MACHINE LEARNING IN CANCER GENOMICS

Student Name: Taejas Gupta
Roll Number: 2016204

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering
on April 28, 2019

BTP Track: Research

BTP Advisor
Dr. Anubha Gupta

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **"Machine Learning in Cancer Genomics"** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Anubha Gupta**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

..... **Place & Date: .....**
Taejas Gupta

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

..... **Place & Date: .....**
Dr. Anubha Gupta

Abstract

Cancer is caused by an increased amount of cell growth in an area due to alterations in protein synthesis caused by mutations in certain genes known as cancer driver genes. Determining the gene regulatory network for the genes involved in the pathway can enable us to identify the driver genes that are responsible for the cancer, so that drugs targeting such genes can be developed. Advancements in DNA microarray technologies have made time series gene expression level data available for further analysis to infer the underlying gene regulatory network. My goal in this project is to infer gene regulatory networks from time series gene expression datasets using machine learning approaches, with a focus towards recurrent neural networks and iteratively reweighted least squares with decorrelation. This report covers the various research papers that I studied, the approaches that I took and the results that I obtained for the task of inferring the gene regulatory network from time series gene expression data.

Keywords: Gene regulatory networks, machine learning, recurrent neural networks, iteratively reweighted least squares, decorrelation

Acknowledgments

I thank my advisor, Dr. Anubha Gupta, for suggesting the research problem to me and giving me an opportunity to work under her guidance. Her advice and constant support have enabled me to think critically about the domain of research and learn to come up with concise, effective solutions to practical problems.

I thank Ms. Akanksha Farswan (PhD, SBILab) for her immense contribution to the project. Her vast knowledge of the various methods pertaining to genomics and readiness to brainstorm solutions for the challenges along the way have been a significant driver to this project.

Contents

Chapter 1

Introduction

Genes are DNA sequences that lead to the formation of mRNA via the process of transcription, which further lead to the synthesis of proteins and other functional gene products. Gene expression refers to the process by which functional gene products are synthesised from genes. The functional gene products synthesised by a gene can act as transcription factors for other genes by binding to their transcriptional regulatory sites, controlling their gene expression levels. Gene regulatory networks provide information about the regulatory actions that take place between genes. These regulatory actions can be of two types -- activation and repression. Activation causes an increase in the expression level of the gene being regulated, whereas repression causes a decrease in the expression level. A gene may thus activate, repress, or have no effect on the expression level of another gene.

In cancer, genetic changes caused by mutations lead to alterations in gene expression. For example, a mutated gene may cause the activation of another gene that leads to the surplus synthesis of a protein responsible for cell growth, resulting in a tumour. Such mutated genes that increase net cell growth leading to a tumour are known as cancer driver genes. If these cancer driver genes can be correctly identified, it can enable the design of drugs targeting those genes for developing potential cures. Not all mutations in the gene sequence are malignant, thus a direct analysis of the gene sequence may not help in identifying cancer driver genes. Moreover, the proteins that are ultimately responsible for the tumour may differ from those that are synthesised by the cancer driver genes; the cancer driver genes may be indirectly responsible for their regulation. Thus, determining cancer driver genes is not a trivial task.

Determining the gene regulatory network can assist in backtracking the regulatory actions from the gene that is causing the synthesis of the protein responsible for the observable tumour to the genes that regulate it. Comparing the gene regulatory network and the extents of regulation

for a person diagnosed with cancer with those of a healthy person can provide insights into the identification of cancer driver genes.

Chapter 2

Motivation and Research Problem

Advancements in DNA microarray technologies have enabled scientists to create time series datasets of gene expression levels. The observed gene expression levels are a result of the regulatory actions that take place between the various genes in the system. Many studies have been conducted to leverage time series gene expression level data for reverse engineering a gene regulatory network, the basis being that the interactions in the gene regulatory network define the expression levels of the genes. Several computational approaches such as Boolean networks, Bayesian models, correlation methods, clustering algorithms, recurrent neural networks and extreme learning machines have been used by researchers for performing this task.

None of the studies so far have been able to produce accurate results on datasets consisting of over a few tens of genes. A key issue that is encountered in the task of genetic regulatory network reconstruction is the limited number of time points for which the data is available. Most in vivo time series gene expression level datasets contain records for a few tens to a hundred time points. The number of genes that are to be analysed are often several orders of magnitude greater than the number of time points for which the data is available. This makes it difficult to draw inferences from the dataset.

Recurrent neural networks (RNN) are a type of artificial neural networks that contain loops. The output of a set of nodes at a time point in an RNN contribute as an input to the same set of nodes at the next time point. RNNs are thus suitable for modelling the behaviour of time series data in which the values of a set of nodes at the next time point is dependent on the current values of the same set of nodes. This is the case for the gene expression dataset from which the genetic regulatory network is to be inferred, thus making RNN a reasonable choice for the task. In this context, an RNN can be regarded as a feedforward neural network formed

by unfolding the RNN over time, with a layer for every time point in the dataset, and each layer containing a node corresponding to each gene in the dataset. Each node in every layer except the first layer has connections to every node in the previous layer, accounting for the regulatory effect that every gene in the system may have on a gene, including itself. Likewise, each node in every layer except the last layer has connections to every node in the next layer, indicating the regulatory effect that a gene may have on every other gene.

