RICH PROBABILISTIC MODELS FOR GENOMIC DATA

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Eran Segal
August 2004

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daphne Koller
Department of Computer Science
Stanford University
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Nir Friedman
School of Computer Science and Engineering
Hebrew University

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stuart Kim
Department of Developmental Biology
Stanford University

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

David Haussler
Department of Computer Science
University of California Santa Cruz

Approved for the University Committee on Graduate Studies.

To Gal.

Abstract

Genomic datasets, spanning many organisms and data types, are rapidly being produced, creating new opportunities for understanding the molecular mechanisms underlying human disease, and for studying complex biological processes on a global scale. Transforming these immense amounts of data into biological information is a challenging task. In this thesis, we address this challenge by presenting a statistical modeling language, based on Bayesian networks, for representing heterogeneous biological entities and modeling the mechanism by which they interact. We use statistical learning approaches in order to learn the details of these models (structure and parameters) automatically from raw genomic data. The biological insights are then derived directly from the learned model. We describe three applications of this framework to the study of gene regulation:

- Understanding the process by which DNA patterns (motifs) in the control regions of genes play a role in controlling their activity. Using only DNA sequence and gene expression data as input, these models recovered many of the known motifs in yeast and several known motif combinations in human.

- Finding regulatory modules and their actual regulator genes directly from gene expression data. Some of the predictions from this analysis were tested successfully in the wet-lab, suggesting regulatory roles for three previously uncharacterized proteins.

- Combining gene expression profiles from several organisms for a more robust prediction of gene function and regulatory pathways, and for studying the degree to which regulatory relationships have been conserved across evolution.

Acknowledgments

First and foremost I would like to thank my adviser, Daphne Koller, for her guidance and support throughout my years at Stanford. Of all the mentors I had, Daphne had the most profound impact on this thesis and on my academic development. I have learned tremendously from her brilliance, high standards and strive for excellence, including how to choose and pursue interesting research directions and how to present them both in papers and in talks. Of all the researchers I know, I cannot imagine a better choice than Daphne for an adviser. I feel priviliged to have had the opportunity to work closely and learn from her.

Excluding Daphne, Nir Friedman, my unofficial co-adviser, had the greatest influence on this thesis and on teaching me how to do research. Most importantly, Nir introduced me to and got me excited about the field of computational biology. Nir's contribution to the work presented in this thesis is fundamental, from the basic ideas of integrating heterogeneous genomic data into a single probabilistic model, through the development of the algorithms that learn these models automatically from data, and finally with ways to present, analyze, and visualize the results. I greatly enjoyed working with Nir and am indebted to him for his dedication and for the many research skills I learned from him.

I would also like to thank Stuart Kim for taking me into his lab and having the patience to teach me biology and the biologists' way of thinking. Through the interaction with Stuart and the members of his lab, I learned how to think critically about biological results and how to select convincing controls to perform.

I am very grateful to David Haussler, the fourth member of my reading committee, for insightful comments and motivating discussions throughout this past year. I also

thank Matt Scott for agreeing to chair my Orals committee and for a stimulating and enjoyable collaboration on the fifth chapter of this thesis.

The work in this thesis is the product of a collaboration with a number of researchers. In particular, I had the good fortune of collaborating with Aviv Regev, a truly phenomenal colleague. Aviv played a key role in the development of the module network framework presented in Chapter 4. This project would not have been possible without her creative ideas, diligence, and thoroughness. My collaboration with Aviv has not only been the most fruitful one, but also a very enjoyable one on the personal level and I look forward to our continued working together.

I would also like to thank David Botstein for his inspiring vision and for his belief in our methods. The collaboration with David resulted in the testing of our computational predictions in his lab and was a significant milestone in my Ph.d. I also thank Michael Shapira for carrying out these experiments.

I am grateful to every one of my other co-authors over the last several years. Audrey Gasch, the first biologist who was brave enough to work with us. Ben Taskar, with whom I worked on the first project of this research direction. Yoseph Barash, who played a major role in developing the ideas and infrastructure of Chapter 3 and Roman Yelensky, for his dedication on continuing the work on this project. Dana Pe'er, for her friendship and numerous insightful discussions and working sessions that lead to the development of the module network framework. It was a great pleasure to work with Josh Stuart on the multi-species project. The intense hours we spent together for over a year were productive, fun, and lead to a great friendship.

I would also like to thank other co-authors with whom I enjoyed working on projects that were not included in this thesis: Alexis Battle, Asa Ben-Hur, Doug Brutlag, Lise Getoor, Uri Lerner, Dirk Ormoneit, Saharon Rosset, Roded Sharan, and Haidong Wang.

As part of the research on this thesis and to facilitate our collaboration with biologists, we developed *GeneXPress*, a statistical analysis and visualization software. GeneXPress has been a critical component of all the projects presented in this thesis and I would like to thank the many students with whom I worked and who contributed to the development of GeneXPress: David Blackman, Jonathan Efrat, Todd Fojo,

Jared Jacobs, Amit Kaushal, William Lu, Tuan Pham, Daniel Salinas, Mark Tong, Paul Wilkins, and Roman Yelensky. Still on the GeneXPress token, thanks to Nir Friedman for the initial motivation and for the many useful feedbacks along the way. Thanks also to Aviv Regev, our number one user, whose many comments and feedback greatly helped in shaping the tool.

In the initial development of the code infrastructure for this thesis, I benefited greatly from the software infrastructure of PHROG (Probabilistic Reasoning and Optimizing Gadgets). I thank Lise Getoor, Uri Lerner and Ben Taskar for their work on PHROG and for sharing their code.

I am proud to have been a member of Daphne's research group. I was greatly motivated and inspired by the productivity of this group and enjoyed the stimulating discussions during our weekly meetings. The high quality and impact of the work produced by this group is a testament to the extraordinary members of the group and to Daphne's vision and advising skills. I would like to thank: Pieter Abbeel, Drago Angelov, Alexis Battle, Gal Chechik, Lise Getoor, Carlos Guestrin, Uri Lerner, Uri Nodelman, Dirk Ormoneit, Ron Parr, Christian Shelton, Ben Taskar, Simon Tong, David Vickrey, Haidong Wang, and Roman Yelensky.

I am grateful to Yossi Matias for taking me through my first steps of academic research and for his help in my application process and in encouraging me to come to Stanford. Thanks also to my friends and colleagues at Zapa, in particular Eyal Gever and Doron Gill, for their encouragement, help and support in my application process to Stanford.

A great deal of the writing part of this thesis was done in the Morning Due Cafe in San Francisco. The friendliness of the owners and customers and the large amounts of coffee I had there were critical for this writing.

Lots of thanks to the many friends who distracted me and kept me sane. The list is too long to fit in these lines, but I am grateful to my friends from back home in Israel, and to my new friends from Stanford, for their ongoing support and encouragement.

I am grateful to my parents, Rachel and Yoffi, for everything they taught me and for the values they have given me. Without them, I would not arrive at this point. I also thank my brother Amir, for being such a great friend, and my little brother

Tomer, for all the joy he brought to our family. I am proud of both of them.

Finally and above all, I am eternally grateful to my wife Keren for her unconditional love and support, for sticking with me from the beginning and throughout this long and at times very difficult journey, and most recently, for sharing with me the most important and most happy moment of my life, the birth of Shira.

Contents

List of Figures

Chapter 1

Introduction

The complex functions of a living cell are carried out through the concerted activity of many genes and gene products. This activity is coordinated through molecular networks involving interacting proteins, RNA, and DNA molecules. Discovering the structure of these networks, the conditions under which they operate, and the molecular mechanisms they employ is a major goal of molecular cell biology.

Recent technological advances have led to an explosion in the availability of data on a genome-wide scale, including complete genome sequences, measurements of mRNA levels of entire genomes under various conditions, and genome-wide measurements of protein-protein and protein-DNA interactions. Such heterogeneous data provide important information about cellular processes and their underlying molecular networks. However, transforming these immense amounts of data into *biological information* is a challenging task, and the key to success lies in our ability to combine theoretically-founded algorithms and techniques from computer science, statistics, and machine learning, with deep understanding of the biological domain. Biology is transitioning into an information science and as such, the forthcoming years will be crucial for laying out the foundations and methodologies that will be employed.

In this dissertation, we address this challenge by developing a statistical modeling language that can represent complex interactions in biological domains. The models are based on the language of *relational Bayesian networks* (Koller and Pfeffer, 1998, Friedman *et al.*, 1999a, Getoor, 2001), which represents a probability distribution

over various entities in a relational domain. In the biological domain, we typically model several classes of objects, such as genes, experiments, biological processes, and activity levels of genes. With each class, we also associate a set of observed attributes, such as the *sequence* of a gene or the *condition* of an experiment, and a set of hidden attributes, such as the *process* a gene participates in, whose value we wish to infer.

Such a modeling language has several advantages. First, it allows us to explicitly represent and reason about biological entities in a modular way, as well as model the mechanistic details of the underlying biological system. Second, we can build on the sound foundation of statistical learning methods to develop algorithms that learn the models directly from data. Finally, by designing models that directly match the biological phenomena we wish to study, we can use this general framework to solve a wide range of problems, and obtain biological insights directly from the model. In this thesis, we formulate a number of such models, develop algorithms to learn their structure automatically from the input genomic data and provide a systematic statistical and biological analysis of our results, including wet lab experiments for testing several of our novel computational predictions.

1.1 Biological Background

We begin with a brief overview of the basic concepts of molecular biology that are relevant to this thesis. The interested reader is referred to molecular biology textbooks (e.g., Alberts *et al.* (2002)) for more information.

Cells are the fundamental working units of every living system. The nucleus of all cells contains a large *DNA* (Deoxyribonucleic acid; see Figure 1.1) molecule, which carries the genetic instructions for making living organisms. A DNA molecule consists of two strands that wrap around each other to resemble a twisted ladder. The sides are made of sugar and phosphate molecules. The "rungs" are made of nitrogen-containing chemicals called *bases*. Each strand consists of a sequence of nucleotides, where each nucleotide is composed of one sugar molecule, one phosphate molecule, and a *base*. Four different bases are present in DNA — adenine (A), thymine (T), cytosine (C), and guanine (G). The particular order of the bases arranged along

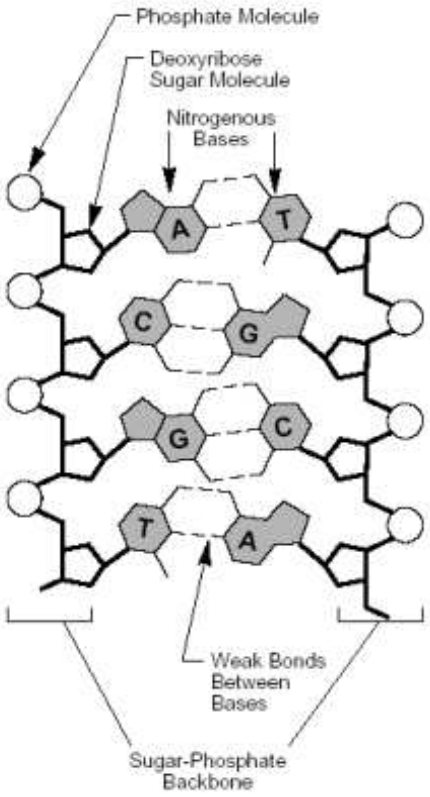


Figure 1.1: Illustration of a DNA molecule (image taken from http://blairgenealogy.com/dna/dna101.html).

the sugar-phosphate backbone is called the *DNA sequence*; the sequence specifies the exact genetic instructions required to create a particular organism with its own unique traits.

The two strands of the DNA molecule are held together by weak hydrogen bonds between opposite bases. The four bases pair in a particular manner: Adenine (A) pairs with thymine (T), while cytosine (C) pairs with guanine (G). These pairs of bases are known as *Base Pairs* (bp). The DNA is organized into separate long segments called *chromosomes*, where the number of chromosomes differs across organisms. For example, in humans there are 46 chromosomes or 23 pairs of chromosomes (each parent contributes 23 chromosomes).

The *DNA* molecule serves as the blueprint of an organism: It contains the instructions for the synthesis and regulation of *proteins* — large molecules composed of one or more chains of amino acids, that determine the shape and structure of the cell. The synthesis instructions and composition of a particular protein is coded on a segment of DNA called a *gene*. The *central dogma* of molecular biology states that information flows from DNA through RNA to protein (see Figure 1.2). Thus, a protein is synthesized from DNA in the following two-step process:

1. DNA → RNA: *Transcription* is the process by which a macromolecular machine called RNA polymerase produces a *messenger RNA* (mRNA) sequence of a gene using the DNA sequence as a template. The process by which genes are transcribed into the mRNA structures present and operating in the cell is termed *gene expression*.
2. RNA → Protein: In the subsequent process, called *translation*, another macromolecule called the ribosome serves as a protein factory and synthesizes the protein according the information coded in the mRNA.

A key observation is that, while each cell contains the same copy of the organism's DNA, different cells express different genes in different conditions. To control gene expression, specialized proteins called *transcription factors* recognize and bind specific sequences on the DNA, called *cis-regulatory sequences*. Through this binding process, transcription factors can enhance or inhibit transcription of nearby genes. The cis-regulatory sequences are typically short sequences, 5-25 DNA base pairs in length, and are typically located in regions upstream of the site on the DNA in which transcription of a gene starts. These upstream regions are termed *promoter regions*. An organism typically has hundreds of different transcription factors, each capable of recognizing different cis-regulatory sequences. We typically represent the cis-regulatory sequences recognized by a transcription factor using sequence *motifs*. Transcription factors often work in combination with other transcription factors to ensure that the required amount of each gene is being transcribed. We note that transcription factors are themselves proteins and are thus subject to transcriptional control. A 3D model of a transcription factor binding DNA is shown in Figure 1.3.

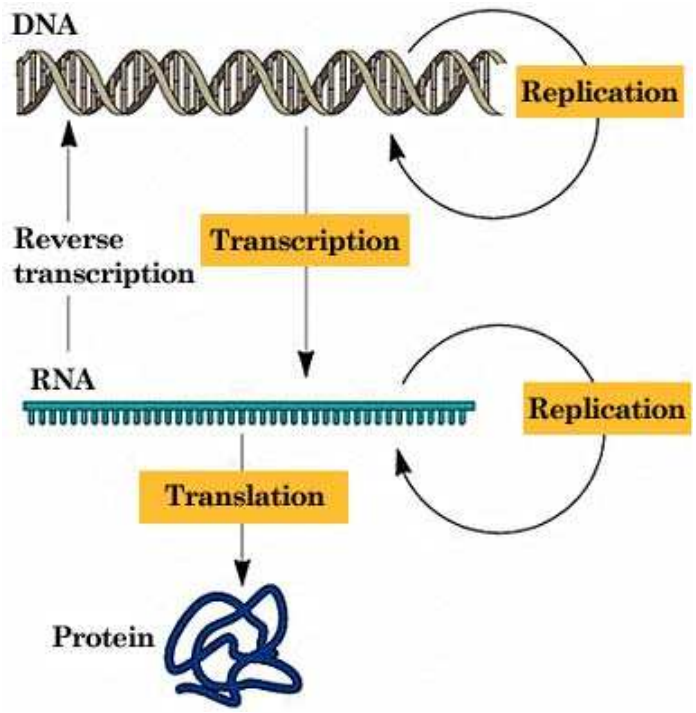


Figure 1.2: The central dogma of molecular biology: information flows from DNA through RNA in a process called transcription, and from RNA to proteins in a process called translation (image taken from http://cats.med.uvm.edu/cats_teachingmod/microbiology/courses/genomics/introduction/1_2_bgd_gen_pro_dog.html.

Transcription factors are by no means the only control over gene expression. Biological regulation is extremely diverse and involves different mechanisms at many layers: Before transcription occurs, another class of proteins regulate the structure of the DNA itself and determine whether specific regions in the DNA are even accessible to binding by transcription factors. Once the mRNA molecule is transcribed, other mechanisms regulate its editing and transport to the ribosome, thereby controlling whether or not the mRNA will be translated into its protein product. In addition, the total mRNA level of a given gene in the cell is regulated not only by transcription (creation) of its mRNA, but also by the degradation of its mRNA. Finally, regulation continues even after the protein is translated: a large part of biological regulation is

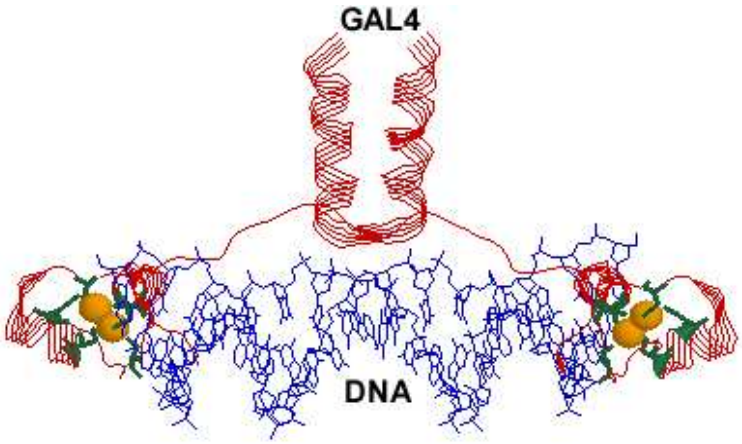


Figure 1.3: Three dimensional model of the zinc-finger transcription factor Gal4 binding DNA. Through this binding process, transcription factors can regulate the expression of nearby genes (image taken from http://www.web-books.com/MoBio/Free/Ch4F2.htm).

via *post-translational modifications* that alter the structure of the protein and determine its activity.

In recent years, technological breakthroughs in spotting hybridization probes and advances in genome sequencing led to the development of *DNA microarrays*, which consist of many types of probes, either oligonucleotides or complementary DNA (cDNA), that are immobilized in a predefined organization to a solid surface. By using DNA microarrays researchers can measure the abundance of thousands of mRNA targets simultaneously (DeRisi *et al.*, 1997, Lockhart *et al.*, 1996), providing us for the first time with a genome-scale view of gene expression.

Microarray technology is based on *DNA hybridization*: a process in which a DNA strand binds to its unique complementary strand. A set of known sequence probes are fixed to a surface and placed in interaction with a set of fluorescently tagged targets of various sequences. After hybridization, the fluorescently lit spots indicate the identity of the targets and the intensity of the fluorescence signal is in correlation to the quantitative amount of each target. Due to different hybridization affinities between clones and the fact that an unknown amount of complementary DNA (cDNA) is fixed for each probe, we cannot directly associate the hybridization level with a

quantitative amount of transcript. Instead, cDNA microarray experiments compare a reference pool and a target pool. Typically, green is used to label the reference pool, representing the baseline level of expression and red is used to label the target sample of interest (e.g., the target samples may be cells of different types or cells that were treated with some condition of interest). We hybridize the mixture of reference and target pools and read a green signal in cases where the expression level is reduced compared to the reference pool and a red signal in cases where the expression level is increased compared to the reference pool. When applying DNA microarray technology to measure mRNA levels, the mRNA are reverse-transcribed into cDNA which is then hybridized to the DNA sequence probes on the microarray. The reason that the mRNA levels are not measured directly is that the environment is full of RNA-digesting enzymes so free RNA is quickly degraded. In contrast, the DNA form is more stable and is therefore used for the hybridization. An example of a DNA microarray is shown in Figure 1.4. We note that there are several microarray technologies (e.g., the oligonucleotide arrays of Affymetrix) that differ by what is printed on the chips and the protocols employed for using them.

A genome wide measurement of transcription is called an *expression profile* and consists of measurements of the mRNA levels of genes in the cell. Biologically speaking, what we measure is how the *gene expression* of each gene changes to perform complex coordinated tasks in adaptation to a changing environment. We note that while DNA microarrays can measure changes in mRNA levels that are a direct consequence of transcriptional regulation, there are many important cellular aspects that are not observed by microarrays (e.g., protein expression and protein activity). Furthermore, due to biological variation and a multi-step experimental protocol, the data measured by microarrays are noisy, and may fluctuate significantly, even between repeated experiments.

In order to obtain a wide range of expression profiles, various perturbations (e.g., mutations (Hughes *et al.*, 2000) or heat shock (Gasch *et al.*, 2000)) are employed and different cell types are measured. The outcome is a matrix that associates an expression level with each gene (row) and each microarray (column). In our setting, this expression matrix may contain thousands of genes and hundreds of arrays. Our

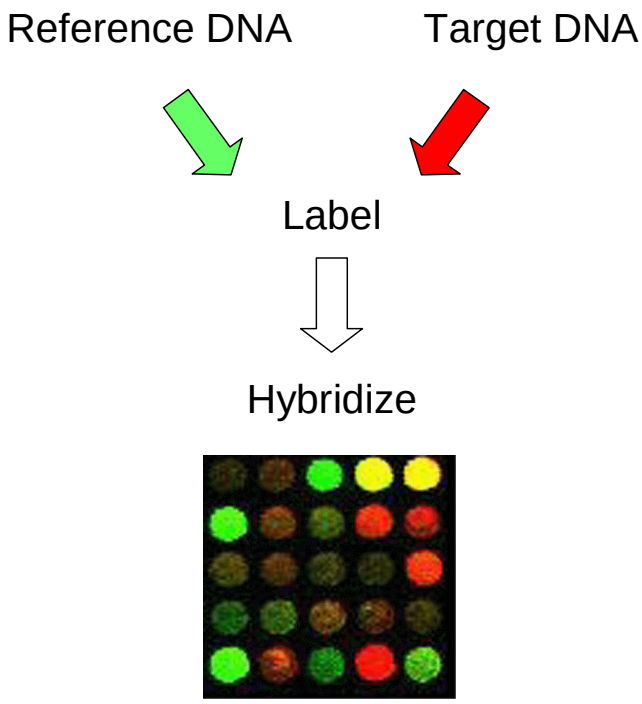


Figure 1.4: An image of a DNA microarray. Each spot represents a different gene

goal is to uncover molecular interactions, most notably regulation, from these data.

We note that other technologies have been developed for probing different attributes of the cell at a genome-wide scale and more technologies are underway. These include assays for measuring protein-protein interactions, protein complexes, protein abundance levels, and protein-DNA interactions. Similar to gene expression data, such measurements are useful for the study of gene regulation.

1.2 Our Approach

In this thesis we study the problem of gene regulation, including: detecting the DNA sequence motifs through which regulation occurs; discovering which regulatory proteins regulate which target genes; understanding how this regulation changes in response to different environmental conditions; and identifying which genes are regulated together, or what are the modules of regulation within the cell. While available

genomic datasets provide an informative view of the gene regulation process, they do not provide direct answers to the questions above. For this reason, computational tools for analyzing genomic data are very much needed.

As many human diseases are caused by defects in regulatory mechanisms, understanding gene regulation is very important. For example, if the mechanism responsible for stopping cells from dividing is not working properly, uncontrolled cell proliferation may occur and can lead to cancerous tumors. Thus, by unraveling the details of gene regulation, we might be able to better diagnose and better treat human diseases.

Several approaches for studying gene regulation have been suggested. One popular class of approaches starts with gene expression data measured via DNA microarrays, and first applies a *clustering* algorithm that attempts to cluster together genes that have similar patterns of expression across the given set of microarrays. As genes with similar patterns of expression may be regulated by the same mechanism, the next step is to search for common DNA sequence motifs in the promoter regions of genes that ended up in the same cluster. Common motifs that are found in the second step can then be suggested as candidate sequences through which regulation, of genes in the cluster where this motif was found, occurs. A schematic diagram of this approach is shown in Figure 1.5.

Note that this approach to gene regulation is procedural in nature: One method (clustering) is applied to the expression data, and a different method (motif discovery) is then applied to the sequence data, using the output of the first method as its input. Thus, the clustering and the motif discovery tasks are decoupled in this approach, and are often developed by different research groups. As neither of these tasks can be solved optimally, this decoupling leads to several serious limitations. For example, a mistake in the clustering algorithm that places co-regulated genes in different clusters will make it difficult for even the best motif discovery tool to detect the underlying sequence motifs. Nevertheless, to date, procedural architectures are perhaps the most popular class of approaches, and provide the basis for many of the methods applied to genomic datasets.

In contrast to procedural methods, our approach is *model-based*. By a model we mean a simplified description of the biological process that could have generated the

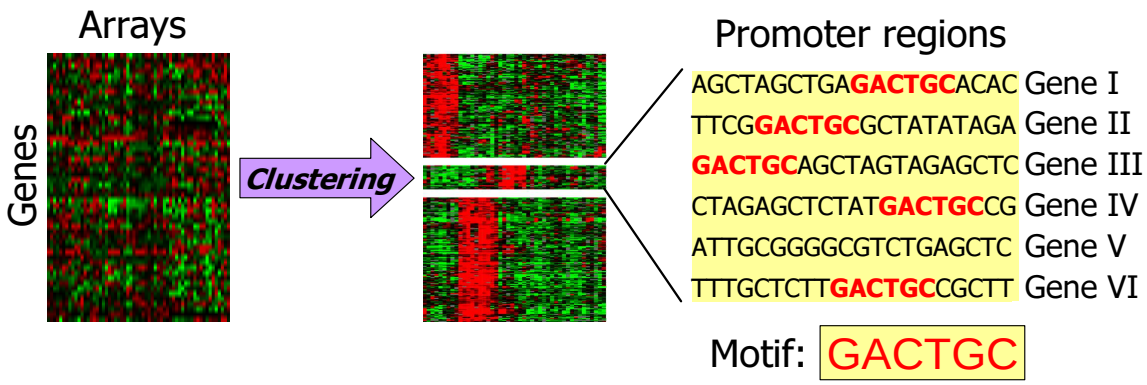


Figure 1.5: Schematic diagram of a popular procedural approach to gene regulation. In this approach, gene expression data (left) is first clustered, such that genes with similar patterns of expression are placed next to each other (middle). The promoter regions of genes in the same cluster are then searched for common DNA sequence motifs (right). As genes in the same cluster have similar mRNA expression profiles, common motifs (e.g., the red GACTGC sequence in this example) can be suggested as potential candidate sequences involved in regulating the genes in the cluster.

observed data. Due to the stochastic nature of the biological system and due to the noise in the technology for measuring its properties, our model must be able to handle uncertainty in a principled way. Recently there has been significant advances in modeling frameworks, most notably Bayesian networks (Pearl, 1988), that can handle and represent such uncertainty. Thus, we base our models on Bayesian networks and more specifically on Probabilistic Relational Models (PRMs) (Koller and Pfeffer, 1998, Friedman *et al.*, 1999a, Getoor, 2001), which extend the flat Bayesian network representation to relational domains.

Such model based approaches have several advantages. First, they allows us to explicitly represent and reason about biological entities in a modular way, as well as model the mechanistic details of the underlying biological system. Second, we can build on the sound foundation of statistical learning methods to develop algorithms that learn the models directly from data. Finally, by designing each model separately, depending on the biological phenomena we wish to study, we can use this general framework to solve a wide range of problems, and obtain biological insights directly from the model.

Based on the PRM framework, this thesis presents a statistical modeling language that we developed for representing complex interactions in biological domains, and several applications of this framework for studying key problems in gene regulation. Our modeling language can represent different classes of objects, such as genes, arrays, and expression. With each object, we can then associate a set of observed properties, such as the DNA promoter sequence of a gene, or the condition under which a given array was measured. In addition, we can also associate objects with hidden properties, such as the functional unit, module, or biological process that to which a gene belongs. Finally, the language also represents probabilistic dependencies between the properties of different objects. For example, the expression of a gene in a particular microarray experiment can depend probabilistically on the sequence of the gene and the condition of the array. A schematic diagram of our modeling language for an example model of gene regulation is shown in Figure 1.6.

Our models always define a joint probability distribution over all the properties in the domain. Each property (hidden and observed) of every object is represented by a random variable. Together, these random variables and the probabilistic dependencies among them define a Bayesian network. As a Bayesian network defines a joint probability distribution over the space represented by its random variables, we use the Bayesian network induced by our models to represent the joint distribution defined by our models in a compact way. Using the Bayesian network representation, this joint distribution can be factored as a product of terms, to which each random variable in the Bayesian network contributes exactly one factor.

We learn the details (dependency structure and parameters) of the models automatically from the data, where the goal is to find a model that maximizes the probability of the model given the observed data. This learning task is challenging, as it involves several steps, some of which are computationally intractable. One step involves learning the form and details of the various local probability distributions associated with each random variable. Another step is to learn the dependency structure among the random variables of the various objects we have. As we typically have thousands of such variables, and as the number of possible network structures may be exponential in the number of these variables, this learning problem is often

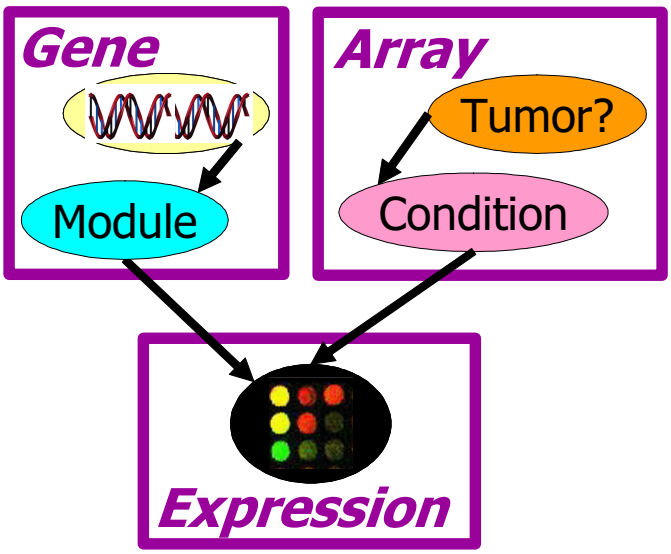


Figure 1.6: Schematic diagram of our statistical modeling language for an example model of gene regulation. The language can represent classes of objects (Gene, Array, and Expression). Objects can have observed properties (DNA sequence for Gene, Condition and Tumor for Array, and mRNA levels for Expression) and hidden properties (Module for Gene). In addition, probabilistic dependencies between the properties of different objects can be represented (e.g., the mRNA level of Expression objects depends probabilistically on the module of Gene objects, and the condition of Array objects).

intractable. Finally, the models we consider have many hidden properties whose values we need to infer. For this task, we need to perform probabilistic inference in the underlying Bayesian network. As the network may contain thousands of highly dependent hidden variable, this inference task is often intractable as well.

Although the learning task is intractable, in some cases we can identify biological structure specific to the problem we are studying, and develop learning algorithms that exploit this structure, leading to efficient and scalable algorithms. From a technical perspective, identifying these structures and representing them within our model is the major modeling challenge, and exploiting these structures in the learning algorithms is the major computational challenge we had to face.

The general scheme of our approach in this thesis is as follows. We start with

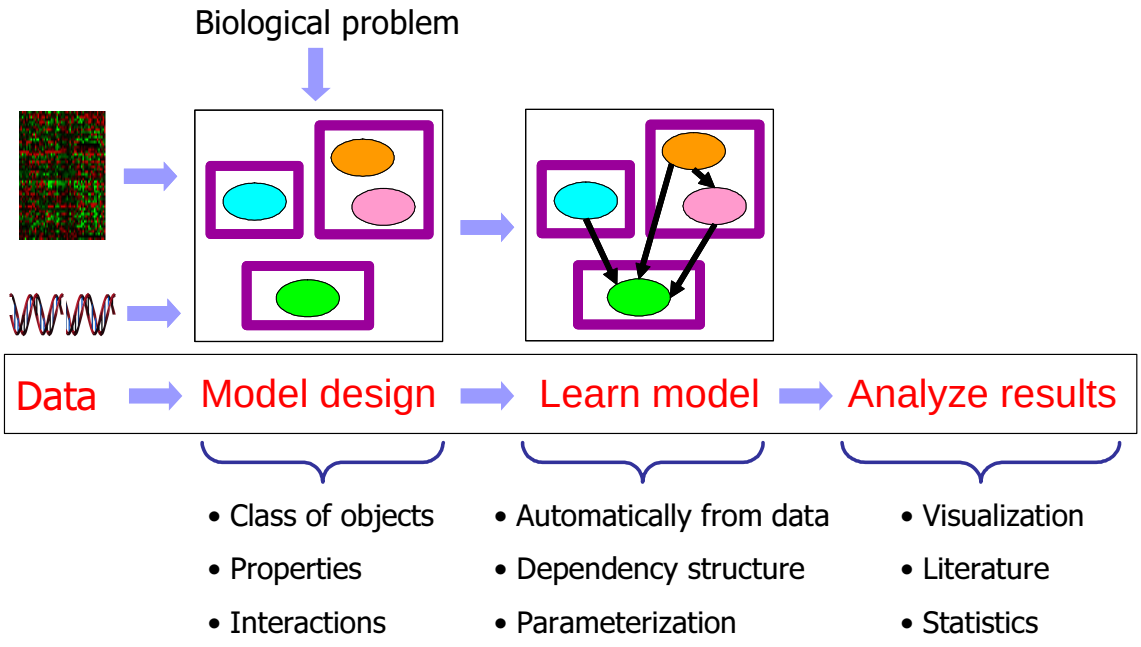


Figure 1.7: Overview of the general scheme of our approach.

genomic data and a biological problem we wish to solve. Next, we design an appropriate model that approximates the corresponding biological process, trading off the accuracy and level of detail of the model with the type and nature of the available data. In the following step, we develop algorithms that learn the details of the model automatically from the observed data, and apply them to the input data. Finally, we derive the biological insights and results directly from the learned models and evaluate them using statistics and the biological literature. An illustration of our general scheme is shown in Figure 1.7.

1.3 Contributions

In this thesis, our contributions include:

- A statistical modeling language, based on Probabilistic Relational Models, for representing complex interactions in biological domains. We show how our language can integrate heterogeneous types of genomic data in a joint probabilistic

model, including gene expression, DNA sequence, and protein-DNA binding data.

- A unified model over gene expression and DNA sequence data, aimed at providing a genome-wide *explanation* of the observed expression patterns as a function of combinations of DNA sequence motifs. We present an algorithm for learning the details of this model automatically from raw expression profiles and sequence data, including *ab-initio* discovery of the sequence motifs involved, their combinations, and the sets of genes they regulate.

- *Module Networks*, a new probabilistic framework for discovering regulatory modules using only gene expression data. Our procedure identifies modules of co-regulated genes, their regulators, and the conditions under which regulation occurs, generating testable hypotheses in the form "regulator 'X' regulates module 'Y' under conditions 'W'". We also present wet lab experiments supporting three of our novel computational predictions, allowing us to suggest regulatory roles for three previously uncharacterized proteins in yeast. This demonstrates one of the first approaches that completes an entire discovery cycle that starts from publicly available data, then applies a computational method to analyze the data, generates testable hypotheses, and finally tests these hypotheses in the wet lab, ending with new insights into the biological system.

- A model that combines gene expression profiles from several organisms for improved discovery of regulatory modules and for detection of regulatory modules that have been conserved across evolution. We present an application of this model to expression profiles from human and mouse brain tumors, and show that the combined model learns better regulatory modules compared to models that use the data from each organism in isolation.

- New model evaluation methods. As our methods are aimed at knowledge discovery in the biological domain where very little is known, it is often hard to evaluate their performance. We present new types of statistical evaluations and

new types of formal evaluations relative to the biological literature that objectively report on our performance and can easily be adapted for evaluating the results of other approaches.

- *GeneXPress*, a software environment for visualization and statistical analysis of genomic data, including gene expression, sequence data, and protein-protein interactions. For computational tools to have a broad impact, they must be accompanied by visualization and browsing tools that are easily accessible to biologists. To support this effort we developed GeneXPress, which can visualize the output of our algorithms. Many of the figures in this thesis were generated directly from this tool. Currently, GeneXPress has been downloaded by over 800 researchers from more than 50 countries.

1.4 Outline

The outline of this thesis is as follows. In Chapter 2 we present the basics of the probabilistic framework that underlie our models. The framework is based on the language of probabilistic relational models (PRMs), as introduced by Koller and Pfeffer (1998) and Getoor (2001). This chapter is intended to give an overview of the basic probabilistic framework.

In Chapter 3, we describe a probabilistic model for understanding transcriptional regulation using both gene expression and promoter sequence data. We aim to identify *cis-regulatory modules* — sets of genes that are co-regulated in a set of microarray experiments, through a common combination of motifs, which we term a *motif profile*. That is, given a gene expression data set as input, and promoter sequences of genes, we aim to identify modules of co-regulated genes and *explain* the observed expression patterns of each module via a motif profile that is common to genes in the same module. Our goal is to provide a genome-wide explanation of the expression data. This chapter is based on the work of Segal *et al.* (2002) and Segal *et al.* (2003e).

In Chapter 4, we attempt to reveal another aspect of the gene regulation process, by presenting *module networks*, a probabilistic model for discovering regulatory

modules from gene expression data. Our procedure identifies modules of co-regulated genes, their regulators, and the conditions under which regulation occurs, generating testable hypotheses in the form "regulator 'X' regulates module 'Y' under conditions 'W'". We demonstrate the ability of the method to identify functionally coherent modules and their correct regulators. However, as knowledge in the biological literature is limited, some hypotheses can neither be confirmed nor refuted. Thus, we have carried out microarray experiments in the wet lab to test the validity of three of these novel hypotheses. We show how these tests support our computational predictions, allowing us to suggest novel regulatory roles for three previously uncharacterized proteins in yeast. This chapter is based on the work of Segal *et al.* (2003c) and Segal *et al.* (2003b).

In Chapter 5 we present an extension to the module network framework of Chapter 4 that combines expression data from multiple organisms for the task of discovering regulatory modules that have been conserved across evolution. As in module networks, our procedure identifies modules of co-regulated genes, their regulators, and the conditions under which this regulation occurs. The key difference is that the regulatory modules in our extended procedure are jointly learned from expression profiles of several organisms. As we show in an application of the method to expression measurements of brain tumors from human and mouse, this joint learning allows us to improve our regulatory predictions compared to models learned from each organism's data in isolation. Moreover, we show that by combining expression data from human and mouse, we can gain insights into the evolution of the regulatory relationships in brain tumors between these organisms.

Finally, Chapter 6 contains concluding remarks as well as discussion of some of the aspects that are missing in our framework and presents the future challenges that need to be addressed.

Chapter 2

Probabilistic Framework

In this chapter, we present the basics of the probabilistic framework that underlie our models. The framework is based on the language of probabilistic relational models (PRMs), as introduced by Koller and Pfeffer (1998) and Getoor (2001), although we have extended the language to be applicable to the biological domain. This chapter is intended to give an overview of the basic probabilistic framework. Our extensions to the framework, which allow us to model interactions in the biological domain, are presented in subsequent chapters.

The probabilistic models that we study exploit conditional independence relationships that hold between properties in the biological system, and we thus start with a discussion of conditional independence. Then, since Bayesian networks provide the foundation for PRMs, we give a brief overview of them, before introducing probabilistic relational models.

2.1 Conditional Independence

Consider a simple biological system with only two genes, an activator gene and a target gene, that may or may not be under stress conditions. Formally, we can represent this system with three attributes, where the value domain of each attribute is shown in parentheses: *Activator* (*low*, *basal*, *high*), *Target* (*low*, *basal*, *high*), and *Stress* (*false*, *true*), where the *Activator* and *Target* attributes correspond to the levels

of expression of the activator and target gene, respectively. Note that for simplicity we assumed here that each gene is either over expressed (*high*), under expressed (*low*), or is expressed at some baseline expression level (*basal*). In practice the expression measurements that come from the microarray data are real valued measurements and we do not usually discretize them. As shorthand, we will use the first letter in each of the names to denote the attribute, using capital letters for the attributes and lower case letters for the particular values of the attributes. We use $P(A)$ to denote a probability distribution over the possible values of attribute A, and $P(a)$ to denote the probability of the event $A = a$.

Assume that the joint distribution of attribute values in measurements of our biological system is as shown in Figure 2.1(a). Using this joint distribution, we can compute the probability of any instantiation of A, T, and S, $P(a, t, s)$. However, to explicitly represent the joint distribution we need to store 18 numbers, one for each possible combination of values for the attributes. (In fact, we can get away with 17 numbers because we know that the entries in the joint distribution must sum to 1)

In many cases, however, our data will exhibit a certain structure that allows us to (perhaps approximately) represent the distribution using a much more compact form. The intuition is that some of the correlations between attributes might be indirect ones, mediated by other attributes. For example, the effect of whether the cell is under stress or not on the expression level of the target gene might be mediated by the expression level of the activator: if the activator is present, regardless of whether the cell is under stress or not, it will bind the DNA of the target gene and increase its expression. Thus, the dominant factor influencing the expression of the target gene is the expression of its activator and not the condition of the cell. This assertion is formalized by the statement that *Target* is *conditionally independent* of *Stress* given *Activator*, i.e., for every combination of values t, a, and s, we have that:

$$P(T = t \mid A = a, S = s) = P(T = t \mid A = a)$$

This assumption holds for the distribution of Figure 2.1.

S	A	T	P(S,A,T)
f	l	l	0.049
f	l	b	0.014
f	l	h	0.007
f	b	l	0.028
f	b	b	0.098
f	b	h	0.014
f	h	l	0.049
f	h	b	0.049
f	h	h	0.392
t	l	l	0.021
t	l	b	0.006
t	l	h	0.003
t	b	l	0.048
t	b	b	0.168
t	b	h	0.024
t	h	l	0.003
t	h	b	0.003
t	h	h	0.024

(a)

S	P(S)
f	0.7
t	0.3

S	A	P(A | S)
f	l	0.1
f	b	0.2
f	h	0.7
t	l	0.1
t	b	0.8
t	h	0.1

A	T	P(T | A)
l	l	0.7
l	b	0.2
l	h	0.1
b	l	0.2
b	b	0.7
b	h	0.1
h	l	0.1
h	b	0.1
h	h	0.8

(b)

S	P(S)
f	0.7
t	0.3

A	P(A)
l	0.1
b	0.38
h	0.52

T	P(T)
l	0.198
b	0.338
h	0.464

(c)

Figure 2.1: (a) The joint probability distribution for a simple example. (b) A representation of the joint distribution that exploits conditional independence. (c) The single-attribute probabilities.

The conditional independence assumption allows us to represent the joint distribution more compactly in a factored form. Rather than representing $P(S, A, T)$, we will represent: the marginal distribution over *Stress* — $P(S)$; a conditional distribution of *Activator* given *Stress* — $P(A \mid S)$; and a conditional distribution of *Target* given *Activator* — $P(T \mid A)$. It is easy to verify that this representation contains all of the information in the original joint distribution, *if the conditional independence*

assumption holds:

$$P(S, A, T) = P(S)P(A \mid S)P(T \mid A, S) = P(S)P(A \mid S)P(T \mid A)$$

where the last equality follows from the conditional independence of T and S given A. In our example, the joint distribution can be represented using the three tables shown in Figure 2.1(b). It is easy to verify that they do encode precisely the same joint distribution as in Figure 2.1(a).

The number of independent parameters for the factored representation is now $1 + 4 + 6 = 11$, as compared to the 17 we had in the full joint distribution. While the savings in this case may not seem particularly impressive, the savings grow exponentially as the number of attributes increases, as long as the number of direct dependencies remains small.

Note that the conditional independence assumption is very different from assuming complete attribute independence. For example, the marginal distributions for the three attributes are shown in Figure 2.1(c). It is easy to see that the joint distribution that we would obtain from this strong attribute independence assumption in this case is very different from the true underlying joint distribution. It is also important to note that our conditional independence assumption is compatible with the strong correlation that exists between $Target$ and $Stress$ in this distribution. Thus, conditional independence is a much weaker and more flexible assumption than standard (marginal) independence.

Context Specific Independence

As described above, we can exploit conditional independence to more compactly represent distributions and thereby considerably reduce the total number of parameters required. In biological domains, which typically contain a large number of attributes, conditional independence can be crucial. However, often the conditional independencies that we encounter are more subtle, and they hold under certain contexts and not in general.

For example, assume that our simple biological system described above also contains a repressor gene, in addition to the activator. Furthermore, assume that the repressor requires the presence of the activator in order to bind the DNA of the target gene, and when it does so, it blocks (represses) the expression of the target gene. In this case, the expression of the target gene thus depends on the expression of both the activator and the repressor and there are no apparent conditional independence relationships that hold. However, since the repressor requires the presence of the activator in order to bind the DNA of the target gene, it implies that if the activator is not present, then the repressor has no effect on the target gene. Thus, in the absence of the activator, the target and the repressor are conditionally independent, although they are not conditionally independent in the presence of the activator. Such a conditional independence relationship is termed *context specific independence*. In our example, denoting the expression level of the repressor with the attribute *Repressor*, the context specific independence implies that:

$$P(\mathit{Target} \mid \mathit{Repressor}, \mathit{Activator} = \mathit{low}) = P(\mathit{Target} \mid \mathit{Activator} = \mathit{low}).$$

That is, given that the activator is not expressed, the repressor exerts no effect on the expression of the *Target* gene.

In the biological domain, it is well known that many relationships hold only under certain contexts. For example, even though the expression of a gene depends on many influencing factors, it is very likely that it does not depend on all factors in all contexts. Thus, the ability to represent and exploit context specific independence relationships is crucial in order to achieve realistic models of biological processes. In fact, in many cases, discovering which factors play a role in which biological contexts, i.e., identifying the context specific independence relationships themselves, is a major part of our goal. We consider several different representations of context specific independence throughout this thesis.

2.2 Bayesian Networks

Bayesian networks (Pearl, 1988) (BNs) are compact graphical representations for high-dimensional joint distributions. They exploit the underlying conditional independencies in the domain — the fact that only a few aspects of the domain affect each other directly. We define our probability space as the set of possible assignments to the set of random variables $A_1, \ldots, A_n$. BNs can compactly represent a joint distribution over $A_1, \ldots, A_n$ by utilizing a structure that captures conditional independencies among attributes, thereby taking advantage of the "locality" of probabilistic influences.

A Bayesian network $\mathcal{B}$ consists of two components. The first component, $\mathcal{G}$, is a directed acyclic graph whose nodes correspond to the attributes $A_1, \ldots, A_n$. The edges in the graph denote a direct dependence of an attribute A_i on its parents $\mathbf{Pa}_{A_i}$. The graphical structure encodes a set of conditional independence assumptions: each node A_i is conditionally independent of its non-descendants given its parents.

Figure 2.2 shows a simple Bayesian network for the simple biological system described above. In this case, there are four attributes: *Stress*, *Activator*, *Repressor*, and *Target*. We see, for example, that the expression of the target gene depends on whether the cell is under stress or not only via the expression of the activator and repressor genes. Thus, *Target* is conditionally independent of *Stress* given *Activator* and *Repressor*.

The second component of a BN describes the statistical relationship between each node and its parents. It consists of a *conditional probability distribution (CPD)*, $P_{\mathcal{B}}(A_i \mid \mathbf{Pa}_{A_i})$ for each attribute, which specifies the distribution over the values of A_i given any possible assignment of values to its parents. Let $Val(A_i)$ denote the space of possible values for A_i and $Val(\mathbf{Pa}_{A_i})$ denote the space of possible values for the parents of A_i. In a CPD, all of the conditional probabilities are positive, and for any particular instantiation of A_i, $a_i \in Val(A_i)$, the sum over all possible instantiations of A_i, for any particular instantiation, $\boldsymbol{u}$, of A_i's parents is 1, i.e.,

$$\sum_{a_i \in Val(A_i)} P_{\mathcal{B}}(A_i = a_i \mid \mathbf{Pa}_{A_i} = \boldsymbol{u}) = 1$$

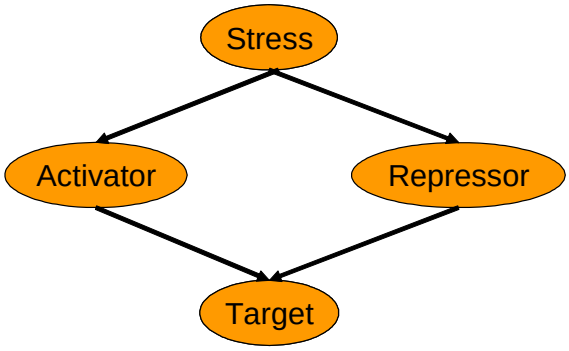


Figure 2.2: A Bayesian network for a simple biological system with three genes: an *Activator*, a *Repressor*, and a *Target*, where the system can either be under stress or not. Note that this network encodes the independence assumption that given the expression of the activator and the repressor, the expression of the target gene does not depend on whether the system is under stress or not.

In the biological domain we also have many real valued measurements which we may wish to model using continuous valued attributes. In this case, we associate a density function over A_i for each instantiation of A_i's parents, and require that it is a non-negative integrable function such that its integral over the space of possible values that A_i can take, is 1, i.e.,

$$\int_{a_i \in Val(A_i)} P_{\mathcal{B}}(A_i = a_i \mid \mathbf{Pa}_{A_i} = \boldsymbol{u}) da_i = 1$$

where $P_{\mathcal{B}}(A_i \mid \mathbf{Pa}_{A_i} = \boldsymbol{u})$ denotes the density function over A_i given that $\mathbf{Pa}_{A_i} = \boldsymbol{u}$.

The conditional independence assumptions associated with the BN $\mathcal{B}$, together with the CPDs associated with the nodes, uniquely determine a joint probability distribution over the attributes via the *chain rule*:

$$P_{\mathcal{B}}(A_1, \ldots, A_n) = \prod_{i=1}^{n} P_{\mathcal{B}}(A_i \mid \mathbf{Pa}_{A_i}). \tag{2.1}$$

Thus, from our compact model, we can recover the joint distribution; we do not need to represent it explicitly.

2.2.1 Conditional Probability Distributions

As discussed above, CPDs may be represented in a number of ways. Obviously, different CPDs are needed for different types (discrete or continuous) of attributes and different types of attribute parents modeled. However, even within the same configuration of attribute and attribute parent types, there is a range of different CPDs. This representation choice is critical as it specifies the nature of the dependency of an attribute on its parents as well as the number of parameters needed to represent this dependency.

In particular, variables in the biological domain may depend on a large number of influencing factors but in a structured way, and in many cases representing and learning this structure is a major part of our overall task. For example, the expression of a target gene may be regulated by a large number of activator and repressor genes, but in a given biological condition, only a small number of these regulators may be active. Such a target gene is thus regulated in a condition-specific manner, and identifying which regulators control its expression in which conditions is an important part of the overall discovery task. As we shall see, such context-specific dependencies can be explicitly modeled by the appropriate choice of CPD representation.

We now describe the different types of CPDs that we use throughout this thesis. We note that this is by no means an exhaustive overview of CPD types, and that many other CPDs have been explored in the literature.

Table CPDs

For attributes that depend only on discrete parent attributes, the most general CPD representation is a *table CPD*. This CPD specifies a probability distribution over the possible values of A (or density function if A is continuous) for each instantiation of A's parents. Note that the number of parameters needed to describe this CPD is equal to the number of joint assignments to the parents of A, $\mathbf{Pa}_A$. This number grows exponentially with the number of parents. Thus, even if we have 5 binary parents of a binary variable A, we need to specify $2^5 = 32$ values; if we have 10 parents, we need to specify $2^{10} = 1024$ values. Clearly, the tabular representation rapidly becomes large

and unsuited for modeling the complex dependencies we encounter in the biological domain.

Tree CPDs

As discussed above, even if a variable depends directly on a large number of parent variables, this dependency may be structured in a context-specific manner, such that the distribution over the values of A does not depend on the value of some subset of A's parents given the value of the other parents. A natural way to represent such context specific independence is to use a *tree* (Boutilier *et al.*, 1996), where the interior vertices represent splits on the value of some parent attribute of A, and the leaves contain distributions over the values of A (or density function if A is continuous). In this representation, we find the conditional distribution over A given a particular choice of values $\mathbf{Pa}_A[1] = \mathbf{pa}_A[1], \ldots, \mathbf{Pa}_A[k] = \mathbf{pa}_A[k]$ for its parent attributes by following the appropriate path in the tree down to a leaf: When we encounter a split on some attribute $\mathbf{Pa}_A[i]$, we go down the branch corresponding to the value of $\mathbf{pa}_A[i]$; we then use the distribution stored at the leaf.

For example, for the simple biological system with context specific independence described above, the tree CPD for the *Target* attribute given the *Activator* and *Repressor* attributes that represents the joint distribution of Figure 2.3 is shown in Figure 2.4. Note that even though this tree CPD has only 6 parameters compared to the 8 parameters in the corresponding table CPD, the two representations define the same conditional probability distribution over *Target* given *Activator* and *Repressor*. Thus, using the tree CPD we can explicitly represent the context specific independencies that hold and reduce the total number of required parameters. Again, even though the savings in this case do not seem significant, they grow exponentially as the number of attributes increases, as long as the number of distinct contexts remains small.

When the parent attributes, $\mathbf{Pa}_A$, are continuous, we use a *regression tree* (Breiman *et al.*, 1984). A regression tree is identical to the tree described above except that the tests on each interior nodes are of the form $\mathbf{Pa}_A[i] < \mathbf{pa}_A[i]$, where $\mathbf{pa}_A[i] \in I\!R$.

A	R	T	P(T | A,R)
l	l	l	0.7
l	l	b	0.2
l	l	h	0.1
l	b	l	0.7
l	b	b	0.2
l	b	h	0.1
l	h	l	0.7
l	h	b	0.2
l	h	h	0.1
b	l	l	0.2
b	l	b	0.7
b	l	h	0.1
b	b	l	0.3
b	b	b	0.4
b	b	h	0.3
b	h	l	0.85
b	h	b	0.1
b	h	h	0.05
h	l	l	0.1
h	l	b	0.1
h	l	h	0.8
h	b	l	0.4
h	b	b	0.4
h	b	h	0.2
h	h	l	0.9
h	h	b	0.06
h	h	h	0.04

Figure 2.3: A simple example of a conditional distribution for the *Target* attribute given the *Activator* and *Repressor* attributes.

Softmax CPDs

In some cases, the dependence of a discrete attribute A on its parent attributes, $\mathbf{Pa}_A$, can be approximated as a linear threshold function. In this case, each parent attribute contributes differently to A taking on each of its values, and the value of A is then determined based on the sum of the contributions of each of A's parents to each of its values.

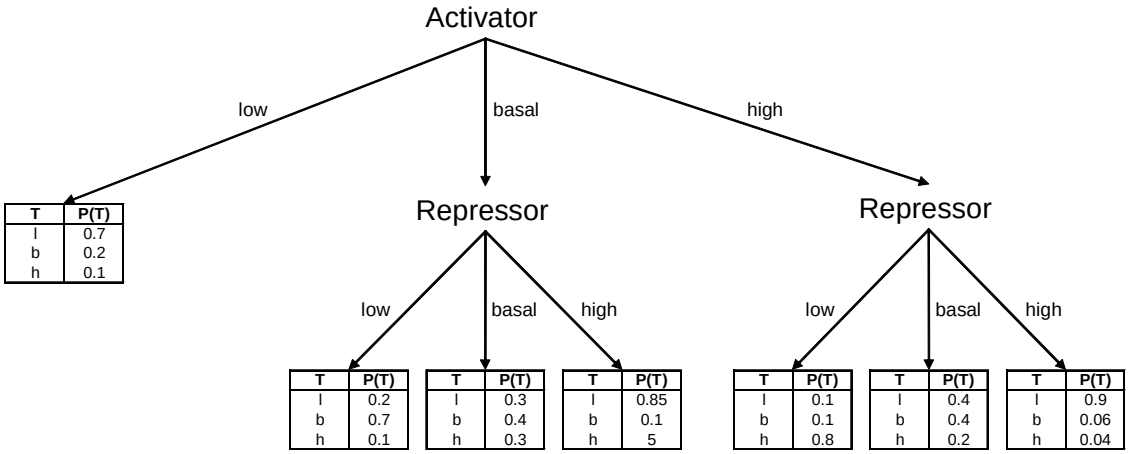


Figure 2.4: A tree CPD for representing the conditional distribution of the *Target* attribute given the *Activator* and *Repressor* attributes shown in Figure 2.3. Note that this tree CPD reduces the number of parameters as compared to a full table CPD, by exploiting context specific conditional independencies (e.g., the fact that $P(\mathit{Target} \mid \mathit{Repressor}, \mathit{Activator} = \mathit{low}) = P(\mathit{Target} \mid \mathit{Activator} = \mathit{low})$.

As a motivating example, assume that a target gene can be in several modes of regulation, where each potential regulator of the target gene (e.g., a transcription factor can be a potential regulator of the target gene if the upstream region of the target gene contains the DNA binding site that the transcription factor binds to) influences the regulation mode that the target gene will be in. If we make the assumption that each potential regulator exerts its influence on the regulation mode of the target gene independently of which other potential regulators are exerting their influence, and that these independent influences are additive, then we can model the regulation mode of the target gene as a linear threshold function. Thus, for binary parent attributes $\mathbf{Pa}_A$, we assume that the effect of $\mathbf{Pa}_A$ on A taking on the value $a \in \mathit{Val}(A)$ can be summarized via a linear function:

$$f_a(\mathbf{Pa}_A[1] = \mathbf{pa}_A[1], \ldots, \mathbf{Pa}_A[k] = \mathbf{pa}_A[k]) = \sum_{i=1}^{k} w_{ai}\mathbf{pa}_A[i],$$

where each $\mathbf{pa}_A[i] \in \{0, 1\}$. In our example, $\mathbf{Pa}_A[i]$ takes on the value 1 if regulator

$\mathbf{Pa}_A[i]$ can potentially regulate the target gene A, and each weight w_{ai} represents the contribution of regulator $\mathbf{Pa}_A[i]$ to the target gene A being in regulation mode a.

The next question is how the probability of A depends on $f_a(\mathbf{Pa}_A[1], \ldots, \mathbf{Pa}_A[k])$. A common choice in this case is the *softmax* distribution, which is the standard extension of the binary logistic conditional distribution to the multi-class case:

$$P(A = a \mid \mathbf{Pa}_A[1] = \mathbf{pa}_A[1], \ldots, \mathbf{Pa}_A[k] = \mathbf{pa}_A[k]) = \frac{\exp\{f_a(\mathbf{Pa}_A[1] = \mathbf{pa}_A[1], \ldots, \mathbf{Pa}_A[k] = \mathbf{pa}_A[k])\}}{\sum_{a' \in Val(A)} \exp\{f_{a'}(\mathbf{Pa}_A[1] = \mathbf{pa}_A[1], \ldots, \mathbf{Pa}_A[k] = \mathbf{pa}_A[k])\}}.$$

where each $\mathbf{pa}_A[i] \in \{0, 1\}$. In some cases, we might put additional constraints on the weights of the softmax distribution. For instance, in our gene regulation example, we may require that the weight matrix be sparse, so that each regulator can only affect the assignment of the target gene for a small number of regulation modes.

Gaussian CPDs

As discussed above, in the biological domain we often encounter real valued measurements that we wish to represent using continuous valued attributes. With such attributes, we associate a density function. A common choice for the density function over a real valued attribute A is the *Gaussian* distribution parameterized by a mean μ and variance σ^2. The attribute A is then said to have a Gaussian distribution with mean μ and variance σ^2, denoted $A \sim N(\mu; \sigma^2)$, if it has the density function:

$$P(A = a) = \frac{1}{\sqrt{2\pi}\sigma} exp\left\{\frac{(a-\mu)^2}{2\sigma^2}\right\}.$$

We typically incorporate this Gaussian distribution within table or tree CPDs. For example, in the case of a table CPD for a real valued attribute A with discrete parents $\mathbf{Pa}_A$, we would associate a separate Gaussian distribution $N(\mu_{\boldsymbol{u}}, \sigma^2_{\boldsymbol{u}})$ with each possible instantiation $\boldsymbol{u} \in \mathbf{Pa}_A$ to the parents of A.

2.3 Probabilistic Relational Models

Over the last decade, Bayesian networks have been used with great success in a wide variety of real-world and research applications. However, despite their success, Bayesian networks are often inadequate for representing large domains and complex interactions, as those we encounter in the biological domain. A Bayesian network involves a pre-specified set of random variables, whose relationship to each other is fixed in advance. Hence, a Bayesian network cannot be used to deal with the biological domain, where we encounter a varying number of entities in a variety of configurations. This limitation of Bayesian networks is a direct consequence of the fact that they lack the concept of an "object" (or domain entity). Hence, they cannot represent general principles about multiple similar objects which can then be applied in multiple contexts.

Probabilistic relational models (PRMs) (Koller and Pfeffer, 1998, Getoor, 2001) extend Bayesian networks with the concepts of objects, their properties, and relations between them. In a way, they are to Bayesian networks as relational logic is to propositional logic. A PRM specifies a template for a probability distribution over a database. The template includes a relational component, that describes the relational schema for our domain, and a probabilistic component, that describes the probabilistic dependencies that hold in our domain. A PRM has a coherent formal semantics in terms of probability distributions over sets of relational logic interpretations. Given a set of ground objects, a PRM specifies a probability distribution over a set of interpretations involving these objects. A PRM, together with a particular database of objects and relations, defines a probability distribution over the attributes of the objects.

2.3.1 Relational language

The relational language allows us to describe the kinds of objects in our domain. For example, Figure 2.5 shows the schema for a simple gene expression domain that we will be using as our running example in this chapter. This domain clearly contains genes, arrays, and expression measurements. The classes in the schema are thus Gene,

Array, and Expression.

More formally, a schema for a relational model describes a set of *classes*, $\mathcal{C} = \{C_1, \ldots, C_n\}$. Each class is associated with a set of *descriptive attributes*. For example, genes may have descriptive attributes such as the cellular location in which they perform their function and the functional unit or module to which they belong; arrays may have descriptive attributes such as whether they are measuring samples from tumor cells and the cluster of arrays to which they belong. Note that in reality, some attributes, such as the cellular location of a gene or whether the array represents tumor samples, may be observed, while others, such as the module of a gene or the cluster of an array, may be hidden.

The set of descriptive attributes of a class C is denoted $\mathcal{A}[\mathsf{C}]$. Attribute A of class C is denoted $\mathsf{C}.A$, and its space of values is denoted $Val(\mathsf{C}.A)$. For example, the value space for Array.*Tumor* in this example is $\{false, true\}$, while the value space for Array.*Cluster* is $\{1, 2, 3\}$ if we assume that there are 3 possible array clusters.

In addition, we need a method for allowing an object to refer to another object. For example, we want an expression measurement to refer both to the gene it is measuring and to the associated array. We achieve this effect using *reference slots*. Specifically, each class is associated with a set of reference slots. The set of reference slots of a class C is denoted $\mathcal{R}[\mathsf{C}]$. We use $\mathsf{C}.\rho$ to denote the reference slot ρ of C. Each reference slot ρ is typed, i.e., the schema specifies the range type of object that may be referenced. More formally, for each ρ in C, the *domain type* $\mathrm{Dom}[\rho]$ is C and the *range type* $\mathrm{Range}[\rho]$ is Y for some class Y in $\mathcal{C}$. For example, the class Expression has reference slots *Gene* (with range type Gene) and *Array* (with range type Array). In Figure 2.5(a) the reference slots are underlined.

There is a direct mapping between our representation and that of relational databases. Each class corresponds to a single table and each attribute corresponds to a column. Our descriptive attributes correspond to standard attributes in the table (though some of these attributes may be hidden and thus no values stored in the table), and our reference slots correspond to attributes that are foreign keys (key attributes of another table).

For each reference slot ρ, we can define an *inverse slot* ρ^{-1}, which is interpreted

as the inverse function of ρ. For example, we can define an inverse slot for the *Gene* slot of Expression and call it *Expressed-In*. Note that this is not a one-to-one relation, but returns a *set* of Expression objects. More formally, if Dom[ρ] is C and Range[ρ] is Y, then Dom[ρ^{-1}] is Y and Range[ρ^{-1}] is C.

Finally, we define the notion of a *slot chain*, which allows us to compose slots, defining functions from objects to other objects to which they are indirectly related. More precisely, we define a *slot chain* $\rho_1, \ldots, \rho_k$ to be a sequence of slots (inverse or otherwise) such that for all i, Range[ρ_i] = Dom[ρ_{i+1}]. For example, Gene.*Expressed-In.Array* can be used to denote the arrays in which a gene's expression was measured. Note that a slot chain describes a *set* of objects from a class.

The relational framework we have just described is motivated primarily by the concepts of relational databases, although some of the notation is derived from frame-based and object-oriented systems. However, the framework is a fully general one, and is equivalent to the standard vocabulary and semantics of relational logic.

2.3.2 Schema Instantiation

An *instance* $\mathcal{I}$ of a schema is simply a standard relational logic interpretation of this vocabulary. It specifies: for each class C, the set of objects in the class, $\mathcal{I}[\mathsf{C}]$; a value for each observed attribute $c.A$ (in the appropriate domain) for each object c; and a value y for each reference slot $c.\rho$, which is an object in the appropriate range type, i.e., $y \in$ Range[ρ]. We use $\mathcal{A}[c]$ as shorthand for $\mathcal{A}[\mathsf{C}]$, where c is of class C. For each object c in the instance and each of its attributes A, we use $\mathcal{I}[c.A]$ to denote the value of $c.A$ in $\mathcal{I}$. For example, Figure 2.5(b) shows an instance of the schema from our running example. In this (simple) instance there are two Genes, three Arrays, and five Expressions. The relations between them show that one gene ("p53") was measured only in two of the three arrays, while the other gene was measured in all three arrays.

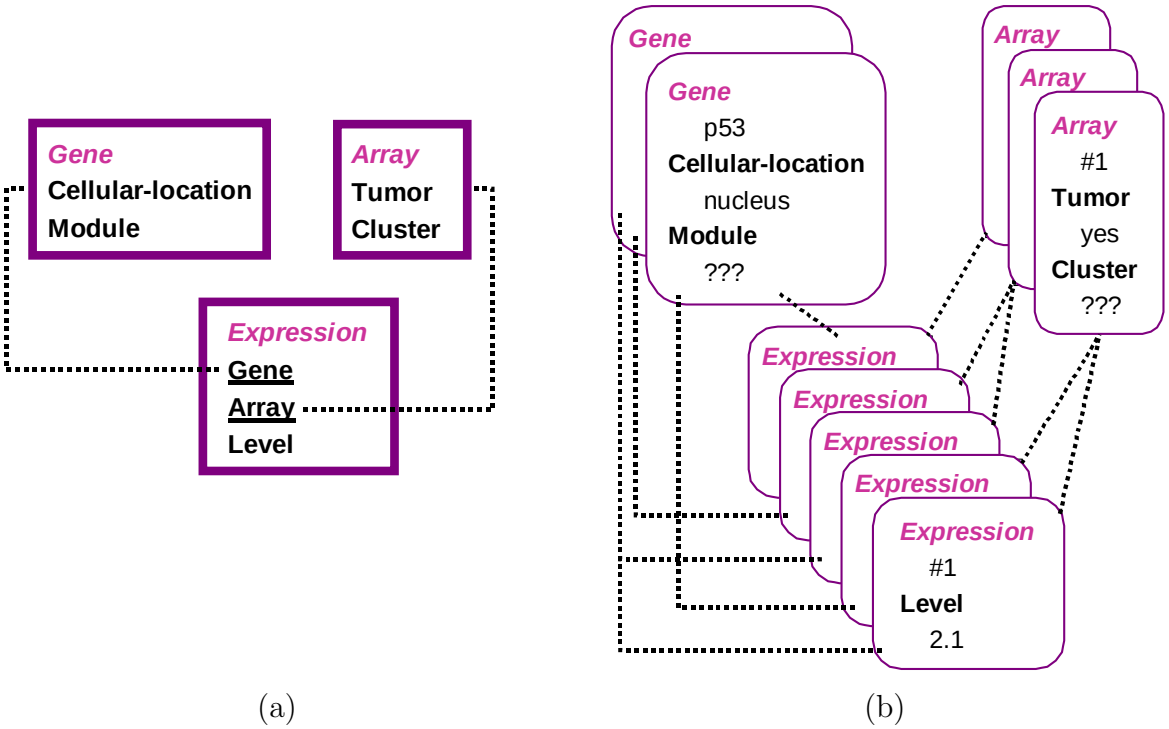


Figure 2.5: (a) A relational schema for a simple gene expression domain. The underlined attributes are reference slots of the class and the dashed line indicates the types of objects referenced. (b) An example instance of the relational schema of (a). Here we do not show the reference slots and used dashed lines to indicate the relationships that hold between objects.

2.3.3 Probabilistic Model

A PRM defines a probability distribution over a set of instances of a schema. Most simply, we assume that the set of objects and the relations between them are fixed, i.e., external to the probabilistic model. Then, the PRM defines only a probability distribution over the attributes of the objects in the model. The *relational skeleton* defines the possible instantiations that we consider; the PRM defines a distribution over the possible worlds consistent with the relational skeleton.

Definition 2.3.1: A *relational skeleton* σ_r of a relational schema is a partial specification of an instance of the schema. It specifies the set of objects $\sigma_r[\mathsf{C}_i]$ for each class C_i and the relations that hold between the objects. However, it leaves the values

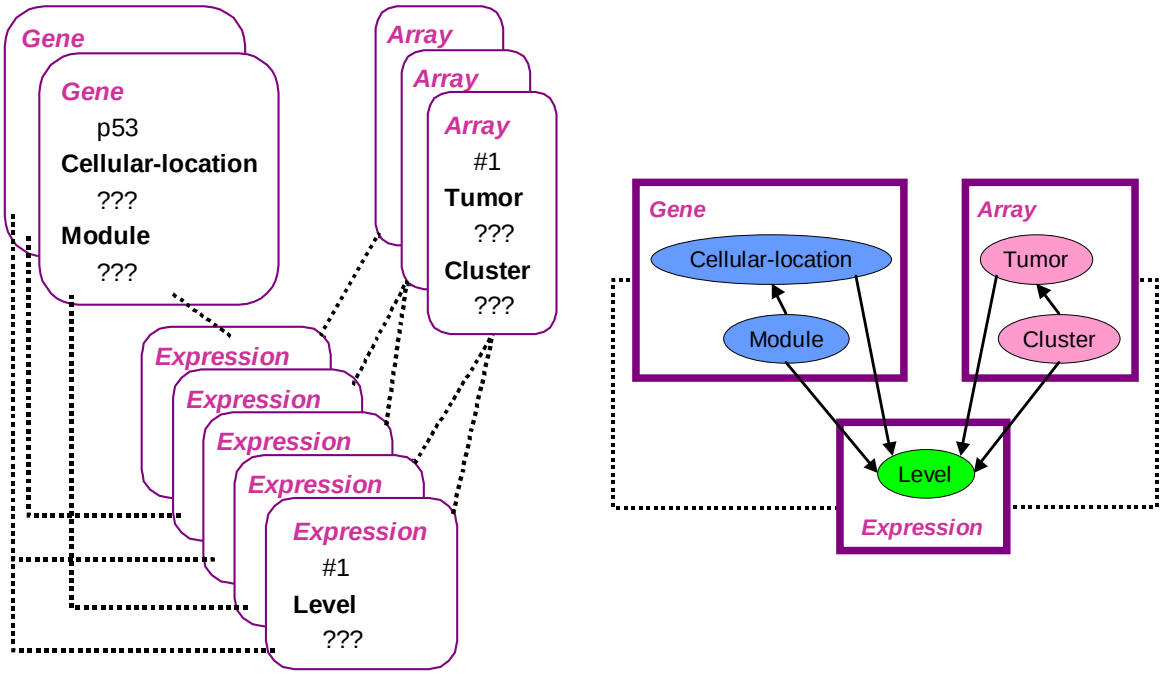


Figure 2.6: (a) The relational skeleton for the gene expression domain. (b) The PRM dependency structure for our gene expression domain.

of the attributes unspecified.

Figure 2.6(a) shows a relational skeleton for our running example. The relational skeleton defines the random variables in our domain; we have a random variable for each attribute of each object in the skeleton. A PRM then specifies a probability distribution over *completions* $\mathcal{I}$ of the skeleton.

A PRM consists of two components: the qualitative dependency structure, $\mathcal{S}$, and the parameters associated with it, $\boldsymbol{\theta}_{\mathcal{S}}$. The dependency structure is defined by associating with each attribute $\mathsf{C}.A$ a set of *parents* $\mathbf{Pa}_{\mathsf{C}.A}$. These correspond to *formal* parents; they will be instantiated in different ways for different objects in C. Intuitively, the parents are attributes that are "direct influences" on $\mathsf{C}.A$. In Figure 2.6(b), the arrows define the dependency structure.

We distinguish between two types of formal parents. The attribute $\mathsf{C}.A$ can depend on another probabilistic attribute B of C. This formal dependence induces a corresponding dependency for individual objects: for any object c in $\sigma_r[\mathsf{C}]$, $c.A$ will depend probabilistically on $c.B$. For example, in Figure 2.6(b), the cellular location of a gene depends on the module to which it belongs. The attribute $\mathsf{C}.A$ can also depend on attributes of related objects $\mathsf{C}.P.B$, where P is a slot chain. In Figure 2.6(b), the expression level depends on Expression.*Gene.Module*, Expression.*Gene.Cellular-location*, Expression.*Array.Cluster*, and Expression.*Array.Tumor*.

Given a set of parents $\mathbf{Pa}_{\mathsf{C}.A}$ for $\mathsf{C}.A$, we can define a local probability model for $\mathsf{C}.A$. We associate $\mathsf{C}.A$ with a CPD that specifies $P(\mathsf{C}.A \mid \mathbf{Pa}_{\mathsf{C}.A})$. Let $\boldsymbol{U}$ be the set of parents of $\mathsf{C}.A$, $\boldsymbol{U} = \mathbf{Pa}_{\mathsf{C}.A}$. Each of these parents U_i — whether a simple attribute in the same relation or an attribute of a related object $\mathsf{C}.P.B$ — has a set of value $Val(U_i)$ in some ground type. For each tuple of values $\boldsymbol{u} \in Val(\boldsymbol{U})$, we specify a distribution $P(\mathsf{C}.A \mid \boldsymbol{u})$ over $Val(\mathsf{C}.A)$. This entire set of parameters comprises $\boldsymbol{\theta}_{\mathcal{S}}$. Figure 2.7 shows an example for the CPD of the expression level.

Definition 2.3.2: A *probabilistic relational model (PRM)* Π for a relational schema Σ is defined as follows. For each class $\mathsf{C} \in \mathcal{C}$ and each descriptive attribute $A \in \mathcal{A}[\mathsf{C}]$, we have:

- a set of *parents* $\mathbf{Pa}_{\mathsf{C}.A} = \{U_1, \ldots, U_l\}$, where each U_i has the form $\mathsf{C}.B$ or $\mathsf{C}.P.B$ where P is a slot chain.
- a *conditional probability distribution (CPD)*, $P(\mathsf{C}.A \mid \mathbf{Pa}_{\mathsf{C}.A})$.

