Five algorithms are used to make predictions for this network. Assume that the gold standard for this set of regulatory relationships is the equation:

$$A * \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T = b \tag{18}$$

The following predictions were given by the five algorithms, S_1 to S_5:

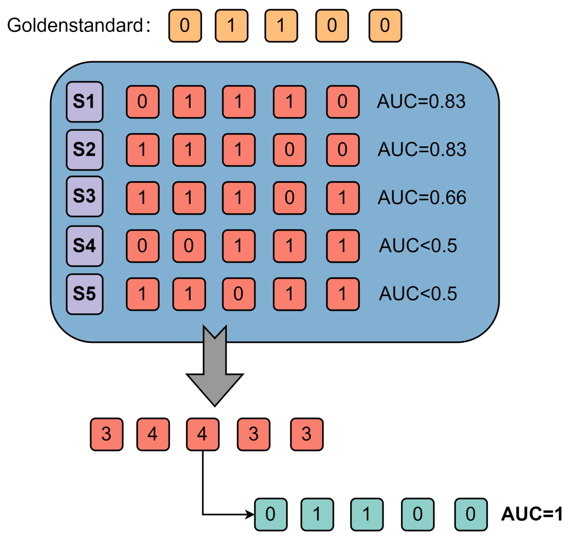


Figure 2: An example of voting algorithm

As we can see from the above figure, the algorithms from A to E contribute 5 weaker predictions respectively, which even have $AUC < 0.5$. But after the integration of the voting algorithms, the final result of the perfect classifier with $AUC = 1$ is obtained.

2.4 work flow

Solving this problem is equivalent to solving a system of linear equations(1). We want to obtain the sparse solution of x. where A is the expression matrix of transcription factors, x is the sparse gene regulation matrix, and b is the expression data of the target gene.

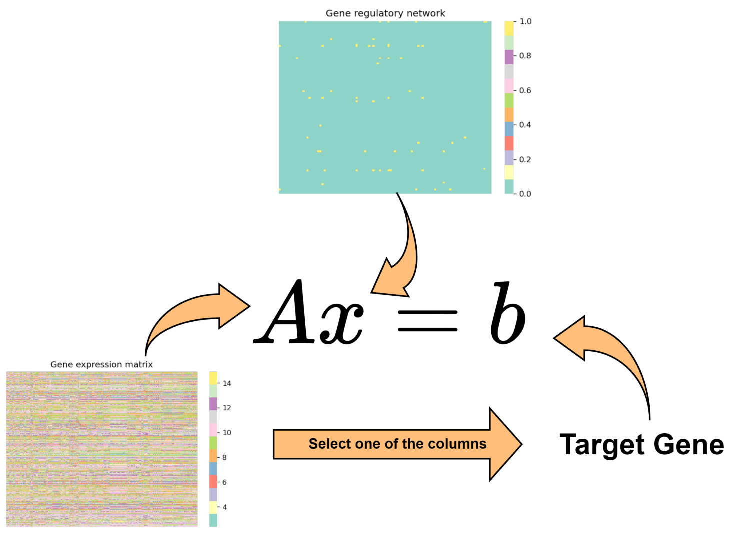


Figure 3: **Linear equations:**This image gives the general form of the gene regulation network prediction problem and what each variable in this system of linear equations represents

We will normalize the entire network inference process to facilitate the subsequent packaging of the code into packages for distribution. First, we will receive four files, which are the gene expression matrix, the transcription factor list, the gene list, and the gold standard for gene regulatory networks. The gold standard of the gene regulatory network can be missing, but then it will not score the final prediction results. The rows and columns contained in each file are shown in the following table:

File names	Introduction
Gene names	A column of genetic names.
TF names	A column of transcription factors names.
Expression matrix	Each column represents a gene and each row represents the expression level of different genes in the same sample.
network	A side list, the first column represents the transcription factor, the second column represents the target gene regulated by that transcription factor, and the third column is the type of regulation: 1 indicates the presence of a regulatory relationship and 0 indicates the absence of a regulatory relationship.