Iteratively reweighted least squares (IRLS) is an approach for solving simultaneous equations with the cost function being of the form of a p-norm. The task of inferring the regulations in a GRN from time series gene expression data can be converted to the task of finding the solution of a system of linear equations where the gene expression level of a gene at a time point is a linear function of the gene expression levels of all genes present in the GRN at the previous time point.

My research problem is to infer plausible gene regulatory networks from temporal gene expression data for potentially determining the driver genes causing various types of cancer. This report comprises of the work I have done over the course of this project.

Chapter 3

Research Approach and Work Done

Most of my initial efforts went into understanding the problem of reconstructing gene regulatory networks by studying the works of other researchers in the field, particularly those that incorporated an RNN approach or a variant of it. I studied machine learning and neural networks in parallel to get the technical knowledge required to understand the RNN approaches taken along with their merits and limitations. Two months into the semester, I got in touch with one of the authors of a paper that I was studying, which involved an RNN model using a bat algorithm inspired particle swarm optimisation approach. I got a preliminary version of their MATLAB code from him, which I further worked on for a month and a half. I then built a simplistic RNN model and an LSTM model using TensorFlow in a fresh attempt to determine the relations among genes. After trying a few other approaches, I finally started work on another particle swarm optimization based RNN approach that I coded from scratch in Python with another paper as its basis.

I started the semester by going over three papers involving gene interactions and biological network modelling with applied machine learning [1–3] suggested to me by my advisor. We then decided to take up the problem of gene regulatory network reconstruction from time series gene expression data and made [1] as the base. My objective was to work on the reconstruction of gene regulatory networks, starting with small scale gene networks having the ground truth values to compare with. I primarily focused on the E. coli SOS DNA repair network dataset (available here), which is one of the most extensively studied in vivo datasets for gene regulatory network reconstruction.

The first approach I followed [1] uses a recurrent neural network (RNN) for modelling the network dynamics with a swarm optimisation based approach, which the authors have referred

to as Bat Algorithm inspired Particle Swarm Optimisation (BAPSO). The approach aims to determine whether there exists a regulatory effect of one gene on another for every pair of genes in the network.

An RNN can be visualised as a feedforward neural network unfolded in time, as shown in Figure 3.1. For a dataset containing N genes and T time points, each layer in the RNN corresponds to a time point and has N nodes, where the i^{th} node in the j^{th} layer corresponds to the gene expression level of the i^{th} gene at the j^{th} time point.

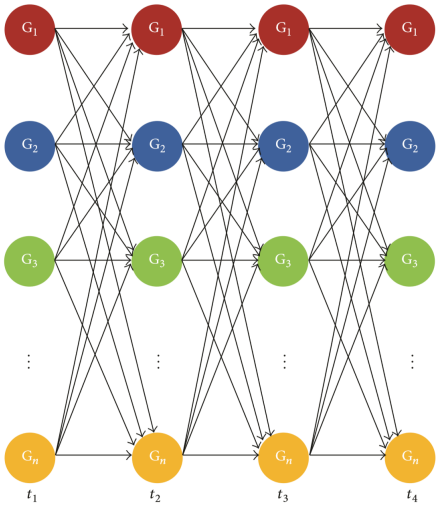

Figure 3.1: RNN representation of a network of n genes unfolded over 4 time points; image source: [1]

In light of this RNN structure, the regulatory effect of all genes on any gene i can be regarded as a weighted sum of the gene expression levels of all genes with an added bias term, as shown in the equation below.

$$g_i = \sum_{j=1}^{n} w_{i,j} x_j + \beta_i \tag{3.1}$$

Here, n is the number of genes in the network, $w_{i,j}$ is the weight of the edge from gene j to gene i, x_j is the gene expression level of gene j, and β_i is the bias term corresponding to gene i.

The rate of expression of a gene i accounts for the regulatory effects of other genes as well as the degradation of the gene, as shown in the equation below.

$$\frac{dx_i}{dt} = \chi_{1i} \cdot f(\sum_{j=1}^{n} w_{i,j} x_j + \beta_i) - \chi_{2i} x_i \tag{3.2}$$

Here, χ_{1i} is the rate constant of genetic expression for gene i, χ_{2i} is the rate constant of degradation for gene i, and f denotes the sigmoid function.