2.3.4 PRM semantics

As mentioned above, given a relational skeleton, σ_r, a PRM defines a distribution over possible worlds: The instantiations of the database that are consistent with σ_r. Given any σ_r, we have a set of random variables of interest: the attributes $c.A$ of the objects in the skeleton. Formally, let $\sigma_r[\mathsf{C}]$ denote the set of objects in skeleton σ_r whose class is C. The set of random variables for σ_r is the set of attributes of the form $c.A$ where $c \in \sigma_r[\mathsf{C}_\mathsf{i}]$ and $A \in \mathcal{A}[\mathsf{C}_\mathsf{i}]$ for some class C_i. The PRM specifies a

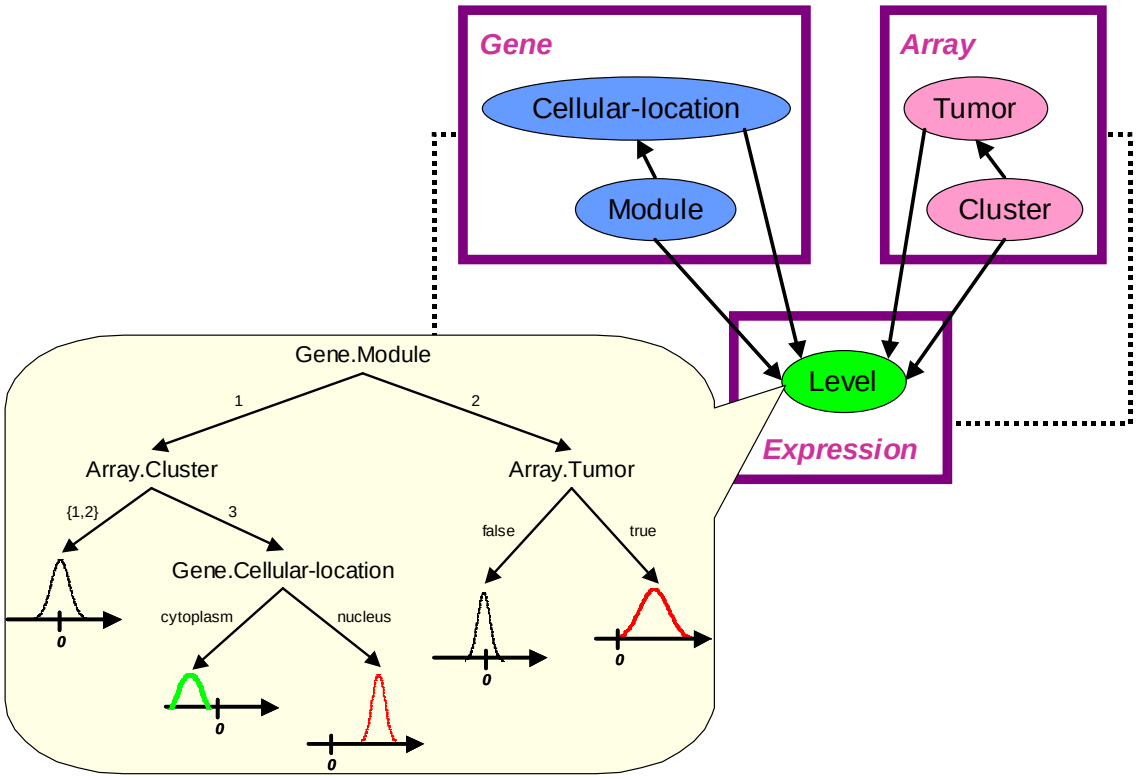


Figure 2.7: The CPD for Expression.*Level* given Gene.*Module*, Gene.*Cellular-location*, Array.*Cluster*, and Array.*Tumor*. We represent this CPD as a tree CPD. Since Expression.*Level* is a continuous random variable, its distribution given any assignment to its parents will be a density function. In this example, we chose the Normal distribution as the density function, shown pictorially for each leaf in the tree CPD.

probability distribution over the possible joint assignments of values to all of these random variables. This joint distribution is specified by a *ground* Bayesian network over the random variables $c.A$, where the ground Bayesian network is induced from the given skeleton σ_r and PRM structure:

Definition 2.3.3: A PRM Π together with a skeleton σ_r defines the following *ground Bayesian network*:

- There is a node $c.A$ for every attribute of every object $c \in \sigma_r[\mathsf{C}]$.
- Each attribute $c.A$ depends probabilistically on its parents $\mathbf{Pa}_{\mathsf{C}.A}$ where each parent is of the form $c.B$ or $c.P.B$ where P is a slot chain.

- The CPD for $c.A$ is $P(\mathsf{C}.A \mid \mathbf{Pa}_{\mathsf{C}.A})$.

Figure 2.8 shows the ground Bayesian network for the PRM skeleton of Figure 2.6(a). As with Bayesian networks, the joint distribution over these assignments is factored. That is, we take the product, over all $c.A$, of the probability in the CPD of the specific value assigned by the instance to the attribute given the values assigned to its parents. Formally, this is written as follows:

$$P(\mathcal{I} \mid \sigma_r, \mathcal{S}, \theta_{\mathcal{S}}) = \prod_{\mathsf{C}_\mathsf{i}} \prod_{A \in \mathcal{A}[\mathsf{C}_\mathsf{i}]} \prod_{c \in \sigma_r[\mathsf{C}_\mathsf{i}]} P(\mathcal{I}[c.A] \mid \mathcal{I}[\mathbf{Pa}_{c.A}])$$

This expression is very similar to the chain rule for Bayesian networks that we presented in Equation 2.1. There are three primary differences. First, our random variables are the attributes of a set of objects. Second, the set of parents of a random variable can vary according to the relational context of the object — the set of objects to which it is related. Third, the parameters of the local probability models for attributes of objects in the same class are identical.

2.4 The Difference between PRMs and BNs

The PRM specifies the probability distribution using the same underlying principles used in specifying Bayesian networks. The assumption is that each of the random variables in the PRM — in this case the attributes $c.A$ of the individual objects c — is directly influenced by only a few others. The PRM therefore defines for each $c.A$ a set of parents, which are the direct influences on it, and a local probabilistic model that specifies the dependence on these parents. In this way, the PRM is like a BN.

However, there are two primary differences between PRMs and Bayesian networks. First, a PRM defines the dependency model at the class level, allowing it to be used for any object in the class. In some sense, it is analogous to a universally quantified statement over all objects of the same class. Second, the PRM explicitly uses the relational structure of the model, in that it allows the probabilistic model of an attribute of an object to depend also on attributes of related objects. The specific set

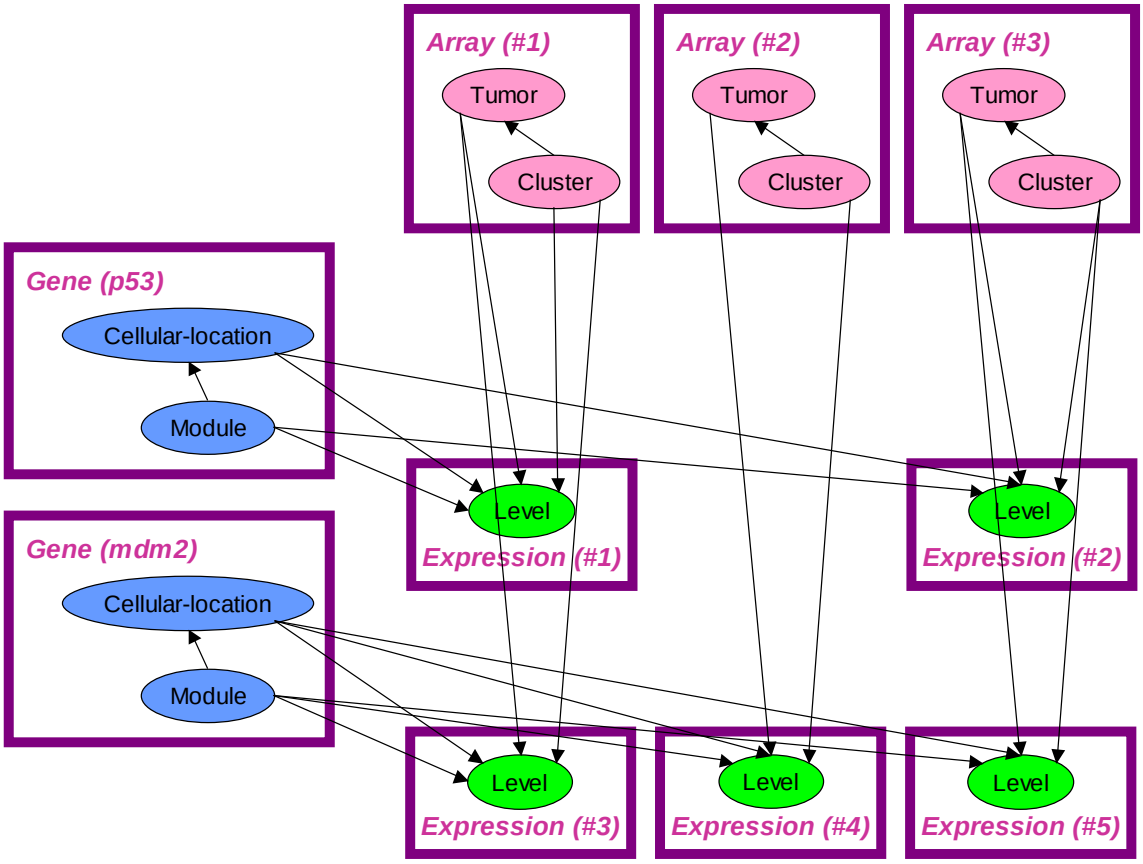


Figure 2.8: The ground Bayesian network for the PRM skeleton of Figure 2.6(a). This skeleton has two Gene objects, three Array objects, and five Expression objects. Note that the measurement of the expression for one of the genes (p53) in one of the arrays ('Array #2') is missing.

of related objects can vary with the skeleton σ_r; the PRM specifies the dependency in a generic enough way that it can apply to an arbitrary relational structure.

One can understand the semantics of a PRM together with a particular relational skeleton σ_r by examining the ground Bayesian network defined earlier. The network has a node for each attribute of the objects in the skeleton. The local probability models for attributes of objects in the same class are identical (we can view the parameters as being shared); however, the distribution over the values of a particular node in the network will depend on the values of its parents in the network, which in turn are determined by the skeleton.

2.5 Inference in PRMs

An important aspect of probabilistic models in general, and of PRMs in particular, is that by representing a joint probability distribution over the attributes of all objects, they can support many types of inferences about our domain. As PRMs represent distributions over relational domains, they support many interesting patterns of reasoning. In the biological domain, we typically have many hidden variables. Given a PRM, we can infer their values. For example, in our gene expression domain, the module that a gene participates in and the cluster that an array belongs to are typically unknown. For a PRM with a skeleton σ_r, we can infer the posterior distribution over the module assignment of a gene g by computing:

$$P(g.Module \mid \mathsf{Gene}.\mathbf{Cellular\text{-}location}, \mathsf{Array}.\mathbf{Tumor}, \mathsf{Expression}.\mathbf{Level}),$$

and the posterior distribution over the cluster assignment of an array object a by computing:

$$P(a.Cluster \mid \mathsf{Gene}.\mathbf{Cellular\text{-}location}, \mathsf{Array}.\mathbf{Tumor}, \mathsf{Expression}.\mathbf{Level}),$$

where $\mathsf{C_i}.\mathbf{A}$ represents the set of all A attributes from all objects of class $\mathsf{C_i}$ (e.g., $\mathsf{Array}.\mathbf{Tumor} = \{a.Tumor \mid a \in \sigma_r[\mathsf{Array}]\}$). Thus, given a PRM we can infer the gene modules in the cell, and identify arrays in which these modules have similar patterns of expression.

Importantly, we can also use a PRM for making predictions. For example, given a new experiment, we can predict whether it was taken from a tumor sample or a healthy sample, and given a newly identified gene, we can attempt to predict its cellular location.

All such inferences can be computed by performing probabilistic inference on the ground BN. Thus, we present the inference algorithms on general BNs, where in our case these BNs will correspond to the ground BN produced by a PRM. Inference in BNs is an active area of research and many general purpose inference methods have been developed. These methods can be divided into two categories: those that are

exact, and those that only compute an approximation to the inference query. Unfortunately, as we describe below, exact inference algorithms will rarely be tractable in our setting, due to the large ground BNs generated by our models, and the many dependencies that exist between their attributes. Thus, we only provide a brief overview of exact inference methods and show where they break down in our case. Since most approximate inference algorithms are applicable in our setting, we outline some of them below. However, as these algorithms are general purpose, often we can design better approximate algorithms by exploiting structure specific to the model we are dealing with. Thus, in practice, we apply approximate algorithms that are tailored to our specific network. These algorithms are described in the respective chapters that describe each model.

2.5.1 Exact inference

As mentioned above, in the most general case we can perform inference on the ground BN, where the inference task is to compute the posterior distribution over a subset of variables of interest $\boldsymbol{X} \subseteq \boldsymbol{Z}$, given an assignment to another subset $\boldsymbol{E} \subseteq \boldsymbol{Z}$, where $\boldsymbol{Z}$ is the set of all variables in the BN. That is, we wish to compute $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$, where $\boldsymbol{e}$ is the assignment to the observed variables. In the PRM setting, these subsets of variables correspond to attributes of objects.

Variable elimination

Several general purpose exact inference methods on BNs have been proposed. One method is *variable elimination*. In short, this method computes $P(\boldsymbol{X}, \boldsymbol{E} = \boldsymbol{e})$, which can then be normalized to obtain $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$. The computation starts from:

$$P(\boldsymbol{X}, \boldsymbol{E} = \boldsymbol{e}) = \sum_{\boldsymbol{z} \in \boldsymbol{Z}} P(\boldsymbol{X}, \boldsymbol{E} = \boldsymbol{e}, \boldsymbol{Z} = \boldsymbol{z})$$

Note that we can write the full joint distribution in its factorized form, according to the chain rule for Bayesian networks that we presented in Equation 2.1. Let a *factor* over a set of variables $\boldsymbol{D}$ be a function from $Val(\boldsymbol{D})$ to $\Re^+$. We can then represent the

joint distribution as a product of factors, one for each variable given its parents in the BN, where each factor initially corresponds to the conditional probability distribution of a variable given its parents.. In each step of the variable elimination algorithm, we then *eliminate* one of the variables $Z_i \in \boldsymbol{Z}$, by taking all the factors in the joint distribution in which Z_i appears, and replacing them with a single factor over all the other variables that appear with Z_i in any of the factors. Formally, let $J(Z_i, \boldsymbol{Y}) = \prod_{j=1}^{k} f_j(Z_i, \boldsymbol{Y}_j)$ be the product of all k factors $f_j(Z_i, \boldsymbol{Y}_j)$ that involve Z_i, where $\boldsymbol{Y} = \bigcup_{j=1}^{k} Y_j$. We replace $J(Z_i, \boldsymbol{Y})$ with a single new factor over $\boldsymbol{Y}$, $f(\boldsymbol{Y})$, such that for each assignment $\boldsymbol{y}$ to $\boldsymbol{Y}$, we have $f(\boldsymbol{Y} = \boldsymbol{y}) = \sum_{z \in Z_i} J(Z_i = z, \boldsymbol{Y})$. That is, the new factor over $\boldsymbol{Y}$ assigns a real value to each possible assignment $\boldsymbol{y}$ to $\boldsymbol{Y}$, which is equal to the number we would get in the product of all factors that involve Z_i and variables in $\boldsymbol{Y}$, when summing over all the possible values of Z_i.

Since each step of the variable elimination algorithm creates a new factor whose representation is exponential in $\boldsymbol{Y}$, the key to an efficient application of this algorithm lies in selecting the ordering of variables to eliminate. By exploiting the form of the joint distribution induced by the structure of the Bayesian network it represents, variable elimination can be efficient for certain network topologies.

Unfortunately, we cannot generally apply variable elimination in our setting. To see why, consider as an example the ground BN of our gene expression domain. An instance of this ground BN for two genes, three arrays, and five expression measurements is shown in Figure 2.8. Assume that we wish to infer the posterior distribution over the module assignment of some gene g. That is, we wish to compute:

$$P(g.M \mid g.C = c, \mathsf{G}.\boldsymbol{C} = \boldsymbol{c}, \mathsf{A}.\boldsymbol{T} = \boldsymbol{t}, \mathsf{E}.\boldsymbol{L} = \boldsymbol{l}) = \sum_{\boldsymbol{m} \in \boldsymbol{M}, \boldsymbol{c} \in \boldsymbol{C}} P(g.C = c, g.M, \mathsf{G}.\boldsymbol{M} = \boldsymbol{m}, \mathsf{A}.\boldsymbol{C} = \boldsymbol{c} \mid \mathsf{G}.\boldsymbol{C} = \boldsymbol{c}, \mathsf{A}.\boldsymbol{T} = \boldsymbol{t}, \mathsf{E}.\boldsymbol{L} = \boldsymbol{l})$$

where we used one letter abbreviations for class and attribute names (i.e., $\mathsf{G} = \mathsf{Gene}$, $\mathsf{A} = \mathsf{Array}$, $\mathsf{E} = \mathsf{Expression}$, $M = \mathit{Module}$, $L = \mathit{Level}$, $T = \mathit{Tumor}$, and $C = \mathit{Cellular\text{-}location}$ or $C = \mathit{Cluster}$), and G represents all gene objects except for the gene g over which we wish to perform the inference.

Following the variable elimination algorithm, we need to start by eliminating one

of the summed out variables, which in our case means eliminating one of the *Module* variable for one of the genes, or one of the *Cluster* variable for one of the arrays. If every gene has an expression measurement in every array, then the *Module* variable of every gene is involved in a factor with the *Cluster* variable of each of the array objects. Similarly, the *Cluster* variable of every array is involved in a factor with the *Module* variable of each of the gene objects. Thus, it is easy to see that the first step in the variable elimination algorithm will either create a joint factor over all the *Cluster* variables of all array objects (if we choose to first eliminate the *Module* variable for one of the genes), or a joint factor over all the *Module* variables of all gene objects (if we choose to first eliminate the *Cluster* variable for one of the arrays). In a typical application we may have thousands of genes and hundreds of arrays, and thus variable elimination will be intractable in our setting.

Clique trees

Another popular exact inference method (whose end result is equivalent to that of the variable elimination algorithm) is called the *Clique Tree* or *Joint Tree* algorithm. As implied by its name, this algorithm first transforms the network into a tree structure, in which nodes correspond to sets of variables from the original network. In this tree, inference then proceeds by iterations, where in each iteration every node performs a local computation in which it sends a *message* or *belief* to each of its neighboring nodes in the network. This procedure is also termed *belief propagation* and it can be shown to converge on a tree structure after at most k iterations, where k is the diameter of the tree.

Each local computation for a node requires time exponential in the number of variables assigned to the node. Thus, the key to an efficient application of the clique tree algorithm lies in the creation of the tree structure, where this tree must obey two properties. The first, called *family preserving*, requires that for each CPD in the original network, there is at least one node in the tree that contains all nodes in the CPD. The second property, called *running intersection*, requires that whenever there is a variable X such that $X \in N_i$ and $X \in N_j$ (N_i and N_j are two nodes in the clique tree), then X is also in every node in the (unique) path in the tree between N_i and

N_j.

Again, in our gene expression domain, it is easy to see that if we want to apply the clique tree algorithm for inferring the module assignment of one of the genes (or for inferring the cluster assignment of one of the arrays), then the clique tree needs to have one node containing the cluster variables of all arrays, or one node containing the module variables of all the genes. As the local computation performed by each node is exponential in the number of variables it contains, the clique tree algorithm is intractable for any realistic setting in the gene expression domain.

Structured exact inference

The exact inference algorithms we discussed thus far are general purpose algorithms that were designed for inference in BNs. Thus, they do not exploit any structure that may exist in our setting. Even though the networks produced by PRMs for the biological domain are large, their connectivity is not arbitrary, but rather follows certain structural properties as it was generated by a compact template representation. Thus, it might be possible to exploit regularities in this network structure in order to make inference tractable. Previous work on inference in structured probabilistic models (Koller and Pfeffer, 1997, Pfeffer *et al.*, 1999, Pfeffer, 2000) shows how effective inference can be done for a number of different structured probabilistic models. The algorithms make use of the structure imposed by the class hierarchy to decompose the distribution and effectively reuse the computation.

There are two ways in which aspects of the structure can be used to make inference more efficient. The first structural aspect is the natural encapsulation of objects that occurs in a well-designed class hierarchy. Ideally, the interactions between objects will occur via a small number of object attributes, and the majority of interactions between attributes will be encapsulated within the class. This can provide a natural decomposition of the model for suitable inference. The complexity of the inference will depend on the 'width' of the connections between objects; if the width is small, we are guaranteed an efficient procedure.

The second structural aspect that is used to make inference efficient is the fact that similar objects occur many times in the model (Pfeffer *et al.*, 1999, Poole, 2003).

Pfeffer *et al.* (1999) describe a recursive inference algorithm that caches the computations that are done for fragments of the model; these computations the need only be performed once, and we can then reuse them for another object recurring in the same context. We can think of this object as a generic object, which occurs repeatedly in the model. Exploiting these structural aspects of the model allow Pfeffer *et al.* (1999) to achieve impressive speedups; in a military battlespace domain the structured inference was orders of magnitudes faster than the standard BN exact inference algorithm.

While it is good to keep in mind that such exact inference algorithms are possible, the structural properties mentioned above hardly occur in the models we present in this thesis: the individual classes of the objects we consider do not encapsulate many interactions. Rather, the complexity most often stems from the complex interactions across objects. Moreover, even though some objects may repeatedly appear, their connectivity within the network will differ, making it hard to reuse the computations performed for one object to reason about other objects.

2.5.2 Approximate inference

As mentioned above, exact inference methods are generally not applicable in our setting, so we must resort to approximate inference. Approximate inference is an active area of research and many general purpose algorithms have been developed for this task. We provide a brief overview of some of these methods below. Since these methods were developed for the general case, it is plausible that in certain cases we might be able to design more efficient approximations by exploiting the structure of the particular model we consider. Indeed, a major algorithmic challenge is to design such algorithms. For many of the models we explore in subsequent chapters, we present such algorithms. However, for classes of models for which it is not clear how to design tailored algorithms, the approximate inference algorithms described below can be used.

Roughly, we can divide the approximate inference algorithms into three classes: *particle-based*, *variational*, and *belief propagation*. We provide a brief overview of

these methods.

Particle-based approximate inference

Recall that in the general inference problem, we observe an assignment, $\boldsymbol{e}$, to a set of variables $\boldsymbol{E} \subseteq \boldsymbol{Z}$, and we wish to compute the posterior distribution $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$ for another set $\boldsymbol{X} \subseteq \boldsymbol{Z}$. In *particle-based* methods, we generate a set of instantiations, often called *particles*, to all or some of the variables in the network, and then use these instantiations to approximate $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$. Numerous variants of particle-based methods have been proposed, where one of the major aspects in which methods differ is in the process by which they generate the particles.

The simplest particle-based method is *rejection sampling*. In this method, we use the ground BN to sample instantiations to all of the variables from the joint distribution represented by the BN, $P_{\mathcal{B}}(\boldsymbol{Z})$. We then *reject* those instantiations that are not consistent with the assignment to $\boldsymbol{E}$ (i.e., instantiations in which $\boldsymbol{E} \neq \boldsymbol{e}$). It is easy to see that this process generates instantiations from the posterior distribution $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$. After sampling M such instantiations, we can then approximate the probability of each assignment $\boldsymbol{x}$ to $\boldsymbol{X}$ given the assignment to the observed variables, as:

$$P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e}) = \frac{1}{M} \sum_{m=1}^{M} \eta\{\boldsymbol{X}[m] = \boldsymbol{x}\},$$

where $\eta\{\boldsymbol{X}[m] = \boldsymbol{x}\}$ is an indicator function that is equal to 1 if and only if the assignment to the set of variables $\boldsymbol{X}$ was $\boldsymbol{x}$ in the m-th sample. To generate a single sample from the joint distribution $P_{\mathcal{B}}(\boldsymbol{Z})$, we sample values for each node in some order consistent with the partial order of the ground BN. This ensures that by the time we sample a node we have values for all its parents. We can then sample a value for a node from the distribution defined by its CPD and by the values chosen for the node's parents.

An obvious problem with the above approach is that if the probability of observing the assignment $\boldsymbol{e}$ to the set of observed nodes $\boldsymbol{E}$ is small, then we will rarely generate samples that are consistent with it, and we will end up rejecting most of the samples

generated. To address this problem, we can generate samples using *likelihood weighting*. In this method, we generate samples as before, except that when our sampling process achieves an observed node $E_i \in \boldsymbol{E}$, we set its value to its observed assignment e_i in $\boldsymbol{e}$. To compensate for setting the value of E_i, we assign a weight to the m-th sample, equal to $P(E_i = e_i \mid \mathbf{Pa}_{E_i} = \mathbf{Pa}_{E_i}[m])$, where $\mathbf{Pa}_{E_i}[m]$ is the assignment to the parents of E_i in the m-th sample. This is the probability of observing the value e_i for the variable E_i, had it been sampled using the standard sampling process. Since we have multiple nodes in $\boldsymbol{E}$ for which we set their values to be consistent with our evidence, the total weight assigned to the m-th sample will be the product of the probability of observing the value of each of the observed nodes given the assignment sampled for their parents:

$$w[m] = \prod_{E_i \in \boldsymbol{E}} P(E_i = e_i \mid \mathbf{Pa}_{E_i} = \mathbf{Pa}_{E_i}[m])$$

We can now use these weighted samples to approximate the probability of each assignment $\boldsymbol{x}$ to $\boldsymbol{X}$ given the assignment to the observed variables, as:

$$P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e}) = \frac{\sum_{m=1}^{M} w[m] 1\{\boldsymbol{X}[m] = \boldsymbol{x}\}}{\sum_{m=1}^{M} w[m]}.$$

One of the limitations of likelihood weighting is that an observed node affects the sampling only for nodes that are its descendants. The effect on nodes that are non-descendants is accounted for only by the weights. In cases where much of the evidence is at the leaves of the network, we are essentially sampling from the prior distribution, which is often very far from the desired posterior.

Markov Chain Monte Carlo (MCMC) is an approach for generating samples from the posterior distribution. An example of a general MCMC algorithm is the *Metropolis-Hastings algorithm*. In this algorithm, we draw sample from a *proposal distribution* $Q(\boldsymbol{x} \rightarrow \boldsymbol{x}')$, which is a distribution over the possible assignments $\boldsymbol{x}'$ to $\boldsymbol{X}$ given the current assignment $\boldsymbol{x}$ to $\boldsymbol{X}$. We then either *accept* or reject the the new

assignment $\boldsymbol{x}'$ proposed by the proposal distribution with an *acceptance probability*:

$$\mathcal{A}(\boldsymbol{x} \to \boldsymbol{x}') = \min\left[1, \frac{P(\boldsymbol{x}')Q(\boldsymbol{x}' \to \boldsymbol{x})}{P(\boldsymbol{x})Q(\boldsymbol{x} \to \boldsymbol{x}')}\right]$$

Thus, the algorithm starts from a random initial assignment $\boldsymbol{x}^0$ to $\boldsymbol{X}$, and repeatedly uses $Q(\boldsymbol{x} \to \boldsymbol{x}')$ to propose a new assignment $\boldsymbol{x}^{t+1}$ given the current assignment $\boldsymbol{x}^t$. Each new assignment is then accepted with probability $\mathcal{A}(\boldsymbol{x}^t \to \boldsymbol{x}^{t+1})$. Note that this is indeed a *Markov Chain* since each assignment $\boldsymbol{x}^{t+1}$ is proposed based only on the previous assignment $\boldsymbol{x}^t$. The algorithm is typically run for many iterations so that the initial assignment $\boldsymbol{x}^0$ is "forgotten". These discarded samples are known as *burn-in* samples. It can be shown that this algorithm generates samples from the posterior distribution $P(\boldsymbol{X} \mid \boldsymbol{E} = \boldsymbol{e})$ which we can then use to compute the probability of a query of interest as described above.

A special case of the Metropolis-Hastings algorithm is *Gibbs-sampling*, which can be especially effective for probabilistic graphical models. In this algorithm, we propose the next assignment $\boldsymbol{x}'$ to $\boldsymbol{X}$ through a series of $|\boldsymbol{X}|$ sequential steps, where in each step we select one of the variables $X_i \in \boldsymbol{X}$ and sample a value for it from the distribution $P(X_i \mid \{\boldsymbol{X} - X_i\} = \boldsymbol{x}[\{\boldsymbol{X} - X_i\}], \boldsymbol{E} = \boldsymbol{e})$, where $\boldsymbol{x}[\{\boldsymbol{X} - X_i\}]$ represents the assignment to all the variables except X_i. Having sampled a value x_i for X_i, we set X_i to x_i in the assignment $\boldsymbol{x}$ so that the next variable sampled uses this new assignment for sampling its own value. It can be shown that if we use this proposal distribution, then the acceptance probability is 1, making the algorithm very efficient as samples are not wasted. Moreover, a value for a variable X_i can be sampled efficiently. To see why, recall that the *Markov blanket* of a variable is defined to be the set of variables that are its parents, children, and parents of its children in the ground Bayesian network. As a variable in a Bayesian network is conditionally independent of all other variables given its Markov blanket, we can usually sample a value for a variable X_i efficiently, by computing the distribution $P(X_i \mid \mathbf{Pa}_{X_i} = \boldsymbol{x}[\mathbf{Pa}_{X_i}], \mathbf{Ch}_{\mathbf{X_i}} = \boldsymbol{x}[\mathbf{Ch}_{\mathbf{X_i}}], \mathbf{Pa}_{\mathbf{Ch}_{\mathbf{X_i}}} = \boldsymbol{x}[\mathbf{Pa}_{\mathbf{Ch}_{\mathbf{X_i}}}], \boldsymbol{E} = \boldsymbol{e})$, where $\mathbf{Ch}_{\mathbf{X_i}}$ correspond to the children variables of X_i in the ground Bayesian network.

Although MCMC generates samples from the posterior distribution, its applicability in practice depends on many factors which are hard to assess in advance. Specifically, it depends on the *mixing-time* of the Markov chain, which is a measure of the number of samples that we need to collect before we can use the samples to get a good estimate of the desired query. In certain cases, the mixing time can be extremely long, requiring us to take many samples before computing the probability of our query of interest.

Structured variational approximations

Another class of approximate inference algorithms are called *structured variational approximations* (Jordan *et al.*, 1998). These algorithms define a class of distributions $\boldsymbol{Q}$ that are simplifications of the original distribution $P_{\mathcal{B}}$, and then search for a particular Q within that class that is a good approximation to $P_{\mathcal{B}}$. Queries of the form $P(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{E} = \boldsymbol{e})$ are then answered using inference on Q rather than $P_{\mathcal{B}}$.

For a given class of distributions $\boldsymbol{Q}$, we search for one that minimizes the *relative entropy* between a particular choice $Q \in \boldsymbol{Q}$ and $P_{\mathcal{B}}$, $\boldsymbol{D}(P_{\mathcal{B}} \parallel Q)$, where the relative entropy is a common distance measure between two distributions, defined as:

$$\boldsymbol{D}(P_{\mathcal{B}} \parallel Q) = \sum_{\boldsymbol{x} \in Val(\boldsymbol{X})} P_{\mathcal{B}}(\boldsymbol{X} = \boldsymbol{x}) \log \frac{P_{\mathcal{B}}(\boldsymbol{X} = \boldsymbol{x})}{Q(\boldsymbol{X} = \boldsymbol{x})}$$

Thus, the variational approximations require us to find $Q = \text{argmin}_{Q' \in \boldsymbol{Q}} \boldsymbol{D}(P_{\mathcal{B}} \parallel Q')$. For classes of distributions $\boldsymbol{Q}$ that are simple, we can usually find a local minimum of this optimization task by solving a set of fixed point equations.

In general there is a trade off in selecting the class of distributions $\boldsymbol{Q}$: simple choices can allow us to solve the above optimization problem as well as compute answers to the original query more efficiently; however, if the class $\boldsymbol{Q}$ is too simple then it cannot capture many of the dependencies in the original distribution $P_{\mathcal{B}}$ and will consequently result in poor approximations. For example, in the commonly used *mean-field approximation*, the class $\boldsymbol{Q}$ is selected to be a product of marginals over all variables $X_i \in \boldsymbol{X}$, so that Q factorizes as: $Q(\boldsymbol{X}) = \prod_{X_i \in \boldsymbol{X}} Q(X_i)$. This is perhaps the simplest choice of distributions $\boldsymbol{Q}$. Other approximations use the structure of the

original ground Bayesian network and select $\boldsymbol{Q}$ so that it captures as many of the dependencies in $P_{\mathcal{B}}$ as possible, while still resulting in distributions $\boldsymbol{Q}$ that are easy to perform inference with. However, in general it is quite difficult to select $\boldsymbol{Q}$ and assess the quality of the approximation for a given choice of $\boldsymbol{Q}$.

Belief propagation

Another approximate inference method is *belief propagation*, a local message passing algorithm, introduced by Pearl (1988). The idea is simple: We convert the original Bayesian network into an undirected graph (described below) and then perform local message passing between neighboring nodes in this graph, in the same way as we pass messages in a clique tree that is used for exact inference. This algorithm is guaranteed to converge to the exact marginal probabilities only if the original Bayesian network is singly connected (i.e., it is a tree). If the Bayesian network is not singly connected, the converted undirected graph will have cycles and the message passing scheme might double count messages (essentially, some of the information from a message passed by a node will be returned to it as a supposedly independent message after several iterations in which the message completes a cycle back to the node that generated the message). However, empirical results (Murphy and Weiss, 1999) show that BP often converges even when applied to graphs with cycles, and when it does, the marginals are in many cases a reasonable approximation to the correct posterior.

Several variants of the BP algorithm have been proposed, where the main difference between the variants is in the scheduling of the messages or the construction of the undirected graph on which BP is applied. We provide a brief outline of one variant of BP, referring the reader to Weiss (2000), Murphy and Weiss (1999), MacKay *et al.* (1997), and Wainwright *et al.* (2001) for additional details. Consider a Bayesian network over some set of nodes $\boldsymbol{Z}$. We first convert the graph into a *family graph*, with a node F_i for each variable Z_i in the BN, containing Z_i and its parents. Two nodes are connected if they have some variable in common. The CPD of Z_i is associated with F_i. Let ϕ_i represent the factor defined by the CPD; i.e., if F_i contains the variables $Z_i, \mathbf{Pa}_{Z_i}$, then ϕ_i is a function from the domains of these variables to $[0, 1]$. We also define ψ_i to be a factor over Z_i that encompasses our evidence about

$Z_i : \psi_i(Z_i) \equiv 1$ if Z_i is not observed. If we observe $Z_i = z$ (i.e., $Z_i \in \boldsymbol{E}$), we have that $\psi_i(z) \equiv 1$ and 0 elsewhere. Our posterior distribution is then $\alpha \prod_i \phi_i \prod_i \psi_i$, where α is a normalizing constant.

The belief propagation algorithm is now very simple. The entries of all messages are initialized to 1. At each iteration, all the family nodes simultaneously send messages to all others, as follows:

$$m_{ij}(F_i \bigcap F_j) \leftarrow \alpha \sum_{F_i - F_j} \phi_i \psi_i \cdot \prod_{k \in N(i) - \{j\}} m_{ki},$$

where α is a (different) normalizing constant and $N(i)$ is the set of families that are neighbors of F_i in the family graph. This process is repeated until the beliefs converge. At any point in the algorithm, our marginal distribution about any family F_i is $b_i = \alpha \phi_i \psi_i \prod_{k \in N(i)} m_{ki}$. Each iteration is linear in the number of edges in the BN. While the algorithm is not guaranteed to converge, it typically converges after just a few iterations. After convergence, the b_i give us approximate marginal distributions of the families in the ground BN.

2.6 Parameter Estimation

In the previous sections, we defined the PRM language and its semantics, and presented a simple example of how PRMs can be applied to the biological domain. However, we have not yet shown where a PRM comes from. One option is to have a domain expert construct the model by hand. This can be a laborious and time-consuming process. It involves first determining the dependencies between the object attributes. In general this task is challenging, but for some well understood domains the domain expert may be able to provide the model structure. Even for experts though, the elicitation of probabilities can be a very difficult task. Thus, this knowledge-engineering approach has limited applicability. It will only be successful in domains in which there is an expert who thoroughly understands the domain. Clearly, in the biological domain, where we wish to study thousands of genes, many of which are not even characterized, this approach is currently infeasible.

Thus, our approach in this thesis will be to learn the details of a PRM, including the structure of the dependencies between the object attributes, and the parameters of the different CPDs automatically from the input data. In the next two sections we review the general algorithms developed for these tasks by Friedman *et al.* (1999a) and Getoor (2001). These algorithms provide the basis for our approach to learn PRMs from data, but we have extended them in important ways to address the specific models we consider in the biological domain. Specifically, the algorithms proposed by Friedman *et al.* (1999a) and Getoor (2001) assumed that all attributes are observed in the training data, an assumption that is clearly too restrictive for the biological domain in which we wish to define various hidden properties and infer their values and dependencies. Thus, one important extension we developed was to learn PRMs in the presence of hidden attributes. We present this extension as part of the following two general sections on learning PRMs from data. Our other extensions are presented in the respective chapters in which we present the various models.

In the learning problem, our input contains a relational schema, that specifies the basic vocabulary in the domain — the set of classes, the attributes associated with the different classes, and the possible types of relations between objects in the different classes. Our training data consists of an instance of that schema, where the values of some of the attributes may be observed, while other attributes may be hidden. There are two variants of the learning task: parameter estimation and structure learning. In the parameter estimation task, we assume that the qualitative dependency structure of the PRM is known; i.e., the input consists of the schema, training data, and a qualitative dependency structure $\mathcal{S}$. The learning task is only to fill in the parameters that define the CPDs of the attributes. In the structure learning task, there is no additional required input and the goal is to extract an entire PRM, structure as well as parameters, from the training data alone.

In this section, we show how to learn the parameters for a PRM where the dependency structure is known. In other words, we are given the structure $\mathcal{S}$ that determines the set of parents for each attribute, and out task is to learn the parameters $\boldsymbol{\theta}_{\mathcal{S}}$ that define the CPDs for this structure.

2.6.1 The Complete Data Case

We start with the more simple setting of parameter estimation, where we assume that the training data specifies the values for attributes of all objects (i.e., we have fully observed data). While this task is relatively straightforward, it is of interest in and of itself, and will be a crucial component in the structure learning algorithms described in the next section.

The key ingredient in parameter estimation is the likelihood function: the probability of the data given the model. This function captures the response of the probability distribution to changes in the parameters. The likelihood of a parameter set is defined to be the probability of the data given the model. For a PRM, the likelihood of a parameter set $\boldsymbol{\theta}_{\mathcal{S}}$ is: $L(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) = P(\mathcal{I} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}})$. As usual, we typically work with the log of this function:

$$\begin{aligned} l(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) &= \log P(\mathcal{I} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}) \\ &= \sum_{\mathsf{C_i}} \sum_{A \in \mathcal{A}[\mathsf{C_i}]} \left[\sum_{c \in \sigma[\mathsf{C_i}]} \log P(\mathcal{I}[c.A] \mid \mathcal{I}[\mathbf{Pa}_{c.A}], \boldsymbol{\theta}_{\mathcal{S}}) \right] \end{aligned}$$

The key insight is that this equation is very similar to the log-likelihood of data given a Bayesian network (Heckerman, 1998). In fact, it is the likelihood function of the Bayesian network induced by the PRM given the skeleton (the ground BN). The main difference from standard Bayesian network parameter learning is that parameters for different nodes in the network are forced to be identical — the parameters are *tied*.

Maximum Likelihood Parameter Estimation

We can still use the well-understood theory of learning from Bayesian networks. Consider the task of performing *maximum likelihood* parameter estimation. Here, our goal is to find the parameter setting $\boldsymbol{\theta}_{\mathcal{S}}$ that maximizes the likelihood $L(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S})$ for a given $\mathcal{I}$, σ, and $\mathcal{S}$. In the case where the assignment to attributes of all objects are observed, this estimation task can be simplified by the *decomposition* of the log-likelihood function into a summation of terms corresponding to the various attributes

of the different classes:

$$l(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) = \sum_{\mathsf{C_i}} \sum_{A \in \mathcal{A}[\mathsf{C_i}]} \left[\sum_{c \in \sigma[\mathsf{C_i}]} \log P(\mathcal{I}[c.A] \mid \mathcal{I}[\mathbf{Pa}_{c.A}], \boldsymbol{\theta}_{\mathcal{S}}) \right]$$

Thus, in the case of parameter estimation from fully observed data, we can find the maximum likelihood setting for the parameters by finding the maximum likelihood setting for the CPD of each attribute given its parents separately. If we are learning CPDs from the exponential family (e.g., multinomial distributions, Gaussian distributions, and many others), then the likelihood function for each attribute given its parents can be reformulated in terms of *sufficient statistics* of the data. The sufficient statistics summarize the relevant aspects of the data. Their use here is similar to that in Bayesian networks (Heckerman, 1998), with one key difference. In a PRM, all of the attributes A of the same class $\mathsf{C_i}$ across all objects $c \in \sigma[\mathsf{C_i}]$ share the same parameters. Thus, we pool all of the data for the attribute $\mathsf{C_i}.A$ across all objects, and calculate our statistics based on this pooled data. More precisely, let $S_{\mathsf{C_i}.A}(\mathsf{C_i}.A, \mathbf{Pa}_{\mathsf{C_i}.A})$ be a sufficient statistic function for the CPD $P(\mathsf{C_i}.A, \mathbf{Pa}_{\mathsf{C_i}.A})$. Then the value of the statistic on the data set $\mathcal{D}$ is

$$\hat{S}_{\mathsf{C_i}.A} = \sum_{c \in \sigma[\mathsf{C_i}]} S_{\mathsf{C_i}.A}(\mathcal{I}[c.A], \mathcal{I}[c.\mathbf{Pa}_A]) \tag{2.2}$$

For example, in the case of multinomial table CPDs, we have one sufficient statistic function for each joint assignment $v \in Val(\mathsf{C_i}.A)$, $\boldsymbol{u} \in Val(\mathbf{Pa}_{\mathsf{C_i}.A})$, which is $\eta\{\mathcal{I}[c.A] = v, \mathcal{I}[\mathbf{Pa}_{c.A}] = \boldsymbol{u}\}$ — the indicator function that takes the value 1 if attribute A of object $c \in \sigma[\mathsf{C_i}]$ in $\mathcal{I}$ takes on the value v and its parents take on the value $\boldsymbol{u}$, and 0 otherwise. The statistic on the data is

$$\hat{S}_{\mathsf{C_i}.A}[v, \boldsymbol{u}] = \sum_{c \in \sigma[\mathsf{C_i}]} \eta\{\mathcal{I}[c.A] = v, \mathcal{I}[\mathbf{Pa}_{c.A}] = \boldsymbol{u}\} \tag{2.3}$$

Given these sufficient statistics, the formula for the log-likelihood of the data is:

$$l(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) = \sum_{\mathsf{C}_\mathsf{i}} \sum_{A \in \mathcal{A}[\mathsf{C}_\mathsf{i}]} \sum_{v \in Val(\mathsf{C}_\mathsf{i}.A)} \sum_{\boldsymbol{u} \in Val(\mathbf{Pa}_{\mathsf{C}_\mathsf{i}.A})} \hat{S}_{\mathsf{C}_\mathsf{i}.A}[v, \boldsymbol{u}] \cdot \log \theta_{v|\boldsymbol{u}} \qquad (2.4)$$

¿From Equation 2.4 we see that we would like to maximize the multinomial distribution:

$$\sum_{v \in Val(\mathsf{C}_\mathsf{i}.A)} \sum_{\boldsymbol{u} \in Val(\mathbf{Pa}_{\mathsf{C}_\mathsf{i}.A})} \hat{S}_{\mathsf{C}_\mathsf{i}.A}[v, \boldsymbol{u}] \cdot \log \theta_{v|\boldsymbol{u}}$$

It is straightforward to set the derivative of this function to zero and solve for the parameters. It is well-known that the values of the parameters that maximize this function are simply their frequencies in the data. Thus, assuming multinomial table CPDs, the maximum likelihood parameter setting of $P(\mathsf{C}_\mathsf{i}.A = v \mid \mathbf{Pa}_{\mathsf{C}_\mathsf{i}.A} = \boldsymbol{u})$ is:

$$P(\mathsf{C}_\mathsf{i}.A = v \mid \mathbf{Pa}_{\mathsf{C}_\mathsf{i}.A} = \boldsymbol{u}) = \frac{\hat{S}_{\mathsf{C}_\mathsf{i}.A}[v, \boldsymbol{u}]}{\sum_{v' \in Val(\mathsf{C}_\mathsf{i}.A)} \hat{S}_{\mathsf{C}_\mathsf{i}.A}[v', \boldsymbol{u}]}$$

As a consequence, parameter estimation in PRMs in the case of fully observed data with multinomial table CPDs is reduced to *counting*. The counts are the sufficient statistics, and we need to count one vector of sufficient statistics for each CPD. We note that in the fully observed case, parameter estimation in PRMs for many other types of CPDs for which we have sufficient statistics (e.g., Gaussian distributions) also reduces to simply collecting statistics from the data, although these summaries might be different than simple counting.

The derivation above also shows that parameter estimation in PRMs is very similar to parameter estimation in Bayesian networks. In fact, we can view this as learning parameters for the ground Bayesian network with tied parameters that the PRM induces given the skeleton. However, the learned parameters can then be used for reasoning about other skeletons, which induce completely different Bayesian networks.

Bayesian Parameter Estimation

In many cases, maximum likelihood parameter estimation is not robust, as it overfits the training data. The Bayesian approach uses a prior distribution over the parameters to smooth the irregularities in the training data, and is therefore significantly more robust. As we will see in the next section, the Bayesian framework also gives us a good metric for evaluating the quality of different candidate structures.

Roughly speaking, the Bayesian approach introduces a prior over the unknown parameters, and performs Bayesian conditioning, using the data as evidence, to compute a posterior distribution over these parameters. Under the assumption that the prior distribution is independent of the skeleton σ, this posterior distribution can be written as:

$$P(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) = \frac{1}{Z} P(\mathcal{I} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}) P(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{S})$$

where Z is a normalization constant such that $Z = P(\mathcal{I} \mid \sigma, \mathcal{S})$. Thus, we see that the posterior probability over parameters is proportional to a product of the log likelihood of the PRM instance and a prior distribution over parameters. As we saw in the previous section, the likelihood decomposes by terms that correspond to the various attributes of the different classes. By making an assumption on the form of the prior distribution, we can get a similar decomposition for the posterior distribution. This assumption is called *global parameter independence* and is a standard assumption on the form of the prior commonly used in learning Bayesian networks (Heckerman *et al.*, 1995). The assumption states that the prior over the parameters for the different attributes $\mathsf{C}_\mathsf{i}.A$ are independent:

$$P(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{S}) = \prod_{\mathsf{C}_\mathsf{i}} \prod_{A \in \mathcal{A}[A]} P(\boldsymbol{\theta}_{\mathsf{C}_\mathsf{i}.A \mid \mathbf{Pa}_{\mathsf{C}_\mathsf{i}.A}} \mid \mathcal{S}) \tag{2.5}$$

Using Equation 2.5, the posterior distribution over parameters decomposes as:

$$P(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) \propto \prod_{\mathsf{C}_\mathsf{i}} \prod_{A \in \mathcal{A}[A]} \left[P(\boldsymbol{\theta}_{\mathsf{C}_\mathsf{i}.A \mid \mathbf{Pa}_{\mathsf{C}_\mathsf{i}.A}} \mid \mathcal{S}) \prod_{c \in \sigma[\mathsf{C}_\mathsf{i}]} P(\mathcal{I}[c.A] \mid \mathcal{I}[\mathbf{Pa}_{c.A}], \boldsymbol{\theta}_{\mathsf{C}_\mathsf{i}.A \mid \mathbf{Pa}_{\mathsf{C}_\mathsf{i}.A}}) \right]$$

We can further decompose this posterior distribution if we also assume *local parameter independence*, also a common assumption in Bayesian network learning (Heckerman *et al.*, 1995), which states that the prior distribution for the different values of the parents are independent:

$$P(\boldsymbol{\theta}_{\mathsf{C_i}.A|\mathbf{Pa}_{\mathsf{C_i}.A}} \mid \mathcal{S}) = \prod_{\boldsymbol{u} \in Val(\mathbf{Pa}_{\mathsf{C_i}.A})} P(\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}} \mid \mathcal{S}) \tag{2.6}$$

Using both the global and local parameter independence assumptions, we can now decompose the posterior distribution as follows:

$$P(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{I}, \sigma, \mathcal{S}) \propto \prod_{\mathsf{C_i}} \prod_{A \in \mathcal{A}[A]} \prod_{\boldsymbol{u} \in Val(\mathbf{Pa}_{\mathsf{C_i}.A})} \left[P(\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}} \mid \mathcal{S}) \prod_{c \in \sigma[\mathsf{C_i}]: \mathcal{I}[\mathbf{Pa}_{c.A}] = \boldsymbol{u}} P(\mathcal{I}[c.A] \mid \boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}}) \right]$$

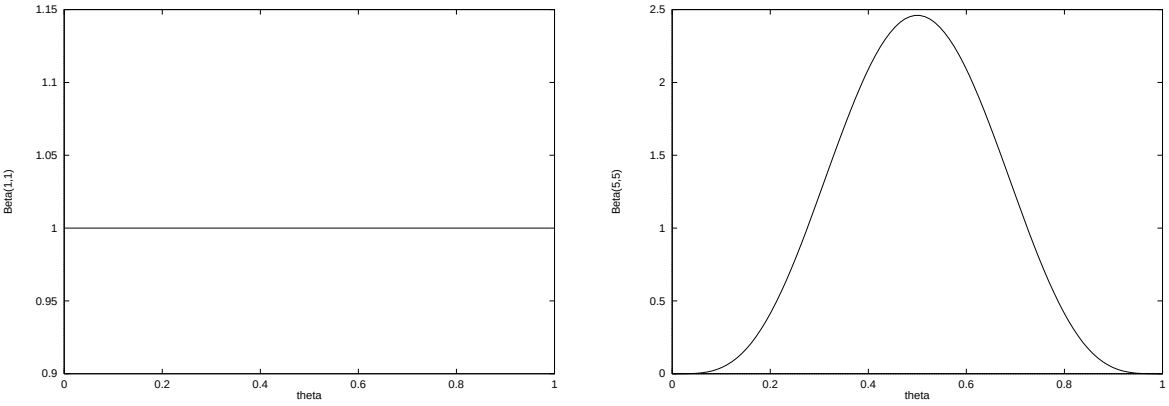
Thus, we see that using the above assumptions, the posterior distribution over parameters decomposes as a product of the posterior distributions of each attribute given a particular assignment to its parents.

As for the choice of priors, it is convenient to select *conjugate priors*, which are parametric distributions defined by *hyperparameters* such that the posterior distribution can also be parameterized by (different) hyperparameters. For discrete attributes $\mathsf{C_i}.A$, a common choice for the prior distribution, $\boldsymbol{\theta}_{\mathsf{C_i}.A}$, is a *Dirichlet* distribution, which is specified by a set of hyperparameters $\{\alpha[v] : v \in Val(\mathsf{C_i}.A)\}$. A distribution on the parameters of $P(\mathsf{C_i}.A)$ is Dirichlet if

$$P(\boldsymbol{\theta}_{\mathsf{C_i}.A}) = \frac{\Gamma(\sum_{v \in Val(\mathsf{C_i}.A)} \alpha[v])}{\prod_{v \in Val(\mathsf{C_i}.A)} \Gamma(\alpha[v])} \prod_{v \in Val(\mathsf{C_i}.A)} \theta_v^{\alpha[v]-1}$$

where $\Gamma(x)$ is the Gamma function defined as $\Gamma(x) = \int_0^\infty t^{x-1} e^{-x} dt$. For more details see (DeGroot, 1970). If $\mathsf{C_i}.A$ can take on k values, then the prior is:

$$P(\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}}) = Dir(\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}} \mid \alpha_1, \ldots, \alpha_k)$$



(a) (b)

Figure 2.9: Example of a Beta distribution. (a) Beta distribution for $\alpha_1 = 1, \alpha_2 = 1$. (b) Beta distribution for $\alpha_1 = 5, \beta_1 = 5$.

As mentioned above, the Dirichlet prior is a conjugate prior, meaning that the posterior also has a Dirichlet form:

$$P(\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}} \mid \mathcal{I}, \sigma, \mathcal{S}) = \\ Dir(\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}} \mid \alpha_{\mathsf{C_i}.A}[v_1, \boldsymbol{u}] + \hat{S}_{\mathsf{C_i}.A}[v_1, \boldsymbol{u}], \ldots, \alpha_{\mathsf{C_i}.A}[v_k, \boldsymbol{u}] + \hat{S}_{\mathsf{C_i}.A}[v_k, \boldsymbol{u}])$$

where $\hat{S}_{\mathsf{C_i}.A}[v, \boldsymbol{u}]$ is the sufficient statistics as defined in Equation 2.3. A special case of the Dirichlet distribution is when $\mathsf{C_i}.A$ is binary. In this case, the Dirichlet is called a *Beta* distribution and has the same form. An example of a Beta distribution is shown in Figure 2.9.

For continuous attributes $\mathsf{C_i}.A$ that we believe are normally distributed with mean μ and precision τ (precision is the inverse of the variance, i.e., $\tau = 1/\sigma^2$) such that $\mathsf{C_i}.A \sim N(\mu, \tau^{-1})$, a common choice for the prior distribution, $P(\mu, \tau)$, over the parameters μ and τ is the *Normal-Gamma* distribution. The mean μ and precision τ are said to follow a Normal-Gamma distribution with hyperparameters $\langle \mu_0, \lambda_0, \alpha_0, \beta_0 \rangle$, if the conditional distribution of μ given τ is a normal distribution with mean μ_0 and precision $\lambda_0 \tau$ ($-\infty < \mu_0 < \infty$ and $\lambda_0 > 0$); and if the marginal distribution of τ is a

Gamma distribution with parameters α_0 and β_0 ($\alpha_0 > 0$ and $\beta_0 > 0$), defined as:

$$P(\tau \mid \alpha_0, \beta_0) = \frac{{\beta_0}^{\alpha_0}}{\Gamma(\alpha_0)} \tau^{\alpha_0 - 1} e^{-\beta_0 \tau}$$

An example of a Normal Gamma distribution is shown in Figure 2.10. The Normal-Gamma prior is a conjugate prior to the normal distribution, so that the posterior also follows a Normal-Gamma distribution:

$$P(\boldsymbol{\theta}_{\mathsf{C_i}.A} \mid \mathcal{I}, \sigma, \mathcal{S}) = NG(\boldsymbol{\theta}_{\mathsf{C_i}.A} \mid \mu_1, \lambda_1, \alpha_1, \beta_1)$$

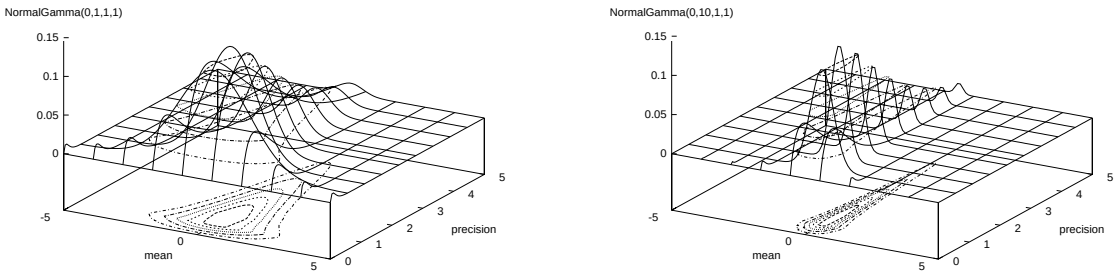
where:

$$\begin{aligned}
\lambda_1 &= \lambda_0 + \langle \hat{S}_{\mathsf{C_i}.A} \rangle_0 \\
\mu_1 &= \frac{\lambda_0 \mu_0 + \langle \hat{S}_{\mathsf{C_i}.A} \rangle_1}{\lambda_1} \\
\alpha_1 &= \alpha_0 + \frac{\langle \hat{S}_{\mathsf{C_i}.A} \rangle_0}{2} \\
\beta_1 &= \beta_0 + \frac{1}{2}\langle \hat{S}_{\mathsf{C_i}.A} \rangle_2 - \frac{1}{2}\langle \hat{S}_{\mathsf{C_i}.A} \rangle_1 + \frac{\langle \hat{S}_{\mathsf{C_i}.A} \rangle_0 \lambda_0 \left(\frac{\langle \hat{S}_{\mathsf{C_i}.A} \rangle_1}{\langle \hat{S}_{\mathsf{C_i}.A} \rangle_0} - \mu_0 \right)^2}{2\lambda_1}
\end{aligned}$$

and $\hat{S}_{\mathsf{C_i}.A}$ is the vector of sufficient statistics for the Gaussian distribution $\hat{S}_{\mathsf{C_i}.A} = \{\langle \hat{S}_{\mathsf{C_i}.A} \rangle_0, \langle \hat{S}_{\mathsf{C_i}.A} \rangle_1, \langle \hat{S}_{\mathsf{C_i}.A} \rangle_2\}$, such that:

$$\begin{aligned}
\langle \hat{S}_{\mathsf{C_i}.A} \rangle_0 &= \sum_{c \in \sigma[\mathsf{C_i}]} 1 \\
\langle \hat{S}_{\mathsf{C_i}.A} \rangle_1 &= \sum_{c \in \sigma[\mathsf{C_i}]} \mathcal{I}[c.A] \\
\langle \hat{S}_{\mathsf{C_i}.A} \rangle_2 &= \sum_{c \in \sigma[\mathsf{C_i}]} \mathcal{I}[c.A]^2
\end{aligned}$$

Now that we have the posterior, we can compute the probability of new data. In the case where the new instance is conditionally independent of the old instances given the parameter values (which is always the case in Bayesian network models,



(a) (b)

Figure 2.10: Example of a Normal-Gamma distribution. (a) Normal-Gamma distribution for $\mu_0 = 0, \lambda_0 = 1, \alpha_0 = 1, \beta_0 = 1$. (b) Normal-Gamma distribution for $\mu_0 = 0, \lambda_0 = 10, \alpha_0 = 1, \beta_0 = 1$.

but may not be true for PRMs), then the probability of the new data case can be conveniently rewritten using the expected parameters. We show this for the case of multinomial CPDs:

Proposition 2.6.1: *Assuming multinomial CPDs, global and local parameter independence, and Dirichlet priors, with hyperparameters* $\alpha_{\mathsf{C_i}.A}[v, \boldsymbol{u}]$*, we have that:*

$$E_\theta[P(\mathsf{C_i}.A = v \mid \mathbf{Pa}_{\mathsf{C_i}.A} = \boldsymbol{u}) \mid \mathcal{I}] = \frac{\alpha_{\mathsf{C_i}.A}[v, \boldsymbol{u}] + \hat{S}_{\mathsf{C_i}.A}[v, \boldsymbol{u}]}{\sum_{v' \in Val(\mathsf{C_i}.A)} \alpha_{\mathsf{C_i}.A}[v', \boldsymbol{u}] + \hat{S}_{\mathsf{C_i}.A}[v', \boldsymbol{u}]}$$

This suggests that the Bayesian estimate for $\boldsymbol{\theta}_{\mathcal{S}}$ should be estimated using this formula as well. Unfortunately, the expected parameters is not the proper Bayesian solution for computing probability of new data instance is not independent of previous data given the parameters. For example, suppose that we want to use the posterior to evaluate the probability of an instance $\mathcal{I}'$ of another skeleton σ'. If there are two objects c and c' of the class $\mathsf{C_i}$ such that $\mathcal{I}'[\mathbf{Pa}_{c.A}] = \mathcal{I}'[\mathbf{Pa}_{c'.A}]$, then we will be relying on the same multinomial parameter vector, $\boldsymbol{\theta}_{\mathsf{C_i}.A|\mathcal{I}'[\mathbf{Pa}_{c.A}]}$, twice. Using the chain rule, we see that the second probability depends on the posterior of the parameter after seeing the training data, *and* the first instance. However, if the posterior is sharply peaked (i.e., we have a strong prior, or we have seen many training instances), we can

approximate the solution using the expected parameters of Proposition 2.6.1. We use this approximation in our computation of the estimates for the parameters.

2.6.2 The Incomplete Data Case

Until now we have assumed that the training data is *fully observed*, meaning that the PRM instance $\mathcal{I}$ specifies a value for the attributes of all the objects in our skeleton. However, in the biological domain, this assumption is unrealistic, as many key variables are never observed. Moreover, in many of our settings, inferring the values of these *hidden* variables is a large part of our task. For example, the *module* that a gene belongs to and the *cluster* of an array are both hidden. Hence, we must address the learning problem in the presence of *incomplete* data. In this section we show how to learn the parameters of a PRM in the presence of incomplete data. In Section 2.7.2 we show how to learn both the structure and the parameters of a PRM from incomplete data. We note that the issues that arise from incomplete data in PRMs are similar to those that arise when learning Bayesian networks from incomplete data and the solutions we present here are similar to those that were devised in the Bayesian networks context.

Unfortunately, parameter estimation in the presence of incomplete data is a hard problem. To see why, recall that the key to our ability to efficiently estimate the parameters of a PRM was the decomposition of the log-likelihood function as a sum of terms corresponding to the various attributes of the different classes, as in Equation 2.4. However, when the values for some of the attributes are unobserved, we no longer get this decomposition. Rather, the likelihood function becomes the sum of likelihood functions, one for each possible completion of the missing values for the attributes, and hence, the log-likelihood function does not decompose by terms that correspond to the different attributes. Moreover, the number of possible completions is exponential in the total number of missing values and in the worst case, each of these completions can contribute to a different peak in the likelihood function. The total likelihood function can therefore be very complex, with multiple local maxima, as opposed to the likelihood function for the fully observed case, which has a single

global maximum.

In fact, even evaluating the likelihood function for a given set of parameters, $\boldsymbol{\theta}_{\mathcal{S}}$, is hard. If we denote the set of missing attributes by $\boldsymbol{H}$, then to evaluate the likelihood function, we need to compute:

$$P(\mathcal{I} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}) = \sum_{\boldsymbol{h} \in \boldsymbol{H}} P(\mathcal{I} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}, \boldsymbol{H} = \boldsymbol{h}) P(\boldsymbol{H} = \boldsymbol{h} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}})$$

which in the general case requires that we perform probabilistic inferences on the ground Bayesian network to compute $P(\boldsymbol{H} = \boldsymbol{h} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}})$. As we discussed earlier, such inferences may be intractable.

Due to this nature of the likelihood function in the incomplete data case, we cannot devise a general procedure for finding the parameter setting that will globally maximize the likelihood function. We thus resort to approaches that attempt to find a local maximum of this likelihood function. One approach is to use a general-purpose function optimization algorithm such as gradient ascent. An alternative approach, tailored specifically to the task of optimizing likelihood functions, is the *Expectation Maximization* (*EM*) algorithm (Dempster *et al.*, 1977). Starting with an initial setting to the parameters, $\boldsymbol{\theta}_{\mathcal{S}}^{(0)}$, the EM algorithm iterates between two steps, an *E-step*, and an *M-step*. In the E-step, we compute the posterior distribution over the unobserved attributes given the observed attributes and the current setting of the parameters, $\boldsymbol{\theta}_{\mathcal{S}}^{(t)}$. Thus, we compute $P(\boldsymbol{H} \mid \mathcal{I}, \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}^{(t)})$. This computation can be done by performing inference in the ground Bayesian network. From the E-step, we can compute the *expected* sufficient statistics for each attribute. For example, in the case of multinomial table CPDs, the expected sufficient statistics for each joint assignment $v \in \mathit{Val}(\mathsf{C}_{\mathsf{i}}.A), \boldsymbol{u} \in \mathit{Val}(\mathbf{Pa}_{\mathsf{C}_{\mathsf{i}}.A})$ is:

$$\hat{S}_{\mathsf{C}_{\mathsf{i}}.A}[v, \boldsymbol{u}] = \sum_{c \in \sigma[\mathsf{C}_{\mathsf{i}}]} P(c.A = v, \mathbf{Pa}_{c.A} = \boldsymbol{u} \mid \mathcal{I}, \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}^{(t)})$$

In the M-step, we then find the setting of the parameters that maximize the likelihood function relative to the distribution computed in the E-step. This step is identical to finding the optimal setting of the parameters for the case of fully observed data, except

that we now use the expected sufficient statistics for this maximization problem.

It is possible to define an energy function that is a lower bound on the log-likelihood function (Neal and Hinton, 1998), and show that each step of the EM algorithm maximizes this energy function. Thus, the EM algorithm converges to a local maximum of the log-likelihood function.

To apply the EM algorithm, we need to perform probabilistic inference in the E-step on the ground Bayesian network. When exact inference is possible, we can use one of the algorithms described in Section 2.5.1. However, as we discussed, in our setting exact inference is intractable. In these cases, we will resort to using one of the approximate inference algorithms described in Section 2.5.2, or use a tailored approximate inference algorithm that we design for a particular model we consider. Note that, if we resort to approximate inference, then from a theoretical perspective, we lose the convergence guarantees of the EM algorithm. However, in practice, as we shall see in subsequent chapters, when using the EM algorithm with an approximate inference for the E-step in our models, we do indeed converge to reasonably good points of the log-likelihood function. One explanation might have to do with the fact that the M-step only relies on the expected sufficient statistics and therefore, even though the approximate inference algorithm we use might compute wrong marginals for individual variables, on average these errors might cancel out in the expected sufficient statistics.

For some models, it is easier to design approximate inference algorithms for finding the maximal assignment to the unobserved attributes, rather than for finding their posterior distribution. In these cases, it is convenient to use the *hard assignment* variant of the EM algorithm. In this variant, rather than computing the posterior distribution over the hidden variables in the E-step, we attempt to find their maximal assignment:

$$\boldsymbol{h}^* = \text{argmax}_{\boldsymbol{h} \in H} P(\boldsymbol{H} = \boldsymbol{h} \mid \mathcal{I}, \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}^{(t)})$$

We then treat this assignment to the missing attributes as if it were observed data, and proceed with computing the sufficient statistics and parameter estimation as

in the case of fully observed data. It can be shown that this *hard assignment EM* algorithm is guaranteed to converge to a local maximum of the log-likelihood of the completed data, $P(\mathcal{I}, \boldsymbol{H} = \boldsymbol{h} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}})$, where $\boldsymbol{h}$ represents a complete assignment to the missing attributes $\boldsymbol{H}$.

Finally, a practical consideration when applying the EM algorithm has to do with the initial setting of the parameters. As with many algorithms that find only local maxima, EM is also sensitive to this choice of starting point. We therefore take care in choosing this starting point, and try to use clever heuristics for this initialization. Rather than choosing the parameters directly, our approach will usually be to select a heuristic that somehow complete the assignment to the hidden variables. We then initialize the parameters to the setting that maximizes the log-likelihood function relative to this completion in the heuristic phase.

2.7 Structure Learning

We now move to the more challenging problem of learning a dependency structure automatically, as opposed to having it be given by the user. There are three important issues that need to be addressed. We must determine which dependency structures are legal; we need to evaluate the "goodness" of different candidate structures; and we need to define an effective search procedure that finds a good structure. By legal structures, we mean structures that given any instantiation of the schema, will always induce a coherent probability distribution. This will always be the case if the induced ground Bayesian network is acyclic. While in general determining whether a schema induces a legal structure must be handled with care (Getoor, 2001), in our setting we will limit the possible structures we can consider such that we do not need to worry about acyclicity. We therefore focus our discussion on evaluating structures and searching for a good structure.

2.7.1 The Complete Data Case

We start with the problem of learning dependency structures for the case of fully observed data.

Evaluating different structures

We use the Bayesian *model selection* methods to evaluate structures. That is, we would like to find the MAP (maximum a posteriori) structure. Formally, we want to compute the posterior probability of a structure $\mathcal{S}$ given an instantiation $\mathcal{I}$. Using Bayes rule we have that

$$P(\mathcal{S} \mid \mathcal{I}, \sigma) \propto P(\mathcal{I} \mid \mathcal{S}, \sigma) P(\mathcal{S} \mid \sigma)$$

This score is composed of two parts: the prior probability of the structure, and the probability of the data assuming that structure.

The first component is $P(\mathcal{S} \mid \sigma)$, which defines a prior over structures. We assume that the choice of structure is independent of the skeleton, and thus $P(\mathcal{S} \mid \sigma) = P(\mathcal{S})$. In the context of Bayesian networks, we often use a simple uniform prior over possible dependency structures. In the case of general PRMs, this assumption may not work, as there may be infinitely many possible structures (Getoor, 2001). However, in the models we consider, the number of possible models is finite and we can usually utilize such a uniform prior over structures. Thus, $P(\mathcal{S})$ will usually not influence our choice of structures and we can ignore this term in evaluating structures.

The second component is the *marginal likelihood*:

$$P(\mathcal{I} \mid \mathcal{S}, \sigma) = \int P(\mathcal{I} \mid \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}, \sigma) P(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{S}) d\boldsymbol{\theta}_{\mathcal{S}}$$

If we use a prior $P(\boldsymbol{\theta}_{\mathcal{S}} \mid \mathcal{S})$ that satisfies global and local parameter independence, as defined in Equation 2.5 and Equation 2.6, respectively, then the marginal likelihood decomposes into a product of integrals:

$$P(\mathcal{I} \mid \mathcal{S}, \sigma) \;=\; \prod_{\mathsf{C}_\mathsf{i}} \prod_{A \in \mathcal{A}[\mathsf{C}_\mathsf{i}]} \prod_{\boldsymbol{u} \in Val(\mathbf{Pa}_{\mathsf{C}_\mathsf{i}})}$$

$$\int P(\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}} \mid \mathcal{S}) \prod_{c \in \sigma[\mathsf{C_i}]: \mathcal{I}[\mathbf{Pa}_{\mathsf{C_i}}]=\boldsymbol{u}} P(\mathcal{I}[c.A] \mid \mathcal{I}[\mathbf{Pa}_{c.A}], \boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}}) d\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}}$$

For discrete variables $\mathsf{C_i}.A$ with multinomial table CPDs, and Dirichlet priors, with hyperparameters $\alpha_{\mathsf{C_i}.A}[v, \boldsymbol{u}]$, the corresponding integral in the decomposed marginal likelihood can be computed as:

$$\int P(\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}} \mid \mathcal{S}) \prod_{c \in \sigma[\mathsf{C_i}]: \mathcal{I}[\mathbf{Pa}_{\mathsf{C_i}}]=\boldsymbol{u}} P(\mathcal{I}[c.A] \mid \mathcal{I}[\mathbf{Pa}_{c.A}], \boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}}) d\boldsymbol{\theta}_{\mathsf{C_i}.A|\boldsymbol{u}} =$$
$$\frac{\Gamma(\sum_{v \in Val(\mathsf{C_i}.A)} \alpha_{\mathsf{C_i}.A}[v, \boldsymbol{u}])}{\prod_{v \in Val(\mathsf{C_i}.A)} \Gamma(\alpha_{\mathsf{C_i}.A}[v, \boldsymbol{u}])} \frac{\prod_{v \in Val(\mathsf{C_i}.A)} \Gamma(\hat{S}[v, \boldsymbol{u}] + \alpha_{\mathsf{C_i}.A}[v, \boldsymbol{u}])}{\Gamma(\sum_{v \in Val(\mathsf{C_i}.A)} \hat{S}[v, \boldsymbol{u}] + \alpha_{\mathsf{C_i}.A}[v, \boldsymbol{u}])}$$

where $\hat{S}[v, \boldsymbol{u}]$ is the sufficient statistics for the joint assignment v and $\boldsymbol{u}$, as defined in Equation 2.3. We show the full derivation of this formula in Appendix A.

For continuous variables $\mathsf{C_i}.A$ with Gaussian CPDs and Normal-Gamma prior distributions, the corresponding integral also has a simple closed form formula. We show this formula and its derivation in Appendix B.

The marginal likelihood term is the dominant term in the probability of a structure. It balances the complexity of the structure with its fit to the data. This balance can be seen explicitly in the asymptotic relation of the marginal likelihood to explicit penalization, such as the MDL score (see, e.g., Heckerman (1998)).

Structure search

Having defined a scoring function for evaluating different structures, we need only provide a procedure for finding high scoring structures. For Bayesian networks, we know that finding the highest scoring structure is NP-hard (Chickering, 1996). As PRM learning is at least as hard as Bayesian network learning (a Bayesian network is simply a PRM with one class and no relations), we cannot hope to find an efficient procedure that always finds the highest scoring structure. Thus, we must resort to heuristic search.

As is standard in Bayesian network learning (Heckerman, 1998), we use a greedy

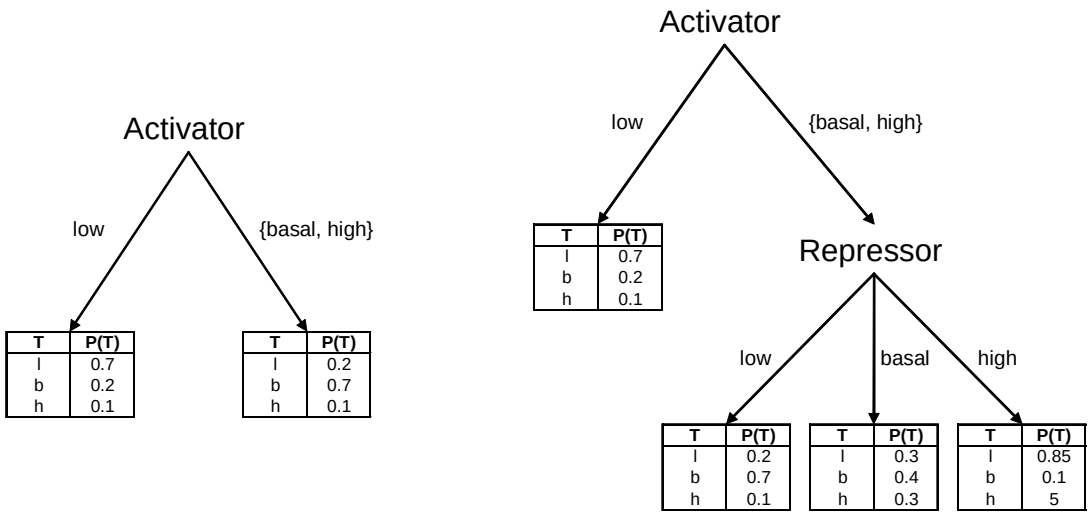


Figure 2.11: An example of the split operator in learning tree CPDs, for a tree CPD representing the conditional distribution of the *Target* attribute given the *Activator* and *Repressor* attributes. (a) The tree CPD before performing the split operator on the *Repressor* attribute. (b) The tree CPD after splitting on the *Repressor* attribute.

local search procedure that maintains a "current" candidate structure and iteratively modifies it to increase the score. At each iteration, we consider a set of simple local transformations to the current structure, score all of them, and pick the one with the highest score. When learning multinomial or Gaussian table CPDs, the three operators we use are: *add edge*, *delete edge* and *reverse edge*. When learning tree CPDs, following (Chickering *et al.*, 1997), our operators consider only transformations to the CPD-trees. The tree structure induces the dependency structure, as the parents of $\mathsf{C}_i.A$ are simply those attributes that appear in its CPD-tree. In this case, the two operators we use are: *split* — replaces a leaf in a CPD tree by an internal node with two leaves; and *trim* — replaces the subtree at an internal nodes by a single leaf. A simple example of the split operator is shown in Figure 2.11.

The simplest heuristic search algorithm over this set of operators is a greedy hill-climbing search, using the Bayesian score as the evaluation metric. We maintain our current candidate structure and iteratively improve it. At each iteration, we consider the appropriate set of local transformations to that structure, score all of them, and

pick the one with the highest score.

We refer to this simple algorithm as the greedy algorithm. There are several common variants to improve the robustness of the hill-climbing methods. One is to make use of random restarts to deal with local maxima. In this algorithm, when we reach a maximum, we take some fixed number of random steps, and then we restart our search process. Another common approach is to make use of a tabu-list, which keeps track of the most recent operators applied, and allows only steps which do not reverse the effect of recently applied operators. A more sophisticated approach is to employ a simulated annealing type of algorithm which uses the following procedure: in early phases of the search we are likely to take random steps (rather than the best step), but as the search proceeds (i.e., the temperature cools) we are less likely to take random steps and more likely to take the best greedy step. Another common search strategy is *beam search*, which always maintains k structures in parallel. In each step, we then select the k highest scoring structures when considering local transformations to any of the k current structures. Finally, rather than scoring each structure by evaluating its score after applying a single local transformation step to it, the *lookahead* search strategy scores each model by considering the score of the structure obtained after applying the proposed transformation and ℓ additional transformations, where these additional transformations are usually selected by taking the next ℓ best greedy steps.

Regardless of the specific heuristic search algorithm used, an important component of the search is the scoring of candidate structures. As in Bayesian networks, the decomposability property of the score has significant impact on the computational efficiency of the search algorithm. First, we decompose the score into a sum of *local scores* corresponding to individual attributes and their parents. Now, if our search algorithm considers a modification to our current structure where the parent set of a single attribute $\mathsf{C}_\mathsf{i}.A$ is different, only the component of the score associated with $\mathsf{C}_\mathsf{i}.A$ will change. Thus, we need only reevaluate this particular component, leaving the others unchanged; this results in major computational savings.

We note that when performing a general search for PRM structures (Getoor, 2001), one issue that needs to be addressed is the selection of the set of potential

candidate parents, since in theory this number can be very large for long slot chains. However, in the models we consider, we will usually restrict the length of the slot chains, which limits the potential candidate parents and avoids this issue. We refer the reader to Getoor (2001) for details on addressing this issue for the general case of structure learning in PRMs.

2.7.2 The Incomplete Data Case

As in the parameter estimation problem, we also need to address the structure learning problem in the context of incomplete data. This problem is hard, as when the values for some attributes is missing, the structure score no longer decomposes as a product of separate integrals for each attribute and its parents. Thus, we lose the decomposability property and with that our ability to provide a simple closed form solution for evaluating a candidate structure.

