Subcellular Regulatory Networks Learning

Submitted in partial fulfilment of the requirements
of the degree of

Bachelor of Technology
in
Electrical Engineering

with

Master of Technology
in
Information and Communication Technology

By
Alok Singhal
2009EE50472

Supervised by
Prof. Sumeet Agrawal
Prof. Parag Singla

Department of Electrical Engineering
Indian Institute of Technology Delhi
June 2014

Dedicated to my loving parents

Certificate

This is to certify that the report entitled “Subcellular Regulatory Networks Learning”, submitted by Alok Singhal (Entry No. 2009EE50472) in partial fulfilment of the requirement of the award of Master of Technology in Information and Communication Technology as a part of the course EED852 at the Indian Institute of Technology, Delhi, is a bona fide work under my guidance and supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on basis of which a degree was conferred on an earlier occasion to this candidate.

Prof. Sumeet Agarwal

Department of Electrical Engineering

Indian Institute of Technology Delhi

Prof. Parag Singla

Department of Computer Sciences

Indian Institute of Technology Delhi

Declaration

I declare that this report explains the work carried out by me in the course EED852, Major Project, under the overall supervision of Prof. Sumeet Agarwal, Department of Electrical Engineering, IIT Delhi. The contents of the report including text, figures, tables, computer programs, etc. have not been reproduced from other sources such as books, journals, reports, manuals, websites, etc. Wherever limited reproduction from another source has been made, the source has been duly acknowledged at that point and also listed in the references.

Alok Singhal

2009EE50472

Acknowledgement

I, Alok Singhal, sincerely wish to express my deep sense of gratitude towards my project supervisors Prof. Sumeet Agarwal and Prof. Parag Singla, for their invaluable guidance and constant support and inspiration throughout the tenure of this work with utmost interest and rigor. Working with them on this project has been a tremendously knowledgeable and humbling experience for me.

I would like to thank the faculty of Electrical Engineering Department, IIT Delhi, for the invaluable education I have received from them. I would also like to thank my parents and friends and colleagues, for their constant support and encouragement.

Alok Singhal

2009EE50472

Abstract

This thesis explores different approaches to learn the structure of regulatory interactions between the genes at a subcellular level. A Markov Logic Networks [4] based framework is developed to learn these interactions based on their probability of occurrence in a world defined by First Order Logic. We develop models based on Markov Logic Networks for 2 different organisms and compare the results with already established methods.

A new approach to model regulatory interactions where we combine two already established methods namely Cmonkey [2] for clustering, and Inferelator [1] for structure learning into a single iterative algorithm to reduce the whole task of structure learning to a single algorithm. Results obtained using this method for different organisms are compared to the already established method, i.e., EGRIN [3].

In this project, we have applied a previously unused technique i.e. Markov Logic networks and shown it to work fairly well even with a very simple model. We have also introduced a new algorithm, which tries to build on the previously done work in the area. We have been able to improve the prediction accuracy for a part of the problem, where previous methods have not done so well.

Table of Contents

Chapter 1
Introduction

In this project, we try to address a biological problem using data driven approaches. The problem that we are trying to solve is to learn how genes within a cell interact with each other. There are some biological terms that would help a reader unfamiliar to appreciate this work. We can view cells as a bag containing many chemicals. Some of these chemicals are proteins, and RNA etc. Exact biological differentiation between these terms is of not that much concern to us. Genes are segments of RNA which play important roles in regulating various functions in the cell. These genes control the functions through initiation or inhibition of generation of proteins. Expression level of a gene can therefore be viewed as its level of activity at a particular point of time. Various interactions between the genes can be modeled as a network, where genes are the nodes and edges between them represent the type of interaction amongst them. In all we can view this network structure as an abstraction of our system's chemical dynamics. Gene/genome is an interchangeable representation for a single entity throughout this chapter, which forms the nodes for the network we are trying to learn. There are some special genes, which are already known through some biological experiments to be the regulators for certain processes. These are known as Transcriptional Factors (TFs). Now there are generally several genes which take part in a particular biological function being controlled by a single Transcriptional Factor (TF) or a group of TFs. So it makes sense if we could somehow group these genes together into a single unit to reduce the complexity of our problem. As we will see later, this is an integral part of some of the present algorithms and also inspires the algorithm that we have proposed in our work.

Environment plays an important role in deciding how different components of the genome of an organism are dynamically expressed. If we could understand how proteins interact with each other and the environment, then it should be possible for us to alter the biochemical processes within the body to our liking. Given the large amount of Genomes that have been completely sampled and given that how little we understand the interactions at this point, a gene by gene approach seems to be very inefficient in accomplishing this. Therefore we need data drive systems approach to solve this kind of problem. One of the major challenges in this setting is to find out a reduced set of factors from the large genomes, which can be used to predict system behavior.

This report begins with a review of EGRIN [3], a predictive model for cell physiology, proposed by Baliga et al. We see how Cmonkey Biclustering Algorithm [2] can be used to reduce the data complexity as well as cluster the genes that are co-regulated. Once we have got a reduced data set, we can use inferelator [1] to learn the regulatory interactions between genes themselves and environmental conditions as well. Mathematical background of both Cmonkey as well as Inferelator are introduced and explained. After that we introduce Markov Logic Networks [4], which are used to combine Markov networks with first-order logic for structure learning and inference. We develop an MLN to model the regulatory interactions, and compare its results with those of the inferelator. In the next chapter we introduce a new iterative algorithm which instead of sequentially clustering the data and then learning the network structure, combines it all into a single iterative task. This is very similar to the EM algorithm used in machine learning. We compare its results with those of the EGRIN/ inferelator and see how it performs with respect to the established methods. The last chapter provides the conclusion to the thesis listing some shortcomings of the present methods. We also provide with some avenues, where future research is possible in the field.