Table 3: Meaning of the input files

After importing the data, we first process the network data (if available) into the form of a matrix. Where the (i,j) position if 1, indicates that the No.j transcription factor has a regulatory relationship with the No. i target gene. We then traversed the entire gene expression matrix, testing each gene as a target gene from the first to the last. It is worth noting that since the target gene itself may also be a transcription factor, in this case, we must first manually delete the column where this gene is located and then insert a zero at that position in the result matrix. otherwise, it will result in an unusually large coefficient at that point. After testing all the independent algorithms, we bagged them using the voting algorithm, calculated their AUC values separately, and performed a comparative analysis.

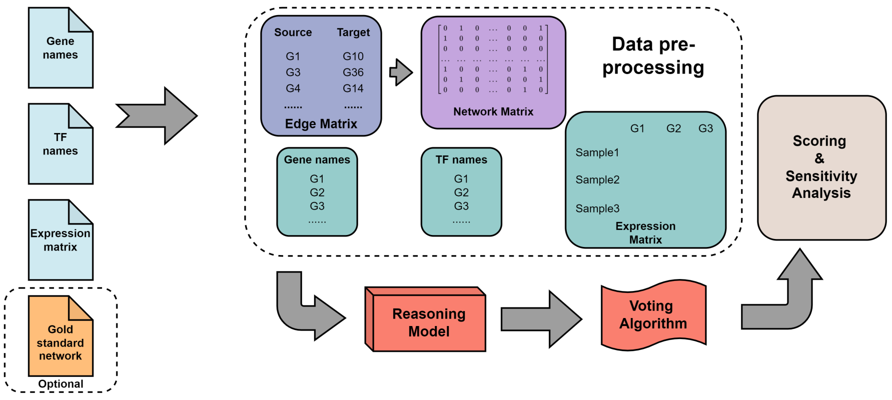


Figure 4: Our workflow

2.5 Sparse tools box

2.5.1 Random data generation

To explore the effects of network size, the number of transcription factors, and other factors on prediction accuracy. We synthesized simulated data based on the following principles. The task of the DREAM challenge is to solve for x when A and b are known. to explore the effects of the shapes of both A and b and the sparsity of x on the experimental results, we generate A randomly and a sparse matrix x with specified sparsity and $Ax + \mu$ as b, the relationships are shown below:

$$\begin{aligned} &A * x = b + \mu \\ &\mu \sim N(normalLoc, normalScale) \end{aligned} \tag{19}$$

The algorithm generates 10 transcription factors, 200 samples, and 2000 target genes by default. The expression matrix obeys a [0,1] uniform distribution and the regulatory network obeys a [-1,1] uniform distribution. For each set of regulatory relationships, 20% ± 10% of the transcription factors are involved in regulation, that is, the sparsity is around 10% to 30%.

2.5.2 Reasoning algorithms

In addition to the random data generation function, the gene regulatory network reconstruction algorithm mentioned above is also integrated into the package. We combine the above-mentioned independent algorithms with the voting algorithm, and the user only needs to simply call the API to implement the network inference process. For detailed usage, the format of input and output can be found in https://pypi.org/project/Sparse-Optimization-Toolbox/

3 Result

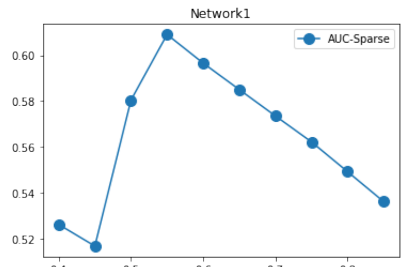
3.1 Parameter Selection

In this problem, the most important hyperparameter is the sparsity of the matrix, and we need to tell the algorithm how many non-zero elements are contained in the final regulatory network. For different matrices, the sparsity varies and is not regular. However, based on biological experiments, some of the regulatory relationships in a system are already proven. Therefore, we take the top 20% of the regulatory relationships in the data set as a priori data and calculate their AUC values by varying the sparsity, to choose a better sparsity for each network.The sparsity there refers to:

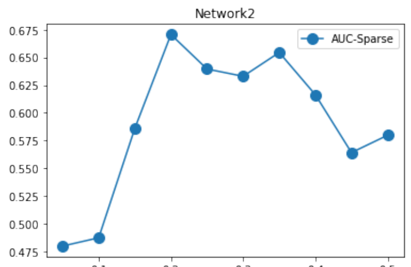
$$sparsity = \frac{S_a}{S_b} \tag{20}$$

Where S_a is the number of non-zero items in a matrix and S_b is the numbers of items in a matrix.

