# Subcellular Regulatory Networks Learning

*A thesis submitted in partial fulfillment of the requirements for the degree of*

DUAL DEGREE

*in*

Computer Science & Engineering

*by*

**ASHESH**

**Entry No. 2010CS50276**

*Under the guidance of*

**Prof. Parag Singla**

**Prof. Sumeet Agarwal**

Department of Computer Science and Engineering,
Indian Institute of Technology Delhi.
May 2015.

# Certificate

This is to certify that the thesis titled **Subcellular Regulatory Networks Learning** being submitted by **ASHESH** for the award of **Dual Degree** in **Computer Science & Engineering** is a record of bona fide work carried out by him under my guidance and supervision at the **Department of Computer Science & Engineering**. The work presented in this thesis has not been submitted elsewhere either in part or full, for the award of any other degree or diploma.

**Prof. Parag Singla**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Delhi**

**Prof. Sumeet Agarwal**
**Department of Electrical Engineering**
**Indian Institute of Technology, Delhi**

# Abstract

This thesis aims to learn regulatory interactions between genes and Transcription factors and Environmental factors using Markov Logic Networks.

In this project, we develop a joint model which would predict the similarity of genes and the regulatory network jointly. Intuitively we use the bijection between similarity of genes and regulatory links: similarity of genes implies similar regulatory links and vice versa.

At the later stage of the project, artificial regulatory network was created from which data was generated and detailed experiments were done to understand the problem at hand.

# Acknowledgments

This project has been a tremendous source of wisdom for me. As a project it made me aware of the joys and challenges of research. It made me realize what it takes for a person to thrive in research. I would like to express gratitude for my supervisors of this project Prof. Parag Singla and Prof. Sumeet Agarwal, for their precious time, guidance and support throughout this project.
I would like to thank my computer science department,my parents and friends for their support.

**ASHESH**
**2010CS50276**

# Contents

# List of Figures

# Chapter 1

# An Introduction

In this project, we try to address a biological problem using data driven approaches. For the purpose of making the reader understand, we will explain very lightly some of the biological terms which are relevant here. Inside the cells we have chemical entities called proteins which are vital for the proper functioning of the cell. Concentration of proteins in the cell varies over time. Proteins formation process is triggered in another important chemical present inside cells called genes. Biologically speaking, genes are 'expressed' to form proteins. Expression of gene is regulated by yet another class of compounds present in cells known as Transcription Factors or in short TFs. Presence/Absence of some TFs induces gene expression where as presence/absence of some TFs inhibits gene expression. This project deals with inferring the regulatory network of genes and TFs from the data.
There are some genes which are regulated by same set of TFs. Generally these are those genes which perform similar functions in the cell. Hence it makes sense if we try to club genes into clusters and infer links of those clusters. Clustering the genes will serve two purposes. On one the one hand it will reduce the complexity of the problem as we will have less number of nodes in our network. The other advantage is that, the biological data is noisy. This clubbing together will help to cancel out the noise from the data.

If we are able to predict how genes are regulated in the body then it would be possible to tweak the regulatory network so as to get the desired changes in the biochemical processes.

Genes are said to be corregulated if they are acted upon by same set of transcription factors in a similar way. Note that if genes are corregulated then they will be have identical links. Moreover if two genes have identical links then they will be be corregulated. Hence learning the regulatory network could in principle be enhanced by feeding in the similarity information and learning the similarity could be enhanced by feeding in the regulatory network knowledge. Hence we propose a joint model which learns the regulatory network with the similarity of genes.

In chapter 2, we do a literature review of some of the approaches taken for solving this problem. We also look at the canopy soft clustering technique. We use it to feed in partial information about the similarity of genes in our joint model. In chapter 3, we introduce Markov Logic Networks and briefly describe the workings of it. We also describe a previous MTP work done in this institute in which a simple MLN network had been developed and it does regulatory network learning from it.In chapter 4, we describe our different formulations for the joint model for capturing gene similarity and regulatory network learning. In chapter 5 we describe the artificial regulatory network generation and data generation framework. Later in this chapter we describe various experiments and their results. In chapter 6, we summarize the work done in this MTP and give a overview of all the things we tried in this project.

# Chapter 2

# Literature Review

In this chapter we will glance through various techniques which have been tried for learning subcellular regulatory networks.This chapter has four sections.EGRIN, ARCANE and some other common methods are discussed in the first section. In the second section, we discuss how to reduce the complexity of the problem by describing a clustering algorithm named cMonkey.It was developed for use with inferelator, which learns the regulatory network which we describe in the third section. In the fourth section, we discuss the canopy approach to cluster multidimensional data.

## 2.1 Learning Subcellular Regulatory Networks

The main advantage in applying systems approach to the biological processes is to be able to predict the behavior of the biological process if we make a small change in environmental conditions or genes. Broadly speaking, there are two groups of algorithms for inferring the biological regulatory network. In the first group of techniques, we will see those which learn a network of unit less regulatory influences. The second group of algorithms differ from the first group in that it also produces some dynamic parameters, using which one can predict expression levels of nodes in different conditions. Below are described some of the popular algorithms for the first group.

- **ARCANE:** Using ARACNE, transcriptional regulatory network for mammalian cells was inferred. The algorithm is based on an important equality from data processing which says that if $G_a$ interacts with $G_b$ which in turn interacts with $G_c$ and there is no direct interaction between $G_a$ and $G_c$ then mutual information between the end nodes (which are $G_a$ and $G_c$) is less than minimum of mutual information between connecting edges ( which are ($G_a$,$G_b$) and ($G_b$,$G_c$)). Mutual information between each pair is stored as weights between them. Those weights which are below a threshold are removed from the network. Due to thresholding, weak interactions among the triplets are dropped.

- **Context Likelihood Relatedness:** Using this technique, transcriptional regulatory network for the organism E.coli was inferred.On the basis of mutual information between a pair of genes(or a gene and a TF), a score is calculated for each pair. After calculating the mutual information between genes and their regulatory agents, it then calculates likelihood of the mutual information in this network's context. It also calculates background distribution of mutual information and using that distribution it asserts those mutual interactions to be most probable, which have mutual information value substantially greater than the background distribution.

In the second group of methods, EGRIN gives the best results.Below we describe EGRIN which ,using the learnt dynamic parameters, predicts the expression level of genes in novel conditions.
**EGRIN:** Environmental and Gene Regulatory Influence Network (EGRIN) is a sequential application of mainly two algorithms so as to infer the regulatory network. EGRIN was successfully used to infer regulatory network of Halobacterium Salinarium.
At first, geneome sequencing of the organism concerned is done and on the basis of structural similarities between genes , different functions are assigned to them. It is followed by generation of data. Enviromental conditions were purturbed and expression level of genes are measured using microarrays. Next step is reducing the data complexity. Cmonkey algorithm is used to find the coregulated genes and those genes are then clustered together. At last inferelator is used to construct a dynamic network model from the clustered data.

Using following procedure, the regulatory network for H. Salinarum was inferred. cMonkey clustering was used to cluster 2400 genes. It clustered 1929 genes out of 2400 into 200 biclusters. Semantically, these biclusters contain genes which are corregulated under certain conditions. On this clustered data, inferelator was run and regulatory network was constructed. The output is essentially a set of equations which takes as input changes in environment and expression level of TFs and outputs the gene expression level of genes. The inferred expression level had a high degree of correlation with the actual expression level of gene.Now we will study in some depth the two algorithms used by EGRIN which are Cmonkey and Inferelator.

## 2.2 Cmonkey Biclustering Algorithm

Clustering of genes is done for various reasons. Major reason being that in biological networks, the number of free paramenters is much higher than the dimensionality of data. So clustering the genes directly reduces the free paraments and thus helps in a major way. The issue being that in general signal to noise ratio is high in biological experimental data. Clustering averages out the noise and imrpoves signal to noise ratio. The most natural way for clustering is to choose as metric the correlation coefficient between the expression of genes over different conditions. There is however, one issue is that some genes could have highly correlated gene expressions just by chance. So if two genes have correlated gene expression doesn't necessarily mean that they are corregulated. On the other hand, if two genes are co-regulated, then by very definition of co-regulation, their expression in different conditions will be very similar. To enforce that genes which are to be clustered in a cluster must have co-regulation, one could use the information about their common tasks, common metabolic pathways, cis-regulatory motifs etc.
However, every biological system is multi functional. This means the genes are co-regulated only in some set of conditions and not over all conditions. For clustering algorithm to encorporate this feature, individual genes must be put in different clusters on the basis of data obtained from a subset of environmental conditions. This type of clustering is also termed as biclustering.
**cMonkey bicluster model:** cMonkey biclustering model starts with initializing biclusters as seeds. In subsequent iterations genes are added and removed. It is modeled on Markov chain process and so the gene addition and removal is solely dependent on the current state of the bicluster. In short

the algorithm can be summarized as follows [3].

- A new bicluster is seeded with one gene.
- motifs are searched in the bicluster
- Conditional probability that each gene being a member of the chosen bicluster is calculated
- Using the conditional probability, moves are performed(gene is added or removed)
- In case the cluster has changed, control goes to step 2, otherwise to step 1.

There are few things to note about seeding a bicluster with a gene. Seeding a bicluster is done until number of biclusters reach a pre-specified limit. Genes which are not present in any bicluster are the ones eligible to be seeded into a new bicluster. Ones seeded, biclusters improve by using the joint likelihood conditions: clusters with high probability of membership are add and those with low probability are dropped. Size of the cluster and the overlap of the clusters are also checked using mathematical constraints. Now we will discuss Inferelator, the regulatory network generation module of EGRIN.

## 2.3 Inferelator

Inferelator is an algorithm which outputs a function which as input expression level of environmental factors , expression levels of TFs and interactions between them and outputs expression level of genes. It uses standard regression along with L1 regularization. One of the major plus points of this algorithm is that it can be applied for obtaining steady state conditions as well as for kinetic expression levels.

**Model Formulation**: Expression level of a gene/cluster say y is assumed to be a function of a vector of levels of TFs and other environmental factors say X,X = $(x_1, x_2, ........, x_n)$. Inferalator selects following kinetic equation to relate X and y:

$\tau dy/dx = -y + g(\beta * Z)$ [1]

- y: gene/cluster expression level
- X: vector of levels of TF and environmental factors
- Z: function of regulatory factors.
- $\beta$: Positive $\beta_i$ implies inductive effect of factor $X_i$ on y, negative $\beta_i$ implies inhibitive effect.
- $\tau$: time constant of the level of y in absence of any external influences.
- Function g(p):
  - g(p) = p if min(y) < p < max(y)
  - g(p) = min(y) if p < min(y)
  - g(p) = max(y) if p > max(y)

It is clear from equation [1] that both steady state experiment or a time series experiment are allowed experimental conditions. Z is used to model interaction between the different enviromental factors and TF. One could formulate various functions like XOR,OR using Z.