Chapter 2
Literature Review

This chapter gives a brief overview of the systems approach for learning subcellular regulatory networks. The chapter is divided into three subsections. In the first section we introduce the general model for EGRIN as well as some other common methods. In the second section, we discuss the Cmonkey biclustering algorithm. In the third section, we see in detail the math behind the Inferelator, the structure learning algorithm.

2.1 Learning Subcellular Regulatory Networks

One of the important reasons to apply systems approach to the biological processes is to understand if we make a small change in environmental conditions or genes, then how it would affect the system. Methods for learning regulatory interactions can be broadly classified into two groups. First group consists of methods which learn a network of unit less regulatory influences. Methods in the second group on the other hand learn regulatory networks as well as some dynamical parameters and use these parameters to predict expression levels in future experiments. First we would see some methods which use associations or relatedness between the levels of expression of transcription factors and their targets. Two of these methods are Context Likelihood Relatedness (CLR) [9] and Algorithms for Reconstruction of accurate Cellular Networks (ARACNE) [10].

1. **Context Likelihood Relatedness:** In the CLR algorithm, we calculate a score for each pair of genes based on the mutual information between them. Once it has calculated the mutual information between regulators and their target genes, it calculates how likely each mutual information value is, within the context of its network. The background distributions of mutual information scores are calculated. Now those interactions are said to be most probable, which have mutual information scores significantly above the background distribution. This method was used to infer E. coli transcriptional Regulatory network.
2. **ARACNE:** This algorithm is based on a data processing inequality which says that If gene X_i interacts with gene X_j through gene X_k then, mutual information between gene X_i and gene X_j is less than the minimum of mutual information between gene X_i and X_k and the mutual information between gene X_j and X_k. A weight is assigned to each pair, which is equal to the mutual information between them. Then all the edges, for which, the weight is less than a given threshold are removed. This results in weakest interactions among the triplets being removed in accordance to the property mentioned above. ARACNE was successfully used to infer transcriptional regulatory network for mammalian cells.

Now coming to the second group of methods, which learn dynamical parameters to predict expression levels for future experiments. The method which gives the best results and is used widely is EGRIN.

EGRIN: This method is a combination of several algorithms, which are applied sequentially to infer the regulatory network for Halobacterium Salinarum. Basic sequence of steps involved in learning the regulatory network can be summarized as:

1. Genome for the organism was sequenced and functions were assigned to the genes based on structural similarities.
2. Environmental conditions were perturbed and resulting changes in gene expression levels were measured using microarrays.
3. Once all the data was gathered, genes which are coregulated under certain conditions are identified using the Cmonkey algorithm and then grouped into biclusters to reduce data complexity.
4. After reducing the data complexity, machine learning algorithm inferelator was used to construct a dynamic network model to measure influence of EFs and TFs on expression levels of genes.

Using these steps, the regulatory model for H. Salinarum was constructed. Data was collected for individual as well as combined perturbations to various Environmental and Transcriptional Factors. Then the Cmonkey algorithm was used to classify the 1929 out of known 2400 genes into 200 biclusters, representing a set of genes which are potentially coregulated under certain environmental conditions. Once we have the biclusters, the regulatory network can be reconstructed using Inferelator. All of this results in a set of equations, which can take as input the changes in environmental and transcriptional factors and predict the expression level of genes of the organism with a high degree of correlation with their measured expression levels.

Two algorithms that are used in EGRIN are Cmonkey, for the biclustering of genes to reduce data complexity and Inferelator, to learn the regulatory network. Now we would take up both of these algorithms one by one and study them in detail.

2.2 Cmonkey Biclustering algorithm:

One of the major issues in learning biological networks is that the problem is severely underconstrained i.e. number of free parameters is much higher than the dimensionality of data. Common practice is to cluster the genes together based on their expression levels, before doing the network inference. It serves several purposes, such as it reduces the data complexity, and also if done properly, then signal to noise ratio in the data can be reduced through signal averaging. The obvious way is to cluster together the genes which have highly correlated expression levels, but there is a catch to that, since the expression levels might be correlated by chance also. Being correlated does not imply that the genes are also coregulated. On the other hand genes which are coregulated are very much likely to be functionally linked, i.e. genes which perform a certain task together are potentially coregulated.

One of the ways to measure the effectiveness of clustering is to check whether genes which are biologically linked through common tasks, metabolic pathways, cis-regulatory motifs etc. are grouped into the same cluster or not. Another way to use this information can be that these factors can be assigned weights and used as priors based on their quality and relevance.

Since any biological system is multi-functional and can be affected by several environmental factors, therefore genes might not be coregulated under all the experimental conditions.

Also a gene can be involved into several different biological processes, so it is important for a clustering method to be able to put genes into multiple clusters, so called biclustering.