To address these computational issues when learning Bayesian networks from incomplete data, Friedman (1998) suggested the *Structural EM* algorithm. The idea behind this approach is that rather than applying the EM algorithm to each candidate structure and use the parameters it finds to compute the expected sufficient statistics, we use the current structure to compute the expected sufficient statistics for all candidate structures we wish to consider. The key insight is that during our search for structures, we typically only consider local transformations to the current structure. Thus, our current structure and parameters might be good enough for approximating the expected sufficient statistics of a candidate structure we consider. In fact, we can use the same structure to compute sufficient statistics for several search steps, thereby incurring additional computational savings. After several such steps, we can apply the EM algorithm to the current structure we arrived at, and use the parameters it finds for computing the expected sufficient statistics needed for the following search steps. It can be shown that the structural EM algorithm is guaranteed to converge to a local maximum (Friedman, 1998).

We also consider a *hard assignment structural EM* variant. Similar to the hard assignment EM algorithm for parameter estimation described in Section 2.6.2, in the

hard assignment structural EM variant, we use the current structure to compute a hard assignment to all missing attributes:

$$\boldsymbol{h}^* = \text{argmax}_{\boldsymbol{h} \in H} P(\boldsymbol{H} = \boldsymbol{h} \mid \mathcal{I}, \sigma, \mathcal{S})$$

where $\boldsymbol{h}^*$ is the set of unobserved attributes and $\mathcal{S}$ represents the current structure. We then treat this completion as if it were observed data, and use it for the structure search, which can now be performed efficiently relative to the completed data. It can be shown that this algorithm is guaranteed to converge to a local maximum of $P(\mathcal{S}, \boldsymbol{H} = \boldsymbol{h} \mid \sigma, \mathcal{I})$. To see why this is true, note that

$$\begin{aligned} P(\mathcal{S}, \boldsymbol{H} = \boldsymbol{h} \mid \mathcal{I}, \sigma) &= P(\mathcal{S} \mid \boldsymbol{H} = \boldsymbol{h}, \mathcal{I}, \sigma) P(\boldsymbol{H} = \boldsymbol{h} \mid \mathcal{I}, \sigma) \\ &= P(\boldsymbol{H} = \boldsymbol{h} \mid \mathcal{S}, \mathcal{I}, \sigma) P(\mathcal{S} \mid \mathcal{I}, \sigma) \end{aligned}$$

The data completion step improves $P(\mathcal{S}, \boldsymbol{H} = \boldsymbol{h} \mid \mathcal{I}, \sigma)$ by optimizing $P(\boldsymbol{H} = \boldsymbol{h} \mid \mathcal{S}, \mathcal{I}, \sigma)$, while leaving $P(\mathcal{S} \mid \mathcal{I}, \sigma)$ unchanged (since the structure $\mathcal{S}$ is fixed during the data completion step). The structure search step also improves $P(\mathcal{S}, \boldsymbol{H} = \boldsymbol{h} \mid \mathcal{I}, \sigma)$ by optimizing $P(\mathcal{S} \mid \ \boldsymbol{H} = \boldsymbol{h}, \mathcal{I}, \sigma)$, while leaving $P(\boldsymbol{H} = \boldsymbol{h} \mid \mathcal{I}, \sigma)$ unchanged (since the completion of the data $\boldsymbol{H} = \boldsymbol{h}$ is fixed during the structure search).

Note, however, that in general it is hard to find the assignment $\boldsymbol{h}$ to $\boldsymbol{H}$ that globally maximizes $P(\boldsymbol{H} = \boldsymbol{h} \mid \mathcal{S}, \mathcal{I}, \sigma)$, since the various assignments are dependent through the unknown parameters. In this case, we can break the data completion into several steps, where in each step we fix a subset of the variables in $\boldsymbol{H}$ and optimize the assignment to the remaining variables given this fixed assignment. Since each such step still optimizes $P(\mathcal{S}, \boldsymbol{H} = \boldsymbol{h})$, this procedure also has convergence guarantees, though it may converge to a weaker local maximum as compared to the local maximum obtained by finding the jointly optimal assignment to all the missing attributes.

An advantage of this hard assignment structural EM variant is that for some models it is easier to find a hard assignment to the missing attributes, as this entire problem can be viewed as a discrete search space in which we are searching for an

optimal structure and assignment. In some models, we can exploit certain structural properties for efficiently traversing this search space. A disadvantage is that the hard assignment approach loses information relative to the soft assignments and might make arbitrary choices in those cases in which several completions have similar probabilities.

2.8 Conclusions

In this chapter, we have reviewed the definition of probabilistic relational models (PRMs), originally introduced by Koller and Pfeffer Koller and Pfeffer (1998). PRMs exploit both the compact representation of a probability distribution by a Bayesian network with the expanded representational power of relational logic. Together, these allow us to define compact representations for joint distributions over objects in structured domains in general and in the biological domain in particular. Here we defined the major components of the PRM model: the relational schema, the probabilistic dependency structure and parameterization, and the relational skeleton. We also outlined several approaches for performing inference in PRMs and discussed the computational challenges involved. Finally, we showed how to estimate the parameters and induce the structure of a PRM automatically from data, both in the case of fully observed data, and in the more realistic case of incomplete data. In the next chapters we show the utility of PRMs for solving several problems in gene regulation, and present extensions to the basic framework presented here, both on the representational level and on the algorithmic level of learning parameters and structure of a PRM automatically from data.

Chapter 3

Discovering *cis*-Regulatory Modules

A central goal of molecular biology is the discovery of the regulatory mechanisms governing the expression of genes in the cell. The expression of a gene is controlled by many mechanisms. A key junction in these mechanisms is mRNA transcription regulation by various proteins, known as *transcription factors* (TFs), that bind to specific sites in the promoter region of a gene and activate or inhibit transcription. Loosely speaking, we can view the promoter region as an encoding of a "program", whose "execution" leads to the expression of different genes at different points in time and in different conditions. To a first-order approximation, this "program" is encoded by the presence or absence of TF binding sites within the promoter.

In this chapter, we attempt to reveal precisely this aspect of the gene regulation story. We describe a probabilistic model for understanding transcriptional regulation using both gene expression and promoter sequence data. We aim to identify *transcriptional modules* — sets of genes that are co-regulated in a set of microarray experiments, through a common combination of motifs, which we term a *motif profile*. That is, given a gene expression data set as input, and promoter sequences of genes, we aim to identify modules of co-regulated genes and *explain* the observed expression patterns of each module via a motif profile that is common to genes in the same module (see Figure 3.1). Our goal is to provide a genome-wide explanation of the

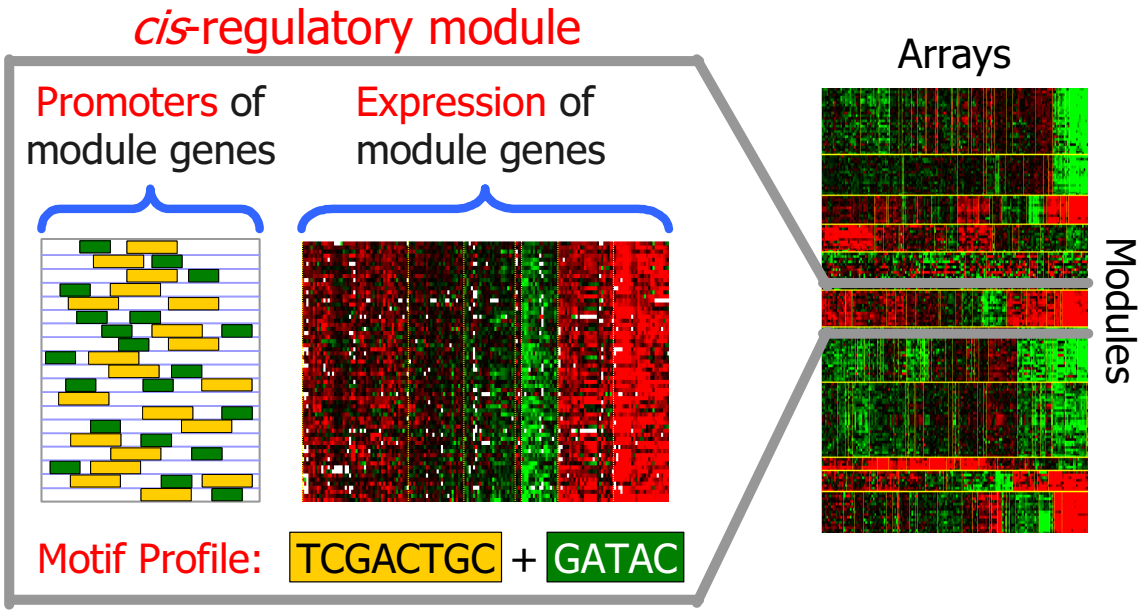


Figure 3.1: Overall goal of our approach: given an expression data set and promoter regions of genes, our goal is to identify modules of co-regulated genes (each module is depicted as a horizontal strip in the expression matrix on the right). For each module, we also wish to identify the combination of motifs that play a role in controlling the expression of the module genes. In this example, the motif profile consists of two motifs, and their hypothetical appearances along the promoter of genes is shown.

expression data.

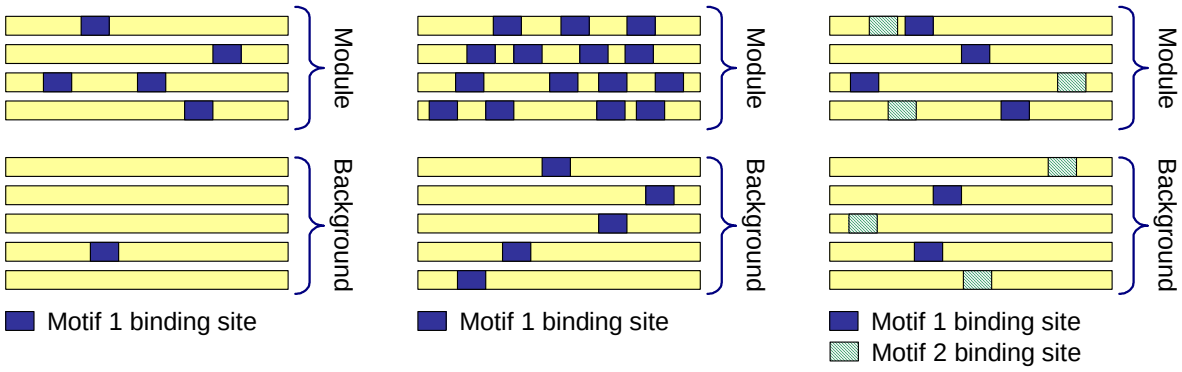
We start with an overview of the key ideas underlying our approach, followed by a formal presentation of the probabilistic model and our algorithm for automatically learning the model from the input gene expression and promoter sequence data. We then present detailed statistical and biological evaluation of the results we obtained when applying the method to two yeast (*S. Cerevisiae*) expression datasets, demonstrating that our approach is better than a standard one at recovering known motifs and at generating biologically coherent modules. Finally, we also combine our results with binding location data to obtain regulatory relationships with known transcription factors, and show that many of the inferred relationships have support in the literature.

3.1 Model Overview

Many cellular processes are regulated at the transcriptional level, by one or more transcription factors that bind to short DNA sequence motifs in the upstream regions of the process genes. These co-regulated genes then exhibit similar patterns of expression. Given the upstream regions of all genes, and measurements of their expression under various conditions, we could hope to "reverse engineer" the underlying regulatory mechanisms and identify *transcriptional modules* — sets of genes that are co-regulated under these conditions through a common motif or combination of motifs.

We take a genome-wide approach for discovering this modular organization, based on the premise that transcriptional elements should "explain" the observed expression patterns as much as possible. We define a probabilistic model which integrates both the gene expression measurements and the DNA sequence data into a unified model. The model assumes that genes are partitioned into modules, which determine the gene's expression profile. Each module is characterized by a *motif profile*, which specifies the relevance of different sequence motifs to the module. A gene's module assignment is a function of the sequence motifs in its promoter region. This motif profile is designed such that it can represent the various *promoter architectures* that are known to be involved in gene regulation. These include regulation through a single binding site for a single transcription factor (see Figure 3.2(a)), regulation through multiple sites for a single transcription factor (see Figure 3.2(b)), and combinatorial regulation through multiple sites for multiple transcription factors (see Figure 3.2(c)).

Our model thus defines a "recipe" for how genes are expressed: the combination of motifs that appear in the promoter of a gene define its motif profile, which in turn defines an expression profile across all experiments. A potential problem with this recipe is that motifs, by being short sequences, may be randomly present in promoters of some genes without playing a role in modulating the expression of the gene (e.g., because the DNA in the region is not accessible to binding by transcription factors). To address this problem, our model does not assume that all motifs are necessarily *active* in all the genes in which they appear. Furthermore, our goal is to discover



(a) (b) (c)

Figure 3.2: Shown are possible promoter architectures of co-regulated genes. For each case, also shown is the worst case scenario, from a computational perspective, of promoter architectures that can be present in promoters of background genes. (a) Single motif. (b) Single motif with multiple binding sites. (c) Motif combinations.

motifs that play a regulatory role in some particular set of experiments; a motif that is active in some settings may be completely irrelevant in others. Our model thus identifies *motif targets* — genes where the motif plays an active role in affecting regulation in a particular expression data set. These motif targets are genes that have the motif and that are assigned to modules containing the motif in their profile.

Our algorithm is outlined in Figure 3.3. It begins by clustering the expression data, creating one *module* from each of the resulting clusters. As the first attempt towards explaining these expression patterns, it searches for a common motif in the upstream regions of genes assigned to the same module. It then iteratively refines the model, trying to optimize the extent to which the expression profile can be predicted transcriptionally. For example, we might want to move a gene g whose promoter region does not match its current module's motif profile, to another module whose expression profile is still a good match, and whose motif profile is much closer. Given these assignments, we could then learn better motif models and motif profiles for each module. This refinement process arises naturally within our algorithm, as a byproduct of the expectation maximization (EM) algorithm for estimating the model parameters.

In general, the motifs learned will not suffice to characterize all of the modules.

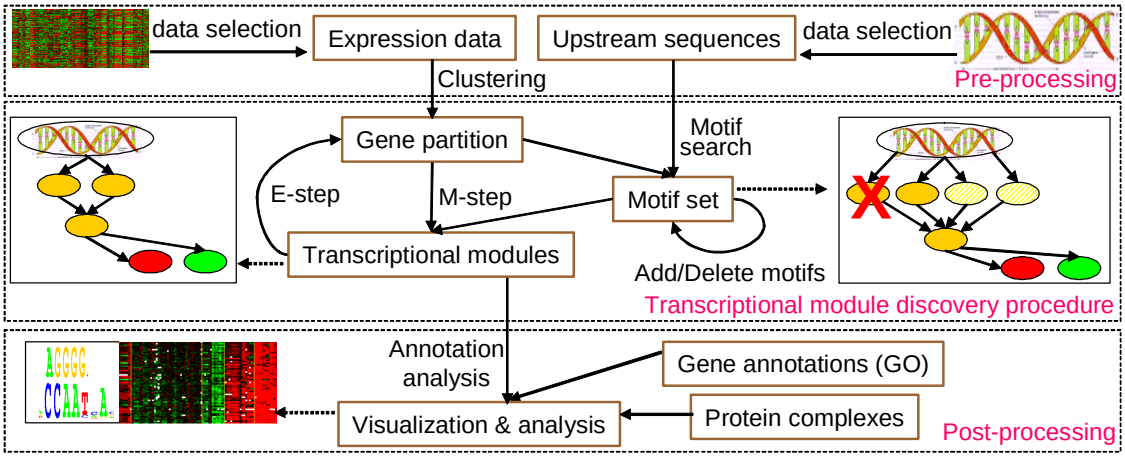


Figure 3.3: Schematic flow diagram of our proposed method. The pre-processing step includes selecting the input gene expression and upstream sequence data. The model is then trained using EM, and our algorithm for dynamically adding and deleting motifs. It is then evaluated on additional data sets.

As our goal is to provide a genome-wide explanation of the expression behavior, our algorithm identifies poorly explained genes in modules and searches for new motifs in their upstream regions. The new motifs are then added to the model and subsequently refined using EM. As part of this dynamic learning procedure, some motifs may become obsolete and are removed from the model. The algorithm iterates until convergence, adding and deleting motifs, and refining motif models and module assignments.

The probabilistic model we develop provides a unified framework that models both the expression and the sequence data. The model consists of three main components: a *motif model*, which describes how individual motifs are defined as patterns in the promoter sequences; a *regulation model*, which describes how the individual motifs assemble into motif profiles that we associate with each module; and a *gene expression model*, which describes the unique expression profile that we associate with each module or motif profile. A schematic diagram of our unified model is shown in Figure 3.4.

A key property of our approach is that these three components are part of a single probabilistic model, and are trained together, to achieve maximum predictiveness.

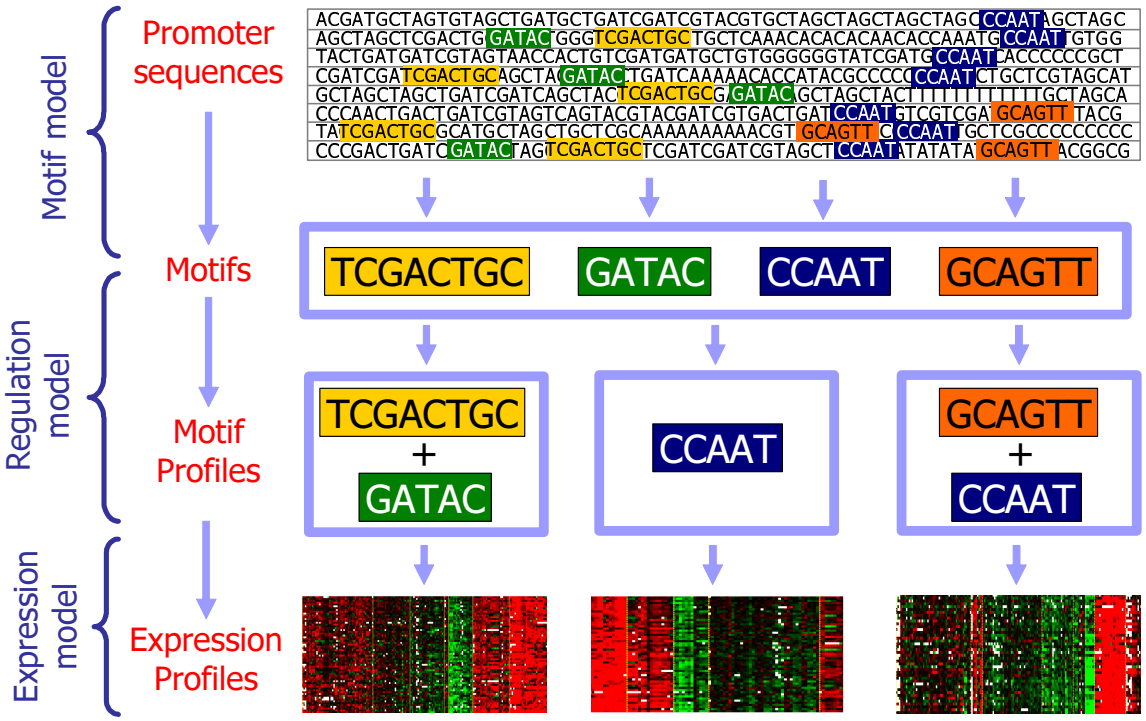


Figure 3.4: Schematic diagram of our unified model, consisting of three components: a *binding component* describes individual motifs; a *regulation component* describes motif profiles that are associated with modules; and an *expression component* describes a unique pattern of expression for each module across all experiments. The example shows four distinct motifs that assemble into three motif profiles, two of which have two motifs each and one with a single motif.

Our algorithm thereby simultaneously discovers motifs that are predictive of gene expression, and discovers clusters of genes whose behavior is well-explained by putative regulation events.

We note that the binding component uses a novel discriminative approach for finding motifs, unlike most other *generative* approaches which try to build a model of the promoter region sequence that gives a high probability to the given sequences. Such approaches can often be confused by repetitive motifs that occur in many promoter sequences (e.g., TATA boxes which appear in the promoter region of many genes). These motifs have to be filtered out by using an appropriate background distribution (Tavazoie *et al.*, 1999). In contrast, our discriminative formulation of

the motif finding task avoids many of the problems associated with modeling of the background sequence distribution. These are key problems that arise in many of the generative motif finding approaches. Thus, this part of the model, described in detail in Section 3.2.1, is interesting in and of itself as it can be used as an alternative to existing motif finding programs.

3.2 Probabilistic Model

We now provide a formal definition of our probabilistic model within the PRM framework outlined in Chapter 2. Recall that to specify a PRM, we first need to specify a relational schema that includes the classes of objects in our domain and the descriptive attributes associated with each class. In our gene regulation model, our basic entities are clearly genes, their corresponding promoter sequences, arrays, and expression measurements. Consequently, the set of classes we define are Gene,Promoter, Array, and Expression.

The descriptive attributes for the Promoter class include N random variables, $S_1, \ldots, S_N$, that represent the nucleotides in each position of the promoter sequence. Each of these S_i variables can take on values that correspond to the four possible nucleotides and thus its domain is $Val(S_i) = \{A, C, G, T\}$. We assume that there is a total of L motifs through which regulation is achieved in the cell and associate a set of binary-valued *Regulates* variables $\mathbf{R} = \{R_1, \ldots, R_L\}$, where $g.R_i$ takes the value *true* if motif i appears in the promoter region of gene g, allowing the motif to play a regulatory role in controlling g's expression.

We also assume that the genes are partitioned into a set of K mutually exclusive and exhaustive *cis-regulatory modules*. Thus, each gene is associated with an attribute $\mathbf{M} \in \{1, \ldots, K\}$ whose value represents the module to which the gene belongs. In the Expression class, we represent its real-valued mRNA expression level measurement by a continuous variable *Level*. Finally, since we want to represent the expression profile for each module by an expression pattern across all arrays, we associate a variable *ID* with each Array class. The full PRM relational schema is shown in Figure 3.5.

The second component in a PRM includes the reference slots $\mathcal{R}[\mathsf{C}]$ for each class

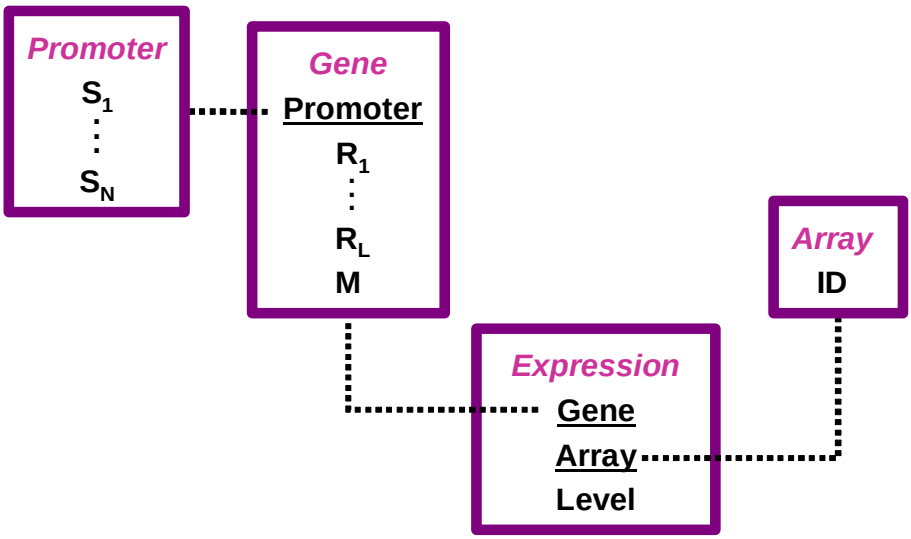


Figure 3.5: PRM relational schema for the cis-regulation model, including the classes, descriptive attributes and Reference slots (underlined). Each S_i variable in the Promoter class represents the nucleotide in position i of the promoter sequence; each R_i variable in the Gene class represents whether motif i appears in the promoter of the gene; the $\mathbf{M}$ variable in the Gene class represents the module assignment of the gene; the *Level* variable in the Expression class represents the real-valued mRNA expression measured for the corresponding gene and array; and the *ID* variable of the Array class represents a unique identifier for each array.

C. These allows an object to refer to other objects. Recall that each reference slot is typed, i.e., we also specify the type of object that may be referenced. In our model, the class Gene has a reference slot Promoter (with range type Promoter); the class Expression has reference slots Gene (with range type Gene) and Array (with range type Array). In Figure 3.5 the reference slots are underlined. Recall that we also defined the notion of a slot chain, which allows us to compose slots, defining functions from objects to other objects to which they are indirectly related. For example, in our model it may be useful to define the slot chain Expression.Gene.Promoter to refer to the promoter sequence of the gene from which the expression measurement was taken.

To complete the PRM specification, we also need to specify the qualitative structure $\mathcal{S}$ and the parameters $\boldsymbol{\theta}_{\mathcal{S}}$ associated with the CPD of each of the descriptive attributes. As we mentioned above, the structure consists of three main components.

We now present the details of each of these components, and then show how they assemble together into our unified probabilistic model over sequence and expression.

3.2.1 Motif Model: A Discriminative Motif Finder

The first part of our model relates the promoter region sequence data to the *Regulates* variables, **R**. We note that this part of the model can be applied independently, as an alternative to existing motif finding programs.

Experimental biology has shown that transcription factors bind to relatively short sequences and that there can be some variability in the binding site sequences. Thus, most standard approaches to uncovering transcription factor binding sites (e.g., Bailey and Elkan (1994), Roth *et al.* (1998), Sinha and Tompa (2000)), search for relatively short sequence motifs in the bound promoter sequences.

A common way of representing the variability within the binding site is by using a *position specific scoring matrix* (PSSM). Suppose we are searching for motifs of length k (or less). A PSSM $\vec{w}$ is a $k \times 4$ matrix, assigning for each position $i = 1, \ldots, k$ and letter $l \in \{A, C, G, T\}$ a weight $w_i[l]$. We can then score a putative k-mer binding site $S = S_1, \ldots, S_k$ by computing $\text{PSSM}(S, \vec{w}) = \sum_i w_i[S_i]$.

The question is how to learn these PSSM weights. We first define the model more carefully. Recall that our model associates with each gene g, a binary variable $g.R_i \in \{\mathit{false}, \mathit{true}\}$ which denotes whether motif i regulates the gene or not. Furthermore, recall that each gene g has a promoter sequence $g.\mathsf{Promoter}.S_1, \ldots, g.\mathsf{Promoter}.S_N$, where each S_i variable takes values from $\{A, C, G, T\}$. To simplify notation in this section, we focus attention on the regulation for a particular motif, and drop the explicit reference to i from our notation. As the motif model we present in this section deals only with Gene objects, we also omit explicit references to gene and promoters. Thus, our notation in this section simply refers to the various variables as $S_1, \ldots, S_N$, and R.

The standard approaches to learning PSSMs is by training a probabilistic model of binding sites that maximizes the likelihood of sequences (given the assignment of the regulates variables; see Bailey and Elkan (1994), Roth *et al.* (1998)). These

approaches rely on a clear probabilistic semantics of PSSM scores. We denote by θ_0 the probability distribution over nucleotides according to the background model. For simplicity, we use a Markov process of order zero for the background distribution. (As we will see, the choice of background model is not crucial in the discriminative model we develop.) We use ψ_j to denote the distribution of characters in the j-th position of the binding site. The model then assumes that if the motif regulates g, then g's promoter sequence has the background distribution for every position in the promoter sequence, except for a specific k-mer, $i, \ldots, i+k-1$, where a transcription factor binds the motif. If the motif does not regulate g, we have the background distribution for all positions in the sequence. If we assume a uniform prior over the binding position within the promoter in the case of regulation, then we get:

$$
\begin{aligned}
&P(S_1, \ldots, S_N \mid R = \mathit{false}) = \prod_{i=1}^{N} \theta_0[S_i] \\
&P(S_1, \ldots, S_N \mid R = \mathit{true}) \\
&= \sum_{i=1}^{N-k+1} \frac{1}{N-k+1} \left(\prod_{j=1}^{i-1} \theta_0[S_j] \right) \left(\prod_{j=1}^{k} \psi_j[S_{j+i-1}] \right) \left(\prod_{j=i+k}^{N} \theta_0[S_j] \right) \\
&= \frac{1}{N-k+1} \sum_{i=1}^{N-k+1} \left(\prod_{j=1}^{N} \theta_0[S_j] \right) \left(\prod_{j=1}^{k} \frac{\psi_j[S_{j+i-1}]}{\theta_0[S_{j+i-1}]} \right) \\
&= \frac{\prod_{j=1}^{N} \theta_0[S_j]}{N-k+1} \sum_{i=1}^{N-k+1} \prod_{j=1}^{k} \frac{\psi_j[S_{j+i-1}]}{\theta_0[S_{j+i-1}]}
\end{aligned}
$$

The probabilistic approaches estimate the θ_0 parameters from the sequence data and train the parameters $\psi_i[l]$ so as to maximize the probability of the training sequences, using for example iterative methods such as EM. Once these parameters are found, they set the PSSM weights to $w_i[l] = \log \frac{\psi_i[l]}{\theta_0[l]}$. Such approaches are *generative*, in the sense that they try to build a model of the promoter region sequence, and training succeeds when the model gives the given sequences high probability. However, these approaches can often be confused by repetitive motifs that occur in many promoter sequences. These motifs have to be filtered out by using an appropriate background distribution (Tavazoie *et al.* (1999)). Selecting this background distribution is difficult and it was shown (e.g., by Liu *et al.* (2001)) that the results obtained

are highly dependent upon this choice.

We take an entirely different approach. Recall that our aim is to model the dependence of the gene's genomic expression profile on its promoter sequence. For this task, we do *not* need to model the sequence; we need only estimate the probability that the motif regulates the gene given the promoter region. Thus, it suffices to find motifs that *discriminate* between promoter regions where the transcription factor binds the motif and those where it does not. As we show, this more directed goal allows us to avoid the problem of learning background distribution of promoters and to focus on the classification task at hand.

More formally, we are only interested in the conditional probability of R given the sequence $S_1, \ldots, S_N$. If we have a model of the form described above, then by applying Bayes rule, we obtain:

$$\begin{aligned}
P(R = \mathit{true} \mid S_1, \ldots, S_N) &= \frac{P(S_1, \ldots, S_N \mid R = \mathit{true})P(R = \mathit{true})}{P(S_1, \ldots, S_N)} \\
&= \frac{P(S_1, \ldots, S_N \mid R = \mathit{true})P(R = \mathit{true})}{P(S_1, \ldots, S_N \mid R = \mathit{true})P(R = \mathit{true}) + P(S_1, \ldots, S_N \mid R = \mathit{false})P(R = \mathit{false})} \\
&= \frac{1}{1 + \frac{P(S_1,\ldots,S_N|R=\mathit{false})P(R=\mathit{false})}{P(S_1,\ldots,S_N|R=\mathit{true})P(R=\mathit{true})}} \\
&= \frac{1}{1 + \exp\left\{-\log\left(\frac{P(R=\mathit{true})}{P(R=\mathit{false})}\frac{1}{N-k+1}\sum_{i=1}^{N-k+1}\prod_{j=1}^{k}\frac{\psi_j[S_{j+i-1}]}{\theta_0[S_{j+i-1}]}\right)\right\}} \\
&= \mathit{logit}(x)
\end{aligned}$$

where $\mathit{logit}(x) = \frac{1}{1+\exp\{-x\}}$ is the logistic function, and

$$x = \log\left(\frac{P(R = \mathit{true})}{P(R = \mathit{false})}\frac{1}{N-k+1}\sum_{i=1}^{N-k+1}\prod_{j=1}^{k}\frac{\psi_j[S_{j+i-1}]}{\theta_0[S_{j+i-1}]}\right)$$

We now note that for the goal of predicting the probability of R given the sequence, the background probabilities are irrelevant as separate parameters. Instead, we can parameterize the above model simply using k position-specific weights $w_j[l] = \log\frac{\psi_j[l]}{\theta_0[l]}$

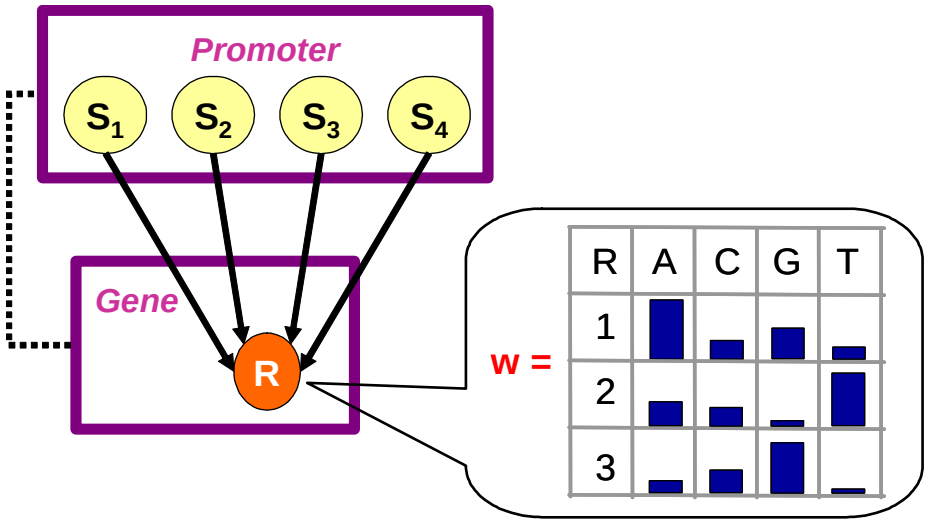


Figure 3.6: Example of the motif model for a single motif of length 3 on a promoter sequence with four nucleotides. In this example, the weights are such that the consensus sequence of the motif is ATG. That is, the largest weight in the first position is $w_1[A]$, the largest weight in the second position is $w_2[T]$, and the largest weight in the third position is $w_3[G]$.

and a threshold $w_0 = \log \frac{P(R=true)}{P(R=false)}$. Thus, we write

$$\begin{aligned} & P(R = true \mid S_1, \ldots, S_N) \\ &= logit\left(w_0 + \log\left(\frac{1}{N-k+1}\sum_{i=1}^{N-k+1}\prod_{j=1}^{k}\exp\{w_j[S_{j+i-1}]\}\right)\right) \\ &= logit\left(w_0 - \log(N-k+1) + \log\left(\sum_{i=1}^{N-k+1}\exp\{\sum_{j=1}^{k} w_j[S_{j+i-1}]\}\right)\right) \end{aligned}$$

An example of the motif model is shown in Figure 3.6. As we discuss in Section 3.3.2, we can train these parameters directly, so as to best predict the $\mathbf{R}$ variables, thereby avoiding the need for defining an explicit background distribution. If we do assume a background model of order zero, then we can recover the $\psi_i[l]$ probability,

of observing nucleotide l in position i in the motif as:

$$\psi_i[l] = \frac{\exp\{w_i[l]\}\theta_0[l]}{\sum_{l\in\{A,C,G,T\}} \exp\{w_i[l]\}\theta_0[l]}$$

and the log-odds ratio as:

$$\log \frac{\psi_i[l]}{\theta_0[l]} = w_i[l] - \log\left(\sum_{l\in\{A,C,G,T\}} \exp\{w_i[l]\}\theta_0[l]\right)$$

3.2.2 Regulation Model: Motif Profiles

The second component of our model describes how the individual motifs assemble into motif combinations, which we termed *motif profiles*. Our goal is to associate a motif profile with each module, such that the motif profile represents a combination of motifs. Genes that have this combination of motifs will then be more likely to be assigned to the module. To achieve this, we define the motif profile of a transcriptional module to be a set of weights u_{mi}, one for each motif, such that u_{mi} specifies the extent to which motif i plays a regulatory role in module m. Roughly speaking, the strength of the association of a gene g with a module m is $\sum_{i=1}^{L} u_{mi}\, g.R_i$. The stronger the association of a gene with a module, the more likely it is to be assigned to it. We use the *softmax* conditional distribution to model this association. The softmax distribution is the standard extension of the binary logistic conditional distribution to the multi-class case:

$$P(g.\mathbf{M} = \bar{m} \mid g.R_1 = r_1, \ldots, R_L = r_L) = \frac{\exp\{\sum_{i=1}^{L} u_{\bar{m}i} r_i\}}{\sum_{m'=1}^{K} \exp\{\sum_{i=1}^{L} u_{m'i} r_i\}}.$$

where each r_i variable is either *false* or *true* depending on whether motif i regulates gene g. An example of the regulation model is shown in Figure 3.7.

As we expect a motif to be active in regulating only a small set of modules in a given setting, we limit the number of weights $u_{1i}, \ldots, u_{Ki}$ that are non-zero to some $h \ll K$. This restriction results in a sparse weight matrix for $P(\mathbf{M} \mid \mathbf{R})$, and ensures that each regulator affects at most h modules. In addition, for interpretability

considerations, we require all weights to be non-negative. Intuitively, this means that a gene's assignment to specific transcriptional modules can only depend on features that correspond to the presence of certain motifs and not on the absence of motifs. For a module m, the set of motifs u_{mi} that are non-zero are called the *motif profile* of m.

3.2.3 Gene Expression Model

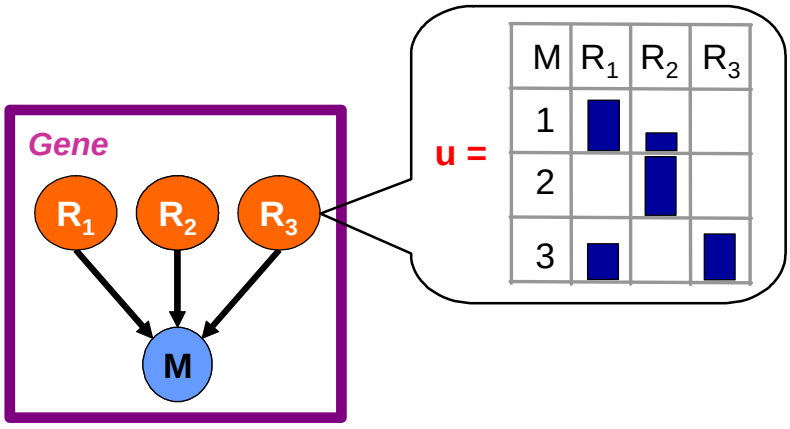
The third and last component of our model includes a model for the unique expression profile, across all arrays, that we associate with each of the K modules. For each module, we specify its expression profile with a separate distribution over the mRNA expression measurements of the genes in the module in each of the arrays.

More formally, the expression level variable *Level* in the Expression class depends on the module assignment Expression.*Gene*.$\mathbf{M}$ and on the unique identifier of the array, Expression.*Array.ID*. The CPD of *Level* is a table conditional probability distribution which specifies a different for each combination of values of $\mathbf{M}$ and *ID*. This model assumes that genes in the same module exhibit the same gene expression behavior, and is equivalent to the simple yet powerful Naive Bayes model (Duda and Hart, 1973, Cheeseman *et al.*, 1988). As the expression measurements are real-valued, we model each conditional probability distribution $P(Level \mid \mathsf{Expression}.Gene.\mathbf{M} = m, \mathsf{Expression}.Array.ID = i)$ using a Gaussian distribution $\mathcal{N}(\mu_{mi}; \sigma^2_{mi})$. The expression model is depicted in Figure 3.8.

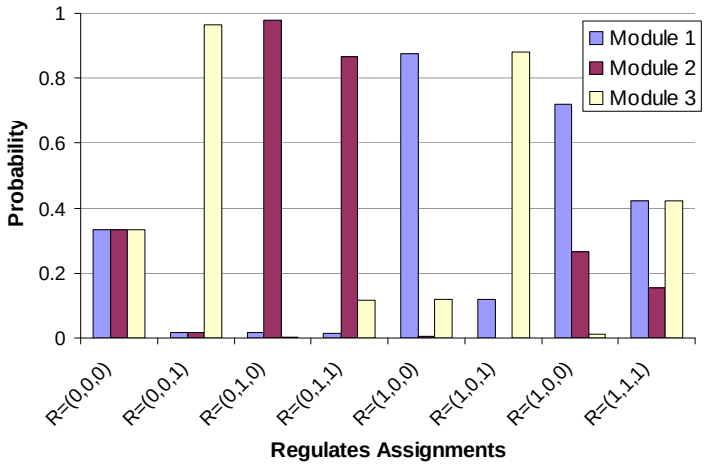
3.2.4 Joint Probabilistic Model

The three model components presented above are put together into our unified model over sequence and expression data, as shown in Figure 3.9: individual motifs are patterns in the promoter sequence; these motifs assemble into motif profiles, where each motif profile is associated with one of the K modules; and each motif profile specifies a unique expression pattern across all arrays.

Given an *instance* $\mathcal{I}$ of the PRM of Figure 3.9, the joint distribution induced by



(a)



(b)

Figure 3.7: Example of the regulation model for a model with three motifs and three modules. (a) Dependency model of the regulation model and an illustration of the weight matrix for the softmax distribution. Weights are indicated with blue bars, where the height of the bar corresponds to the actual weight. (b) The probability distribution over module assignments for each possible instantiation to the $\mathbf{R}$ variables, in the case of the softmax weights equal to: $u_{11} = 5$, $u_{12} = 2$, $u_{22} = 6$, $u_{31} = 3$, and $u_{33} = 4$. For example, if only $R_2 = 1$, then most of the probability mass is in module 2, as expected since only module 1 and 2 have a positive weight for motif 2, and module 2 places a higher weight than module 1 on motif 2.

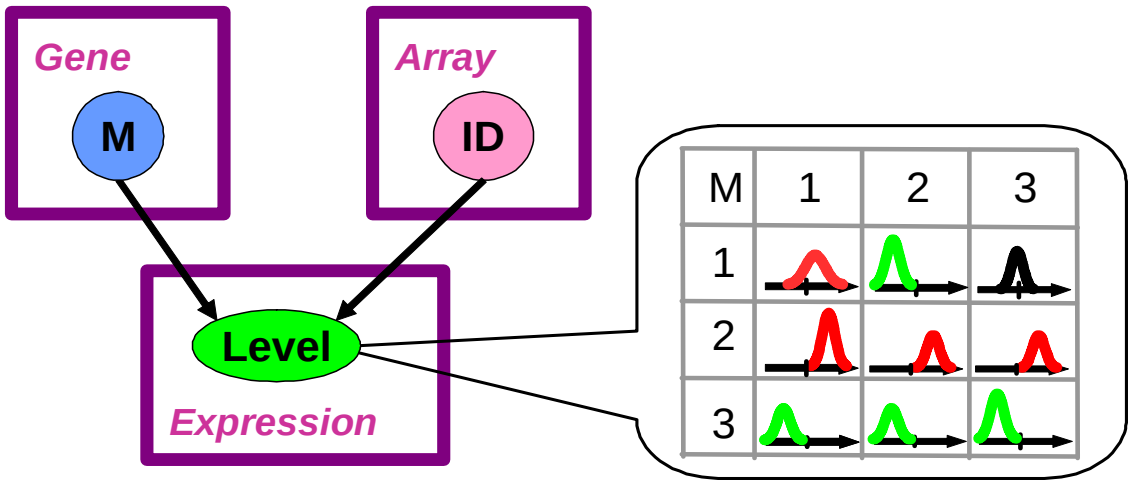


Figure 3.8: Example of the expression model for a model with three arrays and three modules. Each Gaussian distribution is depicted in the corresponding entry in the CPD of the *Level* variable. The Gaussian distribution is colored according to the mean of the distribution (red for means above zero; green for means below zero; black for means close to zero). These Gaussian distributions define an expression profile for each module across all arrays. For example, the expression profile for module 3 is down-regulated in all three arrays, while the expression profile for module 2 is up-regulated across all three arrays.

the model is given by:

$$
\begin{aligned}
P(\mathcal{I} \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}) \;=\; & \prod_{p \in \sigma[\mathsf{Promoter}]} P(\mathcal{I}[p.\mathbf{S}]) \\
& \prod_{a \in \sigma[\mathsf{Array}]} P(\mathcal{I}[a.ID]) \\
& \prod_{g \in \sigma[\mathsf{Gene}]} P(\mathcal{I}[g.\mathbf{M}] \mid \mathcal{I}[g.\mathbf{R}]) \prod_{i=1}^{L} P(\mathcal{I}[g.R_i] \mid \mathcal{I}[g.Promoter.\mathbf{S}]) \\
& \prod_{e \in \sigma[\mathsf{Expression}]} P(\mathcal{I}[e.Level] \mid \mathcal{I}[e.Gene.\mathbf{M}], \mathcal{I}[e.Array.ID])
\end{aligned}
$$

where each of the above conditional probability distributions is parameterized as described in the previous sections. As usual in the PRM framework, this distribution is equivalent to the distribution specified by the ground Bayesian network induced from the PRM. For simplicity of notation, in the remainder of this chapter we omit

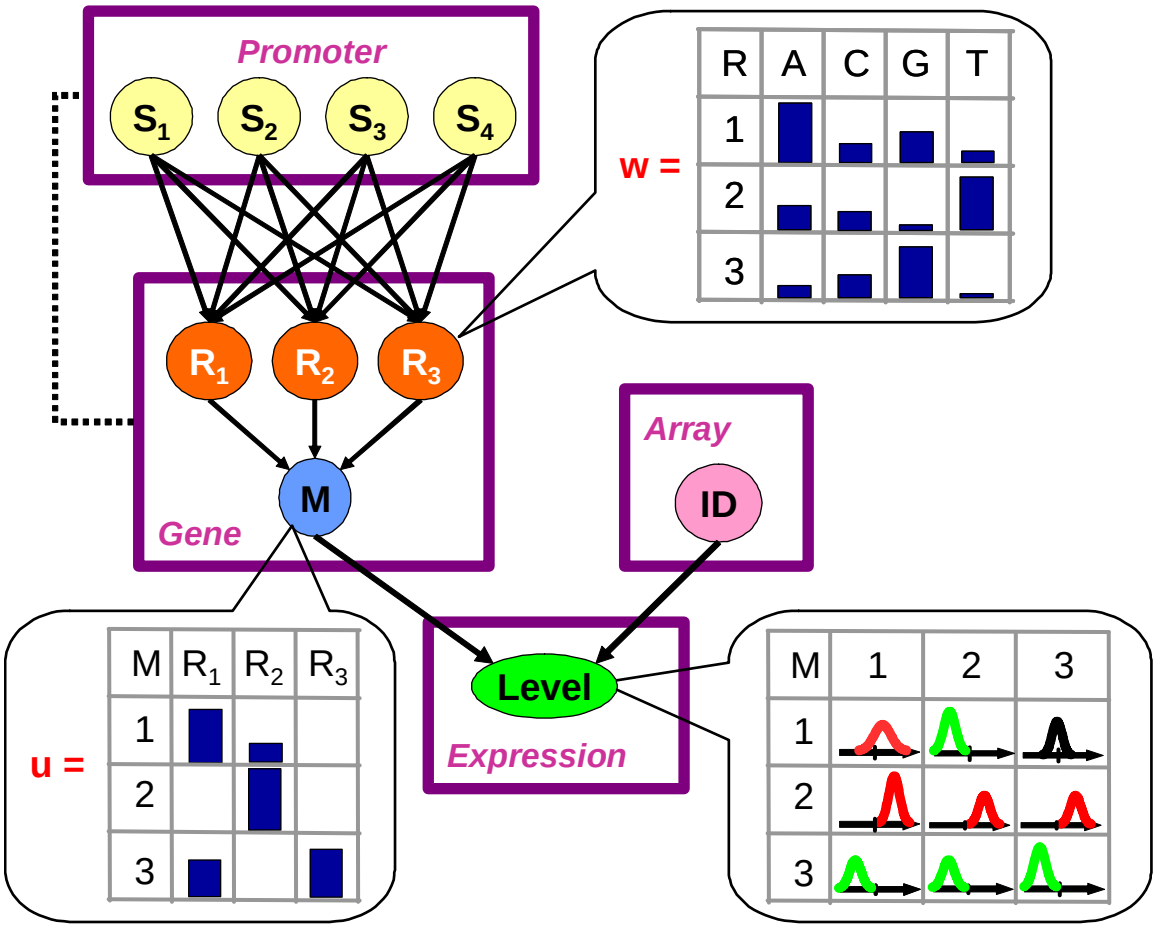


Figure 3.9: Example of the unified model for a model with three arrays, three modules, three motifs, and promoter sequences with a length of four nucleotides. The model consists of the three components described in the previous sections: a motif model; a regulation model; and an expression model.

the explicit mention of $\mathcal{I}$ for referring to values of attributes (e.g., we use $g.\mathbf{M}$ as shorthand for $\mathcal{I}[g.\mathbf{M}]$).

An example of the induced ground Bayesian network for the unified model is given in Figure 3.10. We note that as the promoter sequence data and the unique identifier of each array are always observed, the distribution over their values is irrelevant when learning the parameters of the model. Thus, we will usually be interested only in the probability $P(\mathcal{I}[\mathsf{Gene}], \mathcal{I}[\mathsf{Expression}] \mid \sigma, \mathcal{S}, \boldsymbol{\theta}_{\mathcal{S}}, \mathcal{I}[\mathsf{Array}], \mathcal{I}[\mathsf{Promoter}])$.

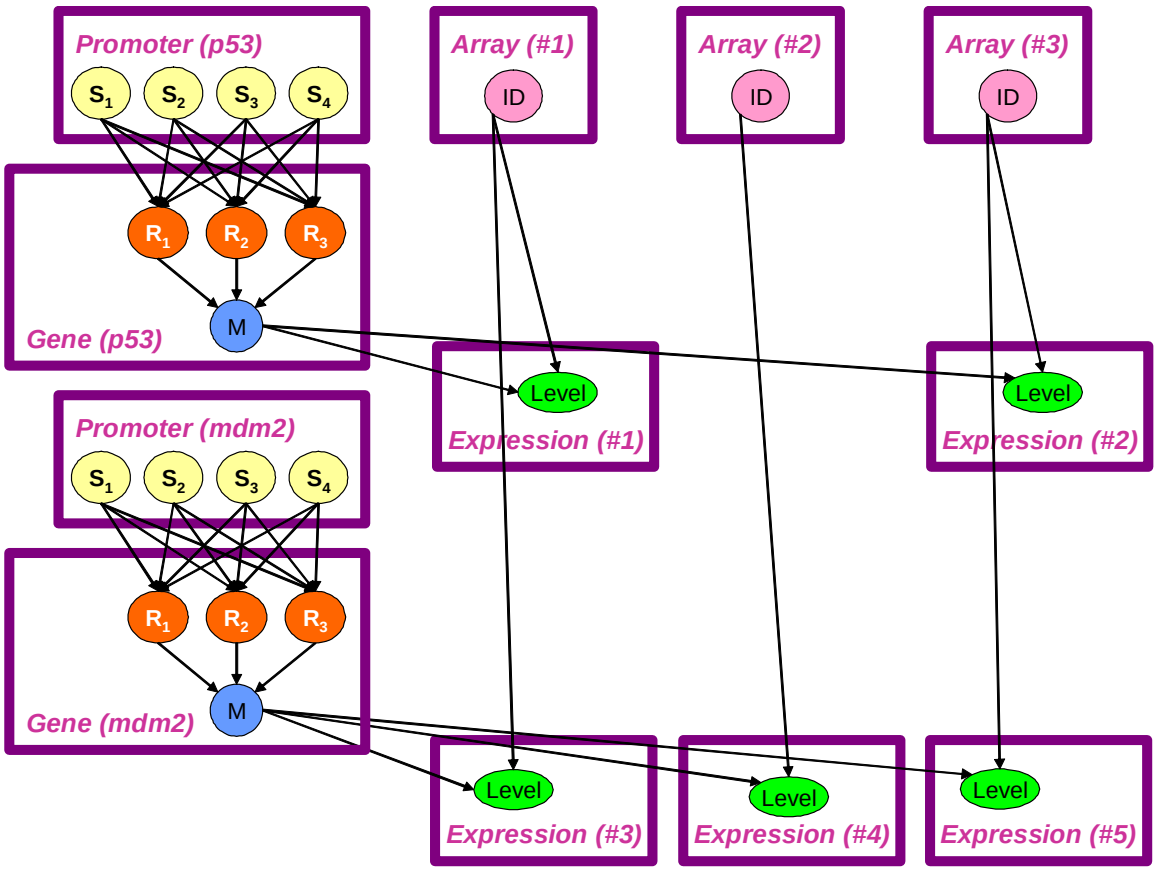


Figure 3.10: Example of the ground Bayesian network for the unified model of Figure 3.9, for a skeleton consisting of two genes, three arrays, and five corresponding expression measurements for these genes and arrays.

3.3 Learning the Model

In the previous section, we described the different components of our unified probabilistic model. We now turn to the task of learning this model from the input promoter sequences and gene expression data. A critical part of our approach is that our algorithm does not learn each part of the model in isolation. Rather, our model is trained as a unified whole, allowing information and (probabilistic) conclusions from one type of data to propagate and influence our conclusions about another type.

The input to our learning problem consists of an instance $\mathcal{I}$ of the PRM model which specifies: a set of gene objects, $\mathcal{I}[\mathsf{Gene}]$; a set of promoter sequence objects,

$\mathcal{I}$[Promoter], such that each gene object has a corresponding promoter sequence object; a set of array objects, $\mathcal{I}$[Array]; and a set of expression measurement objects, $\mathcal{I}$[Expression]. The PRM instance also specifies: for each promoter object, p, a value for each nucleotide variable S_i; for each array object, a, its unique identifier $a.ID$; and for each expression object, e, its real-valued mRNA expression level measured, $e.Level$ (expression measurements for some genes in some arrays may be missing). Note that the module variable, $g.\mathbf{M}$, and the *Regulates* variables, $g.\mathbf{R}$, of each gene object, g, are not specified by the PRM instance and are thus hidden. In this section we restrict our attention to a model with a fixed number of L motif variables per gene. In the next section we show how to dynamically add and remove motif variables from the model.

We note that the dependency structure of the model, $\mathcal{S}$, is fixed as specified in Figure 3.9. Thus, for simplicity of notation, we omit references to the dependency structure. Since the structure is fixed, our learning problem consists of estimating the model parameters so as to maximize their fit to the data. The model parameters to be estimated are: the PSSM weights for each sequence motif i, $P(\mathsf{Gene}.R_i \mid \mathsf{Gene}.Promoter.\mathbf{S})$; the softmax weights and structure of the module assignments (i.e., which sequence motifs each module depends on), $P(\mathsf{Gene}.\mathbf{M} \mid \mathsf{Gene}.\mathbf{R})$; and the means and variances of the Gaussian distributions of the expression model, $P(\mathsf{Expression}.Level \mid \mathsf{Expression}.Gene.\mathbf{M}, \mathsf{Expression}.Array.ID)$. Note that since the promoter sequence data and the unique identifier of each array are always observed, we do not estimate their parameters, $P(\mathsf{Promoter}.\mathbf{S})$, $P(\mathsf{Array}.ID)$, as they will be fixed for all models we consider (for a given PRM instance).

We follow the standard approach of *maximum likelihood* estimation: we find the parameters $\boldsymbol{\theta}$ that maximize $P(\mathcal{I} \mid \boldsymbol{\theta})$. Our learning task is made considerably more difficult by the fact that both the module assignment $g.\mathbf{M}$ and the *Regulates* variables $g.\mathbf{R}$ are unobserved in the training data. As we discussed in Section 2.6.2, in this case the likelihood function has multiple local maxima, and no general method exists for finding the global maximum. We thus use the *Expectation Maximization (EM)* algorithm (Dempster *et al.*, 1977), which provides an approach for finding a local maximum of the likelihood function.

Starting from an initial guess $\boldsymbol{\theta}^{(0)}$ for the parameters, EM iterates the following two steps. The *E-step* computes the distribution over the unobserved variables given the observed data and the current estimate of the parameters. We use the *hard assignment* version of the EM algorithm, outlined in Section 2.6.2, where this distribution is used to select a likely completion of the hidden variables. The *M-step* then re-estimates the parameters by maximizing the likelihood with respect to the completion computed in the E-step. Usually, the estimation task in the M-step is simple and can be done in closed form. However, as we show next (following a presentation of the E-step), this is not the case for our choice of CPDs, where this estimation task differs for the different parts of the model.

3.3.1 E-step: Inferring Modules and Regulation

Our task in the E-step is to compute the distribution over the unobserved data, which in our setting means computing $P(g.\mathbf{M}, g.\mathbf{R} \mid \mathcal{I}[Array], \mathcal{I}[Promoter], \boldsymbol{\theta})$. Note that, as the unique identifiers of each array are always observed, all gene objects are in fact independent (see Figure 3.10). Thus, we can perform the computation $P(g.\mathbf{M}, g.\mathbf{R} \mid \mathcal{I}[Array], \mathcal{I}[Promoter], \boldsymbol{\theta})$ separately for each gene g. Moreover, as the promoter sequence variables are observed as well, the local Bayesian network structure for each gene is effectively a tree, for which there exists efficient algorithms for performing the inference task. However, although the softmax distribution for $P(g.\mathbf{M} \mid g.\mathbf{R})$ has a compact parameterization, inference using this distribution is still exponential in the number of *Regulates* variables. Even if only a small number of these variables are associated with any single module, for the purpose of module assignment, we need to consider all of the variables associated with any module; this number can be quite large, rendering exact inference intractable.

We devise a simple approximate algorithm for doing this computation, which is particularly well suited for our setting. It exploits our expectation that, while a large number of sequence motifs determine the module assignment, only a small number of motifs regulate a particular transcriptional module. Consequently, given the module assignment for a gene, we expect a small number of *Regulates* variables for that gene

```
For each gene g ∈ 𝓘[Gene]
  Set g.M = 1
  Set g.R_i = false for 1 ≤ i ≤ L
  Set p = P(g.M, g.R, g.Expressed-In.Level | g.S, g.Expressed-In.Array.ID)
  For m = 1 to K // for all modules
    Repeat // Find g.R_i that increases p
      Set p_best = p
      For i = 1 to L // for all regulates variables
        Set g.R_i = true
        p' = P(g.M = m, g.R, g.Expressed-In.Level | g.S, g.Expressed-In.Array.ID)
        if p' > p
          Set g.M = m
          Set p = p'
        else
          Set g.R_i = false
    Until p_best = p
```

Figure 3.11: Pseudo code for the search procedure of the E-step of EM

to take the value *true*. Our approximate algorithm therefore searches greedily for a small number of *Regulates* variables to activate for each module assignment. For each gene g, it considers every possible module assignment m, and finds a good assignment to the *Regulates* variables given that $g.\mathbf{M} = m$. This assignment is constructed in a greedy way, by setting each of the $g.\mathbf{R}$ variables to *true* one at a time, as long as $P(g.\mathbf{M}, g.\mathbf{R}, g.\textit{Expressed-In}.\mathbf{Level} \mid g.\mathbf{S}, g.\textit{Expressed-In.Array}.\mathbf{ID})$, the probability of g's expression level and assignment to the *Module* and *Regulates* variables given g's sequence, improves, where $g.\textit{Expressed-In}.\mathbf{Level}$ represents all expression level variables *Level* measured for gene g, and $g.\textit{Expressed-In.Array}.\mathbf{ID}$ represents the unique identifiers of all the arrays in which the expression of g was measured. The joint setting for $g.\mathbf{M}$ and $g.\mathbf{R}$ which gives the overall best likelihood is then selected as the (approximate) most likely assignment. Pseudo-code of the algorithm are given in Figure 3.11. For the remainder of this section, let $g.\bar{m}$ and $g.\bar{r_1}, \ldots, g.\bar{r_L}$ represent the values selected for $g.\mathbf{M}$ and $g.R_1, \ldots, g.R_L$ respectively by the E-step.

3.3.2 M-step: Motif Model

We want the motif model to be a good predictor of the assignment $\bar{\mathbf{r}}$ to the *Regulates* variables computed in the E-step. Thus, for each R_i, we aim to find the values of the parameters $w_0, w_j[\ell]$ for $1 \leq j \leq k$ that maximize the conditional log probability:

$$\begin{aligned} f(\vec{w}) &= \sum_{g \in \mathcal{I}[Gene]} \log P(g.R_i = g.\bar{r}_i \mid g.S_1, \ldots, g.S_N) \\ &= \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = true} \log logit(x) + \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = false} \log(1 - logit(x)) \end{aligned} \tag{3.1}$$

where as above, $logit(x) = \frac{1}{1+\exp\{-x\}}$ is the logistic function, and

$$x = w_0 - \log(N-k+1) + \log\left(\sum_{i=1}^{N-k+1} \exp\{\sum_{j=1}^{k} w_j[S_{j+i-1}]\}\right) \tag{3.2}$$

Unfortunately, this optimization problem has no closed form solution, and there are many local maxima. We therefore use a conjugate gradient ascent to find a local optimum in the parameter space. Conjugate gradient starts from an initial guess of the weights $\vec{w}^{(0)}$, and then uses conjugate directions (instead of the local gradient as in steepest gradient ascent) for going uphill in the function space, until arriving at a local maximum. We use a standard implementation of the conjugate gradient method. This requires us to compute the value of the function we are maximizing and its derivative at various points of the parameter space $\vec{w}$. The value of the function, $f(\vec{w})$, can be computed according to Equation 3.1. We now show how to compute the derivative, $\frac{\partial f}{\partial \vec{w}}$, for w_0:

$$\begin{aligned} &\frac{\partial \sum_{g \in \mathcal{I}[Gene]} \log P(g.R_i = g.\bar{r}_i \mid g.S_1, \ldots, g.S_N)}{\partial w_0} \\ &= \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = true} \frac{logit(x)(1 - logit(x))}{logit(x)} + \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = false} \frac{-logit(x)(1 - logit(x))}{(1 - logit(x))} \\ &= \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = true} logit(-x) - \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = false} logit(x) \end{aligned}$$

where x is defined as in Equation 3.2. For all other weights $w_p[l]$, the derivative can

be computed as:

$$\frac{\partial \sum_{g\in\mathcal{I}[Gene]} \log P(g.R_i = g.\bar{r}_i \mid g.S_1, \ldots, g.S_N)}{\partial w_p[l]} =$$
$$\sum_{g\in\mathcal{I}[Gene]:g.\bar{r}_i=true} logit(-x)\frac{\sum_{i=1}^{N-k+1}\eta\{g.S_{i+p-1}=l\}\exp\{\sum_{j=1}^{k} w_j[g.S_{j+i-1}]\}}{\sum_{i=1}^{N-k+1}\exp\{\sum_{j=1}^{k} w_j[g.S_{j+i-1}]\}} -$$
$$\sum_{g\in\mathcal{I}[Gene]:g.\bar{r}_i=false} logit(x)\frac{\sum_{i=1}^{N-k+1}\eta\{g.S_{i+p-1}=l\}\exp\{\sum_{j=1}^{k} w_j[g.S_{j+i-1}]\}}{\sum_{i=1}^{N-k+1}\exp\{\sum_{j=1}^{k} w_j[g.S_{j+i-1}]\}}$$

where $\eta\{g.S_{i+p-1} = l\}$ is the indicator function which is equal to 1 if and only if the nucleotide at position $i + p - 1$ of the promoter sequence of gene g is l. This corresponds to the case when the parameter $w_p[l]$ is used when the motif binds the promoter sequence starting at position i.

Motif Model Initialization

Conjugate gradient starts from an initial guess of the weights $\vec{w}^{(0)}$. As for all local hill climbing methods, the quality of the starting point has a large impact on the quality of the local optimum found by the algorithm. In principle, we can use any motif learning algorithm to initialize the model. We use the method of Barash *et al.* (2001), which efficiently scores many motif "signatures" for significant over-abundance in the promoter sequences p_1, in which we expect to find the motif, compared to all other promoter sequences p_0. It uses the random projection approach of Buhler and Tompa (2001) to generate motif seeds of length 6–15. The over-abundance of each motif seed is then scored by the hypergeometric significance test, using the following equation:

$$P(X \geq k) = \sum_{i=k}^{n} \frac{\binom{K}{i}\binom{N-K}{n-i}}{\binom{N}{n}}$$

where k is the number of promoters from p_1 that contain the seed, n is the number of promoters in p_1, K is the number of promoters from p_0 and p_1 that contain the

seed, $N = p_0 + p_1$ is the total number of promoters, and $P(X \geq k)$ represents the probability that a set of n randomly selected promoters from all N promoters will have k or more promoters that contain the seed, given that a total of K promoters contain the seed. The use of the hypergeometric test allows efficient detection of potential initialization points that are discriminative in nature. Each seed produced by this method is then expanded to produce a PSSM of the desired length, whose weights serve as an initialization point for the conjugate gradient procedure.

3.3.3 M-step: Regulation Model

Next, we consider the task of estimating the parameters for the softmax distribution $P(g.\mathbf{M} \mid g.\mathbf{R})$. Our goal is to find a setting for the softmax weights $\{u_{mi}\}_{1 \leq m \leq K, 1 \leq i \leq L}$ so as to maximize the conditional log probability:

$$\sum_{g \in \mathcal{I}[\mathsf{Gene}]} \log P(g.\mathbf{M} = g.\bar{m} \mid g.\mathbf{R} = g.\bar{\mathbf{r}}) = \sum_{g \in \mathcal{I}[\mathsf{Gene}]} \log \frac{\exp\{\sum_{i=1}^{L} u_{\bar{m}i} g.\bar{r}_i\}}{\sum_{m'=1}^{K} \exp\{\sum_{i=1}^{L} u_{m'i} g.\bar{r}_i\}} \quad (3.3)$$

Although this optimization does not have a closed form solution, the function is convex in the weights of the softmax. Thus, a unique global maximum exists, which we can find using the conjugate gradient method where the function is computed as in Equation 3.3, and the derivative for each weight, $\frac{\partial f}{\partial u_{mi}}$, is computed as:

$$\begin{aligned}
&\frac{\partial \sum_{g \in \mathcal{I}[\mathsf{Gene}]} \log P(g.\mathbf{M} = g.\bar{m} \mid g.\mathbf{R} = g.\bar{\mathbf{r}})}{\partial u_{mi}} \\
&= \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = true} \frac{\sum_{m'=1}^{K} \exp\{\sum_{i=1}^{L} u_{m'i} g.\bar{r}_i\}}{\exp\{\sum_{i=1}^{L} u_{\bar{m}i} g.\bar{r}_i\}} \cdot \\
&\quad \frac{\eta\{g.\bar{m} = m\} \exp\{\sum_{i=1}^{L} u_{\bar{m}i} g.\bar{r}_i\} \sum_{m'=1}^{K} \exp\{\sum_{i=1}^{L} u_{m'i} g.\bar{r}_i\} - \left(\exp\{\sum_{i=1}^{L} u_{\bar{m}i} g.\bar{r}_i\}\right)^2}{\left(\sum_{m'=1}^{K} \exp\{\sum_{i=1}^{L} u_{m'i} g.\bar{r}_i\}\right)^2} \\
&= \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = true} \frac{\eta\{g.\bar{m} = m\} \sum_{m'=1}^{K} \exp\{\sum_{i=1}^{L} u_{m'i} g.\bar{r}_i\} - \exp\{\sum_{i=1}^{L} u_{\bar{m}i} g.\bar{r}_i\}}{\sum_{m'=1}^{K} \exp\{\sum_{i=1}^{L} u_{m'i} g.\bar{r}_i\}} \\
&= \sum_{g \in \mathcal{I}[Gene]: g.\bar{r}_i = true} \eta\{g.\bar{m} = m\} - \frac{\exp\{\sum_{i=1}^{L} u_{\bar{m}i} g.\bar{r}_i\}}{\sum_{m'=1}^{K} \exp\{\sum_{i=1}^{L} u_{m'i} g.\bar{r}_i\}} \qquad (3.4)
\end{aligned}$$

where $\eta\{g.\bar{m} = m\}$ is the indicator function which is equal to 1 if and only if the value assigned to gene g in the E-step, $g.\bar{m}$, is equal to the module of the parameter for which we are computing the derivative, u_{mi}.

However, while we can find the optimal setting to the weight matrix of the softmax distribution $P(g.\mathbf{M} \mid g.\mathbf{R})$, we would like to place certain constraints on this weight matrix. More specifically, as discussed in Section 3.2.2, we constrain the weight matrix to be sparse and each weight to be non-negative. These constraints lead to more desirable models from a biological standpoint, but also turn our task into a hard combinatorial optimization problem. We use a greedy selection algorithm, that tries to include non-zero weights for the most predictive motifs for each *Regulates* variable R_i. The algorithm, shown in Figure 3.12, first finds the optimal setting to the full weight matrix; as we discussed, the optimal setting can be found using gradient ascent with Equation 3.3 and Equation 3.4. For each variable R_i, it then selects the most predictive motif — the one whose weight is largest — and adds it to the motif profile U, which contains motifs that have non-zero weight. The optimal setting for the weights in U is then found by optimizing these weights, under the constraint that each weight in U is non-negative and the weights not in U must be zero. This problem is also convex, and can be solved using the same gradient methods with a reduced set of parameters. The algorithm then continues to search for additional motifs to include in the profile U: It finds the optimal setting to all weights while holding the weights in U fixed; it then selects the highest weight motifs not in U, adds them to U, and repeats. Weights are added to U until the sparseness limit is reached, or until the addition of motifs to U does not improve the overall score.

3.3.4 M-step: Expression Model

Given the assignments of genes to modules as computed in the E-step, and the unique identifiers for all arrays, our goal is to find the parameter setting for the Gaussian distribution associated with each combination of module and identifier that maximize

Set $U = \{\}$
Set *iteration* $= 0$
Let $V = \{v_{mi}\}_{1 \le m \le K, 1 \le i \le L}$
Set *MaxScore* $= \max_V$ *Score*$[V]$ // *MaxScore* = score of unconstrained fit
Set $T =$ *Threshold for closeness to MaxScore*
Repeat
  Set *iteration* $=$ *iteration* $+ 1$
  Let $U' = \{u'_{mi}\}_{1 \le m \le K, 1 \le i \le L} - U$
  Set $U' = \text{argmax}_{U' \ge 0}$*Score*$[U', U]$
    // Optimize weights not in U; weights in U are held fixed
  For $i = 1$ to L // for all regulates variables
    Let $m = \text{argmax}_m \{u'_{mi}\}_{1 \le m \le K}$
    Set $U = U \bigcup \{u'_{mi}\}$ // Add new non-zero weight
  Set $U = \text{argmax}_{U \ge 0}$*Score*$[U, \mathbf{0}]$
    // Reoptimize weights in U; other weights $= 0$
Until *iteration* $=$ *max iteration* or *Score*$[U] >=$ *MaxScore* $- T$

Figure 3.12: Learning the softmax distribution for $P(g.\mathbf{M} \mid g.\mathbf{R})$ in the M-step. *Score* is computed using Equation 3.3.

the conditional log probability:

$$\sum_{e \in \mathcal{I}[\mathsf{Expression}]} \log P(e.Level \mid e.Gene.\mathbf{M}, e.Array.ID) =$$
$$\sum_{e \in \mathcal{I}[\mathsf{Expression}]} -\frac{(e.Level - \mu[e.Gene.\mathbf{M}, e.Array.ID])^2}{2\sigma^2[e.Gene.\mathbf{M}, e.Array.ID]}$$
$$-\frac{1}{2} \log 2\pi\sigma[e.Gene.\mathbf{M}, e.Array.ID]$$

where $\mu[e.Gene.\mathbf{M}, e.Array.ID]$ and $\sigma[e.Gene.\mathbf{M}, e.Array.ID]$ represent the mean and standard deviation, respectively, of the Gaussian distribution for module $\mathbf{M}$ and array identifier *ID*. Fortunately, the maximum likelihood setting for the parameters of the expression model Gaussian distributions has a closed form solution. It can easily be shown that the maximum likelihood parameter setting for the mean and variance of

the Gaussian distribution associated with module m and array identifier i are:

$$\mu_{mi} = \frac{\sum_{e \in \mathcal{I}[\mathsf{Expression}]} \eta\{e.Gene.\mathbf{M} = m\}\eta\{e.Array.ID = i\}e.Level}{\sum_{e \in \mathcal{I}[\mathsf{Expression}]} \eta\{e.Gene.\mathbf{M} = m\}\eta\{e.Array.ID = i\}1}$$

$$\sigma^2_{mi} = \frac{\sum_{e \in \mathcal{I}[\mathsf{Expression}]} \eta\{e.Gene.\mathbf{M} = m\}\eta\{e.Array.ID = i\}e.Level^2}{\sum_{e \in \mathcal{I}[\mathsf{Expression}]} \eta\{e.Gene.\mathbf{M} = m\}\eta\{e.Array.ID = i\}1} - \mu^2_{mi}$$

3.3.5 Dynamically Adding and Removing Motifs

In the previous section, we showed how to optimize the model parameters given a fixed set of motifs. We now wish to devise a dynamic learning algorithm, capable of both removing and adding sequence motifs as part of the learning process. As we learn the models, some motifs may not turn out to be predictive, or redundant given the newly discovered motifs. Conversely, some modules may not be well explained by sequence motifs. This means that the model is not fully explaining the expression data as a function of motif profiles. By dynamically removing and adding motifs as necessary during the model learning phase, we wish to improve this situation and arrive as close as possible to a genome-wide explanation of the expression data.

We add and remove motifs after each completion of the EM algorithm. (Note that EM itself iterates several times between the E-step and the M-step.) To determine whether R_i should be deleted, we compute the conditional log probability $\sum_{g \in \mathcal{I}[\mathsf{Gene}]} \log P(g.\mathbf{M} \mid g.\mathbf{R})$ both with and without $g.R_i$, leaving the values of the other *Regulates* variables fixed. This computation tells us the contribution that R_i makes towards the overall fit of the model. Variables that contribute below a certain threshold are subsequently removed from the model.

We try to add motifs when the current set of motifs does not provide a satisfactory explanation of the expression data: when there are genes for which the sequence predictions do not match the expression profile. More precisely, we define the *residual* for a transcriptional module m to be the set of genes that are assigned to module m in the E-step, but would not be assigned to m based on the sequence alone. An example of these residuals for two modules is shown in Figure 3.13. We determine

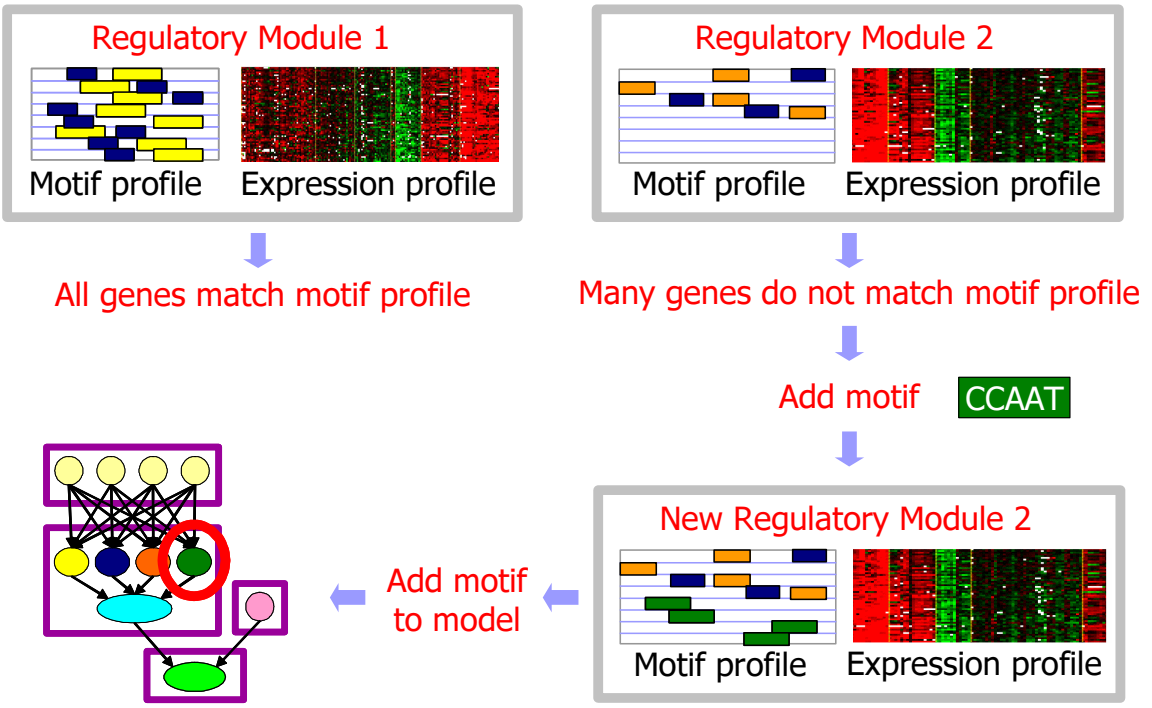


Figure 3.13: Scheme of our algorithm for identifying modules for which adding a motif variable can potentially increase their score. After each EM iteration, we examine all modules and compute their residual: the set of genes assigned to the module by the E-step but not based on the sequence alone. Modules with many residual genes are candidates for adding motifs, initialized to find a motif in the residual genes. Once a motif variable has been added, it is added to the model so that it can get refined in subsequent iterations of the learning algorithm.

the sequence-only assignment of each gene by computing

$$g.\mathbf{r} = \text{argmax}_{\mathbf{r}'} P(g.\mathbf{R} = g.\mathbf{r}' \mid g.Promoter.\mathbf{S})$$

and the resulting module assignment will then be:

$$g.m' = \text{argmax}_m P(g.\mathbf{M} = m \mid g.\mathbf{R} = g.\mathbf{r}).$$

We then attempt to provide a better prediction for the residual genes by adding a sequence motif that is trained to match these genes. To this end, we use our algorithm for learning motifs from Section 3.3.2 to search for a motif that is over-abundant in

Dynamically delete motif variables:
For $i = 1$ to L // for all regulates variables
  Set $U' = U$
  Set $u'_{mi} = 0$ for $1 \leq m \leq K$
  If $Score[U] - Score[U'] \leq threshold$
    Delete R_i
    Set $U = U'$
Dynamically add motif variables:
For $m = 1$ to K // for all modules
  Let $\mathbf{G}' = \{\}$
  For each $g \in \mathcal{I}[\mathsf{Gene}]$ such that $g.\bar{m} = m$
    Set $g.\mathbf{r} = \text{argmax}_{\mathbf{r}'} P(g.\mathbf{R} = g.\mathbf{r}' \mid g.\mathbf{S})$
    Set $g.m' = \text{argmax}_m P(g.\mathbf{M} = m \mid g.\mathbf{R} = g.\mathbf{r})$
    If $m' \neq m$
      Set $\mathbf{G}' = \mathbf{G}' \bigcup \{g\}$
  Learn motif with positive set $\mathbf{G}'$
  Add new *Regulates* variable with learned PSSM

Figure 3.14: Procedure for dynamically adding and deleting regulates variables.

the promoters of the residual genes of module m compared to the promoters of all other genes that are not assigned to module m (the non-residual genes from module m are not used in training the motif). This newly learned motif is then used to initialize a new *Regulates* variable that we add to the model.

