# UNIFIED MODELS FOR REGULATORY MECHANISMS

Thesis submitted for the degree
"Doctor of Philosophy"
by

**Yoseph Barash**

Submitted to the Senate of the Hebrew University
2005

This work was carried out under the supervision of
Prof. Nir Friedman

## Abstract

The code book for any living organism is given by a four letter alphabet molecule, the DNA. Nowadays, in the "post genomic era", we already have this code at hand, but very little understanding of how to read it. Obviously, living cells use this code to adopt their spatial content as a response to a wide range of external stimuli. The mechanisms by which this is achieved are mostly unknown. However, past research has established that the DNA contain not only genes' code but also "programs" that regulate the expression of these genes. Many of these programs control the level of gene transcription. In this dissertation we try to gain better understanding of these transcriptional regulatory programs by analyzing experimental data. Our approach is based on probabilistic generative modeling of experimental observations.

The challenge of inferring transcriptional regulatory programs includes unraveling their structure (which gene or gene sets define a program and which genes they control), their context (in what cellular conditions a program is activated by the cell), their quantitative effect ( to what extent these programs affect gene expression), and the programs' actual code (*e.g.*, the binding sites used by regulatory elements to attach to DNA). Learning these regulatory mechanisms has become possible as new high throughput experimental methods now enable researchers to measure which genes are differentially expressed under specific cellular conditions and which genes are bound by certain regulatory elements. However, learning regulatory programs from such data still poses several challenges. First, the biological processes involved are stochastic in nature. The experimental procedures involve a wide range of noise sources and still cover a small subset of cellular conditions, time effects etc. This results in noisy, sparse and partial data. Second, the search space of possible regulatory programs is large. So is the amount of possible matching codes to be found in the DNA (*e.g.*, binding sites of regulatory elements).

Given this setting for learning regulatory programs from high throughput data, the use of generative probabilistic graphical modeling in particular and machine learning in general seems naturally appealing. It facilitates incorporating prior knowledge we might have about the mechanisms involve; it is easy to interpret and extract meaningful biological hypotheses from the learned models, and is inherently adequate for handling stochastic and noisy data as input. In the theoretical part of this work, we present novel models for regulatory programs and regulatory elements binding sites. We introduce new algorithms to learn such models by incorporating various sources of data (*e.g.*, genomic sequence and expression level) in a unified probabilistic framework. We also develop algorithms to asses the statistical significance of our results. In the experimental part, we demonstrate the usefulness of this approach over several yeast data sets.

Since our work focuses on inferring the regulatory programs from the DNA code much of it is dedicated to unraveling the code of *cis*-regulatory elements *i.e.*, the locations along the DNA to which regulatory elements bind as part of executing the regulatory program. We start by introducing a novel discriminative approach towards regulatory elements binding site identification. We then extend the models used for binding sites to capture dependencies between biding sites' positions. As we show, this has the benefit of greatly improving binding site prediction accuracy. The next part is dedicated to experimental results derived from these novel algorithms for biding site identification in yeast. Our approach is modular and flexible in nature. It facilitates incorporating the improved methods of *cis*-regulatory element identification we developed in our higher level models for regulatory programs. We present two novel approaches to model and learn such programs. Both approaches involve combining different sources of genomic and genetic data into a unified probabilistic framework. We demonstrate how these models are able to identify regulatory programs in yeast, including combinatorial interactions of regulatory elements, their biding sites in the target genes, and the experimental conditions in which they are activated. We conclude with a discussion of future and related work.

# Acknowledgements

First and foremost, I would thank my adviser, Nir Friedman. Nir, a brilliant scientist, has showed me the Way (”Do” in Japanese) of scientific research. After being fascinated by the field of machine learning during my first year of graduate studies, I came to Nir who suggested I look into a biologically oriented problem as a summer project. After the summer ended and little progress was made, I went back to Nir and suggested I pay him back since I accomplished so little. Nir wisely explained to me that in the world of scientific research, you get payed based on your past achievements and what is hoped you will achieve in the future. Instead, he suggested I keep working on the problem which resulted in this dissertation. Along the way I learned much from Nir, ranging from theoretical computer science, through how to approach, formulate, and investigate a scientific problem, and ending with the more daily skills of a researcher that include writing papers, reviewing ones or preparing talks. I am grateful to Nir for teaching me all this while keeping an open and friendly approach.

I would like to thank several people from my more distant past for inspiring me to practice research. I was generally lucky to have good teachers along the way, starting from elementary school. Mr. Baroch Magad taught me history rather then science in eight grade, but he was the first to show me how to question things and explore outside the school books in the search for answers. During my high school years, I got the chance to research at the Beer-Sheva University under the supervision of the late Shlomo Efrima. He introduced me to scientific research and was my first close example of being a great scientist and a great man at the same time.

The Hebrew university served as a great environment for both my undergraduate and graduate studies. The general non formal yet non compromising scientific approach I felt in the computer science department suited me well. The members of the machine learning lab, Eli Shamir, Tali Tishby, Yoram Singer and Yair Weiss, have all educated me. Tali was the first to introduce me to the field of Machine learning and related information theory. His passion and vision about these areas have struck a cord in me. The members of my research committee: Hannah Margalit, Nati Linial and his wife Michal (not formally a committee member), had been helpful along the way.

Outside the Hebrew University I got the chance to work with Naftali Kaminski for over a year at the center of genomic research in Tel Hashomer. Naftali served as a co adviser with respect to our common research projects, giving me the freedom to ”do my thing” while instructing me at the same time on real lab work and medical research. I hold dear his advice, support and friendship along the way.

All my collaborators contributed to the level of scientific research we performed and enriched my knowledge. Gill Bejerano, the charming red pirate, was responsible to publishing our WABI paper and for accompanying me on a great bike trip in Denmark. Eran Segal and Daphne Koller had a major contribution in the PRM model for regulation. Aviv Regev proved to be an excellent collaborator to learn from when working with Erin O'Shea. Gadi Iris Elinor and Zohar contributed to my work in Tel Hashomer. Gal Elidan and Tommy Kaplan were a wonderful team to work with on various projects, a few of which appear here.

I was fortunate to work with people in the machine learning lab that were both nice and talented: Elad Schneidman and Shai Fine my first ML TAs who explained what Tali was talking about; Ran Bachrach and Kobi Kramer, with whom I was fortunate to study since my first year; Shai and Lavi, who shared a room with us; Gill, Eylon, Gal C., Amir N., Amir G., Noam S. and Cheni (who did a wonderful job teaching last year's course with me) - all these people created a fun and stimulating environment to work in, people you could always count on to enrich you.

Within the ML lab, Nir's group formed a close community that accompanied me. Danna, was always there to offer advice; Iftach, (who also did a great job teaching cbio with me), was a good friend, enriched my knowledge and spread calmness; Tommy, my scripting guru; Matan, with his scientific insights (and knifes, sushi, cats etc.); Yuval, Hillel, Ori, Noam L. who also taught D.S. with me, Tal (who recently returned) - all contributed in their own way to my research work as well as enjoyment in doing it. Special thanks for Gal Elidan who was not only a great friend but also served to show me how one can manage his time well and do good science, while immensely enjoying life going on trips all over the world. I hope to be able to follow this example in years to come.

I had the pleasure of instructing several people closely through these years. Omri, was a pleasure to work with. Noa, grew up to be the next generation in Nir's lab, along with Ariel. Noa and Ariel also became good friends and are sure to guaranty future excellence in Nir's group. Finally, Aviad deserves special thanks. Starting with a course project in his second year of studies, Aviad kept working with me on research projects because of his increasing interest and because we got along so well. He was personally responsible to turn the SeedSearcher algorithm from it's original implementation into a highly efficient software.

I am grateful to the Israeli ministry of science for awarding me the Eshkol fellowship. Handling finance as well as any other administrative issue was done by a great team in the cs department. Eli, Yaeli, Rina, Lital, Ayelet, Irit, Regina and others - all handled whatever problem came their way in an efficient and most friendly way. I am in debt for that.

I want to thank my parents and sisters for giving me a good foundation during my childhood. Hadassah, my mother, served as a living example for me in many good ways and never refrained during all these years to help me become the best I can, while doing it my own way.

Finally, I want to thank my family. First, my wife and my light, Ofrit. She who supports me through all the troubles and successes, lets me be the best I can and makes me a better human being along the way. I thank our sons, Omer and Oron, just for being the way they are. Together with Ofrit they are the true joy of my life. I dedicate this work to my family.

# Contents

# Chapter 1

# Introduction

A common practice is to introduce the DNA code as a *blueprint*, like the ones engineers use to build buildings. This is actually an oversimplification of the “code for life”. A blue print, by definition, is fixed and has no time factor involved. The DNA code, more like a cooking recipe, also tells *when* certain things should be activated and how - for example genes controlling the embryonic stages. A *meta recipe* may be a better allegory since the DNA is an “active” code, telling the living cell also how to adopt its spatial content as a response to a wide range of external stimuli. Following our recipe allegory, this means it would tell us how to increase/decrease the amount of baking soda or the time in the oven as a response for a particularly hot day. This dissertation is all about increasing our limited understanding of this “meta recipe”.

Given that the DNA sequence is known to us, the first level of understanding is where the functional units, called genes, are coded in it. Assuming this is also roughly known via various experimental and computational analysis performed, we focus in this work on a certain type of “meta recipes” - programs regulating the production of the genes at the *transcription* level *i.e.*, when DNA is transcribed into mRNA. This is a necessary step in gene production and much of the living cell’s control on its spatial content is believed to be conducted via regulatory programs at this transcriptional stage, hence their importance to us.

To aid us in this task, experimental methods developed in recent years have enabled us to simultaneously measure the mRNA levels of thousands of genes under diverse conditions. Other new methods help us by testing thousands of genes to see whether certain proteins called *transcription factors* are involved in regulating their production. These high throughput experiments, combined with the huge amounts of DNA coding sequence, are actually responsible for developing our field of research, that of computational biology. They pose us the challenge of turning these vast amounts of data into valid biological hypotheses that can later be tested in a lab. We need to develop computational tools that are able to cope with large amount of data that is usually very noisy and partial.

Specifically, using this data we aim to answer questions such as: how are the transcriptional regulatory programs coded? Which transcription factors do they involve? In what cellular conditions do they operate? Which gene do they affect?

We now give a brief description of the biology behind genetic regulation, followed by some of the experimental methods used to produce high throughput data. We then review common approaches for this problem before we describe our approach and outline its presentation.

## 1.1 Gene Transcriptional Regulation

Our bodies are made of different types of living cells making up tissues that in turn make up our organs. Yet all the cells within one organism share exactly the same code for creating them and making them live. This is the general rule for all living organisms ranging from those composed of just a single cell (*e.g.*, yeast, the organism that helps us make both bread and alcoholic beverages) to more complex creatures such as ourselves. So how come a neuron and a liver cell can differ so greatly in their look, spatial content and reactions to stimuli yet share exactly the same code?

The central dogma in molecular biology is that this “code of life” is given by a very long molecule called DNA which is used to transcribe RNA, later to be translated into the proteins which make up much of the cell function and content. As Watson and Crick found in the fifties of the last century, the DNA consists of two strands that wrap around together in a structure named *alpha helix*. Each strand is made of four different versions of a molecule. The difference is in the content of the *base* part. The four versions of the base are adenine (A), thymine (T), cytosine (C), and guanine(G). The DNA double strand structure is based on covalent bonds between the molecules on each side. Usually adenine (A) pairs with thymine (T) and cytosine (C) with guanine (G). Certain regions in this four letter code [1] named *genes*, encode how to make the molecules, mainly proteins, that compose the cell’s content and are responsible for carrying out different functions within it. In order to create these molecules, a DNA strand is first *transcribed* to a similar molecules called *mRNA*. These molecules are then processed, exported out of the nucleus (in eukaryotic organisms) later to be *translated* into the needed protein (see Figure **??**). Controlling what proteins to make and when is what makes the neuron so different from the liver cell. So how is this achieved?

The *regulation* of gene production is achieved via many regulatory mechanisms acting at different production stages. First, it is important to realize DNA does not just waver in the cell as a huge double helix molecule. [2] The double helix is coiled up around *nucleosomes* and packed to

[1]Note that current estimation is that only about 2% of the genome actually encodes for genes. Some other parts encode regulatory programs of genes production as explained below while much of the genome is now believed to be “junk” DNA with no functional rule.

[2]The human genome is a $3 * 10^9$ base pairs long molecule. If we were to stretch the DNA from a single cell of our body as a thin cord it would reach about 2.5 meters long.

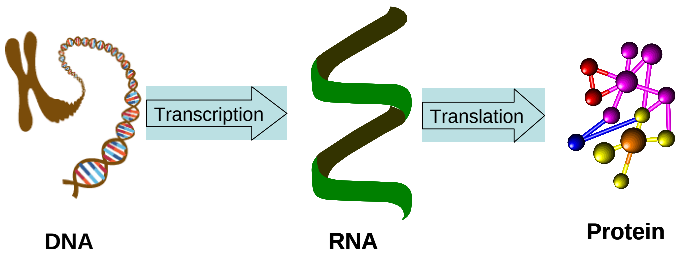


Figure 1.1: The central dogma in molecular biology for the "code of life".

form the *chromatin* matter made of several *chromosomes* (in our case there are 46 such chromosomes making up 23 pairs). In order for the DNA to be transcribed it has to be opened up so that the *RNA Polymerase* mechanism can work (Figure **??**). Some regulatory mechanisms control the DNA structure by what is called *chromatin remodeling* and hence the possibility to transcribe genes. Other mechanisms affect the amount of *active proteins* in the cell *i.e.*, proteins in a conformation able to play some functional role, by attaching/detaching other molecules to it. Yet another type of mechanisms control protein levels by affecting the rate it is being destroyed, called the *degradation* rate.

Some of the major mechanisms to regulate gene production are at the transcription stage illustrated in Figure **??**. These mechanisms are the research target of this dissertation. Transcriptional regulatory mechanisms involve proteins named *transcription factors*, or the *trans* regulatory elements, that attach to certain locations along the DNA sequence, thereby affecting transcription of nearby genes. Some transcription factors help initiate the transcription process while other factors help to enhance or repress the rates of the process, serving as *enhancers* or *repressors* respectively. Transcription factors attach to relatively short stretches of DNA, usually 5-20bp long. They can then

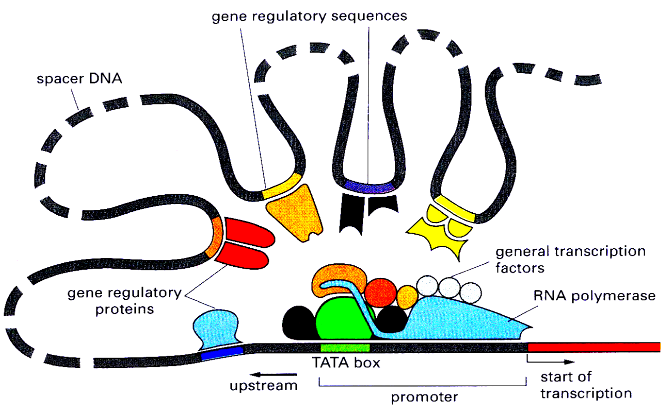


Figure 1.2: Illustration of the Polymerase complex with the transcription factors involved in transcription process.

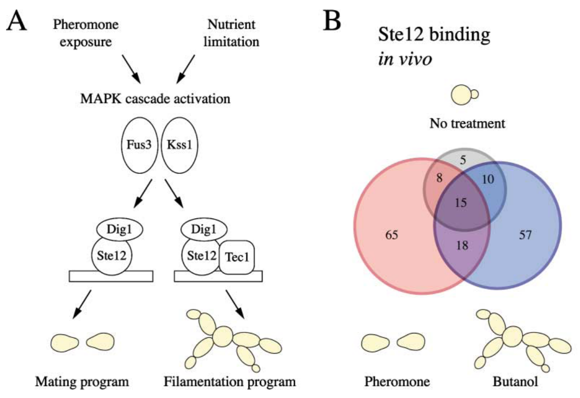


Figure 1.3: Illustration of regulatory programs involving STE12 in yeast. Taken from **?**.

form various 3D complexes that affect transcription, possibly by cooperating with other transcription factors or other mediator molecules that attach with them. The effect of these factors is obtained in many cases by a certain combinatorial combination under specific cellular conditions. For example, Figure **??** illustrates two regulatory mechanisms involving STE12 in yeast, as described by **?**. On the left side of part (A) we see a regulatory program that is part of the mating pathway. As a response to pheromone exposure stimulus, STE12 attaches to the promoter regions of a set of genes, regulating their expression. On the right side of part (A) of Figure **??** we see another regulatory program in the filamentation pathway. As a reaction to nutrient limitation, STE12, combined with another factor, TEC1, bind to another set of genes and regulates their production. In both cases a third factor is known to be involved, DIG1, but probably does not bind directly to DNA by itself. On part (B) of Figure **??** we see that these two programs involve only partially overlapping sets of genes, which in turn are both different then the set of genes STE12 regulates under normal growth conditions (see **?** for more details).