The problem is to come up with a structure that most accurately estimates the true values of the gene expression levels at every time point. The mean squared error is used for finding the deviation of the structure from the original network, with a lower value indicating a better solution. The corresponding equation is as follows.

$$MSE = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} (x_i(t) - \widetilde{x}_i(t))^2 \tag{3.3}$$

Here, N is the number of genes in the network, T is the number of time points, $x_i(t)$ is the true expression level of gene i and $\widetilde{x}_i(t)$ is the simulated expression level.

The BAPSO algorithm proposed by the authors has been used for training the RNN model. It is essentially the particle swarm optimisation (PSO) algorithm in which the inertia weight is updated by the bat algorithm (BA). PSO comprises of a set of particles, where each particle has an associated position and velocity. The position has the parameters to be optimised (weights, biases and time constants) as its dimensions, and each position corresponds to a potential solution. The velocity has the same number of dimensions as the position, and it is used to update the position of the particle. It accounts for the current direction of updation of the particle, the best position of the particle so far and the global best position from the best positions of all particles taken together. In each iteration of the algorithm, the new velocity v'_{pso_i} and position p'_{pso_i} of each particle are updated by the following equations.

$$v'_{pso_i} = w_I \otimes v_{pso_i} + r_1 c_1 \otimes (p^b_{pso_i} - p_{pso_i}) + r_2 c_2 \otimes (g^b - p_{pso_i}) \tag{3.4}$$

$$p'_{pso_i} = p_{pso_i} + v'_{pso_i} \tag{3.5}$$

Here, w_I is the inertia weight term, r_1 and r_2 are random numbers in the range $[0, 1]$, c_1 and c_2 are cognitive and social components that are set to a value of 2, $p^b_{pso_i}$ is the best solution achieved by particle i, g^b is the best solution achieved by the entire swarm so far, and $\otimes$ denotes elementwise multiplications.

Bat algorithm has been used for selecting the inertia weight term in the PSO velocity update equation 3.4 using the following equation.

$$w_I = w_{min} + \mu \otimes (w_{max} - w_{min}) \tag{3.6}$$

Here, μ is a random vector with values in the range $[0, 1]$, $w_{min} = 0$, and $w_{max} = 1$. This boils down to the inertia weight simply being a number randomly chosen from a uniform distribution in the range $[0, 1]$.

A gene regulatory network (GRN) has been represented using a directed graph in the form of an $N \times N$ adjacency matrix $G = [g_{i,j}]_{N \times N}$, where N is the number of genes in the network. The edge $g_{i,j}$ is 1 if there exists a regulatory effect from gene j to gene i, and is 0 otherwise. The authors have introduced sparsity constraints by allowing a gene to be regulated by at most 4 other genes. This reduces the search space complexity from exponential to combinatorial.

The BAPSO algorithm has been employed for every possible structure with at most 4 genes regulating another gene. For each structure, every particle only has non-zero weights allowed only for the positions in which $g_{i,j}$ of the GRN matrix of the structure has a value of 1. After training all the structures, the structure that has the least mean squared error for the global best position is taken as the final gene regulatory network. To account for the stochasticity of the approach, this entire procedure is repeated a number of times and the frequency with which each edge is predicted is recorded. The edges that are predicted more than a threshold number of times are taken to be present in the final gene regulatory network.

After thoroughly studying this paper, I got in touch with Mr. Abhinandan Khan, one of the authors of the paper, who shared an incomplete version of the code with me. I added the code for performing a min-max normalization on the input dataset, introducing the sparsity constraints, and accounting for the stochastic nature of the algorithm by thresholding the output over multiple runs. I used the E. coli SOS DNA repair network dataset containing gene expression levels for 8 genes over 50 time points to test the performance of the system. The dataset had several missing values, and Ms. Akanksha Farswan helped with accounting for those by employing matrix completion methods. The results that I obtained had a high number of false positives and failed to recognise many of the true regulatory effects. They also differed from execution to execution. (Further details are given in the Results section).

To check for the validity of the E. coli SOS DNA repair network dataset, I scanned several papers to determine the state-of-the-art implementation of GRN reconstruction, and came across an implementation using extreme learning machines [4]. The authors had provided a link to their web demo in their paper. The web demo allows the user to upload a time series gene expression dataset in CSV format and outputs a graph of the discovered GRN. The output graph did not contain a few true edges and had some false edges, but it was significantly closer to the true GRN. This approach, however, suffered from the inability to identify self-regulations. (Further details are given in the Results section).

I decided to try some new approaches by treating the task of gene regulatory network reconstruction as a regression problem of predicting the gene expression levels and drawing inferences from the weight matrix. In a naive attempt to do this, I coded a basic RNN containing a single $N \times N$ weight matrix initialised with random weights from a standard Gaussian distribution, where N is the number of genes in the network. For each time point t from 1 to $(T-1)$, where T is the total number of time points in the dataset, I took the vector of gene expression values at t as the input and multiplied it with the weight matrix to obtain an output vector of expression values for time point $(t+1)$. I determined the mean squared errors by comparing the obtained expression values with the true values in the dataset and updated the weight matrix by backpropagating the errors over a single layer for each time point. The goal was to see whether the final weight matrix obtained reflected the connections present in the true gene regulatory network.