After formulating the model, we turn our attention to learning the parameters. We present a high level pseudocode representation of the algorithm [3]:

```
• "For each bicluster k
    – For each (TF $t_i$: transcription factor $t_i$)
        * Update the list of best single influences
        * For each TF $t_j$: update the best interaction list(min[$t_i$,$t_j$])
    – select from predictors and estimate model parameters with L1-shrinkage/CV
    – Model has been generated for bicluster k. Store it.
• Club together individual bicluster models into one global network"
```

## 2.4 Efficient Clustering of High-Dimensional Datasets

This section describes a two step approach to cluster the high dimensional data.The main idea is that one reduces the domain of potential matches for clusters using a cheap metric. This results in overlapping clusters called canopies. Second step involves using a more time expensive metric for clustering on objects belonging to same canopy.Since the domain of object is of significantly low size for the second step, with canopies, one can cluster high-dimensional data sets which was earlier not possible to do.
Two different metrics are used in this algorithm: a less time consuming similarity metric and a more time expensive and accurate similarity metric. Intuition behind the algorithm is that if those objects which are sufficiently different from each other can't be in the same clusters. So it makes sense to use the expensive clustering metric only on those objects which are not so different. There is one important concern regarding the loss of accuracy. For this algorithm to produce same output as that by using the more expensive and accurate metric of the two, it is imperative that all elements of any actual cluster must be a subset of atleast one canopy. For the second stage, one could use any of the stable clustering algorithms like k-means or agglomerative clustering with the restriction that distance between points which donot share a single canopy are set to infinity.
For this project, we needed the first step of the canopy algorithm, which is formation of overlapping subsets or canopies.

### 2.4.1 Inverted Index: A Cheap Metric

Inverted index is one of the cheap metrics which can be used as the cheap and approximate similarity metric to be used for canopy generation.For documents as examples with tokens being their boolean features, one can access list of all documents containing a token using inverted index. If we want to find all documents similar to a particular query we can first find all documents which have atleast one token of the query using inverted index. We can then use the actual distance metric to calculte the best match over this reduced set of documents. We can do this because a vast majority of documents which don't contain a single similar token will never be similar to it. So using inverted index one gets rid of all irrelevant documents and is left with a small set of potentially similar documents.So we see that inverted index can easily be used to efficiently calculate a distance metric that depends on simply the count of common tokens between the documents.

### 2.4.2 Canopy Creation Algorithm

We describe below the steps to create canopies. We need two thresholds T1 and T2, with T2¿T1. ( They can be optimally found out by using cross validation )

- Select a point p as the center for a new canopy from the list.
- Measure distance of the point from all other points using a cheap distance metric.
- All the points which are atmost T1 distance away from p are put in that canopy.
- All those points which are atmost T2 distance away from p are removed from the list.( They are ineligible to be canopy centers)
- Repeat until the list becomes empty.

# Chapter 3

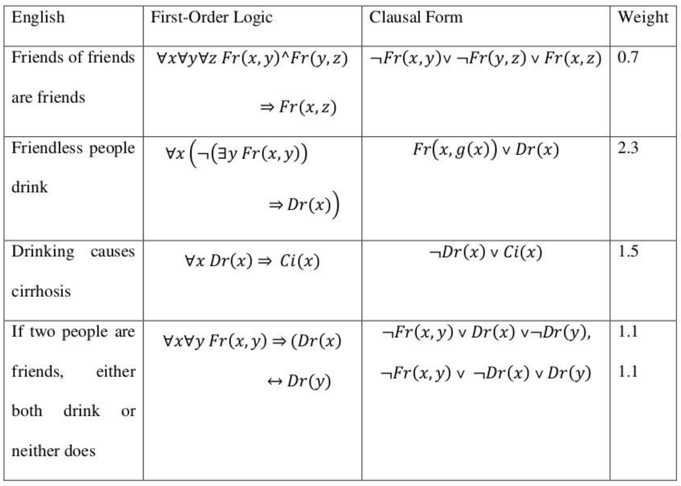
# Markov Logic Networks

For the entirety of the project we rely on MLN as our model framework and experiment with various models which we create in it. To begin with we reciprocate the results of a previous MTP done on the same problem[4]. We take its model as a Basic Model and do some more analysis on it which we describe in later sections. In the next chapter we describe our joint model which jointly infers the links and identifies corregulated genes. First we go to the basics of MLN.

## 3.1 Basics of MLN

A first-order knowledge base (KB) is a set of sentences or formulas in first-order logic (Genesereth & Nilsson, 1987).There is hardness in terms of realizability of a world defined by first-order logic which means, even on a voilation of one formula within the KB, the probablity of the world goes to zero. But real world scenarios don't have the luxury of boolean states. Hence Markov Logic Network was developed where a world has a finite real probablity of existance based on the number of formulae satishfied in its KB and also on the importance of the formulae quantified by their weights. Formula with high weights on getting untrue reduce the probability of world being true by a bigger margin than the formulae with low weights. If one changes the ground values of instance predicates so that more and more formulae become true, then the probablity of world increases(provided the weights of rules being made true are positive). Definition of Markov Logic Networks as proposed by Richardson and Domingos is [6]:

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



| 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 | $\forall x \left(\neg(\exists y\ Fr(x,y)) \Rightarrow Dr(x)\right)$ | $Fr(x,g(x)) \vee Dr(x)$ | 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 |

Figure 3.1: Example of a First-Order Knowledge Base and MLN, (Richardson, Domingos 2005)[4]

finite set of constants C = $\{c_1, c_2, ......, c|C|\}$, it defines a Markov Network $M_{L,C}$ as follows:

- $M_{L,C}$ 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.
- $M_{L,C}$ 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.

The probability of a particular world x which is specified by ground Markov Network $M_{L,C}$ comes directly from definition 1 as:

$$P(X = x) = \tfrac{1}{Z} exp(\textstyle\sum_i w_i n_i(x)) = \tfrac{1}{Z}\phi_i(x_i)^{n_i(x)}$$

Where $n_i(x)$ is the true number of groundings of $\phi_i$ in x, $x_i$ is the truth value of atoms appearing in $\phi i$ and $\phi_i(x_i) = e^{w_i}$ . One could also look at $M_{L,C}$ in a graphical perspective. This can be done directly applying the definition 1.

Nodes in the graph will be the ground atoms/predicates. There will be an edge between two node if the corresponding ground atoms come in atleast one ground instance of a formula. If we have the ground network and a database pertaining to that network, one could calculate the weight of different rules. Note that higher the weight, higher is the probability of the rule being true.

## 3.2 Learning Subcellular Regulatory Network: Basic Model

This model was developed in a previous MTP work by Alok Singhal at IIT Delhi. In this model MLN was used to learn the regulatory network for the organism H. Salinarium. H. Salinarium was picked because there are established works on the same theme and results of those works are easily replicable. In particular, we have EGRIN which is sequential application of cMonkey followed by Inferelator algorithm which learnt the regulatory network of H. Salinarium which served as a good benchmark for them.
**Dataset Description:** There are 2400 genes and 280 conditions in Salinarium dataset. Out of 280, 100 are TFs and rest being environmental factors. MLN in its current state works well with only binary data. Hence thresholding was done on the expression level of genes ,TFs and environmental factors to make them binary valued.
For the problem mentioned, following first order KB was defined:

1. ExpressesTF(+t, c)  Expresses (+g, c)
2. ! ExpressesTF(+t, c)  Expresses (+g, c)
3. ! Expresses(+g, c)

First rule says that if Transciption factor (TF) t is active in a condition c implies that gene g should be active in that condition. This captures the activating nature of TFs vis-a-vis Gene expression. Second rule says that when TF t is inactive in a condition, gene g should be active. This captures the inhibitory effect of TFs. Third rule says that in general genes should be

inactive.
+ sign indicates for each variable of that type, a separate weight was learnt. For example in the first rule, there is + sign of t and g. So for every TF t, and gene g, a weight was learnt over all the conditions. It makes sense because if for some TF t, gene g, weight comes out to be high, then MLN framework says that if TF t is active then it is highly probable that gene g is also active. We instantly recognize the role of weight here as an indicator of the presence of an actvating link from TF t to gene g. Lets do some numerical analysis on this MLN. Having 2400 genes,300 conditions and 100 TFs, we have $10^5$ weights to be learnt with $10^8$ ground clauses. At that point of time, there were no open source MLN implementations which could handle that number of clauses. In order to make problem solvable, they reduced the data complexity by clustering the genes using cMonkey clustering. MLN framework was then applied on clusters rather than on genes. Expression levels of cluster centers are just the mean of the expression levels of member genes.
For learning weights, discriminative learning was used. It was used as opposed to generative learning since evidence nodes were apriori known. (Evidence nodes are TF nodes with query nodes being gene nodes). Aim is prediction of expression levels of gene clusters given expression levels for TFs and other environmental factors. Let X denote the set of evidence atoms. Let Y be the query atoms, then conditional likelihood of Y given X is:

$$P(Y|X) = \frac{1}{Z_x} exp(\textstyle\sum_{j \in G_y} w_j g_j(x, y))$$

- $G_y$: set of clauses with atleast one query atom in it.
- $w_j$ : weight of $j^{th}$ clause
- $g_j(x, y)$: $g_j(x, y) = 1$ if jth clause becomes true when Y takes value y and X takes x otherwise 0.

Using MAP inference one can find the most probable state of Y given X. In their case out of 278 conditions, discriminative learning was used to learn from first 250 conditions and last 28 conditions were used for testing. After having learnt the regulatory network, one can get the activity level of genes in a novel condition using the expression levels of TFs and environmental conditions in that condition.

### 3.2.1 Results

We have done all our experiments, including the joint model and the improved basic model on Halobacterium dataset. We created the model in Deepdive and reciprocated the results. We got 73.5 % accuracy on cluster activity prediction. Going further from there, we did an eight point cross validation,ran the model on clusters as before and calculated the genewise accuracy. It came out to be 69%
One may observe that rule 1 and rule 2 are creating a subtle problem. Problem is that we don't have a semantic explanation for negative weights for rules 1 and rule 2. This also can cause our model to differ from the real data as we are adding unnecessary degeneracy in the model. So at the later half of the project we decided to stick with only one of the rules. This way the negative weights immediately meant the other rule and very low weight meant TF doesn't affect genes. It will be explicitly mentioned which rules are picked. If not mentioned, all rules can be safely assumed to be present.

# Chapter 4

# A Joint Model for Similarity Prediction and Regulatory Network Learning

## 4.1 Motivation and Predicate Definitions

We saw in previous chapter that due to the enormity of data, genes were clustered and subsequently the clustered data was fed into a MLN framework to get the regulatory network. Since the similarity between genes depends upon the underlying links to Transcription Factors, it is but natural to come up with a model which would learn the simialrity and the links jointly. For that to work we will need to relate links to similarity. This necessitates the introduction of new predicates. Note that in the previous simpler model we were no predicates for links or for similarity. Strength of links were inferred from the weight of rules. Below are the predicates with their meaning

- Active(g,c) : gene g is active in condition c
- ActiveTF(t,c): TF t is active in condition c
- LinkedA(t,g): TF t activates gene g
- LinkedI(t,g): TF t inhibits gene g
- SimilrGn(g1,g2): Gene g1 is similar to gene g2

From the data we are getting values of predicates Active and ActiveTF. LinkedA, LinkedI and SimilrGn are almost hidden.(SimilrGn(g,g) is True). We will see in the next section that we will achieve partial observability for SimilrGn predicate by applying the concept of canopies.