The short DNA subsequences that transcription factors bind to are termed *binding sites*. They might be characterized by a certain distance from the transcription start site (TSS), by a certain relative distance between binding sites of certain transcription factors acting together, etc. All these parameters, and probably many other we know little about, such as the position of the nucleosomes along the DNA, may affect the affinity of the transcription factor to bind and its consequent effect on the transcription process. The actual sequence at the binding site is of course of major importance in characterizing the binding site and the effect the transcription factor has. Transcription factors tend to bind to only certain, partially specific, subsequences that match (energetically speaking) their DNA binding domain. In many cases these short (5-20bp) binding sites can be found in long stretches of non coding areas called *promoter regions* though of course some binding sites can be found within the actual coding areas too. The distance between a transcription factor binding site and the code of the gene it helps regulate vary between few to several hundred base pairs. Characterizing these binding sites in order to understand the regulation process and infer about the functional rule of uncharacterized genes is a major objective in current molecular biology research, both experimental and computational.

Can we now hope to elucidate the regulatory mechanisms governing transcription? We turn now to describe the recent high throughput experimental methods that can help us in this task.

## 1.2 Experimental Methods

### 1.2.1 DNA Microarrays

The basis of the DNA microarrays technology is the phenomena of *hybridization*: a single strand RNA (or DNA) molecule will hybridize under the right conditions with a molecule containing a reverse complementary sequence (*i.e.*, a T/A matching each A/T and a C/G matching each G/C in the target sequence, as we reviewed about the double strand structure). A *hybridization probe* is a small area on a solid surface containing many copies of the reverse complement of a segment of a specific gene. The DNA microarrays, based on new advances in array inkjet printing technologies, contain a few thousands of these probes on a small solid surface, covering thousands of genes. In the microarray experiments the extracted mRNA from the studied sample is reverse transcribed into complementary DNA (cDNA) molecules, which are brought to contact with the probes microarray to hybridize. The idea is that the amount of hybridized molecules in the probe will be proportional to their amount in the sample. Because the cDNA created is also tagged fluorescently, we can quantitate the amount of hybridized molecules by measuring the intensity of the fluorescent signal.

There are two main microarray technologies currently in use. **cDNA microarrays** probes are made of cDNA molecules, each a few hundred bp long. With this technology experiments are usually conducted under two conditions: the condition of interest (*e.g.*, heat shock) and a reference condition (*e.g.*, normal cell growth conditions). Extractions from the condition of interest are fluorescently colored with one dye (either red or green) while extractions from the reference condition are colored with the second dye. The experiment then involves *competitive hybridization* where the two extractions compete in binding to the same probes and the relative ratio of green/red colors indicate up/down regulation in the condition of interest.

In the second *oligonucleotide* microarrays technology, each gene has several probes created syntheticly. The probes, only 25-80 bases long (depending on the exact technology used) matches a small area of the gene's code. In some arrays there is a set of probes corresponding to each gene (around 16 per gene), termed the gene's *probe set*. In case of the Affymetrix microarrays commonly used, each probe is actually made of two separate probes: one matching exactly the target subsequence, the other matching it but with one mismatch. The later is used to assess the amount of non-specific binding to the probes. Using a single fluorescent dye these microarrays experiments measure the amount of biding to each probe in the probe set and combine them together to assess the amount of the matching gene mRNA levels.

Microarray experiments enable researchers to measure expression of thousands of genes at the mRNA level under diverse conditions/tissues. By doing so and subsequently track the genes differentially expressed in each condition/tissue we can hope to reveal which genes are the target of transcriptional regulation and the genes involved in executing the regulation programs. For example, **?** measured the expression in yeast cells over diverse stress conditions such as amino acid starvation, heat shock (at various time points), etc. As an example of another type of study, **?** have tested the effect of mutating known regulators genes (snf,swi) on the expression profile of the whole set yeast's genes.

### 1.2.2 ChIP Assays

The *chromatin immunoprecipitation* (ChIP) assays enable an even more direct high throughput measure of the gene targets of known transcription factors, under specific cellular conditions, *in vivo*.

This genome-wide location analysis method, first introduced by **?**, allows protein-DNA interactions to be monitored across the entire yeast genome. It combines a modified Chromatin Immunoprecipitation (ChIP) procedure, used previously to study *in vivo* protein-DNA interactions at a small number of specific DNA sites, with DNA microarray technology. Briefly, cells are fixed with formaldehyde, harvested by sonication, and DNA fragments that are cross-linked to a protein of interest (*e.g.*, a transcription factor of interest), are enriched by immunoprecipitation with a specific antibody. After reversal of the cross-linking, the enriched DNA is amplified and labeled with a fluorescent dye using ligation-mediated PCR (LM-PCR). [3] Similarly, a sample of DNA that has not been enriched by immunoprecipitation is subjected to LM-PCR in the presence of a different fluorophore, and both IP-enriched and un-enriched pools of labeled DNA are hybridized to a single DNA microarray containing sequences of interest (*e.g.*, the long promoter regions in yeast where transcription factors are suspected to bind). The experimental procedure then assigns a $p$-value to the IP-enriched/unenriched ratio of fluorescence intensity (ChIP enrichment). According to the original experimental procedure of **?**, a sequence with a $p$-value less than $0.001$ is considered to be bound by the protein.

A recent paper by **?** included ChIP measurements for 204 transcription factors on all 6229 known yeast genes. This work serves as a valuable tool for revealing the transcriptional regulation

[3] Polymerase chain reaction (PCR), is a common method of creating copies of specific fragments of DNA. PCR rapidly amplifies a single DNA molecule into many billions of molecules.

mechanisms in the yeast cell, *in vivo*.

### 1.2.3 Limitations of Current Experimental Procedures

As promising as the these new technologies are, it is also important to realize their inherent limitations before one just tries to "read off" transcriptional regulation events from them.

Gene expression measurements gives us only a partial and very noisy pictures of what is happening in the cell. First, we must remember that adjusting the levels of transcription is only one of many ways a cell can control the amounts of active proteins in it. For example, it is possible that the levels of *active* molecules of a certain transcription factor may rise, causing a regulation program that uses it to be activated, but it will not be observed in the mRNA level of that transcription factor since this was done by phosphorylation of the transcription factor molecules already present in the cell. Also, expression adjustment is a dynamic effect that a single experiment, or even several, might fail to capture.

Another important drawback of gene expression experiments is the inherent level of noise in their output. Both variants of these experiments, *i.e.*, oligonucleotide and cDNA arrays, output real numbers which convey the image brightness of some fluorescent dye that binds to the relevant targets. The relation between the image brightness reported and the amount of mRNA in the cell is not exactly linear. The readings are influenced by a wide variety of factors. Some are related to the technology used (*e.g.*, variation in the target spot on the slide), others correlate to the experimental setup (*e.g.*, temperature), while others still are simple outcomes of the biological variability of genes (*e.g.*, cell cycle) that has nothing to do with the biological signal tested in the experiment. All this results in noisy data. In fact, some computational works in recent years (*e.g.*, **?**) have been aimed specifically to remove noise in these data sets. [4]

ChIP experiments, although valuable for inferring regulation events, are also limited. They are influenced from many chemical parameters too, and are suited for a limited number of proteins. In general, they can provide evidence about some binding events when they occur, while the fact that there is no ChIP indication of binding events cannot serve as an evidence for their absence. For example, ChIP experiments are carried out under specific cellular conditions. This means that a transcription factor *might* bind to a very different set of target genes under different conditions. This is illustrated in Figure **??**(B), were we see the overlap of STE12 targets under different conditions. Moreover, the experimental protocol of **?** was defined to set a conservative threshold over the $p$-value computed. By definition this means that many false negative errors might occur. To get a quantitative illustration of this problem and the noisy nature of ChIP data we plot in Figure **??** the

[4]Since some of our collaborators have been producing these data sets themselves we were interested in measuring the effect of these noise reduction algorithms. see **?** for more details.

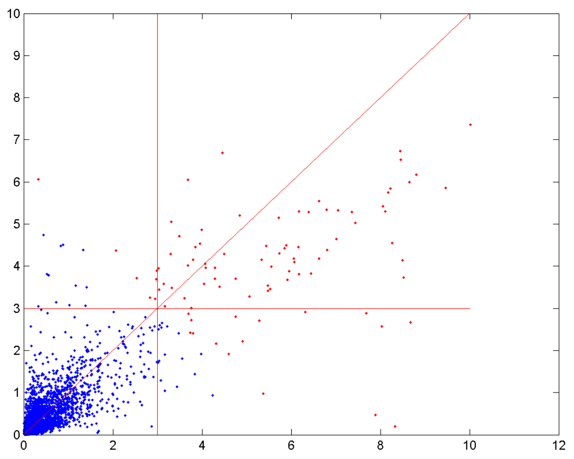


Figure 1.4: Scatter plot of $p$-values (after $-\log_{10}$ transformation) obtained for ChIP results in two experiments involving STE12 binding. The red lines designate the cutoff $p$-value of $10^{-3}$ used by the experimental protocol to designate putative targets. Red points are genes for which the *average* value over the two experiments is below the cutoff value. We found the target overlap between the two experiments to be 70% in this case, which is considered a good result for ChIP experiments. (Taken from raw data published by **?**).

$p$-value for genes identified as STE12 targets in two different experiments under the same cellular conditions. The target overlap (*i.e.*, genes with a ChIP $p$-value less then 0.001) is only about 70% for the better cases as the one illustrated here.

The take-home message of the above description is that when using these sources of data as input we must first be aware of their characteristics and limitations. We then must be able to develop computational methods well suited to handle this type of noisy data so as to extract the relevant signal from it and yield novel biological knowledge about the regulation processes involved.

## 1.3 Previous Analysis Approaches

Many computational works done on regulatory mechanisms focused solely on the task of finding the binding sites of transcription factors, using mostly promoter sequence data as input. These can be broadly divided to works that use only sequence data (*e.g.*, **?????**) and those who aim to find binding sites for which they have some prior knowledge/data, like a set of aligned sequences already known to serve as binding sites (*e.g.*, **?**). Works from the first category can be referred to as handling the *ab initio* motif finding problem, or motif finding using *unaligned* sequences. Similarity, works using known binding sites as samples can be referred to as solving the motif finding problem with *aligned* sequences.

Parallel to this, there have been several papers that suggested methods to exploit not just sequence data but combine it with expression and ChIP data to infer the regulatory mechanisms. Most of these works suggest a “step by step” approach to combine these sources of data. One of the most popular approaches has been to first cluster genes according to expression measurements and then use the resulting gene group definitions to search for common motifs in their promoters that might explain co-regulation (**?????**). We note that according to this approach the various tasks, such as motif search and clustering expression profiles, are decoupled. This means that an error made at one step (*e.g.*, misclassifying some genes) can lead to a problem in the second (motif finding) and cannot be corrected.

Our approach is quite different from the "step by step" approach just described. We turn to outline it now.

## 1.4 Our Approach

In contrast with the methods described above, our approach is *model based*, using probabilistic graphical models. It defines a *unified* probabilistic framework for all biological entities involved and all the data sources we use as input. The idea is that by building such a unified framework information can "flow" in the model and the evidence from various sources of data can be combined. For example, the cluster to which a gene is associated can be updated based not only on its expression profile but also on the regulatory motifs found in its promoter. This way, initially missed classifications based solely on expression data can be corrected during the learning of a model that combines both sources of data together.

In a nutshell, our approach involves the following: We model the various entities in the domain and their properties as *random variables*. By definition, some of the properties are observed (*e.g.*, sequence) and some are hidden (*e.g.*, active regulators). Together these random variables along with the dependencies between them define a *Bayesian network* model that assigns a probability distribution over the domain. Our learning machinery tries to optimize the probability defined by this model so as to best match the observed data. This optimization involves two components: the model's *structure* and the model's *parameters*. The first defines the direct dependencies between the random variables (*e.g.*, which regulators are involved in a certain regulatory program) and the second one defines the exact functional description of these dependencies (*e.g.*, what is the expected expression profile of genes controlled by a specific program).

Of course, optimizing the model by simply testing all possible model configurations is not practical. In order to achieve good results we need to handle several challenges. First, we need to decide how to formally model the entities in our domain. One must bare in mind the model we learn is just a mathematical proxy to the real biological relations involved. For example, we can not model time effects directly when we have no such data available, but we might want to introduce a hidden variable into our model to compensate for time effects. Then we need to decide how to constrain the model in a way that would best match the biological mechanisms and reflect prior knowledge. This can involve, for example, choosing only a subset of genes to be considered as regulators involved in a process instead of testing all the thousands of genes we have in the genome. Even after all these prior "human" definitions are given, the machine learning task is not trivial. It usually involves thousands of variable and sparse, noisy data. This means we also need to develop *efficient* algorithms for the learning task at hand. Finally, there is a major question about the biological meaning of the results we get. There are two types of results: qualitative (*e.g.*, which regulator control which gene) and quantitative (*e.g.*, prediction expression given the gene's promoter sequence). For both types of results we need to develop methods to assess how confident we are in our predictions and how well our algorithms are doing.

The specific tasks for which we develop our models include:

1. **Identifying transcription factors binding sites.**
   The identified binding sites will enable us to conclude on the role of known transcription factors, suggest novel transcription factors not yet identified, and infer about the functionality of unknown genes through the identification of the regulatory mechanisms governing their expression levels.

2. **Understanding combinatorial interactions between different transcription factors.**
   Identification of these interactions is vital in advancing our understanding how regulatory mechanisms are activated and adopted in different cellular conditions.

3. **Predicting expression profiles directly form sequence data.**
   Prediction of expression profiles by our models serves as a validation for their accuracy and allows us to compare performance of various learning algorithms and models in a rigorous way. Furthermore, these predictions can be used to direct us in designing future experiments. Given finite funding and time, we can use expression predictions to avoid some (expensive) experiments for which we have very good prediction confidence. Similarly, we can avoid a research effort where evidence indicates inherent randomness. Finally, we can decide to extend or repeat some experiments, when our learned models indicate this to be necessary.

4. **Identifying "modules" of co-regulated genes and inferring how they are controlled under specific biological conditions.**
   "Modules", or functional complexes, identified by the model can be linked to known or novel regulatory pathways in biological processes. This will enable us to move into a higher level understanding of the regulatory processes. For example, we can gain insights on regulatory mechanisms in different cellular conditions and tissues, or mechanisms related to specific diseases (like the silencing of some regulatory pathway due to a genomic mutation).

There are several important advantages to our model based approach for handling these tasks. First, the probabilistic setting we use is naturally appealing given the noisy and stochastic nature of the data. *Bayesian networks* (**?**) and *PRM* (Probabilistic Relations Models) (**??**) offer a formal modeling language which is particularly suitable for learning with sparse and noisy data. Such models are especially suitable for representing and reasoning about complex systems where each variable is dependent on only relatively few other variables, but the large amount of variables in the system and the network of dependencies between them results in a very complex picture. It is enough to think of each gene mRNA production as a variable, the other genes that affect it as the dependency structure and the total mRNA expression profile in the cell as the complex picture we observe to understand the natural appeal of these modeling languages for our domain.

Another benefit of our approach is the fact we do not need to develop all the theory and algorithms from scratch. We can utilize many of the techniques developed in the field of machine learning and rely on their sound theoretical foundations. Our modeling approach also carries the benefit of being intuitive for this domain. Modeling the biological entities via random variables and their relations via dependency structure makes it easier to introduce prior biological knowledge. For example, we can constrain a model to have specific regulators affecting certain genes under some cellular conditions if they were already proved to do so. Similarly, hypotheses generated from these type of models can be relatively easily interpreted.

Another advantage of our approach is its *modularity*. It gives us the possibility to improve parts of our model whenever we find the need and then integrate these relatively easily with other model parts, creating an improved unified model. For example, we show how to learn discriminative binding sites in Section **??** and then incorporate the same model over sequence along with the matching learning algorithm into a higher level model for regulatory programs in Section **??**.

It is important to understand that our modeling approach, although aimed to model regulatory programs, is not confined to modeling just them. As we show, using our approach we can model the actual binding sites of transcription factors with Bayesian networks. In this case the variables in our domain are the binding site positions and we are able to learn dependencies between them. Our modular framework means we can then use such improved binding sites models in the learning of more accurate models for regulatory interactions.

Finally, we point out that our approach can be characterized as "sequence oriented". In practical terms, all the works we present in this dissertation involve genomic sequence as part of the input. In a more conceptual sense, we aim to build models that explain the observed expression profiles in terms of codes of regulatory programs found in the DNA sequence. The modular nature of our approach enable us to first address the task of motif finding (from both *aligned* and *unaligned* sequences), and then incorporate the solutions we develop to infer regulatory programs in higher level models. This modular nature is also reflected in the structure of this dissertation, as we outline below.

## 1.5 Outline

Since our approach is sequence oriented we start by developing methods to improve identification of transcription factors binding sites. In Chapter **??** we describe the motif finding problem where a set of sequences suspected to be co-regulated are given as input and the goal is to accurately predict binding sites locations of transcription factors that control these genes. We present a novel *discriminative* approach, implemented in the *SeedSearcher* and the *LearnPSSM* algorithms, to efficiently handle this task. As we describe, our algorithm tries to learn sequence patterns that are not only *common* to the given set of sequences, but also *unique* to them. Learning these patterns is made of two stages. First, using the *SeedSearcher* algorithm, we find relatively simple patterns and score the statistical significance of their appearance in the given set of sequences. The second stage involves refining the representation of these patterns using the *LearnPSSM* algorithm.

In Chapter **??**, we present an alternative way to solve the motif finding problem. Using the same initial stage of finding discriminative motifs with the *SeedSearcher*, we now show how to learn more complex motif models that capture dependencies between the binding site positions. In Chapter **??** we describe a systematic evaluation of these algorithms and show the gain from using our discriminative approach and the binding site models with interdependent positions.