I tried this approach on the E. coli SOS DNA repair network dataset, but I found no significant similarities between the weight matrix and the true connections. Moreover, the weights differed over different executions of the code. To determine the cause behind it, I calculated the Pearson correlation coefficient for the gene expression levels between every pair of genes in the network (results are shown in the Results section). It established that there existed high degree of correlations between several genes that had no edge between them in the true network, and low degree of correlations between those that did. Hence, a simple linear approach such as the one I attempted would not be able to capture the dynamics of the network.

After a discussion with Mr. Dhruva Sahrawat, I coded an LSTM model, again, treating the task of gene regulatory network reconstruction as a regression problem. I used the TensorFlow library for creating a model having three LSTM layers using ReLU activation, dropouts and batch normalisation, followed by two fully connected layers. However, the challenge with this approach was obtaining the appropriate weight matrix.

I identified a paper on learning graph dynamics using deep neural networks [5] during a meeting with Dr. Anubha Gupta and Ms. Akanksha Farswan to see if it could be of any use to our work. Upon studying it, I found that focus of the paper was on determining related subgraphs from larger graphs with high dimensional data, which I did not find to be useful for solving the problem at hand. I checked another paper on an RNN approach to model gene regulatory networks [6] on the suggestion of Ms. Akanksha Farswan. However, it did not provide sufficient information about the backpropagation through time based learning algorithm and hyperparameters to be able to replicate the results. The lack of sufficient information for replication was also the case with another paper that I found, which used an RNN approach with bat algorithm [7].

I had an in-depth discussion on my RNN and LSTM approaches with Mr. Anil Sharma, a PhD student who has worked on LSTMs before. We determined that the saturating values at later time points for the gene expression values could be a hindrance to the weight updates for my RNN implementation, since there are not enough variations for the model to backpropagate the errors. As for the LSTM implementation, we determined that the issue was in the inability of drawing inferences from the model that was learned, since the original input is significantly altered as it moves across different layers, and any weight matrix in the model would not capture end-to-end relations between the inputs and the final outputs. Other than these, we found one key issue in the use of machine learning models for the task. The datasets that we have possess gene expression levels for a few tens to at most a hundred time points. The number of features to be learned are of the order N^2, where N is the number of genes, which is comparable to or exceeds the number of time points for most datasets. The dataset is too small for the model to be able to train in the first place. This limits the ability to use a standard learning approach for the task of gene regulatory network reconstruction.

I identified another paper that used an RNN based approach with particle swarm optimisation [8], which was used as a reference in several other papers that proposed variations of RNN based approaches for gene regulatory network reconstruction, and I decided to code the approach from scratch. The authors explain the rationale behind using PSO as it can achieve a faster convergence than conventional backpropagation. It is similar to the first approach [1] that I had tried.

The RNN model used to calculate the gene expression level for the i^{th} gene at the time $t + \Delta t$ is given by the equation below.

$$e_i(t+\Delta t) = \frac{\Delta t}{\tau_i} \times f(\sum_{j=1}^{N} w_{ij} e_j(t) + \beta_i) + (1 - \frac{\Delta t}{\tau_i}) e_i(t) \tag{3.7}$$

Here, N is the number of genes in the network, w_{ij} denotes the regulatory effect of gene j on gene i, β_i is the bias term corresponding to gene i, τ_i is the time constant corresponding to gene i, Δt is the time interval between two consecutive time points in the dataset, and $e_i(t)$ is the gene expression level of gene i at time t. The first term on the RHS accounts for the regulatory effects of all genes on the i^{th} gene, and the second term accounts for the decay.

The training algorithm used is particle swarm optimization. As was the case in the first approach [1], for each particle, the position corresponds to a potential solution and is an $N(N+2)$ dimensional vector containing the $N \times N$ weight matrix, N length bias vector, and N length time constant vector. Likewise, the velocity is an $N(N+2)$ dimensional vector with each value corresponding to another value in the position vector. The velocities v_i and positions x_i for the i^{th} gene are updated as per the following equations.

$$v_i(t+1) = W_I \times v_i(t) + c_1 \times \eta_1 \times (p_i - x_i(t)) + c_2 \times \eta_2 \times (p_g - x_i(t)) \tag{3.8}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.9}$$

Here, W_I is the inertia weight, c_1 and c_2 are acceleration constants, η_1 and η_2 are uniform random functions in the range $[0, 1]$, p_i is the best position of particle i that has been attained so far, and p_g is the best position that has been attained by any particle so far.

The fitness function used is the mean squared error function, as given below.

$$Fit(x_i) = \frac{1}{TN} \sum_{t=1}^{T} \sum_{i=1}^{N} (e_i(t) - d_i(t))^2 \tag{3.10}$$

Here, T is the number of time points, N is the number of genes in the network, $e_i(t)$ is the expression level of gene i at time t calculated using equation 3.7, and $d_i(t)$ is the true expression level of gene i at time t.