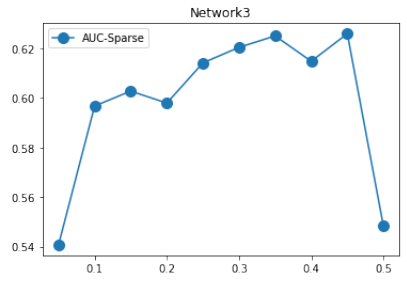
By experimenting with different sparsity selected with 20% of the data, we selected their optimal sparsity for the four networks.The experimental results are as follows:



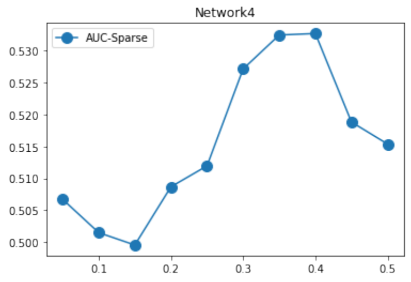
(a) Network1



(b) Network2



(c) Network3



(d) Network4

Figure 5: AUC of four networks with different sparsity

Considering that it takes more time to reason a matrix containing more non-zero elements, we finally choose the optimal sparsity as 0.55, 0.2, 0.35, 0.35. As the sparsity rises, the time consumed by inference is shown in Figure 3.

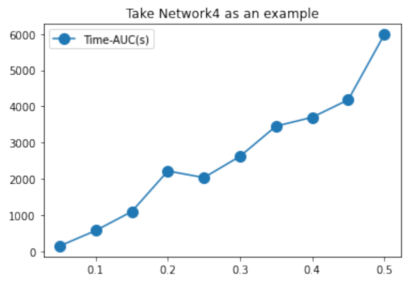


Figure 6: The relationship between time consumption and sparsity

Meanwhile, the threshold a of the voting algorithm also affects the sparsity of the result. Therefore, we use the same method as above for different values of a and obtain the following results.

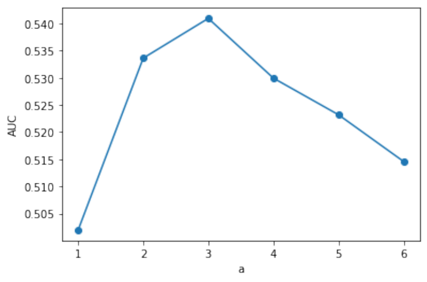


Figure 7: Parameters of the voting algorithm

The results show that the algorithm has the highest AUC when $a = 3$. This means that when more than three of the six independent algorithms give positive judgments, we should assume that there is indeed a moderating relationship here.

3.2 Prediction results for the DREAM5 datasets

Based on the M1 Pro Apple silicon platform, we used the above method to predict the four modulation networks and obtained the following test scores and time consumption:

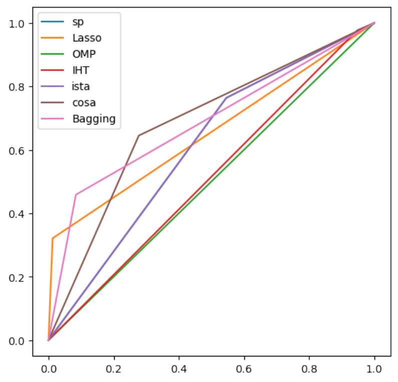
	Network1	**Network2**	**Network3**	**Network4**
Lasso	0.6873	0.5554	0.5601	0.5295
OMP	0.6543	0.5322	0.5806	0.5117
ISTA	0.5173	0.5338	**0.7093**	0.5211
CoSaMP	0.6115	0.6549	0.5735	0.5117
SP	0.6090	0.6669	0.6224	0.5325
IHT	0.5013	0.7570	0.5027	0.5049
Voting	**0.6878**	**0.7698**	0.5962	**0.5409**