Since our motif finding algorithms use a probabilistic framework we can later incorporate them into our models for regulatory modules, presented in Chapter **??**. We present two very different models, that reflect two very different approaches for modeling regulatory modules. The first, *NBClust* is a more *generative* approach that tries to group genes into cluster that share common expression profiles as well as a common profile of motifs in their promoter. The other model, *DiscRegProg*, is much more complex and integrates genomic sequence, expression measurements and ChIP assays in a unified probabilistic model. It reflects a more *discriminative* modeling approach, as it tries to explain gene expression via (hidden) regulators controlling it under specific experimental conditions. As we show, the *LearnPSSM* algorithm, is incorporated as part of the learning process of the *DiscRegProg*. We conclude with a discussion and point to future directions in Chapter **??**.

# Chapter 2

# Discriminative Motif Finding

## 2.1 Background

Much of the specificity in transcription regulation is achieved by combinatorial combinations of *transcription factors*. Recall that these regulators are proteins that, when in the suitable state, can bind to specific DNA sequences. A combination of such bound/unbound transcription factors to a gene's promoter sequence that affect its transcription level can be thought of as a regulatory program. The direct connection between the *trans* elements (*i.e.*, transcription factors) and their matching *cis* elements (*i.e.*, binding sites), makes the immediate aim of finding the binding sites naturally appealing if one is to infer about the regulatory programs governing the cell's activity. This part of the dissertation is therefore dedicated for elucidating the binding sites of transcription factors. In the section that follow we develop higher level models that incorporate the machinery developed here to learn the actual regulatory programs.

Finding the exact binding sites of unknown transcription factors in long stretches of DNA sequence is computationally hard. Such factors tend to bind to relatively short, only partially specific, subsequences about 5-15bp long. However, recent advances in experimental technologies can come to our aid by defining groups of genes suspected to be co-regulated. Microarray experiments allow to characterize groups of genes that have similar expression patterns across a wide range of conditions (**?**). Arguably, the simplest biological explanation of co-expression is co-regulation by the same transcription factors. ChIP experiments allows us to measure directly the binding of some known regulators to thousands of DNA promoter regions. In prospect of the motif finding problem, all these experiments can be viewed as producing groups of sequences in which we would like to find common sequence patterns that serve as putative binding sites of regulatory elements *i.e.*, transcription factors.

This observation sparked several works on *in-silico* identification of putative transcription factor binding sites (**?????**). Assuming such groups of genes are either given or inferred in a first stage, they search for short DNA patterns *common* to the promoter region of the genes in each particular group. We term this computational task the *ab initio* motif finding problem, because there is no prior knowledge about the binding site (like previous known examples of it). Many of the works on *ab initio* motif finding were based to a large extent on methods that were developed to find common motifs in protein and DNA sequences. These include combinatorial methods (**?????**), parameter optimization methods such as Expectation Maximization (EM) (**?**), and Markov Chain Monte Carlo (MCMC) simulations (**??**). See **?** for a review of these lines of work.

The main drawback of these methods is that they try to optimize motifs which are only *common* to a given group of sequences. As it turns out, in many cases there are motifs *common* to a group but are far from being *unique* to it. In yeast for example there are long poly A/T subsequences, common in gene promoters. Such motifs may have some functional rule (*e.g.*, **?**) but they are not good candidate of unique regulation of a specific set of genes in specific cellular conditions.

To answer this problem we suggested a novel approach for motif finding based on *discriminative* considerations. The idea is to find motifs which best *discriminate* a given set of sequences from a larger pull of "negatively" labeled set. An important underlying concept is that we can use the information from the negative set to extract the relevant bit of information found in the "positive" group of interest.

When reviewing solutions to the *ab initio* motif finding problem, one should reflect on the following:

1. How are the motifs defined? How similar they are to "real" motifs ? What type of sites are they able to capture?
2. How are motifs evaluated/scored?
3. What guarantees can be given about the results (*e.g.*, is a global optimum in the defined search space reached?)
4. How is the motif space searched ? How efficient is it?
5. Does the method suggest a way to assess the statistical significance of the results reported ?
6. What assumptions are made by the algorithm about the motifs, in input and the nature of the regulatory program?
7. What are the input and output? How can the output be used for binding sites predictions?

Because so many algorithms have been developed in recent years to solve the motif finding problem, using the above list as reference points enables us to better understand what characterize a specific solution and how it differs from others. We will refer to all of these points in turn when we present our algorithms for motif finding. We start by presenting the *SeedSearcher*. This algorithm enables us to efficiently search an enumerable space of motif *seeds* that are like partial sequence patterns of the "real" biding sites motifs. Moreover, it enables us to estimate the statistical significance of our results. However, the seeds are still a crude representation of real motifs, not suitable for high accuracy genomic scan for binding sites. Towards this end, we show how the promising seeds can be used as an excellent starting point for learning a more expressive binding site model named *PSSM*. PSSM (**P**osition **S**pecific **S**core **M**atrix) is the most commonly used model for binding sites. Section **??** reviews the PSSM model and introduces a discriminative probabilistic algorithm to learn PSSMs from positively (*i.e.*, presumably regulated) and negatively (*i.e.*, presumably non regulated) labeled sequences. For practical usage of such algorithms, when performing genomic wide scans for target genes one needs a sound criterion for declaring a certain subsequence as a putative binding site. We develop in Section **??** an efficient algorithm to assess the statistical significance of putative sites, given a very general definition for a probabilistic background model. We evaluate the results of *TestPSSM* before we conclude with a discussion in Section **??**. The evaluation of the algorithms for motif finding presented here is postponed to Section **??**, after we develop the related *LearnMotif* algorithm in the next chapter.

## 2.2 *SeedSearcher*: Finding Discriminative Sequence Patterns

### 2.2.1 Preliminaries - Scoring Patterns for Significance

Suppose we are given a set of genes $\mathcal{G}$. Ideally, these are all the known and putative genes in a genome. With each gene $g \in \mathcal{G}$ we associate a promoter sequence[1] $S[1], \ldots, S[M]$. For simplicity we assume that each of these sequences is of the same size, $L$.

Suppose we are now given a subset of genes $G \subset \mathcal{G}$ suspected to be co-regulated by some transcription factor (*e.g.*, based on clustering of co-expressed genes). Our aim is to find patterns in the promoter region of these genes that can be considered as putative binding sites of any factor. The assumption being that the co-regulation is mediated by factors that are present in most of the genes in group $G$, but overall rare in $\mathcal{G}$. Thus, a pattern is considered significant if it is characteristic of $G$ compared to the background $\mathcal{G}$.

Before we discuss what constitutes a pattern in our context, we address the basic statistical definition of a characteristic property. Suppose we find a pattern that appears in the promoter sequences of several genes in $G$. How do we measure the significance of these appearances with respect to $\mathcal{G}$? A related question one may ask, is whether the set $G$ is significantly different, in terms of the composition of its upstream region, from $\mathcal{G}$.

[1]Or an upstream region that best approximates it, when the transcription start site is unknown.

For now, we concentrate on events occurring in the promoter region of a gene. We focus on *binary* events, such as "$S[m]$ contains the subsequence ACGTTCG or its reverse complement". Alternatively, one can consider *counting* the number of occurrences of an event in each promoter sequence, *e.g.*, "the number of times the subsequence ACGTTCG appears in $S[m]$". The analysis of such counting events, while attractive in our biological context, is more complex, in particular since multiple occurrences of an event in a sequence are not independent of each other. See **??** for approximate solutions to this problem.

Formally, a binary event $E$ is defined by a *characteristic function* $I_E : \{A, C, G, T\}^\star \to \{0, 1\}$, that determines whether that event occurred or not in any given nucleotide sequence. Given a set $G$, we define $\#_E(G) = \sum_{m \in G} I_E(S[m])$ to be the number of times the event $E$ occurs in the promoter regions of group $G$. We want to assess the significance of observing the event $E$ at least $\#_E(G)$ times in $G$, when taking the set of genes $\mathcal{G}$ as the background for our decision.

There are two general approaches for testing such significance. In both cases we compute $p$-*values*: the probability of the observations occurring under the *null-hypothesis*. This value serves as a measure of the significance of the pattern - the lower $p$-value is, the more plausible it is that an observation is significant, rather than a chance artifact. The two approaches differ, however, in the nature of each null-hypothesis.

### Random Sequence Null Hypothesis

In this approach, the null hypothesis assumes that the sequences $S[m]$ for $g \in G$ are generated from a background sequence model $P_{BG}(S)$. This background distribution attempts to model "prototypical" promoter regions, but does not include any group-specific motifs. Thus, if the event $E$ detects such special motifs, then the probability of randomly sampling genes that satisfy $E$ is small.

The background sequence model can be, for example, a Markov process of some order (say 2 or 3) estimated from the sequences in $\mathcal{G}$ (or, preferably, from $\mathcal{G} - G$). Using this background model we need to compute the probability $p_E = P_{BG}(I_E(s) = 1)$ that a random sequence of Length $L$ will match the event of interest. Now, if we also assume under the null hypothesis that the $n$ sequences in $G$ are independent of each other, then the number of matches to $E$ in $G$ is distributed $Bin(n, p_E)$. We can then compute the $p$-value of finding $\#_E(G)$ or more such random sequences by the tail weight of a Binomial distribution.

The key technical issue in this approach is computing $p_E$. This, of course, depends on the assumed form of the background distribution, and on the complexity of the event. However, even for the simple definition of a pattern as an exact subsequence (*i.e.*, $I_E(s) = 1$ iff $s$ contains a specific subsequence) and background probability of the form of an order 1 Markov chain, the required computation is not trivial. This forces the development of various approximations to $p_E$ of varying accuracy and complexity (**???**). We develop an algorithm to approximate this *p-value* for more general PSSM motif models in Section **??**.

### Random Selection Null Hypothesis

Alternatively, in the approach we focus on here, one does not make any assumption about the distribution of promoter sequences. Instead, the null hypothesis is that $G$ was selected at random from $\mathcal{G}$, in a manner that is independent of the contents of the genes' promoter regions.

Assume that $K = \#_E(\mathcal{G})$ genes out of $N = |\mathcal{G}|$ genes satisfy $E$. The probability of this observation under the null hypothesis is the probability of randomly choosing $n = |G|$ genes in such a way that $k = \#_E(G)$ of them include the event $E$. This is simply the *hyper-geometric* probability of finding $k$ red-balls among $n$ draws without replacement from an urn containing $K$ red balls and $N - K$ black ones:

$$P_{\text{hyper}}(k \mid n, K, N) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} \tag{2.1}$$

The $p$-value of the observation is the probability of drawing $k$ or more genes that satisfy $E$ in $n$ draws. This requires summing the tail of the hyper-geometric distribution

$$\textit{p-value}(E, G) = \sum_{k'=k}^{n} P_{\text{hyper}}(k' \mid n, K, N) \tag{2.2}$$

The main appeal of this approach lies in its simplicity, both computationally and statistically. This null hypothesis is particularly attractive in the post-genomic era, where nearly all promoter sequences are known. Under this assumption, irrelevant clustering selects genes in a manner that is independent of their promoter region.

### Dealing with Multiple Hypotheses

We have just defined the significance of a single event $E$ with respect to a group of genes $G$. But when we try many different events $E_1, \ldots, E_M$ over the same group of genes long enough, we will eventually stumble upon a surprising event even in a group of randomly selected sequences, chosen under the null hypothesis.

Judging the significance of findings in such repeated experiments is known as *multiple hypotheses testing*. More formally, in this situation we have computed a set of $p$-values $p_1, \ldots, p_M$, the smallest corresponding to the most surprising event. We now ask how significant are our findings considering that we have performed $M$ experiments.

One approach is to find a value $q = q(M)$, such that the probability that any of the events (or the smallest one) has a $p$-value less than $q$ is small. Using the union bound under the null hypothesis we get that

$$P(\min_m p_m \leq q) \leq \sum_m P(p_m \leq q) = M \cdot q$$

Thus, if we want to ensure that this probability of a false recognition is less than $\alpha = 0.01$ (*i.e.*, 99% confidence), we need to set the *Bonfferoni* threshold to be (*e.g.*, **?**):

$$q = \frac{\alpha}{M} \tag{2.3}$$

The Bonfferoni threshold is strict, as it ensures that each and every validated scoring event is not an artifact. Our aim, however, is a bit different. We want to retrieve a set of events, such that *most* of them are not artifacts. We are often willing to tolerate a certain fraction of artifacts among the events we return. A statistical method that addresses this requirement is the *False Discovery Rate* (FDR) method of **?**. The FDR procedure controls over the expected ratio of false positive identifications from the total number of hypotheses reported as true (positive). Formally:

$$F(\alpha) = E_p(\frac{FP}{P}) = \alpha$$

Where $FP$ is the number of false positive identifications, $P$ is the total number of hypotheses reported as true and $p$ is the overall distribution over hypotheses, either true or false, we observe.

As **?** have shown, to control FDR we need to do the following. First, we sort the events by their observed $p$-values, so that $p_1 \leq p_2 \leq \ldots \leq p_M$. We then return the events $E_1, \ldots, E_k$ where $k \leq M$ is the maximal index such that $p_k \leq \frac{kq}{M}$ and $q$ is the FDR level we are willing to tolerate. Note that the Bonfferoni and FDR thresholds coincide only for the hypothesis with the best $p$-value. For the other $p$-values the FDR test is less stringent, and the final threshold is set to the *maximal* $k$ (*i.e.*, the worse $p$-value) that still holds the above equation. More importantly, by switching from Bonfferoni to FDR we replaced a strict validation test of *single* events, with a more tolerable version validating a *group* of events. We may now detect significant patterns, weaker than the most prominent one, that were previously below the threshold computed for the later.

### 2.2.2 Defining the Search Space - Motif Representation

When defining the representation of binding site motifs two main considerations should be taken. The first is that the motifs searched should resemble “real” binding sites motifs. The second is that the search space resulting from this definition could be searched efficiently. Of course these two considerations are hard to satisfy. In general, a more expressive binding site representation would

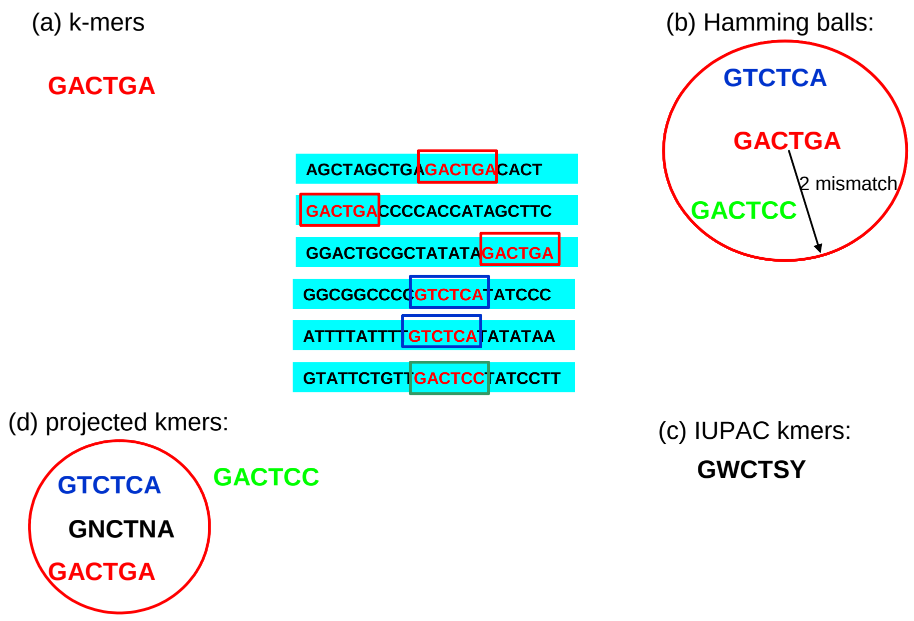


Figure 2.1: Illustration of possible motif definitions which try to capture the "real" binding sites given in the center.

usually result in a larger motif search space which is harder to explore. We therefore take the following approach. We first search motifs using representations which can be efficiently explored. In a second stage, we refine the highest scoring motifs using a more complex representation. This second stage will be the subject of Section **??**. Here we will describe how we define *crude* representations of binding site motifs. In the next section we will review how we efficiently explore the search space defined by these crude motif representations.