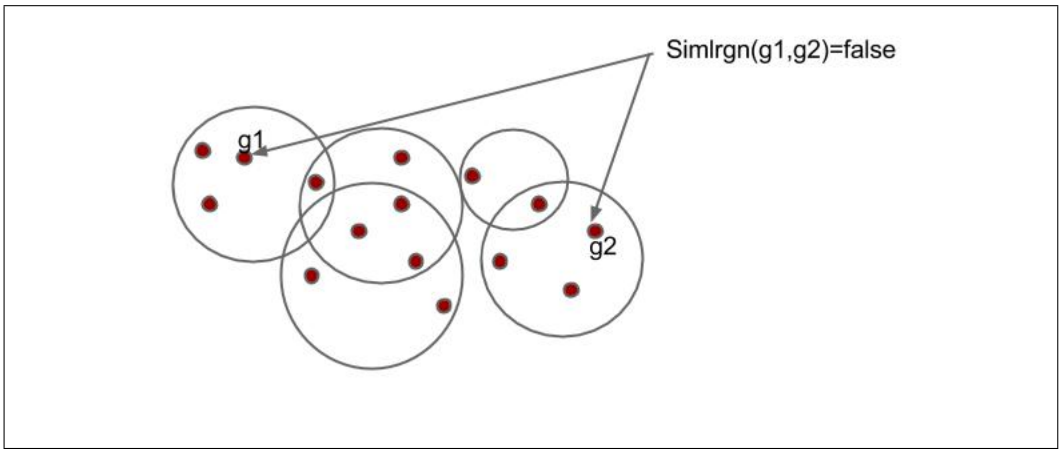


Figure 4.1: Figure showing genes as dots, and circles enclosing them as canopies. Those gene pairs which don't share atleast one canopy are set to false.

## 4.2 Partial Obseravability of SimilrGn Predicate

Intuition behind this section is that if the gene expression of some gene pair is very different then they can't have the same set of TFs affecting them. So the idea is to form canopies of genes. For those gene pairs which donot occur in any common canopy, we set the SimilrGn predicate to false as can be seen in Fig [4.1]. For other gene pairs, it is unknown.
For each gene we construct a vector of length equal to number of conditions. This vector contains the gene expression of that gene over those conditions. We use correlation coefficient as a metric for constructing canopies with our two thresholds being 0.2 and 0.4.

## 4.3 Model 0: Basic Model in Terms of New Predicates

This is a model which is conceptually equivalent to the Basic model. In this model we formulate rules only with LinkedA and LinkedI predicates

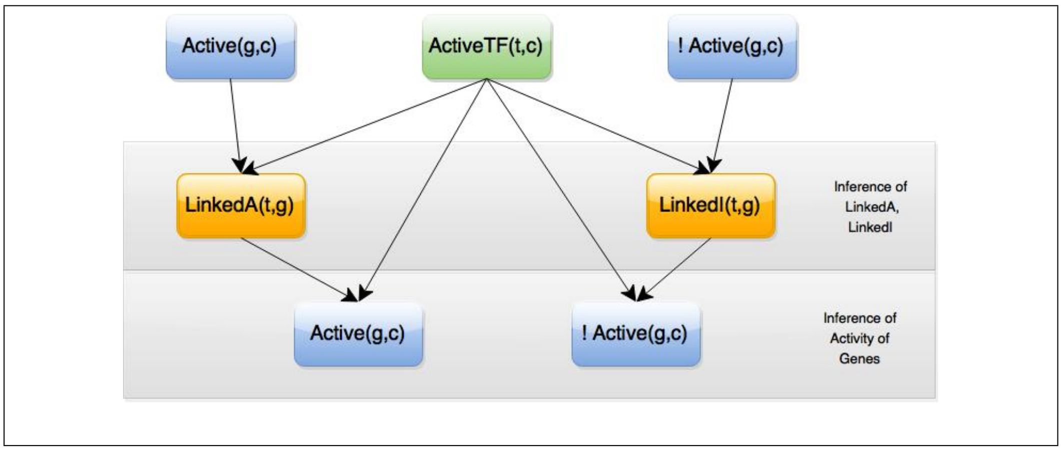


Figure 4.2: LinkedA is inferred when gene as well as TF is active in a condition. LinkedI is inferred when TF is active but gene is inactive in a condition. Finally a gene is active if a TF which activates it is active in a condition.Similarly a gene is inactive if a TF which inhibits it is active in a condition.

and we do not touch similarity in this model. LinkedA and LinkedI predicates are inferred from Activity of genes (Active(g,c)) and activity of TFs (ActiveTF(t,c)) as can be seen from figure [4.2]. Below are the rules of the model:

1. Active(g,c) $\wedge$ ActiveTF(t,c) => LinkedA(t,g)  +(t,g)
2. !Active(g,c) $\wedge$ ActiveTF(t,c) => LinkedI(t,g)  +(t,g)
3. ActiveTF(t,c) $\wedge$ LinkedA(t,g) => Active(g,c)  +(t,g)
4. ActiveTF(t,c) $\wedge$ LinkedI(t,g) => !Active(g,c)  +(t,g)
5. !Active(g,c)  +(g)

In the first rule LinkedA is inferred. a TF activates a gene g if when gene g as well as TF t are active in a condition. In second rule, LinkedI is inferred. TF t inihibits gene g when TF t is active but gene g is inactive in a condition. Next two rules are about inferring activity of gene. Gene g is active if there is a TF t which activates it and TF t is active.Gene g is inactive if there is a TF t which inhibits it and TF t is active.

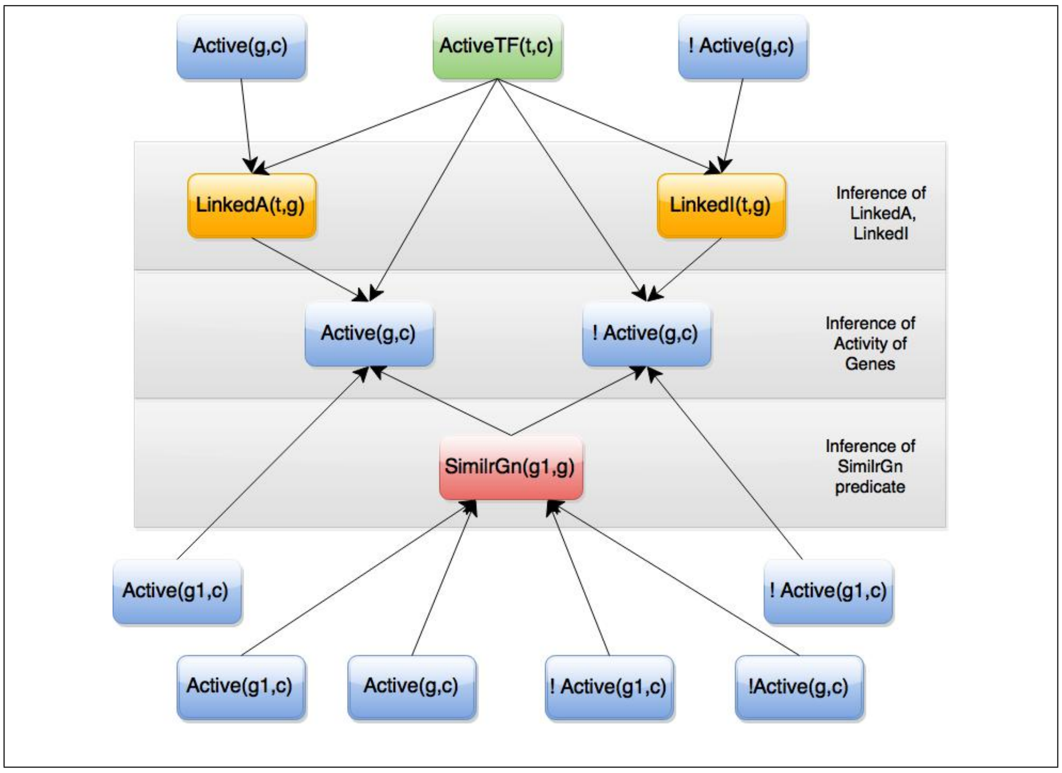


Figure 4.3: LinkedA is inferred when gene as well as TF is active in a condition. LinkedI is inferred when TF is active but gene is inactive in a condition. Genes are similar when they are either both active or both inactive in a condition. Finally a gene is active if a TF which activates it is active in a condition or a gene which is similar to it is active in that condition. Similarly a gene is inactive depends whether a TF which inhibits it is active in a condition or a gene which is similar to it is inactive in that condition.

## 4.4 Model 1: Joint model, Inferring Similarity from Activity

This is the first joint model we experimented with. This model has rules which predicts Linked predicates (LinkedI and LinkedA) and Similarity predicates(SimilrGn) both from the gene expression. Gene expression is inferred from combination of Linked predicate with Activity of TFs and Similarity with Activity of genes. This can be seen in Fig [4.2]

Below are the rules of the model. Figure [4.2] captures the semantics of these rules:

1. Active(g,c) $\wedge$ ActiveTF(t,c) => LinkedA(t,g) +(t,g)
2. !Active(g,c) $\wedge$ ActiveTF(t,c) => LinkedI(t,g) +(t,g)
3. ActiveTF(t,c) $\wedge$ LinkedA(t,g) => Active(g,c) +(t,g)
4. ActiveTF(t,c) $\wedge$ LinkedI(t,g) => !Active(g,c) +(t,g)
5. Active(g1,c) $\wedge$ Active(g2,c) => SimilrGn(g1,g2) +(g1,g2)
6. !Active(g1,c) $\wedge$ !Active(g2,c) => SimilrGn(g1,g2) +(g1,g2)
7. Active(g1,c)$\wedge$ SimilrGn(g1,g2) => Active(g2,c) +(g1,g2)
8. !Active(g1,c)$\wedge$ SimilrGn(g1,g2) => !Active(g2,c) +(g1,g2)
9. !SimilrGn(g1,g2) +(g1,g2)
10. !Active(g,c) +(g)

In the first rule LinkedA is inferred. a TF activates a gene g if when gene g as well as TF t are active in a condition. In second rule, LinkedI is inferred. TF t inihibits gene g when TF t is active but gene g is inactive in a condition. Next two rules are about inferring activity of gene. Gene g is active if there is a TF t which activates it and TF t is active.Gene g is inactive if there is a TF t which inhibits it and TF t is active. Next two rules infer similarity. Genes are similar when they are either both active or both inactive in a condition. Next two rules (7,8) infer gene activity. gene g is active if there exist a gene g2, which is similar to it and it is active in that condition. Similarly gene g is inactive if a gene which is similar to it is inactive in that condition. Next two rules say that by default genes are inactive and are dissimilar to each other.