Once a new *Regulates* variable is added, it becomes part of the model and its assignment and parameterization are adapted as part of the next EM iteration, as described in the previous section. This process tests whether a new motif contributes to the overall model fit, and may assign it a non-zero weight. Importantly, a motif that was trained for the residuals of one module often gets non-zero weights for other modules as well, allowing the same motif to participate in multiple modules. Pseudo code of the algorithm is given in Figure 3.14.

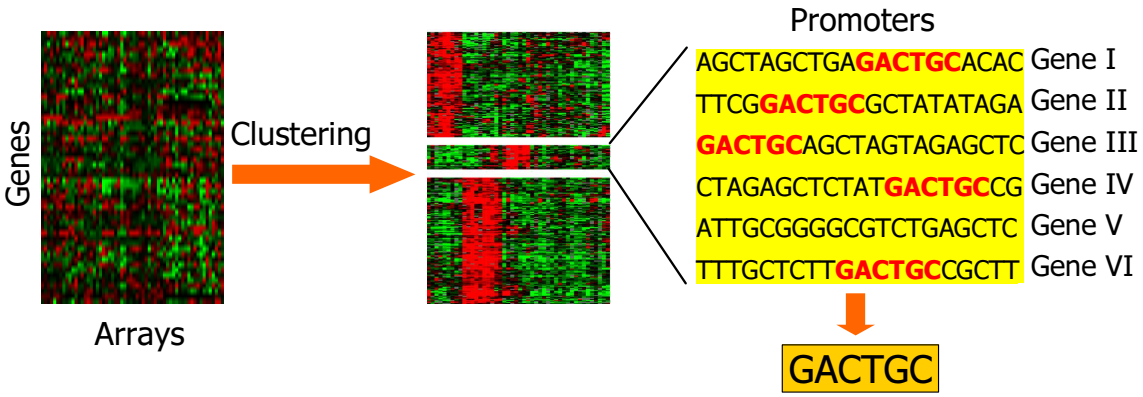


Figure 3.15: General scheme of common clustering-based approaches for finding motifs from gene expression and sequence data. In this approach, the input expression data is first clustered into clusters of genes with coherent expression patterns. A motif finding algorithm is then applied to the promoters of genes in the same cluster, with the goal of finding a common motif in such genes.

3.4 Related Work

Many researchers have studied the problem of identifying *cis*-regulatory motifs and many automated procedures have been proposed for this purpose. Broadly, these can be classified as being of one of two types. Approaches of the first and more common type use gene expression measurements to define groups of genes that are potentially co-regulated. They then attempt to identify regulatory elements by searching for commonality (e.g., a commonly occurring motif) in the promoter regions of the genes in the group (see for example Brazma *et al.* (1998), Liu *et al.* (2001), Roth *et al.* (1998), Sinha and Tompa (2000), Tavazoie *et al.* (1999)). An illustration of this approach is shown in Figure 3.15. Approaches of the second type work in the opposite direction. They first reduce the sequence data into some predefined features of the gene, e.g., the presence or absence of various potential transcription factor binding sites. These features are derived using either an exhaustive approach, say, all DNA-words of length 6-7, or a knowledge-based approach, say, all TRANSFAC sites (Wingender *et al.*, 2001). These approaches then try and exploit this feature set as well as the expression data in a combined way. Some build models that characterize the expression profiles

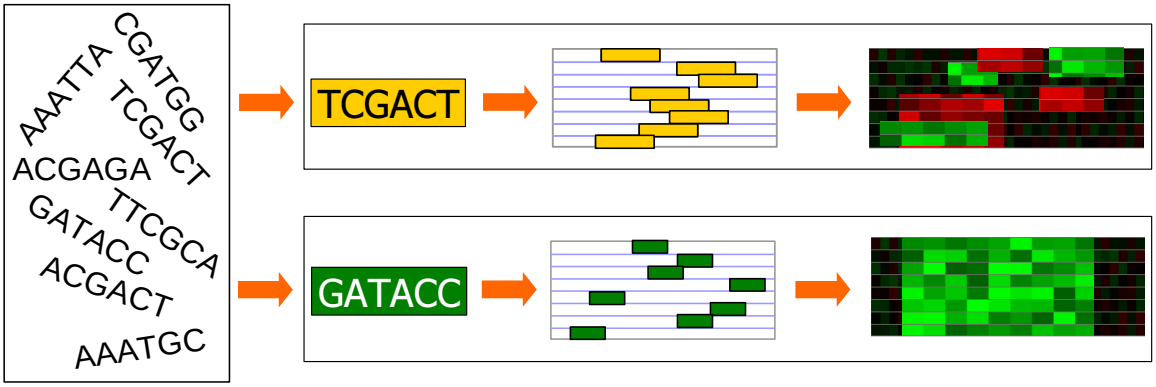


Figure 3.16: General scheme of common sequence-based approaches for finding motifs from gene expression and sequence data. In this approach, all sequences of length k are traversed. For each such sequence, all the genes containing it are first identified. For each such group of genes, the expression data is then used to evaluate whether the group of genes have coherent expression patterns. The green sequence in this example is a good motif candidate.

of groups or clusters of genes (e.g., Barash and Friedman (2001), Segal *et al.* (2001)). An illustration of this approach is shown in Figure 3.16.

A key advantage of our algorithm over these approaches lies in our ability to relate upstream sequences and expression data in a single unified probabilistic framework, which allows us to refine both the cluster assignments and motifs within the same algorithm. However, there have also been other approaches that tried to provide a unified probabilistic framework for both sequence and expression data. Holmes and Bruno (2000) describe a simple *Naive Bayes* model for promoter sequence and expression, where the genes are partitioned into disjoint clusters using a hidden *Cluster* variable, whose value probabilistically determines both the sequence (and presence of motifs) of the promoter region and the expression profile. However, this model is much simpler than ours, and fails to capture important aspects such as combinatorial effects between transcription factors, or the effect of array properties on the expression data.

Also related is the work of Pilpel *et al.* (2001), which uses standard techniques to discover motifs in the genome, and then tries to discover significant combinatorial interactions between pairs of these putative transcription factors by comparing the

expression of genes containing one, the other, or both. This approach is based on some of the same insights as ours, but it does not build a unified framework, and hence cannot use results from one part of the analysis to improve the models of the other; in particular, it does not use expression data to refine the transcription factor motifs and perhaps get better discrimination.

The work of Bussemaker *et al.* (2001) also related expression and sequence by modeling the expression of each gene as a weighted linear combination of the number of occurrences of individual k-mers. However, their motifs are fixed DNA-words and they do not allow variation in their motifs. In their approach, variation in the motifs can exist by selecting variants of the same word as the motifs. In contrast, we search for the common probabilistic representation of motifs using PSSM. More importantly, in their model, every occurrence of a motif contributes to the expression and thus the active targets are not identified, as is achieved in our model through the use of the *regulates* variables, $\mathbf{R}$.

Finally, we note that subsequent to our work (Segal *et al.*, 2002, 2003e), Beer and Tavazoie (2004) devised a method, based on Bayesian networks, whose goal is also to explain the expression data as a function of features in the sequence. In addition to using motif appearances as features, their approach also constructs features from positional information, such as the orientation of the motif, and the proximity and order between pairs of motifs. They use a clustering approach to construct modules of genes with similar expression profiles, and then learn a Bayesian network that explains the expression data of genes in the same module as a probabilistic function of the set of features above. However, in contrast to our approach, they do not construct a unified model over the expression and sequence features, and thus their model does not allow genes to change module assignment and for information to flow across different modules during the learning process.

With the recent technological advances and reductions in the cost of sequencing, new approaches have been proposed for the motif discovery task, based on the incorporation of other types of genomic data. For example, Lee *et al.* (2002) designed a genome-wide assay for experimentally measuring the target promoters that are bound by every transcription factor in yeast. They then searched for common motifs in the

promoters of genes that are bound by the same transcription factor, resulting in a large collection of both novel and previously known motifs.

Another parallel approach which is gaining in popularity uses sequences from multiple related organisms. The underlying idea is that motifs, by being functionally important elements, are more likely to be selected for during evolution. Thus, by searching for short DNA sequences that have been conserved in promoters of genes that evolved from a common ancestral gene, we might be able to identify functional motif elements. This was the approach of Cliften *et al.* (2004) and Kellis *et al.* (2004), who sequenced several species of yeast and then searched for motifs based on the idea above, resulting in a large compendium of conserved motifs.

3.5 Biological Evaluation

We evaluated our method separately on two different yeast (*S. Cerevisiae*) gene expression datasets, one consisting of 173 microarrays, measuring the responses to various stress conditions (Gasch *et al.*, 2000), and another consisting of 77 microarrays, measuring expression during the cell cycle (Spellman *et al.*, 1998). We also obtained the 500bp upstream region of each gene (sequences were retrieved from SGD (Cherry *et al.*, 1998)).

The EM algorithm requires an initial setting to all parameters. We use the standard procedure for learning motifs from expression data to initialize the model parameters: we first cluster the expression profiles, resulting in a partition of genes to clusters, and then learn a motif for each of the resulting clusters. For clustering the expression, we use the probabilistic hierarchical clustering algorithm developed by Friedman (first applied in Segal *et al.* (2001)). For learning motifs, we use the discriminative motif finder described in Section 3.2.1. To specify the initial parameterization of our model, we treat these clusters and motifs as if they were the result of an E-step, assigning a value to all of the variables $g.\mathbf{M}$ and $g.\mathbf{R}$, and learn the model parameters as described in Section 3.3.

For the stress data, we use 1010 genes which showed a significant change in expression, excluding members of the generic stress response cluster (Gasch *et al.*, 2000).

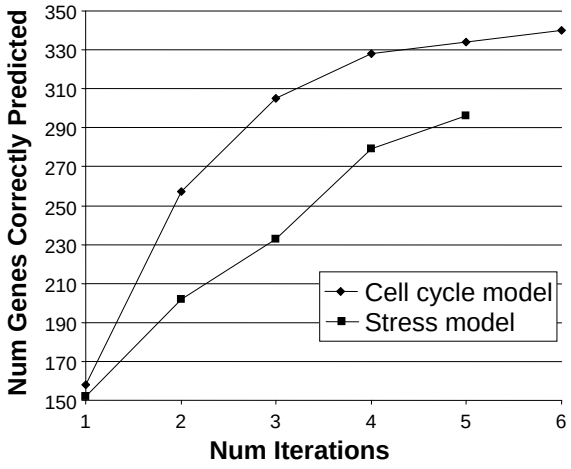


Figure 3.17: Predicting expression from sequence: Number of genes whose module assignment can be correctly predicted based on sequence alone, where a correct prediction is one that matches the module assignment when the expression is included. Predictions are shown for each iteration of the learning procedure.

We initialized 20 modules using standard clustering, and learned the associated 20 sequence motifs. ¿From this starting point, the algorithm converged after 5 iterations, each consisting of an EM phase followed by a motif addition/deletion phase, resulting in a total of 49 motifs. For the cell cycle data, we learned a model with 15 clusters over the 795 cell cycle genes defined in Spellman *et al.* (1998). The algorithm converged after 6 iterations, ending with 27 motifs.

We generally compared our results to the standard approach reviewed in Section 3.4, which first clusters the expression data into clusters of genes with similar patterns of expression, and then searches for motifs that are common in the promoter sequences of genes in the same cluster (see for example Brazma *et al.* (1998), Liu *et al.* (2001), Roth *et al.* (1998), Sinha and Tompa (2000), Tavazoie *et al.* (1999)).

3.5.1 Predicting Expression From Sequence

Our approach aims to explain expression data as a function of sequence motifs. Hence, one metric for evaluating a model is its ability to associate genes with modules based on their promoter sequence alone. Specifically, we compare the module assignment

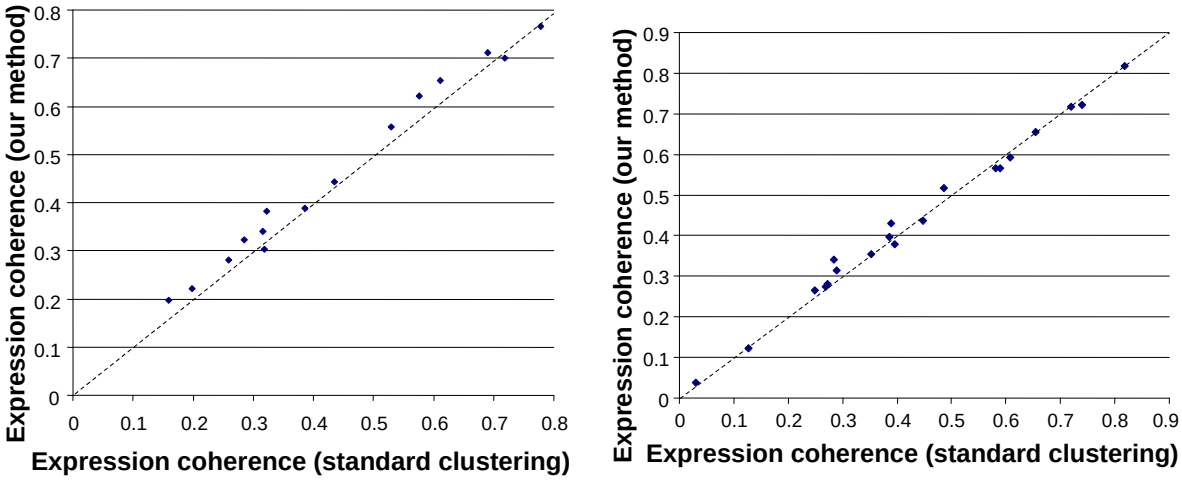
of each gene when we consider only the sequence data to its module assignment considering both expression and sequence data. The assignment of gene g based on sequence alone can be computed as:

$$\begin{aligned} g.\mathbf{r} &= \text{argmax}_{\mathbf{r}'} P(g.\mathbf{R} = g.\mathbf{r}' \mid g.\mathbf{S}) \\ g.m' &= \text{argmax}'_m P(g.\mathbf{M} = m' \mid g.\mathbf{R} = g.\mathbf{r}) \end{aligned}$$

which we can then compare to the assignment $g.\bar{m}$, which was computed in the E-step using both the sequence and the expression data (see Section 3.3.1).

Figure 3.17 shows the total number of genes whose expression-based module assignment is correctly predicted using only the sequence (i.e., genes for which $g.m' = g.\bar{m}$), as the algorithm progresses through the learning iterations and sequence motifs are added. As can be seen, the predictions improve across the learning iterations, converging to 340 (out of 795; 42.7%) and 296 (out of 1010; 29.3%) genes correctly predicted in the cell cycle and stress models, respectively. Moreover, recall that we initialize our models by first clustering the expression data and then learning a single motif for each cluster. Thus, the first iteration in our learning algorithm is precisely equivalent to one of the standard common approaches to finding cis-regulatory motifs. As can be seen in Figure 3.17, the standard approach correctly predicted only 158 (out of 795; 19.8%) and 152 (out of 1010; 15%) genes in the cell cycle and stress models, respectively, significantly lower than the number of corresponding correct predictions made by our method.

While these results are encouraging, they were measured on the genes in our training set and thus we cannot interpret them as a true measure of our ability to predict expression patterns from sequence data. To obtain such a measure, we extracted 1345 additional genes that showed a significant change in expression in the stress data (together with the 1010 genes on which we trained, these 2355 genes are the same ones to which we applied the module network procedure in Section 4.6). For these genes, we performed the same test as described above. That is, for each gene, we computed its predicted module assignment using only its sequence data and compared that to its module assignment as inferred by applying an E-step computation to this



(a) (b)

Figure 3.18: Comparison of the expression coherence for each inferred module (or cluster in the standard clustering model). (a) For the cell cycle dataset (Spellman *et al.*, 1998). (b) For the stress expression dataset (Gasch *et al.*, 2000).

held out gene using the learned model. Of these 1345 held out genes, 309 (23%) were correctly predicted compared to only 172 (12.8%) correct predictions using the standard approach (the first iteration of our model). These results show that, with an accuracy of 23%, our model can correctly predict the expression pattern of a gene by selecting from among 20 expression patterns. Moreover, this accuracy is significantly better than that achieved by a standard approach for finding motifs.

3.5.2 Gene Expression Coherence

These results indicate that our model assigns genes to modules such that genes assigned to the same module are generally enriched for the same set of motifs. However, we can achieve such an organization by simply assigning genes to modules based only on their sequence, while entirely ignoring the expression data. To verify the quality of our modules relative to gene expression data, we define the *expression coherence* of a module m to be the average Pearson correlation between the expression vectors of each pair of genes g_i and g_j assigned to it, g_i.*Expressed-In*.**Level** and g_j.*Expressed-In*.**Level**. The Pearson correlation between two vectors $\vec{v_1}$ and $\vec{v_2}$, each

of length k, is computed as:

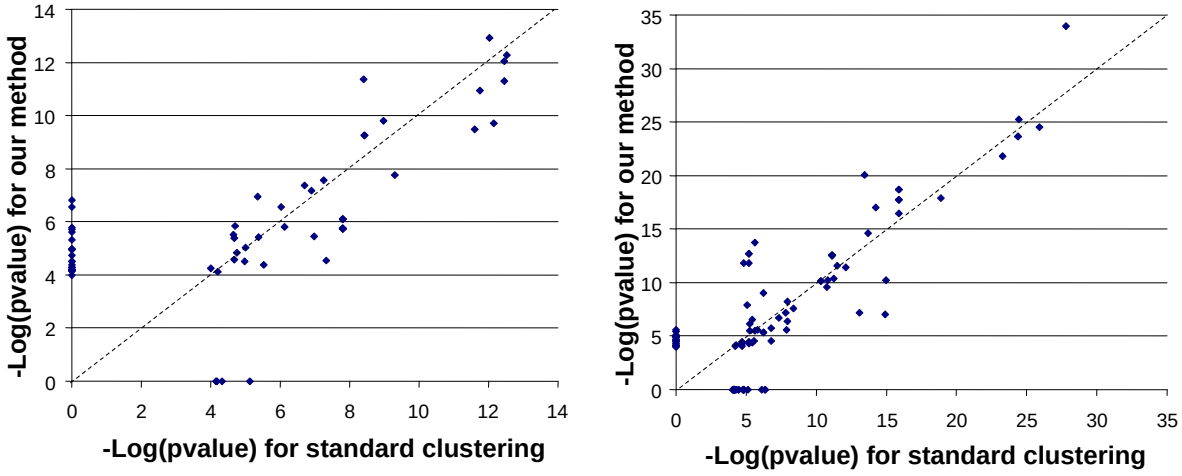
$$Pearson(\vec{v_1}, \vec{v_2}) = \frac{1}{k} \sum_{i=1}^{k} \frac{(\vec{v_1}[i] - \mu_{\vec{v_1}})}{\sigma_{\vec{v_1}}} \frac{(\vec{v_2}[i] - \mu_{\vec{v_2}})}{\sigma_{\vec{v_2}}}$$

where $\mu_{\vec{v}}$ and $\sigma_{\vec{v}}$ are the mean and standard deviation of the entries in $\vec{v}$, respectively. Figure 3.18 compares the expression coherence of our modules to those built from standard clustering (the resulting clusters after the first iteration of our learning algorithm) for the cell cycle and stress data, showing identical coherence of expression profiles. For the cell cycle data, there was even a slight increase in the coherence of the expression profiles for our model. Thus, our model results in clusters that are more enriched for motifs, while achieving the same quality of expression patterns as a standard clustering which only tries to optimize the expression score. This result indicates that the integration of expression and sequence data into the same unified framework allows us to achieve an organization of genes to modules comparable in terms of expression coherence to that of standard clustering, but one that corresponds much better to motifs found in the promoter sequences.

3.5.3 Coherence of Motif Targets

As we discussed, the motif profile characterizing a module allows us to define a notion of *motif targets* — genes that contain the motif, and where the motif plays a role in its expression profile, i.e., those genes assigned to a module whose motif profile contains the motif. In the standard clustering model, we can define the targets of a motif to be those genes that have the motif and belong to the cluster from which the motif was learned.

We tested whether our motif targets correspond to functional groups, by measuring their enrichment for genes in the same functional category according to the gene annotation database of Gene Ontology (GO, (Ashburner *et al.*, 2000)). We used only GO categories with 5 or more genes associated with them, resulting in 537 categories. For each annotation and each motif, we computed the fraction of genes in the targets



(a) (b)

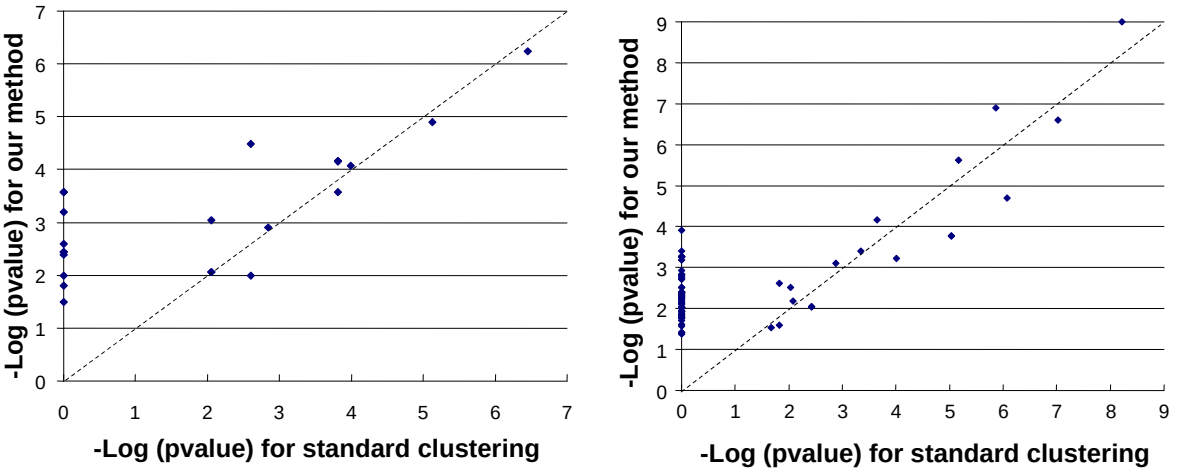
Figure 3.19: Comparison of enrichment of the targets of each motif for functional annotations from the GO database. For each annotation, the largest negative log p-value obtained from analyzing the targets of all motifs is shown. (a) is for the cell cycle dataset (Spellman *et al.*, 1998) and (b) is for the stress expression dataset (Gasch *et al.*, 2000).

of that motif associated with that annotation and used the hypergeometric distribution to calculate a p-value for this fraction. For a set of motif targets with n genes, of which k are annotated with a certain GO annotation that exists in K of the N genes in the GO database, the hypergeometric p-value is given by:

$$P(X \geq k) = \sum_{i=k}^{n} \frac{\binom{K}{i}\binom{N-K}{n-i}}{\binom{N}{n}}$$

where $P(X \geq k)$ represents the probability that the motif targets has k or more genes with the annotation. We performed a Bonferroni correction for multiple independent hypotheses and took p-values $< 0.05/537 = 9.3 \cdot 10^{-9}$ to be significant.

We compared, for both expression data sets, the enrichment of the motif targets for GO annotations between our model and standard clustering. We found many annotations that were enriched in both models. However, there were 24 and 29



(a) (b)

Figure 3.20: Comparison of enrichment of the targets of each motif for protein complexes. For each protein complex, shown is the largest negative log p-value obtained from any of the motifs. (a) is for the cell cycle dataset (Spellman *et al.*, 1998) and (b) is for the stress expression dataset (Gasch *et al.*, 2000).

annotations that were significantly enriched in our cell cycle and stress models, respectively, that were not enriched at all in the standard clustering model, compared to only 4 and 14 annotations only enriched in the standard clustering model for these respective models. Among those annotations enriched only in our model were carbohydrate catabolism, cell wall organization and galactose metabolism, all of which are processes known to be active in response to various stress conditions that we can now characterize by sequence motifs. A full comparison of the GO enrichment for both datasets is shown in Figure 3.19.

Since functional categories do not necessarily correspond to co-regulation groups, we also tested the enrichment of our motif targets for protein complexes, as compiled experimentally in the assays of Gavin and *et al.* (2002) and Ho and *et al.* (2002), consisting of 590 and 493 complexes, respectively. The member genes of protein complexes are often co-regulated and we thus expect to find enrichment for them in our motif targets. We associated each gene with the complexes in which it is part of according to the assays of Gavin and *et al.* (2002) and Ho and *et al.* (2002), and computed the p-value of the enrichment of the targets of each motif for each complex

using the hypergeometric distribution, as we did above for the GO annotations. The results for the cell cycle and stress datasets are summarized in Figure 3.20, showing much greater enrichment of our motif targets compared to the targets of the motifs identified using the standard approach, with 63 and 10 complexes significantly enriched only in our model, and no complexes only enriched in the standard approach, for the stress and cell cycle models, respectively.

3.5.4 Motifs and Motif Profiles

Next, we compared the motifs we identified to motifs from the literature, published in the TRANSFAC database (Wingender *et al.*, 2001). Of the 49 motifs learned for the stress model, 22 are known, compared to only 10 known motifs learned using the standard approach. For the cell cycle model, 15 of the 27 learned motifs are known, compared to only 8 known motifs learned using the standard approach. Many of the known motifs identified, such as the stress element STRE, the heat shock motif HSF and the cell cycle motif MCM1, are also known to be active in the respective datasets. Interestingly, 30 of the 49 motifs were among those motifs dynamically added as part of our learning procedure.

To examine some global characteristics of the motifs we learned, we also examined the distribution of the number of targets per motif, shown in Figure 3.21. When comparing this distribution to the distribution obtained if we randomized the association of each motif with its targets, we find that the true motif target distribution is highly non-random, with two *classes* of motifs emerging: one class represents general motifs, i.e., motifs that have a much larger number of target genes than would be expected by chance); the other class represents highly specific motifs, i.e., motifs with much fewer targets than would be expected by chance. We note that these two classes of motifs are consistent with the results reported by Lee *et al.* (2002) using independent data sources.

A powerful feature of our approach is its ability to characterize modules by motif profiles. This ability is particularly important for higher eukaryotes, in which regulation often occurs through multiple distinct motifs. To illustrate the motif profiles

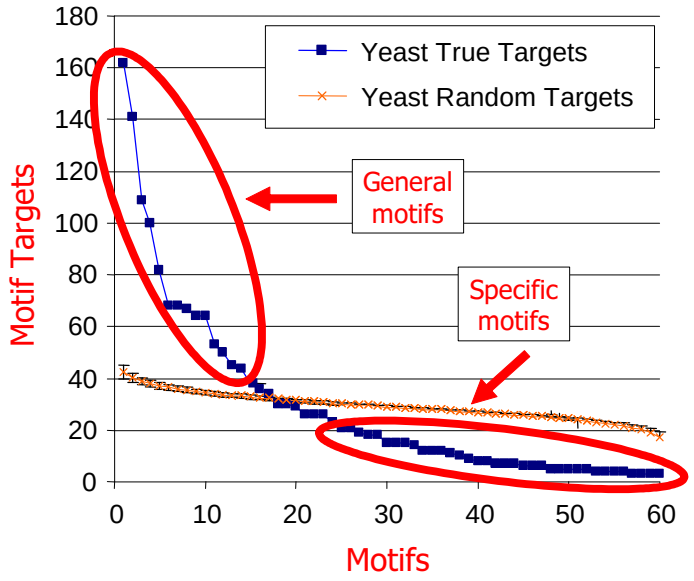


Figure 3.21: Distribution of the number of targets per motif, sorted by the number of targets (blue points). Also shown is the motif target distribution obtained for a similar number of total motif-gene pairs, but where we randomized the true motif-gene associations. The two classes of general and specific motifs are indicated by red arrows and red ovals.

found by our approach, we found for each motif all modules enriched for the presence of that motif. This was done by associating each gene with the motifs in its upstream region, and then computing the p-value of the enrichment of the member genes of each module (using the hypergeometric distribution as described above for the case of enrichment for GO annotations). Figure 3.22 shows all the module-motif pairs in which the module was enriched for the motif with p-value < 0.05. In addition, the figure indicates (by red circles) all pairs in which the motif appears in the module's motif profile. As can be seen, many profiles contain multiple motifs, and many motifs were used by more than one module. Even though modules share motifs, each module is characterized by a unique combination of motifs. Note that there are several enriched module-motif pairs where the motif was not chosen as part of the motif profile of the module (indicated by those colored entries in Figure 3.22 which are not circled). We return to this issue in Section 3.6.

Motifs that appear in the same motif profile are predicted by our method to

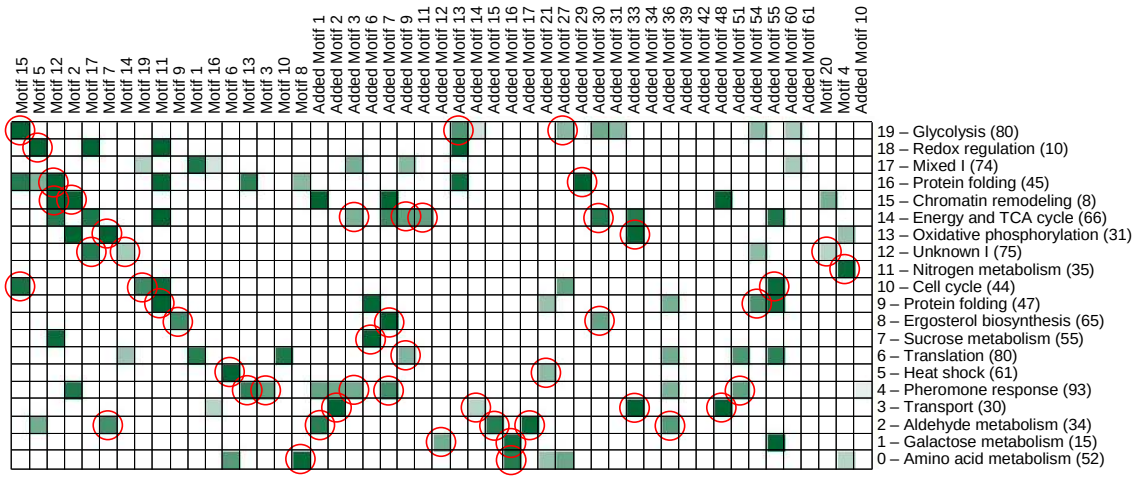


Figure 3.22: Matrix of motifs vs. modules for the stress data, where a module-motif entry is colored if the member genes of that module were enriched for that motif with p-value < 0.05. The intensity corresponds to the fraction of genes in the module that had the motif. Entries in the module's motif profile are circled in red. Modules were assigned names based on a summary of their gene content. Motifs that were dynamically added as part of the learning algorithm are named "Added Motif X".

participate together in the regulation of genes in the module for this shared motif profile. Thus, each co-appearance of two motifs is a prediction of a combinatorial interaction. Since some motifs appear in multiple motif profiles, we obtained a global view of all such combinatorial interactions, by creating a *motif interaction network*, where nodes correspond to motifs and two motifs are connected if they appear in the same motif profile. Interestingly, as shown in Figure 3.23, the resulting network is highly non-random: some motifs are connected to a large number of other motifs, while others are connected to very few. Moreover, these *hub motifs*, that are connected to many other motifs and thus are predicted by our method to participate in multiple processes, are in fact motifs that are known in the literature as general motifs whose binding transcription factor regulates a large number of genes. These hub motifs included, for example, the general stress related motifs STRE, YAP1, XBP1, and ADR1, as well as the general cell cycle motif MCM1 (these motifs are indicated by a blue arrow in Figure 3.23. We note that with the exception of STRE, the hub motifs are not part of the class of general motifs with many targets shown in Figure 3.21.

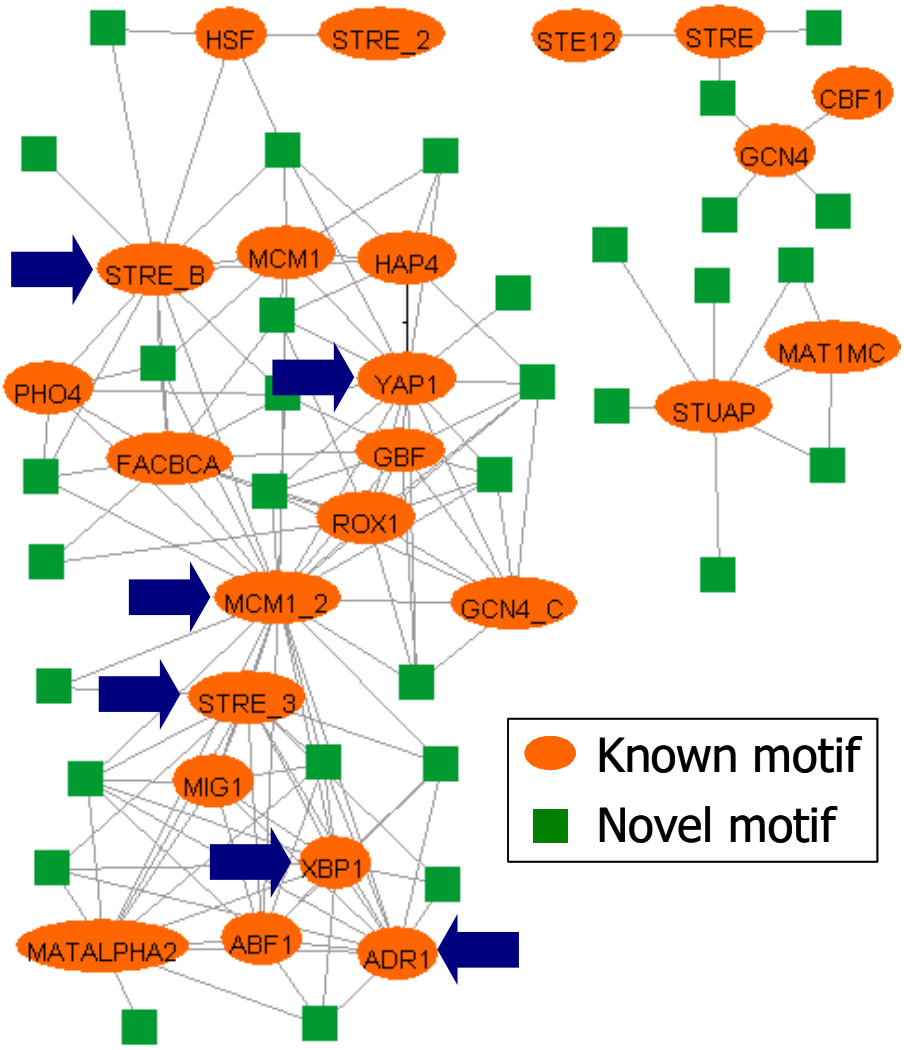


Figure 3.23: Shown is a motif interaction network, in which we connected every pair of motifs that appeared together in at least one motif profile. Motifs that are known in the literature are colored orange. The hub motifs, discussed in the text, are indicated by blue arrows.

3.5.5 Inferring Regulatory Networks

Identifying the active motifs and motif profiles is a significant step towards understanding the regulatory mechanisms governing gene expression. However, we would also want to know the identity of the transcription factor (TF) molecules that bind

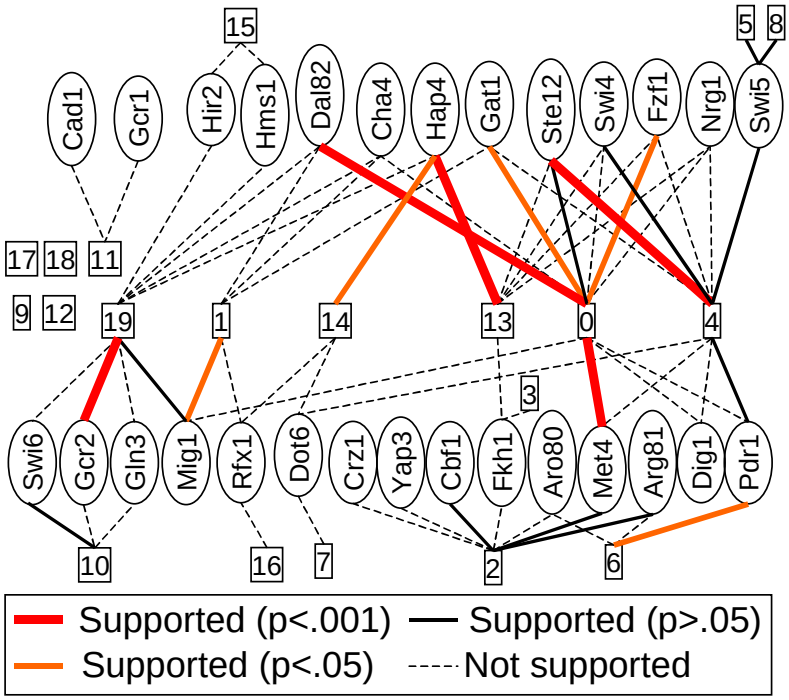


Figure 3.24: Regulatory network inferred from our model using the DNA binding assays of Lee *et al.*. Ovals correspond to transcription factors and rectangles to modules (see Figure 3.22 for module names). Edge thickness and color represent various levels of literature support for the association.

to these sequence motifs. We used the DNA binding assays of Lee *et al.* (2002), that directly detect to which promoter regions a particular TF binds *in vivo*, and associated TFs with the motifs we learned. For each motif, we computed the fraction, among the motif targets, of genes bound by each TF, as measured in the data of Lee *et al.*. We used the hypergeometric distribution to assign a p-value to each such fraction and took p-value < 0.05 (Bonferroni corrected for multiple hypothesis testing) to be significant. Inspection of the significant associations showed that, in most cases, there was a unique motif that was significant for the TF and that a high fraction (> 0.5) of the TF's binding targets were among the motif target genes.

Based on this strong association between TFs and motifs, for each such TF-motif pair, we predicted that the TF regulates all the modules that are characterized by the motif. By combining all associations, we arrived at the regulatory network shown

in Figure 3.24. Of the 106 transcription factors measured in Lee *et al.*, 28 were enriched in the targets of at least one motif and were thus included in the resulting network. Of the 20 modules, 16 were associated with at least one TF. To validate the quality of the network, we searched the biological literature and compiled a list of experimentally verified targets for each of the 28 TFs in our network. We then marked each association between a TF and a module as *supported* if the module contains at least one gene that the TF is known to regulate from biological experiments. As current knowledge is limited, there are very few known targets for most TFs. Nevertheless, we found support for 21 of the 64 associations. We also computed the p-value for each supported association between a TF and a module, using the binomial distribution with probability of success $p = t/N$, where K is the total number of known targets for the TF and N is the total number of genes (1010). The p-value is then $P(X \geq \ell \mid X \sim B(p, n))$, where ℓ is the total number of known targets of the regulator in the supported module and n is the number of genes in the supported module. The resulting p-values are shown in Figure 3.24 by edge thickness and color.

We assigned a name to each module based on a concise summary of its gene content (compiled from gene annotations and the literature). The regulatory network thus contains predictions for the processes regulated by each TF, where for each association the prediction includes the motif through which the regulation occurs. In many cases, our approach recovered coherent biological processes along with their known regulators. Examples of such associations include: Hap4, the known activator of oxidative phosphorylation, with the oxidative phosphorylation module (13); Gcr2, a known positive regulator of glycolysis, with the glycolysis module (19); Mig1, a glucose repressor, with the galactose metabolism module (1); Ste12, involved in regulation of pheromone pathways, with the pheromone response module (4); and Met4, a positive regulator of sulfur amino acid metabolism, with the amino acid metabolism module (0).

3.6 Discussion

A central goal of molecular biology is the discovery of the regulatory mechanisms governing the expression of genes in the cell. In this chapter, we attempt to reveal one aspect of this regulation process, by using a unified probabilistic model over both gene expression and sequence data, whose goal is to identify transcriptional modules and the regulatory motif binding sites that control their regulation within a given set of experiments. Our results indicate that our method discovers modules that are both highly coherent in their expression profiles and significantly enriched for common motif binding sites in upstream regions of genes assigned to the same module. A comparison to the common approach of constructing clusters based only on expression and then learning a motif for each cluster shows that our method recovers modules that have a much higher correspondence to external biological knowledge of gene annotations and protein complex data.

As our results show, much can be gained by integrating data from multiple sources into a single unified model. In particular, the high correspondence between motif targets and protein complexes motivates an integration of the protein complex data together with the expression and sequence data that we used here into a single framework. Such integration can lead to constructing models with even higher consistency across different biological datasets.

We also showed how we can make use of the recent binding assays of Lee *et al.* (2002) to relate the actual transcription factors to the modules they regulate, resulting in a regulatory network; we show that many of the regulatory relationships discovered have support in the literature. Similar to the protein complex data, it is reasonable to assume that integrating this genome-wide location data into the model can actually improve our ability to learn biologically relevant regulatory relationships. Such an integration is very natural within our model: the *regulates* variable $g.R_i$ represents whether motif i is active (regulates) in gene g, and the genome-wide location data measures whether transcription factor i binds gene g anywhere in its promoter. Thus, we can view the location data as a noisy sensor to $g.R_i$ and connect it in such a way to our model, as shown in Figure 3.25. This demonstrates the power of the

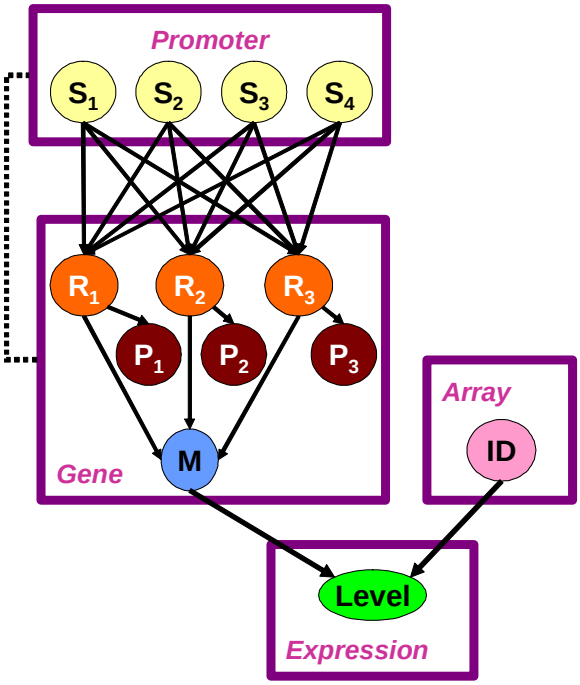


Figure 3.25: Shown is the PRM model that integrates genome-wide location data, in addition to the gene expression and sequence data, into a single unified probabilistic framework. Each P_i variable represents whether transcription factor i is bound anywhere in the promoter of the corresponding gene, as measured in the location data. This measurement is incorporated into the model as a noisy sensor to whether the motif that corresponds to transcription factor i is active in the gene.

probabilistic model representation, in that it provides us with a formal language for describing increasingly more detailed and complete pictures of the underlying biological system. Indeed, in a separate project (Segal *et al.*, 2002) we showed that this more comprehensive model, which also includes the genome-wide location data, performs better than models that use each data type in isolation. Moreover, this model also leads to reconstruction of key regulatory events in the cell cycle.

Despite the successes described above, our approach has several limitations. First, while our choice of the softmax distribution for the regulation model allows us to represent motif profiles in a natural and compact way, there are some regulation functions that it cannot express. For example, an *exclusive or* (XOR) regulation function, where a module is activated by one of two transcription factors but not by

both, cannot be represented using the additive softmax model. An example of a more common regulation function that cannot be represented using the softmax distribution is an *or* of *and* regulation function in which the two operands of the *and* are opposite of each other (i.e., a regulation function that is activated by $(A \cap B) \cup (\bar{A} \cap C)$ where A, B, and C represent transcription factors). Although such regulation functions are either less common or more complex, it is important to be aware of the softmax limitations.

A second and more important limitation of the regulation part of our model stems from the restrictions we placed on the weight matrix of the softmax distribution. Recall that we constrained this matrix to be sparse and required that all weights be positive. While these positivity and sparseness constraints lead to interpretable and concise motif profiles, they do not allow us to explicitly represent the absence of certain motifs in promoter regions of genes as part of the motif profile description. As such features may be important for characterizing some regulatory modules, our model may seek alternatives ways to represent them. For example, instead of representing the importance of the absence of a motif m from promoters of genes in some module j by a negative weight for motif m in the motif profile of module j, our model can achieve the same effect by placing positive weights for motif m in the motif profiles of all modules other than j. This implies that some motif weights that we learn may be added as an artifact of the positivity constraint and we must thus take care in their interpretation.

Given this limitation of the positivity constraint, it might be tempting to relax this assumption and allow the model to use negative weights within motif profiles. We note, however, that such an approach must be handled with care, as allowing for negative weights increases the degrees of freedom that the model has and may result in less intuitive and less interpretable models.

Recall from our analysis of the motif profiles shown in Figure 3.22, that we found several cases in which the promoters of the genes assigned to some module j were enriched for some motif m, yet our model did not place a positive weight for motif m in the motif profile of module j. While some of these cases may correspond to true biological findings regarding the activity of the involved motifs, others may be

explained by artifacts of the softmax model. For example, consider two modules, where the genes in one module are regulated by transcription factor A (i.e., the motif of factor A is present in the promoter regions of these genes), and the genes in the other module are regulated by two transcription factors A and B. A softmax model can distinguish the assignment of genes to these two modules from the assignment of genes to all other modules in at least two equivalent ways. One way is to add positive weights for the motif of factor A in both modules as well as a positive weight for the motif of factor B in the second module, such that the weight for the motif of factor A in the first module is greater than each of the other two weights. Such a model would indeed describe the full details of the regulation process in these two modules. However, a mathematically equivalent model would place all the weight of the second module on the motif of factor B. Thus, although motif A may be enriched in the second module, our learning procedure may still set its weight to zero, thereby missing some aspects of the regulation process.

There are several other important aspects of the gene regulation process that our approach does not model, such as the position and orientation of the motif relative to the site on the DNA in which transcription starts, and the proximity and relative order among the appearances of motifs within a motif combination. There is evidence that such features play a role in the regulation process of several modules (e.g., Pilpel *et al.* (2001), Segal and Sharan (2004), Beer and Tavazoie (2004)). As our method does not represent these features, it might fail to detect regulation in such cases. We note, however, that due to the modular nature of our probabilistic framework such extensions can easily be incorporated into our model by replacing the motif model with one that takes such features into account (e.g., the model that we developed in Segal and Sharan (2004)).

Another limitation of our model lies in the motif model, where we use PSSMs for representing the binding site motifs of transcription factors. As PSSMs assume independence among the positions within the binding site, they clearly represent an over-simplified model of the process by which transcription factors bind DNA. Barash *et al.* (2003) showed that better motif models can be built by relaxing this independence assumption and modeling dependencies among the binding site positions. A

complementary approach for improving our motif model is to use a more detailed quantitative model of the interaction between the transcription factor and the DNA (e.g., see Kalir and Alon (2004)).

Finally, in eukaryotes, there are many cases where a motif appears quite far from the promoter of the gene that it regulates (hundreds of thousands and sometimes millions of nucleotides away). This implies that in these organisms, we need to extend the regions in which we search for motifs beyond the 500 or 1000 base pairs upstream of the site where transcription starts. However, extending these search regions will introduce noise that is likely to confuse our current models. One way to address this issue and reduce the noise is to use additional features similar to those described above (e.g., proximity of motif appearances). Another option is to model the DNA state and accessibility in the various regions and exclude from consideration those sequences that are not accessible to binding by transcription factors. As new technologies for measuring these properties of the DNA are underway, such an approach may soon be feasible and is thus an interesting direction for future work.

Chapter 4

Module Networks: Identifying Regulators of Biological Processes

The complex functions of a living cell are carried out through the concerted activity of many genes and gene products. This activity is often coordinated by the organization of the genome into regulatory modules, or sets of co-regulated genes that achieve a common function. Such is the case for most of the metabolic pathways as well as for members of multi-protein complexes. Revealing this organization is essential for understanding cellular responses to internal and external signals.

In Chapter 3, we presented a method that attempts to reveal one aspect of this organization, namely identifying the cis-regulatory motifs through which regulation occurs and the combinations of motifs that play a role in controlling the expression of sets of co-regulated genes. As we showed, using only the raw sequence and expression data as input, our method identified many of the known motifs in yeast, and discovered modules of co-regulated genes and the combination of motifs that play a role in controlling their expression. However, while finding the motifs is highly informative, it only provides a partial picture of gene regulation as the identity of the transcription factors that bind these motifs can only be inferred indirectly. Thus, to complete the picture, we also need to identify which regulatory proteins bind these motifs and control the expression of the genes in the module.

In this chapter, we attempt to reveal precisely this missing aspect of the gene regulation story, by presenting a probabilistic model for discovering regulatory modules from gene expression data. Our procedure identifies modules of co-regulated genes, their regulators, and the conditions under which regulation occurs, generating testable hypotheses in the form "regulator 'X' regulates module 'Y' under conditions 'W'". We start with an overview of the key ideas underlying our approach, followed by a formal presentation of the probabilistic model and our algorithm for automatically learning the model from an input gene expression dataset. We then present detailed statistical and biological evaluation of the results we obtained when applying the method to an expression dataset that measured the response of yeast (*S. cerevisiae*) to various environmental stress conditions, demonstrating the ability of the method to identify functionally coherent modules and their correct regulators.

Finally, as mentioned above, our method generates testable and detailed hypotheses about gene regulation. As knowledge in the biological literature is limited, some hypotheses can neither be confirmed nor refuted. Thus, we have carried out microarray experiments in the wet lab to test the validity of three of these novel hypotheses. We close the chapter with the results from these experiments, demonstrating how our computational framework allowed us to suggest novel regulatory roles for three previously uncharacterized proteins in yeast.

4.1 Model Overview

Our goal in this chapter is to identify sets of co-regulated genes as well as find their (context-specific) regulators. Genome-wide expression profiles provide important information about the gene regulation process. Yet, the regulatory mechanisms of a cell are far from transparent in these data. Friedman *et al.* (2000) proposed to reveal such mechanisms by attempting to "explain" the expression of genes in the expression data via the expression of other genes in the data. The key idea is that if a gene's expression profile can be predicted as a function of the expression some set of genes, then there might be a regulatory relationship between the gene and the gene set. To infer such relationships, Friedman *et al.* (2000) learned a Bayesian network from the

expression data, in which the expression level of each gene is represented by a random variable. The learned network models the dependencies among the expression levels of the various genes, and regulator-regulatee relationships can then be inferred directly from the structure of the learned Bayesian network.

However, an organism typically has thousands of genes. Thus, if we follow the approach of Friedman *et al.* (2000), we could end up with a very large and detailed network. Unfortunately, in the gene expression domain, as in other complex domains, the amount of data is rarely enough to robustly learn such detailed models: while a typical gene expression dataset describes thousands of genes, there are at most a few hundred microarray experiments from which we can learn our network. In such situations, statistical noise is likely to lead to spurious dependencies, resulting in models that significantly overfit the data.

Moreover, if our goal is to use the learned structure of the Bayesian network for suggesting regulatory relationships, then we face additional challenges. First, due to the small number of microarray experiments, we are unlikely to have much confidence in the learned structure (Pe'er *et al.*, 2001). Second, a Bayesian network structure over thousands of variables is typically highly unstructured, and therefore very hard to interpret.

In this chapter, we propose an approach to address these issues. We start by observing that genes required for the same biological function or response are often co-regulated in order to coordinate their joint activity. Thus, the variables corresponding to the expression level of genes can be partitioned into sets so that, to a first approximation, the variables within each set have a similar set of dependencies and therefore exhibit a similar behavior.

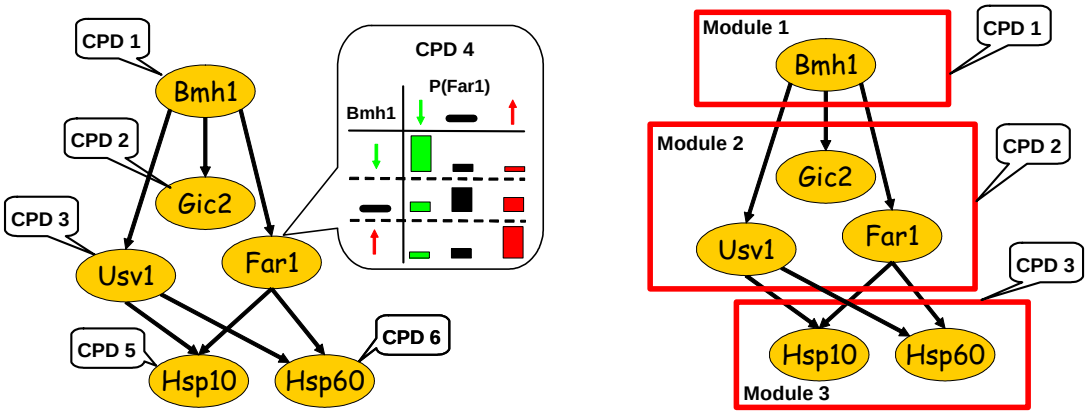
We define a new representation called a *module network*, which explicitly partitions the variables (genes) into *modules*. Each module represents a set of variables that have the same statistical behavior, i.e., they share the same set of parents and local probabilistic model. In the gene expression domain, this implies that co-regulated genes are grouped together into a module, and the model then uses the same parameterization and regulators to explain the expression of all the genes that are assigned

to the same module. By enforcing this constraint on the learned network, we significantly reduce the complexity of our model space as well as the number of parameters. As we show, these reductions lead to more robust estimation and better generalization on unseen data.

A module network consists of two components. The first is a component that defines a *template probabilistic model* shared by all the variables assigned to that model. This template includes both a set of regulating parents and the conditional probability distribution they specify over the expression of the module genes. The second component of a module network is a *module assignment function* that assigns each variable (gene) to one of the modules.

A module network can be viewed simply as a Bayesian network in which variables in the same module share parents and parameters (see Figure 4.1). Indeed, probabilistic models with shared parameters are common in a variety of applications, and are also used in other general representation languages, such as *dynamic Bayesian networks* (Dean and Kanazawa, 1989), *object-oriented Bayesian Networks* (Koller and Pfeffer, 1997), and the *probabilistic relational models* (Koller and Pfeffer, 1998, Friedman *et al.*, 1999b) we presented in Chapter 2. However, there are important differences between module networks and these other formalisms, which we further discuss in Section 4.4. In particular, we highlight the differences between module networks and PRMs, justifying the introduction of the new formalism of module networks.

The key difference between most of the above alternative formalisms and module networks worth mentioning at this point, is that in most other formalisms, the shared structure is imposed by the designer of the model, using prior knowledge about the domain. In contrast, we design a learning algorithm that directly searches for and finds sets of variables with similar behavior, which are then defined to be a module. By making the modular structure explicit, the module network representation can also provide important insights about the domain. In the gene regulation domain, such insights may include identifying the modular organization of genes into regulatory modules and the regulator genes controlling the expression of these modules. In contrast, even if a modular structure exists in the domain, it can be obscured by a



(a) Bayesian network (b) Module network

Figure 4.1: (a) A simple Bayesian network over gene expression variables; the expression level of Far1 is annotated with a visualization of its CPD, described as a different multinomial distribution for each value of its regulator gene Bmh1. (b) A simple module network; the boxes illustrate modules, where expression level variables share CPDs and parameters.

general Bayesian network learning algorithm which does not have an explicit representation for modules. For instance, it would be very difficult to interpret a Bayesian network on a genome-wide scale because of its intricate circuitry over thousands of genes.

We propose a fully automated procedure that discovers regulatory modules and their associated template probabilistic modules, which we term *regulation programs*, directly from gene expression profiles. For clarity, we now present a simplified overview that captures the algorithm's essence, and describe the full details in Section 4.3. Given a gene expression data set and a precompiled set of candidate regulatory genes, the algorithm simultaneously searches for a partition of genes into modules, and for a regulation program for each module that explains the behavior of the genes in the module. Our algorithm takes an iterative approach. We begin with an initial clustering of the genes based on their expression profiles. For each cluster of genes, the procedure searches for a regulation program that provides the best prediction of the expression profiles of genes in the module as a function of the expression of a small

number of genes from the candidate regulator set. After identifying regulation programs for all clusters, the algorithm re-assigns each gene to the cluster whose program best predicts that gene's behavior. The algorithm iterates until convergence, refining both the regulation program and the gene partition in each iteration. The procedure outputs a list of modules and associated regulation programs.

Our two-step iterative learning procedure attempts to search for the model with the highest score, in each step optimizing one of the models components: regulation programs or gene partition. An important property of our algorithm is that each iteration is guaranteed to improve the likelihood of the model, until convergence to a local maximum of the score.

4.2 Probabilistic Model

We now provide a formal definition of a module network. We assume that we are given a domain of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$. We use $Val(X_i)$ to denote the domain of values of the variable X_i.

As described above, a module represents a set of variables that share the same set of parents and the same CPD. As a notation, we represent each module by a *formal variable* that we use as a placeholder for the variables in the module. A *module set* $\mathcal{C}$ is a set of such formal variables $\mathbf{M}_1, \ldots, \mathbf{M}_K$. As all the variables in a module share the same CPD, they must have the same domain of values. We represent by $Val(\mathbf{M}_j)$ the set of possible values of the formal variable of the j'th module.

A module network relative to $\mathcal{C}$ consists of two components. The first defines a template probabilistic model for each module in $\mathcal{C}$; all of the variables assigned to the module will share this probabilistic model.

Definition 4.2.1: A *module network template* $\mathcal{T} = (\mathcal{S}, \boldsymbol{\theta})$ for $\mathcal{C}$ defines, for each module $\mathbf{M}_j \in \mathcal{C}$:

- a set of parents $\mathbf{Pa}_{\mathbf{M}_j} \subset \mathcal{X}$;
- a *conditional probability distribution template* $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$ which specifies a distribution over $Val(\mathbf{M}_j)$ for each assignment in $Val(\mathbf{Pa}_{\mathbf{M}_j})$.

We use $\mathcal{S}$ to denote the dependency structure encoded by $\{\mathbf{Pa}_{\mathbf{M}_j} : \mathbf{M}_j \in \mathcal{C}\}$ and $\boldsymbol{\theta}$ to denote the parameters required for the CPD templates $\{P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j}) : \mathbf{M}_j \in \mathcal{C}\}$. ∎

In our example from Figure 4.1, we have three modules M_1, M_2, and M_3, with $\mathbf{Pa}_{\mathbf{M}_1} = \emptyset$, $\mathbf{Pa}_{\mathbf{M}_2} = \{Bmh1\}$, and $\mathbf{Pa}_{\mathbf{M}_3} = \{Usv1, Far1\}$.

The second component is a module assignment function that assigns each variable $X_i \in \mathcal{X}$ to one of the K modules, $\mathbf{M}_1, \ldots, \mathbf{M}_K$. Clearly, we can only assign a variable to a module that has the same domain.

Definition 4.2.2: A *module assignment function* for $\mathcal{C}$ is a function $\mathcal{A} : \mathcal{X} \rightarrow \{1, \ldots, K\}$ such that $\mathcal{A}(X_i) = j$ only if $Val(X_i) = Val(\mathbf{M}_j)$. ∎

In our example, we have that $\mathcal{A}(Bmh1) = 1$, $\mathcal{A}(Gic2) = 2$, $\mathcal{A}(Usv1) = 2$, and so on.

A module network defines a probabilistic model by using the formal random variables $\mathbf{M}_j$ and their associated CPDs as templates that encode the behavior of all of the variables assigned to that module. Specifically, we define the semantics of a module network by "unrolling" a Bayesian network where all of the variables assigned to module $\mathbf{M}_j$ share the parents and conditional probability template assigned to $\mathbf{M}_j$ in $\mathcal{T}$. For this unrolling process to produce a well-defined distribution, the resulting network must be acyclic. Acyclicity can be guaranteed by the following simple condition on the module network:

Definition 4.2.3: Let $\mathcal{M}$ be a triple $(\mathcal{C}, \mathcal{T}, \mathcal{A})$, where $\mathcal{C}$ is a module set, $\mathcal{T}$ is a module network template for $\mathcal{C}$, and $\mathcal{A}$ is a module assignment function for $\mathcal{C}$. $\mathcal{M}$ defines a directed *module graph* $\mathcal{G}_{\mathcal{M}}$ as follows:

- the nodes in $\mathcal{G}_{\mathcal{M}}$ correspond to the modules in $\mathcal{C}$;
- $\mathcal{G}_{\mathcal{M}}$ contains an edge $\mathbf{M}_j \rightarrow \mathbf{M}_k$ if and only if there is a variable $X \in \mathcal{X}$ so that $\mathcal{A}(X) = j$ and $X \in \mathbf{Pa}_{\mathbf{M}_k}$.

We say that $\mathcal{M}$ is a *module network* if the module graph $\mathcal{G}_{\mathcal{M}}$ is acyclic. ∎

For example, for the module network of Figure 4.1(b), the module graph has the structure $\mathbf{M}_1 \rightarrow \mathbf{M}_2 \rightarrow \mathbf{M}_3$.

We can now define the semantics of a module network:

Definition 4.2.4: A module network $\mathcal{M} = (\mathcal{C}, \mathcal{T}, \mathcal{A})$ defines a *ground Bayesian network* $\mathcal{B}_\mathcal{M}$ over $\mathcal{X}$ as follows: For each variable $X_i \in \mathcal{X}$, where $\mathcal{A}(X_i) = j$, we define the parents of X_i in $\mathcal{B}_\mathcal{M}$ to be $\mathbf{Pa}_{\mathbf{M}_j}$, and its conditional probability distribution to be $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$, as specified in $\mathcal{T}$. The distribution associated with $\mathcal{M}$ is the one represented by the Bayesian network $\mathcal{B}_\mathcal{M}$. ∎

Returning to our example, the Bayesian network of Figure 4.1(a) is the ground Bayesian network of the module network of Figure 4.1(b).

Using the acyclicity of the module graph, we can now show that the semantics for a module network is well-defined.

Proposition 4.2.5: *The graph $\mathcal{G}_\mathcal{M}$ is acyclic if and only if the dependency graph of $\mathcal{B}_\mathcal{M}$ is acyclic.*

Proof: The proof follows from the direct correspondence between edges in the module graph and edges in the ground Bayesian network. Consider some edge $X_i \rightarrow X_j$ in $\mathcal{B}_\mathcal{M}$. By definition of the module graph, we must have an edge $\mathbf{M}_{\mathcal{A}(X_i)} \rightarrow \mathbf{M}_{\mathcal{A}(X_j)}$ in the module graph. Thus, any cyclic path in $\mathcal{B}_\mathcal{M}$ corresponds directly to a cyclic path in the module graph, proving one direction of the theorem. The proof in the other direction is slightly more subtle, as modules may (in principle) be empty. Assume, by way of contradiction, that there exists a cyclic path $\boldsymbol{p} = (\mathbf{M}_{k_1}, \ldots, \mathbf{M}_{k_l})$ in the module graph, where we have $\mathbf{M}_{k_i} \rightarrow \mathbf{M}_{k_{i+1}}$ for each $i = 1, \ldots l-1$ and $k_1 = k_l$. By definition of the module graph, $\mathbf{M}_{k_{i+1}}$ has some parent X_{k_i} such that $\mathcal{A}(X_{k_i}) = \mathbf{M}_{k_i}$ for $i = 1, \ldots, l-1$. It now follows that X_{k_i} is a parent of $X_{k_{i+1}}$ for each $i = 1, \ldots, l-1$, and that X_{k_l} is a parent of X_{k_1}. Thus, we also have a cyclic path in $\mathcal{B}_\mathcal{M}$, proving the desired contradiction. ∎

Corollary 4.2.6: *For any module network $\mathcal{M}$, $\mathcal{B}_\mathcal{M}$ defines a coherent probability distribution over $\mathcal{X}$.*

As we can see, a module network provides a succinct representation of the ground Bayesian network. A Bayesian network for the gene expression domain needs to represent thousands of CPDs. On the other hand, a module network can often represent a good approximation of the domain using a model with only few dozen CPDs.

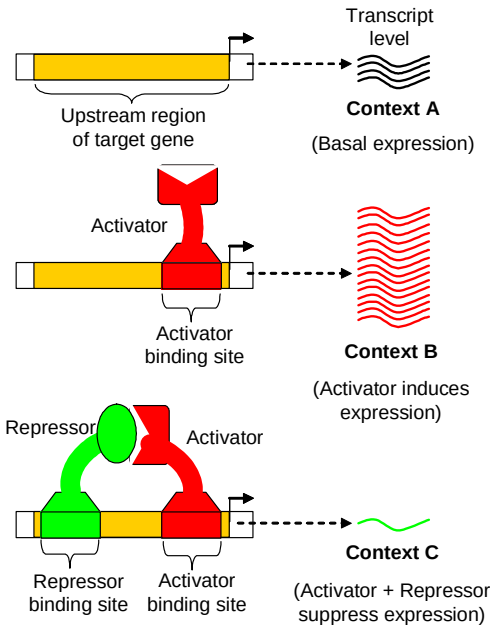
4.2.1 Application to Gene Expression

In applying the module network framework to the gene expression domain, we associate a variable with each gene, corresponding to its mRNA expression level. Genes with similar behavior are grouped into the same module and share the same CPD template, which specifies a distribution over the expression of the genes in the module as a function of the expression of the module's parents. In the gene expression domain, we refer to the CPD template of a module as a *regulation program*. Thus, the regulation program of a module specifies the set of regulatory genes that control the mRNA expression profile of the genes in the module as a function of the expression of the module's regulators.

Note that our approach relies on the assumption that the regulators are themselves transcriptionally regulated, so that their expression profiles provide evidence as to their activity level. Clearly, this assumption is sometimes violated, a common instance being transcription factors that are regulated post-translationally. In some cases, however, we can obtain additional evidence about regulation by considering the expression levels of those signaling molecules that may have an indirect transcriptional impact. We return to this issue in Section 4.9.

We represent each regulation program using a *regression tree* (Breiman *et al.*, 1984), as described in Section 2.2.1. We learn the structure of these regression trees automatically from the data, thereby discovering the control program for each module, including its set of regulators, their effect and combinatorial interactions.

We start by an example that demonstrates how a regression tree models biological regulation. Consider a group of genes (module) that are all regulated by the same combination of activator and repressor genes, resulting in the three distinct modes of regulation depicted in Figure 4.2(a). In context A, the genes in the module are not under transcriptional regulation and are thus in their basal expression level. In context B, the activator gene is up-regulated and as a result binds the upstream regions of the module genes, thereby inducing their transcription. In context C, a repressor gene is also up-regulated and as a result blocks transcription of the genes in the module, thereby decreasing their expression levels. We assume that our artificial biological system is such that the repressor gene requires the presence of the activator



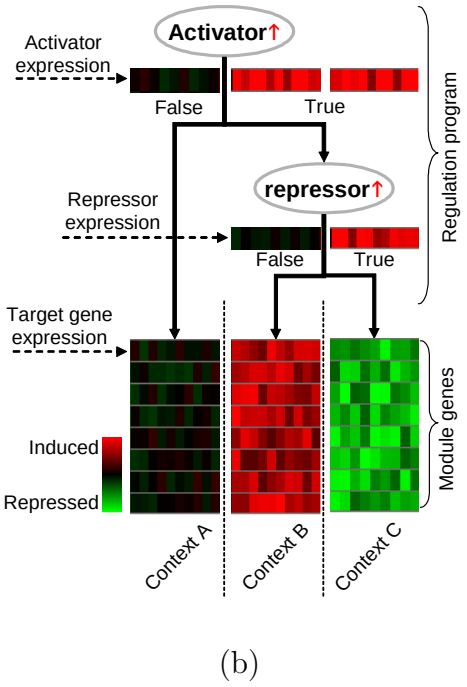


Figure 4.2: (a) Cartoon depicting three distinct modes of regulation for a group of genes. (b) Regulation tree that represents these modes of regulation. The expression of the regulatory genes is shown below their respective node. Each leaf of the regulation tree is a regulation context (bordered by dotted lines) as defined by the queries leading to the leaf. The arrays are sorted each into their respective context and the expression of the regulatory genes themselves is shown: each row corresponds to a gene and each column to an array.

gene in order to perform its function. Under this assumption, there is no need for a context in which only the repressor is up-regulated, as it will have no effect on the expression of the genes in the module and will be equivalent to context A.

A regulation program can represent the module's response to these different regulatory *contexts*. A *context* is a rule describing the qualitative behavior (up-regulation, no change or down-regulation) of a small set of genes that control the expression of the genes in the module. These rules are organized as a tree, where each path to a leaf in the tree defines a context via the tests on the path. This tree is composed of two basic building blocks: decision nodes and leaf nodes. Each decision node corresponds to one of the regulatory inputs and a query on its value (e.g., "is Hap4

up-regulated?"). Each decision node has two child nodes: the right child node is chosen when the answer to the corresponding query is true, the left node is chosen when not. For a given microarray, one begins at the root node and traverses down the tree in a path depending on the answers to the queries in that particular array, until a leaf is reached. We use Figure 4.2(b) to illustrate the path for context B: The root node's query is "Is the activator up-regulated?", which in context B is true and we thus continue to the right child. This right child contains the query "Is the inhibitor up-regulated?", which in context B is false and we thus take the left child. The resulting leaf corresponds to a regulation context in which the genes are over-expressed, which is indeed the behavior of the module genes in context B.

Each leaf of the regulation tree is a context that specifies the behavior of a set of arrays: those in which the tree traversal reaches the corresponding leaf. The response in each context is modeled as a Gaussian distribution over the expression values of the module's genes in these arrays; this distribution is encoded using a mean and variance stored at the corresponding leaf. The model semantics is: given a gene g in the module and an array a in a context, the probability of observing some expression value for gene g in array a is governed by the Gaussian distribution specified for that context. For each array, all genes in the same module follow the same Gaussian distribution. In a context where the genes are tightly co-regulated, the distribution will have a small variance. In a context where the genes are not tightly regulated, the distribution may have a large variance. Thus, a regression tree allows for expression profiles with different degrees of conservation of the mean behavior of the module.

This notion of a regulatory program has several key advantages. First, it captures combinatorial interactions; e.g., the module is strongly up-regulated if both Hap4 and Alpha2 are up-regulated. Second, it handles context specific effects and environmental conditions; e.g., Hap4 regulates the module only in the presence of respiratory carbon sources. Finally, it allows modules that have a conserved signature only in certain contexts; e.g., a module may have strong coordinated response to Gat1 up-regulation, yet have diffuse behavior in other settings. Within our framework, such context-specific signature is modeled through the variance of the Gaussian distribution in each context: the variance of the Gaussian will be small in contexts where the genes

are strongly coordinated and larger in contexts where the behavior of the genes is more diffused.

4.3 Learning the Model

We now turn to the task of learning module networks from data. Recall that a module network is specified by a set of modules $\mathcal{C}$, an assignment function $\mathcal{A}$ of nodes to modules, the parent structure $\mathcal{S}$ specified in $\mathcal{T}$, and the parameters $\boldsymbol{\theta}$ for the local probability distributions $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$. We assume that the set of modules $\mathcal{C}$ is given, and omit reference to it from now on.

One can consider several learning tasks for module networks, depending on which of the remaining aspects of the module network specification are known. We focus on the most general task of learning the network structure and the assignment function, as well as a Bayesian posterior over the network parameters. The other tasks are special cases that can be derived as a by-product of our algorithm.

Thus, we are given a training set $\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M]\}$, consisting of M instances drawn independently from an unknown distribution $P(\mathcal{X})$. Our primary goal is to learn a module network structure and assignment function for this distribution. We take a *score-based approach* to this learning task. We first define a scoring function that measures how well each candidate model fits the observed data. We adopt the Bayesian paradigm and derive a Bayesian scoring function similar to the Bayesian score for Bayesian networks (Cooper and Herskovits, 1992, Heckerman *et al.*, 1995). We then consider the algorithmic problem of finding a high scoring model.

4.3.1 Likelihood Function

We begin by examining the *data likelihood* function

$$L(\mathcal{M} : \mathcal{D}) = P(\mathcal{D} \mid \mathcal{M}) = \prod_{m=1}^{M} P(\boldsymbol{x}[m] \mid \mathcal{T}, \mathcal{A}).$$

This function plays a key role both in the parameter estimation task and in the definition of the structure score.

As the semantics of a module network is defined via the ground Bayesian network, we have that, in the case of complete data, the likelihood decomposes into a product of *local likelihood functions*, one for each variable. In our setting, however, we have the additional property that the variables in a module share the same local probabilistic model. Hence, we can aggregate these local likelihoods, obtaining a decomposition according to modules.