The key difference in this approach from the first approach [1] is in the way the structures are evaluated. A population of particles is initialised with random values for the positions and velocities, where the weights, biases and time constants are initialised from the range $[w_{min}, w_{max}]$,

$[\beta_{min}, \beta_{max}]$ and $[\tau_{min}, \tau_{max}]$ respectively, and the velocity for each dimension is initialised with random values from the range $[-V_{max}, V_{max}]$. In each iteration, the fitness function is calculated for each particle using the equation 3.10. The best values of the positions are then updated if the new fitness is better (lesser in value) than the best fitness for the particle so far. Likewise, the global best position is updated if any of the new fitness values is better than the fitness of the global best position so far. This is followed by updating the velocities and positions for each particle based on the update rule given in equation 3.8 and equation 3.9. The iterations are repeated until the maximum number of iterations are reached. The weight matrix of the global best position is treated as the weights for the connections in the final gene regulatory network.

To determine the existence of edges in the network, a discrete binary version of the PSO algorithm, as explained in [9], has been employed in the above approach. For each particle, for each value in the weight matrix, there exists a corresponding matrix of binary values that denotes the existence of an edge. This matrix of binary variables forms the structure of the network. Equation 3.7 now only accounts for those weights that have the corresponding binary variable as 1. The update procedure for the true weights remains the same. However, the binary values x_{il} $(1 \leq l \leq N^2)$ corresponding to each of the weights in the position vector of the i^{th} particle are updated using the following probability based equation.

$$x_{il} = \begin{cases} 1, & \text{if } \eta_3 + \delta < f(v_i(t+1)) \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

Here, η_3 is a uniform random function in the range $[0, 1]$, δ is a parameter used to limit the number of edges that are selected, and f denotes the sigmoid function.

The structure corresponding to the global best position at the time of completion of training is taken as the final gene regulatory network, with a 1 at position (i, j) in the adjacency matrix indicating that gene j regulates gene i. To account for the stochastic nature of the algorithm, the program is executed a number of times and edges that occur with a frequency greater than a threshold are taken to be present in the final inferred gene regulatory network.

The authors have tested this approach by first using an artificial dataset consisting of 4 genes. I wrote the code for the synthesis of the dataset from the given values of the weights, biases and time constants using equation 3.7, the values of which are given in the Results section. This model implementation is able to identify 6 of the 8 regulations present in this artificial network correctly, with 2 false positives. However, this model did not give satisfactory results for the E.

Coli dataset. Upon further brainstorming with my advisor, I decided to change my approach to solving a regression problem using iteratively reweighted least square (IRLS) approach with L_1-norm for sparse recovery.

The objective now was to estimate the value of β_i for the following system for each gene.

$$X\beta_i = y_i \tag{3.12}$$

Here, X is a matrix in which the element in the t^{th} row and j^{th} column corresponds to the gene expression level of the j^{th} gene at the t^{th} time point in the dataset, y_i is a vector in which the element at the t^{th} position corresponds to the gene expression level of gene i at the $(t+1)^{th}$ time point in the dataset, and β_i is a vector in which the element at the j^{th} position corresponds to the regulatory effect of the j^{th} gene on the i^{th} gene.

The IRLS algorithm involves iteratively assigning weights to each cross-sectional data point such that the outliers are assigned small weights so as to minimise their influence in estimating the best fit model.

All weights are initialised to have an equal value of 1. The β coefficients of the generalised linear model for each iteration are obtained using the following equation.

$$\beta^{(n+1)} = (X^T W^{(n)} X)^{-1} X^T W^{(n)} y \tag{3.13}$$

Here, X is the independent variable matrix, y is the dependant variable vector, $W^{(n)}$ is a diagonal matrix containing the weights associated with each data point in the n^{th} iteration, and β^{n+1} is the beta coefficient vector.

After calculating the beta coefficients, the weight for each data point is updated using the following equation.

$$w_i^{(n+1)} = \frac{1}{max(\delta, abs(y_i - X_i\beta^{(n+1)}))} \tag{3.14}$$

Here, w_i, y_i and X_i are weight, dependent variable and independent variable vector corresponding to i^{th} data point in the data set, and δ is a small value, say 10^{-5}, which is used for

regularization in case the value of $y_i - \beta^{(n+1)} x_i$ becomes zero.

To ensure that no independent variable skews the results to its favour, I decided to standardise the dataset. The advantage of using a standardised dataset over a scaled or normalised dataset is that the standardised beta coefficients obtained after applying IRLS directly correspond to the weights of the connections present in the gene regulatory network.

Before executing the above algorithm, the dataset was analysed for multicollinearity using correlation coefficients and variance inflation factors. Genes exhibiting a very high degree of correlation with correlation coefficients of more than 0.95 and variance inflation factor of more than 10 were clustered together. Then, the weight matrix and beta coefficients were obtained for cross-sectional data corresponding to multiple time windows, including $t-1$, $t-2$, $t-3$ and average cross sectional data for previous two and previous three time periods.