The bicluster model: In Cmonkey biclustering model, each biclusters begins as a seed, and this bicluster is iteratively optimized by adding/removing genes. Each of the biclusters is modeled via a Markov chain process. We calculate conditional probability distributions based on the biclusters' previous state, and use them to update the cluster in the next step. In this way we can calculate probabilities that each of the genes belongs to a particular biclusters based upon the current state of the bicluster.

In short the algorithm can be summarized as follows (Reiss, Baliga, and Bonneau 2006)[2]:

Step 1: "Seed" a new bicluster

Step 2: Search for motifs in the bicluster

Step 3: Compute conditional probability that each gene/condition is a member of the bicluster

Step 4: Perform moves sampled from the conditional probability

Step 5: If the cluster changes go to **step 2**; else go to **step 1**

New biclusters are seeded until we have a prespecified number of clusters. Since biclusters are optimized one by one, to optimize the utilization of search space, a new bicluster is seeded only with a gene that has not been previously included into any of the biclusters. Once a bicluster has been seeded, it is iteratively improved with respect to the joint likelihood conditions. Genes/ conditions are added to the bicluster if they have a high probability of membership, and they are dropped if they have a low probability of membership. There are some additional mathematical constraints applied to limit the bicluster size and overlap among the biclusters. For example, to reduce the total number of biclusters n_i , into which a particular gene I may fall is modeled as a Poisson process. Then the probability of adding or dropping i from bicluster k is multiplied by the prior probability of observing the gene in so many biclusters. This constrains the solution to a very biologically intuitive model: Each gene is included in an average of 2 clusters. If we have already have the gene in two other biclusters, then the tendency to drop the gene from the bicluster is highly increased due to the constraint. In a similar manner another constraint is applied to limit the size of each bicluster. Result of all this is a biclustering algorithm which takes into account the biological factors to divide the genes into clusters, which are highly representative of their relatedness.

Now we would discuss the Inferelator, network inference algorithm used in EGRIN

2.3 Inferelator

Inferelator is an algorithm which uses standard regression along with L1 regularization to design predictive models for expression levels of genes as a function of levels of Transcription Factors, environmental factors and interactions among these factors. One of the major advantages is that it can be used to model steady state as well as kinetic expression levels.

Model Formulation: Expression level of a gene/group of genes y is assumed to be influenced by N other factors: X = (X_1, X_2, …….., X_n). In the model Transcriptional factor levels and external conditions are used as predictors and gene/bicluster levels are used as the response. The relation between X and y can be modeled by the following kinetic equation:

$$\tau \frac{dy}{dt} = -y + g(\beta.\boldsymbol{Z}) \qquad (1)$$

Here, Z is a set of functions of the regulatory factors X. The coefficient β_i models the effect of the ith regulator on the expression level of y, with positive values of β signifying an inductive effect and negative values signifying an inhibitory effect. τ is the time constant of the level of y when there are no external influences present. We can use various functional forms for the function g. For the inferelator a truncated linear form has been employed:

$$g(\beta Z) = \begin{cases} \beta Z: & if\ \min(y) < \beta Z < \max(y) \\ \max(y): & if\ \beta Z > \max(y) \\ \min(y): & if\ \beta Z < \min(y) \end{cases}$$

The experimental conditions for the data can belong to either a steady state experiment or a time series experiment. By suitable manipulation of the equation (1), we can combine both type of data to fit into parameters β and T.

In the steady state, $\frac{dy}{dt} = 0$ and equation 1 reduces to:

$$y = g(\beta . Z_{ss}) \qquad (4)$$

Where Z_{ss} is the value of Z measured in the steady state. For time series data, taken at time (t_1, t_2,…………., t_n), we can approximate the equation (1) as follows:

$$\mathrm{T}\frac{y_{m+1}-y_m}{\Delta t_m} + y_m = g(\textstyle\sum_{j=1}^{p} \beta_j Z_{mj)}\,) \qquad \text{For m = 1, 2,………n-1 (5)}$$

By comparison of equations (4) and (5), it can be seen that right hand sides for both of them are identical, hence we can simultaneously fit the model using both time-series as well as equilibrium data. Design matrix Z also has an important role to play. It can be used to encode interactions among predictor variables. We can approximate the interactions between Transcription Factors by allowing functions in Z to be either identity functions of a single variable or the minimum of two variables. For example, the inner product of Design matrix and linear coefficients for two predictors which are participating together will be:

$$\beta Z = \beta_1 x_1 + \beta_2 x_2 + \beta_3 \min(x_1, x_2)$$

Using this encoding, we can model interactions like OR, AND and XOR etc. For example, if we have an interaction, where x_1 and x_2 activate y, only when they are expressed together, we would expect the inferelator to learn the weights β_1=0, β_2 =0, β_3=1

Once we have formulated our model, our next step would be learn the parameters. There are several methods which can be used to estimate the parameters for the model formulation. For example, using all the predictors to estimate the values of variables would amount to multivariate least squares regression. The model used here is the L1 shrinkage for predictor selection, which involves the following minimization:

$$(\dot{\alpha}, \dot{\beta}) = \arg min_{\alpha,\beta} \left\{ \sum_{i=1}^{n} \left(\gamma_i - \alpha - \sum_{j=1}^{p} \beta_j Z_{ij} \right) \hat{}2 \right\}$$

Subject to the following additional constraint:

$$\sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} |\beta_{\boldsymbol{j}}| \leq \boldsymbol{t} |\beta_{\boldsymbol{ols}}|$$

Where $\beta_{\boldsymbol{ols}}$ is the Ordinary least Squares estimate of β. The shrinkage parameter t can range from 0 to 1, where t= 0 amounts to the selection of null model, whereas t = 1 gives ordinary least squares estimate. Optimal value of t is estimated by minimizing the prediction error using tenfold Cross Validation.