Figure **??** illustrates various patterns definitions that try to capture a real motif. The set of subsequences in the center represent a set of promoters we suspect to be co regulated. The known binding sites, verified by experimental procedures, are colored. Each color corresponds to a slightly different $k$-mer that serves as a binding site. Of course in the *ab initio* motif search we do not have these exact locations at hand but rather just the entire promoter sequences. These binding sites will serve as the "biological truth" *i.e.*, the real motif we try to capture. Figure **??**(a) tries to capture the sequence motif with a specific $k$-mer of the matching length. Of course this representation is very simplistic and would overlook the other $k$-mers (colored green and blue) that serve as a binding site. However, in some cases the score of just this $k$-mer, as reviewed in previous section, would be high enough to appear significant so that we would later be able to refine this simple $k$-mer motif using a more complex motif representation. Formally, it means we evaluate all $\{A, C, G, T\}^k$ $k$-mers as possible motifs and score them using the scheme we developed in previous section. Usually such a search is done for $K_1 \leq k \leq K_2$ for some user defined length limits $K_1, K_2$. Another possible motif definition is by defining it to be "variations" over a certain "core" $k$-mer. Formally, given some metric of distance between $k$-mers $\sigma(\mathbf{x}, \mathbf{x}'), \mathbf{x}, \mathbf{x}' \in \{A, C, G, T\}^k$ we define the motif corresponding to $\mathbf{x}$ to be all the $k$-mers which are no more then $C$ distant from the "core" $k$-mer $\mathbf{x}$. In principle prior biological knowledge can be poured into the definition of $\sigma$, (*e.g.*, allowing the mid area to be less conserved). Given no such prior knowledge, we usually use the *hamming distance* to define $\sigma$. This is illustrated in Figure **??**(b). As we can see, using hamming distance as $\sigma$ with $C = 2$ gives us a motif around the "core" GACTGA that covers all the $k$-mers that serve as binding sites, but it also covers many other "false" $k$-mers. One main reason for this is that this definition, without any prior knowledge, treats all positions of the binding site on an equal basis. In contrast, experimental evidence show binding sites tend to have some positions more conserved then others. The motif representation suggested in Figure **??**(c) tries to remedy this by using the much more expressive IUPAC alphabet. The IUPAC alphabet represent all subsets of the basic DNA alphabet $\{A, C, G, T\}$. This means in more conserved positions we can use an IUPAC letter that stands for only a small subset of the DNA letters (like $W = \{A, T\}$ in position 2 in the figure). comparing Figure **??**(b) and (c), the motif GWCTSY still includes all the binding site $k$-mers while allowing much fewer "false" $k$-mers to be included in the motif definition (*e.g.*, the "false" GTCTGA subsequence is still included in this definition). However, the search space now has grown from $4^K$ in the case of short subsequence to $16^K$.

An "in between" possible solution is to use a motif representation we term *projected $k$-mers*. In this case we use subsequence of length $K_1 \leq k \leq K_2$ over the alphabet $\{A, C, G, T, N\}^k$ where $N$ is a wild card matching any of the DNA bases. The number of wildcards is set to be in a certain range $D_1 \leq d \leq D_2$ and their locations are set to some $\vec{d} = \{i_1, \ldots, i_d\}$ where $1 \leq i_j \leq k, \forall j \in \{1, \ldots, d\}$. The term projected $k$-mers comes from the fact we project vectors of length $k$ into a $k - d$ space, an idea first suggested in the context of motif search by **?**. Formally, the projection operator $\mathcal{D}_{k,\vec{d}}$ projects any $k$-mer $\mathbf{x}$ so that:

$$\mathcal{D}_{k,\vec{d}}(x_i) = x_i, \forall i \notin \vec{d}$$
$$\mathcal{D}_{k,\vec{d}}(x_i) = N, \forall i \in \vec{d}$$

This definition puts $\mathbf{x}$ and $\mathbf{x}'$ to belong to the same projected motif if and only if they match exactly in the non wildcards positions. In our example in Figure **??**(d) we see that the projected $k$-mer motif GNCTNA, using only two "wild card" N's in positions (2,5), covers 5 of the 6 "real" binding sites that occur in the data set. In general, $k$ and $d$ are user defined, usually in the range of $5 \leq k \leq 14$ and $0 \leq d \leq 5$. Typically, a search for motifs involves a subset of these ranges. We note that for each $k, d$ setting there are $\binom{k}{d}$ possible ways to choose $\vec{d}$. To search for motifs, a set of $\{k, \vec{d}\}$ is chosen for each $(k, d)$ setting. This set can either be exhaustive (for moderate $(k, d)$ combinations), or randomly selected, as part of the programs settings. Also, the *SeedSearcher* allows to use prior knowledge to constrain the $k, \vec{d}$ settings (*e.g.*, $d_1 < d$ of the wild cards be set in the middle).

Obviously, all the "crude" motif definitions we now covered corresponds to different levels of complexity. Yet all are in an enumerable space in which patterns can be evaluated as defined in Section **??**. The *SeedSearcher* algorithm permits all these definitions to be used. However, in practice we found the projected $k$-mers to be most useful, giving a good tradeoff between computational efficiency and expressiveness. Intuitively, projected $k$-mers proxy real binding sites were there is usually a set of relatively conserved positions and a few positions in which variations are more tolerable. For the sake of clarity, we will therefore assume from now on that the crude motifs we search and use are projected $k$-mers.

### 2.2.3 Efficiently Searching the Motif Space

As we have just described, using the "projected $k$-mers" motif representation to search for statistically significant motif in a given $G \subset \mathcal{G}$ involves a set of $\{k, d\}$ and for each $(k, d)$ setting a set of specific $\{k, \vec{d}\}$ projections to try. We therefore need an efficient way to find and score all motifs we find for any $(k, \vec{d})$ definition. In this section we review how the *SeedSearcher* algorithm search and score all possible motifs under a $\{k, \vec{d}\}$ definition.

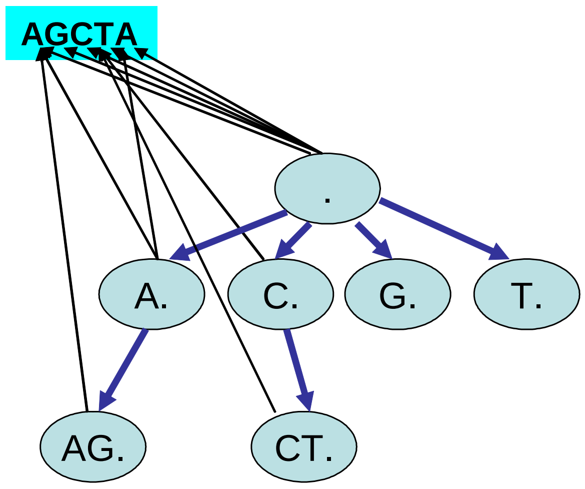


Figure 2.2: Schematic representation of the prefix tree data structure used by the *SeedSearcher*.

As a preprocessing stage we first parse the input sequence data and represent all the sequences of $\mathcal{G}$ in a prefix tree $\mathcal{T}$. This representation was originally developed by **??**. Briefly, it means we build a tree where each node corresponds to a different sequence prefix. The root is the empty string. Its four immediate descendents correspond to the four possible DNA prefix strings that can be derived from it: $A.$,$C.$,$G.$ and $T.$. Each node of $\mathcal{T}$ holds pointers to all sequence positions with the matching prefix. This is illustrated in Figure **??**. For simplicity we draw only one short sequence and only some of the nodes and pointers. In practice $\mathcal{T}$ includes all of $\mathcal{G}$ and its depth matches the motif length. If we search for motifs of length $K_1 \leq k \leq K_2$ then $\mathcal{T}$ maximal depths is set to be $K_2$.

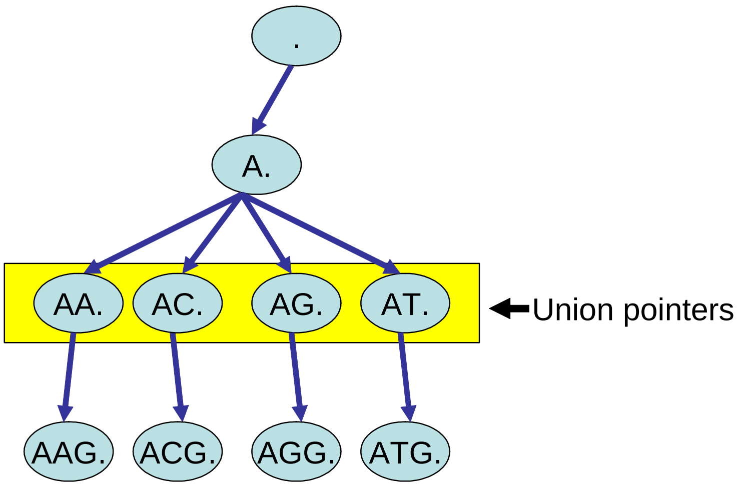


Figure 2.3: Schematic representation of the prefix tree traverse resulting in the motif ANG.

By traversing $\mathcal{T}$ we can efficiently find all the projected $k$-mers motifs in our dataset and score them. Let us start with the simpler case were $\vec{d} = \{\}$ *i.e.*, , no positions were chosen to be projected. In this case we need only evaluate the score of all $k$-mers, for some range $K_1 \leq k \leq K_2$. According to $\mathcal{T}$ definition, each node in depth $k$ is a specific $k$-mer, which we can score immediately since we have at the node all the pointers to all positions of this $k$-mer, along with the matching sequence label (either it belongs to $G$ or not). This means that by a single traverse of the $\mathcal{T}$ up to depth $K_2$ we can score all relevant $k$-mers. Moreover, we can prone the tree by avoiding any branch (*i.e.*, prefix) which does not have any pointers of sequences in $G$.

When we actually need to handle projected positions *i.e.*, $\vec{d} \neq \{\}$ traversing the tree becomes somewhat more involved. We illustrate this in Figure **??** for $\vec{d} = \{2\}$ and the motif "ANG". In principal, if $\vec{d} = \{i\}$ we need to combine all the paths that have the same prefix sequence up to positions $i$ (*i.e.*, the same path up to a node in depth $i$) and the same suffix in positions $i+1, \ldots, k$ (*i.e.*, the same relative path from the node at position $i$). All the pointers at the end nodes of these paths are *unionized* and the score of the motif can be thus computed. In our example all the paths share the prefix $A.$ in position $1$ and the suffix $.G$ in position $3$. By unionizing the pointers in the paths matching this description we can compute the score of the motif "ANG". The generalization to any $\vec{d}$ is straightforward and involved traversing $\mathcal{T}$ in order while keeping track of the union operation performed.

### 2.2.4 Extensions to the *SeedSearcher* algorithm

Through the past few years, the *SeedSearcher* has become a major tool in identifying putative binding sites in numerous research projects we were involved in. In some cases we wanted to relax some of our basic assumptions in motif searching. For example, in some cases it might be beneficial to test not just the number sequences which contain a motif but also the total amount of motif occurrences we identify. In other cases we might have prior knowledge about specific areas along the promoter were a binding site is more likely. We therefore extended the SeedSearcher's basic definition and options and incorporated more input sources into it. Many of these extensions remain as part of future research and testing. We review some of these briefly here.

1. **Total counts** - Instead of summing over binary events

   $$\#_E(G) = \sum_{m \in G} I_E(S[m])$$

   *i.e.*, counting the number of promoters in $G$ in which $E$ occurs, we now use

   $$\#_E(G) = \sum_{m \in G} N_E(S[m])$$

   Where $N_E(S[m])$ is the *maximal non overlapping number of occurrences* of $E$ in $S[m]$ (Overlapping occurrences are ignored to reduce motif's autocorrelation). This setting can help raise significance of motifs with multiple occurrences in promoter regions, but is more sensitive to common low complexity regions in promoters.

2. **Partial counts** - Instead of having each sequence $\mathbf{S}[m]$ considered positive (*i.e.*, belong to $G$) or negative (not in $G$), we have a *weight* $W(m) \in \{0, 1\}$ attached to each sequence. This can be used for example to incorporate prior belief of regulation. In this case we still use the discriminative hyper geometric $p$-value, rounding the counts (which are real numbers now, instead of integers). We note this score does not carry the original statistical meaning in this case.

3. Positional bias priors - Just as we are able to have weights $W(m) \in \{0, 1\}$, we can also have another multiplicative factor $W(m, i) \in \{0, 1\}, i \in 1, \ldots, L$ for any position along each promoter. This allows us to combine for example prior knowledge about positional preference for binding site location based on phylogenetic comparison or the new ChIP tilling arrays. The score is computed just as in the partial counts setting.

4. **Exponential scores** - We experimented with alternative discriminative scores motivated from

machine learning online algorithms (*e.g.*, **?**). We have already defined

$$\#_E(G) = \sum_{m \in G} I_E(S[m])$$

for the number of times a motif appeared in $G$'s promoters. Similarly, we can also define:

$$\#_E(\mathcal{G} \backslash G) = \sum_{m \in \mathcal{G} \backslash G} I_E(S[m])$$

*i.e.*, the number of the motif appeared in other promoters. We then define the exponential score of the motif to be:

$$\alpha^{\#_E(G)} \beta^{\#_E(\mathcal{G} \backslash G)}$$

Where typically $\alpha > 1$ and $\beta < 1$. This means we boost our belief in the motif as regulation predictor each time it predicts $G$ correctly and penalize it each time it makes a false positive mistake. This type of score is good to find a set of low complexity crude motifs that can be later combined to predict regulation.

## 2.3 *LearnPSSM*: Discriminative Learning of PSSM models

In previous section we introduced the *SeedSearcher* algorithm to quickly and efficiently find "crude" representations of transcription factors binding sites. In this section we describe how we can use these "crude" motif approximations as starting points to learn a refined binding site model, using a discriminative machine learning approach.

We start by reviewing the PSSM binding site model and its possible benefits over the more "crude" motifs we used before. We then turn to define the probabilistic framework we use to search for PSSM motifs. Given this framework we next introduce our discriminative approach to learn PSSM motifs as implemented in our *LearnPSSM* algorithm.

### 2.3.1 The PSSM Motif Model

PSSM are currently the most commonly used models for transcription factors binding sites. Formally, a PSSM (or a **P**osition **S**pecific **S**core **M**atrix) model is a matrix $A_i[a], 1 \leq i \leq k, a \in \{A, C, G, T\}$, representing the "score" of having *any* subsequence of length *exactly* $k$ as the binding site. The score of $S = S^{(1,\ldots,k)}$ where $S^{(j)} \in \{A, C, G, T\}$ is given by the matrix as:

$$\mathcal{S}(S) = \sum_{i=1}^{k} A_i[S^{(i)}] \tag{2.4}$$

We note that according to this model the score for each position is independent. In a probabilistic framework the PSSM entries are the probabilities of observing base $a$ at position $i$ so the matrix satisfies $A_i[a] >= 0, \forall i, a$ and $\sum_a A_i[a] = 1, \forall i$. The score of $S$ in this setting is the probability of observing this $k$-mer at the binding site. A related and commonly used interpretation of $A_i[a]$ is as the *log odds* score of $S$. In this setting we assume $S$ can come either from a probabilistic PSSM binding site model or from a Markov order 0 background model $BG$ for sequence generation. The entries $A_i[a]$ in this case represent the log ratio of the probability of observing $S^{(i)} = a$ by $BG$ and by the PSSM model:

$$\mathcal{S}(S) = \sum_{i=1}^{k} A_i[S^{(i)}] = \sum_{i=1} \frac{\psi_i(S^{(i)})}{BG(S^{(i)})} \tag{2.5}$$

Where $\psi_i(a)$ is the probability of $a$ in position $i$ by the motif model. A commonly used graphical representation of PSSM models is by *Sequence logo*, first suggested by **?**. An example of it is given in the top right picture of Figure **??**. Intuitively , a PSSM sequence logo enables to observe the amount of conservation in each position. Formally, this amount of conservation is represented by the hight $h$ of the letter $a$ in position $i$ so that:

$$h_i(a) = H(U) - H(\psi_i) \tag{2.6}$$

Where $H$ is the *Entropy* [2] associated with the distribution, and $U$ is the uniform distribution.

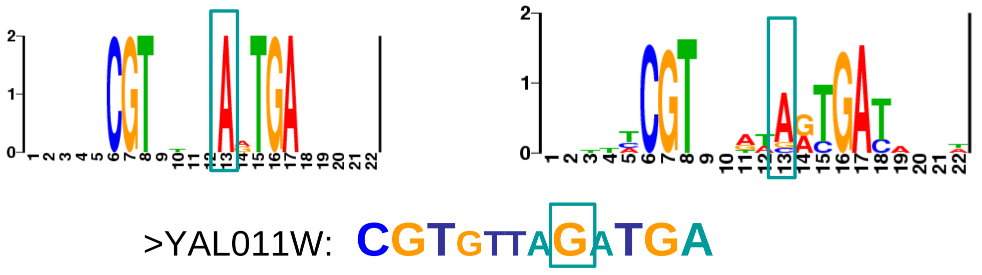


Figure 2.4: PSSM model vs. projected $k$-mer for motif representation.On the left side is a projected $k$-mer motif identified by the *SeedSearcher* for the ABF1 transcription factor motif. On the right side is the known PSSM motif model for this binding site, represented as a *Sequence logo*. At the bottom we see and example of a binding site that the projected $k$-mer motif completely misses because of the “G” in position 13. The PSSM motif in this case simply gives this binding site a slightly lower score.

To understand the possible benefits of using a PSSM model let us review the example in Figure **??**. On the right side we see a PSSM representation of the ABF1 transcription factor binding site motif, based on experimentally verified binding sites, as collected in the TRANSFAC database (**?**). Using ChIP experiments of **?** we defined a set of promoters suspected to be regulated by

[2]The entropy of a random variable $X$ with a distribution $P(X)$ is defined as $H(X) = -\sum_x P(x)\log(P(x))$. See **?** for more details

ABF1. Applying the *SeedSearcher* algorithm on this set, we were able to identify the crude motif presented on the left side of the figure. Intuitively the sequence logos of both motif representations are very similar. However, the later would not permit the real binding site found in the promoter of YAL011W presented in the bottom part of the figure. The reason for this is the "G" in position 13. On the other hand the PSSM model for this motif still permit this binding site, giving it a slightly lower score then having a "C" at this position. In general, compared to all the crude models for biding sites we described before, the PSSM model enables us to describe more accurately the affinity in each position of the binding site. As a result we also have a flexible tradeoff between sensitivity and specificity when we try and predict binding sites positions.