The best fit curve identified using the information from the weight matrix and the standardised beta coefficients was used to determine the weights of the connections of the GRN. The beta coefficients having an absolute value greater than the mean of all beta coefficients for a gene were considered as connections present in the GRN.

Chapter 4

Results Obtained

The original structure of the DNA SOS repair network of E. coli consisting of 8 genes is shown in Figure 4.1. A directed edge denotes that the gene indicated at the tail of the edge regulates the gene indicated at the head of the edge.

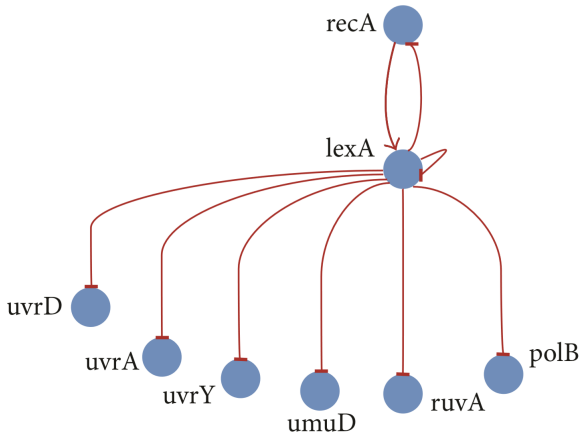

Figure 4.1: Original structure of DNA SOS repair network of E. coli; image source: [1]

Figure 4.2 shows the adjacency matrix of the inferred network using the approach given in [1] using the code that I completed after receiving an incomplete version of it from one of the authors. The dataset used is the E. Coli DNA SOS repair network containing gene expression levels of 8 genes for 50 time points at intervals of 6 minutes. A 1 at position (i, j) indicates that the presence of a directed edge has been inferred from gene j to gene i.

The results from this implementation showed a large number of false positives and failed to recognise many true edges present in the network.

	uvrD	lexA	umuDC	recA	uvrA	uvrY	ruvA	polB
uvrD	0	1	0	0	0	0	0	1
lexA	0	1	0	0	0	0	1	0
umuDC	0	0	0	0	0	0	1	1
recA	0	0	0	0	1	0	0	1
uvrA	0	0	1	0	0	0	1	0
uvrY	0	0	0	0	1	0	1	0
ruvA	1	1	0	0	0	1	0	0
polB	1	0	0	0	1	0	0	0

Figure 4.2: Results obtained using the RNN BAPSO approach given in [1]; 1 indicates the inference of the presence and 0 indicates the inference of the absence of a regulatory effect in the GRN; green indicates correct and red indicates incorrect inference

Figure 4.3 shows the structure of the GRN corresponding to the E. coli DNA SOS repair network inferred by the web demo of the state-of-the-art implementation of GRN reconstruction using an extreme learning machine approach given in [4].

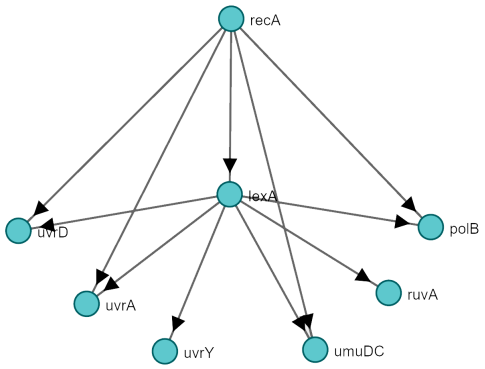

Figure 4.3: Inferred structure of DNA SOS repair network of E. coli via the web demo using the extreme learning machine approach given in [4]

The model is able to correctly infer 7 out of 9 interactions, and has 4 false positives. Each of the false positives corresponds to a forward edge in the directed graph. Despite the good results, this approach suffers from the inability to detect self-regulations.

Figure 4.4 shows the plots of the gene expression level versus the time for the 8 genes in the E. Coli DNA SOS repair network dataset.