More precisely, let $\boldsymbol{X}^j = \{X \in \mathcal{X} \mid \mathcal{A}(X) = j\}$, and let $\boldsymbol{\theta}_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}}$ be the parameters associated with the CPD template $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$. We can decompose the likelihood function as a product of *module likelihoods*, each of which can be calculated independently and depends only on the values of $\boldsymbol{X}^j$ and $\mathbf{Pa}_{\mathbf{M}_j}$, and on the parameters $\boldsymbol{\theta}_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}}$:

$$\begin{aligned} & L(\mathcal{M} : \mathcal{D}) \\ &= \prod_{j=1}^{K} \left[\prod_{m=1}^{M} \prod_{X_i \in \boldsymbol{X}^j} P(x_i[m] \mid \mathbf{pa}_{\mathbf{M}_j}[m], \boldsymbol{\theta}_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}}) \right] \\ &= \prod_{j=1}^{K} L_j(\mathbf{Pa}_{\mathbf{M}_j}, \boldsymbol{X}^j, \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D}) \end{aligned} \tag{4.1}$$

If we are learning conditional probability distributions from the exponential family (e.g., discrete distribution, Gaussian distributions, and many others), then the local likelihood functions can be reformulated in terms of *sufficient statistics* of the data. The sufficient statistics summarize the relevant aspects of the data. Their use here is similar to that in Bayesian networks (Heckerman, 1998), with one key difference. In a module network, all of the variables in the same module share the same parameters. Thus, we pool all of the data from the variables in $\boldsymbol{X}^j$, and calculate our statistics based on this pooled data, similar to the pooling of data described for PRMs in Equation 2.2. More precisely, let $S_j(M_j, \mathbf{Pa}_{\mathbf{M}_j})$ be a sufficient statistic function for

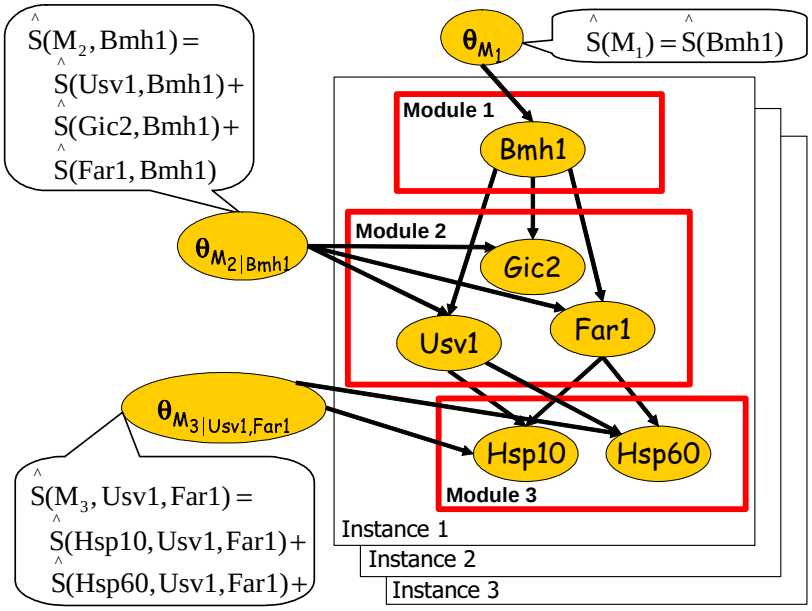


Figure 4.3: Shown is a plate model for three instances of the module network example of Figure 4.1(b). The CPD template of each module is connected to all variables assigned to that module (e.g., $\theta_{\mathbf{M}_2|Bmh1}$ is connected to *Usv1*, *Gic2*, and *Far1*). The sufficient statistics of each CPD template are the sum of the sufficient statistics of each variable assigned to the module and the module parents.

the CPD $P(M_j \mid \mathbf{Pa}_{\mathbf{M}_j})$. Then the value of the statistic on the data set $\mathcal{D}$ is

$$\hat{S}_j = \sum_{m=1}^{M} \sum_{X_i \in \boldsymbol{X}^j} S_j(x_i[m], \mathbf{pa}_{\mathbf{M}_j}[m]). \tag{4.2}$$

For example, in the case of networks that use only multinomial table CPDs, we have one sufficient statistic function for each joint assignment $x \in Val(\mathbf{M}_j), \boldsymbol{u} \in Val(\mathbf{Pa}_{\mathbf{M}_j})$, which is $\eta\{X_i[m] = x, \mathbf{pa}_{\mathbf{M}_j}[m] = \boldsymbol{u}\}$ — the indicator function that takes the value 1 if the event $(X_i[m] = x, \mathbf{Pa}_{\mathbf{M}_j}[m] = \boldsymbol{u})$ holds, and 0 otherwise. The statistic on the data is

$$\hat{S}_j[x, \boldsymbol{u}] \quad = \quad \sum_{m=1}^{M} \sum_{X_i \in \boldsymbol{X}^j} \eta\{X_i[m] = x, \mathbf{Pa}_{\mathbf{M}_j}[m] = \boldsymbol{u}\}$$

Given these sufficient statistics, the formula for the module likelihood function is:

$$L_j(\mathbf{Pa}_{\mathbf{M}_j}, \boldsymbol{X}^j, \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D}) = \prod_{x,\boldsymbol{u} \in Val(\mathbf{M}_j, \mathbf{Pa}_{\mathbf{M}_j})} \theta_{x|\boldsymbol{u}}^{\hat{S}_j[x,\boldsymbol{u}]}.$$

This term is precisely the one we would use in the likelihood of Bayesian networks with multinomial table CPDs. The only difference is that the vector of sufficient statistics for a local likelihood term is pooled over all the variables in the corresponding module.

For example, consider the likelihood function for the module network of Figure 4.1(b). In this network we have three modules. The first consists of a single variable and has no parents, and so the vector of statistics $\hat{S}[\mathbf{M}_1]$ is the same as the statistics of the single variable $\hat{S}[Bmh1]$. The second module contains three variables; thus, the sufficient statistics for the module CPD is the sum of the statistics we would collect in the ground Bayesian network of Figure 4.1(a): $\hat{S}[\mathbf{M}_2, Bmh1] = \hat{S}[Usv1, Bmh1] + \hat{S}[Gic2, Bmh1] + \hat{S}[Far1, Bmh1]$. Finally, $\hat{S}[\mathbf{M}_3, Usv1, Far1] = \hat{S}[Hsp10, Usv1, Far1] + \hat{S}[Hsp60, Usv1, Far1]$. An illustration of the decomposition of the likelihood and the associated sufficient statistics using the plate model is shown in Figure 4.3.

As usual, the decomposition of the likelihood function allows us to perform maximum likelihood or MAP parameter estimation efficiently, optimizing the parameters for each module separately.

4.3.2 Priors and the Bayesian Score

As we discussed, our approach for learning module networks is based on the use of a Bayesian score. Specifically, we define a model score for a pair $(\mathcal{S}, \mathcal{A})$ as the posterior probability of the pair, integrating out the possible choices for the parameters $\boldsymbol{\theta}$. We define an assignment prior $P(\mathcal{A})$, a structure prior $P(\mathcal{S} \mid \mathcal{A})$ and a parameter prior $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$. These describe our preferences over different networks *before* seeing the data. By Bayes' rule, we then have

$$P(\mathcal{S}, \mathcal{A} \mid \mathcal{D}) \propto P(\mathcal{A})P(\mathcal{S} \mid \mathcal{A})P(\mathcal{D} \mid \mathcal{S}, \mathcal{A})$$

where the last term is the *marginal likelihood*

$$P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) = \int P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}, \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid \mathcal{S}) d\boldsymbol{\theta}.$$

We define the Bayesian score as the log of $P(\mathcal{S}, \mathcal{A} \mid \mathcal{D})$, ignoring the normalization constant

$$\text{score}(\mathcal{S}, \mathcal{A} \ : \ \mathcal{D}) = \ \log P(\mathcal{A}) + \log P(\mathcal{S} \mid \mathcal{A}) + \log P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) \tag{4.3}$$

As with Bayesian networks, when the priors satisfy certain conditions, the Bayesian score decomposes. This decomposition allows to efficiently evaluate a large number of alternatives. The same general ideas carry over to module networks, but we also have to include assumptions that take the assignment function into account. Following is a list of conditions on the prior required for the decomposability of the Bayesian score in the case of module networks:

Definition 4.3.1: Let $P(\mathcal{A})$, $P(\mathcal{S} \mid \mathcal{A})$, $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$ be assignment, structure, and parameter priors.

- $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$ satisfies *parameter independence* if

$$P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A}) = \prod_{j=1}^{K} P(\boldsymbol{\theta}_{\mathbf{M}_j | \mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}, \mathcal{A}).$$

- $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$ satisfies *parameter modularity* if $P(\boldsymbol{\theta}_{\mathbf{M}_j | \mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}_1, \mathcal{A}) = P(\boldsymbol{\theta}_{\mathbf{M}_j | \mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}_2, \mathcal{A})$ for all structures $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\mathbf{Pa}_{\mathbf{M}_j}^{\mathcal{S}_1} = \mathbf{Pa}_{\mathbf{M}_j}^{\mathcal{S}_2}$.
- $P(\boldsymbol{\theta}, \mathcal{S} \mid \mathcal{A})$ satisfies *assignment independence* if $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A}) = P(\boldsymbol{\theta} \mid \mathcal{S})$ and $P(\mathcal{S} \mid \mathcal{A}) = P(\mathcal{S})$.
- $P(\mathcal{S})$ satisfies *structure modularity* if $P(\mathcal{S}) \propto \prod_j \rho_j(\mathcal{S}_j)$ where $\mathcal{S}_j$ denotes the choice of parents for module $\mathbf{M}_j$, and ρ_j is a distribution over the possible parent sets for module $\mathbf{M}_j$.
- $P(\mathcal{A})$ satisfies *assignment modularity* if $P(\mathcal{A}) \propto \prod_j \alpha_j(\mathcal{A}_j)$, where $\mathcal{A}_j$ is the

choice of variables assigned to module $\mathbf{M}_j$, and $\{\alpha_j : j = 1, \ldots, K\}$ is a family of functions from $2^{\mathcal{X}}$ to the positive reals. ∎

Parameter independence, parameter modularity, and structure modularity are the natural analogues of standard assumptions in Bayesian network learning (Heckerman *et al.*, 1995). Parameter independence implies that $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$ is a product of terms that parallels the decomposition of the likelihood in Equation 4.1, with one prior term per local likelihood term L_j. Parameter modularity states that the prior for the parameters of a module M_j depends only on the choice of parents for M_j and not on other aspects of the structure. Structure modularity implies that the prior over the structure $\mathcal{S}$ is a product of terms, one per each module.

Two assumptions are new to module networks. Assignment independence makes the priors on the parents and parameters of a module independent of the exact set of variables assigned to the module. Assignment modularity implies that the prior on $\mathcal{A}$ is proportional to a product of local terms, one corresponding to each module. Thus, the reassignment of one variable from one module $\mathbf{M}_i$ to another $\mathbf{M}_j$ does not change our preferences on the assignment of variables in modules other than i, j.

As for the standard conditions on Bayesian network priors, the conditions we define are not universally justified, and one can easily construct examples where we would want to relax them. However, they simplify many of the computations significantly, and are therefore useful even if they are only a rough approximation. Moreover, the assumptions, although restrictive, still allow broad flexibility in our choice of priors. For example, we can encode preference (or restrictions) on the assignments of particular variables to specific modules. In addition, we can also encode preference for particular module sizes.

For priors satisfying the assumptions of Definition 4.3.1, we can prove the decomposability property of the Bayesian score for module networks:

Theorem 4.3.2: Let $P(\mathcal{A})$, $P(\mathcal{S} \mid \mathcal{A})$, $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$ be assignment, structure, and parameter priors. When the assignment prior $P(\mathcal{A})$, structure prior $P(\mathcal{S} \mid \mathcal{A})$, and parameter prior $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$, satisfy the assumptions of Definition 4.3.1, the Bayesian

score decomposes into local *module scores*:

$$\begin{aligned}
\text{score}(\mathcal{S}, \mathcal{A} \ : \ \mathcal{D}) &= \sum_{j=1}^{K} \text{score}_{\mathbf{M}_j}(\mathbf{Pa}_{\mathbf{M}_j}, \mathcal{A}(\boldsymbol{X}^j) \ : \ \mathcal{D}) \\
\text{score}_{\mathbf{M}_j}(\boldsymbol{U}, \boldsymbol{X} \ : \ \mathcal{D}) &= \log \int L_j(\boldsymbol{U}, \boldsymbol{X}, \theta_{\mathbf{M}_j|\boldsymbol{U}} : \mathcal{D}) P(\boldsymbol{\theta}_{\mathbf{M}_j} \mid \mathcal{S}_j = \boldsymbol{U}) d\theta_{\mathbf{M}_j|\boldsymbol{U}} \quad (4.4) \\
&\quad + \log P(\mathcal{S}_j = \boldsymbol{U}) + \log P(\mathcal{A}_j = \boldsymbol{X})
\end{aligned}$$

where $\mathcal{S}_j = \boldsymbol{U}$ denotes that we chose a structure where $\boldsymbol{U}$ are the parents of module $\mathbf{M}_j$, and $\mathcal{A}_j = \boldsymbol{X}$ denotes that $\mathcal{A}$ is such that $\boldsymbol{X}^j = \boldsymbol{X}$.

Proof: Recall that we defined the Bayesian score of a module network as:

$$\text{score}(\mathcal{S}, \mathcal{A} \ : \ \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) + \log P(\mathcal{S} \mid \mathcal{A}) + \log P(\mathcal{A})$$

By the *structure modularity* and *assignment independence* assumptions in Definition 4.3.1, $\log P(\mathcal{S} \mid \mathcal{A})$ decomposes by modules, resulting in the second term, $\log P(\mathcal{S}_j = \boldsymbol{U})$, of Equation 4.4 of the module score for module j. By the *assignment modularity* assumption in Definition 4.3.1, $\log P(\mathcal{A})$ decomposes by modules, resulting in the third term, $\log P(\mathcal{A}_j = \boldsymbol{X})$ of Equation 4.4. For the first term of Equation 4.4, we can write:

$$\begin{aligned}
\log P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) &= \log \int P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}, \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A}) d\boldsymbol{\theta} \\
&= \log \prod_{i=1}^{K} \int L_j(\boldsymbol{U}, \boldsymbol{X}, \theta_{\mathbf{M}_j|\boldsymbol{U}} : \mathcal{D}) P(\boldsymbol{\theta}_{\mathbf{M}_j} \mid \mathcal{S}_j = \boldsymbol{U}) d\theta_{\mathbf{M}_j|\boldsymbol{U}} \\
&= \sum_{i=1}^{K} \log \int L_j(\boldsymbol{U}, \boldsymbol{X}, \theta_{\mathbf{M}_j|\boldsymbol{U}} : \mathcal{D}) P(\boldsymbol{\theta}_{\mathbf{M}_j} \mid \mathcal{S}_j = \boldsymbol{U}) d\theta_{\mathbf{M}_j|\boldsymbol{U}}
\end{aligned}$$

where in the second step we used the likelihood decomposition of Equation 4.1 and the parameter independence, parameter modularity, and assignment independence assumptions in Definition 4.3.1. ∎

As we shall see below, the decomposition of the Bayesian score plays a crucial rule in our ability to devise an efficient learning algorithm that searches the space of

module networks for one with high score. The only question is how to evaluate the integral over $\boldsymbol{\theta}_{\mathbf{M}_j}$ in $\text{score}_{\mathbf{M}_j}(\boldsymbol{U}, \boldsymbol{X} \ : \ \mathcal{D})$. This depends on the parametric forms of the CPD and the form of the prior $P(\boldsymbol{\theta}_{\mathbf{M}_j} \mid \mathcal{S})$. As discussed in Section 2.7.1, usually we choose priors that are *conjugate* to the parameter distributions. Such a choice often leads to closed form analytic formula of the value of the integral as a function of the sufficient statistics of $L_j(\mathbf{Pa}_{\mathbf{M}_j}, \boldsymbol{X}^j, \theta_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} : \mathcal{D})$. For example, using Dirichlet priors with multinomial table CPDs leads to the following formula for the integral over $\theta_{\mathbf{M}_j}$:

$$\log \int L_j(\boldsymbol{U}, \boldsymbol{X}, \theta_{\mathbf{M}_j|\boldsymbol{U}} : \mathcal{D}) P(\boldsymbol{\theta}_{\mathbf{M}_j} \mid \mathcal{S}_j = \boldsymbol{U}) d\theta_{\mathbf{M}_j|\boldsymbol{U}} = \sum_{\boldsymbol{u} \in \boldsymbol{U}} \log \frac{\Gamma(\sum_{v \in Val(\mathbf{M}_j)} \alpha_{\mathbf{M}_j}[v, \boldsymbol{u}])}{\prod_{v \in Val(\mathbf{M}_j)} \Gamma(\alpha_{\mathbf{M}_j}[v, \boldsymbol{u}])} \frac{\prod_{v \in Val(\mathbf{M}_j)} \Gamma(\hat{S}_j[v, \boldsymbol{u}] + \alpha_{\mathbf{M}_j}[v, \boldsymbol{u}])}{\Gamma(\sum_{v \in Val(\mathbf{M}_j)} \hat{S}_j[v, \boldsymbol{u}] + \alpha_{\mathbf{M}_j}[v, \boldsymbol{u}])}$$

where $\hat{S}_j[v, \boldsymbol{u}]$ is the sufficient statistics function as defined in Equation 4.2, and $\alpha_{\mathbf{M}_j}[v, \boldsymbol{u}]$ is the hyperparameter of the Dirichlet distribution given the assignment $\boldsymbol{u}$ to the parents $\boldsymbol{U}$ of $\mathbf{M}_j$. We show the full derivation of this formula in Appendix A. We note that in the above formula we have also made use of the *local parameter independence* assumption on the form of the prior (Heckerman, 1998), which states that the prior distribution for the different values of the parents are independent:

$$P(\boldsymbol{\theta}_{\mathbf{M}_j|\mathbf{Pa}_{\mathbf{M}_j}} \mid \mathcal{S}) = \prod_{\boldsymbol{u} \in Val(\mathbf{Pa}_{\mathbf{M}_j})} P(\boldsymbol{\theta}_{\mathbf{M}_j|\boldsymbol{u}} \mid \mathcal{S})$$

For continuous variables with Gaussian CPDs and Normal-Gamma prior distributions, the corresponding integral also has a simple closed form formula. We show this formula and its derivation in Appendix B.

4.3.3 Structure Search Step

Given a scoring function over networks, we now consider how to find a high scoring module network. This problem is a challenging one, as it involves searching over two combinatorial spaces simultaneously — the space of structures and the space of

module assignments. We therefore simplify our task by using an iterative approach that repeats two steps: In one step, we optimize a dependency structure relative to our current assignment function, and in the other, we optimize an assignment function relative to our current dependency structure.

The first type of step in our iterative algorithm learns the structure $\mathcal{S}$, assuming that $\mathcal{A}$ is fixed. This step involves a search over the space of dependency structures, attempting to maximize the score defined in Equation 4.3. This problem is analogous to the problem of structure learning in Bayesian networks. We use a standard heuristic search over the combinatorial space of dependency structures. We define a search space, where each state in the space is a legal parent structure, and a set of operators that take us from one state to another. We traverse this space looking for high scoring structures using a search algorithm such as greedy hill climbing.

In many cases, an obvious choice of local search operators involves steps of adding or removing a variable X_i from a parent set $\mathbf{Pa}_{\mathbf{M}_j}$. (Note that edge reversal is not a well-defined operator for module networks, as an edge from a variable to a module represents a one-to-many relation between the variable and all of the variables in the module.) When an operator causes a parent X_i to be added to the parent set of module $\mathbf{M}_j$, we need to verify that the resulting module graph remains acyclic, relative to the current assignment $\mathcal{A}$. Note that this step is quite efficient, as cyclicity is tested on the module graph, which contains only K nodes, rather than on the dependency graph of the ground Bayesian network, which contains n nodes (usually $n \gg K$).

Also note that, as in Bayesian networks, the decomposition of the score provides considerable computational savings. When updating the dependency structure for a module $\mathbf{M}_j$, the module score for another module $\mathbf{M}_k$ does not change, nor do the changes in score induced by various operators applied to the dependency structure of $\mathbf{M}_k$. Hence, after applying an operator to $\mathbf{Pa}_{\mathbf{M}_j}$, we need only update the change in score for those operators that involve $\mathbf{M}_j$. Moreover, only the delta score of operators that add or remove a parent from module $\mathbf{M}_j$ need to be recomputed after a change to the dependency structure of module $\mathbf{M}_j$, resulting in additional savings. This is analogous to the case of Bayesian network learning, where after applying a step that

changes the parents of a variable X, we only recompute the delta score of operators that affect the parents of X.

4.3.4 Module Assignment Search Step

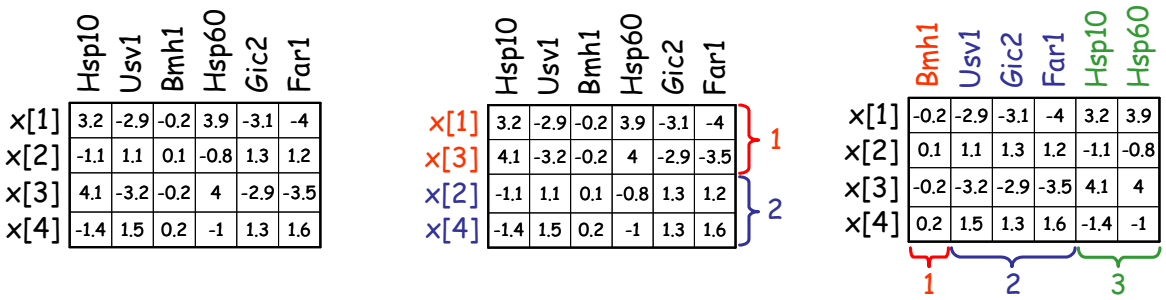
The second type of step in our iteration learns an assignment function $\mathcal{A}$ from data. This type of step occurs in two places in our algorithm: once at the very beginning of the algorithm, in order to initialize the modules; and once at each iteration, given a module network structure $\mathcal{S}$ learned in the previous structure learning step.

Module Assignment as Clustering

In this step, our task is as follows: Given a fixed structure $\mathcal{S}$ we want to find $\mathcal{A} = \text{argmax}_{\mathcal{A}'}\text{score}_{\mathbf{M}}(\mathcal{S}, \mathcal{A}' : \mathcal{D})$. Interestingly, we can view this task as a clustering problem. A module consists of a set of variables that have the same probabilistic model. Thus, for a given instance, two different variables in the same module define the same probabilistic model, and therefore should have similar behavior. We can therefore view the module assignment task as the task of clustering variables into sets, so that variables in the same set have a similar behavior across all instances.

For example, in our gene expression domain, we would cluster genes based on the similarity of their behavior over different arrays. Note that this clustering task is the "inverse" of the standard clustering task applied to our data set: In a standard clustering algorithm (e.g., AutoClass (Cheeseman *et al.*, 1988)), we cluster data instances (arrays) based on the similarity of the variables characterizing them. Here, we view instances as features of variables, and try to cluster variables. (See Figure 4.4.)

However, there are several key differences between this task and a standard clustering task. First, in general, the probabilistic model associated with each cluster has structure, as defined by the CPD template associated with the cluster (module). Moreover, our setting places certain constraints on the clustering, so that the resulting assignment function will induce a legal (acyclic) module network.



(a) Data (b) Instance-based clustering (c) Initialization

Figure 4.4: Relationship between the module network procedure and clustering. Finding an assignment function can be viewed as a clustering of the variables whereas standard clustering typically clusters instances. Shown is sample data for the example of Figure 4.1, where the rows correspond to instances and the columns correspond to variables. (a) Data. (b) Standard clustering of the data in (a). Note that $x[2]$ and $x[3]$ were swapped to form the clusters. (c) Initialization of the assignment function for the module network procedure for the data in (a). Note that variables were swapped in their location to reflect the initial assignment into three modules.

Module Assignment Initialization

In the initialization phase, we exploit the clustering perspective directly, using a form of hierarchical agglomerative clustering that is tailored to our application. Our clustering algorithm uses an objective function that evaluates a partition of variables into modules by measuring the extent to which the module model is a good fit to the features (instances) of the module variables. This algorithm can also be thought of as performing *model merging* (as in (Elidan and Friedman, 2001, Cheeseman *et al.*, 1988)) in a simple probabilistic model.

In the initialization phase, we do not yet have a learned structure for the different modules. Thus, from a clustering perspective, we consider a simple naive Bayes model for each cluster, where the distributions over the different features within each cluster are independent and have a separate parameterization. We begin by forming a cluster for each variable, and then merge two clusters whose probabilistic models over the features (instances) are similar.

¿From a module network perspective, the naive Bayes model can be obtained by introducing a dummy variable U that encodes training instance identity — $u[m] = m$

for all m. Throughout our clustering process, each module will have $\mathbf{Pa}_{\mathbf{M}_i} = \{U\}$, providing exactly the effect that, for each variable X_i, the different values $x_i[m]$ have separate probabilistic models. We then begin by creating n modules, with $\mathcal{A}(X_i) = i$. In this module network, each instance and each variable has its own local probabilistic model.

We then consider all possible legal module mergers (those corresponding to modules with the same domain), where we change the assignment function to replace two modules j_1 and j_2 by a new module $j_{1,2}$. This step corresponds to creating a cluster containing the variables X_{j_1} and X_{j_2}. Note that, following the merger, the two variables X_{j_1} and X_{j_2} now must share parameters, but each instance still has a different probabilistic model (enforced by the dependence on the instance ID U). We evaluate each such merger by computing the score of the resulting module network. Thus, the procedure will merge two modules that are similar to each other across the different instances. We continue to do these merge steps until we construct a module network with the desired number of modules, as specified in the original choice of $\mathcal{C}$.

Module Reassignment

In the module reassignment step, the task is more complex. We now have a given structure $\mathcal{S}$, and wish to find $\mathcal{A} = \text{argmax}_{\mathcal{A}'}\text{score}_{\mathbf{M}}(\mathcal{S}, \mathcal{A}' : \mathcal{D})$. We thus wish to take each variable X_i, and select the assignment $\mathcal{A}(X_i)$ that provides the highest score.

At first glance, we might think that we can decompose the score across variables, allowing us to determine independently the optimal assignment $\mathcal{A}(X_i)$ for each variable X_i. Unfortunately, this is not the case. Most obviously, the assignments to different variables must be constrained so that the module graph remains acyclic. For example, if $X_1 \in \mathbf{Pa}_{\mathbf{M}_i}$ and $X_2 \in \mathbf{Pa}_{\mathbf{M}_j}$, we cannot simultaneously assign $\mathcal{A}(X_1) = j$ and $\mathcal{A}(X_2) = i$. More subtly, the Bayesian score for each module depends non-additively on the sufficient statistics of all the variables assigned to the module. (The log-likelihood function is additive in the sufficient statistics of the different variables, but the log marginal likelihood is not.) Thus, we can only compute the delta score for moving a variable from one module to another given a *fixed* assignment of the other variables to these two modules.

Input:
  D // Data set
  $\mathcal{A}_0$ // Initial assignment function
  $\mathcal{S}$ // Given dependency structure
Output:
  $\mathcal{A}$ // improved assignment function
Sequential-Update
  $\mathcal{A} = \mathcal{A}_0$
  Loop
    For $i = 1$ to n
      $score^* = \text{score}(\mathcal{S}, \mathcal{A} \ : \ \mathcal{D})$
      For $j = 1$ to K
        $\mathcal{A}' = \mathcal{A}$ except that $\mathcal{A}'(X_i) = j$
        If $\langle \mathcal{G}_\mathcal{M}, \mathcal{A}' \rangle$ is cyclic, **continue**
        If $\text{score}(\mathcal{S}, \mathcal{A}' \ : \ \mathcal{D}) > score^*$
          $\mathcal{A} = \mathcal{A}'$
          $score^* = \text{score}(\mathcal{S}, \mathcal{A}' \ : \ \mathcal{D})$
  Until no reassignments to any of $X_1, \ldots X_n$
  Return $\mathcal{A}$

Figure 4.5: Outline of sequential algorithm for finding the module assignment function

We therefore use a sequential update algorithm that reassigns the variables to modules one by one. The idea is simple. We start with an initial assignment function $\mathcal{A}^0$, and in a "round-robin" fashion iterate over all of the variables one at a time, and consider changing their module assignment. When considering a reassignment for a variable X_i, we keep the assignments of all other variables fixed and find the optimal legal (acyclic) assignment for X_i relative to the fixed assignment. We continue reassigning variables until no single reassignment can improve the score. An outline of this algorithm appears in Figure 4.5

The key to the correctness of this algorithm is its sequential nature: Each time a variable assignment changes, the assignment function as well as the associated sufficient statistics are updated before evaluating another variable. Thus, each change made to the assignment function leads to a legal assignment which improves the score. Our algorithm terminates when it can no longer improve the score. Hence, it converges

Input:
D // Data set
K // Number of modules
Output:
$\mathbf{M}$ // A module network
Learn-Module-Network
$\mathcal{A}_0$ = cluster $\mathcal{X}$ into K modules
$\mathcal{S}_0$ = empty structure
Loop $t = 1, 2, \ldots$ until convergence
$\mathcal{S}_t$ = Greedy-Structure-Search$(\mathcal{A}_{t-1}, \mathcal{S}_{t-1})$
$\mathcal{A}_t$ = Sequential-Update$(\mathcal{A}_{t-1}, \mathcal{S}_t)$;
Return $\mathbf{M} = (\mathcal{A}_t, \mathcal{S}_t)$

Figure 4.6: Outline of the *module network* learning algorithm. Greedy-Structure-Search successively applies operators that change the structure as long as each such operator results in a legal structure and improves the module network score

to a local maximum, in the sense that no single assignment change can improve the score.

The computation of the score is the most expensive step in the sequential algorithm. Once again, the decomposition of the score plays a key role in reducing the complexity of this computation: When reassigning a variable X_i from one module $\mathbf{M}_{old}$ to another $\mathbf{M}_{new}$, only the local score of these modules changes. The module score of all other modules remains unchanged. The rescoring of these two modules can be accomplished efficiently by subtracting X_i's statistics from the sufficient statistics of $\mathbf{M}_{old}$ and adding them to those of $\mathbf{M}_{new}$.

4.3.5 Algorithm Summary

To summarize, our algorithm starts with an initial assignment of variables to modules. In general, this initial assignment can come from anywhere, and may even be a random guess. We choose to construct it using the clustering-based idea described in the previous section. The algorithm then iteratively applies the two steps described above: learning the module dependency structures, and reassigning variables to modules.

These two steps are repeated until convergence. An outline of the module network learning algorithm is shown in Figure 4.6.

Each of these two steps — structure update and assignment update — is guaranteed to either improve the score or leave it unchanged. We can thus prove:

Theorem 4.3.3: The iterative module network learning algorithm converges to a local maximum of score$(\mathcal{S}, \mathcal{A} : \mathcal{D})$.

4.3.6 Learning with Regression Trees

We now review the family of conditional distributions we use in the gene expression domain. As gene expression data is continuous, we use a conditional probability model represented as a *regression tree* (Breiman *et al.*, 1984). We reviewed regression trees in Section 2.2.1. For convenience, we repeat their presentation here, focusing on the implementation details relevant to our application.

For our purposes, a regression tree T for $P(X \mid \boldsymbol{U})$ is defined via a rooted binary tree, where each *node* in the tree is either a *leaf* or an *interior node*. Each interior node is labeled with a test $U < u$ on some variable $U \in \boldsymbol{U}$ and $u \in \mathbb{R}$. Such an interior node has two outgoing *arcs* to its children, corresponding to the outcomes of the test (true or false). The tree structure T captures the *local* dependency structure of the conditional distribution. The parameters of T are the distributions associated with each leaf. In our implementation, each leaf ℓ is associated with a univariate Gaussian distribution over values of X, parameterized by a mean μ_ℓ and variance σ_ℓ^2. An example of a regression tree CPD is shown in Figure 4.7.

To learn module networks with regression-tree CPDs, we must extend our previous discussion by adding another component to $\mathcal{S}$ that represents the trees $T_1, \ldots, T_K$ associated with the different modules. Once we specify these components, the above discussion applies with several small differences. These issues are similar to those encountered when introducing decision trees to Bayesian networks (Chickering *et al.*, 1997, Friedman and Goldszmidt, 1998), so we discuss them only briefly.

Given a regression tree T_j for $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$, the corresponding sufficient statistics are the statistics of the distributions at the leaves of the tree. For each leaf ℓ in the

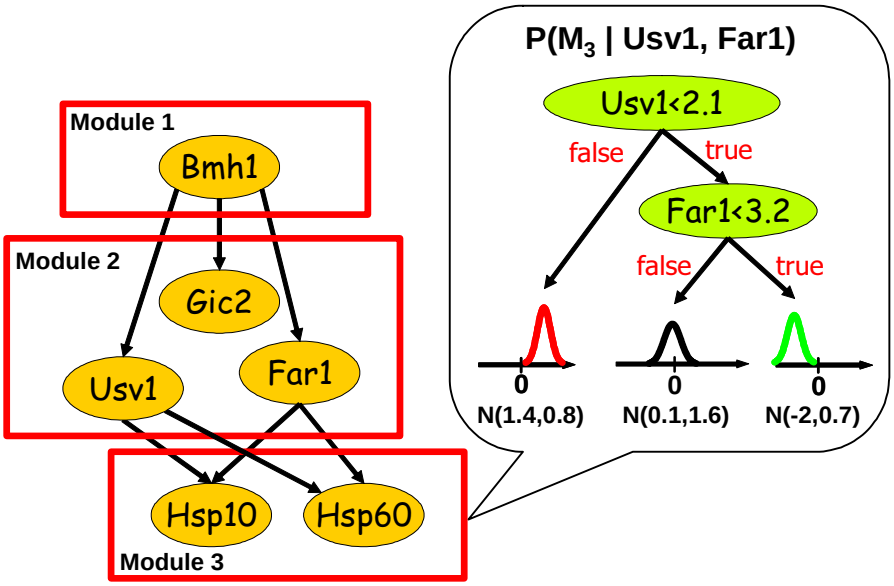


Figure 4.7: Example of a regression tree with univariate Gaussian distributions at the leaves for representing the CPD $P(\mathbf{M}_3 \mid \mathit{Usv1}, \mathit{Far1})$, associated with $\mathbf{M}_3$. The tree has internal nodes labeled with a test on the variable (e.g., $\mathit{Usv1} < 2.1$). Each univariate Gaussian distribution at a leaf is parameterized by a mean and a variance. The tree structure captures the local dependency structure of the conditional distributions. In the example shown, when $\mathit{Usv1} \geq 2.1$, then the distribution over values of variables assigned to $\mathbf{M}_3$ will be Gaussian with mean 1.4 and standard deviation 0.8 regardless of the value of *Far1*.

tree, and for each data instance $\boldsymbol{x}[m]$, we let $\ell_j[m]$ denote the leaf reached in the tree given the assignment to $\mathbf{Pa}_{\mathbf{M}_j}$ in $\boldsymbol{x}[m]$. The module likelihood decomposes as a product of terms, one for each leaf ℓ. Each term is the likelihood for the Gaussian distribution $\mathcal{N}(\mu_\ell; \sigma^2_\ell)$, with the sufficient statistics for a Gaussian distribution.

$$\begin{aligned}
\langle \hat{S}_{j,\ell} \rangle_0 &= \sum_m \sum_{X_i \in \boldsymbol{X}^j} \eta\{\ell_j[m] = \ell\} \\
\langle \hat{S}_{j,\ell} \rangle_1 &= \sum_m \sum_{X_i \in \boldsymbol{X}^j} \eta\{\ell_j[m] = \ell\} x_i \\
\langle \hat{S}_{j,\ell} \rangle_2 &= \sum_m \sum_{X_i \in \boldsymbol{X}^j} \eta\{\ell_j[m] = \ell\} x_i^2
\end{aligned} \tag{4.5}$$

The local module score further decomposes into independent components, one for

each leaf ℓ. Here, we use a Normal-Gamma prior (DeGroot, 1970) for the distribution at each leaf: Letting $\tau_\ell = 1/\sigma_\ell^2$ stand for the precision at leaf ℓ, we define: $P(\mu_\ell, \tau_\ell) = P(\mu_\ell \mid \tau_\ell)P(\tau_\ell)$, where $P(\tau_\ell) \sim \Gamma(\alpha_0, \beta_0)$ and $P(\mu_\ell \mid \tau_\ell) \sim \mathcal{N}(\mu_0; (\lambda_0\tau_\ell)^{-1})$, where we assume that all leaves are associated with the same prior. Letting $\langle \hat{S}_{j,\ell} \rangle_i$ be defined as in Equation 4.5, we have that the component of the log marginal likelihood associated with a leaf ℓ of module j is given by (see Appendix B for a full derivation):

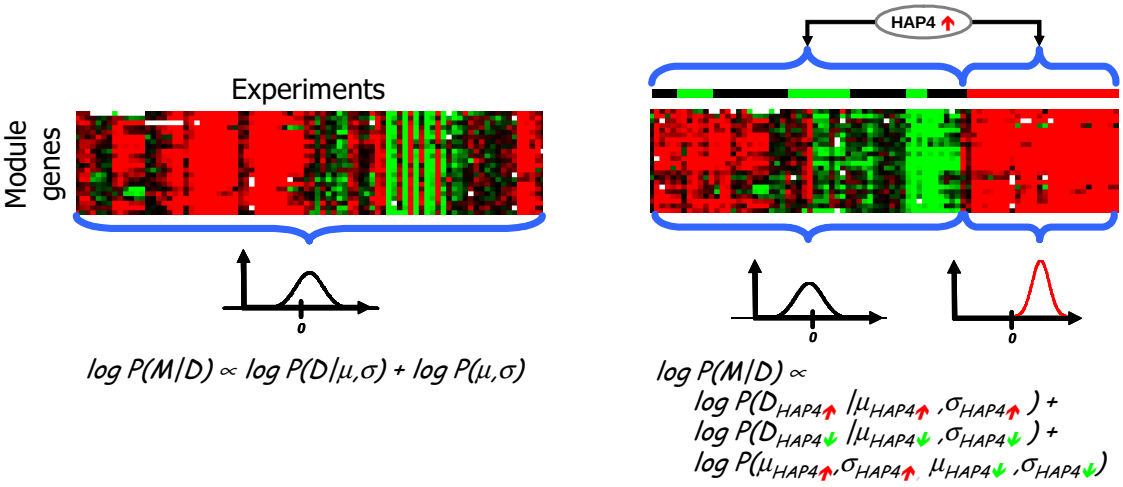
$$-\tfrac{N}{2}\log(2\pi) + \tfrac{1}{2}\log\lambda_0 + \alpha_0 \log\beta_0 - \log\Gamma(\alpha_0) + \log\Gamma(\alpha_1) - \alpha_1\log\beta_1 - \tfrac{1}{2}\log\lambda_1$$

where

$$\begin{aligned}
\lambda_1 &= \lambda_0 + \langle \hat{S}_{j,\ell} \rangle_0 \\
\mu_1 &= \frac{\lambda_0\mu_0 + \langle \hat{S}_{j,\ell} \rangle_1}{\lambda_1} \\
\alpha_1 &= \alpha_0 + \frac{\langle \hat{S}_{j,\ell} \rangle_0}{2} \\
\beta_1 &= \beta_0 + \frac{1}{2}\langle \hat{S}_{j,\ell} \rangle_2 - \frac{1}{2}\langle \hat{S}_{j,\ell} \rangle_1 + \frac{\langle \hat{S}_{j,\ell} \rangle_0 \lambda_0 \left(\frac{\langle \hat{S}_{j,\ell} \rangle_1}{\langle \hat{S}_{j,\ell} \rangle_0} - \mu_0 \right)^2}{2\lambda_1}
\end{aligned}$$

When performing structure search for module networks with regression-tree CPDs, in addition to choosing the parents of each module, we must also choose the associated tree structure. We use the search strategy proposed by Chickering *et al.* (1997), where the search operators are leaf splits. Such a *split* operator replaces a leaf in a tree T_j with an internal node with some test on a variable U. The two branches below the newly created internal node point to two new leaves, each with its associated Gaussian. This operator must check for acyclicity, as it implicitly adds U as a parent of $\mathbf{M}_j$. The tree is grown from the root to its leaves. Figure 4.8 illustrates the split operation in the context of the gene expression domain, where the split partitions the arrays into two different modes of regulation. A good partition is one that results in two distinct distributions. In the example of Figure 4.8, the module genes are all strongly up-regulated in arrays in which Hap4 is up-regulated.

When performing the search, we consider splitting each possible leaf on each



(a) (b)

Figure 4.8: Example of a search step in the regression tree learning algorithm. (a) Before introducing a split, the arrays are in no particular order. (b) The query "is Hap4 up-regulated" partitions the arrays into two distinct distributions. In arrays in which Hap4 is up-regulated (right) the module genes are strongly up-regulated. The distribution for each of the two leaves is distinctly different.

possible parent U and each value u. As always in regression-tree learning, we do not have to consider all real values u as possible split points; it suffices to consider values that arise in the data set. Moreover, under an appropriate choice of prior (i.e., an independent prior for each leaf), regression-tree learning provides another level of score decomposition: The score of a particular tree is a sum of scores for the leaves in the tree. Thus, a split operation on one leaf in the tree does not affect the score component of another leaf, so that operators applied to other leaves do not need to re-evaluated.

4.4 Related Work

¿From the gene regulation discovery perspective, module networks represent a significant advance over existing approaches for analyzing gene expression data. These approaches (Eisen *et al.*, 1998, Wu *et al.*, 2002, Ihmels *et al.*, 2002, Halfon *et al.*, 2002, Tanay *et al.*, 2002) allow the identification of groups of co-expressed genes —

genes that have similar patterns of expression. However, the regulatory programs of these groups can be suggested only indirectly, for example, by finding common *cis*-regulatory binding sites in the upstream regions of genes within each group (Spellman *et al.*, 1998, Roth *et al.*, 1998, Tavazoie *et al.*, 1999, Pilpel *et al.*, 2001, Segal *et al.*, 2002, 2003e), as discussed in Section 3.4. In contrast, our method discovers both regulatory modules and their control programs, suggesting concrete regulators for each module, their effect and combinatorial interactions, and the experimental conditions under which they are active.

We note that other global models for regulatory networks have been suggested, based on the following general idea: The regulatory network is a directed graph $\mathcal{G}$ and each node in $\mathcal{G}$ corresponds to a specific gene that behaves according to some function of its parents in $\mathcal{G}$. These approaches include: Boolean network models (Weaver *et al.*, 1999, D'Haeseleer *et al.*, 1999), where each gene is either on or off depending on some boolean function of its parents; Linear models (Akutsu *et al.*, 1998, Somogyi *et al.*, 1996), where each gene is modeled as a continuous linear function of its parents; and Bayesian networks (Friedman *et al.*, 2000), where each gene is a modeled as a stochastic function of its parents. Of these approaches, the one shown to produce the best biological results and the one closest to module networks is the Bayesian network approach. We review this approach next and discuss its relation to our models.

4.4.1 Relation to Bayesian Networks

As mentioned above, Friedman *et al.* (2000) suggested the use of Bayesian networks for discovering regulatory relationships from gene expression data. However, there are several reasons why a learned module network is a better model than a learned Bayesian network for this task. Most obviously, parameter sharing between genes (variables) in the same module allows each parameter to be estimated based on a much larger sample. Moreover, this allows us to learn dependencies that are considered too weak based on statistics of single genes. These are well-known advantages of parameter sharing; but a novel aspect of our method is that we automatically determine which genes have shared parameters and regulatory structure.

Furthermore, the assumption of shared structure significantly restricts the space of possible dependency structures, allowing us to learn more robust models than those learned in a classical Bayesian network setting. While the genes in the same module might behave (approximately) according to the same model in the underlying distribution, this will often not be the case in the empirical distribution based on a limited number of samples. A Bayesian network learning algorithm will treat each gene separately, optimizing the parent set and CPD for each gene in an independent manner. In the very high-dimensional domains in which we are interested, there are bound to be spurious correlations that arise from sampling noise, inducing the algorithm to choose parent sets that do not reflect real dependencies, and will not generalize to unseen data. Conversely, in a module network setting, a spurious correlation would have to arise between a possible parent and a large number of other genes before the algorithm would find it worthwhile to introduce the dependency.

4.4.2 Relation to OOBNs

¿From the representation perspective, module networks are related both to the framework of *object-oriented Bayesian networks* (OOBNs) (Koller and Pfeffer, 1997) and to the framework of *probabilistic relational models* (PRMs) (Koller and Pfeffer, 1998, Friedman *et al.*, 1999b). As discussed in Chapter 2, these frameworks extend Bayesian networks to a setting involving multiple related objects, and allow the attributes of objects of the same *class* to share parameters and dependency structure. We can view the module network framework as a restriction of these frameworks, where we have one object for every variable X_i, with a single attribute corresponding to the value of X_i. Each module can be viewed as a class, so that the variables in a single module share the same probabilistic model. As the module assignments are not known in advance, module networks correspond most closely to the variant of these frameworks where there is *type uncertainty* — uncertainty about the class assignment of objects. However, despite this high-level similarity, the module network framework differs in certain key points from both OOBNs and PRMs, with significant impact on the learning task.

In OOBNs, objects in the same class must have the same internal structure and parameterization, but can depend on different sets of variables (as specified in the mapping of variables in an object's interface to its actual inputs). By contrast, in a module network, all of the variables in a module (class) must have the same specific parents. This assumption greatly reduces the size and complexity of the hypothesis space, leading to a more robust learning algorithm. On the other hand, this assumption requires that we be careful in making certain steps in the structure search, as they have more global effects than on just one or two variables. Due to these differences, we cannot simply apply an OOBN structure-learning algorithm, such as the one proposed by Langseth and Nielsen (2003), to such complex, high-dimensional domains.

4.4.3 Relation to PRMs

To see the relationship between module networks and PRMs, we first consider a special case of module networks, in which each variable X, that is a parent of at least one module (i.e., $X \in \mathbf{Pa}_{\mathbf{M}_j}$ for some $1 \leq j \leq K$), is assigned to a designated module that contains only X and has no parents. We further assume that the module assignments of these parent variables are fixed during the entire module network learning process. We can represent this restricted module network using the PRM framework described in Section 2.3. In the gene expression domain, where each variable corresponds to a gene and the parent variables are the candidate regulatory genes, one possible PRM model would have: a $\mathsf{Module_j}$ class for each of the K modules with no attributes; an object for each gene that is not a parent of any module, whose class is $\mathsf{Module_j}$ if the gene is assigned to module j; an Array class with one $Regulator_i$ attribute for each of the regulatory genes that are parents of at least one module, representing the mRNA expression level measured for the regulator in the array; and an $\mathsf{Expression}_j$ class for each of the K modules, with reference slots *Module* (with range type $\mathsf{Module_j}$) and *Array* (with range type Array). Each $\mathsf{Expression_j}$ class also has a single *Level* attribute representing the mRNA expression level measured for the corresponding gene in the corresponding array. The relational schema of such a PRM for an example module

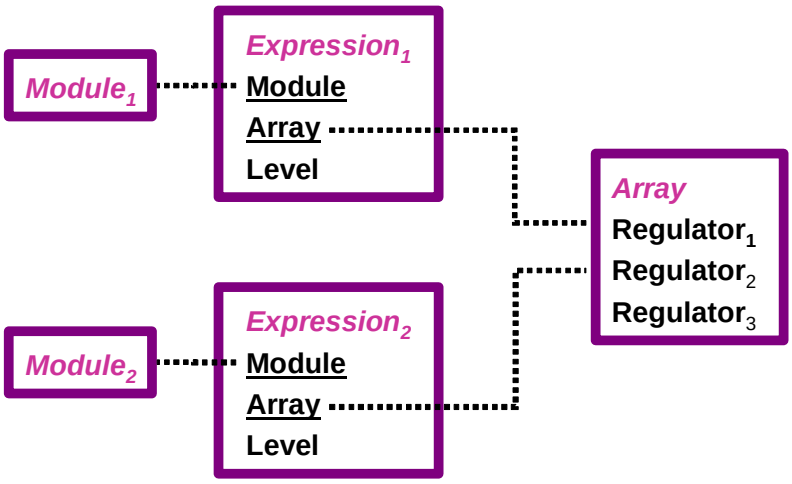


Figure 4.9: A relational schema of a PRM for a simple module network with two modules and three regulatory genes. The underlined attributes are reference slots of the class and the dashed line indicates the types of objects referenced.

network with two modules and three regulatory genes is shown in Figure 4.9.

To complete the mapping of this restricted module network to the PRM framework, we need to specify the dependency structure of the PRM and the CPDs that we associate with the attributes of each class. For this special case, the specification is straightforward: the parents of the *Level* attribute of each $\mathsf{Expression}_j$ class are simply $\mathbf{Pa}_{\mathbf{M}_j}$ — the regulatory genes that are parents of module j in the module network; the CPD for the *Level* attribute of each $\mathsf{Expression}_j$ class is simply $P(\mathbf{M}_j \mid \mathbf{Pa}_{\mathbf{M}_j})$ — the CPD template associated with module j. Finally, the CPD for the $Regulator_i$ attribute of the Array class is also taken from the designated module that $Regulator_i$ is assigned to in this restricted module network. An example of the resulting full PRM dependency structure for a simple module network with two modules and three regulatory genes is shown in Figure 4.10.

The mapping above highlights some of the key differences between module networks and PRMs. The first aspect has to do with the algorithm for learning the module network or its corresponding PRM. As we discussed earlier, a key feature of module networks is that the assignment of variables to modules is learned automatically as part of the algorithm for inducing a module network automatically from

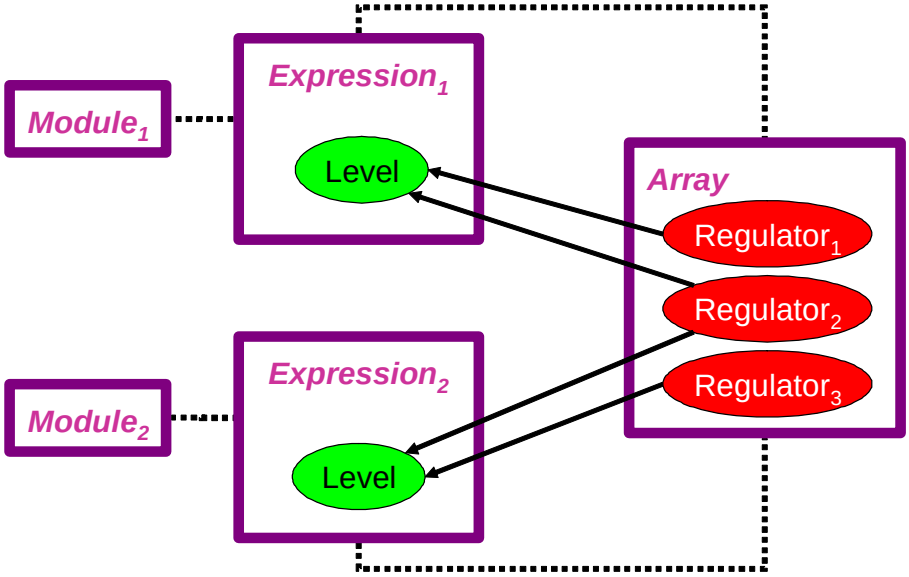


Figure 4.10: A PRM dependency structure for a simple module network with two modules and three regulatory genes. Note that each module has its own set of regulatory genes controlling its expression. In this example, the expression of module 1 is regulated by regulator 1 and regulator 2, while the expression of module 2 is regulated by regulator 2 and regulator 3.

data. In the PRM above, this means that we have uncertainty over the class type of each variable, which in the gene expression domain translates to our uncertainty over which of the K Module_j classes each gene as instance of. Thus, if we want the PRM framework to be compatible with that of module networks, we also need to devise a general algorithm for learning the types of objects automatically as part of the PRM learning process.

Another aspect of the mapping above is that the mapped module network was restricted to assigning each parent variable to a designated module containing only the parent variable without any parent variables of its own. Removing this restriction within the PRM framework implies that each regulator gene will also be an instance of one of the Module_j classes whose type needs to be learned. However, this presents us with another challenge: we must make sure that the ground Bayesian network induced from this PRM is acyclic. As a simple example, a regulator cannot be a parent of the module class that it itself is an instance of, as that would cause a

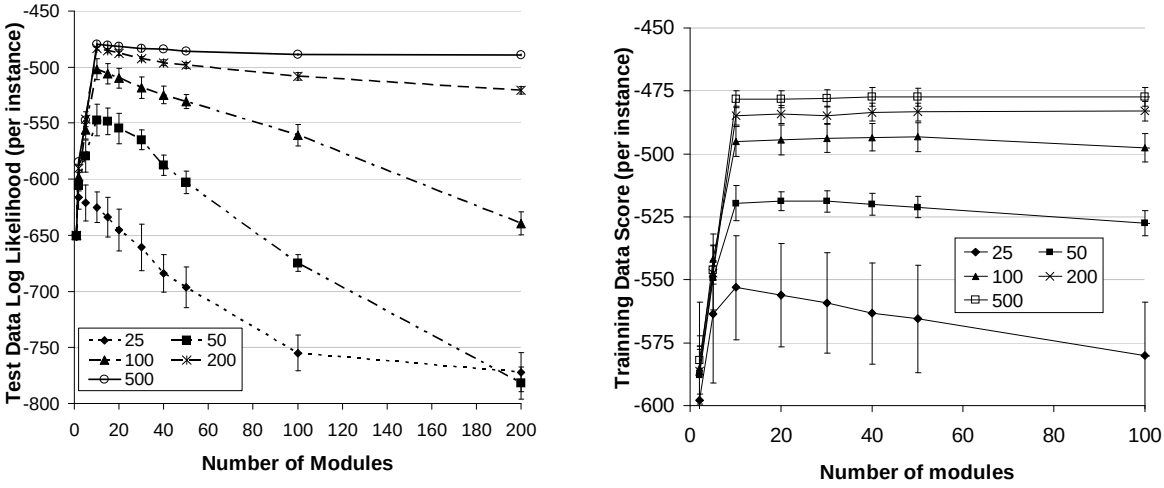
cyclic dependency. The ability to learn probabilistic dependencies for the regulatory genes is an important feature of the module network framework, as we can discover compound regulatory pathways, which are often of great interest. Thus, in order to allow the parent variables to be themselves assigned to modules, we must guarantee that our PRM learning algorithm results in acyclic models. Adding this feature to the learning algorithm is not trivial.

In summary, to achieve the full representation and functionality of module networks, the current PRM framework needs to be enhanced with two important features: a learning algorithm capable of automatically inferring the class type of objects; and the ability to learn PRMs that are guaranteed to produce acyclic ground Bayesian networks, while learning the class type of objects. Due to these missing features in PRMs, we introduced the new framework of module networks. We note that Getoor *et al.* (2000) attempt to address some of the issues above within the PRM framework using a class hierarchy. However, their approach is very different from ours, requiring some fairly complex search steps, and is not easily applied to the gene expression domain.

Even though the PRM framework is missing features that are key in module networks, its representation is rich in other aspects that may be useful to exploit within the module networks context. For example, in module networks, each gene is mapped to a single random variable. However, in the PRM framework, where we map each gene to a class, we can add additional descriptive attributes (e.g., the cellular-location of genes). These attributes can then participate in the PRM dependency structure and if necessary, we even have the flexibility of associating a different set of descriptive attributes for each module class. Thus, combining the module network framework with the PRM modeling language can result in a more expressive framework with useful applications to the gene regulation domain.

4.5 Statistical Evaluation on Synthetic Data

In all the experiments below, our data consists solely of continuous values. As all of the variables have the same domain, the definition of the module set reduces simply to



(a) (b)

Figure 4.11: Performance of learning from synthetic data as a function of the number of modules and training set size. The x-axis corresponds to the number of modules, each curve corresponds to a different number of training instances, and each point shows the mean and standard deviations from the 10 sampled data sets. (a) Log-likelihood per instance assigned to held-out data. (b) Average score per instance on the training data.

a specification of the total number of modules. We used regression trees as the local probability model for all modules. As our search algorithm, we used beam search, using a lookahead of three splits to evaluate each operator (see Section 2.7.1 for a description of these search algorithm variants). When learning Bayesian networks, as a comparison, we used precisely the same structure learning algorithm, simply treating each variable as its own module.

As a basic test of our procedure in a controlled setting, we used synthetic data generated by a known module network. This gives a known ground truth to which we can compare the learned models. To make the data realistic, we generated synthetic data from a model that was learned from the gene expression dataset described below. The generating model had 10 modules and a total of 35 variables that were a parent of some module. From the learned module network, we selected 500 variables, including the 35 parents. We tested our algorithm's ability to reconstruct the network using different numbers of modules; this procedure was run for training sets of various

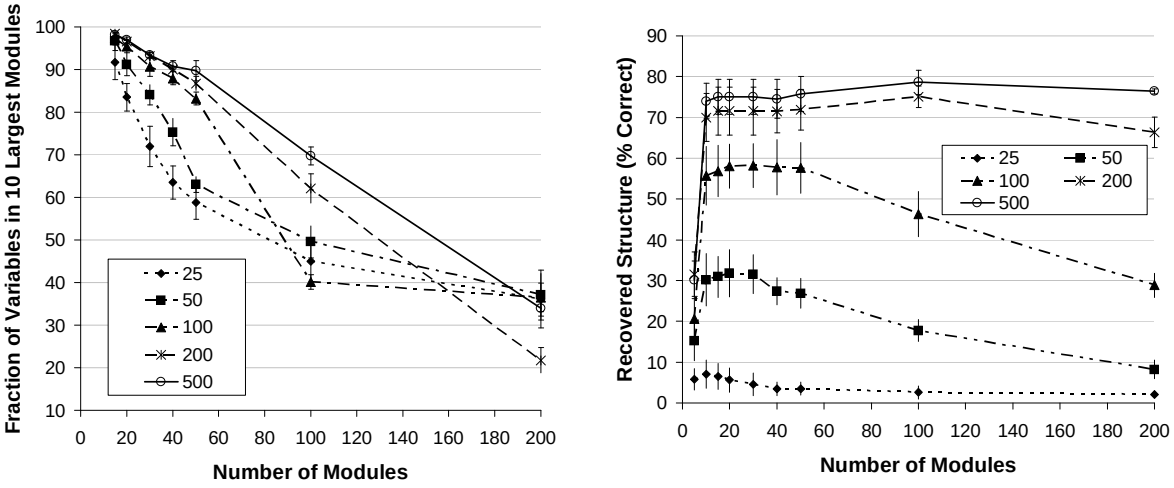
sizes ranging from 25 instances to 500 instances, each repeated 10 times for different training sets.

We first evaluated the generalization to unseen test data, measuring the likelihood ascribed by the learned model to 4500 unseen instances. The results, summarized in Figure 4.11(a), show that, for all training set sizes, except the smallest one with 25 instances, the model with 10 modules performs the best. As expected, models learned with larger training sets do better; but, when run using the correct number of 10 modules, the gain of increasing the number of data instances beyond 100 samples is small and beyond 200 samples is negligible.

To test whether we can use the score of the model to select the number of modules, we also plotted the score of the learned model on the training data (Figure 4.11(b)). As can be seen, when the number of instances is small (25 or 50), the model with 10 modules achieves the highest score and for a larger number of instances, the score does not improve when increasing the number of modules beyond 10. Thus, these results suggest that we can select the number of modules by choosing the model with the smallest number of modules from among the highest scoring models.

A closer examination of the learned models reveals that, in many cases, they are almost a 10-module network. As shown in Figure 4.12(a), models learned using 100, 200, or 500 instances and up to 50 modules assigned $\geq 80\%$ of the variables to 10 modules. Indeed, these models achieved high performance in Figure 4.11(a). However, models learned with a larger number of modules had a wider spread for the assignments of variables to modules and consequently achieved poor performance.

Finally, we evaluated the model's ability to recover the correct dependencies. The total number of parent-child relationships in the generating model was 2250. For each model learned, we report the fraction of correct parent-child relationships it contains. As shown in Figure 4.12(b), our procedure recovers 74% of the true relationships when learning from a dataset with 500 instances. Once again, we see that, as the variables begin fragmenting over a large number of modules, the learned structure contains many spurious relationships. Thus, our results suggest that, in domains with a modular structure, statistical noise is likely to prevent overly detailed learned models such as Bayesian networks from extracting the commonality between different



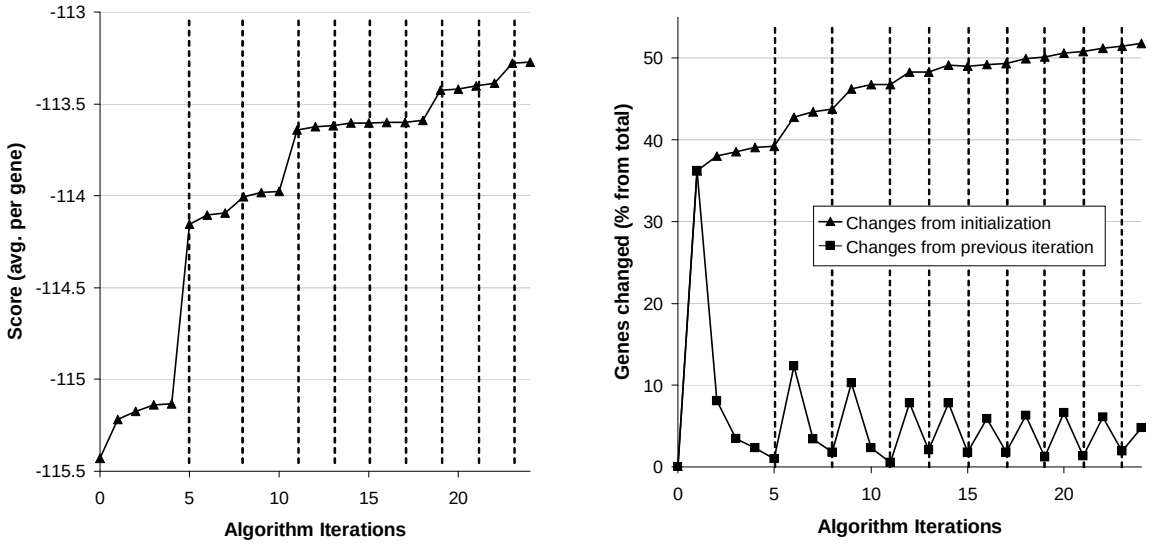
(a) (b)

Figure 4.12: (a) Fraction of variables assigned to the 10 largest modules. (b) Average percentage of correct parent-child relationships recovered when learning from synthetic data for models with various number of modules and different training set sizes. The x-axis corresponds to the number of modules, each curve corresponds to a different number of training instances, and each point shows the mean and standard deviations from the 10 sampled data sets.

variables with a shared behavior.

4.6 Evaluation on Real Data

We next evaluated the performance of our method on a real world data set of gene expression measurements. We used the expression data of Gasch *et al.* (Gasch *et al.*, 2000), which measured the response of yeast to different stress conditions. The data consists of 6157 genes and 173 experiments. As we have prior knowledge of which genes are likely to play a regulatory role (e.g., based on properties of their protein sequence), we restricted the possible parents to yeast genes that may play such a role. To this end, we compiled a set of 466 candidate regulators whose SGD (Cherry *et al.*, 1998) or YPD (Hodges *et al.*, 1999) annotations suggest a potential regulatory role in the broad sense: both transcription factors and signaling proteins that may have transcriptional impact. We also included genes described to be similar



(a) (b)

Figure 4.13: (a) Score of the model (normalized by the number of variables/genes) across the iterations of the algorithm for a module network learned with 50 modules on the gene expression data. Iterations in which the structure was changed are indicated by dashed vertical lines. (b) Changes in the assignment of genes to modules for the module network learned in (a) across the iterations of the algorithm. Shown are both the total changes compared to the initial assignment (triangles) and the changes compared to the previous iteration (squares).

to such regulators. We excluded global regulators, whose regulation is not specific to a small set of genes or processes. (See Segal *et al.* (2003c) for a complete listing of all candidate regulators). We then selected 2355 genes that varied significantly in the data and learned a module network over these genes. We also learned a Bayesian network over this data set.

4.6.1 Statistical Evaluation

We first examined the behavior of the learning algorithm on the training data when learning a module network with 50 modules. This network converged after 23 iterations. To characterize the trajectory of the algorithm, we plot in Figure 4.13 its

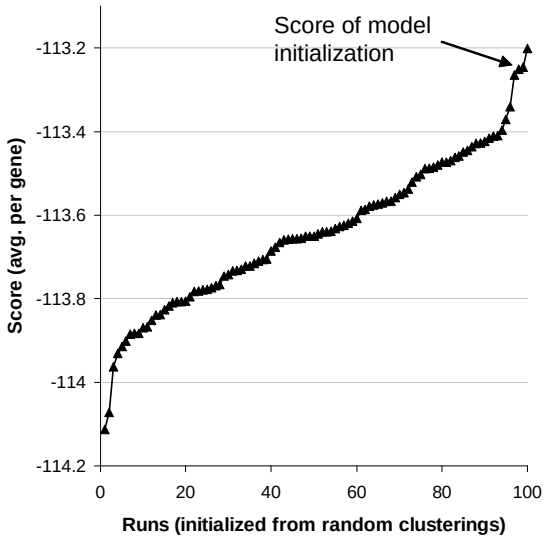


Figure 4.14: Score of 100 module networks (normalized by the number of variables/genes) each learned with 50 modules from a random clustering initialization, where the runs are sorted according to their score. The score of a module network learned using the deterministic clustering initialization described in Section 4.3.4 is indicated by an arrow.

improvement across the iterations, measured as the score on the training data, normalized by the number of genes (variables). To obtain a finer-grained picture, we explicitly show structure learning steps, as well as each pass over the variables in the module reassignment step. As can be seen in Figure 4.13(a), the model score improves nicely across these steps, with the largest gains in score occurring in steps in which the structure was changed. Figure 4.13(b) demonstrates how the algorithm changes the assignments of genes to modules, with 1221 of the 2355 (51.8%) genes changing their initial assignment upon convergence, and the largest assignment changes occurring immediately after structure modification steps.

As for most local search algorithms, initialization is a key component: A bad initialization can cause the algorithm to get trapped in a poor local maximum. As we discussed in Section 4.3.4, we initialize the assignment function using a clustering program. The advantage of a simple deterministic initialization procedure is that it is computationally efficient, and results in reproducible behavior. We evaluated this

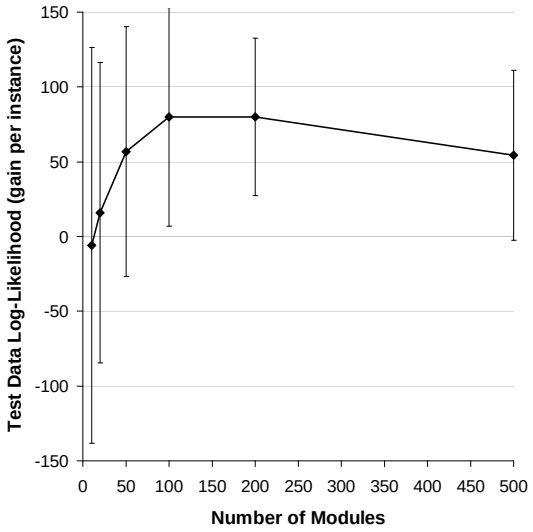


Figure 4.15: Comparison of generalization ability of module networks learning with different numbers of modules on the gene expression data set. The x-axis denotes the number of modules. The y-axis denotes the difference in log-likelihood on held out data between the learned module network and the learned Bayesian network, averaged over 10 folds; the error bars show the standard deviation.

proposed initialization by comparing the results to module networks initialized randomly. We generated 100 random assignments of variables to modules, and learned a module network starting from each initialization. We compared the model score of the network learned using our deterministic initialization, and the 100 networks initialized randomly. A plot of these sorted scores is shown in Figure 4.14. Encouragingly, the score for the network initialized using our procedure was better than 97/100 of the runs initialized from random clusters, and the 3/100 runs that did better are only incrementally better.

Moreover, we found a high correspondence between the assignment of genes to modules in the network initialized using our procedure and the assignment of genes to modules in the 100 randomly initialized networks: between 45 and 50 of the modules in the analyzed run could clearly be matched to a module in each of the 100 runs initialized randomly. For this mapping between modules, at least 50% of the genes were placed in the same module in 56 of the 100 runs.

We evaluated the generalization ability of different models, in terms of log-likelihood

of test data, using 10-fold cross validation. In Figure 4.15, we show the difference between module networks of different size and the baseline Bayesian network, demonstrating that module networks generalize much better to unseen data for almost all choices of number of modules.

4.6.2 Sample Modules

We now turn to a detailed biological evaluation of the results when learning a module network with 50 modules. With few exceptions, each of the inferred modules (46/50) contained a functionally coherent set of genes (see Appendix C for a description of how a module is tested for functional coherence). Together the modules spanned a wide variety of biological processes including metabolic pathways (e.g., glycolysis), various stress responses (e.g., oxidative stress), cell-cycle related processes, molecular functions (e.g., protein folding), and cellular compartments (e.g., nucleus). (See Segal *et al.* (2003c) for a complete listing of modules). Most modules (30/50) included genes previously known to be regulated by the module's predicted regulators. Many modules (15/50) had a match between a predicted regulator and its known *cis*-regulatory binding motif (i.e., a statistically significant number of the module's genes contained the known motif in their upstream regions). Overall, our results provide a global view of the yeast transcriptional network, including many instances in which our method discovered known functional modules and their correct regulators, demonstrating the ability of our method to derive regulation from expression.

We now present in detail several of the inferred modules, selected to show the method's ability to recover diverse features of regulatory programs.

The Respiration Module (see Figure 4.16) provides a clear demonstration of a predicted module and of the validation process. It consists primarily of genes encoding respiration proteins (39/55) and glucose metabolism regulators (6/55). The inferred regulatory program specifies the Hap4 transcription factor as the module's top (activating) regulator, primarily under stationary phase (a growth phase in which nutrients, primarily glucose, are depleted). This prediction is consistent with Hap4's

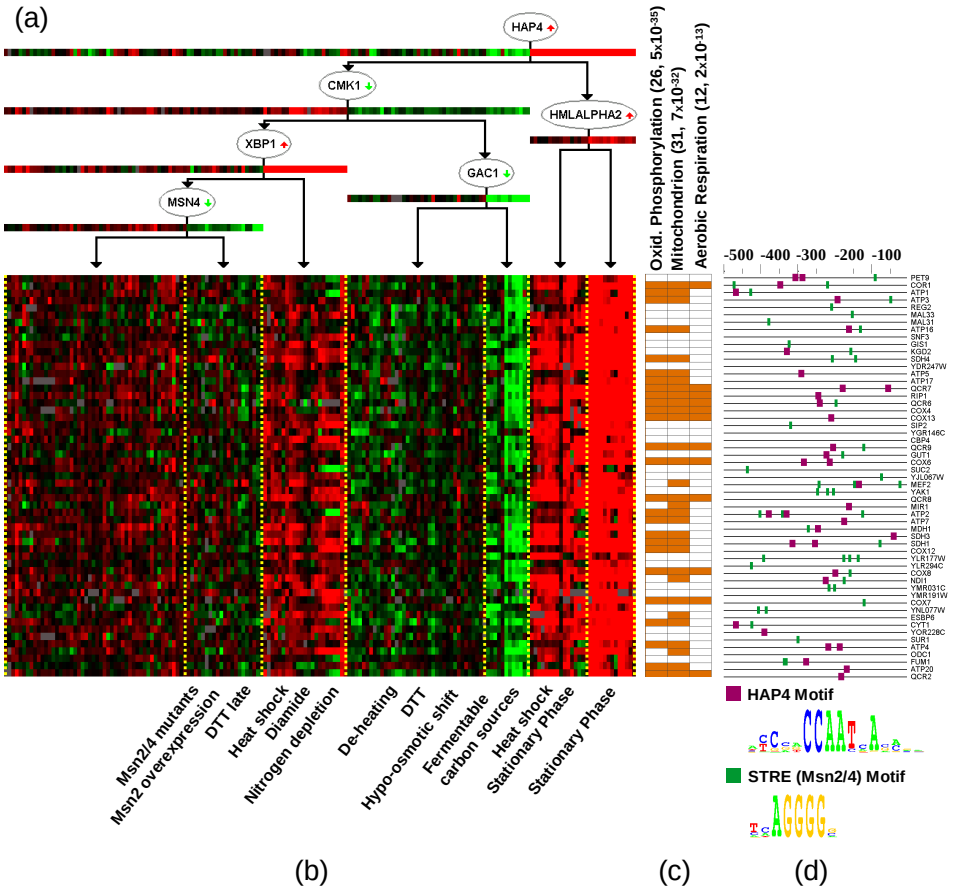


Figure 4.16: The Respiration and carbon regulation module (55 genes). (a) Regulation program (see Figure 4.2). (b) Gene expression profiles. Genes (rows), arrays (columns). Arrays are arranged according to the regulation tree. For example, the rightmost leaf includes the arrays in which both Hap4 and Alpha2 are up-regulated. Contexts that consist primarily of one or two types of experimental conditions are labeled. (c) Significant annotations: Colored entries indicate genes with the respective annotation. The most significantly enriched annotations for this module were selected for display (the number of annotated genes and the calculated p-value for the enrichment of each annotation are shown in parentheses). Note the enrichment of three annotations representing a biochemical process, cellular compartment, and physiological process, respectively, all relating to cellular respiration. (d) Promoter analysis. Lines represent 500bp of genomic sequence located upstream to the start codon of each of the genes; colored boxes represent the presence of *cis*-regulatory motifs located in these regions.

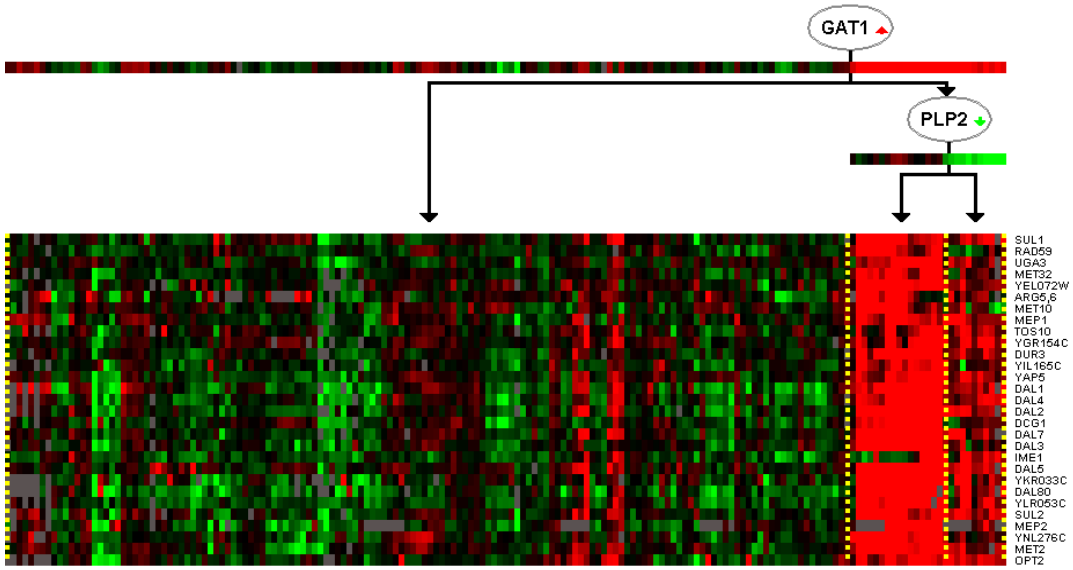


Figure 4.17: The nitrogen catabolite repression module

known role in activation of respiration (DeRisi *et al.*, 1997, Forsburg and Guarente, 1989). Indeed, our post-analysis detected a Hap4-binding DNA sequence motif (bound by the Hap2/3/4/5 complex) in the upstream region of 29/55 genes in the module ($p < 2 \cdot 10^{-13}$). Note that this motif also appears in non-respiration genes (mitochondrial genes and glucose metabolism regulators), which together with their matching expression profiles, supports their inclusion as part of the module. When Hap4 is not induced, the module is activated more mildly or is repressed. The method suggests that these changes are regulated by other regulators, such as the protein phosphatase type 1 regulatory subunit Gac1 and the transcription factor Msn4. Indeed, the stress response element (STRE), recognized by Msn4, appears in the upstream region of 32/55 genes in the module ($p < 10^{-3}$) as well as in those of many of the genes containing the Hap4 motif (17/29 genes; $p < 7 \cdot 10^{-10}$), supporting our placement of both regulators in one control program.

The Nitrogen Catabolite Repression Module (see Figure 4.17) demonstrates the ability of our method to capture an entire cellular response whose genes participate in diverse metabolic pathways and cellular roles (12/29 in allantoin and urea metabolism,

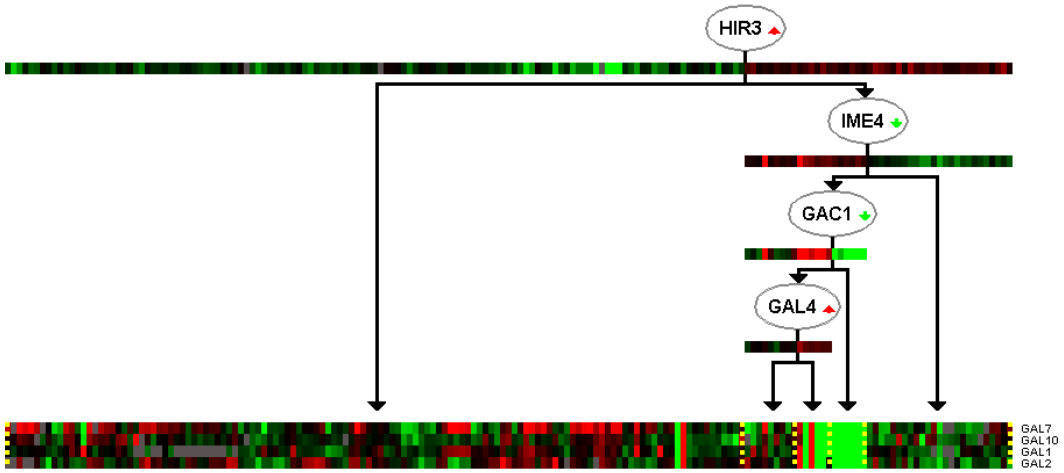


Figure 4.18: The galactose metabolism module

5/29 in amino acid metabolism, and 6/29 in sulfur/methionine metabolism), all of which relate to the process by which the yeast utilizes the best available nitrogen source. Gat1 is suggested as the key (activating) regulator of this module, further supported by the presence of the GATA motif, the known binding sequence for Gat1, in the upstream region of 26/29 genes ($p < 10^{-17}$). This module also demonstrates that the method can discover context-specific regulation, as the similarity in expression of genes in the module is mostly pronounced in stationary phase (17/22 experiments; $p < 10^{-4}$), amino acid starvation (5/5; $p < 9 \cdot 10^{-5}$), and nitrogen depletion (10/10; $p < 8 \cdot 10^{-9}$), all of which are conditions where utilizing alternative nitrogen sources is crucial. Note that two additional known regulators involved in this response, Uga3 and Dal80, are incorrectly suggested as members, rather than regulators, of the module.