Table 4: AUC result

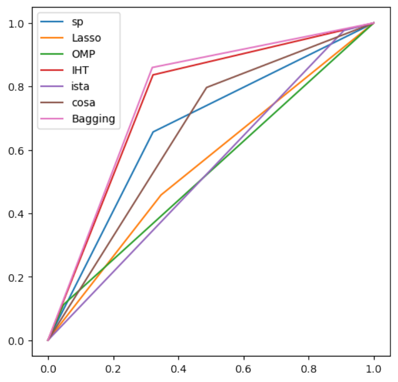
	Network1	**Network2**	**Network3**	**Network4**
AUC	0.809	0.5845	0.65	0.528

Table 5: result from DREAM5

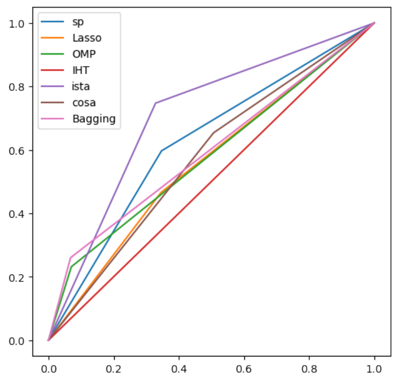
The results show that the accuracy of our algorithm exceeds the official DREAM5 ranking for Network2, Network3, and Network4 inference.The voting algorithm performs well in the inference of all four networks and is the best performing algorithm in Network1, Network2, and Network4. Except for the prediction for Network2, the IHT algorithm performs poorly with an AUC around 0.5. To our surprise, the traditional Lasso and OMP algorithms, although not the best performers, always give a satisfactory prediction.



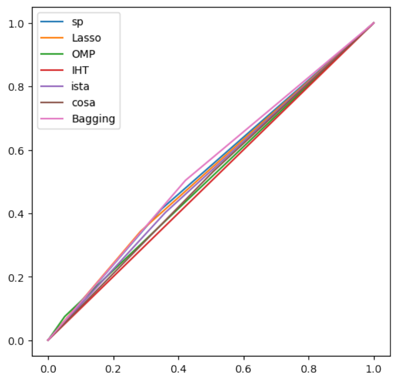
(a) Network1



(b) Network2



(c) Network3



(d) Network4

Figure 8: AUC of four networks with different algorithms

However, except for the voting algorithm, the performance of the other independent algorithms varies greatly across the data sets. For example, the coordinate descent-based Lasso algorithm performs well in Network1, while it is in the bottom position in the rest of the datasets.The best performer in Network2 was the IHT algorithm, ISTA achieved the highest score in Network3, and the best algorithm in Network4 was the SP algorithm.

In terms of time complexity, shown as table 6, the OMP algorithm always takes the shortest time. Among the independent algorithms, CosaMP takes the longest time, even close to 24 hours in the inference for Network3 and Network4.Because the voting algorithm is developed on top of the rest of the independent algorithms, it requires the longest inference time.

	Network1	**Network2**	**Network3**	**Network4**
Lassotime	25m24s	37m35s	53m36s	1h21m6s
OMPtime	15m06s	13m38s	31m43s	30m41s
ISTAtime	5h48m13s	1h14m33s	8h2m42s	7h3m31s
CoSaMPtime	4h22m56s	42m28s	23h19m43s	20h22m18s
SPtime	3h17m44s	43m45s	8h4m26s	7h4m33s
IHTtime	2h21m56s	33m19s	15h30m46s	13h1m53s
Votingtime	15h53m05s	3h45m15s	56h28m25s	49h54m32s

Table 6: Time consumption of the algorithms

Then, we calculated the average performance and average elapsed time of the four networks and obtained the following figure.

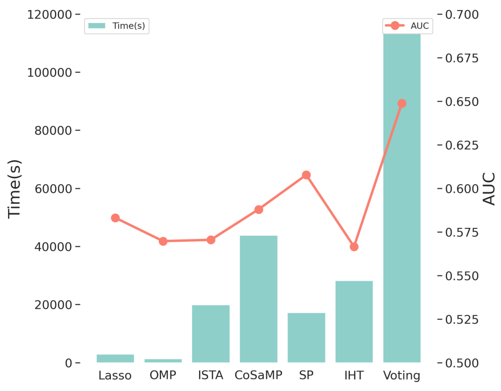


Figure 9: time & AUC

From that result we can find that the cost of good performance of the voting algorithm is much higher than the time consumption of the independent algorithm. Among the independent algorithms, SP clearly has the best performance, which takes little time but has the highest accuracy rate.