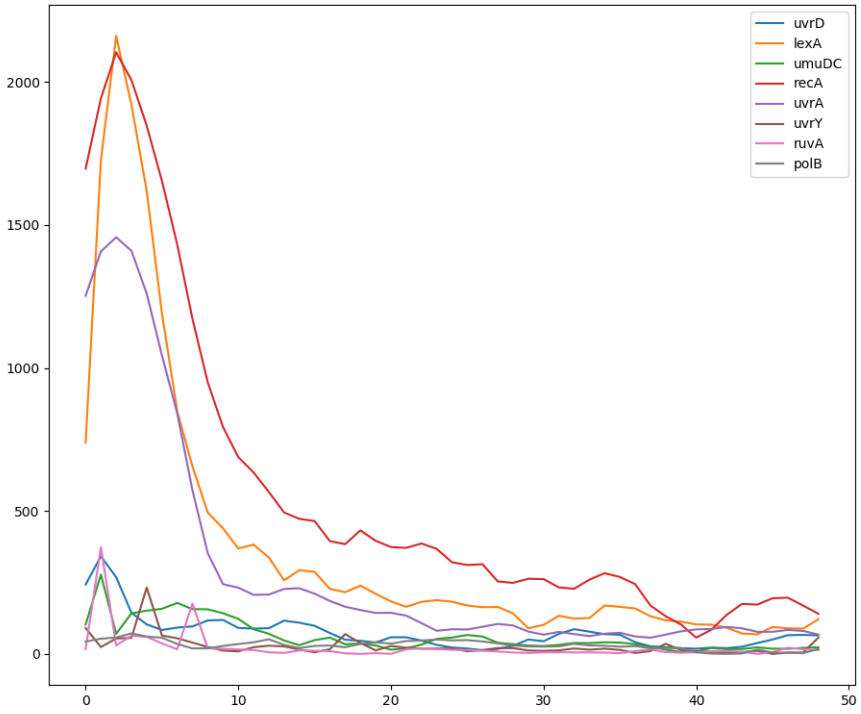

Figure 4.4: plots of the gene expression level versus the time for the 8 genes in the E. Coli DNA SOS repair network dataset

This shows the saturation of values at later time points, which limits the ability of the entire dataset to prove useful for predicting the relations between genes.

Figure 4.5 shows the Pearson correlation coefficients for the gene expression levels between every pair of genes in the E. Coli DNA SOS repair network dataset.

It can be seen that there exist high degrees of correlation between genes that do not have an edge between them in the original network, such as a correlation coefficient of 0.978 between umuDC and recA, which may be due to the indirect regulation of umuDC by recA via lexA. Likewise, there exist low degrees of correlation between genes that have an edge between them in the original network, such as a correlation coefficient of 0.551 between lexA and ruvA, which may be due to a weak regulatory action of lexA on ruvA. This shows why a basic linear model such as the naive RNN approach cannot capture the true regulations of the network.

	0	1	2	3	4	5	6	7
0	1	0.762	0.675	0.797	0.8	0.303	0.64	0.418
1	0.762	1	0.728	0.954	0.96	0.585	0.551	0.632
2	0.675	0.728	1	0.822	0.748	0.402	0.714	0.504
3	0.797	0.954	0.822	1	0.978	0.616	0.54	0.642
4	0.8	0.96	0.748	0.978	1	0.629	0.552	0.589
5	0.303	0.585	0.402	0.616	0.629	1	0.146	0.454
6	0.64	0.551	0.714	0.54	0.552	0.146	1	0.25
7	0.418	0.632	0.504	0.642	0.589	0.454	0.25	1

Figure 4.5: Pearson correlation coefficient matrix for the 8 genes in the E. coli DNA SOS repair network dataset; the genes are indexed as follows: (0, uvrD), (1, lexA), (2, umuDC), (3, recA), (4, uvrA), (5, uvrY), (6, ruvA), (7, polB)

Figure 4.6 shows the values of the weights, biases and time constants for generating the artificial dataset consisting of 4 genes over 50 time points using equation 3.7 with $\Delta t = 0.1$.

w_{ij}				β_i	τ_i
20.0	-20.0	0.0	0.0	0.0	10.0
15.0	-10.0	0.0	0.0	-5.0	5.0
0.0	-8.0	12.0	0.0	0.0	5.0
0.0	0.0	8.0	-12.0	0.0	5.0

Figure 4.6: Parameter values for generating the artificial dataset

Figure 4.7 shows the adjacency matrix of the inferred network using the approach given in [8] using the code that I implemented from scratch.

	Gene 1	Gene 2	Gene 3	Gene 4
Gene 1	1	1	0	0
Gene 2	0	1	0	0
Gene 3	1	0	1	1
Gene 4	0	0	1	1

Figure 4.7: Results obtained using the RNN PSO approach given in [8]; 1 indicates the inference of the presence and 0 indicates the inference of the absence of a regulatory effect in the GRN; green indicates correct and red indicates incorrect inference

The artificial dataset comprising of 4 genes and 30 time points generated using the values given in Figure 4.6 has been used for inferring the network. The code was executed 100 times. Edges that were inferred in over 50% of the executions of the code have been taken in the final inferred gene regulatory network. A 1 at position (i, j) indicates that the presence of a directed edge has been inferred from gene j to gene i. This method was able to correctly infer 6 of the 8 edges, showing 2 false positives.

IRLS with ten thousand iterations was used to infer the connections for the same dataset generated using the values given in Figure 4.6, and it was able to identify 6 out of 8 connections correctly with one false positive.

Figure 4.8 shows the adjacency matrix of the inferred network using the IRLS approach I implemented.