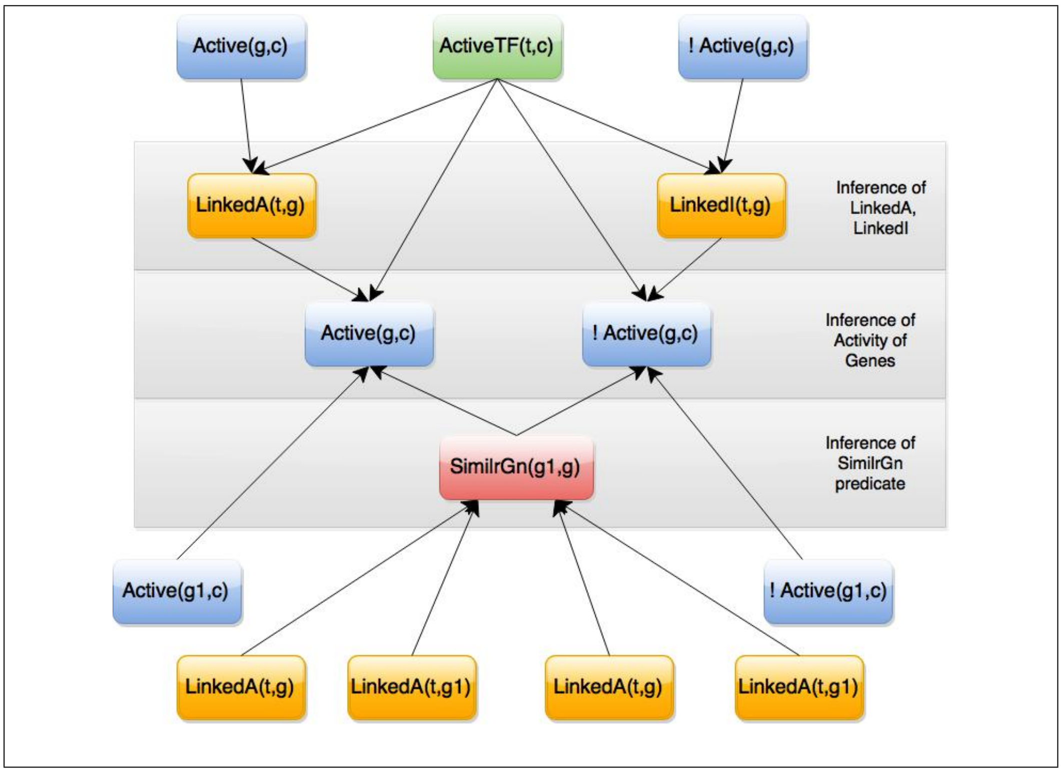


Figure 4.4: LinkedA is inferred when gene as well as TF is active in a condition. LinkedI is inferred when TF is active but gene is inactive in a condition. Genes are similar when they have same TFs which link with them via LinkedA and LinkedI. Finally a gene is active if a TF which activates it is active in a condtion or a gene which is similar to it is active in that condition. Similarly a gene is inactive if a TF which inhibits it is active in a condition or a gene which is similar to it is inactive in that condition.

## 4.5 Model 2: Joint Model, Inferring Similarity from Links

This model is different from Model1 in its inference rules of similarity. Here we are inferring similarity not from gene activity but from similar links. We say that a gene pair is similar if they have identical links. This definition is closer to the actual semantic meaning of similarity. It is so because there can be noise in the activity data. On inferring links, we hope that noise gets canceled out and hence the inference of similarity will be more authentic.

Below are the rules of the model. Figure [4.2] captures the semantics of these rules:

1. Active(g,c) $\wedge$ ActiveTF(t,c) => LinkedA(t,g) +(t,g)
2. !Active(g,c) $\wedge$ ActiveTF(t,c) => LinkedI(t,g) +(t,g)
3. ActiveTF(t,c) $\wedge$ LinkedA(t,g) => Active(g,c) +(t,g)
4. ActiveTF(t,c) $\wedge$ LinkedI(t,g) => !Active(g,c) +(t,g)
5. LinkedA(t,g1) $\wedge$ LinkedA(t,g2) => SimilrGn(g1,g2) +(g1,g2)
6. LinkedI(t,g1) $\wedge$ LinkedI(t,g2) => SimilrGn(g1,g2) +(g1,g2)
7. Active(g1,c)$\wedge$ SimilrGn(g1,g2) => Active(g2,c) +(g1,g2)
8. !Active(g1,c)$\wedge$ SimilrGn(g1,g2) => !Active(g2,c) +(g1,g2)
9. !SimilrGn(g1,g2) +(g1,g2)
10. !Active(g,c) +(g)

## 4.6 Model 3: Model 2 with Basic Model Rules Added

We see that there are so many unknowns that the learning may converge to some local minima. Hence we try to enforce the learning to go in the ’region’ where the basic model took us by adding those rules in our model.This model was added for the artificial data experiments.

## 4.7 Results

We used deepdive software for MLN Weight learning and inference. For weight learning it uses gradient descent. We did a grid search on the decay factor and the stepsize parameter of gradient descent to get to the optimal

| Different Models | Genewise Accuracy |
|---|---|
| Majority Class | 51.2 |
| Basic Model(on clusters) | 65.67 |
| Basic Model(on genes) | 66.74 |
| Model0 | 62.25 |
| Model1 | 63.27 |
| Model2 | 66.43 |

learning parameters. Note that we are partitioning our data into training and validation set and we check the correctness of the model by comparing the predicted and actual output of activity of genes. In short the accuracy is measured on Active predicate.
All of the three models performed similar to that of the basic model. Due to the large size of the data, we took 130 genes from 6 clusters obtained by cmonkey Clustering and did our experiments with models. We notice that our best model, namely Model2 also doesn't outperform the basic model. The reasoning behind Model2 to work better was that it conceptually captured the links as well as the similarity between genes. When we analyzed the weights, we found out that the similarity was not getting captured as can be seen in Figure [4.5] and Figure [4.6]. From the figure we see that the weights for those rules which include the similarity predicate came out to be negative, which shows that the interpretation of the learnt model doesn't match with expectations on which the model was constructed.

We note that here we have 190 Transcription factors and 130 genes. For each gene, only a handful of TFs are responsible for its activation and its deactivation. This enormous degree of freedom is what causes the model to fit a different set of weights. To simplify things first, we move to generating our own data from a artificially created regulatory network and test the models.

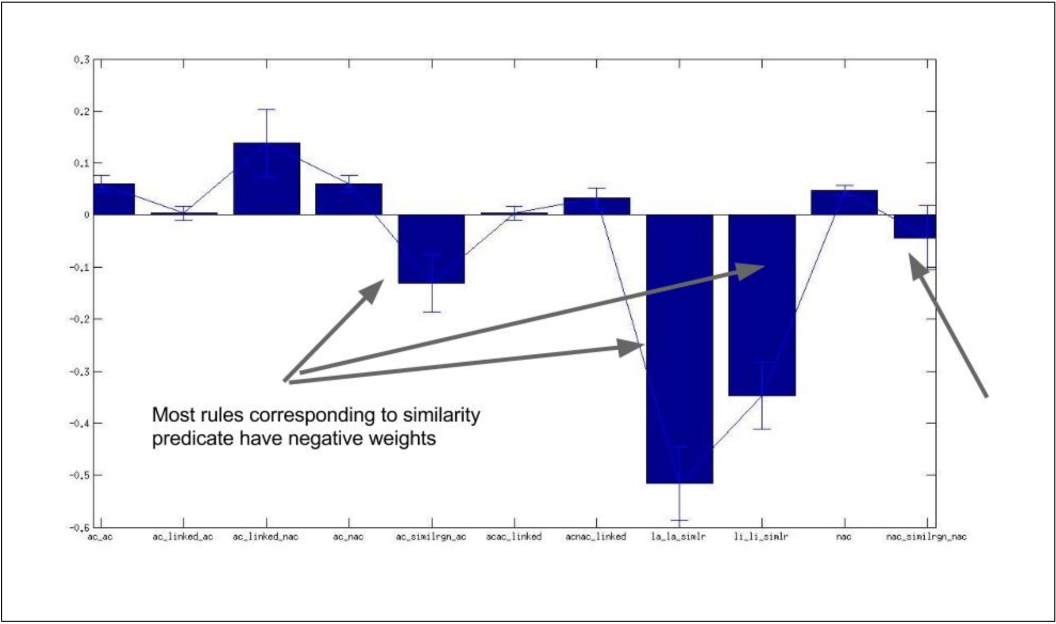


Figure 4.5: Figure shows the mean and the standard deviation of weights learnt by deepdive for Model1. We see that for the weights involving similarity predicates, mean is negative and even mean + std deviation is also negative. This indicates the weights learnt for rules involving similarity predicates have a different meaning. Hence Similarity is not getting captured.

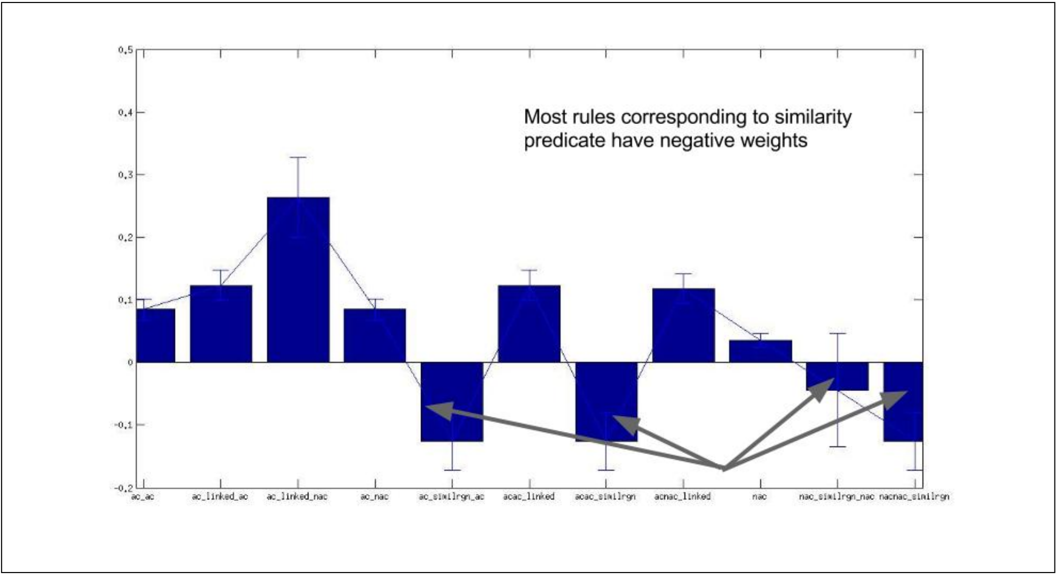


Figure 4.6: As in previous figure, for Model2 also, we see the same outcomes namely similarity is not getting captured.