3.3 Sensitivity analysis with simulated data

In the above experiments, we found that for the same algorithm, the accuracy may be related to factors such as the number of transcription factors and the number of samples. Therefore, we used the data generation function of Sparse Tools Box to generate data of different sizes for our experiments. First, we tested the change of the prediction accuracy when the number of transcription factors changed from 10 to 210, controlling other variables constant

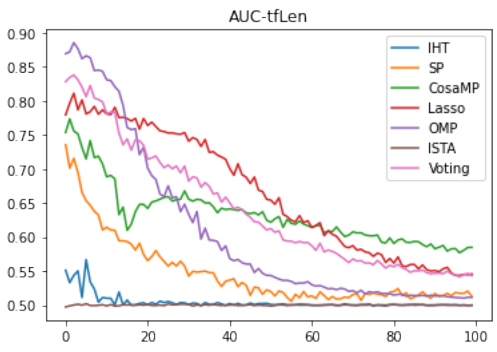


Figure 10: AUC-tfLens

We can find that the accuracy of all types of algorithms decreases as the number of transcription factors rises. This is due to the fact that the rising number of transcription factors implies the expansion of network size, and the more the system of linear equations to be solved tends to be pathological at a certain number of samples. The difficulty of calculating the exact solution increases greatly at this point.In this set of experiments, the SP algorithm and the OMP algorithm were more influenced by the number of transcription factors. There is a large decrease in accuracy right before the number rises. The CosaMP algorithm, as an improved version of the OMP algorithm, has a good performance, and the accuracy is more stable, and the best result is obtained when the number of transcription factors reaches the maximum.

Next, the effect of sample size on accuracy is tested. We set the sample size to vary from 40 to 300 and setting parameters $tfLen = 10$, $geneNums = 200$ and $noise = 0.1$.

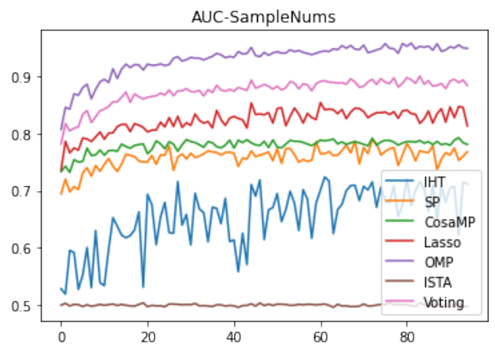


Figure 11: AUC-sampleNums

The accuracy of each algorithm increases significantly as the number of samples increases. Considering from the perspective of the system of linear equations, as the number of samples rises, the pathological nature of the system of linear equations subsequently decreases and exact solutions are more easily obtained. On the other hand, from the machine learning point of view, the rising number of samples corresponds to providing more independent and identically distributed data for the algorithms to learn. In the ideal case of no noise, as long as these data obey the same distribution, the accuracy of the final learning should also be monotonically undiminished.

Finally we consider noise. We let the mean value of the noise vary between 0 and 1, setting the other parameters $tfLen = 10$, $geneNums = 200$ and $sampleNums = 40$.The resulting accuracy variation curves are as follows.

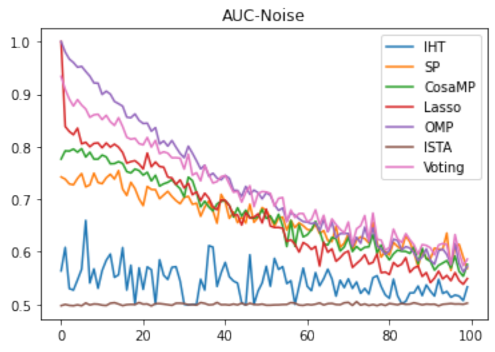


Figure 12: AUC-noise

As can be seen from the figure, the traditional OMP and Lasso algorithms are more affected by noise, while the CosaMP and SP algorithms and the voting algorithm perform better.