### 2.3.2 The Generative Model for Promoter Sequence

To learn PSSM motif models we are going to use a simple, generative, probabilistic framework first introduced by **?**. We briefly review this framework before moving to describe how this can be used to perform learning of PSSM motifs in a *discriminative* fashion.

The generative probabilistic framework we use makes a few basic assumptions. First, sequences are assumed to be generated either by a background model $BG$ or from the binding site model $M$. As we shell shortly see, we assume a Markov order 0 as the background model $BG$. However, one of the benefits of the discriminative approach is there is no need to explicitly model the background. the model for $M$ is a probabilistic PSSM as we just described. The model assumes that either all the sequence was generated from $BG$, in which case it is not regulated by $\mathcal{T}$, or that it has some (single) binding site sampled from $M$. Accordingly, our model has the following random variables:

- $S^{(l)} \ldots S^{(L)}$ - the letters in the sequence $S$.
- $R \in \{-1, 1\}$ - A random variable designating whether $\mathcal{T}$ regulates $S$ (in which case $R = 1$) or not.
- $H$ - the starting position of the PSSM (if indeed found in the sequence *i.e.*, if $R = 1$)

The parameters in the model are:

- $\theta_0$ - distribution of characters according to Markov order 0 background model $BG$.
- $\psi_j$ - distribution of characters in $j$'s position of the binding site

We denote by $\theta_0[c]$ or $\psi_j[c]$ the entry corresponding to the letter $c$.

According to this model we have:

$$P(S \mid R = -1) \quad = \quad \prod_{\ell} \theta_0[S^{(l)}] \tag{2.7}$$

And similarly:

$$P(S \mid R = 1) \quad = \quad \left( \prod_{\ell} \theta_0[S^{(l)}] \right) \sum_h P(H = h) \prod_{j=1}^{k} \frac{\psi_j[S^{(h+j-1)}]}{\theta_0[S^{(h+j-1)}]} \tag{2.8}$$

Where $P(H)$ is our prior belief about the binding site location. Assuming a uniform distribution, we set $P(H = h) = \frac{1}{L-k+1}, \forall h \in 1, \ldots, L - k + 1$. Note that in the above equation is the implicit assumption of a *single* binding site along the sequence. To relax this assumption and allow a *range* of binding sites number the above equation should be altered. It would have then have a prior term over the number of sites ($N$) in the promoter, and for each assignment $N = n$, we would sum over all possible assignments to the $n$ locations of the binding sites. This would make all the developed equations that follow much more complex and we therefore avoid this.

For now, we assume that the input for our motif finding problem contains:

- A set of promoter regions $S[1] \ldots S[M]$ for each gene $g_1, \ldots, g_M$.
- An observation $R[m] \in \{-1, 1\}$ indicating whether $\mathcal{T}$ regulate gene $g_m$ $\forall m \in \{1, \ldots, M\}$.

In many cases assuming $R[m]$ is observed is not realistic but we might have some prior knowledge about its possible assignment, based on expression or ChIP measurements. We will later show how these observation can be utilized in our model to relax the assumption that $R[m]$ is observed.

To complete this framework description we note that in the generative approach, first suggested by **?**, one tries to optimize the model's parameters to maximize the likelihood of the *observed* sequence given by:

$$LL(\mathcal{D}) = \sum_{m, R[m]=1} \log(P(S \mid R[m] = 1, \psi) + \sum_{m, R[m]=-1} P(S \mid R[m] = -1)) \tag{2.9}$$

We note that the probability of all the sequences in the right term for which $R[m] = -1$ is not dependent on the motif model parameter (Eq. (**??**)). As a result, these sequences have no effect on our optimization of the motif model parameters. This reflects the fact that in the generative approach we try and find motifs that are *common* to the regulated sequences, even if they are not necessarily *unique* to them. Instead, we will introduce now the discriminative approach to learn PSSM motifs, where both regulated and non regulated sequences have an effect on the motif model parameters.

**The Discriminative Approach for Motif Learning**

In the discriminative approach we try to optimize our model's parameters so as to best explain the observation that some sequence are regulated (*i.e.*, $P(R[m] = 1)$) while others are not ($P(R[m] =$

$-1)$). Formally, in our probabilistic framework this means we maximize the *conditional* log-loss:

$$\sum_{m=1}^{M} \log P(R = R[m] \mid S[m]) \tag{2.10}$$

The parameters we optimize are $\psi_j, j \in \{1, \ldots, k\}$. We can write the optimization as:

$$\max_{\psi} \sum_{m,R[m]=1} \log P(R = R[m] \mid S[m], \psi) + \sum_{m,R[m]=-1} \log(1 - P(R = R[m] \mid S[m], \psi)) \tag{2.11}$$

The above formulation makes the difference between the *generative* and *discriminative* approaches more evident. Instead of maximizing the overall likelihood of the sequence data as in Eq. (**??**), we try to optimize $\psi$ to maximize the conditional probability that $R = 1$ for the regulated sequences and $R = -1$ for those which are not. Note that in this formulation the parameters of the motif model $\psi$ also affect the right term for which $R[m] = -1$. The motif is now optimized not only to be common for the regulated sequences, but also unique to them. For example, if we now set $\psi$ to be a motif common to genes with $R[m] = 1$ as well genes with $R[m] = -1$ the right term of the above equation will decrease. We will now show how this discriminative target function can be optimized within our probabilistic framework.

### 2.3.3 Discriminative Training of PSSM models

To optimize $\psi$ according to the above formulation the *LearnPSSM* algorithm uses the technique of *gradient ascent* (see for example **?**). We will now develop the necessary equations for this procedure. This will also gain us some insights on the discriminative approach we suggest here.

First, we note it is easier to refer to the learning procedure as optimizing the parameters:

$$w_j[c] = \log \frac{\psi_j[c]}{\theta_0[c]}$$

This does not merely simplify notation. optimizing $w_j$ directly also carries the benefit there is no implicit representation of background model within our framework. There is no need to be given or to learn this separately. To ease notation we also write:

$$v = \log \frac{P(R = 1)}{P(R = -1)}$$

Which is simply the log odds ratio of regulation prior. Together, $\{v, w_j\}$ make the set of parameter we actually optimize in the learning process.

Now, Deriving from Eq. (**??**), Eq. (**??**) and using Bayes rule, we can write:

$$\begin{aligned} P(R=1 \mid S) &= \frac{P(S,R=1)}{P(S)} = \frac{P(S,R=1)}{P(S,R=1)+P(S,R=-1)} \\ &= \mathit{logit}(\log \frac{P(S,R=1)}{P(S,R=-1)}) = \mathit{logit}(\mathcal{R}) \end{aligned} \tag{2.12}$$

Where $\mathit{logit}(x) = \frac{1}{1+e^{-x}}$ is the *logistic* function and $\mathcal{R}$ is simply the *log odds* ratio for regulation of this specific sequence $S$. Similarly, we have:

$$P(R=-1 \mid S) = 1 - P(R=1 \mid S) = 1 - \mathit{logit}(\mathcal{R}) = \mathit{logit}(-\mathcal{R}) \tag{2.13}$$

$\mathcal{R}$ can be also written as

$$\mathcal{R} = \log\left[\frac{P(S,R=1)}{P(S,R=-1)}\right] = \log\left[\frac{P(R=1)}{P(R=-1)}\frac{1}{L-k+1}\sum_h\prod_{j=1}^{k}\frac{\psi_j[S^{(h+j-1)}]}{\theta_0[S^{(h+j-1)}]}\right] \tag{2.14}$$

Using the definition of $w_j$ we can define the last term of the above equation to be:

$$F(S) = \left(\sum_h \prod_{j=1}^{k} e^{w_j[S^{(h+j-1)}]}\right)$$

Now we combine the last few equations with the definitions of $v$ and have:

$$P(R \mid S) \quad = \quad \mathit{logit}\left(R\left(v + \log F(S)\right) - \log(C)\right) \tag{2.15}$$

Where we used $C = L - k + 1$. Note the above formulation is correct for both $R = 1$ and $R = -1$.

Summing up to this point, the function we optimize in Eq. (**??**) can be re-written as:

$$\begin{aligned} \mathcal{W} &= \sum_{m,R[m]=1} \log \mathit{logit}(\mathcal{R}[m]) + \sum_{m,R[m]=-1} \log \mathit{logit}(-\mathcal{R}[m]) & (2.16) \\ &= \sum_{m=1}^{M} \log \mathit{logit}\left(R[m]\left(v + \log F(S[m])\right) - C\right) & (2.17) \end{aligned}$$

The formulation of $\mathcal{W}(\mathcal{R})$ also gives us insight on the direction in which we optimize our target function $\mathcal{W}$. Note that the derivative by $\mathcal{R}$ gives us:

$$\frac{\partial \log \mathit{logit}(\mathcal{R})}{\partial \mathcal{R}} = \mathit{logit}(-\mathcal{R}) \tag{2.18}$$

This means sequences with small log-loss ($\mathit{logit}(\mathcal{R}) \sim 1$) get a dumping effect on their contribution to the derivative (because $\mathit{logit}(\mathcal{R}) \sim 1$ implies $\mathit{logit}(-\mathcal{R}) \sim 0$) .

To maximize $\mathcal{W}$ using gradient ascent, we need to compute the gradient $\nabla\mathcal{W}$ with respect to the parameters we defined, $w_j$ and $v$. We start with $v$:

$$\begin{aligned} \frac{\partial \mathcal{W}}{\partial v} &= \sum_m \frac{\partial \log \mathit{logit}\left(R[m]\left(v + \log F(S[m])\right) - C\right)}{\partial v} \\ &= \sum_m R[m] \mathit{logit}\left(-R[m]\left(v + \log F(S[m])\right) - C\right) && (2.19) \\ &= \sum_m R[m] P(-R[m] \mid S[m]) && (2.20) \end{aligned}$$

Here we used again Eq. (**??**). The last formulation shows we want to change $v$ in the direction that will improve the classification of the example (this is determined by the term $R$ in the derivative), in proportion to the mass we assign to the wrong label (this is determined by the term $P(-R \mid S)$).

Next, we compute $\mathcal{W}$ derivative with respect to the $w_j$. First, we have:

$$\frac{\partial \log F(S)}{\partial w_j[c]} = \frac{1}{F(S)} \frac{\partial F(S)}{\partial w_j[c]} = \frac{1}{F(S)} \sum_{h:S_{h+j}=c} \prod_{j'=1}^{k} e^{w'_j[S_{i+j'}]}$$

We can now use this to get:

$$\begin{aligned} \frac{\partial \mathcal{W}}{\partial w_j[c]} &= \sum_m \frac{\partial \log P(R[m] \mid S)}{\partial w_j[c]} = \\ &= \sum_m R[m] \cdot P(-R[m] \mid S) \cdot \frac{1}{F(S)} \cdot \sum_{h:S_{h+j-1}=c} \prod_{j'=1}^{k} e^{w_{j'}[S_{i+j'-1}]} && (2.21) \end{aligned}$$

Note that we have free parameters here. Thus, it is reasonable to require that $\sum_c w_j[c] = 1, \forall j$. In practice, this requires us to either use more condense parameterization or use constrained optimization.

To summarize this section, the *LearnPSSM* algorithm for discriminative learning of PSSM motifs is as follows:

1. Given sequence sets $G, \mathcal{G}$ we use the *SeedSearcher* algorithm described in Section **??** to find the best discriminative crude motifs (usually, projected $k$-mers described in Section **??**.
2. For the crude motifs found to be statistically significant according to *random selection* null hypothesis and its matching the *hyper geometric* distribution (Section **??**), perform PSSM motif optimization by starting from the crude motif as an initial PSSM and optimize the target function of Eq. (**??**). For this perform conjugate gradient ascent by computing the gradient of

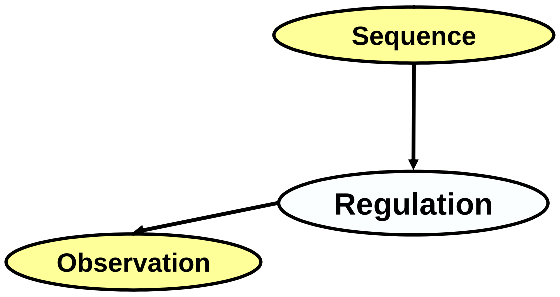


Figure 2.5: Graphical representation of the relations between the variables in the *LearnPSSM* framework. The observed entities are colored yellow.

the target function given by Eq. (**??**), Eq. (**??**).

### 2.3.4 Incorporating Prior Biological Knowledge

When developing the equations for training PSSM motifs we assumed we *know* $R[m]$ for all the sequences. In many settings this is not the case. For example, we might cluster genes bases on their expression profile and then search for a common regulator for genes in the same cluster. In this case $R[m]$ is not really observed but rather some other entity (the expression level of the gene). A simple way to incorporate information such as co-expression into our framework is by setting $\mathcal{R}[m] = 1$ for genes in the cluster of interest and $\mathcal{R}[m] = -1$ for all the others. The problem with the simplistic approach is that often we are not that confident in our training data. The cluster of co-expressed genes might contain false positive (*i.e.*, non-regulated genes that appear in the cluster), and similarly there might be false negative genes (that are regulated but were not included in the cluster). To capture such considerations, we exploit the probabilistic framework we introduced in Section **??** and set the observations we have (in this case - the cluster tag based on expression profile) to be a *noisy sensor* of the (hidden) regulation event in our model. This allows the learning algorithm to view the promoter sequences of genes in the cluster as having high probability of being regulated ($P(R[m] = 1) \sim 1$) , and all other sequences as having lower probability of being regulated ($P(R[m] = 1) \sim 0$).

Formally, we introduce a random variable $O$ that denotes our *observation* about the gene and is dependent of the regulated status of the gene, but not on its promoter sequence. This is illustrated in Figure **??**. Under this model definition, instead of maximizing the conditional probability of the regulation events (which are no longer assumed to be observed), we now maximize the conditional probability of the noisy sensor observations, given the promoter sequence of the genes. The target function of Eq. (**??**) is now replaced with:

$$\sum_m \log P(O[m] \mid S[m]) = \sum_m \sum_{r\in\{1,-1\}} P(O[m] \mid R[m] = r) P(R[m] = r \mid S[m]) \qquad (2.22)$$

Thus, we view $O$ as a *noisy sensor* of the underlying biological regulation. A crucial detail lies in the choice of $P(O \mid R)$. If the observation is that of co-expressed genes or genes with similar functional annotation, we can set this distribution to reflect the fact that most regulated genes will appear in the co-regulated cluster. Similarly, we want the distribution to reflect that few non-regulated genes will appear in the cluster.

A more interesting case involves ChIP location data (**???**). This data provide strong evidence about regulation relationships, when available. As reviewed in Section **??**, these experiments measure the ratio of "hits" for each DNA fragment between a control set of DNA extracted from cells, and DNA that was filtered for binding to the transcription factor of interest. This assay is noisy, and thus we cannot simply "read off" binding. Instead, the experimental protocol by **?** uses a statistical model to assign a *p-value* to various ratios. A ratio with a small p-value suggests significant evidence for binding by the transcription factor. Larger p-values can indicate a weaker binding or experimental noise.

To incorporate this type of data as *noisy sensor* for regulation events as given by Figure **??**, we note that in our probabilistic framework there are two cases. If $R = 1$, we expect $O$ to be determined by the noise in the assay. By design, the statistical procedure used in ChIP experiments is such that $P(O = \rho \mid R = -1)$ where $\rho \in [0, 1]$ has a uniform distribution (this is exactly the definition of the $p$-value reported as the ChIP result). In the second case, when $R = 1$, we expect $O$ to be small. We choose to model this using a density $p(O = \rho \mid R == 1) = c \exp(-w\rho)$, the exponential distribution with weight $w$, where $c = w/(1 - \exp(-w))$ is a normalization constant ensuring that the density function integrates to 1. Based on examination of p-values in the experiments of **?**, we choose $w = 20$ in our experiments. Now, once we observe $O$, the probability of this observation propagates to $R$. If $O$ is very small, then it is more likely that it was generated from $R = 1$. If it is larger, then it is more probable that it was generated from $R = -1$. This model allows us to use the location-specific binding data as guidance for inferring regulation relationships, without making overly strong assumptions about their accuracy.

Finally, we note that whatever definition we use for $P(O \mid R)$, the affect of having the structure of Figure **??** on the actual computation procedure done by the *LearnPSSM* algorithm is trivial. We can see from Eq. (**??**) that now instead of having "hard" definitions of gene groups for which $R[m] = 1$ or $R[m] = -1$ as in Eq. (**??**), each gene is now *weighted* in the terms for $P(R[m] = 1)$ and $P(R[m] = -1)$ by $P(O[m] \mid R[m] = r)$.

### 2.3.5 Reweighting of Samples

We described how our discriminative approach takes into account genes for which $R[m] = 1$ as well as genes for which $R[m] = -1$ when optimizing the motif model. We have shown how this is done both in the simpler case where we assume $R[m]$ is observed, and for the more general case were we have other observations, as ChIP assays or expression measurements, that serve as noisy sensors for the hidden value of $R[m]$.