# Chapter 5

# Experiments on Artificial Data

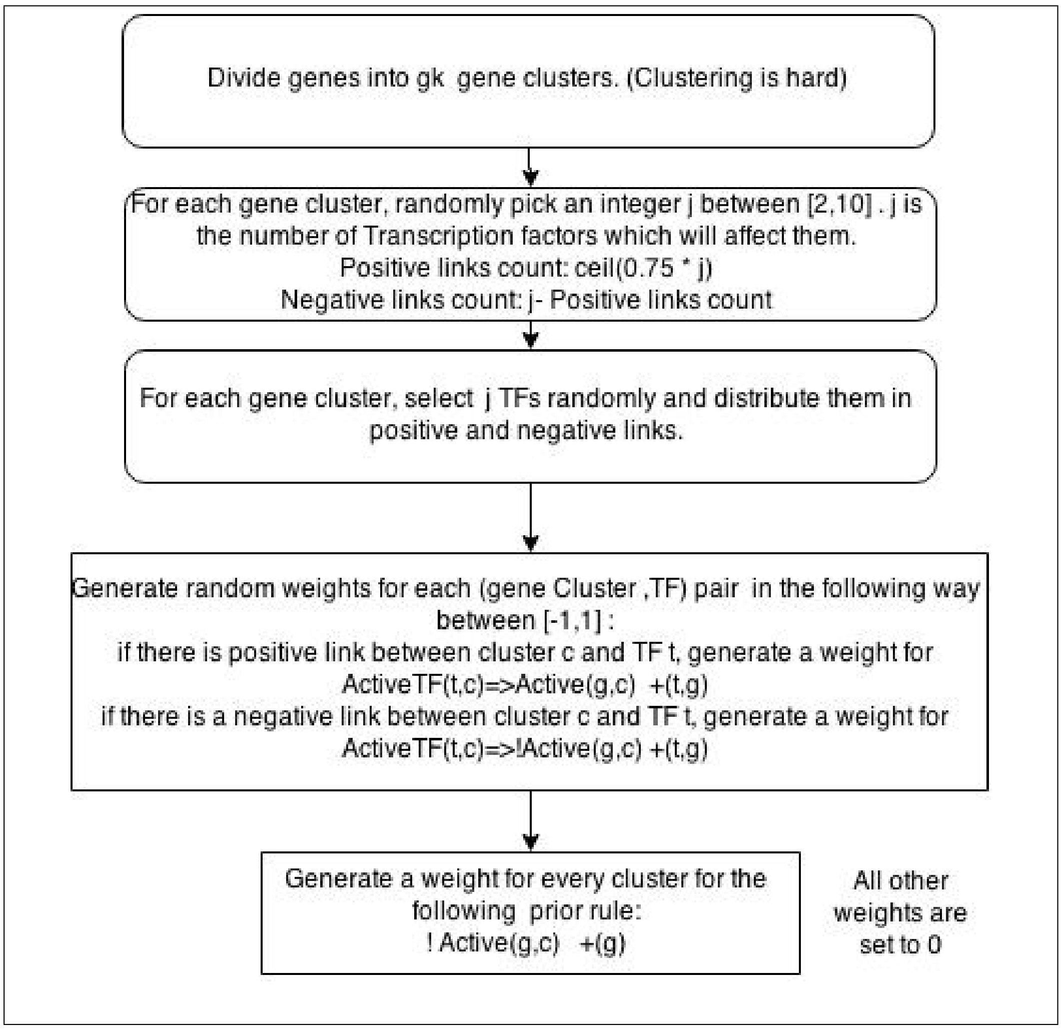
In this chapter we will first describe the methodology of Regulatory Network generation and generation of data from it. In the next section we will discuss the results of the different models on it.

## 5.1 Artificial Regulatory Network Generation

The idea is to fix the number of (hard) clusters. For each cluster pick the TFs randomly which will have links to it. Next we divide the selected TFs into activating TFs and inhibitory TFs. We then randomly generate weights for each such cluster TF pair. In later experiments only one type of link is kept between TFs and genes. In that case weights are allowed to be negative. Positive and negative weights captures what was intended to capture from two types of links between TFs and genes. MLN generation is explained in the figure [5.1]. All genes belonging to a cluster will inherit the links generated for that particular cluster. So two genes belonging to same cluster will have identical set of links. It is coded in MATLAB and parameters such as number of TFs, number of Genes, number of Conditions, Range of weights, minimum number of links allowed, maximum number of links allowed can easily be configured.

## 5.2 Data Generation

For generating one example(in real data it corresponds to one condition) from the Regulatory network, we first randomly set the ActiveTF predicate instances to true or false. Now for each gene g and condition c, we first find out the set S of clauses in which Active(g,c) predicate occurs. Let $S'$ be a subset of S which corresponds to all clauses which become true when


Divide genes into gk gene clusters. (Clustering is hard)
For each gene cluster, randomly pick an integer j between [2,10] . j is the number of Transcription factors which will affect them.
Positive links count: ceil(0.75 * j)
Negative links count: j- Positive links count
For each gene cluster, select j TFs randomly and distribute them in positive and negative links.
Generate random weights for each (gene Cluster ,TF) pair in the following way between [-1,1] :
if there is positive link between cluster c and TF t, generate a weight for ActiveTF(t,c)=>Active(g,c) +(t,g)
if there is a negative link between cluster c and TF t, generate a weight for ActiveTF(t,c)=>!Active(g,c) +(t,g)
Generate a weight for every cluster for the following prior rule:
! Active(g,c) +(g)
All other weights are set to 0
Figure 5.1: A step by step procedure to generate the regulatory network.

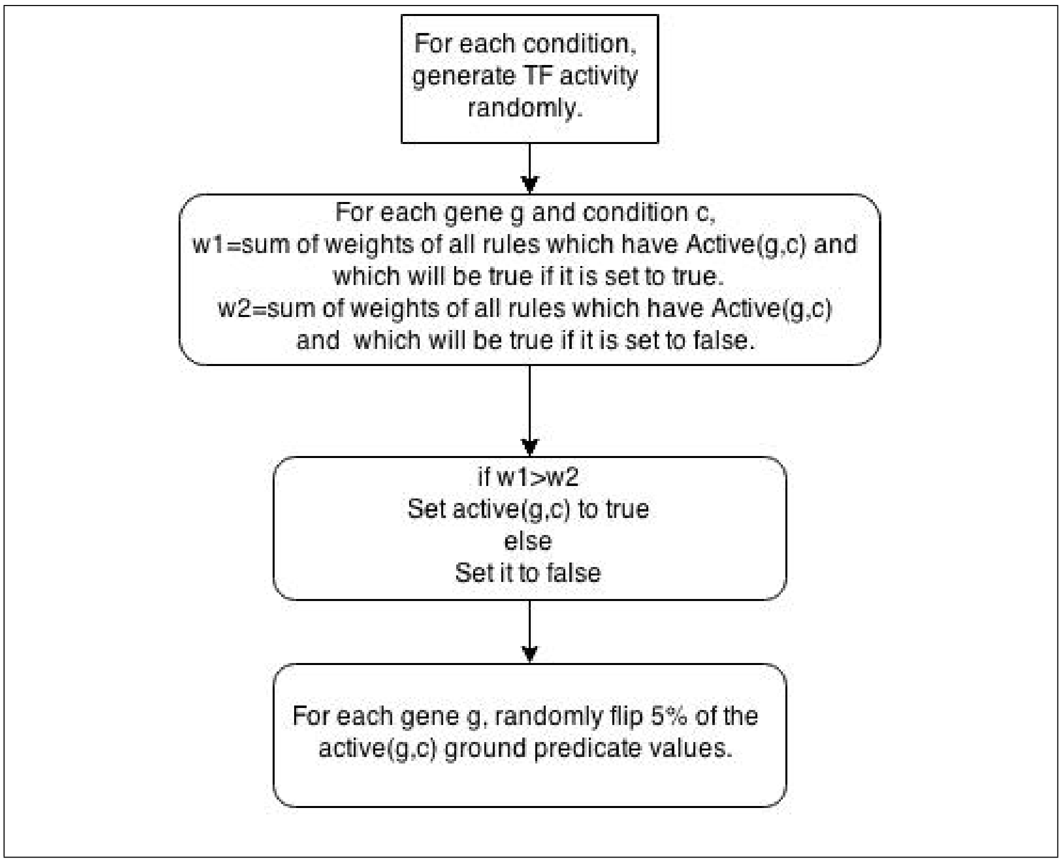
Active(g,c) is set to true. We calculate the sum of weights of every clause in $S'$, call it $sum_true$. Let $S''$ be set of all clauses in S which becomes true when Active(g,c) is set to false. Let us call the sum of weights of $S''$ as $sum_false$. We set Active(g,c) to true if $sum_true$ is greater than $sum_false$, otherwise we set it to false. Data generation procedure is explained in Figure [5.2]. Since for each gene in a cluster, links are identical, so equivalently , clauses in which they occur are identical 'functionally'. So the setting of true/false will also be identical for all genes in a cluster. In order to add some noise to it so that it corresponds to real world scenario, we flip 5 percent of all instantiated predicates of Active(g,c) for each gene g.
We have made some changes with respect to the graph generation process. For example we have changed domain of weights from [0,1] to finite domain {-1,1}. At some places we have changed the sparsity of the network by changing the maximum/minimum number of links allowed. All configuration specifications will be mentioned at the start of experiment section.

## 5.3 Results

Over all experiments we have taken 86 % of the generated conditions(examples) as training set and remaining 14 % as test set. This is done for all (example counts) conditions. Note that in this artificial setting, we may explicitly check the quality of links retrieval, in addition to checking the accuracy of Active(g,c) prediction.

### 5.3.1 Experiment 1: Comparing Basic Model and Joint Model on Artificial Data.

- Gene count = 50
- TF count = 60
- Minimum Links Allowed = 2
- Maximum Links Allowed = 10


For each condition, generate TF activity randomly.
For each gene g and condition c, w1=sum of weights of all rules which have Active(g,c) and which will be true if it is set to true. w2=sum of weights of all rules which have Active(g,c) and which will be true if it is set to false.
if w1>w2 Set active(g,c) to true else Set it to false
For each gene g, randomly flip 5% of the active(g,c) ground predicate values.
Figure 5.2: Procedure for generating 'noisy' data from the regulatory network

- Fraction of Links which are activating = 75%
- Weights domain = [0,1]
- Rules Included = 1,2,3

**Basic Model**:
Below table gives the test accuracy of prediction of Active(g,c) predicates.

| Model Type | 50 | 150 | 300 | 600 | 750 | 900 | 1500 |
|---|---|---|---|---|---|---|---|
| Majority Class | 0.72 | 0.79 | 0.75 | 0.77 | 0.78 | 0.78 | 0.75 |
| Basic Model | 0.76 | 0.81 | 0.78 | 0.78 | 0.80 | 0.82 | 0.87 |

From above table we observe that when the condition count goes to 1500, then we see some learning. Before it the basic model is around the same ballpark as Majority Class.

**Model 3**:
Below table gives the test accuracy of prediction of Active(g,c) predicates.

| Model Type | 100 | 150 | 300 | 450 | 600 | 750 |
|---|---|---|---|---|---|---|
| Majority Class | 0.80 | 0.73 | 0.74 | 0.77 | 0.76 | 0.77 |
| Model3 | 0.79 | 0.74 | 0.76 | 0.78 | 0.76 | 0.77 |

We see that Model3 doesn't do better than the Majority class prediction. In the hindsight it seems intuitive. For Basic model, we see that effective learning takes place when we reach 1500 conditions. This model have much more complexity and hence needs even greater amount of data for proper learning. However, the size of MLN grows 'fastly' with increase in conditions count and is not possible in deepdive to run the model3 with big enough condition count for which the joint model shows some improvement.
Another point comes up when we inspect the links retrieval from the model. From the table below it can be seen clearly that the model is learning totally different network which 'fits' the data in terms of prediction of Active predicate. Columns correspond to condition count. posRecall is the Recall for the positive links (LinkedA predicate). posPrecision is the precision for the positive links. Similarly negRecall and negPrecision are the recall and precision corresponding to negative Links (LinkedI).