In short the algorithm can be summarized as [1]:

```
“For each (bicluster k) {
    For each (TF or environmental factor i) {
        Update list of best single influences
        For each (TF j) {
    Update list of best interactions list (min [i, j])}
    }
    Select from predictors and estimate model parameters with
    L1-shrinkage/CV
Store model for gene/bicluster k}
Combine models for individual biclusters into global network”
```

Predictive accuracy: Measured values are compared to the predicted values using Root Mean Square Deviation:

$$RMSD(Y, \widetilde{Y}) = \frac{\sqrt{\sum_{n=1}^{N}\left(Y_n - \widetilde{Y}_n\right)^2}}{N}$$

Where Y is the true response and $\check{Y}$ is the predicted response over N conditions. One of the advantages of the measure is that it does not emphasize the low variance segments of the signal.

Chapter 3
Markov Logic Networks

In the first part of our project we try to learn Subcellular Regulatory Networks using Markov Logic Networks which combine Markov Networks with First-Order Logic for learning. The MLNs were introduced by Richardson and Domingos in [4] in 2005. The advantage of MLNs is that it allows us to model systems using first-order clauses.

3.1 Basics of Markov Logic Networks

A *first-order knowledge base (KB)* is a set of sentences or formulas in first-order logic (Genesereth & Nilsson, 1987). A world defined by first-order logic has hard constraints imposed upon it. If even a single formula within the knowledge base is violated, then that world has a zero probability of existence. But in the world defined by Markov Logic Networks, violation of a rule makes a particular world less probable, not entirely impossible. The strength of a formula is defined by the weight associated with it. Higher is the weight of a formula, more is the penalty for its violation. A world that violates less number of formulas has a high probability of existence. Definition of Markov Logic Networks as proposed by Richardson and Domingos is:

Definition 1: A Markov Logic Network L is a set of pairs (F_i, w_i), Where F_i is a formula in First-order Logic and w_i is a real number. Together with a finite set of constants C = {c_1, c_2, ……, $c_{|C|}$}, it defines a Markov Network $M_{L,C}$ as follows:

1. *$M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if ground atom is true, and 0 otherwise.*
2. *$M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true and 0 otherwise. The weight of the feature is w_i associated with F_i in L.*

Table 1: Example of a First-Order Knowledge Base and MLN, (Richardson, Domingos 2005)[4]

English	First-Order Logic	Clausal Form	Weight
Friends of friends are friends	$\forall x \forall y \forall z\ Fr(x,y) \wedge Fr(y,z) \Rightarrow Fr(x,z)$	$\neg Fr(x,y) \vee \neg Fr(y,z) \vee Fr(x,z)$	0.7
Friendless people drink	$\forall x \left(\neg\left(\exists y\ Fr(x,y)\right) \Rightarrow Dr(x)\right)$	$Fr\left(x, g(x)\right) \vee Dr(x)$	2.3
Drinking causes cirrhosis	$\forall x\ Dr(x) \Rightarrow\ Ci(x)$	$\neg Dr(x) \vee Ci(x)$	1.5
If two people are friends, either both drink or neither does	$\forall x \forall y\ Fr(x,y) \Rightarrow (Dr(x) \leftrightarrow Dr(y)$	$\neg Fr(x,y) \vee Dr(x) \vee \neg Dr(y)$, $\neg Fr(x,y) \vee\ \neg Dr(x) \vee Dr(y)$	1.1 1.1

From Definition 1, we can write the probability of a particular world x, specified by ground Markov Network $M_{L,C}$ as:

$$P\ (X = x) = \frac{1}{Z}\ exp\left(\sum_i w_i n_i(x)\right) = \frac{1}{Z}\prod_i \Phi_i\left(x_{\{i\}}\right)^{n_i(x)}$$

Where $n_i(x)$ is the true number of groundings of F$_i$ in x, $x_{\{i\}}$ is the truth value of atoms appearing in F$_i$ and $\Phi_i\left(x_{\{i\}}\right) \; = e^{w_i}$.

We can also draw the graphical structure of $M_{L,C}$ using Definition 1. Definition 1 implies that there will be an edge between two nodes of the graphical network $M_{L,C}$ if and only if the corresponding ground atoms appear together in at least one grounding of one formula in L.

Once we have got the Ground Markov Network, and some data from the real world observation, we can use the data to learn the weights for the formulas in the network, i.e. strength of each formula.