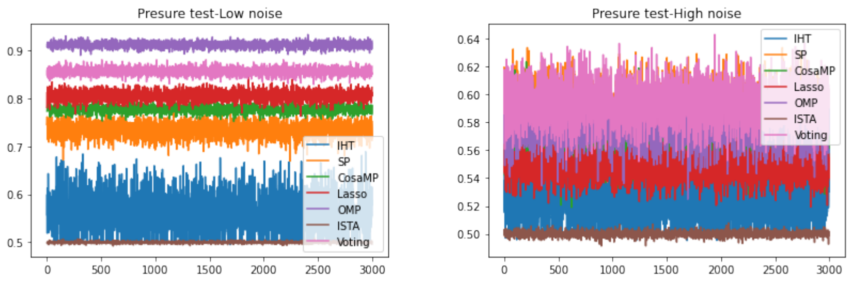
3.4 Stress Test

To test the stability of the package, we will repeat a large number of experiments to record parameters such as average consumption time, variance and mean of accuracy. We tested the data generation-network inference in the low-noise case and the high-noise case, respectively, and the parameters for both cases were chosen as shown in the following:

	tfLen	**sampleNums**	**geneNums**	**variance of noise**
Zero	10	40	200	0
Low	10	40	200	0.15
High	10	40	200	1

Table 7: Pressure test parameter

First we tested the stability of the algorithm in the low-noise and high-noise cases, with the following results.



(a) Low noise (b) High noise

Figure 13: Stress test in low noise and high noise

	Low-noise		**High-noise**	
	Std.	**mean-AUC**	**Std.**	**mean-AUC**
IHT	0.03630	0.5596	0.01718	0.5261
SP	0.01478	0.7269	0.01435	0.5892
CosaMP	0.00759	0.7630	0.01466	0.5728
Lasso	0.01014	0.7868	**0.009006**	0.5481
OMP	**0.007466**	**0.8758**	0.01216	0.5775
ISTA	0.001853	0.4999	0.001840	0.4999
Voting	0.008284	0.8320	0.01373	**0.5934**

Table 8: Accuracy mean and standard deviation in two cases

From the experimental results, we found that the ISTA algorithm performed poorly in repeated experiments because it relied too much on manual tuning of the parameters, and its score tended to be a random classifier. Therefore, we compare several algorithms other than ISTA, and we find that the coordinate descent-based Lasso algorithm and OMP algorithm have higher stability and accuracy in the case of low noise.And for the high noise case, we can find that the mean of accuracy of each algorithm decreases and the standard deviation increases. In this case, our voting algorithm can take advantage of the strengths of all the algorithms and achieve the best prediction results.

We also tested the noise-free case, in which the Lasso algorithm with the OMP algorithm was able to reconstruct the gene regulatory network nearly perfectly.

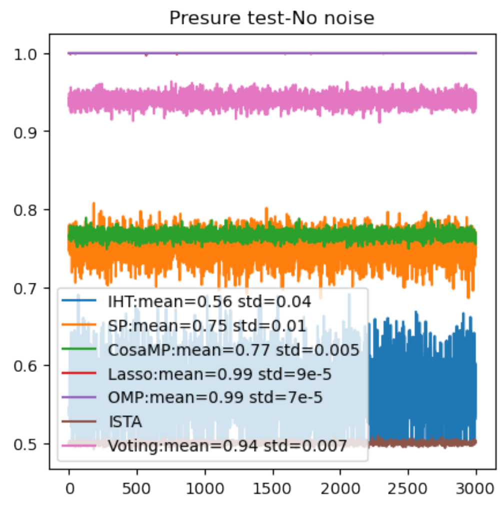


Figure 14: No noise case

4 Discussion

In this paper, we start from the DREAM5 dataset and reason about the four networks it contains using various independent methods with a bagged voting algorithm. Through experiments, we find that there does not exist a general algorithm that can make the reasoning for each network have the best results. The voting algorithm we developed can perform the task better but at the cost of higher time complexity. For the independent algorithms, we can find by analyzing the results that the traditional Lasso regression and OMP algorithms can achieve better results with shorter time consumption for smaller and less sparse networks. However, it was also found that for larger-scale network inference (Network3 & Network4), improved algorithms such as ISTA and SP were able to provide better prediction accuracy. For the differences arising between traditional and improved algorithms, subsequent experiments based on simulated data offer one possibility: noise. algorithms such as OMP have good convergence for noise-free data, and in some cases, even exact reductions can be obtained. However, as the noise rises, their accuracy decreases very quickly.