The Galactose Metabolism Module (see Figure 4.18) illustrates our method's ability to recover small expression signatures, as the module consisted of only four Gal4-regulated genes and predicted Gal4 as a regulator, with a predicted regulatory role that includes activation in galactose-containing medium.

The Energy, Osmolarity and cAMP Signaling Module (see Figure 4.19) demonstrates that our method can discover regulation by proteins other than transcription

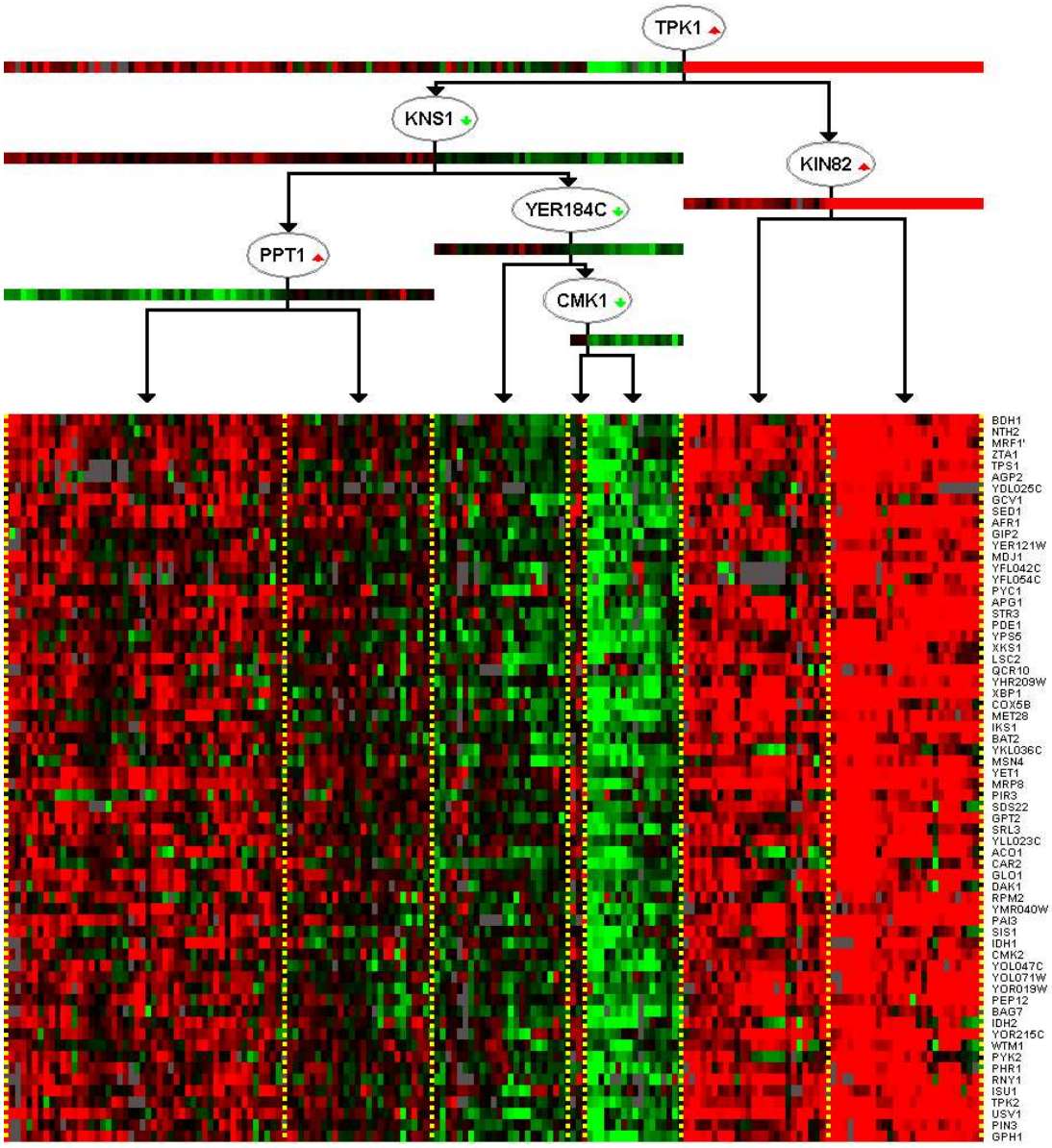


Figure 4.19: The energy, osmolarity and cAMP signaling module

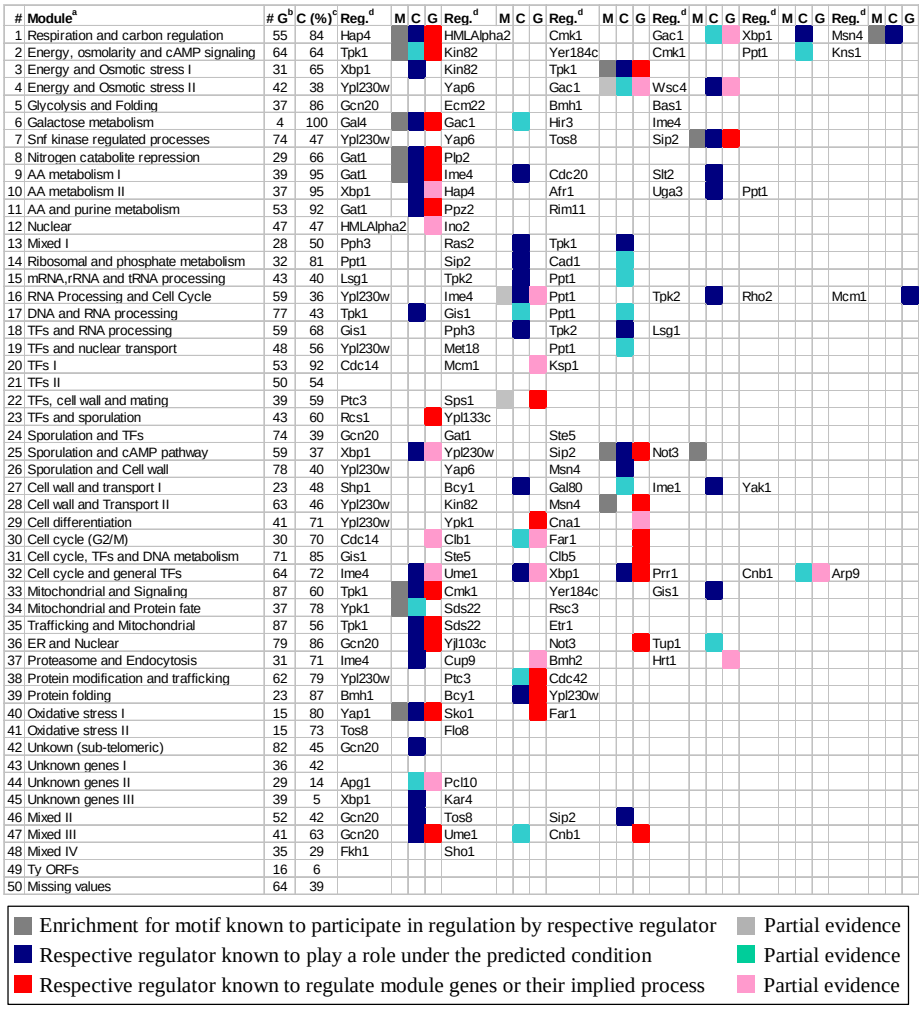
factors, as the top predicted regulator was Tpk1, a catalytic subunit of the cAMP dependent protein kinase (PKA). This prediction is supported by a recent study (Norbeck and Blomberg, 2000) showing that the expression of several genes in the module (e.g., Tps1) is strongly affected by Tpk1 activity in osmotic stress, which was among the conditions predicted by the method to be Tpk1-regulated. (A regulator

is predicted to regulate a module under a set of conditions C if the contexts split by the regulator contain a large number of arrays from condition C). Further support is given by the presence of the STRE motif, known to be bound by transcription factors that are regulated by Tpk1 (Norbeck and Blomberg, 2000), in the upstream region of most genes in the module (50/64; $p < 3 \cdot 10^{-11}$), often in combination with other motifs bound by Tpk1-modulated transcription factors, such as Adr1 (37/64; $p < 6 \cdot 10^{-3}$) and Cat8 (26/64; $p < 2 \cdot 10^{-3}$). However, our method suggests that Tpk1 is an activator of the module in contrast to its known role as a repressor (Lenssen *et al.*, 2002). We discuss this discrepancy below.

4.6.3 Global View

We next evaluated all 50 modules to test whether the proteins encoded by genes in the same module had related functions. To this end, we associated each gene with the processes it participates in. We removed all annotations associated with less than 5 genes from our gene set. This resulted in a list of: 923 GO (Ashburner *et al.*, 2000) categories, 208 MIPS (Mewes *et al.*, 1997) categories, and 87 KEGG (Kanehisa *et al.*, 2002) pathways. We scored the functional/biological coherence of each module (Figure 4.20, C column) by the percent of its genes covered by annotations significantly enriched in the module ($p < 0.01$). We assigned a name to each module based on the largest one or two categories of genes in the module (combining gene annotations from the databases above and from the literature). Note that these concise names are introduced to facilitate the presentation, and do not always convey the full content of some of the more heterogeneous modules (see modules and their significant annotations in Figure 4.21). Most modules (31/50) exhibited a coherence level above 50% and only 4/50 had gene coherence below 30%. Note that the actual coherence levels may be considerably higher, as many genes are not annotated in current databases. Indeed, an in-depth inspection revealed many cases where genes known to be associated with the main process of the module were simply not annotated as such.

We obtained a global view of the modules and their function by compiling all gene



#	Module[a]	# G[b]	C (%)[c]	Reg.[d]	M	C	G	Reg.[d]	M	C	G	Reg.[d]	M	C	G	Reg.[d]	M	C	G	Reg.[d]	M	C	G	Reg.[d]	M	C	G
1	Respiration and carbon regulation	55	84	Hap4				HMLAlpha2				Cmk1				Gac1				Xbp1				Msn4			
2	Energy, osmolarity and cAMP signaling	64	64	Tpk1				Kin82				Yer184c				Cmk1				Ppt1				Kns1			
3	Energy and Osmotic stress I	31	65	Xbp1				Kin82				Tpk1															
4	Energy and Osmotic stress II	42	38	Ypl230w				Yap6				Gac1				Wsc4											
5	Glycolysis and Folding	37	86	Gcn20				Ecm22				Bmh1				Bas1											
6	Galactose metabolism	4	100	Gal4				Gac1				Hir3				Ime4											
7	Snf kinase regulated processes	74	47	Ypl230w				Yap6				Tos8				Sip2											
8	Nitrogen catabolite repression	29	66	Gat1				Plp2																			
9	AA metabolism I	39	95	Gat1				Ime4				Cdc20				Slt2											
10	AA metabolism II	37	95	Xbp1				Hap4				Afr1				Uga3				Ppt1							
11	AA and purine metabolism	53	92	Gat1				Ppz2				Rim11															
12	Nuclear	47	47	HMLAlpha2				Ino2																			
13	Mixed I	28	50	Pph3				Ras2				Tpk1															
14	Ribosomal and phosphate metabolism	32	81	Ppt1				Sip2				Cad1															
15	mRNA,rRNA and tRNA processing	43	40	Lsg1				Tpk2				Ppt1															
16	RNA Processing and Cell Cycle	59	36	Ypl230w				Ime4				Ppt1				Tpk2				Rho2				Mcm1			
17	DNA and RNA processing	77	43	Tpk1				Gis1				Ppt1															
18	TFs and RNA processing	59	68	Gis1				Pph3				Tpk2				Lsg1											
19	TFs and nuclear transport	48	56	Ypl230w				Met18				Ppt1															
20	TFs I	53	92	Cdc14				Mcm1				Ksp1															
21	TFs II	50	54																								
22	TFs, cell wall and mating	39	59	Ptc3				Sps1																			
23	TFs and sporulation	43	60	Rcs1				Ypl133c																			
24	Sporulation and TFs	74	39	Gcn20				Gat1				Ste5															
25	Sporulation and cAMP pathway	59	37	Xbp1				Ypl230w				Sip2				Not3											
26	Sporulation and Cell wall	78	40	Ypl230w				Yap6				Msn4															
27	Cell wall and transport I	23	48	Shp1				Bcy1				Gal80				Ime1				Yak1							
28	Cell wall and Transport II	63	46	Ypl230w				Kin82				Msn4															
29	Cell differentiation	41	71	Ypl230w				Ypk1				Cna1															
30	Cell cycle (G2/M)	30	70	Cdc14				Clb1				Far1															
31	Cell cycle, TFs and DNA metabolism	71	85	Gis1				Ste5				Clb5															
32	Cell cycle and general TFs	64	72	Ime4				Ume1				Xbp1				Prr1				Cnb1				Arp9			
33	Mitochondrial and Signaling	87	60	Tpk1				Cmk1				Yer184c				Gis1											
34	Mitochondrial and Protein fate	37	78	Ypk1				Sds22				Rsc3															
35	Trafficking and Mitochondrial	87	56	Tpk1				Sds22				Etr1															
36	ER and Nuclear	79	86	Gcn20				Yjl103c				Not3				Tup1											
37	Proteasome and Endocytosis	31	71	Ime4				Cup9				Bmh2				Hrt1											
38	Protein modification and trafficking	62	79	Ypl230w				Ptc3				Cdc42															
39	Protein folding	23	87	Bmh1				Bcy1				Ypl230w															
40	Oxidative stress I	15	80	Yap1				Sko1				Far1															
41	Oxidative stress II	15	73	Tos8				Flo8																			
42	Unkown (sub-telomeric)	82	45	Gcn20																							
43	Unknown genes I	36	42																								
44	Unknown genes II	29	14	Apg1				Pcl10																			
45	Unknown genes III	39	5	Xbp1				Kar4																			
46	Mixed II	52	42	Gcn20				Tos8				Sip2															
47	Mixed III	41	63	Gcn20				Ume1				Cnb1															
48	Mixed IV	35	29	Fkh1				Sho1																			
49	Ty ORFs	16	6																								
50	Missing values	64	39																								

Figure 4.20: Summary of module analysis and validation. [a] Module name. [b] Number of genes in module. [c] Functional/biological coherence of each module, measured as the percent of genes in the module covered by significant gene annotations ($p < 0.01$). [d] Regulators predicted to regulate each module are listed, along with three scores for each regulator. Filled boxes indicate biological experiments supporting the prediction; washed-out colored boxes indicate indirect or partial evidence. M, enrichment for a motif known to participate in regulation by the respective regulator, in upstream regions of genes in the module; C, experimental evidence for contribution of the respective regulator to the transcriptional response under the predicted conditions; G, direct experimental evidence showing that at least one of the genes in the module, or a process significantly over-represented in the module genes, is regulated by the respective regulator.

annotations and motifs significantly enriched in each module into a single matrix (see Figure 4.21(a), left matrices). For deriving significant gene annotations, we calculated, for each module and for each annotation, the fraction of genes in the module associated with that annotation and used the hypergeometric distribution to calculate a p-value for this fraction (see Appendix C). We performed a Bonferroni correction for multiple independent hypotheses and took p-values $< 0.05/N$ (N=923, 208, 87 for GO, MIPS, and KEGG annotations, respectively) to be significant for Figure 4.20 and Figure 4.21. For deriving significant motifs, we first obtained a collection of both known motifs (Heinemeyer *et al.*, 1999) and de novo motifs (discovered by the motif-finding program described in Section 3.2.1 applied to the upstream regions of the genes in each module). For each module and each motif, we then calculated the fraction of genes in the module associated with that motif and used the hypergeometric distribution to calculate a p-value for this fraction (Bonferroni corrected for multiple hypotheses testing) as described above for deriving significant gene annotations.

This presentation enables an automatic approach for deriving rich descriptions for modules. For example, the attributes for the respiration module shown in Figure 4.16 are immediately apparent in this representation, including the Hap4 and Msn4 (STRE) binding sites, and the ion transport, TCA cycle, aerobic respiration, and mitochondrion annotated genes (see Figure 4.21(a), highlighted rectangles labeled '1'). The matrix representation also provides further support for the inferred modules. For example, it justifies the division of amino acid (AA) metabolic processes into four modules (see Figure 4.21(b)): while the modules share certain attributes (e.g., AA metabolism), each is characterized by a unique combination of gene annotations (e.g., only module 9 is also annotated as starvation response). Furthermore, all of the modules in this module group are associated with a common cis-regulatory motif (Gcn4), but each has a unique signature of cis-regulatory motifs.

To obtain a global perspective on the relationships between different modules, and the extent to which they group together, we compiled a graph of modules and cis-regulatory motifs, and connected modules to their significantly enriched motifs (see Figure 4.22). In this view, sets of similar but distinct modules, such as AA metabolism modules (8-11), energy modules (1-3, 25, 33, 41), and DNA/RNA modules (13-15,

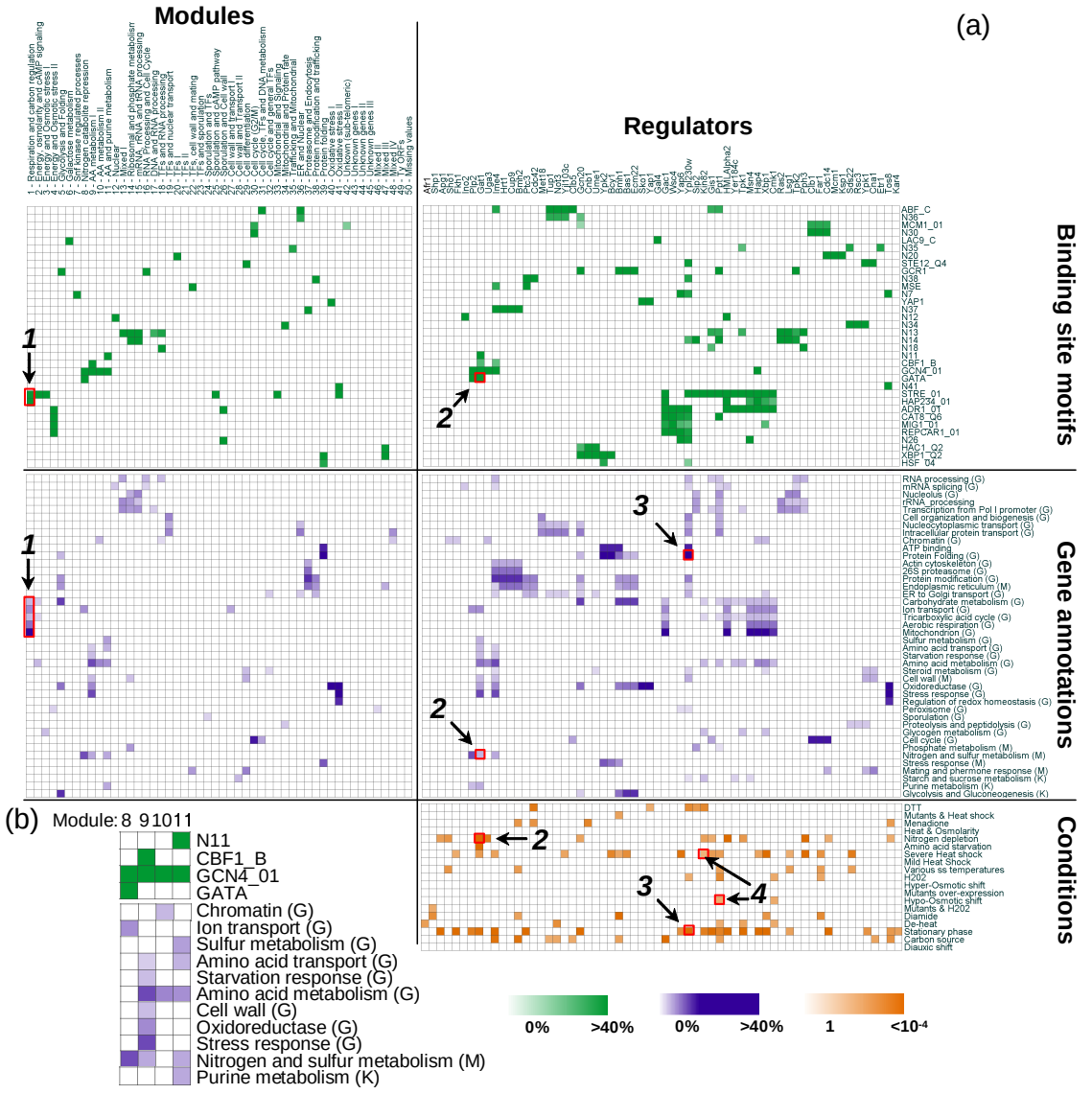


Figure 4.21: Enrichment of annotations and motifs in modules and in predicted targets of regulators. (a) Entries represent the percentage of genes associated with annotations in each module or in all predicted targets of each regulator. De novo motifs are named with 'N' and the module number from which the motif was discovered). Gene annotations were compiled from Gene Ontology (G), MIPS (M), and KEGG (K). The bottom right matrix displays significant conditions regulated by each regulator and the entries correspond to p-values for each condition. (b) Submatrix of significant annotations for amino acid metabolism related modules (8-11).

17, 18) form module groups, where all modules share at least one motif. It can be seen that different modules in a group are again characterized by partly overlapping but distinct combinations of motifs. We also searched for pairs of motifs that are significantly enriched (as a pair) in the upstream regions of module genes. While different modules were characterized by distinct motif pairs, there is overlap between the motif pairs of modules within a module group, providing further support for the combinatorial nature of the inferred regulation programs. When we examine the predicted regulators of modules (see Figure 4.22), we see that modules belonging to the same module group appear to share some, but not all, of their regulators. These results suggest a higher level of modularity of the yeast transcriptional network, in which functionally related modules share some of their regulatory elements, yet each module is characterized by a unique regulatory program.

We next turned to the evaluation of the inferred regulation programs. We compared the known function of the inferred regulators with the method's predictions, where known function is based on a compiled list of literature references (see (Segal *et al.*, 2003c) for a list of all references used), in which direct experimental evidence exists for the role of the predicted regulators. We note that some modules (21, 43, 49, 50) did not have regulators, as none of the candidate regulators was predictive of the expression profile of their gene members. In most modules (35/50), the regulators were predicted to play a role under the expected conditions (Figure 4.20, C columns). For most modules (30/50) there is direct experimental evidence showing that at least one of the genes in the module, or a process significantly over-represented in the module genes, is regulated by at least one of the module's predicted regulators (Figure 4.20, G columns). Many modules (15/50) also had an exact match between cis-regulatory motifs enriched ($p < 10^{-4}$) in upstream regions of the module's genes and the regulator known in the biological literature to bind to that motif (Figure 4.20, M columns).

To identify the function of the regulators, we associated each regulator with biological processes, experimental conditions, and possibly a binding motif. As a regulator 'X' may regulate more than one module, its targets consist of the union of the genes in all modules predicted to be regulated by 'X'. We tested the targets of each regulator

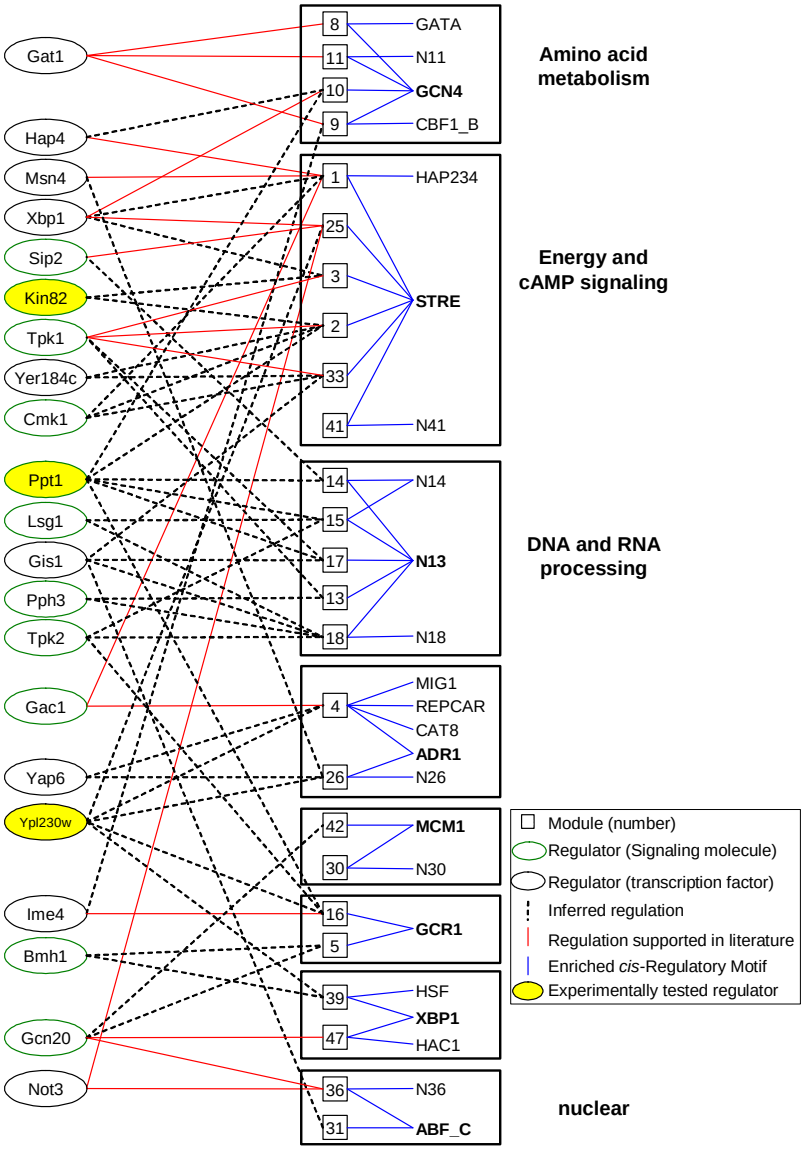


Figure 4.22: Global view and higher order organization of modules. The graph depicts inferred modules (middle, numbered squares), their significantly enriched motifs (right), and their associated regulators (left). Red edges between a regulator and a module are supported in the literature. Module groups are defined as sets of modules that share a single significant cis-regulatory motif. Module groups whose modules are functionally related are labeled (right). Modules belonging to the same module group appear to share regulators and motifs, with individual modules having different combinations of these regulatory elements.

for enrichment of the same motifs and gene annotations as above (see Figure 4.21(a), two upper right matrices), using the hypergeometric p-value. We took p-values < 0.05 (Bonferroni corrected as for module annotations) to be significant.

In addition, we tested each regulator for experimental conditions that it significantly regulates by examining how conditions are split by each relevant regulation tree. Experimental conditions were extracted from the array labels (Gasch *et al.*, 2000). For each occurrence of a regulator as a decision node in a regression tree, we computed the partition of each experimental condition between the right branch (the true answer to the query on the regulator) and the left branch (the false answer), and used the binomial distribution to compute a p-value on this partition (see Appendix D). We took p-values < 0.05 to be significant. For example, in the respiration module (see Figure 4.16), Hap4 up-regulation distinguishes stationary phase conditions from the rest (right branch; Hap4 activates the module), and would thus be associated with regulation in stationary phase. Significant conditions for a particular regulator can thus be discovered either by visual inspection of the regulation tree or by the automated statistical procedure described above. The results of this procedure are summarized in Figure 4.21(a) (bottom right matrix). As an example of the resulting associations, the matrix suggests that Gat1 regulates nitrogen and sulfur metabolism processes, binds to the GATA motif, and works under conditions of nitrogen depletion (Figure 4.21(a), highlighted rectangles labeled '2').

When we consider uncharacterized regulators, the predicted regulator annotations provide focused hypotheses about the processes they regulate, the conditions under which they work, and the cis-regulatory motifs through which their regulation is mediated. For example, we can predict that the putative transcription factor Ypl230w regulates genes important for protein folding during stationary phase (Figure 4.21(a), highlighted rectangles labeled '3'). The ability to generate detailed hypotheses, in the form "regulator 'X' regulates process 'Y' under conditions 'W'", is among the most powerful features of the module networks procedure, as it also suggests the specific experiments that can validate these hypotheses.

4.7 Wet Lab Experiments

A regulation program specifies that certain genes regulate certain processes under certain conditions. Our method thus generates detailed, testable hypotheses, suggesting specific roles for a regulator and the conditions under which it acts. We tested experimentally the computational predictions for three putative regulators with as yet unknown functions (a transcription factor and two signaling molecules). To test our ability to predict different types of regulatory mechanisms, we selected a putative zinc-finger transcription factor, Ypl230w, and two putative signaling molecules, the protein kinase Kin82 and the phosphatase Ppt1. We obtained the relevant yeast deletion strains (Winzeler *et al.*, 1999). Under normal growth conditions, all three deletion strains showed no apparent abnormalities as compared to the wild-type strain.

As discussed above, each hypothesis generated by the method provides the significant conditions under which the regulator is active, and thereby specifies the experimental conditions under which the mutant should be tested. In concordance with the method's hypotheses (see Figure 4.21(a), highlighted rectangles labeled '3' and '4' in the bottom right regulator-condition matrix), we tested $\triangle$Kin82 under severe heat shock conditions (25°C to 37°C), $\triangle$Ppt1 during hypo-osmotic shift, and $\triangle$Ypl230w during the entry to stationary phase.

In each experiment, we employed microarray analysis to compare the transcriptional response in the deletion strain to that of the wild-type strain, under the same conditions. These genome-wide experiments enable a complete evaluation of the accuracy of our predictions for each regulator: whether it plays a regulatory role in the predicted conditions; whether it regulates genes in modules that it was predicted to regulate; and most importantly, whether it regulates processes that the method predicted it regulates.

To test whether the mutants had any effect, we used a paired two-tailed t-test to identify the genes that showed differential expression between wild-type and mutant strains under the tested conditions. Genes with $p < 0.05$ from the t-test were considered differentially expressed. Each time series was zero-transformed to enable comparison of the response to the tested condition. For $\triangle$Kin82 and $\triangle$Ppt1, we gave

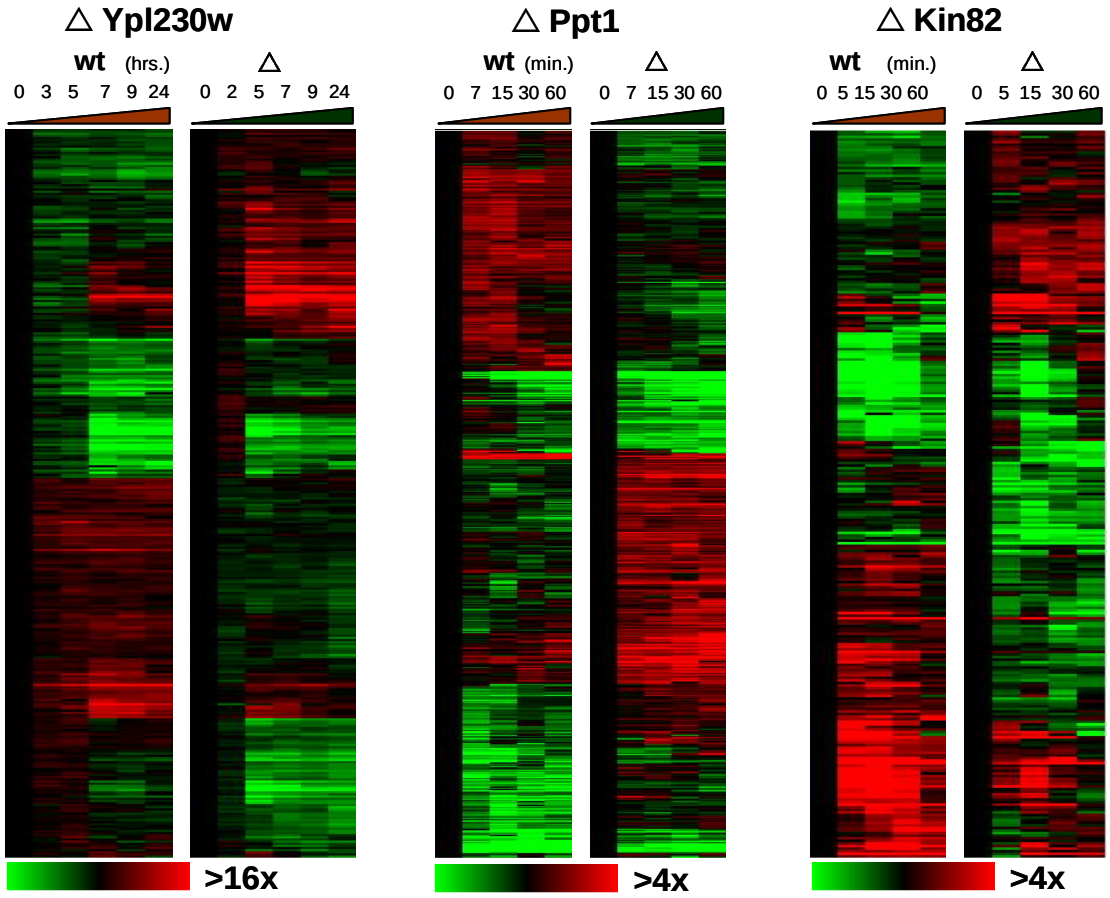


Figure 4.23: Microarray experiments testing functional predictions for putative regulators. Expression data for the differentially expressed genes (extracted using paired t-test) for both the wild-type (wt) and mutant time series in the following experiments: △Ypl230w during stationary phase, △Ppt1 during hypo-osmotic shift, and △Kin82 under heat shock.

all time points as input to the t-test (5, 15, 30, and 60 minutes for △Kin82; 7, 15, 30, and 60 minutes for △Ppt1). For the △Ypl230w experiment, measuring response during stationary phase, we used only the late time points (7, 9, and 24 hours), since the response to this growth condition starts at 7 hours. To ensure that only genes with large differences are included, we also required that at least half the time points compared are different by at least 2-fold change in expression. The only exception was △Ppt1, where we required a 1.3 fold difference, as the overall signal in these

arrays was weaker.

The number of such genes was much higher than expected by chance (1034 for △Kin82, 1334 for △Ppt1, and 1014 for △Ypl230w), thus showing that all three regulators play a role in the predicted conditions, in contrast to normal growth conditions where there were few differences between the wild-type and mutant strains. To focus on the most significant changes, we examined only genes with a significant fold change in expression between the wild-type and mutant expression profiles, resulting in 281 genes for △Kin82, 602 genes for △Ppt1, and 341 genes for △Ypl230w (see Figure 4.23 for expression patterns).

To test whether our method correctly predicted the targets of each regulator, we examined the distribution of the differentially expressed genes among the modules. For each putative regulator 'X', we calculated a p-value for the enrichment of differentially expressed genes in each module, and ranked the modules according to these p-values. In all three cases, the highest ranking module was predicted to be regulated by 'X' (Figure 4.24(a)), with: 25% (△Ppt1, $p < 9 \cdot 10^{-3}$), 26% (△Kin82, $p < 10^{-4}$), and 30% (△Ypl230w, $p < 10^{-4}$) of the genes in the highest ranking module showing differential expression.

Finally, we tried to identify the process regulated by each regulator, by searching for significantly enriched functional annotations in its set of differentially expressed genes. In two cases (△Ypl230w, and △Ppt1), the annotations matched those predicted for the regulator (Figure 4.24(b)), supporting the method's suggestions for the regulatory roles of the tested regulators: Ypl230w activates protein folding, cell wall, and ATP binding genes, and Ppt1 represses phosphate metabolism and rRNA processing.

Altogether, deletion of each of the three regulators caused a marked impairment in the expression of a significant fraction of their computationally predicted targets, supporting the method's predictions and providing important insight regarding the function of these uncharacterized regulators.

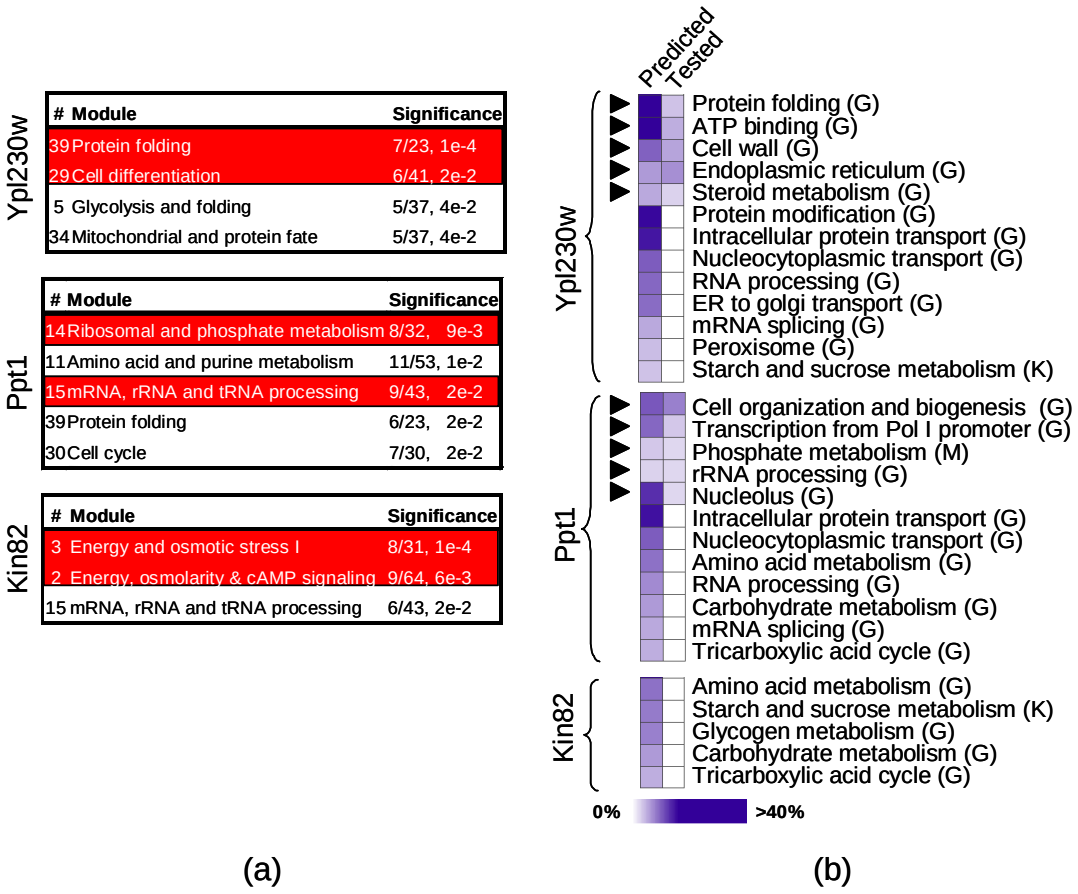
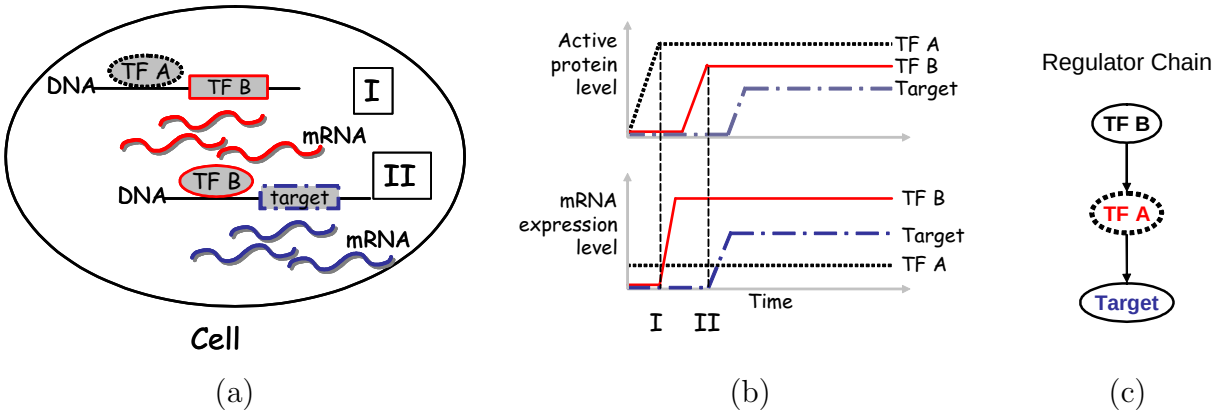


Figure 4.24: (a) Ranked modules table for each tested regulator 'X'; ranking is based on p-value calculated for enrichment of differentially expressed genes in each module. All modules significantly enriched for these genes ($p < 0.05$) are shown along with the number of differentially expressed genes out of the total number of genes in the module and the corresponding p-value for the enrichment. Modules predicted to be regulated by the respective regulator 'X' are highlighted in red. (b) Functional predictions for tested regulators. The left column (Predicted) for each regulator shows all annotations predicted by the method to be associated with that regulator (extracted from the corresponding column in Figure 4.21(a)). The right column (Tested) shows which annotations were also significantly enriched in the set of differentially expressed genes of each regulator ($p < 0.05$; black triangles), where the intensity of each entry represents the fraction of genes with the annotation from the set of differentially expressed genes.

4.8 From Gene Expression to Regulation

In summary of our biological evaluation, we performed a comprehensive evaluation of the validity of the structures reconstructed by our method. By analyzing biological databases and previous experimental results in the literature, we confirmed that many of the regulatory relations that our method automatically inferred are indeed correct. Furthermore, our model provided focused predictions for genes of previously uncharacterized function. We performed wet lab biological experiments that confirmed the three novel predictions we tested. Thus, we have demonstrated that the module network model is robust enough to learn a good approximation of the dependency structure between 2355 genes using only 173 instances. These results show that, by learning a structured probabilistic representation, we identify regulation networks from gene expression data and successfully address one of the central problems in analysis of gene expression data. This finding is far from trivial and it is thus interesting to try and gain insight into the biological mechanism that makes it possible to infer regulatory events from gene expression data. In this section we explore this question.

Our method detects regulatory events based on the statistical association in gene expression. In order to identify a regulatory relation in expression data, both the regulator and its targets must be transcriptionally regulated, resulting in detectable changes in their expression. Therefore, our approach relies on the assumption that the gene expression profile of the regulators provides evidence as to their activity level. This assumption is currently part of an ongoing biological debate: some researchers do not believe that gene expression data can reveal the actual regulators themselves. Indeed, due to the complex, often post-transcriptional nature of regulation, our assumption does not always hold. However, to the contrary, recent large-scale analysis of the regulatory networks of *E. Coli* (Shen-Orr *et al.*, 2002) and *S. Cerevisiae* (Lee *et al.*, 2002, Milo *et al.*, 2002) revealed the prevalence of cases in which the regulators are themselves transcriptionally regulated, a process whose functional importance is supported both theoretically and experimentally (Hlavacek and Savageau, 1996, Rosenfeld *et al.*, 2002). Such concordant changes in the expression of both the



(a) (b) (c)

Figure 4.25: Regulator chain A → B → C. (a) Cartoon of the cell at two time points. At time I, transcription factor (TF) A activates transcription factor B, raising the levels of mRNA and protein for transcription factor B. In time II, transcription factor B activates its target. (b) Time series plotting the protein and mRNA levels. (c) Network motif. The gene with the dotted border is predicted as the regulator.

regulator and its targets might allow our automated procedure to detect statistical associations between them.

When a transcription factor is transcriptionally regulated, its mRNA expression might correlate well with its activity. Therefore, in cases where transcription factor A activates transcription factor B, which in turn activates some target C, our approach can possibly capture the regulatory relationship between B and C (see Figure 4.25). Note that steady state gene expression profiles do not observe temporal changes (as those plotted in Figure 4.25). Instead, perturbations of the cell state (mutations and environmental treatments) create statistical associations between a regulator and its target, which are then detected by our learning algorithm.

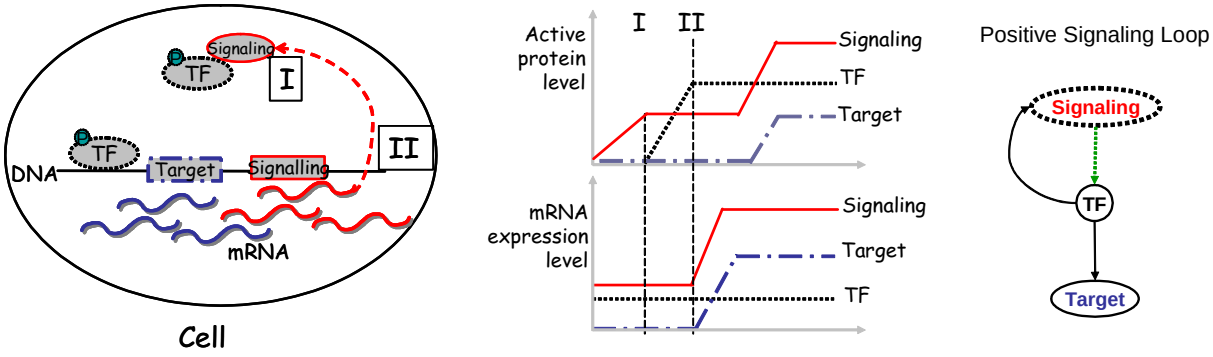
Figure 4.25 provides a very simplistic view of gene regulation. In many cases, gene expression data provides only part of the story: while transcription factors directly control transcription, the factors themselves are frequently regulated at the post-translational level. Furthermore, many transcription factors are active at very low levels of expression, and thus can not be detected reliably with microarrays. In such cases, even if the transcription factor is regulated transcriptionally, current microarray technology can not observe its change. In these cases, our basic assumption does not

hold. While the gene expression data is inherently oblivious to regulation of this kind, an indirect regulatory relationship can be detected. Sometimes our method identifies the regulatory relationship between a signal transduction molecule and its indirect targets. We attribute this ability to the presence of positive and negative feedback loops: a signaling molecule post-transcriptionally activates a transcription factor, which in turn activates its targets. If the signaling molecule is in turn a target of the transcription factor, a feedback loop is formed (see Figure 4.26). In the first step, the cell contains an inactive form of the transcription factor and its targets are not transcribed. Small amounts of the signaling molecule activate the transcription factor without inducing any change to its mRNA level. The activated transcription factor induces the transcription of all its targets, including the signaling molecule. While there is no detectable change in the expression of the transcription factor, the expression of the signaling molecule is concordant with its indirect targets.

Shen-Orr *et al.* (2002) use known *E. Coli* regulatory relations to break down the design of the *E. Coli* transcriptional network into basic building blocks. They define network motifs as patterns of interconnections that recur in the transcriptional network at frequencies much higher than those found in randomized networks. They find that a number of such sub-structures include regulators that are themselves transcriptionally regulated. Lee *et al.* (2002) construct a genome wide regulatory map for *S. Cerevisiae* using a new high throughput experimental approach: cis-regulatory location analysis. This technology measures the binding of transcription factors to the promoter regions of an entire genome. Using genome-wide location data for 106 *S. Cerevisiae* transcription factors they found prevalence of many of the motifs previously detected in the *E. Coli* regulatory network.

We used the data of Lee *et al.* (2002) to test if some of our inferred regulator-regulatee relationships participate in such network motifs. Indeed, many of the inferred regulatory relations are part of such regulatory motifs. We present different types of regulatory components and an example in which such a component is found (these are summarized in Figure 4.27):

- *Regulator chain:* In this chain, a primary transcription factor activates a secondary transcription factor by enhancing its transcription. After the secondary



(a) (b)

Figure 4.26: Positive signaling feedback loop (a) In time I, the signaling molecule activates the transcription factor (TF). In time II, the transcription factor activates all its targets including the signaling gene. Therefore, the expression of the signaling molecule correlates with its indirect targets (b) Time series plotting the protein and mRNA levels. (c) Network motif for the positive signaling feedback loop.

transcription factor is translated into protein, it activates its own targets. In steady state expression profiles, this can result in a statistical association between the secondary transcription factor and targets of both the primary and secondary transcription factors. Note that in regulator chains, only the secondary transcription factor is inferred as a regulator (see Figure 4.25). For example, the transcription factor Phd1 activates a secondary transcription factor, Hap4 (see Figure 4.27(a)). The module network procedure found twenty one genes that are bound by Hap4 in the location dataset to be part of the respiration module which Hap4 regulates. Note that Pet9 was also inferred to be regulated by Hap4, while based on the location data it is regulated by Phd1. In this case our method possibly detected co-regulation instead of regulation. Such cases will be further discussed below.

- *Auto-regulation:* A transcription factor activates both its own transcription and that of its target genes. Thus, the transcription factor is co-expressed along with its targets. For example, Yap6 activates its own transcription and that of its target genes (see Figure 4.27(b)). These were also inferred as part of the Snf

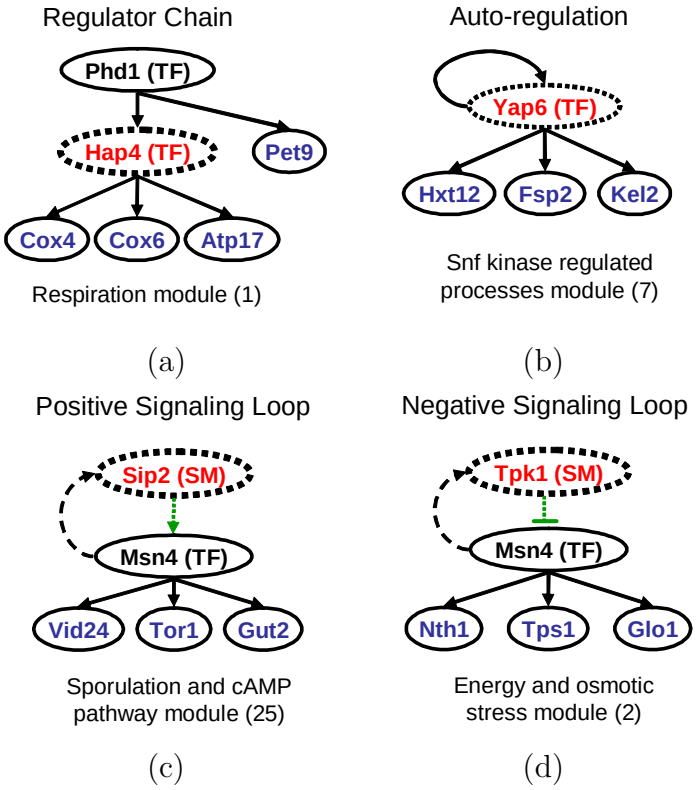


Figure 4.27: Network motifs: For each regulatory component, the relevant transcription factors (TF), signal transduction molecules (SM), target genes and their relations are shown. Cis-regulation events via transcription factors are shown in solid arrows, post-transcriptional events via signaling molecules in dotted arrows. The gene with dotted border is that predicted as a regulator by our method. The solid bordered regulators are not expected to be inferred from gene expression data. The targets included are those genes that are both predicted by our method to be regulated by the dotted regulator and also bound by the transcription factor according to the cis-location data ($p < 0.001$ in Lee *et al.* (2002)).

kinase regulated processes module which Yap6 regulates.

- *Positive signaling loop:* A signaling molecule activates (post transcriptionally) a transcription factor which induces the transcription of various targets, possibly including the signaling molecule. The coordinated expression changes in the signaling molecule and its indirect targets allow the signaling molecule (but

not the transcription factor) to be correctly inferred as a regulator (see Figure 4.26). For example, the signaling molecule Sip2 activates the transcription factor Msn4, which in turn induces transcription of its various targets, including Sip2 (see Figure 4.27(c)).

- *Negative signaling loop:* Similar to the previous example, a negative feedback loop is also possible: a signaling molecule inhibits activity of a transcription factor, which induces transcription of its targets and possibly of the signaling molecule. For example, Tpk1 inhibits the activity of Msn4. Msn4 regulates the targets: Nth1, Tps144, and Glo145 (see Figure 4.27(d)). These are part of the Energy and osmotic stress module regulated by Tpk1. Tpk1's upstream region includes the Msn4 bound STRE motif, supporting Tkp1's regulation by Msn4. However, as both the signaling molecule (Tpk1) and the targets are up-regulated, the method predicts that the signaling molecule's role is activation, in contrast to its actual inhibitory role.

Overall, our results demonstrate that regulatory events, including post-transcriptional ones, have a detectable signature in the expression of genes encoding transcription factors and signal transduction molecules. Nevertheless, the ability of our methods to discover regulatory events is due in part to secondary effects. The methods do not directly detect the post-translational activation of a transcription factor. Rather, due to a feedback loop, they identify the resulting change in the transcription of the signaling molecule.

Despite the successes described above, our methods fails to identify regulatory relations. These false negatives are divided between cases when the regulator's expression does not change sufficiently to merit detection, and cases where our method failed to identify the correct regulatory relation, despite a detectable change in the regulator's expression pattern. A close examination of the relationships missed by our method reveals four categories of false negatives:

- *Undetectable regulators:* If the change in a regulator's activity is attributed exclusively (or mostly) to post-transcriptional changes, we do not expect our

method to capture it. For example, the Dal81 gene is a known transcriptional activator of allantoin, GABA, and urea catabolic genes, when nitrogen sources are low (e.g., in later stationary phase). Although such conditions are included in our data set, there is little change in the expression of the Dal81 gene itself. Thus, our methods could not detect Dal81-mediated regulation. Furthermore, if the regulatory event and the concomitant change in the regulator's expression occur primarily under very specific conditions, which are not included in our data set, we do not expect our methods to capture it. While missed in our current analysis, these latter relations can be captured by our methods given a richer dataset.

- *Regulator redundancy:* If several regulators participate in the same regulatory event, due to "explaining away" we expect our methods to capture only some representatives, missing the remaining regulators. This limitation applies both when several transcription factors work in one complex and when several signal transduction molecules and/or transcription factors are concatenated in one regulator chain. For example, the Hap2/3/4/5 transcription factors work in a ternary complex to regulate the expression of respiration genes. While the module network procedure correctly captured Hap4 as a regulator of respiration it failed to identify Hap2, 3, and 5 which have a "redundant function". Note that the changes in Hap4's expression are the most pronounced among these four potential regulators, explaining the method's specific choice. Note that this limitation of our methods does not apply to combinatorial regulation, when several regulators have only partly overlapping roles (e.g., Hap4 and Msn4 in the respiration module).

- *Co-regulation redundancy:* Many modules contain target genes that also happen to belong to the candidate regulator set. As these genes often have an expression profile which is similar to that of the other target genes in the module, they may mistakenly be chosen as a regulator for the module, in some cases rendering the true regulator redundant. In such cases, the true regulator is often assigned as a module member, along with its targets (e.g., Uga3 and Dal80 in the Nitrogen

Catabolite Repression module).

- *Gene specific regulation:* Even when a regulator's expression pattern is highly (and possibly uniquely) predictive of that of its known target, this relationship may be specific to a single target, and cannot be generalized to an entire module or set of genes. In such cases, the module network procedure, which is aimed at identifying shared regulatory responses, would not detect the regulatory relation. We believe that the statistical robustness and biological conciseness gained by taking a general perspective of regulation outweighs this particular limitation of our approach.

Thus, while we attempt to limit the false positives, our reconstruction approach inherently leads to many false negatives. We do not reconstruct the entire regulatory program of an entire organism, rather only certain aspects of it. Further work is required in order to explain why a specific transcription factor is chosen over other potential ones, why a signaling molecule is identified as a regulator instead of its cognate transcription factor in certain cases (e.g., Tpk1 and Msn4, both of which have a strong expression signature) but not in others, or why a particular combination of regulators has been selected. While some of the reasons may have to do with our computational method, others may be related to critical biological issues such as the strength of feedback regulation and the coordination of action between various regulators. Answers to these questions can both aid the design of better regulatory models and algorithms to reconstruct them and can illuminate the regulation of regulators and the coordination of their actions.

4.9 Conclusions

Discovering biological organization from gene expression data is a promising but challenging task. In this chapter, we introduced the framework of *module networks*, which offers unique capabilities in extracting modularity and regulation from expression data. Furthermore, the framework of *module networks* developed for this task can be successfully applied to other domains. For example, we applied the module network

procedure to stock market data (Segal *et al.*, 2003b), where the data consisted of changes in the price of stocks in consecutive trading days and our goal was to identify sets of stocks whose pricing patterns are similar and can be explained as a function of the pricing patterns of a small set of other stocks.

The statistical advantages of module networks allow us to learn detailed regulatory programs over thousands of gene solely from approximately one hundred gene expression data samples. Overall, our resulting model provides a clear global view of functional modules and their regulation, that corresponds well to the known biological literature. Perhaps the most powerful feature of our method is its ability to generate detailed testable hypotheses concerning the role of specific regulators and the conditions under which this regulation takes place. Microarray experiments that we carried out based on the computational predictions for three putative regulators with as yet unknown functions (a transcription factor and two signaling molecules) validated the method's predictions and provided important insight regarding the function of these uncharacterized regulators. As more diverse gene expression datasets become available, it is our belief that applying the module networks may result in important new insights in the ongoing endeavor to understand the complex web of biological regulation.

Despite the success of the module network procedure in identifying regulators of biological processes our approach has several limitations. First, the assumption that the regulators themselves have detectable changes in their expression that are coordinated with the expression of their targets results in several types of regulatory relationships that cannot be inferred by our method. We discussed the implications of this assumption in detail in Section 4.8.

A second limitation of our approach is that each gene is assigned to exactly one module. Thus, a regulatory module is defined over the entire set of arrays in the input dataset. However, in the biological domain, many genes are known to be involved in several different processes where the processes differ in their associated genes. As these processes may be activated under different conditions, a module network, by restricting each gene to only one module, cannot represent such overlapping processes with different regulatory mechanisms. In a separate project (Segal *et al.*, 2003a, Battle

et al., 2004), we presented one way to extend our probabilistic model that allows genes to be assigned to several modules. The expression of a gene in a particular array is then modeled as a sum of its expression in each of the modules in which it participates, and each module can potentially have a different set of regulators.

Another limitation of our approach is that the number of modules is determined in advance and given as input to the method. Obviously, the number of regulatory modules of an organism in an expression dataset is not known and thus determining the number of modules should be part of the regulatory module discovery task. In Section 4.5, we showed that at least in synthetic data, where the number of modules is known, we can use the score of the model to select the correct number of modules by choosing the model with the smallest number of modules from among the highest scoring models. This observation is encouraging, as it suggests that we can extend our approach to select the number of modules automatically by adding search steps that modify the number of modules and use the model score to compare models that differ in their number of modules.

Finally, in the application of the module network procedure to the biological domain, we pre-selected a candidate set of regulatory genes and allowed the learned model to only use these candidate regulators as parents in the learned regulation programs. Thus, we had to know the potential regulators in advance and our results may be sensitive to this choice of input regulators. Furthermore, it precludes the identification of regulatory genes that were not included in the candidate regulator set. Thus, an interesting extension would be to learn regulatory modules while allowing all genes to be regulators. However, in such an extension we must find other means for biasing the learned model towards assigning regulatory proteins as parents of regulation programs. For example, if we can identify distinguishing features of the general class of regulatory proteins, we might be able to use those to bias the learned model. In addition to expanding the set of regulators that can be considered as regulators, such an extension can also reveal interesting aspects of the behavior of regulators and their interaction with their target genes.

Chapter 5

Conserved Regulatory Modules

The recent sequencing efforts are producing complete genomes for many organisms at a fast rate. Combined with advances in new technologies for obtaining genome-wide information at the sub-cellular level, there are several organisms for which we now have several types of data on a genome-wide scale. Each of these data types provides an informative view of one aspect of the cellular machinery. Thus, by combining different types of data we can arrive at more reliable and biologically meaningful conclusions. We saw one such example in Chapter 3, where we combined both DNA sequence data and genome-wide expression measurements into a single probabilistic framework. However, an interesting question is whether we can further improve our understanding of the biological system by combining data across multiple organisms.

While there has been much work on combining sequence data across organisms, very little work has been done on combining other types of functional data. In this chapter, we present an extension to the module network framework of Chapter 4 that combines expression data from multiple organisms for the task of discovering regulatory modules that have been conserved across evolution. As in module networks, our procedure identifies modules of co-regulated genes, their regulators, and the conditions under which this regulation occurs. The key difference is that the regulatory modules in our extended procedure are jointly learned from expression profiles of several organisms. As we show in an application of the method to expression measurements of brain tumors from human and mouse, this joint learning allows us to

improve our regulatory predictions compared to models learned from each organism's data in isolation. Moreover, we show that by combining expression data from human and mouse, we can gain insights into the evolution of the regulatory relationships in brain tumors between these organisms.

We start this chapter with an overview of our approach, followed by a formal presentation of the underlying probabilistic model and our algorithm for learning this model automatically from expression data from multiple organisms. We then show the results obtained when applying the framework to the human and mouse brain tumor expression profiles mentioned above.

5.1 Model Overview

An implicit assumption made in the models we presented thus far has been that co-regulation of the mRNA expression profiles of a set of genes, observed in expression data, implies some functional relatedness among the gene set. While this assumption is true in many cases, it is definitely not true in all cases. Some reasons have to do with noise in the technology for measuring the data. For example, the mRNA products of two genes with similar sequences might cross-hybridize and bind to each other's designated spots on the microarray chip. Such genes may appear to be co-regulated even though they really are not.

However, even if the measuring technology was perfect, there may still be cases in which truly co-regulated genes are not functionally related. For example, physically close neighboring genes on the DNA that are transcribed from opposite DNA strands may share the same DNA control element or promoter. A transcription factor that binds this common promoter may thus regulate the expression of both genes. Such gene pairs will be co-regulated but the reason may have to do with their commonly shared promoter rather than their participation in the same biological process. As another example, cis-regulatory DNA motifs can occur by chance in the genome and might lead to serendipitous transcriptional regulation of nearby genes. Such genes may be co-regulated with the 'true' targets of the binding transcription factor, even though they are not related in function to the true targets. In both examples, genes

that are not related to the cellular response or the process carried out by the cell may be co-regulated with the genes that are. In most cases, the biological system is robust enough to handle such redundancies.

The discussion above implies that in some cases unrelated genes may be co-regulated. How can we distinguish accidentally regulated genes from those that are physiologically important?

One powerful way, which has been employed with great success in sequence analysis, is to use evolution as a filter. As organisms evolved from a common ancestor in an evolutionary process that selects against mutations in functional elements, the idea is that by finding those sequence elements that have been conserved between organisms throughout evolution, we are likely to enrich for the more functionally relevant elements.

In this chapter, we extend the use of the evolution filter idea beyond sequence data, and ask whether we can improve our ability to discover regulatory relationships by combining expression data across different organisms. We (Stuart *et al.*, 2003) and others (Bergmann *et al.*, 2004) have shown that by integrating expression data from multiple organisms we can improve the prediction of gene function. The basic idea was to identify those co-expression relationships that have been conserved across organisms and use only those for the gene function prediction task. The assumption was that the functionally irrelevant co-expression relationships that are due to random noise would not be preserved across organisms, while the relevant co-expression relationships, by being functionally important will be selected for and thus conserved across evolution. Thus, by identifying the conserved co-expression relationships we were able to filter out many irrelevant relationships and improve our ability to predict the function of genes.

We use a similar assumption for the regulatory module discovery task: regulator-regulatee relationships that have been conserved across evolution confer some selective advantage. Thus, by finding them we might be able to improve regulatory module discovery.

We design a joint probabilistic model over expression data from multiple organisms, which directly extends the module network framework presented in Chapter 4.

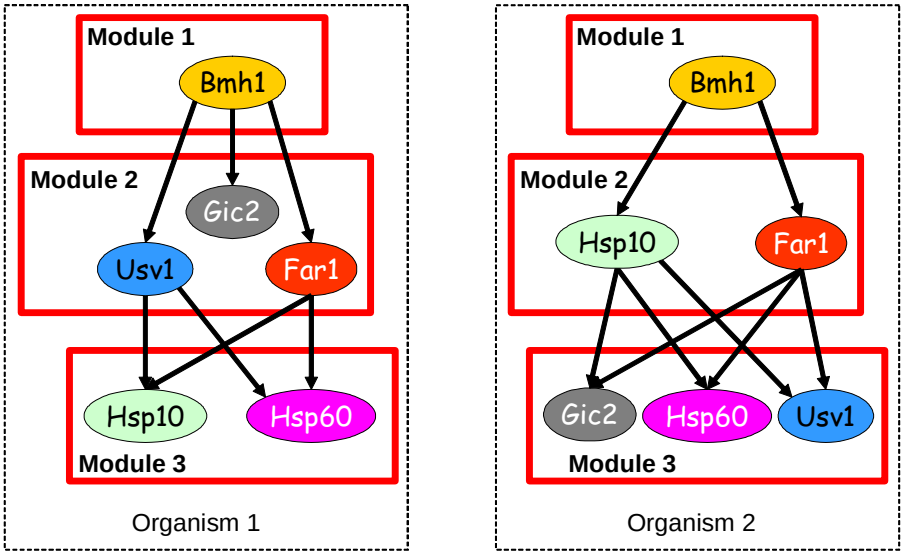


Figure 5.1: Example of a parallel module network with three modules for two organisms, each with five orthologous genes. Genes are shown as filled ovals. Orthologous genes are given the same name and their corresponding ovals are filled with the same color in both organisms. In the example shown, three genes are assigned to the same module in both organisms (Bmh1, Far1, and Hsp60) while three genes have different module assignments in the two organisms (Usv1, Gic2, and Hsp10). Two regulatory relationships are shared between organisms (Bmh1 as a regulator of module 2, and Far1 as a regulator of module 3), while two relationships exist only in one of the organisms (Usv1 as a regulator of module 3 in organism 1, and Hsp10 as a regulator of module 3 in organism 2).

We refer to this extension as a *parallel module network*. The input for a parallel module network consists of expression data from ℓ organisms. As our focus is discovering conserved regulatory modules, we construct the model only over the data for evolutionary conserved genes. We use *orthologous* genes as a proxy for evolutionary conserved genes, where an orthologous pair of genes consists of two genes from two organisms that are derived from a common ancestral gene. We assume that orthologous genes perform the same function in the two organisms. We further assume that the set of orthologs is known. Thus, in addition to the input expression datasets, our method also takes as input an *orthology map* which specifies the orthology relationships that hold between all the genes in the ℓ input organisms.

Under these assumptions, a parallel module network consists of ℓ separate module networks, where each module network is defined over the expression data of one of the organisms and has K modules. An example of a parallel module network for two organisms and five genes is shown in Figure 5.1. The probabilistic model is such that each module network is independent of the other networks, except for two modifications that bias the model towards finding regulatory modules that are conserved across the input organisms.

The first modification captures our intuition that orthologs are more likely to be assigned to the same module across different organisms. We encode this modification in the module assignment prior probability distribution, which assigns a higher likelihood to an assignment in which orthologs are assigned to the same module in different organisms. Note that encoding this preference in the module assignment prior does not force orthologs to be assigned to the same module. This feature of the model is important, as we want to allow for the possibility that orthologs diverged in function during evolution and thus their module assignment differs.

The second modification we make to the model favors the sharing of regulator-regulatee relationships across the otherwise ℓ independent module networks. We encode this bias in the structure prior probability distribution, which assigns a higher likelihood to models in which corresponding modules in different organisms share the same parent regulators. As in the case of the module assignment prior, this sharing of relationships is not forced.

We present an algorithm for learning a parallel module network from data that takes the above module assignment and structure prior into account. Thus, by changing the strength of these priors we can learn models with a varying degree of shared regulatory relationships. Interestingly, as we show in an application of this model to expression data from human and mouse brain tumors, a parallel module network learns better regulatory modules in both human and mouse compared to the regulatory modules learned by a separate module network over the human expression data and a separate module network over the mouse data. Moreover, we show that this gain is indeed due to evolutionary conservation of regulatory relationships, as no gain is observed when combining random data from another organism or when scrambling

the mapping of orthologous genes across organisms.

5.2 Probabilistic Model

We now provide a formal definition of our probabilistic model. Although the model can be applied in other settings, we focus the presentation on the application to gene expression data. As discussed above, our model is a direct extension of the module network framework presented in Chapter 4. We refer to this extension as a *parallel module network*.

¿From the representation perspective, a parallel module network over ℓ organisms is nothing more than ℓ separate instances of the module network model, each defined over the expression data of one of organism. The only two modifications we make to this model are in the module assignment and structure prior probability distributions. We introduce these changes in Section 5.3, where we discuss how we learn a parallel module network automatically from data.

As discussed above, we construct the model only over genes that have at least one ortholog in one of the other organisms, and assume that this orthology map is given to us as input. We represent this orthology map by a graph $\mathcal{O}$ that has a node for every gene in each of the ℓ organism, and every pair of orthologous genes are connected by an edge in $\mathcal{O}$. Using the orthology map, and following the module network definitions in Section 4.2, we define a parallel module network as:

Definition 5.2.1: A *parallel module network* $\mathcal{G}$ over ℓ organisms consists of an orthology map $\mathcal{O}$ and a set of module networks $(\mathcal{M}_1, \ldots, \mathcal{M}_\ell)$, where each module network $\mathcal{M}_i$ is a triple $(\mathcal{M}_i.\mathcal{C}, \mathcal{M}_i.\mathcal{T}, \mathcal{M}_i.\mathcal{A})$ defined as in Section 4.2, i.e., $\mathcal{M}_i.\mathcal{C}$ is a module set, $\mathcal{M}_i.\mathcal{T}$ is a module network template for $\mathcal{M}_i.\mathcal{C}$ (see Definition 4.2.1), and $\mathcal{M}_i.\mathcal{A}$ is a module assignment function for $\mathcal{M}_i.\mathcal{C}$ (see Definition 4.2.2). In addition:

- The module network $\mathcal{M}_j$ for each organism j is defined over a set of n_j random variables $\mathcal{M}_j.\mathcal{X} = \{X_1, \ldots, X_{n_j}\}$ of the same type (i.e., for every pair of variables $X_i \in \mathcal{M}_j.\mathcal{X}$ and $X_l \in \mathcal{M}_j.\mathcal{X}$, $Val(X_i) = Val(X_l)$).

- The module set $\mathcal{M}_j.\mathcal{C}$ for each organism j has exactly K modules, i.e., $\mathcal{M}_j.\mathcal{C} = \{\mathbf{M}_1, \ldots, \mathbf{M}_K\}$ of the same type (i.e., for every pair of modules $\mathbf{M}_i$ and $\mathbf{M}_l$, $Val(\mathbf{M}_i) = Val(\mathbf{M}_l)$).

- The orthology map $\mathcal{O}$ is a graph that has a node for each random variable $\mathcal{M}_j.X_i$ of every module network $\mathcal{M}_j$, and every pair of orthologous genes from two different organisms are connected by an edge in $\mathcal{O}$ (i.e., $\mathcal{M}_j.X_{i_1}$ and $\mathcal{M}_l.X_{i_2}$ are connected by an edge if gene i_1 in the j-th organism and gene i_2 in the l-th organisms are orthologs).

∎

In our gene expression domain, the random variable $\mathcal{M}_j.X_i$ represents the mRNA expression level of the i-th gene in organism j. In our example from Figure 5.1 there are two organisms. The genes for both organism 1 and organism 2 are {*Bmh1*, *Gic2*, *Usv1*, *Far1*, *Hsp10*, *Hsp60*}, where $\mathcal{M}_1.Bmh1$ and $\mathcal{M}_2.Bmh1$ are orthologs, $\mathcal{M}_1.Gic2$ and $\mathcal{M}_2.Gic2$ are orthologs, and so on. In addition, the module assignments for organism 1 are $\mathcal{M}_1.\mathcal{A}(Bmh1) = 1, \mathcal{M}_1.\mathcal{A}(Gic2) = 2, \mathcal{M}_1.\mathcal{A}(Usv1) = 2$, and so on, while the module assignments for organism 2 are $\mathcal{M}_2.\mathcal{A}(Bmh1) = 1, \mathcal{M}_2.\mathcal{A}(Gic2) = 3, \mathcal{M}_2.\mathcal{A}(Usv1) = 3$, and so on.

A parallel module network defines a probabilistic model using the probabilistic model induced by the module network associated with each organism. Specifically, we define the semantics of a parallel module network by "unrolling" a Bayesian network where all of the variables assigned to module $\mathcal{M}_i.\mathbf{M}_j$ share the parents and conditional probability template assigned to $\mathcal{M}_i.\mathbf{M}_j$ in $\mathcal{M}_i.\mathcal{T}$. Thus, the resulting Bayesian network consists of ℓ independent Bayesian networks, each unrolled from a module network of one of the organisms. For this unrolling process to produce a well-defined distribution, the resulting network must be acyclic. By our construction, acyclicity can be guaranteed by requiring the Bayesian network induced from the module network associated with each organism to be acyclic, as described in Definition 4.2.3.

We can now define the semantics of a parallel module network:

Definition 5.2.2: A parallel module network $\mathcal{G} = (\mathcal{M}_1, \ldots, \mathcal{M}_\ell)$ defines a *ground Bayesian network* $\mathcal{B}_\mathcal{G}$ over $\mathcal{M}_1.\mathcal{X}, \ldots, \mathcal{M}_\ell.\mathcal{X}$ as follows: For each variable $\mathcal{M}_i.X_j \in \mathcal{M}_i.\mathcal{X}$, where $\mathcal{M}_i.\mathcal{A}(\mathcal{M}_i.X_j) = k$, we define the parents of $\mathcal{M}_i.X_j$ in $\mathcal{B}_\mathcal{G}$ to be $\mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_k}$, and its conditional probability distribution to be $P(\mathcal{M}_i.\mathbf{M}_k \mid \mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_k})$, as specified in $\mathcal{M}_i.\mathcal{T}$. The distribution associated with $\mathcal{G}$ is the one represented by the Bayesian network $\mathcal{B}_\mathcal{G}$. To guarantee that this distribution is coherent, we require the module graph $\mathcal{G}_{\mathcal{M}_i}$ (see Definition 4.2.3) of each module network $\mathcal{M}_i$ to be acyclic. ∎

5.3 Learning the Model

We now turn to the task of learning parallel module networks from data. Recall that a parallel module network is specified by an orthology map $\mathcal{O}$ and a set of ℓ module networks, one for each of the ℓ organisms. Each module network $\mathcal{M}_i$ is in turn specified by a set of K modules, an assignment function $\mathcal{M}_i.\mathcal{A}$ of genes to modules, the parent structure $\mathcal{M}_i.\mathcal{S}$ specified in $\mathcal{M}_i.\mathcal{T}$, and the parameters $\mathcal{M}_i.\boldsymbol{\theta}$ for the local probability distributions $P(\mathcal{M}_i.\mathbf{M}_j \mid \mathcal{M}_i.\mathbf{Pa}_{\mathbf{M}_j})$.

As in Chapter 4, we focus on the most general task of learning the network structure and the assignment function, as well as a Bayesian posterior over the network parameters for each of the ℓ module networks.

Thus, we are given a training set $\mathcal{D}$ which includes ℓ gene expression datasets, i.e., $\mathcal{D} = \{\mathcal{M}_1.\mathcal{D}, \ldots, \mathcal{M}_\ell.\mathcal{D}\}$, where the expression dataset for the i-th organism, $\mathcal{M}_i.\mathcal{D} = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[M_i]\}$, consists of M_i instances drawn independently from an unknown distribution $P(\mathcal{M}_i.\mathcal{X})$. Our primary goal is to learn a module network structure $\mathcal{M}_i.\mathcal{S}$ and assignment function $\mathcal{M}_i.\mathcal{A}$ for each distribution $P(\mathcal{M}_i.\mathcal{X})$.

In Chapter 4, we took a *score-based approach* to the learning task of each module network, and defined a Bayesian scoring function that measures how well each candidate model fits the observed data. If we use this scoring function for each of the ℓ module networks, then we can learn each module network independently of the other networks using the algorithms presented in Section 4.3. However, recall that our goal is to learn networks in which orthologs are assigned to the same module in

different organisms and regulator-regulatee relationships are shared across the regulation programs of corresponding modules in different organisms. We encode these preferences in a module assignment and network structure prior distributions that are defined jointly over all ℓ module networks and take into account the joint assignments and structures of all module networks. Thus, as the prior distributions are no longer independent across the ℓ module networks, we cannot learn each module network independently. We thus focus this section on the algorithmic problem of finding a high scoring model for all ℓ module networks in the presence of these joint prior distributions.