In practice, our ability to learn predictive motifs that characterize the regulated genes might be compromised when the set of genes that do *not* appear to be regulated is much larger then those suspected of co-regulation. This is a well known problem in discriminative machine learning algorithms for classification tasks. When the positively labeled group (*i.e.*, the genes for which $R[m] = 1$) is much smaller then the negatively labeled samples (genes for which $R[m] = 1$) these algorithms may tend to optimize their parameters so as to best characterize the negative group, thus avoiding classification error on the much larger group. In our case, this means the optimized motif might not characterize well sequence patterns found in the positive group. We can easily see why this can may happen if we revisit Eq. (**??**), where we assumed $R[m]$ is observed:

$$\sum_{m,R[m]=1} \log P(R = R[m] \mid S[m], \psi) + \sum_{m,R[m]=-1} \log(1 - P(R = R[m] \mid S[m], \psi))$$

Since this is our target function we might converge to motif parameters which greatly increase the probability of *not* being regulated since most of the sequences belong to the second term. Note that moving to the setting of Eq. (**??**) with the noisy sensor model does not change the picture much in this case. Since each sample (gene) is now simply weighted by $P(O[m] \mid R[m] = r)$ we still face the same problem if the vast majority of the genes have an observation which does not support a regulation event for them.

There are two main ways to avoid this problem in our case. The first is to make sure the starting point in the motif parameters search space is such that it characterize well the group of genes suspected to be co-regulated. This is done by the *SeedSearcher* as we already described. The second way, utilized in general machine learning tasks, is to artificially increase the weight of the positive group *i.e.*, the genes for which $R[m] = 1$ is more probable. If $R[m]$ is observed we can make sure the ratio between the regulated and non regulated genes is fixed to some constant:

$$W \frac{\sum_{m:R[m]=1} I}{\sum_{m:R[m]=-1} I} = C$$

Where $I$ is the indicator function. This means each regulated sequence is now weighted as if it was observed $W$ times in our data set.

In the case where $R[m]$ is hidden and we have the noisy sensor model we do the following. we use Bayes law to compute the posterior probability of the gene being regulated, given the noisy sensor observation:

$$P(R = 1 \mid O) = P(R = 1)\frac{P(O \mid R = 1)}{P(O)}$$

Where $P(R = 1)$ is a prior over regulation events and $P(O) = \sum_{r \in \{1,-1\}} P(R = r)P(O \mid R = r)$ is the marginal probability for the observation. We then compute the relative weight of positive and negative samples we have:

$$W_n = \sum_m (1 - P(R[m] = 1 \mid O[m])), W_p = \sum_m P(R[m] = 1 \mid O[m])$$

We then adjust the weights so that $\frac{W_p}{W_n} = C$. Note this definition avoids the need to use a hard threshold to define positive and negative samples. The weight of each gene is now adjusted according to how much we believe it is regulated by our observations. As for the ratio constant $C$ we tested it to be in the range $[0.3, 07]$ with similar results.

## 2.4 *TestPSSM*: Evaluating Statistical Significance of PSSM Scores

The careful reader might notice that at this point we are still missing an important part in order to handle the motif finding problem. Recall that our original aim was to computationally predict putative binding sites locations. Up to now we managed to learn a discriminative PSSM motif. However, according to this motif model, *any* subsequence of length $k$ (the PSSM length) is plausible. $K$-Mers do not get a $\{0, 1\}$ score for matching or not matching the motif. As a consequence, the *random selection* null hypothesis approach we took in Section **??** cannot be applied here directly and we need to define some other way to measure the statistical significance of observing a given $k$-mer in a promoter sequence.

Given the probabilistic model we defined in Section **??** it is naturally appealing to use the approach we termed in Section **??** as the *random generation* null hypothesis. According to our model a $k$-mer can be either generated from a background model $BG$ or from the motif model $M$. The score $\mathcal{S}(S^{(i,\ldots,i+k-1)})$ of a $k$-mer in position $i$ of sequence $S$ is:

$$\mathcal{S}(S^{(i,i+k-1,\ldots,)}) = \sum_{j=1}^{k} w_j[S^{(i+j-1)}] \tag{2.23}$$

Which is simply the *log odds score* for this $k$-mer under the motif PSSM model and the background model $BG$. The *random generation* null hypothesis would therefore be generation of the $k$-mer by $BG$. The statistical significance of a score $\mathcal{S}$ should be measured by the $p$-value of $\mathcal{S}$ *i.e.*, the probability of achieving this score or higher according to the background distribution over $k$-mers.

$$\boldsymbol{E}_{P_{BG}(X)}[1\{Score(X) \geq \mathcal{S}\}] \tag{2.24}$$

where $1\{\}$ is the indicator function, $P_{BG}(X)$ is the background distribution over a random variable $X$ that ranges over possible DNA motifs, and $Score()$ is the scoring function as defined above. In our case $X$ is a subsequence of a fixed length $k$.

During recent years several approaches have been suggested to compute the $p$-value of a score $S$ under the above definition, either precisely or approximately. The most naive approach is probably to compute the p.d.f over $\mathcal{S}$ by enumerating over all possible $4^k$ assignments to $X$. **?** have suggested a more efficient way to compute the p.d.f of $\mathcal{S}$ using a branch and bound algorithm. However, their solution still involves the exact computation of the entire p.d.f which is not practical for $k > 10$. **?** use large deviation approximations to compute the $p$-value of PSSM scores. This seems to work both accurately and efficiently, at least for simple Markov order 0 background models. Recently, **?** offered an efficient analytical algorithms for the exact $p$-value computation. They use a matrix multiplication approach which is similar to dynamic programming. The formulation of Eq. (**??**) suggests another possible approach to approximate the $p$-values of scores, based on sampling. However, simply sampling $k$-mers from the background model becomes unpractical for $p$ values of the order of $10{-}5$ because you need about $10^7$ to have an accurate enough estimate of $p$-values this low.

The various shortcomings of methods available at the time motivated us to develop the TestPSSM algorithm we present in this section. The idea behind the algorithm is to simultaneously compute the *exact* $p$-value of a set of scores $\{S_1, \ldots, S_N\}$ $(N \sim 10)$ , using the branch and bound approach similar to **?**. Because we are only interested in the very low $p$-value range that match putative binding sites, our algorithm is able to avoids computing the entire p.d.f over $\mathcal{S}$ . Moreover, since our motivation comes from assessing only promising putative binding sites, the set of scores we define lies only in the very high region of the possible scores, where the branch and bound technique is most efficient. To compute the $p$- value for any score $S_i < \mathcal{S} < S_j, i, j \in \{1, \ldots, N\}$ we use linear extrapolation from $S_i, S_j$ to estimate the $p$-value of $\mathcal{S}$ . As we will show, this enables us to save much computation time yet get very accurate results.

## 2.4.1 Algorithm

The input for the algorithm is the PSSM motif $M$ given as $w_j[c]$ for $j \in \{1, \ldots, k\}, c \in \{A, C, G, T\}$ and the background probability model $BG$. We make no specific assumptions about the background model, though in practice we use *Bayesian network* representation for it. The only formal requirement from $BG$ is that the computation of $P_{BG}(X)$ can be done efficiently for any partial or full assignment.

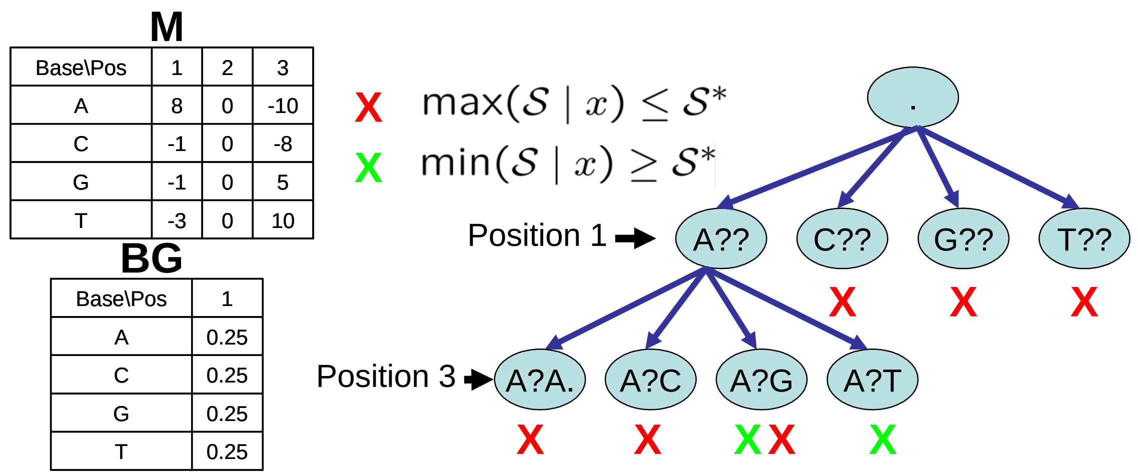
We start by computing:

$$\mathcal{S}_{min} = \sum_j \min(w_j[c]), \mathcal{S}_{max} = \sum_j \max(w_j[c])$$

$[\mathcal{S}_{min}, \mathcal{S}_{max}]$ gives the range of possible scores. Given this range we define the set of scores for which we will compute the exact $p$-value to be:

$$\begin{aligned} \mathcal{S}_1 &= \mathcal{S}_{max} - R(\mathcal{S}_{max} - \mathcal{S}_{min}) \\ \mathcal{S}_M &= \mathcal{S}_{max} \\ \mathcal{S}_i &= \frac{\alpha^{i-1}}{\alpha^M}(\mathcal{S}_M - \mathcal{S}_1) + \mathcal{S}_1, \forall i \in \{2, \ldots, M-1\} \end{aligned} \tag{2.25}$$

Where $M$ is the number of scores $R \in [0,1]$ is the ratio from the score range the user is interested to cover and $\alpha > 1$ is some constant. Both have default values (10 and $0.3$, respectively) and can be set by the user. Another parameter the user sets is the maximal $p$-value to consider, denoted $p_{max}$. Note that we do not know in advance the exact correspondence between scores and $p$-values. Therefore by setting $R$ and $p_{max}$, the user can save a lot of the programs running time directing the program on the range in which to perform the computation of exact $p$-values.



**M**

| Base\Pos | 1 | 2 | 3 |
|---|---|---|---|
| A | 8 | 0 | -10 |
| C | -1 | 0 | -8 |
| G | -1 | 0 | 5 |
| T | -3 | 0 | 10 |

**BG**

| Base\Pos | 1 |
|---|---|
| A | 0.25 |
| C | 0.25 |
| G | 0.25 |
| T | 0.25 |

X $\max(\mathcal{S} \mid x) \leq \mathcal{S}^*$
X $\min(\mathcal{S} \mid x) \geq \mathcal{S}^*$
.
Position 1
A??
C??
G??
T??
X
X
X
Position 3
A?A.
A?C
A?G
A?T
X
X
XX
X

Figure 2.6: Illustration of the branch & bound technique used by *TestPSSM*. For implicitly the background model $BG$ (bottom left) is set to the uniform distribution. The motif $M$ model is the matrix on the top left corner. Note $M$ is not a probabilistic PSSM model, and no constraints are posed on its attributes. The search illustrated is for the score $S^* = S(AAG)$.

The branch and bound technique used by *TestPSSM* is based described using an assignment tree $\mathcal{T}$. $\mathcal{T}$ is defined as a tree where each level in $\mathcal{T}$ corresponds to a position in $X$ and each node corresponds to the partial assignment of some letters from the alphabet to positions matching the path to this node. We note the tree depth is exactly $k$ and any full assignment $x$ corresponds to some leaf in $\mathcal{T}$. However, the *order* of the assignments in $\mathcal{T}$ does not necessarily match the order of the positions in $X$. We denote this order $V(j)$ so that $\{V(j)\}_{j=1}^{k}$ is some permutation over $\{1,\ldots,k\}$. An illustration of $\mathcal{T}$ for some assignments can be seen in Figure **??**. We see that in this case the first level corresponds to the first position and the second matches the third position of possible assignments.

An important feature of $\mathcal{T}$ is that we can easily compute the maximal and minimal possible scores given that partial assignment at any node of it $x$:

$$\max(\mathcal{S} \mid x) = \mathcal{S}(x^*), x_i^* = x_i \forall i \in Pos(x), x_i^* = Argmax(w_i[c]) \forall i \notin Pos(x)$$

Where $Pos(x)$ are the indexes of the positions already assigned in $x$ ($\min(\mathcal{S} \mid x)$ is defined the same way with the $Argmin$ operator). Now, when traversing the tree *in order*, we can apply the branch and bound approach in the following way. For a given a score $S^*$, we initiate $P_v(S^*) = 0, \overline{P}_v(S^*) = 0$. We use $P_v$ to compute the score's $p$-value and $\overline{P}_v$ to for the cumulative distribution function at $S^*$. As we traverse $\mathcal{T}$ in order and get to some node corresponding to the assignment $x$, we compute $\max(\mathcal{S} \mid x), \min(\mathcal{S} \mid x)$. If $\max(\mathcal{S} \mid x) \leq S^*$ then $\overline{P}_v(S^*) + = P_{BG}(x)$. In practice this means we do not need to traverse the tree any further down the node of $x$, since all possible assignments are going to have a score less or equal to $S^*$. This is illustrated by red "X" signs in Figure **??**. If, on the other hand, $\min(\mathcal{S} \mid x) \geq S^*$ then $P_v(S^*) + = P_{BG}(x)$ since all possible assignments downward from $x$'s node have a score at least as good as $S^*$. In this case too we need not traverse the tree any further down this point. This is illustrated by green "X" signs in Figure **??**. If none of the above two conditions are met, we keep traversing $\mathcal{T}$ down from the node of $x$. Of course, if during our path through $\mathcal{T}$ we find that $\overline{P}_v(S^*) > p_{max}$, then we can abort and report $S$ to be not significance enough. If we reach a leaf matching a full assignment $x$ we just compute $P_{BG}(x)$ and add this to either $P_v$ or $\overline{P}_v$ if $\mathcal{S}(x) \geq S^*$ or $\mathcal{S}(x) \leq S^*$ respectively. It follows that upon finishing traversing $\mathcal{T}$ in this way we have the $p$-value of $S$ stored in $P_v$. To make the algorithm more efficient, we compute this for the whole set of scores $S_1,\ldots,S_M$ simultaneously. We store $P_v(S_i), \overline{P}_v(S_i), \forall i$, and traverse $\mathcal{T}$ down a node corresponding to a partial assignment $x$ only if there is *some* $S_i$ for which this is relevant according the criteria defined above. As a result we visit any node $x$ during the single traverse of $\mathcal{T}$ only if we need it for some $P_v(S_i)$, and only once.

Another important issue to make the branch and bound work efficient is the order over the motif's positions ($V(j)$). The efficiency of the algorithm can be measured by the number of nodes in $\mathcal{T}$ the algorithm actually has to visit. Using the same idea as in the information theory view of code generation (*e.g.*, **?**), we try to minimize this number by defining $V(j)$ by the information

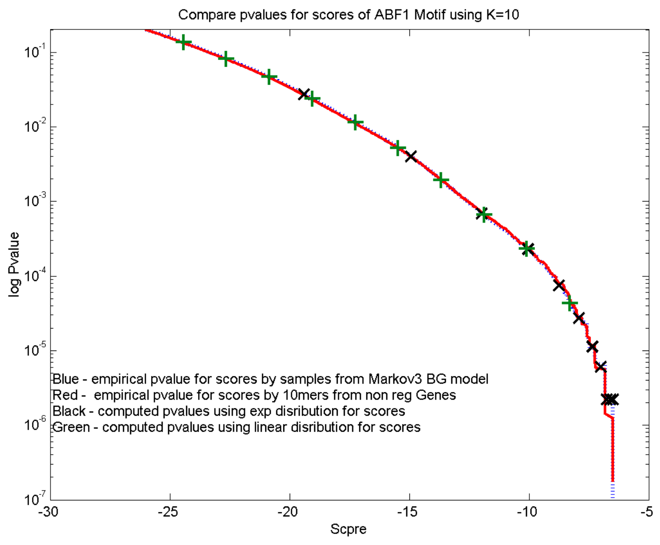


Figure 2.7: Accuracy evaluation of *TestPSSM* using the ABF1 motif in yeast.

content in each position of the motif. Another similar criteria is the variability in each position. By first assigning letters to more conserved positions the score can be pushed more quickly to the high or low ends and this way satisfy the conditions of $\max(\mathcal{S} \mid x) \leq S^*$ and $\min(\mathcal{S} \mid x) \geq S^*$ more quickly, saving the need to traversing $\mathcal{T}$ further down. In the example of Figure **??**, we start with position 1 and next assign position 3. The branch and bound technique in this case results in visiting only 8 nodes out of 84 nodes of the full assignment tree $\mathcal{T}$.