**Model3: Links Analysis**

| Metric | 100 | 300 | 450 | 600 | 750 |
|---|---|---|---|---|---|
| posRecall | 0.18 | 0.16 | 0.20 | 0.27 | 0.25 |
| posPrecision | 0.06 | 0.07 | 0.08 | 0.09 | 0.08 |
| negRecall | 0.62 | 0.18 | 0.45 | 0.75 | 0.49 |
| negPrecision | 0.02 | 0.01 | 0.02 | 0.03 | 0.02 |

For the case of Basic Model also we get similar results. In basic model, we are inferring a link to be a positive link if the weight of the first rule of the model comes out to be positive. Similarly, negative link is assumed if the weight of the second rule comes out to be positive. We inferred three possible reasons for the poor precision and recall values for link retrieval. The first reasoning was that since every weight in MLN setting gets some finite weight, so this will result in bad precision and recall values for links. So we did thresholding of weights, and assumed links to be present only if weights were above certain threshold. But thresholding the links didn't improve the link retrieval as is evident from the precision and recall curves. Graphs [5.3] and [5.4] are precision recall curves for positive and negative links respectively .

This second argument for low links retrieval was based on high degeneracy present in the model. On an average, number of links for a gene is 6 (mean of minimum and maximum allowed link count). But we are given 50 TFs. Due to this large degeneracy, MLN framework apparently 'overfits' the data to get some regulatory network which behaves similarly as far as Active predicates are concerned. The third argument also is based on degeneracy of the model. We had two different rules which were semantically negation of one another. This could in essence be captured in one single rule whose weights can be both positive and negative.
Above two arguments motivated us to check the performance of the model when the TF count is low and also with only one of the two (activating or negating TF) rules. This would mean less number of possible Regulatory networks satisfying the generated data. This would also mean a simpler domain, and hence learning is expected to be accomplished with relatively low condition count.

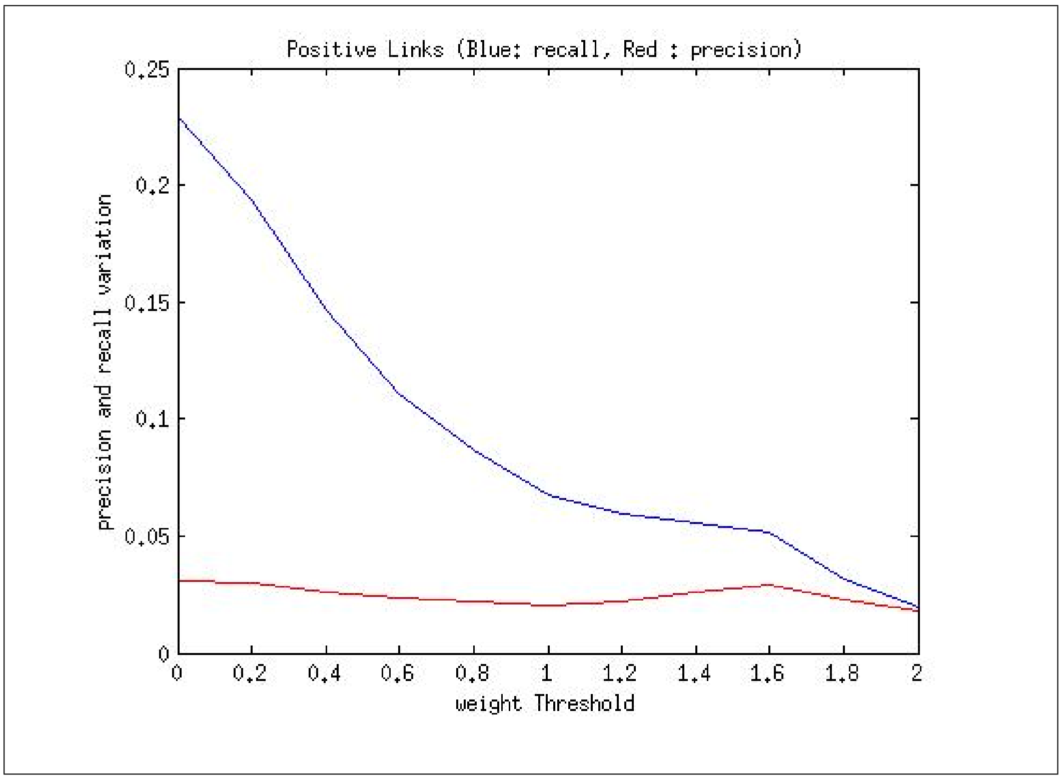


Figure 5.3: Figure shows variation of positive Link precision and recall with thresholding. Even with high thresholds, precision doesn't go up, thereby indicative of the fact that a different model has been learnt.

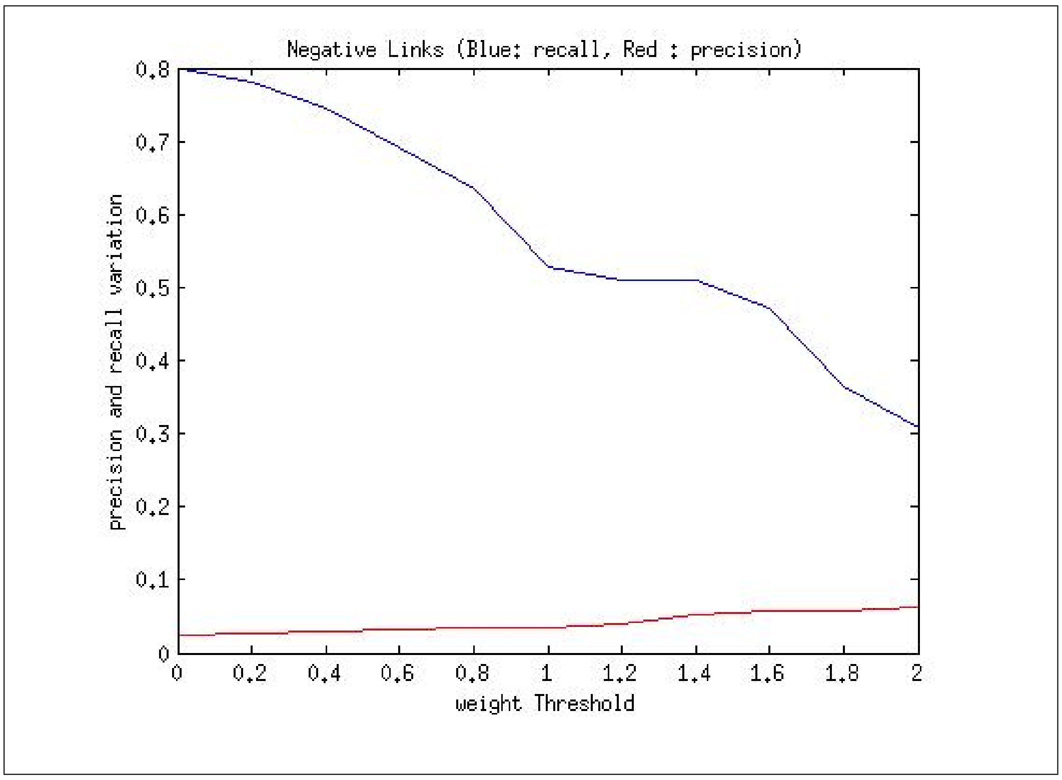


Figure 5.4: Figure shows variation of negative Link precision and recall with thresholding. As in case of positive links, even with high thresholds, precision doesn't go up, thereby indicative of the fact that a different model has been learnt.

### 5.3.2 Experiment 2: Use of Improved Basic Model on Simpler Network

- Gene count = 50
- TF count = 10
- Minimum Links Allowed = 1
- Maximum Links Allowed = 3
- Weights domain = {-1,1}
- Rules Included = 2,3

Basic model(improved) was able to give high accuracy of 87.6% over the validation set (for prediction of gene activity) and was able to retrieve the links to a appreciable degree. Below is the precision recall curve for the link retrieval and the learning curve(wrt gene activity prediction).

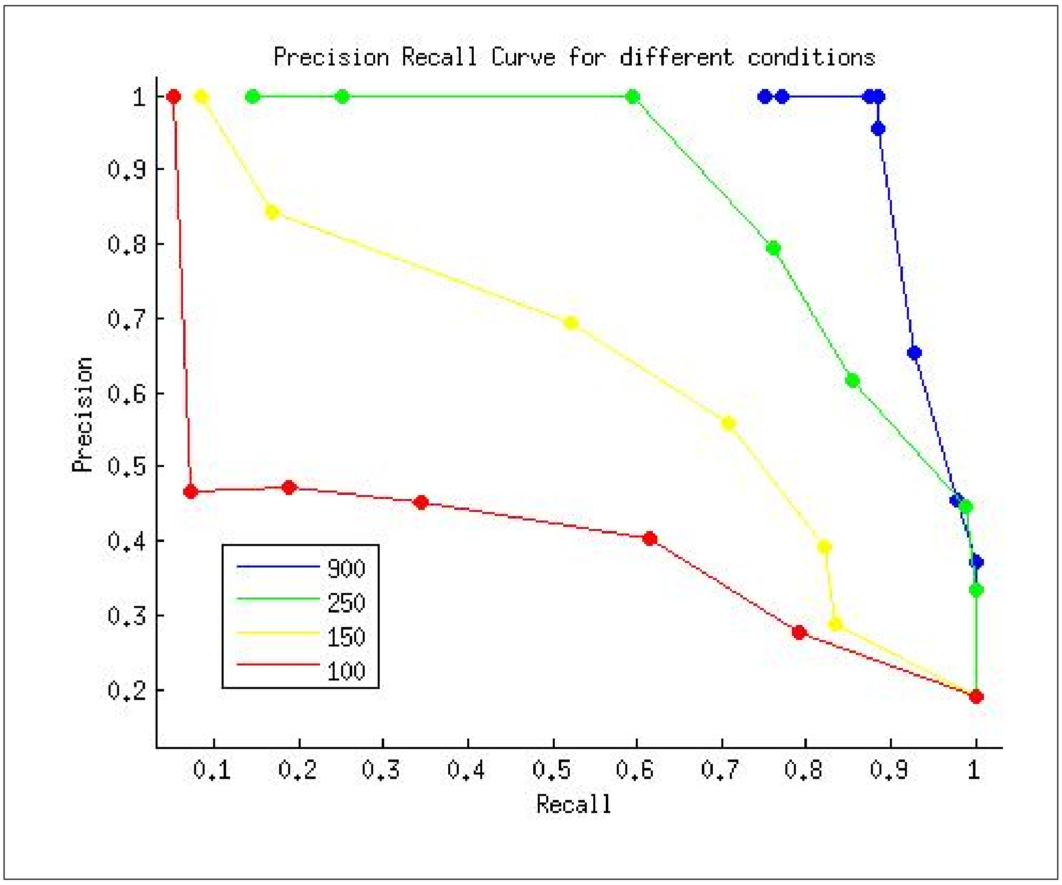


Figure 5.5: Figure shows variation of precision recall curve with number of conditions(examples). We see a nice improvement in link prediction with the increase in number of conditions

To understand more about the problem at hand we did a third experiment where we varied the link upper and lower limits. Before we move on, it is worth noting here that gene expression prediction in case of Joint model (Model 3) is good. Issue is the poor link retrieval. Below is the learning curve for Model 3 and Basic Model over gene expression prediction.