5.3.1 Likelihood Function

We begin by examining the *data likelihood* function

$$L(\mathcal{G} : \mathcal{D}) = P(\mathcal{D} \mid \mathcal{G}) = \prod_{i=1}^{\ell} L(\mathcal{M}_i : \mathcal{M}_i.\mathcal{D}) = \prod_{i=1}^{\ell} \prod_{m=1}^{M_i} P(\mathcal{M}_i.\boldsymbol{x}[m] \mid \mathcal{M}_i.\mathcal{T}, \mathcal{M}_i.\mathcal{A}).$$

This function plays a key role both in the parameter estimation task and in the definition of the structure score.

As the semantics of a parallel module network is defined via a ground Bayesian network composed of ℓ independent ground Bayesian networks, one for each organism, we have that the likelihood decomposes into a product of likelihood functions, one for each module network $\mathcal{M}_i$. As in Equation 4.1, the likelihood for each module network decomposes according to modules, such that we can write:

$$\begin{aligned} & L(\mathcal{G} : \mathcal{D}) \\ &= \prod_{i=1}^{\ell} \prod_{j=1}^{K} \left[\prod_{m=1}^{M_i} \prod_{\mathcal{M}_i.X_k \in \mathcal{M}_i.\boldsymbol{X}^j} P(\mathcal{M}_i.x_k[m] \mid \mathbf{pa}_{\mathcal{M}_i.\mathbf{M}_j}[m], \boldsymbol{\theta}_{\mathcal{M}_i.\mathbf{M}_j \mid \mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j}}) \right] \\ &= \prod_{i=1}^{\ell} \prod_{j=1}^{K} L_j(\mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j}, \mathcal{M}_i.\boldsymbol{X}^j, \theta_{\mathcal{M}_i.\mathbf{M}_j \mid \mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j}} : \mathcal{M}_i.\mathcal{D}), \end{aligned} \tag{5.1}$$

where $\mathcal{M}_i.\boldsymbol{X}^j = \{X \in \mathcal{M}_i.\mathcal{X} \mid \mathcal{M}_i.\mathcal{A}(X) = j\}$, and $\boldsymbol{\theta}_{\mathcal{M}_i.\mathbf{M}_j \mid \mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j}}$ are the parameters associated with the CPD template $P(\mathcal{M}_i.\mathbf{M}_j \mid \mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j})$.

As described in Section 4.3.1, when learning conditional probability distributions from the exponential family (e.g., discrete distribution, Gaussian distributions, and many others), then the module likelihood functions can be reformulated in terms of *sufficient statistics* of the data. As the likelihood for a parallel module network decomposes as a product of likelihoods, one for each module network, the formula for the module likelihood as a function of the sufficient statistics are identical to those presented in Section 4.3.1.

5.3.2 Priors and the Bayesian Score

As we discussed, our approach for learning parallel module networks is based on the use of a Bayesian score. Specifically, we define a model score for a pair $(\mathcal{S}, \mathcal{A})$ as the posterior probability of the pair, integrating out the possible choices for the parameters $\boldsymbol{\theta}$, where $\mathcal{S} = \{\mathcal{M}_1.\mathcal{S}, \ldots, \mathcal{M}_\ell.\mathcal{S}\}$ and $\mathcal{A} = \{\mathcal{M}_1.\mathcal{A}, \ldots, \mathcal{M}_\ell.\mathcal{A}\}$ are the vectors of network structures and assignment functions for all ℓ module networks, respectively. We define an assignment prior $P(\mathcal{A})$, a structure prior $P(\mathcal{S} \mid \mathcal{A})$ and a parameter prior $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$. These describe our preferences over different networks *before* seeing the data. By Bayes' rule, we then have

$$P(\mathcal{S}, \mathcal{A} \mid \mathcal{D}) \propto P(\mathcal{A})P(\mathcal{S} \mid \mathcal{A})P(\mathcal{D} \mid \mathcal{S}, \mathcal{A})$$

where the last term is the *marginal likelihood*

$$P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) = \int P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}, \boldsymbol{\theta})P(\boldsymbol{\theta} \mid \mathcal{S})d\boldsymbol{\theta}.$$

We define the Bayesian score as the log of $P(\mathcal{S}, \mathcal{A} \mid \mathcal{D})$, ignoring the normalization constant

$$\text{score}(\mathcal{S}, \mathcal{A} \ : \ \mathcal{D}) = \ \log P(\mathcal{A}) + \log P(\mathcal{S} \mid \mathcal{A}) + \log P(\mathcal{D} \mid \mathcal{S}, \mathcal{A}) \tag{5.2}$$

As discussed above, a parallel module network is equivalent to ℓ independent module networks with two modifications. The first modification biases the learned models

to ones in which orthologous genes are assigned to the same module in the module networks of the different organisms. We encode this bias in the prior probability over assignments, $P(\mathcal{A})$. Intuitively, assignments in which more orthologs are assigned to the same module should receive higher prior probabilities. We choose to represent this probability by pairwise terms over the assignments of each pair of organisms as follows:

$$P(\mathcal{A}) \propto \prod_{j=1}^{K} \left(\prod_{i=1}^{\ell} \alpha(\mathcal{M}_i.\mathcal{A}_j) \right) \left(\prod_{i_1=1}^{\ell} \prod_{i_2=i_1+1}^{\ell} f_{\mathcal{A}}(\mathcal{M}_{i_1}.\mathcal{A}_j, \mathcal{M}_{i_2}.\mathcal{A}_j) \right) \tag{5.3}$$

where $\mathcal{M}_i.\mathcal{A}_j$ is the choice of variables assigned to module j in the module network for organism i, α is a function from a set of variables to the positive reals, and $f_{\mathcal{A}}$ is a function from two sets of variables to the positive reals such that:

$$\begin{aligned} f_{\mathcal{A}}(\mathcal{M}_{i_1}.\mathcal{A}_j, \mathcal{M}_{i_2}.\mathcal{A}_j) \;\; = \;\; & \exp[\lambda \sum_{i=1}^{n_{i_1}} \sum_{l=1}^{n_{i_2}} \eta\{\mathcal{M}_{i_1}.\mathcal{A}(\mathcal{M}_{i_1}.X_i) = \mathcal{M}_{i_2}.\mathcal{A}(\mathcal{M}_{i_2}.X_l) = j)\} \\ & \eta\{(\mathcal{M}_{i_1}.X_i, \mathcal{M}_{i_2}.X_l) \in \mathcal{O}\}] \end{aligned}$$

where $\eta\{\mathcal{M}_{i_1}.\mathcal{A}(\mathcal{M}_{i_1}.X_i) = \mathcal{M}_{i_2}.\mathcal{A}(\mathcal{M}_{i_2}.X_l) = j)\}$ is an indicator function that is equal to 1 if and only if $\mathcal{M}_{i_1}.X_i$ is assigned to module j in the module network of organism i_1, and $\mathcal{M}_{i_2}.X_l$ is assigned to module j in the module network of organism i_2, and $\eta\{(\mathcal{M}_{i_1}.X_i, \mathcal{M}_{i_2}.X_l) \in \mathcal{O}\}$ is an indicator function that is equal to 1 if and only if gene i in organism i_1 and gene l in organism i_2 are orthologs according to the orthology map $\mathcal{O}$. Thus, $f_{\mathcal{A}}$ is parameterized by a single parameter $\lambda \geq 0$ and assigns a 'bonus' of e^{λ} to every orthologous gene pair in which the two orthologs from the respective organisms are assigned to the same module.

We note that other choices for the form of $f_{\mathcal{A}}$ are also possible. Our choice of decomposing $f_{\mathcal{A}}$ by pairs of organisms, and parameterizing it with a single λ parameter is motivated by its simplicity and extensibility to several organisms. The function α is similar to its definition in Definition 4.3.1 and may be used to encode our prior preferences over the number of genes assigned to each module. We also note that we can describe the part of the assignment prior that favors assigning orthologs to the same module using a *Markov network* (Pearl, 1988) that has a random variable

for every gene in each organism representing its module assignment. We then have pairwise potentials over every pair of orthologous genes, where this potential has the value λ for entries in which the pair of variables have the same module assignment and 1 for all other entries. Note that this Markov network has the same structure as the orthology map $\mathcal{O}$.

The second modification we make in a parallel module network biases the learned models to ones in which regulator-regulatee relationships are shared across the module networks for the different organisms. We encode this bias in the prior probability over network structures, $P(\mathcal{S})$. Intuitively, structures in which orthologs are assigned as parents of corresponding modules in different organisms should receive higher probabilities. We represent this probability in a similar form to the prior over module assignments, using pairwise terms over the network structures of each pair of organisms as follows:

$$P(\mathcal{S}) \propto \prod_{j=1}^{K} \left(\prod_{i=1}^{\ell} \rho(\mathcal{M}_i.\mathcal{S}_j) \right) \left(\prod_{i_1=1}^{\ell} \prod_{i_2=i_1+1}^{\ell} f_{\mathcal{S}}(\mathcal{M}_{i_1}.\mathcal{S}_j, \mathcal{M}_{i_2}.\mathcal{S}_j) \right) \tag{5.4}$$

where $\mathcal{M}_i.\mathcal{S}_j$ denotes the choice of regulator parents for module j, ρ is a function from a set of parents for module j to the positive reals as described in Definition 4.3.1, and $f_{\mathcal{S}}$ is a function from two sets of parents to the positive reals such that:

$$\begin{aligned} f_{\mathcal{S}}(\mathcal{M}_{i_1}.\mathcal{S}_j, \mathcal{M}_{i_2}.\mathcal{S}_j) \;=\; & \exp[\gamma \sum_{i=1}^{n_{i_1}} \sum_{l=1}^{n_{i_2}} \eta\{\mathcal{M}_{i_1}.X_i \in \mathcal{M}_{i_1}.\mathcal{S}_j\} \eta\{\mathcal{M}_{i_2}.X_l \in \mathcal{M}_{i_2}.\mathcal{S}_j\} \\ & \eta\{(\mathcal{M}_{i_1}.X_i, \mathcal{M}_{i_2}.X_l) \in \mathcal{O}\}] \end{aligned}$$

where $\eta\{\mathcal{M}_{i_1}.X_i \in \mathcal{M}_{i_1}.\mathcal{S}_j\}$ is an indicator function that is equal to 1 if and only if $\mathcal{M}_{i_1}.X_i$ is a parent of module j in the module network of organism i_1, and $\eta\{(\mathcal{M}_{i_1}.X_i, \mathcal{M}_{i_2}.X_l) \in \mathcal{O}\}$ is an indicator function that is equal to 1 if and only if the parent gene i in organism i_1 and the parent gene l in organism i_2 are orthologs according to the orthology map $\mathcal{O}$. Thus, $f_{\mathcal{S}}$ is parameterized by a single parameter $\gamma \geq 0$ and assigns a 'bonus' of e^{γ} to every orthologous gene pair and every module j where each gene in the pair is a parent (regulator) of module j in its respective organism. Note that in our construction, an orthologous pair may receive more than

one bonus, if there is more than one module that has both genes as its parents. Again, we note that while our choice of $f_{\mathcal{S}}$ is motivated by simplicity, other choices are also possible. We also note that we can represent the part of the structure prior that favors assigning orthologous pairs to corresponding modules in different organisms by a Markov network using a similar network construction to the one described above for the assignment prior.

In Section 4.3.2 we presented a list of conditions (see Definition 4.3.1) under which the Bayesian score for a module network decomposes into local *module scores*. To allow efficient evaluation of a large number of alternative models, we want similar conditions to hold in the case of parallel module networks. As the module assignment prior and structure prior are different in a parallel module network, we need to revisit two conditions from Definition 4.3.1. The first is structure modularity, which states that the prior over structures decomposes as a product over terms, one for each module. The second is assignment modularity, which states that the prior over assignments decomposes as a product of terms, one for each module. Note that in both the modified assignment prior (Equation 5.3) and the modified structure prior (Equation 5.4), we took care to maintain this property of decomposability according to modules. Thus, following the same arguments of Theorem 4.3.2, we can prove the decomposability property of the Bayesian score for parallel module networks:

Theorem 5.3.1: Let $P(\mathcal{A})$, $P(\mathcal{S} \mid \mathcal{A})$, $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$ be assignment, structure, and parameter priors. When the assignment prior $P(\mathcal{A})$ follows the form of Equation 5.3, the structure prior $P(\mathcal{S} \mid \mathcal{A})$ follows the form of Equation 5.4, and the parameter prior $P(\boldsymbol{\theta} \mid \mathcal{S}, \mathcal{A})$ satisfies the assumptions of Definition 4.3.1, then the Bayesian score of a parallel module network decomposes into local *module scores*:

$$\text{score}(\mathcal{S}, \mathcal{A} \ : \ \mathcal{D}) = \sum_{j=1}^{K} \text{score}_{\mathbf{M}_j}(\mathbf{Pa}_{\mathcal{M}_1.\mathbf{M}_j}, \mathcal{M}_1.\mathcal{A}_j, \ldots, \mathbf{Pa}_{\mathcal{M}_\ell.\mathbf{M}_j}, \mathcal{M}_\ell.\mathcal{A}_j \ : \ \mathcal{D})$$

$$\text{score}_{\mathbf{M}_j}(\mathcal{M}_1.\mathcal{S}_j, \mathcal{M}_1.\mathcal{A}_j, \ldots, \mathcal{M}_\ell.\mathcal{S}_j, \mathcal{M}_\ell.\mathcal{A}_j \ : \ \mathcal{D}) = \tag{5.5}$$

$$\sum_{i=1}^{\ell} \log \int L_j(\mathcal{M}_i.\mathcal{S}_j, \mathcal{M}_i.\mathcal{A}_j, \theta_{\mathcal{M}_i.\mathbf{M}_j|\mathcal{M}_i.\mathcal{S}_j} : \mathcal{M}_i.\mathcal{D}) P(\boldsymbol{\theta}_{\mathcal{M}_i.\mathbf{M}_j} \mid \mathcal{M}_i.\mathcal{S}_j) d\theta_{\mathcal{M}_i.\mathbf{M}_j|\mathcal{M}_i.\mathcal{S}_j} +$$

$$\log P(\mathcal{M}_1.\mathcal{S}_j, \ldots, \mathcal{M}_\ell.\mathcal{S}_j) + \log P(\mathcal{M}_1.\mathcal{A}_j, \ldots, \mathcal{M}_\ell.\mathcal{A}_j)$$

where $\mathcal{M}_i.\mathcal{S}_j$ denotes that we chose a structure in which $\mathcal{S}_j$ are the parents of module j in the module network of organism i (i.e., $\mathcal{M}_i.\mathcal{S}_j = \{X \in \mathcal{M}_i.\mathcal{X} \mid X \in \mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j}\}$), and $\mathcal{M}_i.\mathcal{A}_j$ denotes that we chose an assignment function in which $\mathcal{A}_j$ is the set of variables assigned to module j in the module network of organism i (i.e., $\mathcal{M}_i.\mathcal{A}_j = \{X \in \mathcal{M}_i.\mathcal{X} \mid \mathcal{M}_i.\mathcal{A}(X) = j\}$).

Note that, while the module likelihood scores further decompose by the module network of each organism, the terms for the module assignment prior and structure prior do not, as we modified them such that they are not independent across organisms. As we shall see below, the decomposition of the Bayesian score plays a crucial rule in our ability to devise an efficient learning algorithm that searches the space of parallel module networks for one with a high score.

5.3.3 Structure Search Step

Given a scoring function over networks, we now consider the task of finding a high scoring parallel module network. This problem is a challenging one, as it involves searching over two combinatorial spaces simultaneously — the space of structures and the space of module assignments. We use a similar approach to that described in Chapter 4 and simplify our task by using an iterative approach that repeats two steps: In one step, we optimize the dependency structures of all module networks relative to our current assignment functions, and in the other, we optimize the assignment functions of all module networks relative to our current dependency structures.

The first type of step in our iterative algorithm learns the structure of each module network, $\mathcal{M}_i.\mathcal{S}$, assuming that the module assignments of each module network, $\mathcal{M}_i.\mathcal{A}$, are fixed. This step involves a search over the space of dependency structures, attempting to maximize the score defined in Equation 5.2.

We first consider optimizing the dependency structure of each module network separately, using the algorithm described in Section 4.3.3. There, we used a standard heuristic search over the combinatorial space of dependency structures. We defined a search space, where each state in the space is a legal parent structure, and a set of operators that take us from one state to another. We traversed this space looking for

high scoring structures using the greedy hill climbing search algorithm.

Our choice of local search operators included steps for adding or removing a variable $\mathcal{M}_i.X_k$ from a parent set $\mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j}$. When an operator caused a parent $\mathcal{M}_i.X_j$ to be added to the parent set of module $\mathcal{M}_i.\mathbf{M}_j$, we had to verify that the resulting module graph remains acyclic, relative to the current assignment $\mathcal{M}_i.\mathcal{A}$. As described in Section 4.3.3, this step is quite efficient, as cyclicity is tested on the module graph, which contains only K nodes, rather than on the dependency graph of the ground Bayesian network, which contains n_i nodes (usually $n_i \gg K$).

However, recall that our structure prior, $P(\mathcal{S})$, defined in Equation 5.4, decomposes by modules and pairs of organisms, such that a bonus of e^{γ} is given for each pair of orthologous genes and module j where both genes from the orthologous pair are parents of module j in the respective organisms. Thus, if we learn the dependency structure of each module network independently using the local search operators described above, we ignore this aspect of the structure prior during the search. Any bonuses we may get for adding a shared regulator for corresponding modules in different organisms are thus accidental.

To take this structure prior into account, we modify the above search algorithm. Rather than searching for dependency structures independently in each module network, we interleave the search steps for the different module networks. We construct one search space that includes the local search operators for all module networks. At each step, we apply the highest scoring step which modifies the structure of only one of the module networks. Upon applying a step that adds (or removes) a variable $\mathcal{M}_i.X_k$ to the parent set of $\mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j}$, we add (or subtract) a bonus of γ to the log-score of the search steps that add (or remove) the ortholog variables of $\mathcal{M}_i.X_k$ according to the orthology map $\mathcal{O}$ in all module networks other than $\mathcal{M}_i$.

Note that, as in module networks, the decomposition of the score provides considerable computational savings. When updating the dependency structure for a module $\mathcal{M}_i.\mathbf{M}_j$, the module score for another module $\mathcal{M}_i.\mathbf{M}_l$ does not change, nor do the changes in score induced by various operators applied to the dependency structure of $\mathcal{M}_i.\mathbf{M}_l$. Hence, after applying an operator to $\mathbf{Pa}_{\mathcal{M}_i.\mathbf{M}_j}$ that involves a variable $\mathcal{M}_i.X_k$, we need only update the change in score for those operators that involve

$\mathcal{M}_i.\mathbf{M}_j$. Note, however, that we do need to modify the operators that involve module j and the orthologs of $\mathcal{M}_i.X_k$ in all other networks, due to the form of the structure prior, although this modification is limited as the likelihood component of the score does not change.

As described in Section 4.3.6, in the gene expression domain, we represent the regulation program of each module using a conditional probability model represented as a *regression tree* (Breiman *et al.*, 1984). When performing structure search for module networks with regression-tree CPDs, we learned the associated tree structures using operators that split leaves. We use the same search strategy for parallel module networks, together with the modifications described above that take the form of the structure prior into account.

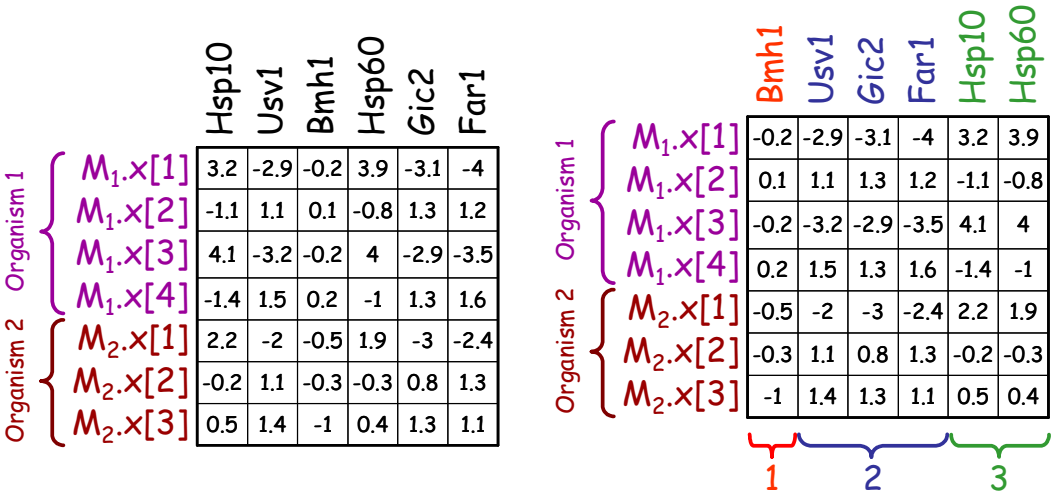
5.3.4 Module Assignment Search Step

The second type of step in our iteration learns an assignment function $\mathcal{M}_i.\mathcal{A}$ from data for each module network $\mathcal{M}_i$. This type of step occurs in two places in our algorithm: once at the very beginning of the algorithm, in order to initialize the modules; and once at each iteration, given the module network structure $\mathcal{M}_i.\mathcal{S}$ for each module network $\mathcal{M}_i$ learned in the previous structure learning step.

Module Assignment Initialization

In Section 4.3.4, we showed how the task of finding an assignment of genes to modules can be viewed as a clustering task: Given arrays (instances) that specify a value for all genes (variables), our goal is to cluster genes that have similar expression levels across the different arrays. We used this clustering idea to initialize the assignment of genes to modules: After applying the clustering algorithm, genes that ended up in the same cluster were initially assigned to the same module. We note that we can also view this clustering task from a probabilistic perspective (see Section 4.3.4).

We use the clustering idea to initialize the module assignments of a parallel module network. However, rather than applying a clustering algorithm separately to the expression data of each organism, we jointly cluster the expression data of all organisms



(a) Data (b) Initialization

Figure 5.2: Example of the joint clustering procedure for initializing a parallel module network of two organisms. In this example, each organism has six genes that are each orthologous to exactly one gene in the other organism (e.g., both organisms have the gene *Hsp10*). The rows correspond to arrays (instances) and the columns correspond to genes (variables). The first organism has four arrays while the second has three. (a) Shown is the expanded expression dataset, in which the data for each gene consists of its data from each of the organisms. (b) Initialization of the assignment function for the parallel module network procedure based on a joint clustering of the data in (a). Note that genes were swapped in their location to reflect the initial assignment into three modules. The initial assignment assigns orthologous genes to the same module in different organisms.

and find the module assignment of all ℓ module networks in one application of the clustering algorithm. For simplicity, assume that each organism has exactly n genes, and that the i-th gene in every pair of organisms are orthologous. Under this assumption, we can create an expanded expression dataset that consists of n genes, where the data for each gene consists of its data from all ℓ organisms. We then use the probabilistic hierarchical clustering algorithm described in Section 4.3.4 to cluster the genes in this expanded dataset into K clusters, and assign the i-th gene to the same module in all organisms based on the cluster it ended up in. That is, if the clustering algorithm assigned the i-th gene to cluster j, then we set $\mathcal{M}_1.\mathcal{A}(\mathcal{M}_1.X_i) = \ldots \mathcal{M}_\ell.\mathcal{A}(\mathcal{M}_\ell.X_i) = j$. An example of this joint clustering procedure is shown in Figure 5.2.

Note that this joint clustering results in a module assignment in which orthologous genes are assigned to the same module in different organisms. Thus, similar to our modified module assignment prior, this initialization point also biases the model to ones in which orthologs are assigned to the same module. However, based on the expression data, the final assignments can certainly change after applying the iterative learning algorithm.

In the general case, where the orthology map is many-to-many and some organisms may be missing some genes, we define an *orthologous group* to be a set of genes that form a maximal connected component in the orthology map $\mathcal{O}$. If a connected component consists of more than one gene from a single organism, then we initialize the clustering so that such genes are assigned to the same cluster. After such an initialization, we continue with the clustering algorithm as described above. Note that this algorithm results in a clustering where genes in the same orthologous group are assigned to the same module in different organisms.

Module Reassignment

In the module reassignment step, the task is more complex. We now have the vector of structures for each module network, $\mathcal{S}$, and wish to find the vector of assignments to all module networks, $\mathcal{A}$, such that $\mathcal{A} = \text{argmax}_{\mathcal{A}'}\text{score}_{\mathcal{G}}(\mathcal{S}, \mathcal{A}' : \mathcal{D})$. We thus wish to select the optimal assignment $\mathcal{M}_i.\mathcal{A}(X)$ for each gene X in each module network $\mathcal{M}_i$. However, as we showed in Section 4.3.4, the score does not decompose by genes, and we thus can not determine the optimal assignment $\mathcal{M}_i.\mathcal{A}(X)$ for each gene X independently. The same is true in the case of a parallel module network.

We therefore follow our approach from Section 4.3.4 and use a sequential update algorithm that reassigns the genes in each orthologous group to modules one by one (recall that a set of genes is called an orthologous group if it forms a maximal connected component in the orthology map $\mathcal{O}$). The idea is simple. We start with an initial assignment function $\mathcal{A}^0$, and in a "round-robin" fashion iterate over all of the orthologous groups one at a time, and consider changing the module assignment of their member genes in all ℓ module networks. When considering a reassignment for the i-th orthologous group of genes in all module networks, we keep the assignments

of all other genes fixed and find the optimal legal (acyclic) assignment for the genes in the i-th orthologous group in each module network relative to the fixed assignment. We continue reassigning variables until no single reassignment can improve the score.

Note, however, that due to the form of the module assignment prior (see Equation 5.3), we cannot find the jointly optimal assignment for all the genes in an orthologous group by searching for the optimal assignment of each gene in the group independently in each module network. The reason is that the module assignment prior assigns a bonus of e^{λ} each time we assign an orthologous pair of genes to the same module in their respective module networks. Thus, finding the assignment in each module network independently might result in sub-optimal joint assignments, if the bonus we get by assigning an orthologous pair to the same module is greater than the loss we incur by setting the orthologous pair to a sub-optimal module relative to their best assignment when each module network is considered in isolation.

In a typical application of the parallel module network procedure the number of organisms and the number of genes in each orthology group is relatively small (e.g., the application we present in Section 5.4 is for two organisms, human and mouse, and the orthology map is one-to-one so each orthologous group consists of exactly two genes). In such settings we can find the jointly optimal assignment for a set of genes $X_1, \ldots, X_k$ in an orthology group of size k, by first enumerating over all of the possible ways to partition the orthologs $X_1, \ldots, X_k$ into nonempty subsets. For example, for an orthology group of two genes there are two possible partitions, one where the two genes are in the same subset and another where each of the two genes is in its own subset. For each such partition, we then require that genes in the same subset are assigned to the same module — the one that gives the highest score for all genes in the subset under this constraint. We can find this optimal module assignment for each subset of size n by computing:

$$m^* = \text{argmax}_{m=1}^{K} \left[\lambda N(\mathcal{O}, n) + \sum_{i=1}^{n} \text{score}_{\mathcal{M}[X_i]}(\mathcal{M}[X_i].\mathcal{S}, \mathcal{M}[X_i].\mathcal{A}[\mathcal{M}[X_i].X_i, m] \ : \ \mathcal{D}) \right] \tag{5.6}$$

where $\mathcal{M}[X_i]$ denotes the module network that the orthologous gene X_i belongs to, $\mathcal{M}[X_i].\mathcal{A}[\mathcal{M}[X_i].X_i, m]$ represents the current assignment function for the module

network of X_i, $\mathcal{M}[X_i].\mathcal{A}$, except that we changed the assignment of $\mathcal{M}[X_i].X_i$ to module m (i.e., we set $\mathcal{M}[X_i].\mathcal{A}(X_i) = m$) and $N(\mathcal{O}, n)$ denotes the number of edges in $\mathcal{O}$ for the subset of size n:

$$N(\mathcal{O}, n) = \sum_{i_1=1}^{n} \sum_{i_2=1_1+1}^{n} \eta\{(X_{i_1}, X_{i_2}) \in \mathcal{O}\}$$

where $\eta\{(X_{i_1}, X_{i_2}) \in \mathcal{O}\}$ is an indicator function that is equal to 1 if and only if X_{i_1} and X_{i_2} are orthologs according to the orthology map $\mathcal{O}$.

Thus, under the constraint that genes in the same subset of size n are assigned to the same module we can find the optimal module assignment for the subset by performing $n \cdot K$ score computations and the optimal assignment for the entire partition by performing $k \cdot K$ score computations. We note that this is actually a lower bound on the optimal assignment for the entire partition as we are not taking into account additional assignment bonuses that may be possible to get if two or more subsets actually assign their genes to the same module. Such cases, however, will be considered as we iterate over all possible partitions into subsets and require that genes in the same subset are assigned to the same module. After iterating over all partitions of $X_1, \ldots, X_k$ into nonempty subsets, we select the joint assignment with the highest score across all partitions.

The number of ways that a set of k elements can be partitioned into nonempty subsets is called a *Bell number* (Bell, 1934), where the first few Bell numbers for $k = 1, 2, \ldots, 10$ are $1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975$. Thus, the above algorithm is feasible for parallel module networks with up to ten or so genes in the same orthology group.

In order to deal with the unlikely scenario in which more than ten genes are assigned to the same orthology group, we note that we can cast the problem of finding the optimal module assignment for genes in the same orthology group across all module networks as an instance of the well studied *minimum multiway cut* problem. In minimum multiway cut, we are given a graph $G = (V, E)$, a subset $S \subseteq V$ of terminal nodes, and a weight function $w : E \rightarrow I\!R$. The problem is then to find a minimum multiway cut, i.e., a set $E' \subseteq E$ such that the removal of E' from E

disconnects each terminal from all the others and the weight of the cut, $\sum_{e \in E'} w(e)$, is minimal. Although minimum multiway cut was shown by Dahlhaus *et al.* (1994) to be NP-hard, there is an algorithm, due to Calinescu *et al.* (1998), which approximates the optimal solution within $3/2 - 1/|S|$ and runs in polynomial time.

We can easily cast our problem as an instance of multiway cut by creating a graph with K terminal nodes, one for each of the K modules $\mathbf{M}_1, \ldots, \mathbf{M}_K$, and k additional nodes for each of the genes $X_1, \ldots, X_k$ in the orthologous group. For a terminal node that represents module j, $\mathbf{M}_j$, and a gene node X_i, the weight function would correspond to the score of the module network $\mathcal{M}[X_i]$ that gene X_i belongs to when assigning X_i to module j, $\text{score}_{\mathcal{M}[X_i]}(\mathcal{M}[X_i].\mathcal{S}, \mathcal{M}[X_i].\mathcal{A}[\mathcal{M}[X_i].X_i, j] \ : \ \mathcal{D})$. For two gene nodes X_i and X_l that are orthologs according to the orthology map $\mathcal{O}$ (i.e., $(X_i, X_l) \in \mathcal{O}$), the weight function would be equal to the module assignment bonus λ (i.e., $w(\mathcal{M}[X_i].X_i, \mathcal{M}[X_k].X_k) = \lambda$). It is easy to show that finding the minimum multiway cut in this problem corresponds to the jointly optimal assignment of the genes in the orthology group across all ℓ module networks. Thus, when the number of genes in an orthology group is large, we can use the approximation algorithms developed for minimum multiway cut to find an assignment which is within $3/2 - 1/K$ to the optimal one.

The key to the correctness of this algorithm is its sequential nature: Each time genes in the same orthologous group change their assignment, the assignment function as well as the associated sufficient statistics are updated before evaluating another orthologous group. Thus, each change made to the assignment function leads to a legal assignment which improves the score. Our algorithm terminates when it can no longer improve the score. Hence, it converges to a local maximum, in the sense that no single assignment change can improve the score.

The computation of the score is the most expensive step in the sequential algorithm. As described in Section 4.3.4, the decomposition of the score (see Theorem 5.3.1) plays a key role in reducing the complexity of the computation as we only need to rescore the module from which we reassign the gene and the module to which we reassign the gene.

Input:
D // Data set
K // Number of modules
ℓ // Number of organisms
Output:
M // A module network
Learn-Module-Network
$\mathcal{A}_0$ = jointly cluster $\mathcal{M}_1.\mathcal{X}, \ldots, \mathcal{M}_\ell.\mathcal{X}$ into K modules
$\mathcal{S}_0$ = empty structure
Loop $t = 1, 2, \ldots$ until convergence
$\mathcal{S}_t$ = Greedy-Structure-Search$(\mathcal{A}_{t-1}, \mathcal{S}_{t-1})$
$\mathcal{A}_t$ = Sequential-Update$(\mathcal{A}_{t-1}, \mathcal{S}_t)$;
Return M $= (\mathcal{A}_t, \mathcal{S}_t)$

Figure 5.3: Outline of the *parallel module network* learning algorithm. Greedy-Structure-Search successively applies operators that change the structure as long as each such operator results in a legal structure and improves the module network score

5.3.5 Algorithm Summary

To summarize, our algorithm starts with an initial assignment of genes to modules in all ℓ module networks. In general, this initial assignment can come from anywhere, and may even be a random guess. We choose to construct it using the joint clustering idea described in the previous section. The algorithm then iteratively applies the two steps described above: learning the module dependency structures of all module networks, and reassigning genes to modules. These two steps are repeated until convergence. An outline of the module network learning algorithm is shown in Figure 5.3.

Each of these two steps — structure update and assignment update — is guaranteed to either improve the score or leave it unchanged. We can thus prove:

Theorem 5.3.2: The iterative module network learning algorithm converges to a local maximum of score$(\mathcal{S}, \mathcal{A} : \mathcal{D})$.

5.4 Application to Brain Tumor Expression Data

We evaluated our method on an expression dataset from human (Pomeroy *et al.*, 2002) and an expression dataset from mouse (Lee *et al.*, 2003b), both measuring the expression of various brain tumors. In human, the data consisted of a total of 90 arrays from various brain tumors: 60 arrays from medulloblastoma, the most common malignant brain tumor of childhood; 10 arrays from malignant gliomas; 10 arrays from atypical teratoid/rhabdoid tumors (AT/RT); 6 arrays from primitive neuroectodernal tumors (PNETs); and 4 arrays from normal cerebella. In mouse, the data consisted of a total of 43 arrays: 23 arrays from medulloblastoma and 20 arrays from normal cerebella.

Our goal is to discover the regulatory modules underlying these malignancies and gain insights into the conservation of regulatory relationships between these tumors in human and mouse. In addition, we test our hypothesis that we can learn better models of gene regulation by combining expression datasets across organisms.

In the application of module networks to yeast (see Section 4.6) we restricted the set of potential parents to those genes for which we had prior knowledge that they are likely to play a regulatory role. We used a similar idea here: We compiled a set of 1537 candidate regulators whose GO (Ashburner *et al.*, 2000) annotations in human or mouse suggest a potential regulatory role in the broad sense: both transcription factors and signaling proteins that may have transcriptional impact. We excluded global regulators, whose regulation is not specific to a small set of genes or processes. Of these 1537 candidate regulators, 604 were both in our orthology map and measured in the microarrays of both human and mouse. Subsequently, we restricted the parent set in both human and mouse to these 604 candidate regulators

Recall that in addition to the expression datasets, our method takes as input an orthology map $\mathcal{O}$ between the organisms. We thus start this section by describing the construction of an orthology map between human and mouse. We then evaluate the results of applying our method to the two expression datasets described above, both from a statistical and a biological standpoint.

5.4.1 Constructing a Human-Mouse Orthology Map

As described above, the input to a parallel module network includes an orthology map between organisms. For simplicity, in our application to human and mouse we assume that this orthology map is one-to-one. This means that for every gene we wish to include from human, we need to know which gene performs a similar function to it in mouse. As the function of many genes is not known, a full human-mouse orthology map is also not known. In fact, obtaining such an orthology map is an area of active research.

As a gene's function is dictated by its protein sequence, a standard computational approach for constructing a proxy to this orthology map (Tatusov *et al.*, 2001) is to base the map on a comparison of the protein sequences of the organisms in question. To this end, we first obtained the $33,367$ human protein sequences (version hg16) and the $30,988$ mouse protein sequences (version mm4) from the UCSC genome browser (Kent *et al.*, 2002). For each gene in human, we used the standard BLAST algorithm (Basic Local Alignment Search Tool (Altschul *et al.*, 1990)) to compute its e-value to each of the proteins in mouse. For a pair of proteins, the BLAST e-value measures the probability of obtaining their observed sequence similarity by chance. Thus, lower e-values correspond to higher similarity among a pair of protein sequences. As the BLAST score is not symmetric, we performed a similar computation for each mouse gene. We ignored e-values higher than 10^{-5}. We used these e-values to construct a one-to-one orthology map between human and mouse. To make the map reliable, we defined a gene g_h from human as an ortholog of a gene g_m from mouse if and only if g_m was the best BLAST hit (lowest e-value) for g_h out of all mouse genes and g_h was the best BLAST hit for g_m out of all human genes (Tatusov *et al.*, 2001).

Using the above procedure, we created $11,983$ orthologous gene pairs between human and mouse. For $3,718$ of these orthologs we also had expression measurements in both the human and the mouse expression datasets (the loss is due to incompleteness of both the mouse and the human microarray chips). We thus used these $3,718$ orthologs for our subsequent analyses.

Examining the functions represented by the selected orthologs according to the gene annotation database of Gene Ontology (GO (Ashburner *et al.*, 2000)), we found

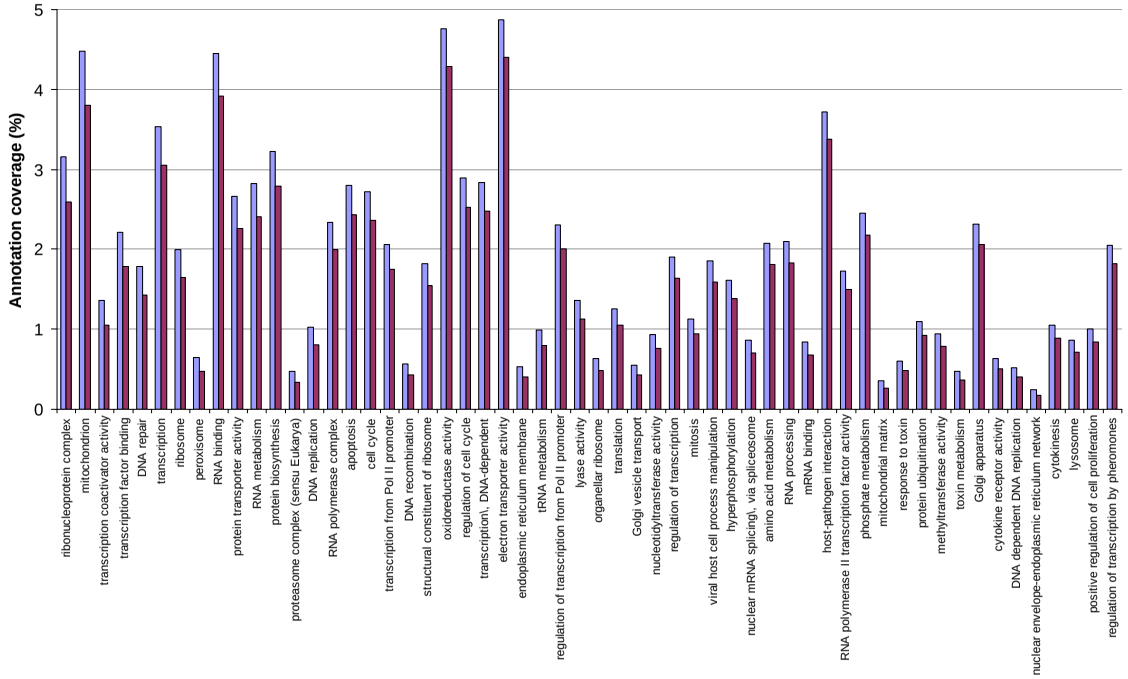


Figure 5.4: Shown are the categories from Gene Ontology (GO) that are enriched in the orthologous genes between human and mouse. Categories are sorted according to their enrichment p-value . Only categories with a p-value ≤ 0.05 (Bonferroni corrected) are displayed. For each category, shown is the percentage of genes associated with it from all orthologous genes (left column; light blue) and from all human genes (right column; purple).

that $3,875$ of the $11,983$ orthologous genes have no GO classification. Thus, although we used a stringent criterion for constructing the orthology map, it contains many genes that are as of yet uncharacterized. We note, however, that of the $3,718$ orthologs for which we also had expression data, only 391 have no GO classification. This reduction in the percentage of uncharacterized genes is largely due to the fact that the set of genes that were selected to be printed on the microarrays were biased towards selecting characterized genes.

We next tested whether certain functional categories are over-represented in the set of orthologous genes, by measuring the enrichment of the orthologs for genes in the same GO functional category. We used the hypergeometric distribution to compute a

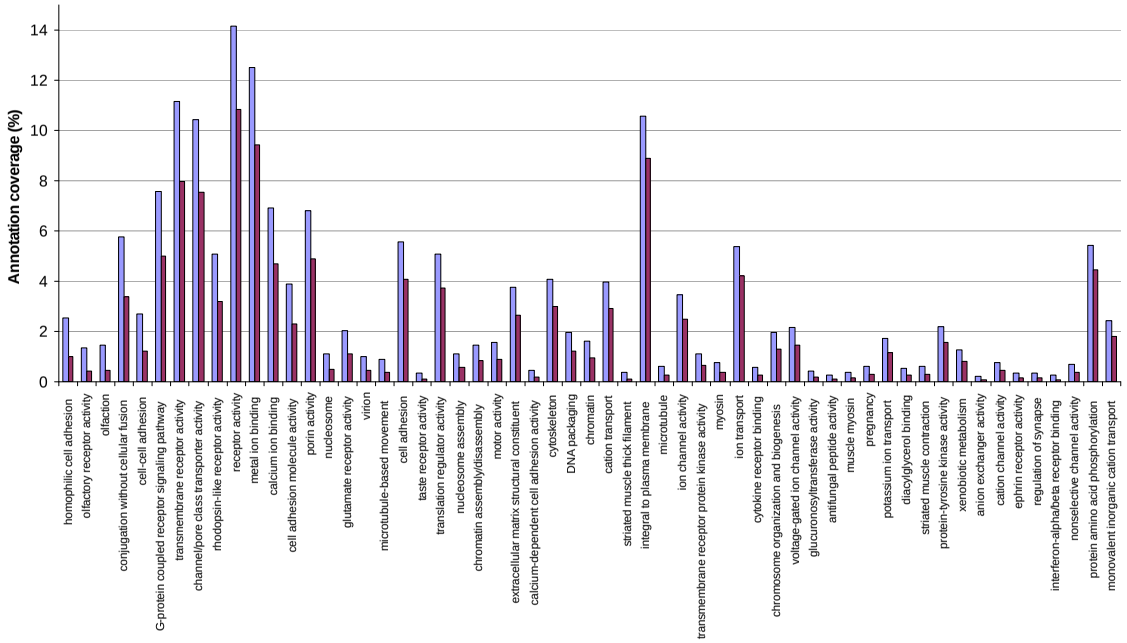


Figure 5.5: Shown are the categories from Gene Ontology (GO) that are enriched in the non-orthologous genes between human and mouse. Categories are sorted according to their enrichment p-value . Only categories with a p-value ≤ 0.05 (Bonferroni corrected) are displayed. For each category, shown is the percentage of genes associated with it from all non-orthologous genes (left column; light blue) and from all human genes (right column; purple).

p-value for this enrichment (see Appendix C). The results, summarized in Figure 5.4, show 56 categories that are over-represented in the set of orthologs. As expected, the orthologs are enriched for functional categories that represent basic cellular processes such as transcription, translation, DNA and RNA binding, cell cycle, apoptosis, RNA processing, and metabolism.

We also tested which functional groups are over-represented in the set of genes for which we did not find orthologs. We used the same statistical test described above to compute p-values for this enrichment. The results, summarized in Figure 5.5, show 57 categories that are over-represented in the set of non-orthologous genes. Examining these missed categories, we found that they contained gene families that evolved differently in human and mouse (e.g., olfactory receptors (Young *et al.*, 2002)) and

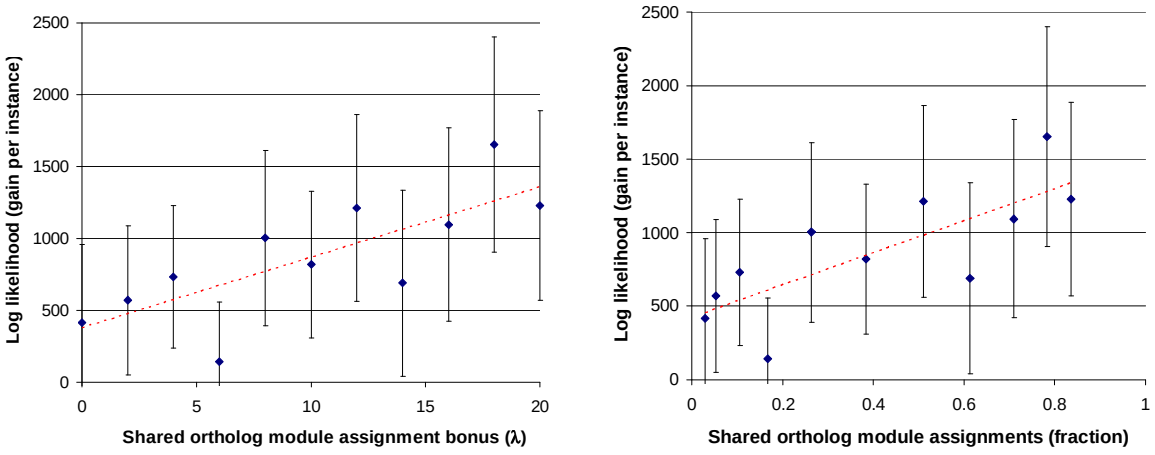
thus contain human-specific and mouse-specific genes that do not have orthologs. However, the missed genes were also enriched for large gene families whose genes are very similar to each other and differ only in a small number of amino acids (e.g., the family of G-protein coupled receptors). These cases point to limitations of our procedure for constructing orthologs, as in such gene families it is often difficult to determine a one-to-one orthology mapping between genes based on the protein sequence alone. We return to the orthology mapping issue at the end of this chapter.

5.4.2 Comparison to Module Networks

We first evaluated whether there is a gain from learning a joint model over both the human and mouse brain tumor expression datasets using the parallel module network procedure described above, as compared to learning separate models on each dataset using the module network procedure described in Chapter 4. Recall that the probabilistic model of a parallel module network is composed of ℓ independent module network models (in our case $\ell = 2$). However, during the learning process, we maximize a score that favors models in which orthologs are assigned to the same module and corresponding modules in different organisms have common regulator parents. We encode this preference by two parameters: λ, which parameterizes the prior over module assignments $P(\mathcal{A})$ (see Equation 5.3) and assigns a bonus of λ to each orthologous pair of genes assigned to the same module; and γ, which parameterizes the prior over network structures $P(\mathcal{S})$ (see Equation 5.4) and assigns a bonus of γ for each module j and orthologous pair of regulators where the orthologous pair are both parents of module j.

Thus, for $\lambda = 0$ and $\gamma = 0$, the parallel module network learning procedure is equivalent to applying the module network learning procedure separately to each of the input expression datasets (except for the initial module assignment which in the case of parallel module networks is obtained through a joint clustering of the input expression datasets). Moreover, by increasing λ and γ we can increase the degree of shared module assignments and regulatory parents, respectively.

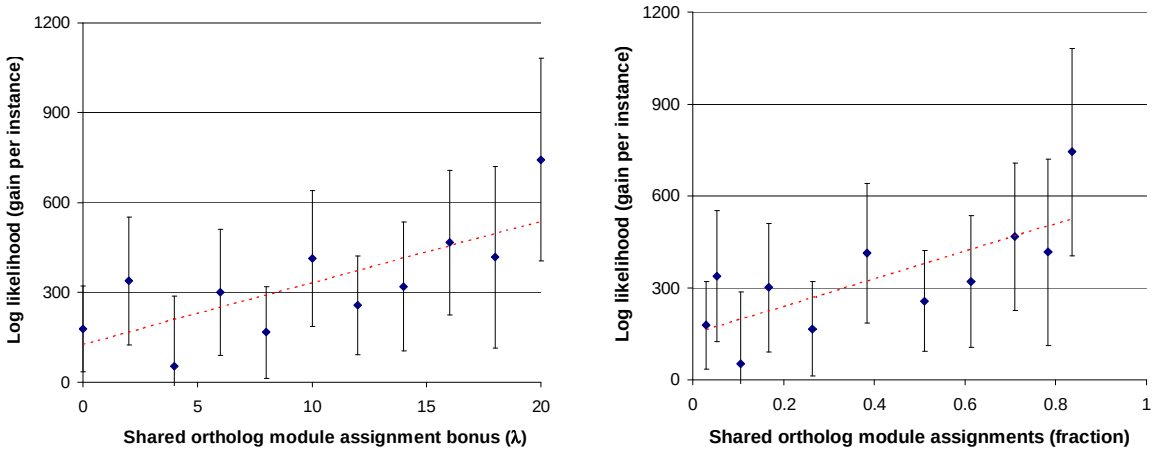
We evaluated the generalization ability of models learned with different settings



(a) (b)

Figure 5.6: Results for human. (a) Shown is the generalization ability of models learned with different settings of λ and with $\gamma = 20$, in terms of log-likelihood of test data, using 5-fold cross validation. The results are shown for the module network over the human expression data, and are displayed as the log-likelihood gain per instance (array). (b) Same as (a), but the x-axis displays the fraction of orthologs that are assigned to the same module in human and mouse.

of λ and γ, in terms of log-likelihood of test data, using 5-fold cross validation. In all cases, we applied the learning procedure described in Section 5.3 with 100 modules (i.e., $K = 100$). As a baseline, we used the models learned when both λ and γ are set to zero. We then learned different models while varying λ between 0 and 20 and varying γ between 0 and 50 (these upper ranges were chosen as they resulted in maximal sharing of module assignments and regulator parents). In Figure 5.6, we show the results for the range of values of λ when setting γ to 20 (qualitatively similar results were obtained for $\gamma = 50$). As can be seen, all modules learned performed better than the baseline in both human and mouse. For most models the gain was also statistically significant. Moreover, there was a strong positive correlation (0.62 in human and 0.8 in mouse) between the fraction of human-mouse orthologs assigned to the same module and the generalization performance on test data. This indicates that, in general, the stronger our bias is to models in which orthologs are assigned to the same module, the larger the gain in performance. Overall, these results indicate



(a) (b)

Figure 5.7: Results for mouse. (a) Shown is the generalization ability of models learned with different settings of λ and with $\gamma = 20$, in terms of log-likelihood of test data, using 5-fold cross validation. The results are shown for the module network over the mouse expression data, and are displayed as the log-likelihood gain per instance (array). (b) Same as (a), but the x-axis displays the fraction of orthologs that are assigned to the same module in human and mouse.

that, by biasing the learned models towards ones in which orthologs are assigned to the same module and corresponding modules share parent regulators, we can learn better regulatory modules in both human and mouse.

Recall that we initialize the assignments of genes to modules from a joint clustering of both input expression datasets. Thus, by comparing the learned models above to a baseline in which both λ and γ are set to zero, we showed that our learning procedure improves significantly over module networks that are learned independently (since $\lambda = \gamma = 0$) but initialized from a joint clustering algorithm. As another baseline, we also compared the learned models to module networks that are both learned and initialized independently from each expression data. To this end, we applied the module network procedure as described in Chapter 4 with 100 modules. In mouse, this new baseline was nearly identical to the previous baseline (the gain per instance was 35.1). In human, the new baseline outperformed the previous baseline (the gain per instance was 346.1) but was still worse than all of the models presented in Figure 5.6.

Thus, our main result, that we can learn better regulatory modules by combining brain tumor expression datasets from human and mouse into a unified probabilistic model also holds in comparison to a pure module network baseline.

A notable difference between the module network procedure and the parallel module network procedure is that a parallel module network is learned from a larger set of input expression data. To verify that the gain in performance is indeed due to utilizing evolutionary conservation and is not merely a consequence of learning from more data, we performed two additional tests. In one test, we permuted the mouse expression data matrix (by swapping every entry in the matrix with another randomly selected entry) and in the other, we left both expression datasets intact, but permuted the human-mouse orthology mapping. In both tests, we applied the parallel module network procedure with 100 modules. These results test, in two different ways, whether we can get a gain in performance even by adding random data. When permuting the mouse expression data, we got essentially identical results (the gain per instance in human was -22.1) to the baseline model which sets both λ and γ to zero and is thus effectively learning independent module networks. The same was true when permuting the orthology mapping (the gain per instance was -311.6 in human and -32.6 in mouse). These results show that merely adding expression data from another organism is not enough for learning better models — we need to add such data in a meaningful way through the orthology map.

In summary, our results show that by combining brain tumor expression datasets from human and mouse, we can learn better regulatory modules compared to modules learned using each expression dataset in isolation. Moreover, this gain in performance can be attributed in part to the evolutionary conservation of regulatory relationships between human and mouse in these tumors.

5.4.3 Biological Evaluation

We now turn to a detailed biological evaluation of the results when learning a parallel module network with 100 modules ($\lambda = 20$ and $\gamma = 20$; these settings were selected based on the generalization tests on held out data) over the brain tumor expression

data from human and mouse. To simplify the analysis, we excluded the 604 regulators from this run and learned the network only over the 3,114 orthologs that were not defined as regulators and for which we had expression data in both organisms. I.e., we assigned each of the 604 regulator to its own module and did not change this assignment during the learning process; regulators were only allowed to be parents of other modules. As in the biological evaluation of our results on yeast (see Section 4.6 and Section 3.5), we evaluate the inferred models relative to the literature and to databases of functional annotations of genes. However, as much less is known about human and mouse, we cannot expect the evaluation to be exhaustive and comprehensive as was it was in yeast. Thus, many of our results remain as novel computational predictions.

Annotation Databases

We evaluated the results relative to gene and array annotation that we collected from four different sources. For gene annotations, we compiled annotations from: the GO database of functional annotations (Ashburner *et al.*, 2000); the PROSITE database of protein families and domains (Bairoch *et al.*, 1997); and the dataset of Su *et al.* (2002) that measured the expression of a variety of human and mouse tissues, organs and cell lines. We ensured that each pair of orthologs have the same gene annotations by completing missing annotations for a gene in one organism using the annotations for its ortholog in the other organism. Thus, for deriving GO annotations, we associated each gene with an annotation if, according to the GO database, the gene was associated with the annotation in either human or mouse (or both). We removed annotations with less than 5 genes, resulting in 1195 GO annotations. We processed the PROSITE database in a similar way, resulting in 490 protein domain annotations. From the dataset of Su *et al.* (2002) we defined one annotation for each of the 101 arrays in this study by taking all genes whose absolute expression was above 400. In order for the annotations to correspond to expression in specific tissue types, we removed genes whose absolute expression was above 400 in more than 50 of the 101 arrays from the dataset.

For array annotations, we associated each array with the annotations it represents, from a total of 27 annotations that we compiled from the published studies (21 annotations for the human arrays of Pomeroy *et al.* (2002) and 6 annotations for the mouse arrays of Lee *et al.* (2003b)). These condition annotations represent information available about the samples, such as tissue sub-type, tumor sub-type, tumor stage, genetic background, clinical treatment given to the patient, and clinical outcome. Note that, in contrast to the gene annotations, the array annotations are defined separately for every organism, as there is no analogous correspondence between arrays as there is for genes (using the orthology mapping). Nevertheless, two annotations, 'Medulloblastoma' and 'Normal cerebellum', were defined in both studies.

Global Conservation Properties of Regulatory Modules

Overall, we learned 100 modules in human and 100 modules in mouse over the expression data of the 3,114 orthologous genes that were not classified as potential regulators and for which we had expression data in both organisms. Of these 3,114 orthologous gene pairs, 2,276 pairs were assigned to the same module in the human and the mouse module network models, representing a 73% level of conservation between the gene assignments across organisms. Moreover, this gene assignment conservation was spread across a large number of modules, with 76 and 77 modules exhibiting a conservation level of 50% or higher in the human and mouse module networks, respectively (see Figure 5.8). Thus, our model predicts that a substantial number of the co-regulation relationships among genes have been conserved between human and mouse.

Examining the learned regulation programs for all modules, we found that the total number of regulators that appeared in at least one regulation program was 212 in the human module network and 199 in the mouse module network. Of these, 108 regulators appeared in the regulation programs of both human and mouse, where 86 of these 108 regulators appeared in the regulation programs of corresponding modules (i.e., for 86 regulators, there was at least one module j such that the regulator appeared in the regulation program of module j in both the human and the mouse module networks). For 30 of these 86 shared regulators, all of their appearances in

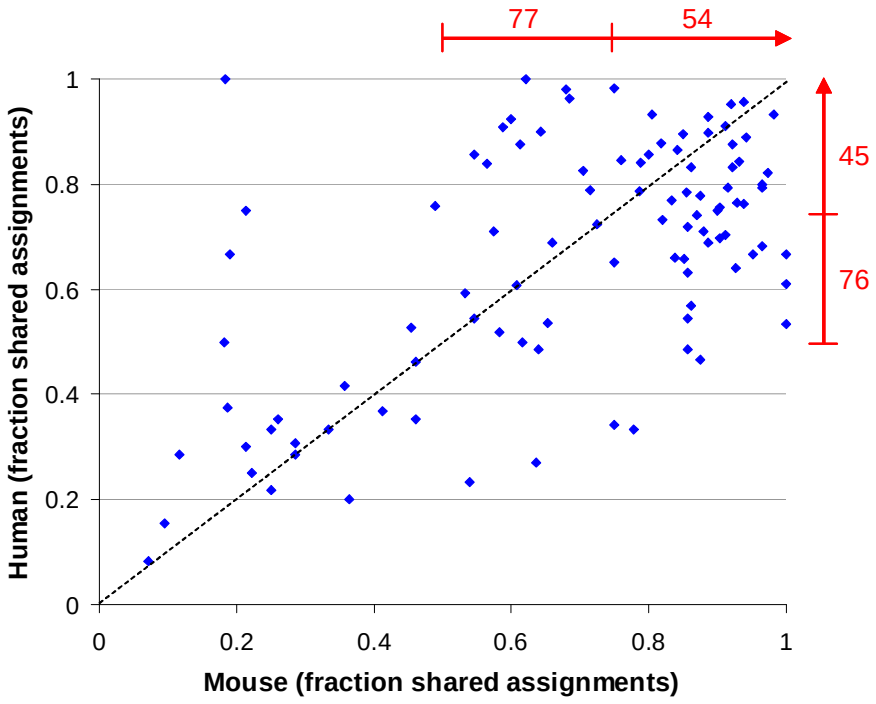


Figure 5.8: Distribution of shared module assignments of orthologs across modules. Each module j is indicated by a blue dot, representing the fraction of genes that were assigned to module j in both the human and the mouse module networks out of the total number of genes assigned to module j in human (y-axis) and the total number of genes assigned to module j in mouse (x-axis). The red bars indicate the total number of modules whose fraction was above 50% or 75% (e.g., in 76 modules from the human module network, the fraction of shared assignments was 50% or higher).

regulation programs were shared (i.e., for each occurrence of these regulators in the regulation program of module j in one organism, there was a corresponding occurrence in the regulation program of module j in the other organism; see Figure 5.9). At the gene level, the total number of regulator-regulatee relationships was $12,226$ in the human module network and $11,056$ in the mouse module network, where $3,086$ of these regulatory relationships (representing a total of $2,012$ genes) appeared in both the human and the mouse module networks. Thus, our model predicts that a significant number of regulators and regulator-regulatee relationships have been conserved between the regulation programs of human and mouse.

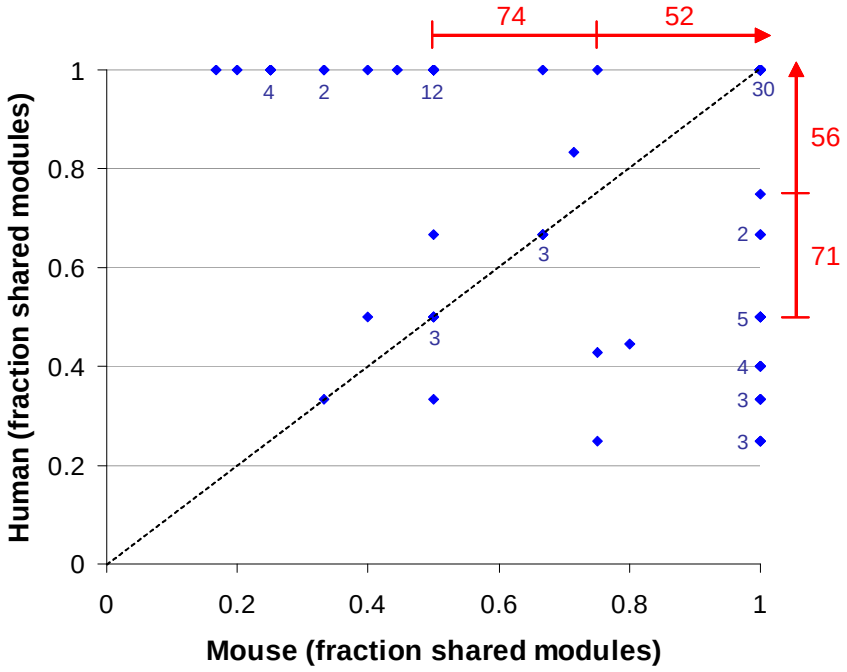


Figure 5.9: Distribution of shared regulators in regulation programs of corresponding modules in human and mouse. Each regulator j is indicated by a blue dot, representing the fraction of modules such that the regulator appeared in their control program in both the human and the mouse module networks out of the total number of regulation programs in which the regulator appeared in human (y-axis) and mouse (x-axis). Only the 86 regulators that were shared in at least one module are shown. The red bars indicate the total number of regulators whose fraction was above 50% or 75% (e.g., in 71 modules from the human module network, the fraction of shared modules was 50% or higher).

Recall that our 604 candidate regulators represent a broad class of proteins that may have a potential regulatory role, consisting of both transcription factors and signaling proteins that may have transcriptional impact. As discussed above, 86 of these 604 regulators appeared in the regulation programs of corresponding modules between human and mouse. We next tested whether these 86 putatively conserved regulators shared any functional properties that distinguish them from the entire set of candidate regulators. To this end, we computed the fraction of conserved regulators that were associated with each of the annotations in the gene annotation databases mentioned

above (GO, PROSITE, and tissue expression dataset), and compared these fractions to those obtained when performing the computation on the entire set of regulators. We used the hypergeometric distribution to obtain a p-value for these fractions and performed a Bonferroni correction for multiple hypotheses (see Appendix C).

We found that the conserved regulators indeed had commonalities beyond what would be expected by randomly selecting 86 regulators from the set of 604 candidate regulator set. These included enrichment for: thermoregulation regulators (the 86 conserved regulators included 4 of the 6 thermoregulation regulators; $p < 0.004$); immune response regulators (e.g., humoral defense mechanism: 5 of 10; $p < 0.007$); nucleic acid binding regulators (7 of 18; $p < 0.007$); phosphatase inhibitor activity (4 of 7; $p < 0.007$); transcription co-repressor activity (9 of 29; $p < 0.01$); regulators expressed in certain tissue types, including heart (11 of 32; $p < 0.003$), brain (10 of 35; $p < 0.02$) and placenta (14 of 58; $p < 0.03$); cyclin regulators, which play an active role in controlling nuclear cell division cycles (4 of the 8 regulators with a cyclin-box domain; $p < 0.02$); and nuclear regulators (27 of 132 with a nuclear localization domain; $p < 0.02$).

We note that while some of these shared properties (e.g., regulators expressed in brain tissues) may be due to our choice of brain tumor expression datasets, others may represent general biological properties of those regulators that have been conserved in the regulatory programs between human and mouse. For example, as our candidate regulators consisted primarily of transcription factors and signaling molecules, the enrichment for nucleic acid binding predicts that transcription factors have been more conserved than signaling molecules in their regulatory role. The enrichment for transcription co-repressor activity predicts that there is a strong selective evolutionary pressure to preserve the role of regulators who act in combination with other regulators to perform their function. A possible explanation for this conservation may have to do with the increased number of interactions that such regulators must have in order to be active, which implies that their protein sequence must obey more constraints (as it needs to preserve both the site that binds the DNA and the site that binds its co-regulators). We note that while the enrichment of other annotations (e.g., heart) is intriguing, we have no satisfactory explanation for their preferential

conservation at this time.

Global View of Module Genes

To obtain a global view of the regulation programs, we identified all the GO and tissue type annotations significantly enriched (p-value ≤ 0.05; Bonferroni corrected) in the genes assigned to each module in both human and mouse and compiled all enriched annotations into the single matrix shown in Figure 5.10. In order to distinguish the conserved annotations from those that are organism specific, we colored in blue the annotations that were enriched only in modules from the human module network, in green the annotations that were enriched only in modules from the mouse module network, and in red the annotations that were enriched in the modules from both organisms.

Overall, we found enrichment for 66 module-GO-annotation pairs (44 human specific, 12 mouse specific, and 10 conserved), spanning 14 modules and 37 annotations (see Figure 5.10). These included enrichment for basic cellular processes such as translation (e.g., of the 16 genes in module 96 in human, 14 were ribosomal genes, out of the 61 ribosomal genes in the dataset; $p < 10^{-22}$), energy related processes (e.g., ATP biosynthesis; $p < 5 \cdot 10^{-7}$), and DNA replication ($p < 5 \cdot 10^{-6}$).

Interestingly, the spliceosome complex, and most (5 of 8) of the energy related annotations were only enriched in corresponding modules in human and mouse. Translation related processes such as protein biosynthesis were enriched in several modules, once in both human and mouse, and in other cases only in human or only in mouse modules. A summary of the enrichments broken down by their conservation properties described above is shown in Figure 5.11.

Among the annotations whose enrichment was conserved, the enrichment for mitochondrial genes (module 50; see Figure 5.12) was particularly intriguing, as these genes were recently reported to play a role in aging in several organisms including mouse (Trifunovic *et al.*, 2004), fly (McCarroll *et al.*, 2004), and worm (McCarroll *et al.*, 2004, Lee *et al.*, 2003a). Moreover, the transcription factor YY1 was predicted as the (activating) regulator in the regulation program of this module in both the human and the mouse module networks. This prediction is consistent with the known

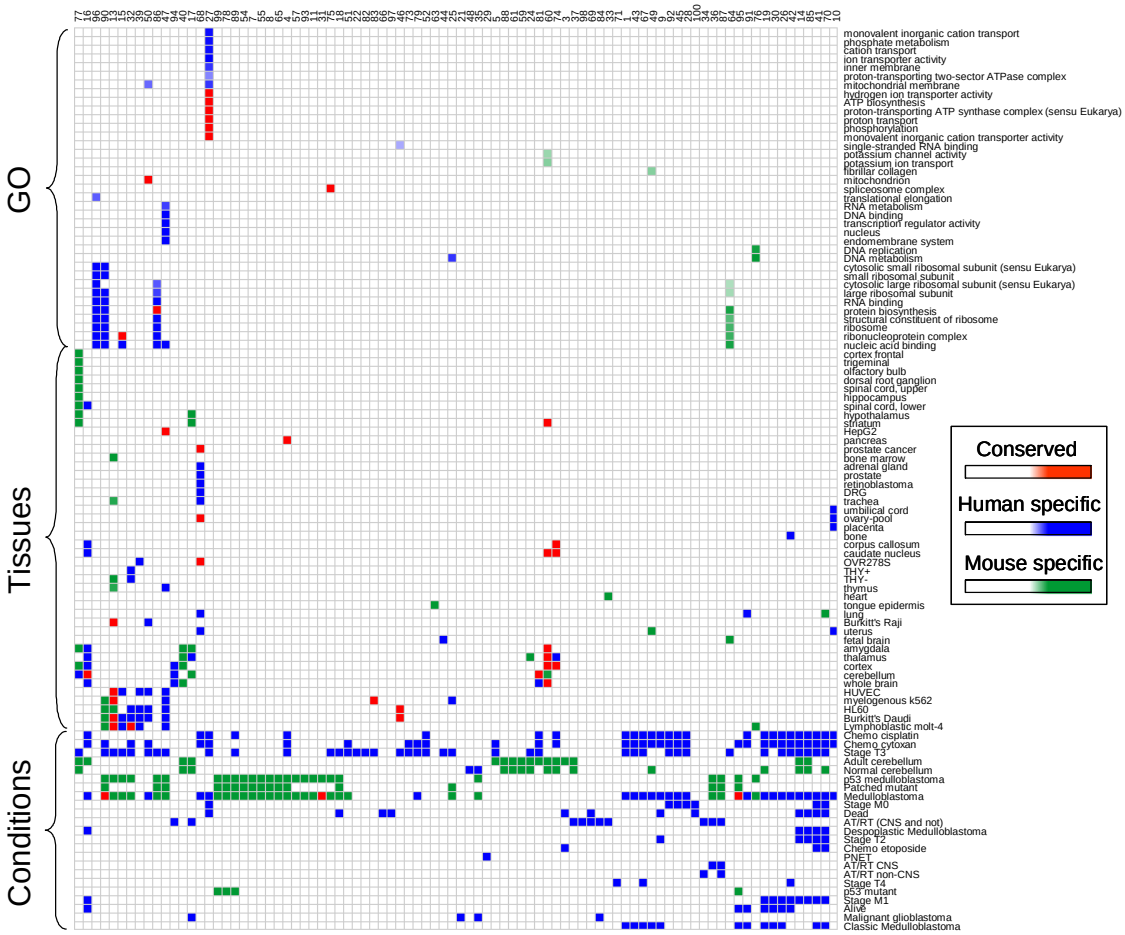


Figure 5.10: Enrichment of gene and array condition annotations in modules. Entries represent the fraction of module genes that are associated with the annotation.

role of YY1 in regulating mitochondrial genes (Seelan and Grossman, 1997, Lescuyer *et al.*, 2002). Indeed, at least 5 of the 39 genes in the human module and 3 of the 38 genes in the mouse module are known targets of YY1. A closer examination of the module in human showed that YY1 was predicted as the regulator primarily in patients with advanced medulloblastoma tumors (stage T3; $p < 0.002$), in which the module genes were repressed. As YY1 itself was also repressed in these tumors, our method suggests that its down-regulation in these tumors may provide a partial explanation for the repression of this module in advanced stages of medulloblastoma tumors. We note that McCarroll *et al.* (2004) found that mitochondrial genes are

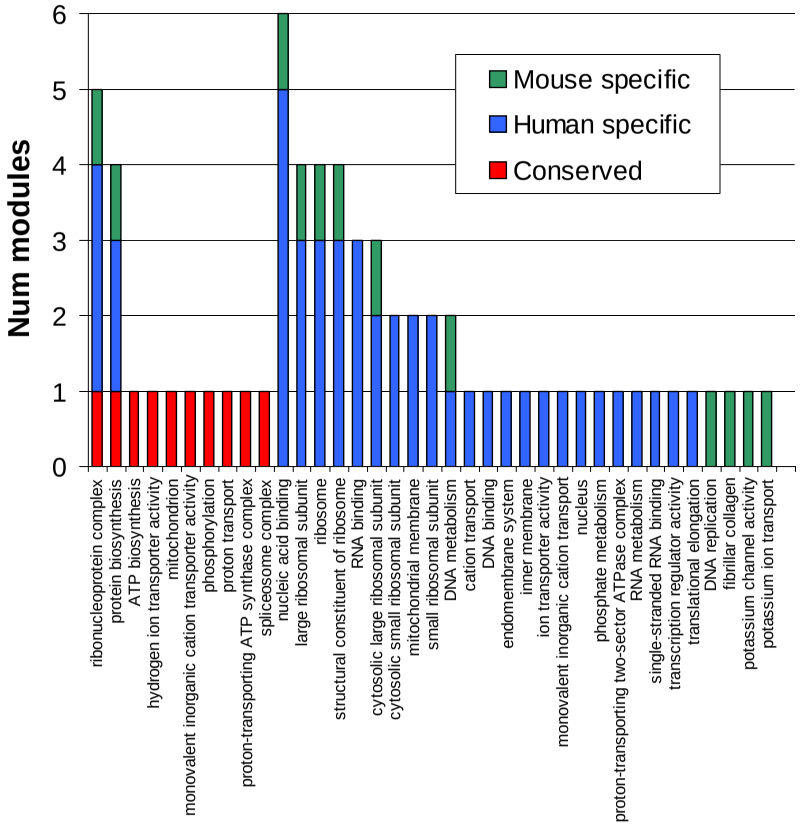


Figure 5.11: Distribution of GO enrichments by conservation.

repressed in the aging worm and in the aging fly, which together with our results may suggest common mechanisms between the expression programs of aging and advanced medulloblastoma tumors.

While the GO annotations provide an informative view of the learned modules, they only cover a small number of modules and are thus limited in scope for a comprehensive evaluation of our results. This small coverage could occur if either our learned modules grouped together unrelated genes, or if the GO database is missing functional information for many genes. Based on our statistical results from Section 5.4.2 and on the results of the module network procedure in yeast (see Section 4.6), we believe that the latter reason, i.e., the incompleteness of GO, accounts for the majority of the uncharacterized modules.

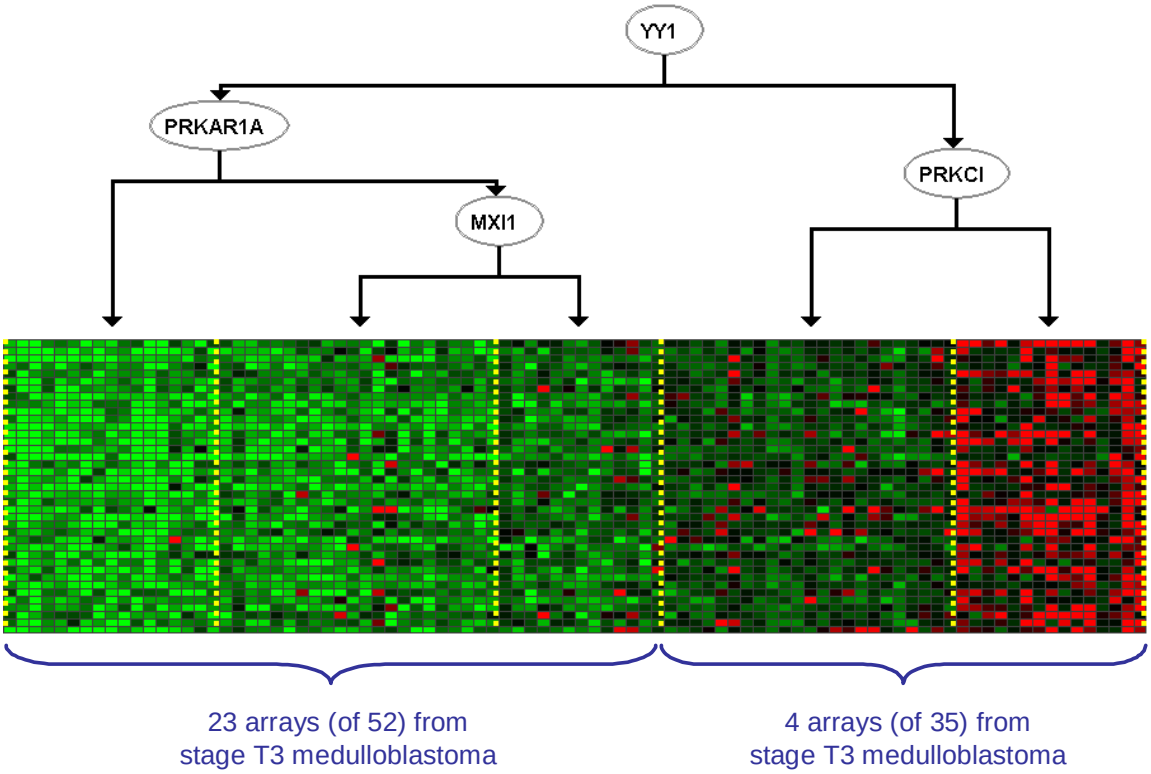


Figure 5.12: Predicted regulatory role for YY1 in patients with advanced (stage T3) medulloblastoma tumors. Shown is the regulation program of module 50 in the human module network, whose primary predicted regulator was the transcription factor YY1. Note that YY1 splits the arrays into two contexts, with one context containing most of the arrays from patients with stage T3 medulloblastoma, thereby predicting a regulatory role for YY1 in these tumors.

To obtain a more comprehensive view of our results, we also tested the enrichment of the annotations that we derived from the genome-wide expression profiles of Su *et al.* (2002). The resulting tests showed enrichment for many additional modules, providing further support our above conjecture: When performing similar enrichment computations for these annotations as we did for the GO annotations, we found enrichment for 112 module-tissue type annotation pairs (51 human specific, 36 mouse specific, and 25 conserved), spanning 31 modules and 44 annotations (see Figure 5.10).