### 2.4.2 Evaluation

There are several aspects of *TestPSSM* performance we should consider when evaluating it. Remember that the algorithm's ultimate goal is to assess the possibility that a given subsequence $S$ of length $k$ is a binding site of a regulator who's biding site The model we use assumes a generative background model ($BG$) so according to it we are interested in the question how rare is it for $BG$ to generate a $k$-mer with a score as high as $\mathcal{S}(S)$. *TestPSSM* uses several assumptions in order to estimates the statistical significance of this event. First, it implicitly assumes that $S$ can be generated either from the biding site model PSSM or from the background model $P_{BG}(X)$. Then it assumes the long promoter sequences are actually i.i.d samples from this background model (when not from PSSM). We would like to test the affect of these on assumption on the algorithm's performance. The algorithm also uses just a relatively small (typically 12) set of points to estimate the PSSM scores p.d.f by $BG$. This again might have an effect on the algorithm's accuracy that should be tested. Finally, there is the question of efficiency. As we described in the previous section, this can be measured by the amount of nodes in the assigning tree $\mathcal{T}$ the algorithm actually visits.

| **Assignment Length** | **# Nodes used** |
|---|---|
| 6 | 9 |
| 7 | 69 |
| 8 | 291 |
| 9 | 868 |
| 10 | 4735 |
| **total** | **5972** |

Figure 2.8: Efficiency evaluation of *TestPSSM* using the ABF1 motif in yeast. Listed is the number of nodes in the assignment tree $\mathcal{T}$ actually used by the algorithm to compute $p$-values for a given set of scores. The assignment length of the nodes is equivalent to the depth in the tree in which the nodes are located. A full, brute force, traverse of $\mathcal{T}$ would have involved computing probabilities for $4^{10} = 1048576$ full assignment nodes at depth 10.

Figure **??** shows that albeit all the simplifying assumptions we just described *TestPSSM* achieves a high level of accuracy. Here we sampled $k$-mers from the commonly used *Markov*-3 model for yeast promoter sequence and using the known PSSM motif for the transcription factor ABF1. Based on $10^7$ such samples we computed the scores p.d.f (blue line). we see that *TestPSSM* is able to follow accurately the scores' p.d.f (black and green lines). Moreover, we see that the default use of 12 scores exponentially distributed on the upper range of possible score (black line) is sufficient to give a close approximation of the scores $p$-value. We also see the advantage of the exponential distribution of the points (black line) over a linear distribution of them (green line). It is evident the exponential distribution of scores gives a much more accurate cover of the p.d.f curve in the high (and more relevant) range of scores. Finally, we compare the estimated p.d.f under the model's assumptions to the the green empirical curve. The later was obtained by sampling $k$-mers from promoter sequences which were labeled as non ABF1 targets by **?** ChIP results. As we see, the red graph follow closely this curve too. We got similar results on other cases tested. However, we note that the empirical curve we compared to is also based on independent samples, which is a simplifying assumption for long promoter sequences. Maybe more importantly for binding site prediction, it has been demonstrated by others that using more flexible background models can improve prediction accuracy (**?**). Another aspect of *TestPSSM* is the efficiency of the Branch and Bound approach. Table **??** illustrates how many nodes from the possible full depth tree of $4^k$ possible assignments the algorithm really needs to visit in order to get the results obtained in Figure **??**. In this case the full tree has $4^{10} = 1048576$ possible full assignment nodes. Summing over all nodes used to compute the set of scores $p$-values, we see the algorithm used only 5972 nodes. The comparison is even more impressive if we consider the total number of nodes visited (including inner nodes) compared to a full tree of depth 10. In practice, we were able to use the *TestPSSM* algorithm for PSSMs of up to 16-18 base pairs long.

## 2.5 Discussion

In this chapter we have presented a novel *discriminative* approach for handling the *ab initio* motif finding problem. Our approach includes efficient search over an enumerable search space of crude motif representations using the *SeedSearcher* algorithm, testing the statistical significance of the found motifs using the *hyper geometric* $p$-value based score, and then refining the best scoring motifs into PSSM representations using a discriminative probabilistic framework via the *LearnPSSM* algorithm. In order to evaluate the statistical significance of putative binding sites and enable us to control our false discovery rate in genomic wide scans we presented the *TestPSSM* algorithm that uses a branch and bound approach to exactly compute the $p$-values of a set of high scores and then use them to estimate the $p$-values of scores within this range. We showed the efficiency of the *TestPSSM* branch and bound approach and its ability to approximate high scores $p$-values accurately.

Our work was certainly not the first one to handle the *ab initio* motif finding problem. The usage of PSSM for regulatory motif finding [3] can be traced back to **?** and was used even before that for motif representation in proteins. However, most methods that were developed during the years and are commonly used (*e.g.*, MEME by **?** and AlignAce by **?**) suffered from a common problem. Since they were trying to optimize motifs that are only *common* to a given set of sequences, in many cases they converged to motifs that were not very specific to the given set of sequences such as poly A/T motifs commonly found in yeast promoters. Their solution was either to remove commonly appearing low complexity regions as a preprocessing step or analyzing the specificity of the found motif in a post processing step. Both solutions are problematic. Sometimes a low complexity region may be important for motif definition while in other cases the definition of the low complexity region might still leave to many common yet not unique motifs to converge to. Removing non unique motifs as a post processing step is also problematic since the algorithm itself is still aimed to find only common motifs and not unique ones and their is also the added computational cost of optimizing many motifs just to discard them as the following stage. Our algorithm avoids these pitfalls by efficiently searching for *discriminative* motifs which are common yet also relatively unique to the given set of sequences, compared to a reference poll of "negative" sequences assumed not to share the same regulation mechanism.

Albeit the natural appeal of the discriminative approach and the relative improvements in motif finding performance (Section **??**), the algorithms we present here have several limitations worth noting. First, we note that by the definition of the statistical significance test we use, we may fail to detect significant motifs in a given set of promoters for various reasons. If the definition of the positive (*i.e.*, presumable co regulated) and negative sets of sequences has to many errors, the

[3] Another name used for the 4xK matrices that represent sequence motifs is PWM (Position Weight Matrices)

correlation between the motif presence and the groups definition as measured by the *hyper geometric* $p$-value may be to weak. This is of course a problem common to all motif finding algorithms (see **?** for an elaboration on this). A related problem is when our crude representation of the motif is such that it doesn't capture the sequence motif closely enough. For example, if a sequence motif is such that there are two conserved patterns but with a short *variable* distance between them our algorithm may fail to detect it because it always searches for motifs with a fixed overall length. In the more theoretical aspects, our approach systematically searches the space of crude motifs in order to later refine PSSM motif representations. Therefore, the only formal guarantees given by our algorithms is for crude motifs found with the *SeedSearcher*. Recently, **?** have showed how an enumerable search space of similar crude motifs can be related to the continues space of all possible PSSM motifs, under some conditions. This in turn enabled him to suggest some formal guarantees for the common PSSM motifs he finds. How this can be used with discriminative scoring in our setting remains an open question.

When it comes to inferring the regulatory program that controls a set of genes, maybe the most limiting assumption of all the motif finding algorithms we reviewed here, including ours, is that of a *single* motif. In many cases the regulatory program may contain more then a single motif and only the combinatorial combination of them, forming *AND,OR,NOT* logical relations between them is what distinguish a specific regulatory program. Sometimes multiple copies of the similar motifs are needed (*e.g.*, the RAP1 motif found in promoters of ribosomal proteins (RPGs) (**?**)). Of course when we identify the statistical significant motifs we do not limit ourselves to just one motif. Therefore a partial remedy to the problem is by reporting more, maybe less significant, motifs and then testing for the statistical enrichment of their combinations, repetitions etc. However, this solution does not answer the question of how to simultaneously learn several motifs together, taking into account the presence of each other while optimizing each one. An important problem related to this is how to decide how many motifs do we actually have in our set of sequences. As we described, our *SeedSearcher* may find motifs which are not only crude but also *partial* representation of the actual motifs. In some cases, by expending the original motif reported by the *SeedSearcher* we might be able to identify the actual motif is longer. In other cases, using a more complex motif representation (as the ones described in the following chapter) might help us identify there are actually similar *subtypes* of the same motif present. But the question how the best combine all the motifs reported by the *SeedSearcher* and to learn a set of motifs simultaneously remains an open challenge.

Obviously, there is a limit to what we can infer from sequence data alone (*e.g.*, **?**). A promising way to overcome noise in motif identification is by incorporating other sources of data. In a sense, this is what this dissertation is all about - we try to combine more sources of data such as expression profiles and ChIP assays to help us infer what are the correct motif definitions and how are they combined to create the regulatory programs at the transcription stage. At the simpler level, we showed how prior biological data can be incorporated into our algorithms as prior over regulation events or prior over the motif location along the promoter. This prior can come from ChIP assays that measure transcription factors binding to promoters or from phylogenetic comparison of closely related species that indicate what areas in the promoter are more conserved and therefore more likely to be functional (**?**). Another promising source of data is based on nucleosome positions and their possible effect on regulation (**?**). However, the framework we described so far involves incorporating these sources of data just as a fixed, predefined prior. As we discussed above, it is also limited to identifying single motifs and not their synergistic effect for regulation. We will see how we can improve on this in order to infer transcriptional regulatory programs in later chapters. But, before we go into more complex regulatory modules models we will first show how we can significantly improve our binding site predictions by questioning the basic modeling assumption of the commonly used PSSM motif model. We will show how alternative motif representation, based on the Bayesian network language, helps us achieve these much higher accuracy levels in binding site predictions. This will be the subject of our next chapter, followed by a systematic evaluation chapter for all motif finding algorithms we presented.

# Chapter 3

# Motif Models with Interdependent Positions

Previous chapter introduced the computational task of *ab initio* motif finding. The input in this case is a set of sequences suspected to be co regulated, though we don't know where the exact binding sites are. The aim is to predict where are the biding sites in the given sequences or in new ones. Because the binding sites locations along the sequences are not known in advance this is also termed the motif finding problem with *unaligned* sequences. A related but somewhat easier task is again to predict binding sites, but this time when we have as input biologically verified binding site examples. Accordingly, we term this the motif finding problem with *aligned* sequences. In both tasks, we still need to produce some representation of the binding site motif. With this motif at hand and algorithms such as *TestPSSM* (Section **??**), we are then able to perform genomic wide scans for putative binding sites locations.

In the previous chapter the binding site model we learned was a PSSM. This is currently the most commonly used model for binding site motifs. As we reviewed in Section **??**, PSSM records the preference for each potential DNA nucleotide at each binding site position. It inherently assumes that positions within the motif are independent of each other. This is reflected by the score a PSSM assigns a putative binding site (Eq. (**??**)). However, it is an open question whether this strong independence assumption is reasonable. Several results indicate that in some cases, there might be dependency between the binding site positions (*e.g.*, **???**). A related yet different question is the practical implications of this assumption: Does this simplifying independence assumption has any affect on our ability to accurately predict binding sites? In this chapter we will develop a set of biding sites models which relax the independence assumption made by the PSSM model. As we show, these carry the benefit of greatly increasing binding sites prediction accuracy.

The main technical question one faces is how to relax the assumption of independence. For this purpose we need to model the joint distribution of all positions. A naive representation requires a large number of parameters (exponential in the motif length). The full independence model (a PSSM) and the full dependence model (an exponentially large joint distribution) represent the two ends of a spectrum. The choice of representation leads to a tradeoff: Less expressive models cannot represent complex dependencies, but have succinct representation and can be learned robustly from few examples. More expressive models can represent complex dependencies, but involve many parameters and require a larger number of examples for learning.

In this chapter we describe the usage of flexible probabilistic binding site models which cover the middle ground of this spectrum. Using *Bayesian Networks* for model representation, we develop algorithms to learn these models from either aligned or unaligned sequences. The usage of well established likelihood based scores with complexity penalty enables our models to fall back to the simple PSSM model when no sufficient evidence of dependencies is observed.

We start by describing the set of motif models in Section **??**. We then develop the machinery needed to learn these models in Section **??**. Just as with the *TestPSSM* algorithm in Section **??**, we need an efficient procedure that measure the statistical significance of an observed putative site. For this we develop the *CIS* algorithm in Section **??**. Finally, the following chapter will summarize our evaluation of motif finding algorithms developed here and in the previous chapter.

## 3.1 Modeling Position Dependencies

We now consider how to model a sequence motif representing the binding sites of a transcription factor. We want to represent the commonalities among different binding sites. One way of going about this is to represent a probability distribution over sequences that will assign high probability to sequences that are likely to be found at the binding site. We assume that binding sites are of length $K$. We want to represent a probability distribution over all $4^K$ possible $K$-mers that can appear at the binding site. Formally, we need a distribution $P(X_1, \ldots, X_K)$ where $X_i$ is a *random variable* that represents the nucleotides in the $i$'th position of the $K$-mer. A naive representation of such a distribution simply lists the probabilities of all $4^K$ assignments. This representation is clearly wasteful and is also unrealistic to estimate from data. Thus, we are interested in more succinct ways of representing such distributions.

The most commonly used binding site model is the PSSM model reviewed in Section **??**. This model assumes there is an independent distribution over $\{A, C, G, T\}$ at each position of the motif. This is reflected in the probability the model assigns to possible binding sites:

$$P(X_1 \ldots X_K) = \prod_{i=1}^{K} P(X_i)$$

Where $P(X_i)$ is the *marginal probability* of each nucleotide $X_i$ in the distribution and is a shorthand for probability events of the form $P(X_i = A)$

PSSM requires $3K$ parameters to describe the marginal distribution of nucleotides at each position. Although very simple, it carries several advantages over simpler, non probabilistic models such as the "projected $k$-mers" or IUPAC based motifs. As we have shown in Section **??** this model is able to capture specific preferences at each position and also different levels of specificity in positions.

Aiming to relax the PSSM's inherent independence assumption, we will now review alternative representations using richer models for binding sites.

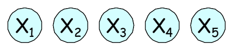
### Mixture of PSSMs

Suppose that the transcription factor can have several "types" of binding. These might correspond to slightly different physical configurations of the protein at the binding site, each with somewhat different preferences. Thus, it can bind to any sequence that fits any one of the configurations. To model this case, we assume there is an additional, unobserved, random variable $T$ that describes the type of the binding. We have some prior probability $P(T)$ (possibly uniform) over the type of binding, and we assume that for each type the positions are independent, just as in the PSSM case. This requires a distribution $P(X_i \mid t)$ over the nucleotide at the $i$'th position given the value $t$ of $T$. The joint probability of the observed positions require that we sum over all possible values of $T$

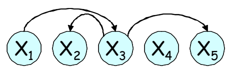
$$P(X_1 \ldots X_K) = \sum_{t=1}^{C} \left[ P(t) \prod_{i=1}^{K} P(X_i \mid t) \right]$$

That is, the distribution is a mixture of PSSM representations where the mixing probabilities are determined by $P(T)$ and $C$ is the number of mixture components in our model *i.e.*, the number of binding "types".

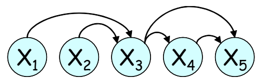
A mixture model has several benefits both in terms of representation and in terms of semantic interpretation. First, the number of parameters is fairly small: $C-1$ parameters for $P(T)$, and $3KC$ parameters for the conditional probabilities $P(X_i \mid T)$. Second, as suggested above, this model offers an important representational concept. Although the nucleotides are dependent when $T$ is unknown, they are independent when $T$ is observed. Thus, $T$ plays the role of a hidden biological mechanism the renders the positions independent. Understanding the interaction between the hidden mechanism $T$ and the nucleotides at each position can provide insights about the physical protein-DNA interactions.

(a) 

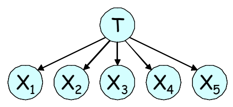
$$P(X_1)P(X_2)P(X_3)P(X_4)P(X_5)$$

(b) 

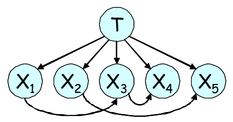
$$P(X_1)P(X_2 \mid X_3)P(X_3 \mid X_1)P(X_4)P(X_5 \mid X_3)$$

(c) 

$$P(X_1)P(X_2)P(X_3 \mid X_1, X_2)P(X_4 \mid X_3)P(X_5 \mid X_3, X_4)$$

(d) 

$$\sum_T P(T)P(X_1 \mid T)P(X_2 \mid T)P(X_3 \mid T)P(X_4 \mid T)P(X_5 \mid T)$$

(e) 

$$\sum_T \begin{array}{l} P(T)P(X_1 \mid T)P(X_2 \mid T)P(X_3 \mid T, X_1) \\ P(X_4 \mid T, X_3)P(X_5 \mid T, X_2) \end{array}$$

Figure 3.1: Examples of different Bayesian network models for a motif with 5 positions. For each model, we show an example of a Bayesian network structure, and the corresponding representation of the joint distribution. (a) PSSM, (b) tree, (c) non-tree, (d) mixture of PSSMs, and (e) mixture of trees.

## Bayesian Networks

Mixtures of PSSMs capture "broad" dependencies among all the positions via the $T$ variable. An alternative approach to describe dependencies is to consider how each position depends on the others. For example, a particular nucleotide in position 1 might cause the conformation of a particular amino acid side-chain to change. This, in turn, affects the conformation of other amino acids in the binding site, and may have an effect on the preference of binding in position 3.

One representation designed to capture "local" dependencies is the language of *Bayesian Networks*. In this representation, we use a *directed acyclic graph* $G$ to represent the dependencies. The vertices of $G$ correspond to the random variables $X_1 \ldots X_K$ and a parameterization which describes a conditional distribution for each variable given its immediate parents in $G$. The corresponding joint probability distribution decomposes into the product form:

$$P(X_1 \ldots X_K) = \prod_{i=1}^{K} P(X_i \mid \mathbf{Pa}_i^G) \tag{3.1}$$

Where $\mathbf{Pa}_i^G$ is the (possibly empty) set of parents of $X_i$ in $G$. Figure **??** (a)–(c) show few examples of Bayesian networks, and their associated form of the probability distribution.