In a real scenario, we believe that a voting algorithm that takes into account both high and low noise environments is a good choice at the application level since there is no Golden Standard for us to compare scores against.

However, when we look at the final obtained prediction network, we also find some interesting facts.The predictor is biased to give a less sparse solution as a way to cover as many of the regulatory relationships proven in Golden Standard as possible. This increases the resulting TPR (the percentage of all samples that are actually positive that are correctly judged as positive) and also increases the FPR (the percentage of all samples that are actually negative that are incorrectly judged as positive).

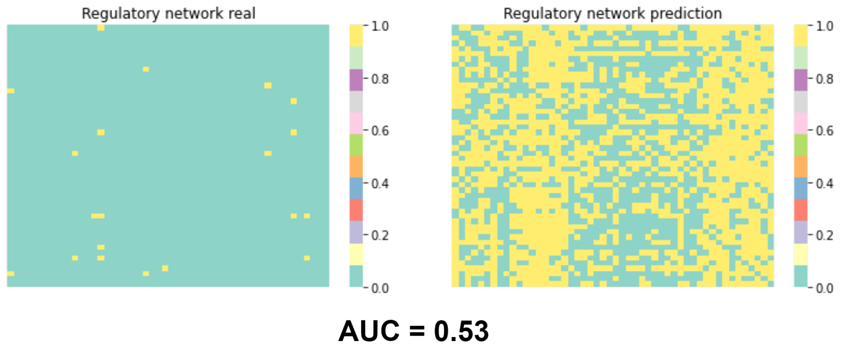


Figure 15: Result GRN of network4

For a community competition, we believe that submission of all loci that are thought to have a possible regulatory relationship should be encouraged. This is because we can still vote the algorithm on the results of all the submissions that participated in the competition. If there is a locus that most contestants believe has a regulatory relationship, Gold Standard does not. Then it would be a good time to modify Gold Standard.

In the second half of the paper, we focus on the sensitivity of the sparse optimization algorithm to noise on this problem. And it is found that different algorithms will have different performances in different noise environments. Many biologists and mathematicians have experimentally demonstrated that cells are by nature biochemical reactors with noise. Both stochasticity inherent in the biochemical process of gene expression (intrinsic noise) and fluctuations in other cellular components (extrinsic noise) contributes substantially to overall variation. Transcription rate, regulatory dynamics, and genetic factors control the amplitude of the noise. These results establish a quantitative foundation for modeling noise in genetic networks and reveal how low intracellular copy numbers of molecules can fundamentally limit the precision of gene regulation[22]. Thattai, M(2001) found in his study that the Fano factor(defined as the ratio of the variance to the mean) is about 20 in the steady-state case. The corresponding steady-state mean value is about 500[23]. Therefore, the noise variance can be calculated to be about 10000. This is also a good example of why traditional algorithms do not perform well on real genetic datasets because they lack stability and convergence on noisy data.

On the other hand, we also need to focus on the difficulty of obtaining samples. Since there are millions of genes in an organism, we necessarily cannot experiment with millions of living individuals. This means that a linear set of equations of gene regulation is always underdetermined. The application of sparse solution problems of underdetermined linear systems of equations to image processing has been studied extensively[24]. This is mainly used to reconstruct a clear image from a noise-laden image. In this study, the sparse optimization method was introduced into the gene regulatory network prediction problem based on the characteristic that gene expression data are difficult to obtain, and it was demonstrated experimentally that good results were obtained. However, this paper is mainly based on the existing algorithms for validation and experiments and does not improve the algorithm from the perspective of the similarities and differences between genetic data and image data. The specific algorithm for genetic data needs to be studied in the future.

References

[1] Emmert-Streib, F., et al. (2014). ”Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks.” Frontiers in Cell and Developmental Biology 2.

[2] Bansal, M., et al. (2007). ”How to infer gene networks from expression profiles.” Molecular Systems Biology 3.

[3] https://dreamchallenges.org/dream-5-network-inference-challenge/

[4] Gama-Castro S, et al. RegulonDB version 7.0: transcriptional regulation of Escherichia coli K-12 integrated within genetic sensory response units (Gensor Units) Nucleic Acids Res. 2011;39:D98–105.