As expected from our choice of datasets, the most significant enrichments were for genes that according to the dataset of Su *et al.* (2002) are expressed in brain tissue

sub-types (e.g., in human: cerebellum, $p < 5 \cdot 10^{-17}$; amygdala, $p < 10^{-12}$; cortex, $p < 2 \cdot 10^{-11}$; and whole brain, $p < 6 \cdot 10^{-11}$). However, we also detected enrichment for other tissue types, some representing samples from cancerous tumors that were not part of our input expression data (e.g., conserved enrichment for prostate cancer in corresponding modules in human and mouse, $p < 3 \cdot 10^{-6}$ in human and $p < 10^{-4}$ mouse; and enrichment of retinoblastoma in human, $p < 8 \cdot 10^{-6}$). Such modules might represent responses that are common between these cancers and the brain tumors in our expression compendium. Note that these tissue annotations are not represented in GO and thus we could not make the above inferences using GO. From the conservation perspective, the enrichment of 5 of the 7 annotations that represent expression in brain tissue sub-types was conserved between human and mouse as was the enrichment in prostate cancer tissues. Figure 5.13 shows the conservation properties for the tissue related annotations.

In addition to enrichment for gene annotations, we also identified condition annotations that are enriched in the regulation programs of the different modules. To this end, for each occurrence of a regulator as a decision node in a regression tree (regulation program), we computed the partition of each experimental condition annotation between the right branch (the true answer to the query on the regulator) and the left branch (the false answer), and used the binomial distribution to compute a p-value on this partition as described in Appendix D. We then defined a module to be significantly enriched for a condition annotation if any of the regulators in the regulation program of the module partitioned the arrays in a way that enriched one of the splits for the tested annotation (p-value ≤ 0.05; Bonferroni corrected).

These evaluations relative to condition annotations in the arrays showed enrichment for 312 module-condition pairs (206 human specific, 103 mouse specific and 3 conserved), spanning 85 of the 100 modules and 23 of the 27 condition annotations (see Figure 5.10). Similar to the enrichment relative to the tissue expression annotations, these results represent another global unbiased measure of evaluation, indicating that the regulation programs in most modules captured responses that are characteristic of certain tumor properties (e.g., sub-type and stage) and clinical outcomes. We note that these condition annotations are by definition specific to

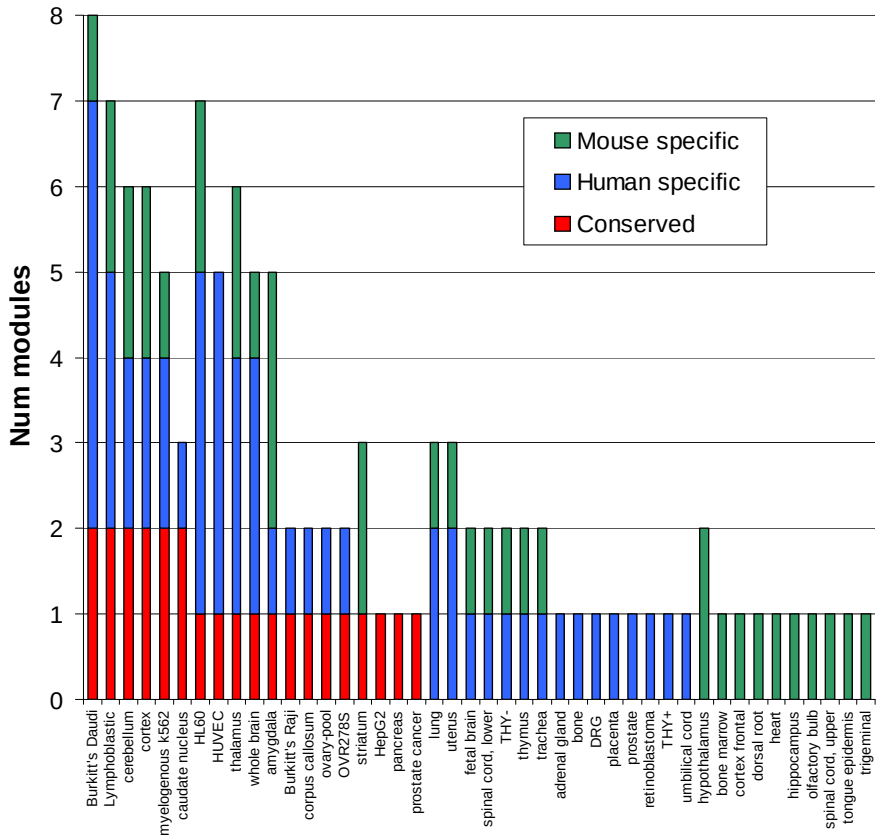


Figure 5.13: Distribution of tissue expression enrichments by conservation.

each organism, since, with the exception of two annotations (normal cerebellum and medulloblastoma), they were all defined only over the arrays of one organism.

Global View of Predicted Regulators

We next turned to a detailed evaluation of the regulators in the predicted regulation programs. To identify the predicted function of the regulators, we associated each regulator with biological processes and experimental conditions. As a regulator 'X' may regulate more than one module, its targets consist of the union of the genes in all modules predicted to be regulated by 'X'. We tested the targets of each regulator in the human and mouse module networks for enrichment of the same GO and tissue expression gene annotations as above. We took p-values < 0.05 (Bonferroni corrected

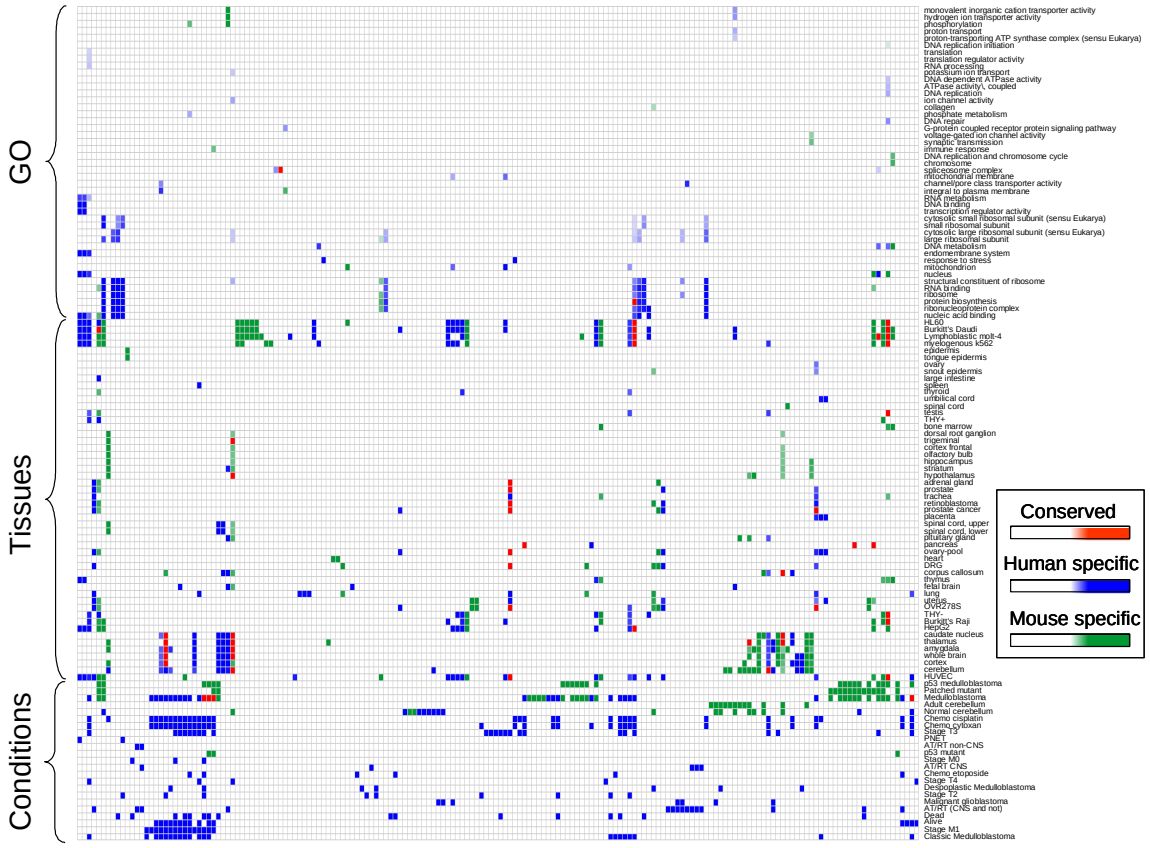


Figure 5.14: Enrichment of annotations in predicted targets of regulators. Entries represent the fraction of genes for each of the predicted targets of a regulator that are associated with the annotation.

as for module annotations) to be significant. We compiled the results into the single regulator-annotation matrix shown in Figure 5.14. As for the module enrichment, we distinguished the conserved regulator-annotation pairs from those that are organism specific, by coloring in blue the annotations that were enriched only in regulator targets from the human module network, in green the annotations that were enriched only in regulator targets from the mouse module network, and in red the annotations that were enriched in the regulation targets from both organisms.

Overall, we found enrichment for 158 regulator-GO-annotation pairs (132 human specific, 24 mouse specific, and 2 conserved), spanning 39 regulators and 45 annotations (see Figure 5.14). These included conserved enrichment for the spliceosome

complex in the targets of the protein kinase MAPKAPK2 in both human and mouse and conserved enrichment for genes involved in protein biosynthesis in the targets of the transcription factor YY1 in both human and mouse. We note that while both MAPKAPK2 and YY1 are known to control the expression of several biological processes, they have not been associated with regulation of the above processes.

As was the case for the module annotations, GO provided a very limited view of the function of the regulators and we obtained a more comprehensive view using the tissue expression annotations. We found enrichment for 446 regulator-tissue annotation pairs (209 human specific, 192 mouse specific, and 45 conserved), spanning 94 regulators and 52 annotations (see Figure 5.14). We found additional support for 59% (261 of 446) of these regulator-tissue pairs, as in these cases the regulator itself was also expressed in the respective tissue. Interestingly, when examining these 261 additionally supported cases by their conservation properties, we found that the regulator-tissue pairs that were predicted in both the human and the mouse module networks had much stronger support: 80% (36 of 45; $p < 0.001$) of them were additionally supported when examining the expression of the regulator, compared to only 65% (136 of 209; $p < 0.005$) and 46% (89 of 192; $p < 0.99$) additionally supported pairs in human and mouse, respectively. These results further indicate that we can better predict regulatory relationships by identifying those relationships that have been conserved between human and mouse.

As discussed above, we also tested each regulator for experimental conditions that it significantly regulates by examining how conditions are split by each relevant regulation tree. We found enrichment for 345 regulator-condition annotation pairs (250 human specific, 91 mouse specific and 4 conserved), spanning 127 regulators and 23 condition annotations. (see Figure 5.10). We again note that except for the 'medulloblastoma' and 'normal cerebellum' annotations, all condition annotations were defined per organism and thus we do not expect to find many conserved regulator-condition annotation pairs.

Nevertheless, among the 4 annotations that were conserved was the enrichment of Cyclin D1, a key cell cycle regulator, for splitting medulloblastoma arrays. This regulator appeared in the control programs of module 78 in both the human and

the mouse module networks and in both cases defined a split on the arrays that was significant with respect to arrays that represent medulloblastoma samples ($p < 0.02$ in human; $p < 10^{-6}$ in mouse). Moreover, our procedure predicted that Cyclin D1 regulates nearly the same set of target genes in both human and mouse, as 89% (34 of 38) and 85% (34 of 40) of the genes were shared in module 78 in human and mouse, respectively. This prediction, that Cyclin D1 is involved in regulation in medulloblastoma tumors, is supported by the recent study of Neben *et al.* (2004) which showed that higher levels of expression of Cyclin D1 predict an unfavorable survival for medulloblastoma patients. Interestingly, Cyclin D1 was also enriched for poor survival ($p < 0.03$) in our human module network (this test was not possible for mouse as this information was not available).

We note that the other three regulators, CSDA (cold shock domain protein), ISGF3G (interferon stimulated transcription factor), and PSMB10 (proteasome subunit), that were enriched for medulloblastoma splits in both human and mouse, are not known in the literature to play a role in this tumor. However, it is noteworthy that according to our method, all three of these regulators were also associated with predicting survival rates in patients (induction of CSDA was associated with high survival rates, $p < 0.03$; induction of ISGF3G and PSMB10 was associated with poor survival rates, $p < 0.002$ and $p < 0.03$, respectively).

Overall, the enriched regulator-condition annotations provide support for our computational predictions, and demonstrate that a large number of our inferred regulatory relationships are predicted as characterizing various tumor properties and clinical outcomes.

Conserved Annotations Enriched in Modules and Regulator Targets

We now present a global view and summary of the properties that we found to be conserved between our predicted regulatory modules in human and mouse. To obtain this view, we combined the module-annotation and regulator-annotation matrices of Figure 5.10 and Figure 5.14 and created a sub-matrix from them that included only those annotations whose enrichment was conserved (i.e., annotations that were enriched in at least one corresponding module between human and mouse or in the

targets of at least one of the regulators in both human and mouse). This sub-matrix consisted of 38 annotations, 18 modules and 16 regulators and is shown in Figure 5.15.

Based on this presentation, we can divide the conserved annotations into two groups. The first consists of annotations whose enrichment was always conserved (i.e., every time the annotation was enriched in the module or predicted targets of a regulator in one organism, it was also enriched in the corresponding module or in the predicted targets of the same regulator in the other organism). This group included the spliceosome (in module 75 and the targets of the regulator MAPKAPK2), mitochondrial genes (in module 50) and several energy related processes (mostly in module 27). Thus, our method suggests that these basic cellular processes are strongly selected against mutations that would alter their behavior. As we noted above, at least in the case of mitochondrial genes, there is mounting evidence from several independent studies in different organisms to support such a prediction.

The second group of conserved annotations consists of annotations that in addition to being conserved in some modules and regulator targets, were also enriched for some modules and regulator targets only in human or only in mouse. Thus, our method suggests that this second group of annotations represent processes that are partly conserved and partly evolving differently between human and mouse. Indeed, this group included annotations that are more broad (e.g., expression in brain tissue sub-types and other types of tumor such as prostate cancer and retinoblastoma) and thus perhaps more likely to evolve differently between human and mouse. We note, however, that this group also included annotations that represent basic cellular processes such as translation, which we might not expect to evolve differently between human and mouse.

Brain Related Modules and Regulators

As our input expression datasets represent expression profiles from human and mouse brain tumors, we next examined our results from the perspective of the information they provided about these processes. To this end, we compiled another sub-matrix from the matrices of Figure 5.10 and Figure 5.14, but this time we created the matrix such that it includes all the modules and regulators that were enriched for genes that

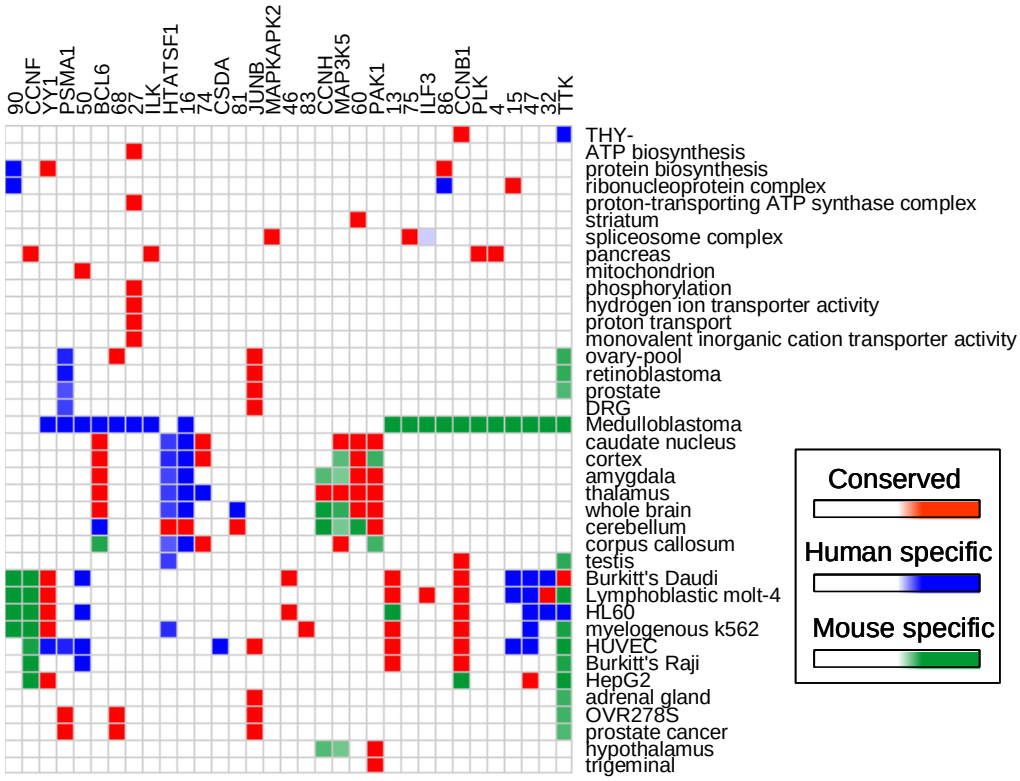


Figure 5.15: Enrichment of conserved annotations in modules and regulator targets.

are expressed in brain tissue sub-types according to Su *et al.* (2002). This sub-matrix, presented in Figure 5.16, consisted of 11 modules and 31 regulators that together were enriched for 59 annotations. As we noted earlier, some of the annotations enriched in these module and predicted targets were conserved between human and mouse while others were specific to human or mouse.

As can be seen from Figure 5.16, in addition to the enrichment of the module genes and the predicted targets for genes expressed in brain tissue sub-types, many of the regulation program splits were enriched for experimental condition annotations. Together, this suggests that the brain related modules and their associated regulators provide an informative view of the regulatory programs that underlie brain tumors in human and mouse. For example, module 17 was enriched for genes expressed in various brain tissue sub-types in both human and mouse. In human, the primary

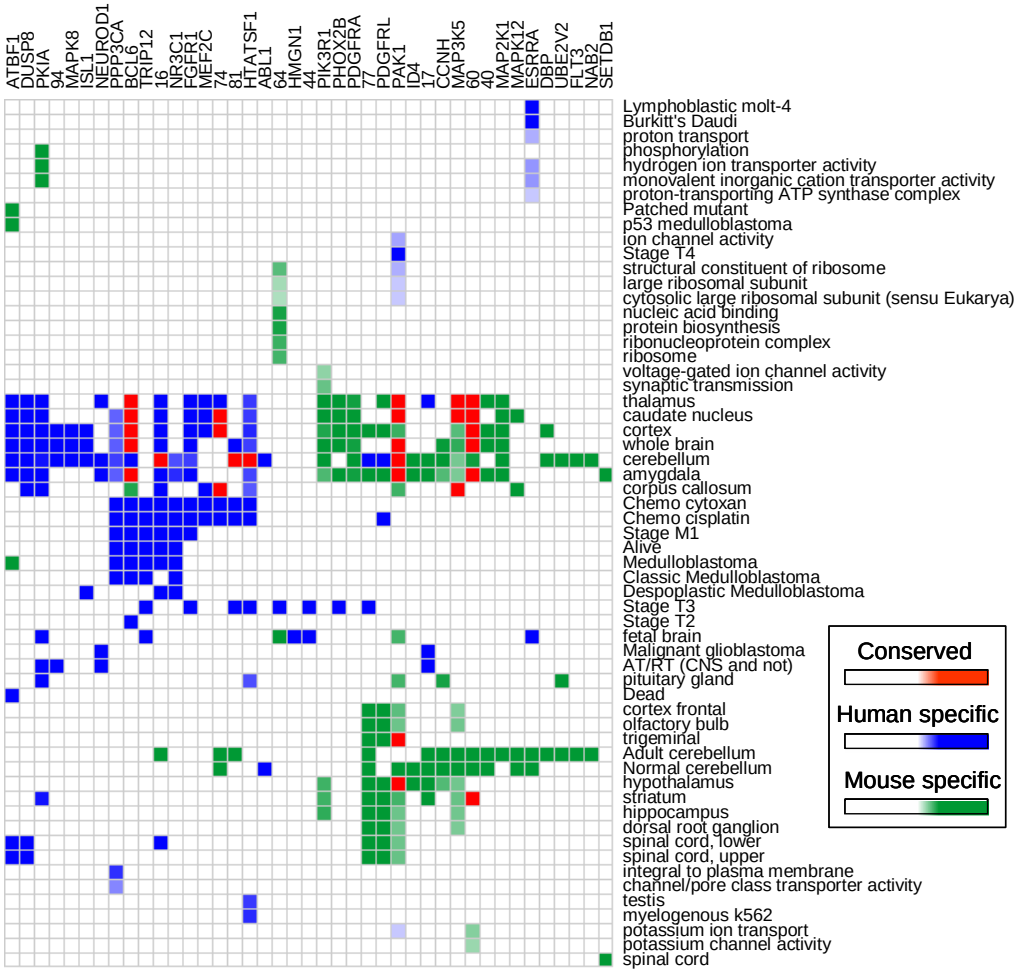


Figure 5.16: Annotation enrichment for brain related modules and regulator targets.

regulator predicted for this module was the neurogenic differentiation regulator Neuro D1 (see Figure 5.17). This regulator split the arrays in the module into two contexts, where one context contained all 10 arrays from the glioblastoma samples ($p < 0.005$) and all 10 arrays from the atypical teratoid/rhabdoid tumors (AT/RT) samples ($p < 0.005$), thereby predicting a regulatory role for Neuro D1 in these tumors. While there is no known role for NeuroD1 in these tumors, the study of Oyama *et al.* (2001) predicted a regulatory role for Neuro D1 in pituitary adenomas (another type of brain tumor of the pituitary gland), based on its selective expression in these tumors.

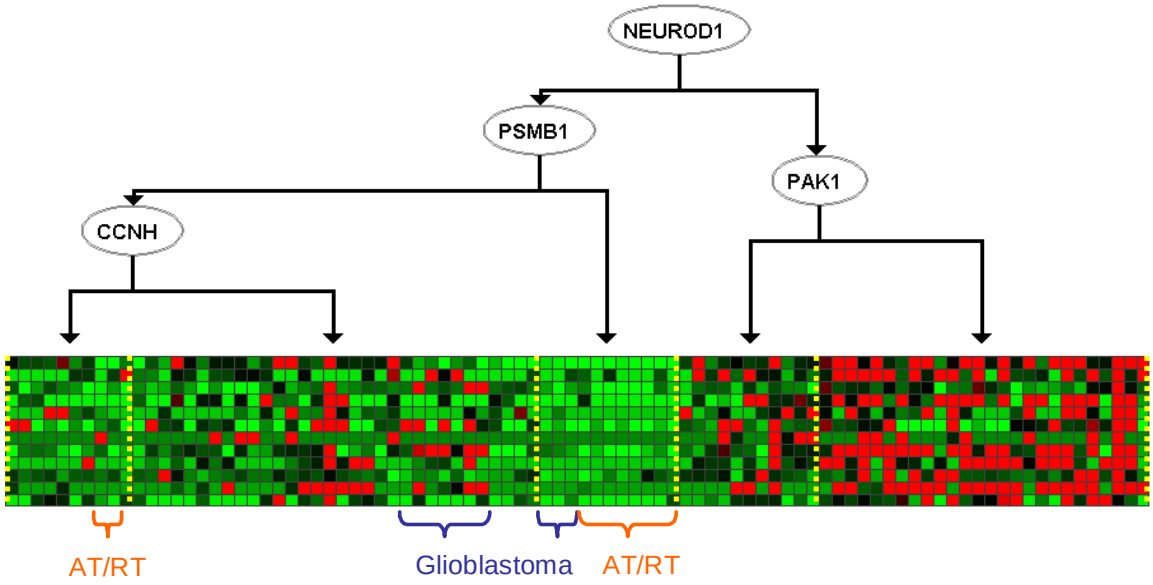


Figure 5.17: Predicted regulatory role for Neuro D1 in human brain tumors. Shown is the regulation program of module 17 in the human module network, whose primary predicted regulator was the neurogenic differentiation regulator Neuro D1. Note that Neuro D1 splits the arrays into two contexts, with one context containing all 10 arrays from the glioblastoma samples and all 10 arrays from the atypical teratoid/rhabdoid tumors (AT/RT) samples, thereby predicting a regulatory role for Neuro D1 in these tumors.

In mouse, the primary regulator predicted for module 17 was Cyclin H (see Figure 5.18). This regulator splits the arrays into two contexts, where one context consisted of 17 arrays, 16 of which represent induction in medulloblastoma samples ($p < 0.006$), thereby predicting a regulatory role for Cyclin H in medulloblastoma. We note that Cyclin H itself is also expressed in several of the brain tissue sub-types in the study of Su *et al.* (2002). Interestingly, the recent study of Neben *et al.* (2004) suggested Cyclin H as one of 54 genes whose expression in medulloblastoma predicts an unfavorable survival rate in patients with these tumors.

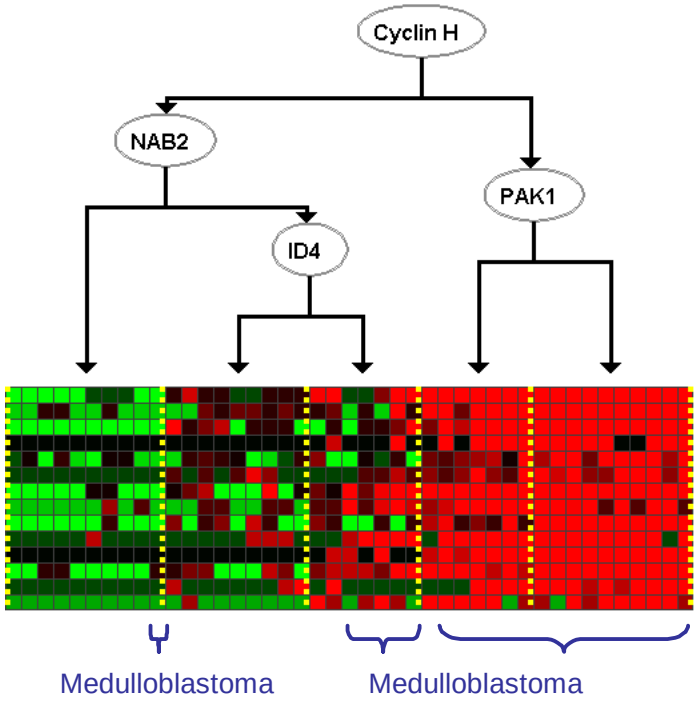


Figure 5.18: Predicted regulatory role for Cyclin H in mouse medulloblastoma tumors. Shown is the regulation program for module 17 in the mouse module network, whose primary predicted regulator was the Cyclin H. Note that this regulator splits the arrays into two contexts, with one context consisting of 17 arrays, 16 of which representing medulloblastoma samples, thereby predicting regulation of medulloblastoma by Cyclin H. A recent study (Neben *et al.*, 2004) suggested Cyclin H as one of 54 genes that predict unfavorable survival rates in medulloblastoma.

5.5 Discussion

In this chapter we presented *parallel module networks*, an extension of the module network framework presented in Chapter 4 for the task of learning regulatory modules from expression datasets of several organisms. We presented a joint probabilistic model over expression data from multiple organisms and an algorithm that learns this model automatically from the input data. Our model allows us to explicitly specify the degree of bias that the algorithm will have for learning conserved regulatory modules between the input organisms. This conservation preference is expressed via two independent parameters, one specifying the bias towards assigning orthologs to

the same module in different organisms, and the other specifying the bias towards assigning the same regulator parents in the regulation programs of corresponding modules in different organisms.

We applied our model to data that measured the expression of human and mouse in brain tumors (mainly medulloblastoma). From a statistical standpoint, we presented generalization results on held-out test data demonstrating that the parallel module network procedure learned better regulatory modules as compared to models learned on the expression data from each organism separately using the module network procedure of Chapter 4. The best performance was achieved for a setting of the conservation parameters that results in substantial conservation of the module assignments (about 70%) and of the regulators (about 50%) between the regulation programs of human and mouse.

¿From a global biological perspective, our model allowed us to suggest hypotheses regarding the properties that have been conserved in the regulation programs of human and mouse. We found that the regulators predicted by our method to be conserved in human and mouse were enriched for sharing several properties, including enrichment for immune response, cyclin, nucleic acid binding, and nuclear regulators, as well as enrichment for co-repressor activity and regulators expressed in brain. We also found several biological processes whose regulation has been conserved, including mitochondrion and several other energy related processes, spliceosome, and translation processes. This finding is consistent with several recent studies that showed conservation of these processes in independent expression datasets from human and mouse and from other organisms (Stuart *et al.*, 2003, Bergmann *et al.*, 2004, Trifunovic *et al.*, 2004, McCarroll *et al.*, 2004, Lee *et al.*, 2003a). In addition, we also showed that the conserved regulatory relationships predicted had stronger support in an independent expression dataset (Su *et al.*, 2002) compared to predictions that were specific to either the human or the mouse module networks.

Finally, the model suggested detailed hypotheses regarding the role of several regulators in our input brain tumors, including a conserved regulatory role for YY1 in regulating mitochondrial genes in medulloblastoma tumors in both human and mouse, a conserved regulatory role for Cyclin D1 in medulloblastoma, a regulatory

role for Cyclin H in medulloblastoma tumors in mouse, and a regulatory role for Neuro D1 in gliblastoma and atypical teratoid/rhabdoid tumors in human. The regulatory roles for YY1, Cyclin D1, and Cyclin H are supported by several recent studies (Seelan and Grossman, 1997, Lescuyer *et al.*, 2002, Neben *et al.*, 2004) and thus suggest interesting directions for future research into the regulatory role of these regulators in medulloblastoma.

Despite the successes described above, our method has several important drawbacks and limitations. First, we predict regulatory relationships using only measurement of mRNA expression data of the regulators and their targets. As many biological processes and events are not regulated at the transcriptional level, expression data provides an incomplete view of the regulation process and our method will thus fail to capture certain regulatory relationships. This assumption is similar to that made in the module network framework and we refer the reader to Section 4.8 for additional details about its entailed limitations.

Second, recall that our method takes as input an orthology mapping which specifies a set of genes that have a similar function across the input organisms. As the function of many genes is not known, such an orthology mapping is also not known. We thus followed the approach used in several recent studies and inferred an orthology map computationally, using the protein sequences of the input organisms. This procedure is bound to produce errors and may cause a problem for our method, which relies heavily on this mapping. Furthermore, as we noted in Section 5.4.1, in some protein families with a high degree of conservation, it is hard to select the exact ortholog based on sequence data alone. This implies that many genes will not have orthologs and will be unnecessarily excluded from our analysis.

One possible solution may be to develop a better procedure for inferring an orthology map, taking into account other information (e.g., syntenic regions). Another option is to use the rich expressive power of our probabilistic framework and represent the orthology mapping as a relationship between the genes of different organisms. By incorporating this relationship into our models, we can explicitly represent our uncertainty over the orthology map and allow the model to modify this mapping based on the expression data (in addition to the protein sequences). We are currently pursuing

this direction as part of our future work on this framework.

We chose to represent the regulation programs of each organism using a separate module network over its expression data. To find conserved regulatory relationships, we then defined two parameters that bias the learning algorithm towards models in which regulation programs of corresponding modules have the same genes and regulators. In this sense, our model does not use the data from one organism directly when learning the structure and estimating the parameters of the regulation programs of another organism. An alternative approach would thus be to allow regulation programs (regulator parents or regulation parameters or both) to be shared across organisms, which may potentially lead to more robust predictions. However, we note that such an approach is challenging, as it might require mapping experiments across expression datasets of different organism and normalizing the expression datasets such that we can compare expression measurements taken in different organisms.

We encode our preference towards assigning orthologs to the same module in different organisms by an assignment prior that gives a bonus each time an orthologous pair is assigned to the same module. Thus, according to our model, cases in which orthologs are assigned to different modules imply that the two genes diverged in function during evolution. As the phylogenetic trees between many organisms are known, an interesting extension is to take these trees into account in the assignment prior. For example, we may want to favor models whose module assignments predict the smallest number of divergence events during evolution, where the number of divergence events can be computed relative to the phylogenetic trees of the organisms.

Another interesting extension to our approach involves the biological question we address. Here, we focused on a model for identifying commonalities between the regulatory modules of different organisms. However, a complementary and interesting question is to focus on those relationships that are different and distinguish one organism from another. Using our method, we can address such questions only indirectly, by extracting those regulatory relationships from our learned models that were not conserved. However, a more direct approach, that also includes genes that have no orthologs as part of the analysis, is likely to provide more insights into this question. We note that even when focusing on the conserved regulatory modules,

including the data for genes that do not have orthologs may help in extracting the meaningful patterns in the data.

Finally, recall that our results included modules that were enriched for genes coordinately expressed in other cancerous tumors (prostate cancer and retinoblastoma). This suggests that if we learn our models over additional expression datasets, we might be able to further refine and improve our predictions. Thus, as more expression datasets are becoming available for many organisms and processes, it might be useful to incorporate them when learning regulatory modules. However, adding datasets raises other computational and scalability issues that will need to be addressed before such an approach is feasible.

Chapter 6

Conclusions

In this final chapter, we summarize the contributions of this thesis, discuss a number of its limitations, and present some challenges and future research directions that build on top of the work in this thesis.

6.1 Summary

In this thesis, we presented a statistical modeling language, based on Probabilistic Relational Models (PRMs), that we developed for representing complex interactions in the biological domain. We showed three applications of this framework for studying different aspects of the gene regulation process.

Discovering *cis*-Regulatory Modules

In the first application, we presented a unified model over gene expression and sequence data, aimed at providing a genome-wide explanation of the observed expression data as a function of combination of DNA sequence motifs. Previous standard approaches for the motif discovery task typically handle the expression and sequence data separately using different algorithms. We presented results on yeast, demonstrating that our method is better than such standard approaches at recovering known motifs and at generating biologically coherent modules. Finally, we also combined our results with binding location data to obtain regulatory relationships with known

transcription factors. As we showed, many of the inferred relationships have support in the literature, allowing us to derive global insights regarding the regulatory network of yeast.

Identifying Regulators of Biological Processes

Identifying the sequence motifs reveals part of the mechanism by which transcription factors regulate their target genes, but does not directly specify the identity of the regulatory proteins. Thus, in the second application of our modeling language, we presented *module networks*, a new probabilistic framework for discovering regulatory modules using only gene expression data. Our procedure identifies modules of co-regulated genes, their regulators, and the conditions under which regulation occurs, generating testable hypotheses in the form "regulator 'X' regulates module 'Y' under conditions 'W'". We presented wet lab experiments supporting three of our novel computational predictions, allowing us to suggest regulatory roles for three previously uncharacterized proteins in yeast. This was one of the first approaches that completed an entire discovery cycle that starts from publicly available data, then applies a computational method to analyze the data, generates testable hypotheses, and finally tests these hypotheses in the wet lab, ending with new insights into the biological system.

Discovering Conserved Regulatory Modules

Finally, due to noise in microarray technology and redundancy in the biological system, sometimes unrelated genes may be co-regulated, although this co-regulation may not be physiologically meaningful. To distinguish accidental co-regulation from functionally relevant co-regulation, our third application presented an extension of the module network procedure that incorporates expression data from several organisms into a unified probabilistic model. The basic idea underlying the unified model was to discover regulatory modules (regulators and their target genes) that are conserved in the regulation programs of all input organisms. As we showed in an application of our extended procedure to expression data of human and mouse brain tumors, by

combining expression data from multiple organisms, we can learn better regulatory modules than those which we can learn by applying the module network procedure separately to the data from each organism. From a global biological perspective, our model also allowed us to gain insights regarding the properties that have been conserved in the regulation programs of human and mouse. We found several commonalities in the set of conserved regulators and several functional properties that were enriched in the set of their conserved target genes. The model also suggested detailed hypotheses regarding a conserved role for several regulators in the regulation programs of human and mouse brain tumors.

Learning the Models

For all of the above models, we presented algorithms that learn the details (structure and parameters) of the model automatically from the input genomic data. This learning task is computationally very challenging, since, as we showed, when applying our models to genome-wide biological datasets, we need to deal with networks that have thousands of highly dependent hidden variables whose values we need to infer and whose structural dependencies we need to discover. To address these challenges, we adapted existing algorithms as well as developed new algorithms that exploit problem specific structures and result in efficient learning algorithms.

Evaluating the Results

As our methods are aimed at knowledge discovery in the biological domain where very little is known, it is often hard to evaluate their performance. Thus, throughout this thesis we presented several new types of statistical evaluations and new types of formal evaluations relative to the biological literature that objectively report on our performance. Many of these new evaluation methods can easily be adapted for evaluating the results of other approaches.

Visualizing the Results

Finally, for computational tools to have a broad impact, they must be accompanied by visualization and browsing tools that are easily accessible to biologists. To support this effort, we developed *GeneXPress*, a software environment for visualization and statistical analysis of genomic data, including gene expression, sequence data, and protein-protein interactions. In particular, GeneXPress can visualize the output of our algorithms and many of the figures in this thesis were generated directly from it. Currently, GeneXPress has been downloaded by over 800 researchers from more than 50 countries.

Incorporating Heterogeneous Types of Genomic Data

One of the key features of our modeling language is that it allows us to incorporate several types of genomic data into a single unified probabilistic model. By learning the details of such unified models, we can integrate different viewpoints of the cellular activity. In this thesis, we showed two such examples, one for integrating gene expression and sequence data and one for integrating gene expression data across multiple organisms. We also used our framework for integrating additional types of genomic data in other projects that were not presented in this thesis. In one case, we combined protein-DNA binding data together with gene expression and sequence data (Segal *et al.*, 2002). This extended model allowed us to reason about both the motifs and identity of the transcription factors that bind them within a single model. In another case, we integrated gene expression and protein-protein interaction data for the purpose of identifying biological pathways (Segal *et al.*, 2003d), based on the assumption that many pathways exhibit two properties: their genes exhibit similar gene expression profile and the protein products of the genes often interact. We showed that such integration is more successful at discovering both functional groups and entire protein complexes as compared to approaches that use each data type in isolation.

Certainly, it is possible to extend our language for integrating other types of

genomic data. As new technologies for obtaining other types of genome-wide measurements of the biological system are being developed, it is our belief that extending our approach to incorporate them can potentially lead to new biological discoveries.

6.2 Limitations and Future Directions

It is our hope that this thesis demonstrated the usefulness of unified probabilistic models for studying key problems in the biological domain. Obviously, our approach has several limitations and there are important aspects of the biological domain that it cannot currently model. In this section, we highlight some of these limitations and discuss some exciting research directions for future work.

6.2.1 Computational Complexity

While our models can integrate different types of genomic data and represent many of the complex interactions they posses, as we showed throughout this thesis, learning such models automatically from the data is computationally very challenging. Consequently, it is often infeasible to develop exact learning algorithms and we must resort to approximation algorithms and heuristics. Even then, when applying the framework to large scale genomic datasets, it may still take up to several hours or even several days to learn a single model. In contrast, many of the currently available procedural approaches are very fast and can sometimes give results within several minutes.

Thus, one challenge is to design better approximations and more efficient algorithms for learning the models from data. However, even with such algorithmic advances, it is hard to imagine achieving performance in terms of computation time similar to that of some of the fast procedural approaches, especially in light of the increasing size of the available genomic datasets. This limitation of our model implies that it will be hard to apply our approach in some settings (e.g., where real-time performance is needed). Thus, when considering the application of our framework to study a biological problem, one should evaluate the added value gained by its use and trade that off with the extra time computation time it requires. We note, however,

that although it may take several hours to learn a single model within our framework, it is still considerably less time compared to the several months or years it takes to measure and collect the biological data.

6.2.2 Estimating the Confidence of Our Predictions

A potential pitfall of our approach is that, given an input dataset, it will always learn a model regardless of the quality of the signal present in the data. For example, given expression data, the module network procedure will always learn gene modules and predict their regulators even if the resulting model poorly predicts the expression of the target genes as a function of the expression of the regulators. This behavior is in contrast to hypothesis-driven approaches that first assume that the data was generated by some null distribution, and then compute the probability that such an assumption indeed holds in the data. Only patterns in the data that are unlikely to have been generated by the null distribution are then reported. Thus, when the signal in the data is weak, these approaches may have the desired property of not reporting any results.

Obviously, hypothesis-driven approaches have other limitations, such as the fact that they rely heavily on the choice of null distribution and that the tested hypotheses must be specified in advance rather than being driven by the input data. Yet, their ability to provide confidence estimates for the results they report is a highly desired feature. In this thesis, we implemented this feature only indirectly, through statistical generalization tests on held out data and through the evaluation of our results relative to the literature and to current biological databases. As a more direct approach, in some cases we can make use of well known methods for confidence estimation such as *bootstrap* (Efron and Tibshirani, 1993), which repeatedly learns models from resamples of the original input data and then estimates the confidence of different features of the model based on the number of times they appear in all models learned. This approach was used by Pe'er *et al.* (2001) for estimating the confidence of the features of regulatory networks inferred from expression data using Bayesian networks. Adapting such confidence estimation approaches for our models can greatly enhance

the reliability of our results.

6.2.3 Incorporating Prior Biological Knowledge

While there are advantages to learning models in an unbiased way directly from the data, in some cases it may be useful to incorporate prior biological knowledge into the learning process. Such prior knowledge can come, for example, from manually curated biological databases such as GO, but also from the results of applying genome-wide computational methods such as the ones presented in this thesis. The idea is that once we are satisfied with some aspects of our model, we can make use of them in designing the next model, rather than having to start the design process each time from scratch. Incorporating prior knowledge of the form that exists in biological databases can also help us in focusing on the novel predictions made by the method, rather than on rediscovering what is already known.

In theory, we can easily encode such prior knowledge into our model. For example, if we know the targets of some regulator, we can bias our model towards including those regulatory relationships. In practice, however, devising a principled approach for incorporating prior knowledge and for reusing parts of previous models is challenging, especially since we may want to represent the various degrees of confidence that we have in this prior knowledge, depending on the source from which it was derived.

6.2.4 Modeling Time and System Dynamics

The cell is constantly responding to the environment by changing its state, as defined by the activity level of its proteins. Clearly, the state of the cell at one time point is highly dependent on its previous state and on the environmental signals it receives. In this thesis, we completely ignored such time dependencies, even though some of the expression data we used had such information as it was measured across consecutive time points. In some cases, ignoring this aspect may result in unrealistic computational predictions. For example, the module network procedure may predict that a regulator activates its targets 5, 15, and 25 minutes after the cell receives some

stress signal, but not 10 or 20 minutes after receiving the signal.

One way to incorporate time into our framework is to add dependencies between variables in consecutive time points, as in the framework of *Dynamic Bayesian Networks* (DBNs). For example, the expression level of a gene at one time point might depend on its expression level in the previous time point and on the expression of its regulator in the current time point. However, a naive application of DBNs assumes that these dependencies are time homogeneous, i.e., that the form of the dependency of a variable on variables from the previous time point does not change under different conditions or in different time points. As the biological system may change its behavior under different conditions, such an approach may be overly simplistic.

We note that, while the time homogeneity assumption can be addressed, a more inherent limitation of DBNs is that they make the *first-order Markov assumption*, which asserts that variables in time point $t + 1$ are independent of variables in time $t - 1$ given an assignment to the variables in time point t. Clearly, this assumption is often too restrictive in the biological domain where, for example, we may want the expression of a gene at each time point to depend on its overall behavior in the time series. Thus, an important direction for future work is to devise a general scheme for modeling time in biological systems that can be incorporated within our modeling language.

6.2.5 Modeling Cyclic Dependencies and Feedbacks

Recall that our models always define a joint distribution by the ground Bayesian network that it induces over the set of random variables in the domain. In order for this joint distribution to be coherent, the resulting Bayesian network must be acyclic. This acyclicity constraint is a serious limitation in the biological domain, where there are many systems that utilize feedback loops, in which one gene affects the expression of another gene, which in turn affects the expression of the first gene.

One way to model such cyclic dependencies is to properly incorporate time into our models as described above. We can then have cyclic dependencies as long as they do not occur in the same time. For example, we can model a dependency of

some variable A at the current time point on the value of some other variable B from the previous time point, and also the dependency of B in the current time point on the value of A in the previous time point. However, such models may be difficult to learn and hard to interpret. We note that Richardson (1996) proposed a different formalism for modeling cyclic dependencies in directed graphs that would be interesting to explore within our framework.

Another approach for relaxing the acyclicity constraint is to use undirected graphical models, such as *Markov Networks* (Pearl, 1988) or their extension to the relational domain as presented by Taskar *et al.* (2002). However, the learning task (especially the structure learning problem) for this class of networks is more complicated than it is for directed graphical models.

6.2.6 Integrating Realistic Models of Biological Interactions

As discussed above, we use our language to represent complex interactions among various entities in the biological domain, such as the binding of a transcription factor to its target site on the DNA or the binding between two physically interacting proteins. In representing such interactions, we make simplifying assumptions regarding the details of the interaction. For example, we represent the interaction between a transcription factor and the motif to which it binds using a PSSM model that assumes independence among the nucleotides in the different positions of the motif. Clearly, such models only roughly approximate the true underlying interactions in the biological system.

Recently, there have been several works that build more sophisticated quantitative models based on the chemistry of molecular interactions (e.g., see Kalir and Alon (2004) and Nachman *et al.* (2004)). By incorporating such detailed quantitative models, we can construct more realistic models of the underlying biological system. However, these models incur additional computational costs and add degrees of freedom to the model that may potentially lead to overfitting. Thus, a key challenge in constructing more realistic models of physical interactions lies in extending the modularity of our framework in order to reduce the computational costs, and in choosing

the appropriate level of detail we include in the model based on the data that we have available for learning the models.

6.2.7 Using the Models for Predicting New Behavior

Finally, our work has focused on the task of learning models from data and on analyzing the patterns that these models find in the input data. However, we do not have to limit ourselves to use the model only for these purposes. If we trust the models that we learn, then once we obtain a model from data, we can use it for reasoning about the underlying biological system, even under conditions that are not represented in the input data. For example, we can predict how the behavior of the biological system will change in response to deletion of some regulatory genes, if we set the expression level of these genes to zero, and then use the model to infer the value of the other genes in the system. While in theory such an approach is feasible, its reliability and usefulness remains to be seen.

6.3 The Challenge Ahead

In this thesis, we presented a statistical modeling language that we developed for the biological domain and several applications of this framework for addressing key problems in gene regulation, including discovering regulators of biological processes, finding the motifs to which these regulators bind, and detecting regulatory relationships that have been conserved across evolution. It is our hope that these applications demonstrate that the general framework of probabilistic relational models is a powerful tool for representing complex interactions and for studying key problems in the biological domain.

Nevertheless, we view our work only as a first step in the direction of laying out the computational frameworks that will be crucial for the transition of biology into becoming an information science. It is evident that in the upcoming years, much more data will become available but more importantly, new types of data, such as protein-level and tissue-specific expression, will be measured on a genome-wide

scale. Combined with the fact that these data will be measured in many different organisms, across a wide range of environmental conditions, it is clear that soon we will be overwhelmed by the amount of data that will be available, and will need to develop new ways of utilizing it to further our understanding of biology. It is our hope that this thesis demonstrates that by continuing to extend and develop our framework along the lines discussed above, we might be able to use it for addressing many of these upcoming challenges.

Appendix A

Data Marginal Likelihood for Multinomial Distribution

Assume that that $X[1], \ldots, X[N]$ form a random sample from a multinomial distribution, where each $X[i]$ can take on one of k different values, and for which the parameter vector $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_k\}$ is unknown. Also assume that the prior distribution over $\boldsymbol{\theta}$ is a Dirichlet distribution with hyperparameters $\alpha = \{\alpha_1, \ldots, \alpha_k\}$:

$$P(\boldsymbol{\theta}) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1}$$

where $\Gamma(x)$ is the Gamma function defined as $\Gamma(x) = \int_0^\infty t^{x-1} e^{-x} dt$. If we let $\hat{S}[j]$ represent the number of instances $X[i]$ in the data that have the j-th value, then the log of the marginal likelihood of the data $D = \{X[1], \ldots, X[N]\}$, $\log P(D)$, is given by:

$$\begin{aligned}
\log P(D) &= \log \int_0^1 P(D \mid \boldsymbol{\theta}) P(\boldsymbol{\theta}) d\boldsymbol{\theta} \\
&= \log \int_0^1 \left(\prod_{j=1}^{N} \theta_j^{\hat{S}[j]} \right) \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} d\boldsymbol{\theta} \\
&= \log \left(\frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \right) + \log \int_0^1 \prod_{j=1}^{N} \theta_j^{\hat{S}[j] + \alpha_j - 1} d\boldsymbol{\theta}
\end{aligned}$$

Note that the quantity inside the integral is proportional to a Dirichlet distribution with parameters $\alpha = \{\hat{S}[1] + \alpha_1, \ldots, \hat{S}[k] + \alpha_k\}$. Since the integral of this Dirichlet distribution is 1, we know that:

$$\int_0^1 \frac{\Gamma(\sum_{i=1}^k \hat{S}[i] + \alpha_i)}{\prod_{i=1}^k \Gamma(\hat{S}[i] + \alpha_i)} \prod_{i=1}^k \theta_i^{\hat{S}[i]+\alpha_i-1} d\boldsymbol{\theta} = 1$$

Using this, we can now derive the log marginal likelihood of the data as:

$$\begin{aligned} \log P(D) &= \log\left(\frac{\Gamma(\sum_{i=1}^k \alpha_i)}{\prod_{i=1}^k \Gamma(\alpha_i)}\right) + \log\left(\frac{\prod_{i=1}^k \Gamma(\hat{S[i]} + \alpha_i)}{\Gamma(\sum_{i=1}^k \hat{S[i]} + \alpha_i)}\right) \\ &= \log\Gamma(\sum_{i=1}^k \alpha_i) - \sum_{i=1}^k \log\Gamma(\alpha_i) + \sum_{i=1}^k \log\Gamma(\hat{S[i]} + \alpha_i) - \Gamma(\sum_{i=1}^k \hat{S[i]} + \alpha_i) \end{aligned}$$

Appendix B

Data Marginal Likelihood for Univariate Gaussian Distribution

Assume that that $X[1], \ldots, X[N]$ form a random sample from a normal distribution for which both the mean μ and the precision τ are unknown ($-\infty < \mu < \infty$ and $\tau > 0$). Also assume that the joint prior distribution of μ and τ is as follows: The conditional distribution of μ given τ is a normal distribution with mean μ_0 and precision $\lambda_0\tau$ ($-\infty < \mu_0 < \infty$ and $\lambda_0 > 0$); and the marginal distribution of τ is a gamma distribution with parameters α_0 and β_0 ($\alpha_0 > 0$ and $\beta_0 > 0$). This prior distribution, $P(\mu, \tau)$, is called a Normal-Gamma distribution with parameters $\langle \mu_0, \lambda_0, \alpha_0, \beta_0 \rangle$. The log of the marginal likelihood of the data $D = \{X[1], \ldots, X[N]\}$, $\log P(D)$, is given by:

$$
\begin{aligned}
\log P(D) &= \log \int_{\mu=-\infty}^{\infty} \int_{\tau=0}^{\infty} P(D \mid \mu, \tau) P(\mu, \tau) d\tau d\mu \\
&= \log \int_{\mu=-\infty}^{\infty} \int_{\tau=0}^{\infty} \left(\prod_{i=1}^{N} (2\pi)^{-1/2} \tau^{1/2} \exp\left[-0.5\tau (X[i] - \mu)^2 \right] \right) \\
&\quad (2\pi)^{-1/2} (\lambda_0 \tau)^{1/2} \exp\left[-0.5\lambda_0 \tau (\mu - \mu_0)^2 \right] \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} \tau^{\alpha_0 - 1} \exp[-\beta_0 \tau] d\tau d\mu \\
&= \log \int_{\mu=-\infty}^{\infty} \int_{\tau=0}^{\infty} (2\pi)^{-N/2} \tau^{N/2} \exp\left[-0.5\tau \sum_{i=1}^{N} (X[i] - \mu)^2 \right] \\
&\quad (2\pi)^{-1/2} (\lambda_0 \tau)^{1/2} \exp\left[-0.5\lambda_0 \tau (\mu - \mu_0)^2 \right] \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} \tau^{\alpha_0 - 1} \exp[-\beta_0 \tau] d\tau d\mu
\end{aligned}
$$

$$
\begin{aligned}
= \quad & -\frac{N}{2}\log(2\pi) + \frac{1}{2}\log\lambda_0 + \alpha_0\log\beta_0 - \log\Gamma(\alpha_0) + \\
& \log\int_{\mu=-\infty}^{\infty}\int_{\tau=0}^{\infty}\tau^{N/2}\exp\left[-0.5\tau\sum_{i=1}^{N}(X[i]-\mu)^2\right] \\
& (2\pi)^{-1/2}\tau^{1/2}\exp\left[-0.5\lambda_0\tau(\mu-\mu_0)^2\right]\tau^{\alpha_0-1}\exp[-\beta_0\tau]d\tau d\mu
\end{aligned}
$$

If we now let the mean of the observed data be $\bar{X} = \frac{1}{N}\sum_{i=1}^{N}X[i]$ and define:

$$
\begin{aligned}
\lambda_1 &= \lambda_0 + N \\
\mu_1 &= \frac{\lambda_0\mu_0 + N\bar{X}}{\lambda_1} \\
\alpha_1 &= \alpha_0 + \frac{N}{2} \\
\beta_1 &= \beta_0 + \frac{1}{2}\sum_{i=1}^{N}(X[i]-\bar{X})^2 + \frac{N\lambda_0(\bar{X}-\mu_0)^2}{2\lambda_1},
\end{aligned}
$$

then after some simple algebra we can show that:

$$
\sum_{i=1}^{N}(X[i]-\mu)^2 + \lambda_0(\mu-\mu_0)^2 = \lambda_1(\mu-\mu_1)^2 + \sum_{i=1}^{N}(X[i]-\bar{X})^2 + \frac{N\lambda_0(\bar{X}-\mu_0)^2}{\lambda_1}
$$

which means that we can now rewrite the log marginal likelihood as:

$$
\begin{aligned}
\log P(D) \quad = \quad & -\frac{N}{2}\log(2\pi) + \frac{1}{2}\log\lambda_0 + \alpha_0\log\beta_0 - \log\Gamma(\alpha_0) + \\
& \log\int_{\mu=-\infty}^{\infty}\int_{\tau=0}^{\infty}(2\pi)^{-1/2}\tau^{1/2}\exp\left[-0.5\lambda_1\tau(\mu-\mu_1)^2\right]\tau^{\alpha_1-1}\exp\left[-\tau\beta_1\right]d\tau d\mu
\end{aligned}
$$

Note that the quantity inside the integral is proportional to a Normal-Gamma distribution with parameters $\langle\mu_1, \lambda_1, \alpha_1, \beta_1\rangle$. Since the integral of this Normal-Gamma distribution is 1, we know that:

$$
\int_{\mu=-\infty}^{\infty}\int_{\tau=0}^{\infty}(2\pi)^{-1/2}(\lambda_1\tau)^{1/2}\exp\left[-0.5\lambda_1\tau(\mu-\mu_1)^2\right]\frac{{\beta_1}^{\alpha_1}}{\Gamma(\alpha_1)}\tau^{\alpha_1-1}\exp\left[-\tau\beta_1\right]d\tau d\mu \quad = \quad 1
$$

Using this, we can now rewrite the log marginal likelihood of the data as:

$$\log P(D) \;=\; -\frac{N}{2}\log(2\pi) + \frac{1}{2}\log\lambda_0 + \alpha_0 \log\beta_0 - \log\Gamma(\alpha_0) + \log\frac{\Gamma(\alpha_1)}{{\beta_1}^{\alpha_1}\lambda_1^{1/2}}$$

Thus, the log marginal likelihood of a univariate Gaussian distribution where we have a Normal-Gamma prior distribution over the mean μ and precision τ with hyperparameters $\langle \mu_0, \lambda_0, \alpha_0, \beta_0 \rangle$, is given by:

$$\log P(D) = \\ -\frac{N}{2}\log(2\pi) + \frac{1}{2}\log\lambda_0 + \alpha_0\log\beta_0 - \log\Gamma(\alpha_0) + \log\Gamma(\alpha_1) - \alpha_1\log\beta_1 - \frac{1}{2}\log\lambda_1$$

where $\langle \mu_1, \lambda_1, \alpha_1, \beta_1 \rangle$ are defined as above.

Appendix C

Computing Enrichment of Annotations

Assume that we are given a set of n objects. If these n objects are chosen randomly (without replacements) from a population of N objects, of which K are known to be of a particular type T, then we can use the hypergeometric distribution to compute the probability that the n randomly selected objects will contain k or more objects of type T. This probability will be:

$$P(X \geq k) = \sum_{i=k}^{n} \frac{\binom{K}{i}\binom{N-K}{n-i}}{\binom{N}{n}}$$

where $P(X \geq k)$ represents the probability that the n randomly selected objects will have k or more objects of type T.

We use the above computation in several places throughout this thesis, usually to test whether a set of genes is enriched for genes in the same functional category according to some gene annotation database (e.g., Gene Ontology (GO, (Ashburner *et al.*, 2000))). Under the null hypothesis that the set of genes is randomly chosen from the population of genes, we can treat the probability above as a p-value for the functional enrichment of the gene set relative to the tested annotation. As we usually test several different sets of genes, each against several different functional

annotations, we need to correct these p-values for the multiple tests we perform. The simplest such correction is a Bonferroni correction for multiple hypotheses, which simply multiplies the above p-value by the total number of tests performed.

Appendix D

Computing Significance of Annotation Partition

Assume that we are given a partition of N objects into two groups of size N_1 and N_2. If we randomly select n objects from the entire set of N objects, then we can use the binomial distribution to compute the probability that the n randomly selected objects will contain k or more objects from partition N_1. This probability will be:

$$P(X \geq k) = \sum_{i=k}^{n} \binom{n}{i} \left(\frac{N_1}{N}\right)^i \left(\frac{N_2}{N}\right)^{n-i}$$

where $P(X \geq k)$ represents the probability that the n randomly selected objects will have k or more objects from the N_1 partition.

We use the above computation in several places throughout this thesis, usually to test whether a set of arrays that are split by a regulator in some regulation program are enriched for arrays with the same experimental condition according to condition annotations for the arrays in the respective study. Under the null hypothesis that the set of arrays are randomly chosen from the population of arrays, we can treat the probability above as a p-value for the enrichment of the array set relative to the tested annotation. As we usually test several different sets of arrays, each against several different condition annotations, we need to correct these p-values for the multiple tests we perform. The simplest such correction is a Bonferroni correction for multiple

hypotheses, which simply multiplies the above p-value by the total number of tests performed.

Bibliography

S. Akutsu, T. Kuhara, O. Maruyama, and S. Minyano. Indentification of gene regulatory networks by strategic gene disruptions and gene over-expressions. In *Proceedings Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1998.

B. Alberts, A. Johnson, J. Lewis, K. Roberts, M. Raff, and P. Walter. *Molecular biology of the cell*. Garland publishing, 2002.

S. F. Altschul, Gish W., Miller W., E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25:25–29, 2000.

T.L. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings International Conference on Intelligent Systems for Molecaulr Biology (ISMB)*, pages 28–36, 1994.

A. Bairoch, P. Bucher, and K. Hofmann. The PROSITE database, its status in 1997. *Nucleic Acids Research (NAR)*, 25:217–221, 1997.

Y. Barash and N. Friedman. Context-specific Bayesian clustering for gene expression data. In *Proceedings Third Interational Conference on Computational Molecular Biology (RECOMB)*, 2001.

Y. Barash, G. Bejerano, and N. Friedman. A simple hyper-geometric approach for discovering putative transcription factor binding sites. In O. Gascuel and B. M. E. Moret, editors, *Algorithms in Bioinformatics: Proceedings First International Workshop*, number 2149 in LNCS, pages 278–293, 2001.

Y. Barash, G. Elidan, N. Friedman, and T. Kaplan. Modeling dependencies in protein-dna binding sites. In *Proceedings Seventh Interational Conference on Computational Molecular Biology (RECOMB)*, 2003.

A. Battle, E. Segal, and D. Koller. Probabilistic discovery of overlapping cellular processes and their regulation using gene expression data. In *Proceedings Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, 2004.

M.A. Beer and S. Tavazoie. Predicting gene expression from sequence. *Cell*, 117(2):185–198, 2004.

E.T. Bell. Exponential numbers. *Journal of the American Mathematical Society*, 41:411–419, 1934.

S. Bergmann, J. Ihmels, and N. Barkai. Similarities and differences in genome-wide expression data of six organisms. *PLoS Biology*, 2:E9, 2004.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. Twelfth Conference on Uncertainty in Artificial Intelligence (UAI '96)*, pages 115–123, 1996.

A. Brazma, I. Jonassen, J. Vilo, and E. Ukkonen. Predicting gene regulatory elements in silico on a genomic scale. *Genome Res.*, 8:1202–15, 1998.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Monterey,CA, 1984.

J. Buhler and M. Tompa. Finding motifs using random projections. In *Proceedings Fifth Interational Conference on Computational Molecular Biology (RECOMB)*, 2001.

H.J. Bussemaker, H. Li, and E.D. Siggia. Regulatory element detection using correlation with expression. *Nature Genetics*, 27:167–71, 2001.

G. Calinescu, H. Karloff, and Y. Rabini. An improved approximation algorithm for multiway cut. In *Proceedings 30th. Annual ACM symposium on theory of computation*, pages 48–52, 1998.

P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: a Bayesian classification system. In *Proceedings Fifth International Conference on Machine Learning (ML)*, pages 54–64, 1988.

J. M. Cherry, C. Ball, K. Dolinski, S. Dwight, M. Harris, J. C. Matese, G. Sherlock, G. Binkley, H. Jin, S. Weng, and D. Botstein. Saccharomyces genome database. *Nucleic Acids Research*, 26:73–79, 1998. http://genome-www.stanford.edu/Saccharomyces/.

D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings International Conference on Uncertainty in Artificial Inteillegence (UAI)*, pages 80–89, 1997.

D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*. Springer Verlag, 1996.

P.F. Cliften, L.W. Hillier, L. Fulton, T. Graves, T. Miner, W.R. Gish, R.H. Waterston, and M. Johnston. Surveying saccharomyces genomes to identify functional elements by comparative dna sequence analysis. *Science*, 11:1175–1186, 2004.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal of Computation*, 23:864–894, 1994.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

M. H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–39, 1977.

J. DeRisi, V. Iyer, and P. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 282:680–686, 1997.

P. D'Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mrna expression levels during cis development and injury. In *Proceedings Pacific Symposium on Biocomputing (PSB)*, pages 41–52, 1999.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, London, 1993.

M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings National Academy of Science (PNAS)*, 95(25):14863–8, 1998.

G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In *Proceedings Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 144–151, 2001.

S.L. Forsburg and L. Guarente. Identification and characterization of HAP4: a third component of the CCAAT-bound HAP2/HAP3 heteromer. *Genes and Development*, 3:1166–1178, 1989.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 421–460. Kluwer, Dordrecht, Netherlands, 1998.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

N. Friedman, I. Nachman, and D. Peér. Learning of Bayesian network structure from massive datasets: The "sparse candidate" algorithm. Submitted, 1999.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology (JCB)*, 7:601–620, 2000.

N. Friedman. The Bayesian structural EM algorithm. In *Proceedings UAI*, 1998.

A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Mol. Bio. Cell*, 11:4241–4257, 2000.

A.C. Gavin and *et al.*. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–7, 2002.

L. Getoor, D. Koller, and N. Friedman. From instances to classes in probabilistic relational models. In *Proceedings ICML Workshop*, 2000.

L. Getoor. *Learning Statistical Models From Relational Data*. PhD thesis, Dept. of Computer Science, Stanford University, 2001.

M.S. Halfon, Y. Grad, G.M. Church, and A.M. Michelson. Computation-based discovery of related transcriptional regulatory modules and motifs using an experimentally validated combinatorial model. *Genome Research*, 12:1019–1028, 2002.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1998.

T. Heinemeyer, X. Chen, H. Karas, A.E. Kel, O.V. Kel, I. Liebich, T. Meinhardt, I. Reuter, F. Schacherer, and E. Wingender. Expanding the TRANSFAC database towards an expert system of regulatory molecular mechanisms. *NAR*, 27:318–322, 1999.

W.S. Hlavacek and M.A Savageau. Rules for coupled expression of regulator and effector genes in inducible circuits. *Journal of Molecular Biology*, 255:121–139, 1996.

Y. Ho and *et al.*. Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectometry. *Nature*, 415:180–3, 2002.

P.E. Hodges, A.H. McKee, B.P. Davis, W.E. Payne, and J.I. Garrels. The yeast proteome database (ypd): a model for the organization and presentation of genome-wide functional data. *Nucleic Acids Research*, 27:69–73, 1999.

I. Holmes and W. Bruno. Finding regulatory elements using joint likelihoods for sequence and expression profile data. In *Proceedings International Conference on Intelligent Systems for Molecaulr Biology (ISMB)*, 2000.

T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–26, 2000.

J Ihmels, G Friedlander, S Bergmann, O Sarig, Y Ziv, and N Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31:370–377, 2002.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational approximations methods for graphical models. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

S. Kalir and U. Alon. Using a quantitative blueprint to reprogram the dynamics of the flagella gene network. *Cell*, 117(6):713–720, 2004.

M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The kegg databases at genomenet. *Nucleic Acids Research*, 30:42–46, 2002.

M. Kellis, N. Patterson, M. Endrizzi, B.W. Birren, and E.S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423:241–254, 2004.

W.J. Kent, C.W. Sugnet, T.S. Furey, K.M. Roskin, T.H. Pringle, A.M. Zahler, and D. Haussler. The human genome browser at ucsc. *Genome Research*, 12:996–1006, 2002.

D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *Proceedings International Conference on Uncertainty in Artificial Inteillegence (UAI)*, pages 302–313, 1997.

D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *AAAI '98*, 1998.

H. Langseth and T. D. Nielsen. Fusion of domain knowledge with data for structural learning in object oriented domains. *Machine Learning Research*, 4:339–368, 2003.

TI Lee, NJ Rinaldi, F Robert, DT Odom, Z Bar-Joseph, GK Gerber, NM Hannett, CT Harbison, CM Thompson, I Simon, J Zeitlinger, EG Jennings, HL Murray, DB Gordon, B Ren, JJ Wyrick, JB Tagne, TL Volkert, E Fraenkel, DK Gifford, and RA Young. Transcriptional regulatory networks in *saccharomyces cerevisiae*. *Science*, 298:799–804, 2002.

S.S. Lee, R.Y. Lee, A.G. Fraser, R.S. Kamath, J. Ahringer, and G. Ruvkun. A systematic rnai screen identifies a critical role for mitochondria in c. elegans longevity. *Nature Genetics*, 161(3):1101–1112, 2003.

Y. Lee, H.L. Miller, P. Jensen, R. Hernan, M. Connelly, C. Wetmore, F. Zindy, M.F. Roussel, T. Curran, R.J. Gilbertson, and P.J. McKinnon. A molecular fingerprint for medulloblastoma. *Cancer Research*, 63(17):5428–5437, 2003.

E. Lenssen, U. Oberholzer, C. Labarre, J.and De Virgilio, and M.A. Collart. Saccharomyces cerevisiae ccr4-not complex contributes to the control of msn2p-dependent transcription by the ras/camp pathway. *Molecular Microbiology*, 43:1023–1037, 2002.

P. Lescuyer, P. Martinze, and J. Lunardi. Yy1 and sp1 activate transcription of the human ndufs8 gene encoding the mitochondrial complex i tyky subunit. *Biochimica et biophysica acta*, 1574(2):164–174, 2002.

X. Liu, D.L. Brutlag, and J.S. Liu. Bioprospector: discovering conserved dna motifs in upstream regulatory regions of co-expressed genes. In *Proceedings Pacific Symposim on Biocomputing*, pages 127–38, 2001.

D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat Biotechnol*, 14(13):1675–80, 1996.

D. MacKay, R. McEliece, and J. Cheng. Turbo decoding as an instance of pearl's belief propagation algorithm. *IEEE J. Selected Areas in Communication*, 16(2):140–152, 1997.

S.A. McCarroll, C.T. Murphy, S. Zou, S.D. Pletcher, C.S. Chin, Y.N. Jan, C. Kenyon, C.I. Bargmann, and H. Li. Comparing genomic expression patterns across species identifies shared transcriptional profile in aging. *Nature Genetics*, 36(2):197–204, 2004.

H.W. Mewes, K. Albermann, K. Heumann, S. Liebl, and F. Pfeiffer. Mips: a database for protein sequences, homology data and yeast genome information. *Nucleic Acids Research*, 25:28–30, 1997.

R Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298:824–7, 2002.

K. Murphy and Y. Weiss. Loopy belief propagation for approximate inference: An empirical study. In *Proc. Fifthteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, 1999.

I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, Suppl 1, 2004.

R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

K. Neben, A. Korshunov, A. Benner, G. Wrobel, M. Hahn, F. Kokocinski, A. Golanov, S. Joos, and P. Lichter. Microarray-based screening for molecular markers in medulloblastoma revealed stk15 as independent predictor for survival. *Cancer Research*, 64(9):3103–3111, 2004.

J. Norbeck and A. Blomberg. The level of camp-dependent protein kinase a activity strongly affects osmotolerance and osmo-instigated gene expression changes in saccharomyces cerevisiae. *Yeast*, 16:121–137, 2000.

K. Oyama, N. Sanno, A. Teramoto, and R.Y. Osamura. Expression of neuro d1 in human normal pituitaries and pituitary adenomas. *Modern Pathology*, 14(9):892–899, 2001.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. In *Proceedings International Conference on Intelligent Systems for Molecaulr Biology (ISMB)*, 2001.

A. Pfeffer, D. Koller, B. Milch, and K. Takusagawa. SPOOK: A system for probabilistic object-oriented knowledge representation. In *Proceedings Thirteenth Annual Conference on Uncertainty in AI (UAI)*, pages 541–550, 1999.

A. Pfeffer. *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford University, 2000.

Y. Pilpel, P. Sudarsanam, and G.M. Church. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nature Genetics*, 29:153–9, 2001.

S.L. Pomeroy, P. Tamayo, M. Gaasenbeek, L.M. Sturla, M. Angelo, M.E. McLaughlin, J.Y. Kim, L.C. Goumnerova, P.M. Black, C. Lau, J.C. Allen, D. Zagzag, J.M. Olson, T. Curran, C. Wetmore, J.A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D.N. Louis, J.P. Mesirov, E.S. Lander, and T.R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442, 2002.

D. Poole. First-order probabilistic inference. In *Proceedings Eighteenth International Joint Conference on Artificial Intelligence (IJACI)*, pages 985–991, 2003.

T. Richardson. A discovery algorithm for directed cyclic graphs. In *Proceedings Thirteenth International Conference on Uncertainty in Artificial Inteillegence (UAI)*, pages 454–461, 1996.

N. Rosenfeld, M.B. Elowitz, and U. Alon. Negative autoregulation speeds the response times of transcription networks. *Journal of Molecular Biology*, 323:785–793, 2002.

F.P. Roth, P.W. Hughes, J.D. Estep, and G.M. Church. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnolology*, 16:939–945, 1998.

R. S. Seelan and L. I. Grossman. Structural organization and promoter analysis of the bovine cytochrome c oxidase subunit viic gene: a functional role for yy1. *Journal of Biological Chemistry (JBC)*, 272(15):10175–10181, 1997.

E. Segal and R. Sharan. A discriminative model for identifying spatial cis-regulatory modules. In *Proceedings Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 141–149, 2004.

E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(Suppl 1):S243–52, 2001.

E. Segal, Y. Barash, I. Simon, N. Friedman, and D. Koller. From sequence to expression: A probabilistic framework. In *Proceedings Sixth Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, 2002.

E. Segal, A. Battle, and D. Koller. Decomposing gene expression into cellular processes. In *Proceedings Eighth Pacific Symposium on Biocomputing (PSB)*, 2003.

E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module networks. In *Proceedings Nineteenth International Conference on Uncertainty in Artificial Inteillegence (UAI)*, pages 525–534, 2003.

E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: Discovering regulatory modules and their condition specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, 2003.

E. Segal, H. Wang, and D. Koller. Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics*, 19(Suppl 1):i264–i272, 2003.

E. Segal, R. Yelensky, and D. Koller. Genome-wide discovery of transcriptional modules from dna sequence and gene expression. *Bioinformatics*, 19(Suppl 1):i273–i282, 2003.

S.S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31:64–68, 2002.

S. Sinha and M. Tompa. A statistical method for finding transcription factor binding sites. In *Proceedings International Conference on Intelligent Systems for Molecaulr Biology (ISMB)*, pages 344–54, 2000.

R. Somogyi, S. Fuhrman, M. Askenazi, and A. Wuensche. The gene expression matrix: Towards the extraction of genetic network architectures. In *The Second World Congress of Nonlinear Analysis (WCNA)*, 1996.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O.

Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9(12):3273–97, 1998.

J.M. Stuart, E. Segal, D. Koller, and S. Kim. A gene co-expression network for global discovery of conserved genetic modules. *Science*, 302:249–255, 2003.

A.I. Su, M.P. Cooke, K.A. Ching, Y. Hakak, J.R. Walker, T. Wiltshire, A.P. Orth, R.G. Vega, L.M. Sapinoso, A. Moqrich, A. Patapoutian, G.M. Hampton, P.G. Schultz, and J.B. Hogenesch. Large-scale analysis of the human and mouse transcriptomes. *Proceedings National Academy of Science (PNAS)*, 99(7):4465–4470, 2002.

A. Tanay, R. Sharan, and R Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18 Suppl 1:S136–S144, 2002.

B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Eighteenth International Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 485–492, 2002.

RL Tatusov, DA Natale, IV Garkavtsev, TA Tatusova, UT Shankavaram, BS Rao, B Kiryutin, MY Galperin, ND Fedorova, and EV Koonin. The COG database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Research*, 29:22–28, 2001.

S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281–5, 1999.

Trifunovic, A., A. Wredenberg, M. Falkenberg, J.N. Spelbrink, A.T. Rovio, C.E. Bruder, Y.M. Bohlooly, S. Gidlof, A. Oldfors, R. Wibom, J. Tornell, H.T. Jacobs, and N.G. Larsson. Premature ageing in mice defective mitochondrial dna polymerase. *Nature*, 429(6990):417–423, 2004.

M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs. In *Advances in Neural Information Processing Systems 14*, Cambridge, Mass., 2001. MIT Press.

D. Weaver, C. Workman, and G. Stormo. Modeling regulatory networks with weight matrices. In *Proceedings Pacific Symposium on Biocomputing (PSB)*, pages 112–123, 1999.

Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1):1–41, 2000.

E. Wingender, X. Chen, Fricke E., R. Geffers, R. Hehl, I. Liebich, M. Krull, V. Matys, H. Michael, R. Ohnhauser, M. Pruss, F. Schacherer, S. Thiele, and S. Urbach. The TRANSFAC system on gene expression regulation. *Nucleic Acids Research*, 29:281–283, 2001.

E. A. Winzeler, D. D. Shoemaker, A. Astromoff, H. Liang, K. Anderson, B. Andre, R. Bangham, R. Benito, J. D. Boeke, H. Bussey, A. M. Chu, C. Connelly, K. Davis, F. Dietrich, S. W. Dow, M. E. Bakkoury, F. Foury, E. Gentalen, Giaever G, J. H. Hegemann, T. Jones, M. Laub, H. Liao, N. Liebundguth, D. J. Lockhart, A. Lucau-Danila, M. Lussier, N. M'Rabet, P. Menard, M. Mittmann, C. Pai, C. Rebischung, J. L. Revuelta, L. Riles, C. J. Roberts, P. Ross-MacDonald, B. Scherens, M. Snyder, S. Sookhai-Mahadeo, R. K. Storms, S. Vronneau, M. Voet, G. Volckaert, T. R. Ward, R. Wysocki, G. S. Yen, K. Yu, K. Zimmermann, P. Philippsen, M. Johnston, and R. W. Davis. Functional characterization of the s. cerevisiae genome by gene deletion and parallel analysis. *Science*, 285:901–906, 1999.

L.F. Wu, T.R. Hughes, A.P. Davierwala, M.D. Robinson, R. Stoughton, and S.J. Altschuler. Large-scale prediction of saccharomyces cerevisiae gene function using overlapping transcriptional clusters. *Nature Genetics*, 31:255–265, 2002.

J.M. Young, C. Friedman, E.M. Williams, J.A. Ross, L. Tonnes-Priddy, and B.J. Trask. Different evolutionary processes shaped the mouse and human olfactory receptor gene families. *Human Molecular Genetics*, 11(14):535–546, 2002.