3.2 Learning Subcellular Regulatory Network using MLN

We design a Markov Logic Network to learn the regulatory interactions for H. Salinarum. The organism was selected because Results obtained by running Cmonkey followed by inferelator on the organism are commonly available and can be easily replicated. This facilitates comparison of performance with the established method, i.e. the Inferelator. Dataset for H Salinarum consists of expression levels for 2400 genes across 280 conditions, out of which around 100 are known Transcription Factors. Since MLN works only for discrete data, we binaries our expression data. If expression level of a particular gene under a particular condition is higher than the reference level, then the gene is said to be expressed, otherwise its not expressed. This representation of genes is kind of like representing them as switches with two states, on and off. A gene is on if its producing proteins at a higher level than the reference level for that gene and vice versa. We can define the first-order knowledge base for the problem as:

1. If the Transcriptional Factor(TF) t is expressed then the gene g is also expressed

$$ExpressesTF(+t, c) \Rightarrow Expresses\ (+g, c) \qquad (1)$$

2. If the Transcriptional Factor(TF) t is not expressed then the gene g is expressed

$$!\,ExpressesTF(+t, c) \Rightarrow Expresses\ (+g, c) \qquad (2)$$

3. Prior probability of expression of a particular gene g

$$!\,Expresses(+g, c) \qquad (3)$$

Here rule 1 represents the inductive interaction between the TFs and the genes. + sign in the rule before t and g implies that we learn a separate weight for each pair of gene and TF. Since there is no plus sign in front of conditions c, it implies that rules to be learned are across the conditions. So if we have 2400 genes, 100 TFs across 300 conditions, number of ground clauses for this MLN is of the order of 10^8. Current implementations of MLN do not have the capacity to process so many clauses. To reduce the data complexity and also to make the starting point for learning same as that for the inferelator, we use Clusters produced by the Cmonkey biclustering algorithm. Hence instead of using the expression levels for individual genes, we replace them with the expression levels for cluster centers and learn the weights for interaction between clusters and TFs. Since we have only 200 clusters, it reduces our data complexity as well.

Once we have got our database and defined the rules for MLN, we need to learn the weights for the rules. We can use either generative learning or discriminative learning to achieve that. Since in this case we know apriori which of the atoms are evidence, i.e. expression levels for TFs and query atoms, i.e. expression levels for clusters, we can use discriminative learning here to learn the weights for regulatory interactions. Our goal here is to predict expression for clusters given expression for TFs. If we partition the ground atoms into two different sets, X for the evidence atoms and Y for the query atoms, the conditional likelihood of Y given X is:

$$P(y|x) = \left(\frac{1}{Z_x}\right) exp\left(\sum\nolimits_{j \in G_Y} w_j g_j(x,y)\right)$$

Where F_Y is the set of all MLN clauses with atleast one grounding involving a query atom, G_Y is the set of Ground clauses in $M_{L,C}$ involving query atoms, and $g_j(x,y)$ = 1 if the jth ground clause is true in the data and 0 otherwise. We can calculate the gradient of CLL by approximating with MAP, using MAP inference to find most probable state of query set Y, given the evidence set X.

In our case we have used discriminative learning to learn the regulatory network across 250 conditions and left out 28 conditions as the test set for effectiveness of predictions. Once we have learned the weights for our MLN, we can infer the state of system under new conditions using the weights and values of evidence atoms under new conditions.

We tested our Model for two different organisms H. Salinarum and E Coli. One of the ways to measure the model's effectiveness is to see whether it is able to capture the links suggested by inferelator or not. If the weights for links which have non-zero βs in inferelator have significantly higher weights than other links, it implies that the model is effective in finding the links.

H. Salinarum	Inferelator	MLN
Average weight for +ve non-zero links(inferelator)	0.35	0.69
Average weight for –ve non-zero links(inferelator)	-0.21	-0.43
Average weight for +ve zero links	0	0.37
Average weight for –ve zero links	0	-0.3

Table 2: Comparison of link weights for MLN with link weights for inferelator for H. Salinarum

Table 3: comparison of link weights for MLN with link weights for inferelator for E Coli.

E.Coli	Inferelator	MLN
Average weight for +ve non-zero links(inferelator)	0.32	0.52
Average weight for –ve non-zero links(inferelator)	-0.23	-0.36
Average weight for +ve zero links	0	0.35
Average weight for –ve zero links	0	-0.31

As can be seen from the tables above, there is a high correlation between links detected by inferelator and those detected by MLN. Those links which have a non-zero value for inferelator have a significantly higher value for MLN also. Whereas links which have zero weight for inferelator have a lower value in MLN also since they are less important than their counterparts. Another measure for performance of model will be its accuracy of prediction under novel test conditions. As mentioned earlier, around 10% of conditions were left out of the set used for learning weights. Prediction accuracy for these conditions along with precision-recall curve are:

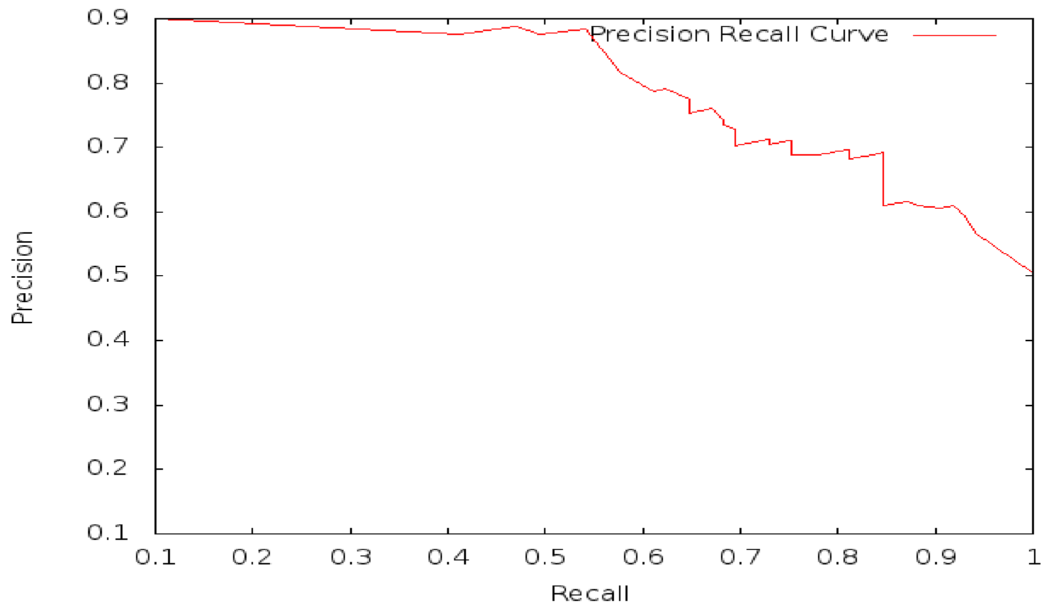

Figure 2: Precision-recall curve for H. Salinarum

Area under the precision-recall curve has a value of 0.84 units. Prediction accuracy for H. Salinarum using MAP inference is 73.6%.

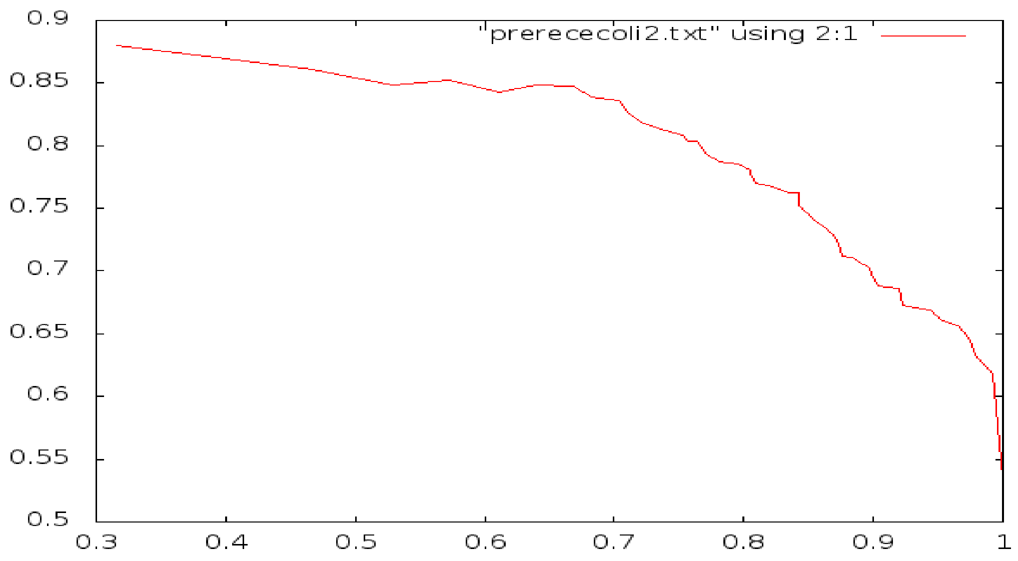

Figure 3: Precision-recall curve for E coli.

Area under the precision-recall curve has a value of 0.81 units. Prediction accuracy for H. Salinarum using MAP inference is 74.4%.

As can be seen from the PR curves above, the model can predict expression levels for genes with appreciable accuracy, even though we have used only basic rules for regulatory interaction, where we have taken into account only the direct interactions.

All of the inference and learning algorithms used in the previous sections are publicly available in the open-source Alchemy system. We have used Alchemy for running all our simulations and tests for this part.

3.3 L1 regularized MLN

Similar to the Inferelator, we can also have L_1 regularization for the MLNs. This helps in reducing the number of non-zero weights since L1 regularization forces many parameters zero due to its tendency to strongly penalize small terms. This helps us to prevent over fitting and weeds out the TFs that are not relevant to the expression levels of the gene/cluster under consideration.

Since we already have a procedure to calculate the conditional log likelihood as well as it's gradient, discriminative learning can be used to learn the weights for the model. However to avoid overfitting and also to select only the best predictors, we follow the model implemented by (Huynh et al., ICML, July 2008)[8]. They introduce a Laplacian prior with zero mean,

$$P(w_i) = \left(\beta/_2\right) .\exp(-\beta|w_i|)$$

On each weight and then optimize the posterior Conditional Log-Likelihood instead of Conditional Log-Likelihood. The final objective Function becomes:

$$\log P(Y|X)P\ (w) = CLL - \beta \sum_i |w_i| + constant$$

As we can see now there is an extra term introduced due to the Laplacian prior, which penalizes every non-zero weight w_i by $|w_i|$. So β is the parameter which decides the scale of penalty for each weight. The larger the β, more is the penalty for a non-zero weight.