	Gene 1	Gene 2	Gene 3	Gene 4
Gene 1	1	1	1	0
Gene 2	0	1	0	0
Gene 3	0	0	1	0
Gene 4	0	0	1	1

Figure 4.8: Results obtained using the IRLS approach; 1 indicates the inference of the presence and 0 indicates the inference of the absence of a regulatory effect in the GRN; green indicates correct and red indicates incorrect inference

I then applied IRLS with decorrelation approach to in vivo E. coli DNA SOS repair network dataset consisting of 8 genes and 50 time points. lexA, recA and uvrA were found to have a high degree of multicollinearity in the E. coli DNA SOS repair network dataset, and were combined into a cluster.

The IRLS algorithm was executed with ten thousand iterations and the weight matrix and beta coefficients were obtained for various time windows. It was found that the best fit curve was obtained when the dataset corresponds to t-1 time window. More than fifteen percent of the predicted values were within one percent error of the actual value and more than one third of the predicted values were within ten percent error of the actual value for all except one gene.

In other words, the gene expression value of a gene at a time point was found to be dependent on the gene expression values of genes at the immediately previous time point, and the standardised β coefficients for this dataset were, therefore, used to determine the weights of the connections of the GRN. The β coefficients having an absolute value greater than the mean of all β coefficients for a gene were considered as connections present in the GRN. Using this approach, 7 out of 8 connections were correctly identified with 6 false positives. Figure 4.9 shows the results obtained.

	uvrD	umuDC	uvrY	ruvA	polB	{lexA, recA, uvrA}
uvrD	1	0	0	0	0	0
lexA	0	0	0	0	0	1
umuDC	0	1	0	1	0	1
recA	0	0	0	0	0	1
uvrA	0	0	0	0	0	1
uvrY	0	0	0	0	1	1
ruvA	0	0	0	0	0	1
polB	1	0	0	0	1	1

Figure 4.9: Results obtained using IRLS with decorrelation; 1 indicates the inference of the presence and 0 indicates the inference of the absence of a regulatory effect in the GRN; green indicates correct and red indicates incorrect inference

This is a significant improvement over the results obtained from the RNN PSO approach which did not make use of clustering.

The source codes for the artificial dataset generation and the RNN PSO approach coded with the implementation given in [8] are available here, and the source codes for the IRLS with decorrelation approach are available here.

Bibliography

[1] A. Khan, S. Mandal, R. K. Pal, G. Saha, "Construction of Gene Regulatory Networks Using Recurrent Neural Networks and Swarm Intelligence", Hindawi Publishing Corporation Scientica Volume 2016, Article ID 1060843

[2] H. Ling, S. Samarasinghe, D. Kulasiri, "Novel recurrent neural network for modelling biological networks: Oscillatory p53 interaction dynamics", Biosystems Volume 114, Issue 3, December 2013, Pages 191-205

[3] S. Uppu, A. Krishna, "Tuning Hyperparameters for Gene Interaction Models in Genome-Wide Association Studies", Published 2017 in ICONIP

[4] M. Rubiolo, D. H. Milone, G. Stegmayer, "Extreme learning machines for reverse engineering of gene regulatory networks from expression time series", Bioinformatics, Volume 34, Issue 7, 1 April 2018, Pages 1253-1260

[5] A. Narayan, P. H. O'N Roe, "Learning Graph Dynamics using Deep Neural Networks", IFAC-PapersOnLine Volume 51, Issue 2, 2018, Pages 433-438

[6] X. Hu, A. Maglia, D. C. Wunsch, "A General Recurrent Neural Network Approach to Model Genetic Regulatory Networks", Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, Shanghai, China, September 1-4, 2005

[7] S. Mandal, G. Saha, R. K. Pal, "Recurrent Neural Network Based Modeling of Gene Regulatory Network Using Bat Algorithm", Journal of Advances in Mathematics and Computer Science 23(5): 1-16, 2017; Article no. JAMCS.34916

[8] R. Xu, D. C. Wunsch II, R. L. Frank, "Inference of Genetic Regulatory Networks with Recurrent Neural Network Models Using Particle Swarm Optimization", IEEE/ACM Transactions on Computational Biology and Bioinformatics, Volume 4, No. 4, November 2007, Pages 681-692

[9] J. Kennedy, R. C. Eberhart, "A Discrete Binary Version of the Particle Swarm Algorithm", 1997 IEEE International Conference on Systems, Man, and Cybernetics, Volume 5, Pages 4104-4108

[10] S. Barbosa, B. Niebel, S. Wolf, K. Mauch, R. Takors, "A guide to gene regulatory network inference for obtaining predictive solutions: Underlying assumptions and fundamental biological and data constraints", BioSystems 174 (2018) 37-48

[11] F. M. Delgado, F. Gomez-Vela, "Computational methods for Gene Regulatory Networks reconstruction and analysis: A review", Artificial Intelligence in Medicine, Volume 95, April 2019, Pages 133-145