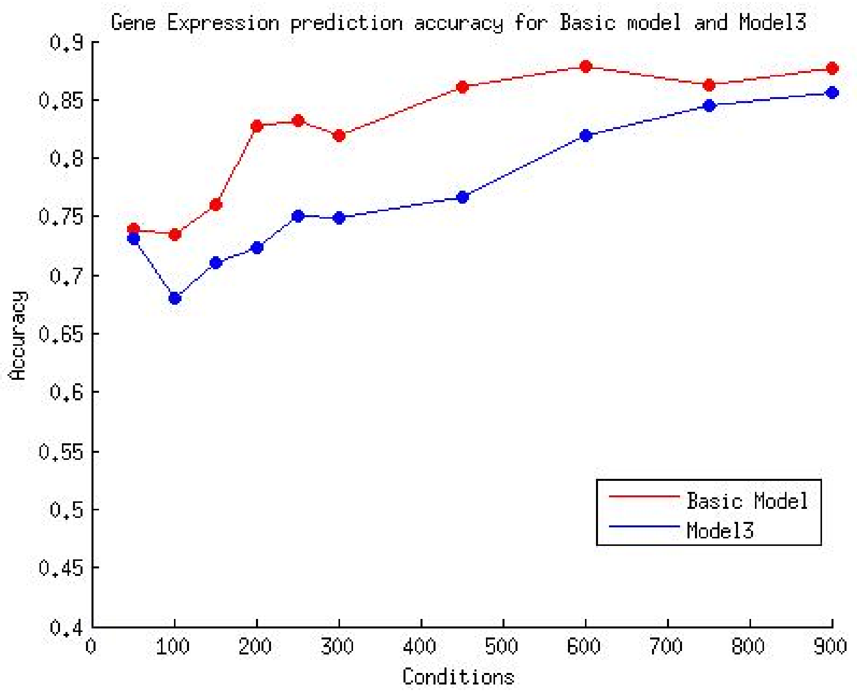


Figure 5.6: Figure shows variation of accuracy (of gene expression prediction) with number of conditions for basic Model and (Joint)Model 3

### 5.3.3 Experiment 3: Understanding Effect of Density Variation of Regulatory Network on Learnability of Links.

- Gene count = 50
- TF count = 10
- Minimum Links Allowed = Variable
- Maximum Links Allowed = Variable
- Weights domain = {-1,0,1}
- Rules Included = 2,3

In the first part of the experiment, we varied the density of the network i.e the links. We calculated the gene expression prediction accuracy and the precision recall for links retrieval for three sets of link limits: {(1,3),(2,6),(4,8)} where in (a,b) a and b are the minimum and maximum links allowed respectively. Here the weights are from {-1,0,1}.

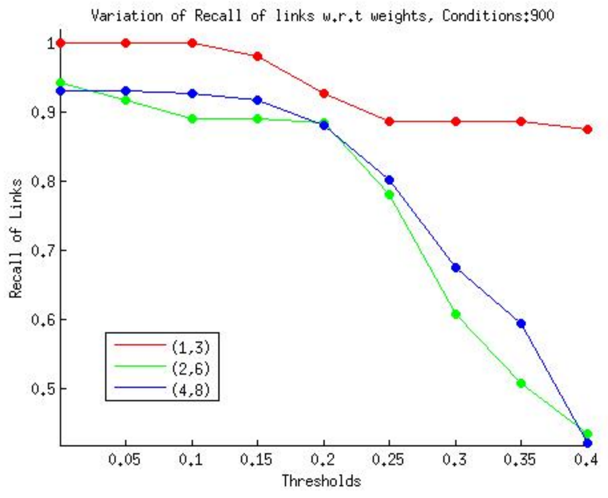
We see that the best precision-recall curve we get is in the (1,3) link set. Intuitively a simpler network is easier to learn.

Results of this experiment is promising with basic model. But in real data, we have 190 TFs. So the completion of this set of experiments demands one with that many TFs. Number of genes doesn't matter for the basic model as genes are independent of each other. Hence we can independently calculate weights and activity of genes.

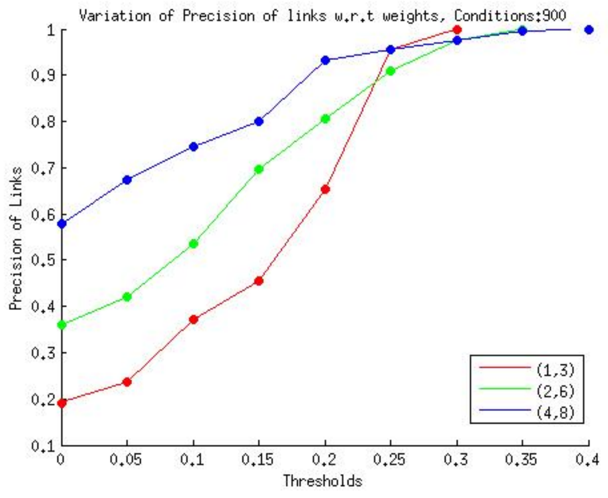
### 5.3.4 Experiment 4: Regulatory Network Density Variation in Real Sized Network

- Gene count = 50
- TF count = 190
- Minimum Links Allowed = Variable
- Maximum Links Allowed = Variable
- Weights domain = Discrete : {-1,0,1}
- Rules Included = 2,3

In this part of the experiment, we experimented on different maximum,minimum weight limits and calculated the area under the precision recall curve. We got same kind of results as we had got in the previous experiment. As we increase the average number of links allowed, precision goes up and recall comes down. In case of recall however, there is significant variance. As for the area under the curve, it comes down as we increase the average number of links per gene.



(a) Model with (1,3) link-set has the best recall. It is due to the simpler model of (1,3)



(b) Model with (4,8) link-set has the best precision. It can be argued that it having the largest number of links makes the precision go up.

Figure 5.7: Precision vs threshold and recall vs threshold plots

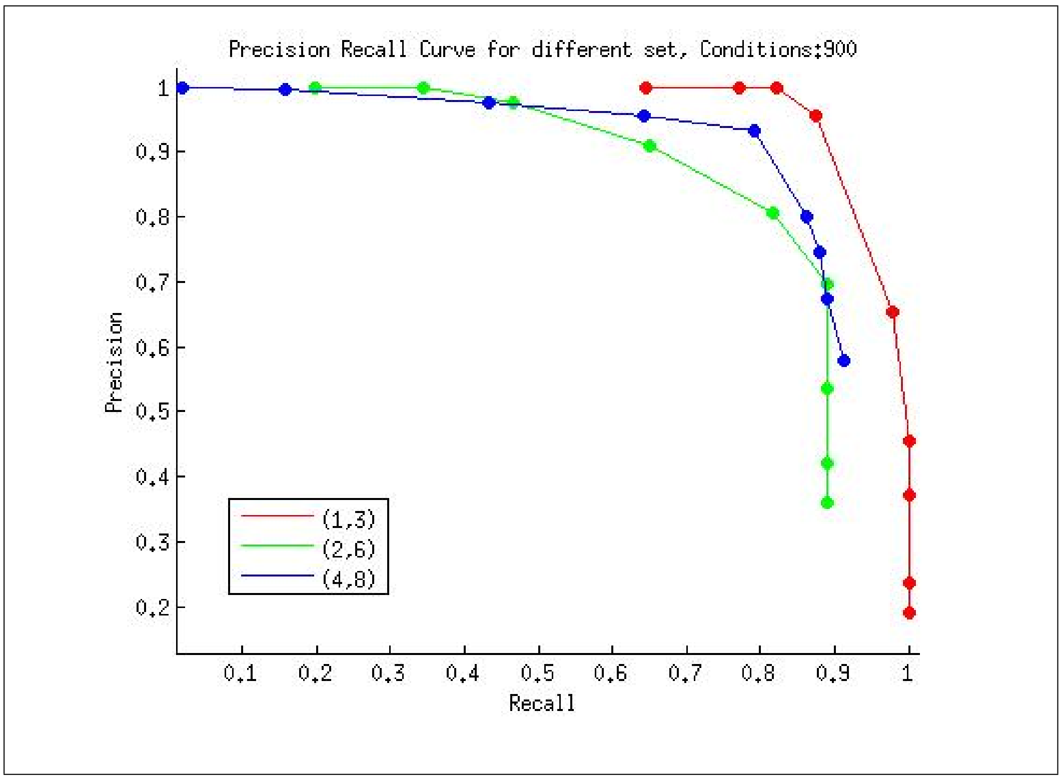


Figure 5.8: Figure shows Precision Recall curves for the three models with differing link sets: (1,3),(2,6),(4,8)

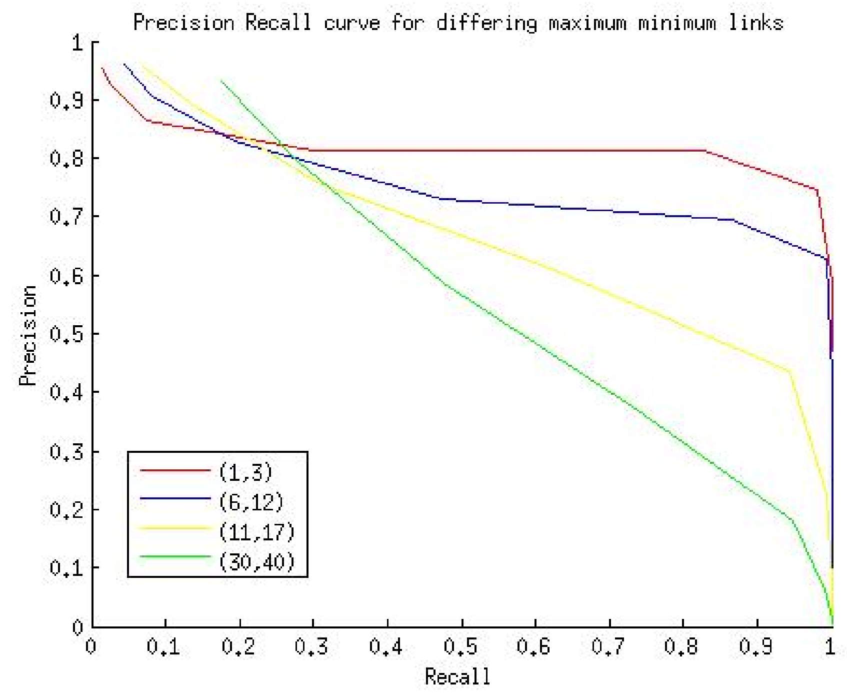


Figure 5.9: Figure shows Precision Recall curve for different link sets for boolean weights with 190 TFs. It is evident from figure that with increase in the average number of links, we see a decline in retrieval of links.