We used the same MLN model that we have defined earlier for the ordinary MLN, and ran it on an implementation, which supports L_1 regularization. We had to use a Trial and error approach to optimize the value of the parameter to get the best prediction accuracy and also reduce the number of predictors which have non-zero weights. Results for both the datasets H. Salinarum and E Coli are as follows:

	H Salinarum	E Coli
Prediction Accuracy for Novel conditions	71.4%	72.9%
Average number of predictors with non-zero weights	10	12
Average weight for the non-zero predictors	0.3	0.23

Table 4: Results for L1 Regularized MLN

As we can see from the table above, prediction accuracy similar to regular MLN can be achieved using L_1 regularized MLN in this case. Average number of predictors for a single gene/cluster in H Salinarum has reduced from 80 to 10, which is a closer representation of biological networks, since a particular cluster is regulated only by a handful of predictors instead of all of them. Another thing to note here is that out of the 7 predictors which are detected by the inferelator for a single gene, on an average 5 are also detected by the L_1-MLN. This shows that the L_1-MLN is able to detect the common predictors efficiently and also potentially detects some new predictors.

In this chapter we tried to apply the MLN model to Network Learning problem for the first time. As can be seen from the results of even a very simple model, it is an effective approach to tackle this problem. In the next chapter we try to develop an iterative algorithm to improve the accuracy of inferelator.

Chapter 4

Network Learning-an iterative approach

In this chapter, we introduce a new algorithm which combines clustering and parameter learning for the network into a single module. The idea here is to perform both the tasks alternatively to improve the clusters as well as to get a better prediction accuracy. Motivation behind this approach is that clustering of genes and learning the interactions between them is an interlinked task rather than being two independent tasks. Genes which are regulated by same transcription factors (TFs) should also be clustered together, since most of the times they are linked to the same biological task. So instead of approaching them as two different problems, we try to combine them into a single one where learning in one part helps improve the second part and vice-versa. We would have ideally liked to implement this as a single MLN, which learns the clusters as well as the network simultaneously. However, due to scaling issues with MLN, we decided to use Inferelator for this. It provides us with a reference point, where we can compare our results with EGRIN, which uses Cmonkey as its clustering algorithm and Inferelator as its network learning tool.

For this purpose we use fuzzy c-means algorithm to perform a soft-clustering of the genes. Once we have got the cluster centers, we then run Inferelator to learn the network. Once we have got the network parameters, we now cluster the genes based on the predictors assigned to them by the Inferelator and get the new cluster centers, since the genes being regulated by same set of

TFs should ideally be in the same clusters. We repeat the process until cluster centers stop changing. In the essence, the algorithm is similar to the EM algorithm, where clustering of the genes can be seen as the E step, and the learning of the network parameters can be viewed as the M step. A detailed description of the model is given below.

4.1 Model and formulation.

Let total number of genes in the system be N. Number of conditions for which we have the data available is M. Let K be the number of clusters. e_{ij} denotes the expression level of the gene g_i under condition Q_j.

<u>Step 1:</u> Use fuzzy c-means clustering to assign the genes into clusters based on their expression levels. For a given gene g_i, its membership to a cluster j is calculated as follows:

$$w_{ij} = \frac{1}{\sum_{k=1}^{K}\left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|}\right) \wedge \frac{2}{m-1}}$$

Where m is the fuzziness coefficient. We can calculate the center vector c_j as follows:

$$c_j = \frac{\sum_{i=1}^{N} w_{ij}^{m} e_i}{\sum_{i=1}^{N} w_{ij}^{m}}$$

Where w_{ij} is the degree of membership calculated in the previous iteration.

<u>Step 2:</u> Once we have got the cluster centers, we learn the network parameters, i.e. the **β** matrix. It will be a $K * T$ matrix where K is the number of clusters and T is the number of TFs. Network parameters for this step are learned using Inferelator.

Step 3: Now we have the membership matrix **W**, which is an $N * K$ matrix and the parameter matrix **β**, which is a $K * T$ matrix. These two can be multiplied together to get the interaction matrix **S** of size $N * T$. It can be seen that element s_{ij} represents the weight for how predictor j influences gene I through this equation

$$s_{ij} = \sum_{k=1}^{K} w_{ik} \boldsymbol{\beta}_{kj}$$

Degree of membership w_{ik} for each gene i in the cluster k is multiplied by the weight associated with TF j for cluster k and then all the products are summed up to yield s_{ij}.

Step 4: Now this interaction matrix **S** is used to cluster the genes again into K clusters using fuzzy C-means clustering. This gives us an updated membership matrix **W** and we can use the updated cluster centers to learn the new updated parameter matrix **β** as well. We repeat the process till membership matrix **W** stops changing, which signifies that the movement of genes between clusters has stopped and convergence has been reached.

4.2 Experiments and results:

We ran our algorithm for the two organisms, H. Salinarum and E. Coli. Measure for effectiveness of the model is kept the same as that used by the inferelator, i.e. RMSD. If Y is the actual value and $\check{Y}$ is the predicted value, then:

$$RMSD(Y, \widetilde{Y}) = \frac{\sqrt{\sum_{n=1}^{N} \left(Y_n - \tilde{Y}_n\right)^2}}{N}$$

In case of H Salinarum, apart from the cluster centers, there are around 160 genes out of a total of 2400, which are not included in any of the clusters due to their inability to fit into any of the clusters. Results for these genes are mentioned separately, as they are particularly hard to model because of their erratic behavior.

	EGRIN	Iterative learner
RMSD for clusters	0.37	0.43
RMSD for unclustered genes	0.67	0.61
RMSD for individual genes	0.44	0.47

Table 5: Performance comparison between EGRIN and iterative learner for H Salinarum

	EGRIN	Iterative Learner
RMSD for clusters	0.41	0.46
RMSD for genes	0.46	0.48

Table 6: Performance comparison between EGRIN and iterative learner for E Coli.