[5] Harbison CT, et al. Transcriptional regulatory code of a eukaryotic genome. Nature. 2004;431:99–104.

[6] Ravcheev DA, et al. Inference of the Transcriptional Regulatory Network in Staphylococcus aureus by Integration of Experimental and Genomics-Based Evidence. Journal of Bacteriology. 2011;193:3228–3240.

[7] Friedman, J., et al. (2010). ”Regularization Paths for Generalized Linear Models via Coordinate Descent.” Journal of Statistical Software 33(1): 1-22.

[8] Beck, Amir, and Marc Teboulle. ”A fast iterative shrinkage-thresholding algorithm for linear inverse problems.” SIAM journal on imaging sciences 2.1 (2009): 183-202.

[9] Joel A Tropp. Greed is good: Algorithmic results for sparse approximation. IEEE Transactions on Information theory, 50(10):2231–2242, 2004.

[10] T. T. Cai and L. Wang. Orthogonal matching pursuit for sparse signal recovery with noise. IEEE Transactions on Information Theory, 57(7):4680–4688, 2011.

[11] Z. Fang, ”Comparisons of Graph-Structure Clustering Methods for Gene Expression Data”, Acta Biochimica et Biophysica Sinica, vol. 38, no. 6, pp. 379-384, 2006.

[12] Z. Xiing and D. Wu, ”Modeling Multiple Time Units Delayed Gene Regulatory Networks Using Dynamic Baeysian Networks”, Proc. IEEE Int’l Conf. Data Mining, pp. 190-195, 2005.

[13] L. Nie, G. Wu, W. Zhang, L. Nie, G. Wu and W. Zhang, ”Correlation of mRNA Expression and Protein Abundance Affected by Multiple Sequence Features Related to Translational Efficiency in Desulfovibrio Vulgaris: A Quantitative Analysis”, Genetics, vol. 147, pp. 2229-2243, 2006.

[14] B. Perrin, ”Gene Networks Inference Using Dynamic Bayesian Networks”, Bioinformatics, pp. 138-148, 2003.

[15] X. Hu, M. Ng, F. Wu and B. Sokhansanj, ”Mining Modeling and Evaluation of Sub-Networks from Large Biomolecular Networks and Its Comparison Study”, IEEE Trans. Information Technology in Biomedicine, vol. 13, no. 2, pp. 184-194, Mar. 2009.

[16] Wang, S. D., et al. (2015). ”Structural Analysis of Leukemia Related Gene Network.” Journal of Computational and Theoretical Nanoscience 12(7): 1445-1451.

[17] Needell, D. and J. A. Tropp (2009). ”CoSaMP: Iterative signal recovery from incomplete and inaccurate samples.” Applied and Computational Harmonic Analysis 26(3): 301-321.

[18] Dai W Milenkovic O Subspacepursuit for compressive sensing signal reconstruction[J] IEEETransactions on Information Theory 2009 55(5) 2230-2249.

[19] Blumensath T, Davies M E.Iterative Thresholding for Sparse Approximations[J]. Journal of Fourier Analysis & Applications,2008, 14(5):629-654.

[20] Julien Mairal. “Incremental majorization-minimization optimization with application to large-scale machine learning”. SIAM Journal on Optimization, 25(2):829–855, 2015.

[21] Qin, J., et al. (2014). "Inferring gene regulatory networks by integrating ChIP-seq/chip and transcriptome data via LASSO-type regularization methods." Methods 67(3): 294-303.

[22] Elowitz, M. B., Levine, A. J., Siggia, E. D., & Swain, P. S. (2002). Stochastic gene expression in a single cell. Science (New York, N.Y.), 297(5584), 1183–1186.

[23] Thattai, M. and A. van Oudenaarden (2001). "Intrinsic noise in gene regulatory networks." Proceedings of the National Academy of Sciences of the United States of America 98(15): 8614-8619.

[24] Bruckstein, A. M., et al. (2009). "From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images." Siam Review 51(1): 34-81.

[25] Mostafa Akhavan-Safar, Babak Teimourpour.(2021) "KatzDriver: A network based method to cancer causal genes discovery in gene regulatory network." Biosystems, Volume 201.

[26] Marbach, D., et al. (2012). "Wisdom of crowds for robust gene network inference." Nature Methods 9(8): 796-+.