This experiment shows that the apparent high degeneracy in the model occurring due to large TF count and less number of links from TF to gene on average doesn't lead to less retrieval of links as compared to the degeneracy occurring due to redundant rule. This is inferred by looking at results of experiment 1 and experiment 4. Note that the prediction of gene activity is high in both cases. The concern was that a totally different regulatory network from the artificial network was being learnt in former case.

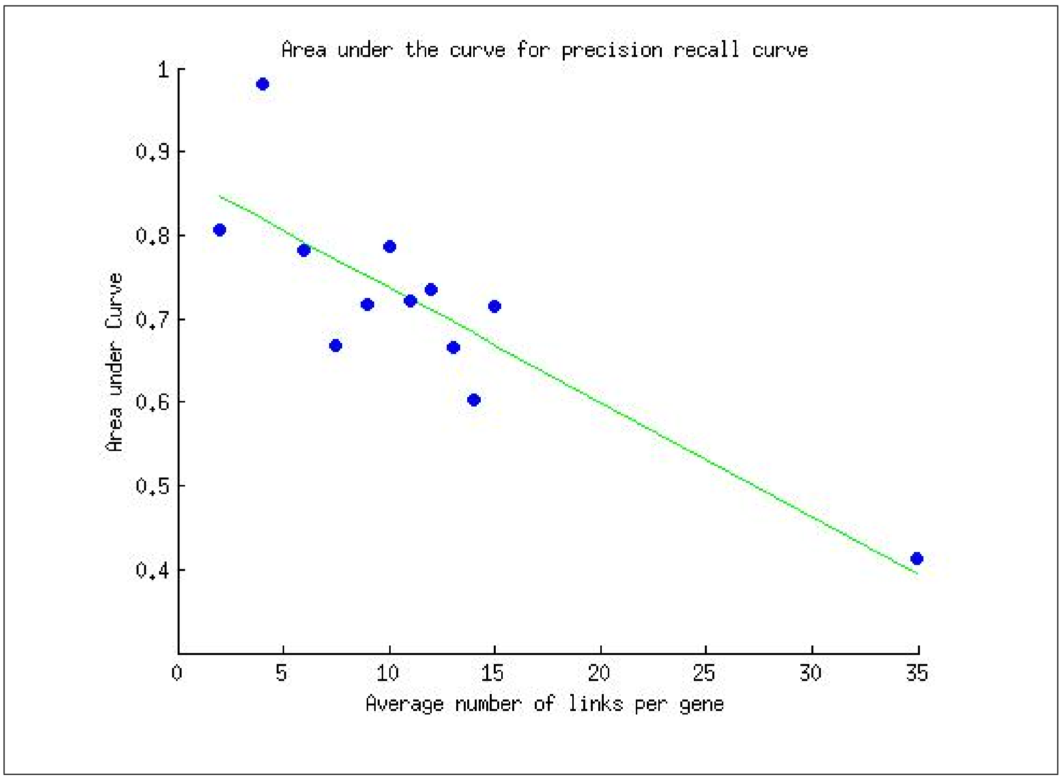


Figure 5.10: Figure shows Area under curve of precision recall for the case of link retrieval plotted with average number of links per gene. We see that as the average number of links increase, area under precision recall curve decreases.

# Chapter 6

# Discussion

Here we will present to the reader a qualitative description of the entire work done in this project more or less in the order in which the project evolved. We start with a simple MLN model developed previously to solve the same problem in an earlier MTP at this institute. In that MTP, the genes were clustered because of two reasons. First was to reduce the data complexity so as to make learning the regulatory network feasible. Second reason was to average out the noise by the clustering.

Our project started with the idea that since clustered genes are co-regulated and that co-regulated genes should be in same cluster, it makes more sense to do a joint inference of the clusters and the regulatory network rather than first clustering the data and then subsequently finding the links as was done in the mentioned MTP work and also in EGRIN. It is so because clustering should re-enforce link retrieval process and vice versa.

There was a problem that biological data is inherently noisy. EGRIN tried to handle this issue by preclustering the data and then feeding clustered data to its regulatory network generation algorithm. We on the other hand tried to handle it inside our algorithm. For inferring similarity, we designed a two step filter in MLN. In the first step we inferred positive and negative links from gene activity and TF and environmental activity. In the second step we inferred similarity from the links. With this two step filtering we hoped to tackle the effect of noise on the inference of gene expression.

On a different note, basic model had a simplification which went unnoticed in the previous MTP. There were no links between the genes in the regulatory network. So subnetwork for each gene was disconnected from the sub network of other gene. Hence it was possible to run the simple model for each of the genes individually. This would reduce the number of ground clauses by $10^3$ times. This perception allowed use to run the simple model on the unclustered data which had not been possible to run in the previous MTP due to enormity of data. And it turned out that besides cancelling out the noise in the data, clustering was also blurring some useful information in the data as we got increment in accuracy of gene expression level prediction by approximately 1 percent.We tried our model on halobacterium dataset but our joint model couldn't outperform the basic model in terms of gene activity prediction. We tried with various slight modifications of the joint model but it was not performing as was expected. In the joint model similarity between two genes was getting inferred if they had similar positive and negative links which were in turn getting inferred from activity of gene and TFs. We changed it from two to one level filtering. Now two genes were similar if they were active and inactive together. But this model didn't produce any better results.

To get more understanding of the problem at hand, we decided to create an artificial regulatory network and generate synthetic data from it. It was done because with it we could control numerous parameters like sparsity of the network, weight distribution, size of the network, rules from which data is to be generated etc. This would enable us to study the problem at hand more carefully and get useful insights in it.

We developed a module to generate a regulatory network and a module to generate the gene expression and TF expression from it. On this generated data we experimented with our model, basic model, and the improved basic model. Improved basic model is essentially basic model minus one rule( Rule 1). We will discuss more about improved model after few paragarphs. Gene activity prediction was high for basic model but link retrieval was poor. We attributed the poor performance on link retrieval to degeneracy in the model and changed the basic model by reducing one rule as explained in the previous paragraph. The improved basic model achieved a high gene activity prediction as well as high link retrieval. In this setting the improved basic model outperformed the joint model. It was expected for the improved basic model to outperform the joint model as for the data generation, rules of improved basic model were used. However the the performance of the joint model with respect to links prediction was unexpectedly poor. Hence we decided to experiment on the basic model itself to gain more understanding of the problem at hand.

We varied the average number of links per gene and observed that the area under the precision recall curve for link prediction decreases as we increase the average number of links per gene. We also observed that the precision increases as we increase average number of links per gene. It was expected because deepdive, our MLN learning and inference module, was generating some weight for every TF-gene link. So as the actual number of links increased, there was an increase in the number of links correctly recovered and hence we saw an increase in the precision. We also observed that as the number of links increased, recall of the model decreased. It is also intuitively explainable as follows: the less the number of determining factors are for gene prediction, the less is the degeneracy and equivalently speaking less are the number of ways in which the gene expression can be explained in all conditions. Lets understand this with an example. Say for a gene $g_i$ there are 4 TFs t1, t2, t3 and t4 which positively induce gene expression for gene $g_i$. If any one of the four becomes active in a condition and all inhibitively linked TFs are inactive, we will see an increase in probability that gene $g_i$ is active in that condition. So it will get more and more difficult for the algorithm to decipher all the links as the total number of links increases. Hence the recall decreases.

On synthetic data we had done our experiments with small number of TFs(10). In order to observe the effects on a scale similar to real world, we did the experiments with number of TFs same as in the Halobacterium case(190). In this setting too, we got the same inferences which we had got in the smaller setting namely that: precision increases with the increase in the average number of links per gene, recall decreases with the average number of links, and the area under precision recall curve decreased with the increase in the average number of links per gene.

While this artificial network module was getting created in MATLAB, we got another insight about the basic model. It has three rules. First rule is about activating link from TF to gene. Second is about the inhibitive link from TF to gene and third rule is prior of gene being inactive. The problem is that we don't have any semantic explanation of negative weights for the first two rules.We have three states for a link. It can be inductive, inhibitive or be not present. All this can be captured in a single rule by assigning semantic meaning to the positive weight, negative weight and zero weight. Note that besides causing the semantic handicap ,the two separate rules also create unnecessary redundancy in the model which may or may not reflect in the prediction accuracy of genes significantly (although we have reasons to believe that improved model should do better even with gene expression prediction accuracy which is described in subsequent paragraphs ) but will certainly reflect in the links retrieval precision and recall. Hence we created an improved basic model by eliminating one of the rules from the Knowledge base. We call it the improved basic model.
We also tested the improved model on the real Halobacterium data set using the same conditions as were present in case of the basic model. Improved basic model achieved 67.35 % accuracy in terms of prediction of gene expression where as the basic model had achieved 66.74% accuracy. Earlier we had seen significant improvement over basic model in link retrieval for artificial data and now we see an improvement in gene expression prediction over basic model in our improved basic model. We try to explain this cause of improvement below.
Assuming our semantic assumptions for the biological regulatory network to be correct, we find that basic model has got the freedom of having negative weights which violates the semantic assumptions. Putting it differently, basic model has the freedom to learn a network which is semantically different from what is intended to learn. We say this because we don't have a semantic explanation for negative weights for first two rules of basic model.Improved basic model, on the other hand, doesn't have that freedom as all three outcomes for weights namely positive weights, negative weights and zero weights have a distinct semantic explanation. Hence basic model can capture the

noise or any other unintended pattern in the data to some extent which will make its performance wrt gene expression level prediction and link retrieval poorer.

We conclude with one understanding that while dealing with MLN models, one should constrain the rules to fit exactly the semantic meaning intended. This reduces the size of parameter space there by increasing the effectiveness of gradient descent or for that matter, any other learning algorithm. Also it is better to work with simpler models as they require less learning data and are relatively easier to learn.

# Bibliography

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

[2] Lyle H.Ungar Andrew McCallum, Kamal Nigam. Efficient clustering of high-dimensional datasets with application to reference matching. *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178, 2000.

[3] Shannon P Facciotti M Hood L Baliga NS Thorsson V Bonneau R, Reiss DJ. The Inferelator: An algorithm for learning parsimonious regulatory networks from systems-biology datasets de novo. *Genome Biol.*, 7(5):R36, 2006.

[4] A.Doan F. Niu, C. R and J.W. Shavlik. Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. *PVLDB 11.*

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

[6] M. Richardson and P. Domingos. Markov Logic Networks. Machine Learning. *J. Comp. Phys.*, 62:107–136, 2006.

[7] Christopher De Sa Jaeho Shin Feiran Wang Sen Wu, Ce Zhang and C. R. Incremental knowledge base construction using Deepdive. *VLDB*, 2015.

[8] Alok Singhal. Subcellular regulatory networks learning, MTP thesis, June 2014.