As can be seen from the table above, the Iterative learner performs comparably well with respect to the EGRIN in case of the genes which could not be assigned to any of the clusters by the Cmonkey algorithm and are handled separately by the EGRIN. Performance over the genes which can be included into the clusters is slightly worse than the EGRIN.

One of the reasons for this behavior might be attributed to the difference in the clustering algorithms used for both of the methods. In case of EGRIN, Cmonkey biclustering algorithm is particularly strict about not assigning a particular gene to more than two clusters. While it more closely resembles the biological model, since it is very unlikely that a particular gene would be included in more than 2 biological functions. Our algorithm does not put any such constraints and as a result, a particular gene might be assigned to any number of clusters. It might be the case that the genes which are hard to classify maybe involved in several biological functions. It can indeed be seen from the final clusters given by our algorithm, that these genes belong to significantly more clusters(4.6) on an average than the genes which are easy to classify(3.1).

In this chapter, we have introduced a new algorithm, which tries to improve the efficiency of network learning by combining its two separate components into an iterative process. Although the algorithm performs slightly worse in the case of genes which are easily classified in the clusters, but for a section of genes whose behavior is harder to predict, it performs better than the established method.

Chapter 5

Conclusion

This project considered the problem of learning the subcellular regulatory networks and the different techniques that have been applied in its solution. Different approaches to the learning of regulatory networks were explored and developed.

We developed a Markov Logic Networks model to solve the problem, which is a novel approach and has not been explored before in great detail. We were able to develop a model which can predict the expression levels under novel conditions with fair amount of accuracy, although we modeled only direct interactions here, due to scaling issues with current implementations of MLN. We also used an L1 regularized variant of MLNs to reduce the number of predictors for the genes. It helps in improving the performance of the system. It also leads better identification of regulators for a particular cluster, since most of the regulators are assigned zero weights due to the Laplacian prior introduced in the formulation. It helps us to weed out the unrelated regulators, since they have a high probability of being assigned zero weights.

In a different approach, we sought to combine the two steps in network learning, i.e. clustering and parameter learning into a single iterative algorithm, since we view them as a single interrelated problem rather than two independent ones. Since we are trying to use the information provided by the clustering step to improve the results for network learning and vice versa, it should be able to improve on the established methods. Results for the algorithm are encouraging, as it is able to improve prediction accuracy for a subpart of the problem over already established methods like EGRIN. Although it still requires some work to make it perform better than the established methods in all the cases.

There are several extensions that can emerge from this project. One of them could be to use MLNs in the iterative learner in the M step instead of using the inferelator. We can also seek to develop an MLN model which combines the two tasks into a single model. A similar problem in a different domain has been studied by Domingos and Singla in [6]. MLN for learning clusters can be modeled using an approach similar to theirs. Future work can also focus on modelling more complex interactions through MLNs, like combination of two TFs regulating a gene.

Another direction to work upon would be to use a different algorithm than the fuzzy c-means clustering in the iterative learner. Modeling the clusters as a mixture of Gaussians i.e. GMM may help in improving the performance of the algorithm, as Fuzzy C-means assigns genes to significantly higher number of clusters than their biological functions. Modelling the System as GMMs might help to alleviate the problem, since membership to a particular cluster falls off exponentially in this case with the distance, which will reduce the probability of a gene being a member of many clusters very small.

.

References

1. Bonneau R, Reiss DJ, Shannon P, Facciotti M, Hood L, Baliga NS, Thorsson V. “The Inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo”**.** Genome Biol. 2006;7(5):R36

2. David J Reiss, Nitin Baliga, Richard Bonneau, “Integrated biclustering of heterogeneous genome-wide datasets for the inference of global regulatory networks” in BMC bioinformatics 2006, published 2nd June 2006

3. Richard Bonneau, Marc Facciotti, David J Reiss, Amy K Schmid, Min Pan, Amardeep Kaur et al.,”A predictive model for transcriptional control of physiology in a free living cell”, Cell 131. 1354-1365

4. M. Richardson and P. Domingos. “Markov logic networks. Machine Learning”, 62:107–136, 2006

5. S. Kok and P. Domingos. “Learning the structure of Markov logic networks. In Proceedings of the Twenty-Second International Conference on Machine Learning”, pages 441–448, Bonn, Germany, 2005. ACM Press

6. P. Singla and P. Domingos. “Entity resolution with Markov logic”. In Proceedings of the Sixth IEEE International Conference on Data Mining, pages 572–582, Hong Kong, 2006. IEEE Computer Society Press.

7. Kok, S., Singla, P., Richardson, M., & Domingos, P. (2005). *The Alchemy system for statistical relational AI* (Technical Report). Department of Computer Science and Engineering, University of Washington. http://www.cs.washington.edu/ai/alchemy.

8. Hunyh, Tuyen N., Mooney, R., ”Discriminative structure and parameter learning for Markov Logic Networks”. In Proceedings of the 25th International Conference on Machine Learning *(ICML),*. Helsinki, Finland, July 2008.

9. J.J. Faith, B. Hayete, J.T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J.J. Collins, and T.S. Gardner. Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. PLoS Biology, 5, 2007

10. A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics, 7, 2006.

11. CMONKEY web site [http://halo.systemsbiology.net/cmonkey]

12. The R project for statistical computing [http://www.rproject.org]