The formal semantics of Bayesian Networks is in terms of conditional independence statements: Each position $X_i$ is independent of its non-descendants in $G$ given its parents in $G$ (see Chapter **??**). In general, the more edges we have in $G$, the more complex the dependencies between positions are. The simplest network has no edges, like the one in Figure **??**(a). It is easy to see that in this case, the distribution is simply a PSSM. For the general case, the number of parameters in the networks depends on the number of edges: The conditional distribution $P(X_i \mid \mathbf{Pa}_i^G)$ requires $3 \cdot 4^{|\mathbf{Pa}_i^G|}$ parameters.

### Tree Bayesian Networks

One sub-class of Bayesian network that we want to single out is the class of *tree* Bayesian networks. In tree models each position has at most one parent, making $G$ a forest. For example, the networks in Figure **??**(a) and (b) are both tree networks. Tree networks also generalize first-order Markov chains (where $G$ is $X_1 \rightarrow X_2 \rightarrow \ldots \rightarrow X_K$). They provide a flexible language for modeling dependencies, while limiting the number of parameters to be at most $3 \cdot 4K$. Another important benefit of this class of models is that there are efficient algorithms to learn the best tree structure (**??**).

### Mixture of Trees

In some cases, a tree structured network might be too limited. One possible approach of enriching the representation is to combine the benefits of a tree structure with the added richness of a hidden mechanism. This leads to a natural extension, similar to mixture of PSSMs, that is a mixture of trees. Each $X_i$ has as parents the hidden variable $T$ and at most one other nucleotide. The network of Figure **??**(e) is an example of this model. Intuitively, the unobserved variable $T$ enhances the ability of the tree to model additional dependencies while multiplying the number of parameters only by a factor of $C$. An important advantage of mixture of trees is that, similarly to trees, there exist efficient algorithms for learning the best structure (**??**).

## 3.2 *LearnMotif*: Learning Motif Models

### 3.2.1 Learning from Aligned Data

We start with the more simple setting of learning motif models from aligned sequences. This is actually an instance of the the well studied problem of learning Bayesian networks from data we reviewed in Section **??**.

**Parameter Learning** Given a fixed structure $G$, our training dataset $\mathcal{D}$ consist of $M$ aligned binding sites. We denote by $x_i[m]$ the value of $X_i$ at the $m$'th example. Our task is to find the parameters $\theta$ that maximize the average (log)-probability of each sample over data. Formally:

$$\ell(G, \theta : \mathcal{D}) = \boldsymbol{E}_{\hat{P}}[\log P(X_1, \ldots, X_K \mid G, \theta)] \tag{3.2}$$

Where $\hat{P}$ is the empirical distribution over $\mathbf{X}$. As we have shown in Section **??**, the *maximum likelihood* parameters for $P(X_i \mid \mathbf{Pa}_i^G)$ are simply the matching empirical conditional distributions in $\hat{P}$. parameter learning in this case reduces to estimating marginal probability from $\hat{P}$. Since we usually have a small number of training examples, we *smooth* the maximum likelihood estimates by using *Dirichlet priors* (see Section **??**). This amounts to adding a small number (5 in our experiments) of *pseudo instances* that are distributed according to a background distribution.

In models where we have a hidden variable $T$, parameter estimation is somewhat more complex. We need to perform the iterative procedure of *Expectation Maximization* (*EM*) as described in Section **??**, to find a (local) maximum of the likelihood function.

**Structure Learning** In addition to estimating parameters, we also want to learn the dependency structure $G$, *i.e.*, which edges to include. For this we use the *BDe score* (Section **??**). This score can be thought of as likelihood penalized by a term that accounts for differences in model complexity, to avoid over-fitting. In models that do not have a hidden variable $T$, finding the best graph is a combinatorial optimization problem. For tree networks, this problem can be reduced to a maximum weighted forest problem and solved efficiently (**??**). For general Bayesian networks, this problem is intractable, and we resort to using a heuristic search.

In models where a hidden variable $T$ is present, the situation is more complex. First, we also need to decide on the cardinality of $T$. Moreover, the structure score becomes intractable and we need to resort to an approximation. We use the the Cheeseman-Stutz approximation of the BDeu score (**?**). To choose the cardinality of $G$, we evaluate the score at each cardinality. For a mixture of trees model where we also face the problem of structure learning, we use, similarly to trees, an efficient maximum weighted forest algorithm (**??**).

### 3.2.2 Learning from Unaligned Data

We now return to the *ab initio* motif finding problem. Here we get promoter sequences for genes suspected of co-regulation $G$, and are asked to identify biding sites that can "explain" this co-regulation. With no known binding sites to use, we try and learn a motif model for the binding site and identify its matching binding site locations along the sequences. Just as we suggested in the previous chapter, we should aim to learn a binding site model that has highly probable $k$-mers in the set of promoters $G$ yet has relatively much fewer such $k$-mers in $\mathcal{G}/G$. We consider such a discriminative motif to be a good candidate to explain co regulation.

## Sequence Model

The generative probabilistic framework we use now is similar to the one we used in Section **??**. In short, it assumes sequence can be generated either by the motif model $M$ or some background model $BG$. The main difference from Section **??** is that we now allow very general models for $M, BG$. The only practical constraint is they should support efficient inference of full/partial assignments. Another difference is that $BG$ is now modeled explicitly. In practice we use a $k$-order Markov chain for $BG$.

Translating the model's assumptions to formal representation we get that the probability of generating a given sequence if it does not contain a binding site is given by:

$$P(S \mid r^f) = \prod_{i=1}^{L} P_{BG}(S^{(i)} \mid \langle S^{(i-k)}, \ldots, S^{(i-1)} \rangle) \tag{3.3}$$

Where, as before $S$ is the generated sequence, $R$ is the variable denoting the regulation event occurring ($r^t$) or not ($r^f$) [1], and $P_{BG}$ is the probability by the background model. If we know that the sequence is regulated we have:

$$P(S \mid r^t, \mathcal{G}) = \sum_{h=1}^{L-K+1} P(h) P(S \mid r^t, h, \mathcal{G}) \tag{3.4}$$

where $P(H)$ is the prior probability over all possible locations of the binding site in the sequence. With no prior knowledge this would simply be a flat prior $P(H = h) = \frac{1}{L-k+1}, \forall h$.

When the binding site location is known, *i.e.*, for a given value $h$ of $H$, the probability of $S$ is:

$$P(S \mid r^t, h, \mathcal{G}) = P(S \mid r^f) \frac{P(S^{(h)}, \ldots, S^{(h+K-1)} \mid \mathcal{G})}{\prod_{i=h}^{h+K-1} P_{BG}(S^{(i)} \mid \langle S^{(i-k)}, \ldots, S^{(i-1)} \rangle)} \tag{3.5}$$

Our input consists of $M$ sequences that belong to $\mathcal{G}$. Just as we relaxed the assumption that $R[m]$ is observed and allowed to incorporate prior biological knowledge into our model in Section **??**, we assume here that $R[m]$ is hidden, but we have some *prior belief* about its value $P(R[m] = r^t) \in \{0, 1\} \forall m$ (We will soon show how prior knowledge from expression or ChIP measurements can be incorporated into $P(R[m])$). The likelihood of the observed (sequence) data according to this

[1] We use this and not $R \in \{1, -1\}$ we had in Section **??** as it simplifies the notations in this case.

model is given by:

$$LL(\mathcal{D} \mid M) = \sum_m \log(P(R[m] = r^t)P(S \mid r^t, M) + (1 - P(R[m] = r^t))P(S \mid r^f)) \quad (3.6)$$

Where we dropped $BG$ from the above equation because we assume it to be fixed and given (*e.g.*, estimated from all yeast promoter regions) and not part of the learning process.

The last three equations also gives us insight to what kind of motif model $M$ would maximize the data likelihood. Sequences which are believed not to be regulated ($P(R[m]) \sim 0$) contribute to the likelihood via the second term of Eq. (**??**), and are (as expected) relatively independent of $M$. However, for regulated sequences ($P(R[m]) \sim 1$) we see via Eq. (**??**) that it is beneficial to have a motif which gives a high probability ratio to some locations, compared to the background. This boosts the second term in Eq. (**??**) which in turn boosts the likelihood in Eq. (**??**). This means that learning a *discriminative* motif, which assigns high probability to subsequences *common* in regulated genes yet are *unique* compared to sequence phenomena captured by the background model, is beneficial for maximizing the data likelihood.

Finally, we should also point out that modifying current model to allow multiple binding sites in each sequence can be achieved quite easily by altering Eq. (**??**). It would then have the form:

$$P(S \mid r^t, \mathcal{G}) = \sum_n P(N = n) \sum_{\vec{h}=\{h_1,\ldots,h_n\}} P(\vec{h})P(S \mid r^t, h_1, \ldots, h_n, \mathcal{G})$$

Where P(N) is the prior on the number of binding sites locations in the sequence, and we enumerate over all the assignments for (non overlapping binding site locations ($\vec{h}$). As we can see, the relaxation of the single motif assumption has only a secondary affect on the likelihood function via the summation over $N$, but is much more complex to carry out (summation over all possible $\vec{h}$ assignments is exponential in $n$). We therefore chose to avoid it in our implementation in sake of computational efficiency.

### Incorporating Biological Observations

An important feature of this model is that we can easily incorporate prior biological knowledge into it via the $P(R = r^t)$ parameter. This parameter represents the prior probability that a sequence is regulated, *before* we see the actual sequence. This prior belief may come for example from clustering co-expressed genes or from ChIP experiments. We use here the same approach we described already in Section **??**. Briefly, this means we introduce a random variable $O$ that denotes our *observation* about the gene and is dependent of the regulated status of the gene, but not on its promoter

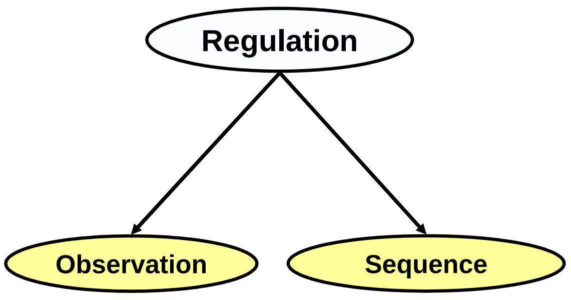


Figure 3.2: Graphical representation of the relations between the variables in the *LearnMOTIF* framework. The observed entities are colored yellow.

sequence. Under this assumption, the overall probability of the observation and the sequence is

$$P(S, O) = \sum_r P(r)P(O \mid r)P(S \mid r)$$

Thus, we view $O$ as a *noisy sensor* of the underlying biological regulation. Given $O$, we can readily calculate the *posterior* probability of regulation given our observation via Bayes rule. Of course the actual affect of the noisy sensor observation depends on the way $P(O \mid R)$ is chosen. We use here the same representation as we did in Section **??** to handle prior knowledge from gene expression or ChIP experiments.

The resulting model's structure with the noisy sensor observation is illustrated in Figure **??**. It reflects the independence between $S$ and $O$ given $R$, and also the generative approach we take, where we try to best explain (*i.e.*, maximize the likelihood) of the observed data $(S, O)$ by adjusting the model's parameters. The following section gives the details of this.

### Learning

The algorithm's input is a dataset $\mathcal{D}$ that consists of $M$ promoter sequences $S[1], \ldots, S[M]$, and their associated observations $O[1] \ldots, O[M]$. We want to learn a motif model that maximizes the log-likelihood of the data

$$\ell(\mathcal{D} : \mathcal{G}) = \sum_{m=1}^{M} \log P(S[m], O[m] \mid \mathcal{G}) \tag{3.7}$$

We assume the background distribution $P_{BG}$ is known and fixed. Our task amounts to estimating the structure and parameters of the motif model. Unfortunately, due to the fact that both $R[m]$ and $H[m]$ are unknown, there is no simple estimation procedure for this task. Instead, we use an *Expectation Maximization (EM)* (**?**) approach. Recall the EM algorithm is a form of hill climbing and is guaranteed to improve the likelihood at each iteration. It uses the current model to "complete" the probability of the hidden values. Given such a completion, we no longer have missing values and a *maximum likelihood* model is computed analytically. We then use the new model for completing the data and so forth. The procedure iterates until it convergences to a (local) maximum. In cases where we need to learn Bayesian network structure, we us the *Structural Expectation Maximization (SEM)* algorithm (**?**). Since both *EM* and *SEM* were already reviewed in Section **??**, we give here only the details for their specific implementation in this case.

For models where the structure is fixed and we learn using maximum likelihood (PSSMs and mixtures of PSSMS), we define EM as progressing through a sequence of models $\mathcal{G}^0, \mathcal{G}^1, \ldots$ such that

$$\mathcal{G}^{t+1} = \arg\max_{\mathcal{G}} Q(\mathcal{G} : \mathcal{G}^t, \mathcal{D})$$

where

$$Q(\mathcal{G} : \mathcal{G}^t, \mathcal{D}) = \sum_{m,r,h} P(r, h \mid S[m], O[m], \mathcal{G}^t) \cdot \log P(S[m], r, h \mid \mathcal{G}) \tag{3.8}$$

A useful property of $\log P(S, R, H \mid \mathcal{G})$ is that it can be further decomposed into a sum of terms. Using Eq. (**??**)–(**??**) and discarding of the terms that do not involve $\mathcal{G}$, we find that maximizing $Q(\mathcal{G} : \mathcal{G}^t, \mathcal{D})$ is equivalent to maximizing

$$\sum_m \sum_h P(r^t, h \mid S[m], O[m], \mathcal{G}^t) \cdot \log P(X_1 = S^{(h)}[m], \ldots, X_K = S^{(h+K-1)}[m] \mid \mathcal{G})$$

This problem is in the form of Eq. (**??**) but where we now consider each $K$-mer in the input sequences as a training sample. Essentially, each $K$-mer is taken into account while learning $\mathcal{G}^{t+1}$ in proportion to our *current* (posterior) belief that it is a binding site. This is achieved probabilistically by weighing each $K$-mer by the term:

$$W(S^{(h,\ldots,h+k-1)}) = P(R[m] = r^t, H[m] = h \mid S[m], O[m], \mathcal{G}^t) \tag{3.9}$$

which is exactly the probability of regulation given the previous model. This is given by:

$$P(R[m] = r^t \mid S[m], \mathcal{G}^t) = \frac{P(R[m] = r^t) P(S[m] \mid R[m] = r^t, \mathcal{G}^t)}{P(S[m] \mid \mathcal{G}^t)} \tag{3.10}$$

and the posterior probability of a specific $K$-mer being a binding site given the motif model $\mathcal{G}^t$ is:

$$P(H[m] = h \mid S[m], R[m] = r^t, \mathcal{G}^t) = \frac{P(h) P(S[m] \mid H[m] = h, R[m] = r^t, \mathcal{G}^t)}{P(S[m] \mid R[m] = r^t, \mathcal{G}^t)} \tag{3.11}$$

Since the *EM* (and *SEM* for that matter), only converge to a *local* optimum, it is crucial to have a good starting point, which, in our case, is an initiation of the motif's ($M$) parameters. For this we use the *SeedSearcher* algorithm described in previous chapter. [2] This allows us to start from a sequence pattern which was found to be discriminative in relation to $G, \mathcal{G}$, and then refine it, using the procedure we just described. Because the actual motif might be wider then the initial pattern identified by the *SeedSearcher*, we usually expend the initial motif, typically 5bp to each side. As we explained, the score we use during learning contains a complexity term. Therefore, if the added positions are not part of the motif they are found uninformative and the learning process will simply ignore them.

To summarize this section, the *LearnMOTIF* algorithm for learning probabilistic *cis*-regulatory sequence motifs is as follows:

1. Given sequence set $\mathcal{G}$ and $P(R[m] = r^t), \forall g_m \in \mathcal{G}$ we use the *SeedSearcher* algorithm described in Section **??** to find the best discriminative crude motifs (usually, projected $k$-mers described in Section **??**).

2. For the crude motifs found to be statistically significant (Section **??**), we perform initial learning as described in Section **??**. For this we take all $k$-mers that match the the crude motif in sequences with a prior on regulation $R[m] \geq 0.5$ and treat them as (aligned) samples of the motif. (as explained above, we actually use the $k$-mers along with their 5bp up/down flanking regions). If we need to perform *EM* at this stage, we typically do only a few ($< 5$) iterations. The aim of this stage is to get a good initial guess of the motif.

3. Perform optimization of the initial motif by performing the *EM* with the following two phases:

    - **E-step:** use $\mathcal{G}^t$ to compute the weight for each $K$-mer in the input sequences, as given by Eq. (**??**).
    - **M-step:** Set $\mathcal{G}^{t+1}$ to be the model learned by the procedures of Section **??** using the weighted dataset of the E-step.

    If we need to perform *SEM*, we use the same *E* step, but choose the best scoring model for $\mathcal{G}^{t+1}$ applying the procedure for structure evaluation described in Section **??**.

    It is important to note that following the freedom we had in choosing a model in Section **??**, we can consider any Bayesian network in the M-Step.

[2]The *SeedSearcher*'s default settings are to define the set $G$ of regulated genes as those with $P(R[m] = r^t) > 0.5$ from the total set of genes $\mathcal{G}$

## 3.3 *CIS*: Evaluating Statistical Significance of Motif's Scores

In the last section we developed the technique to learn sequence motif which is common to a set of genes $G$ suspected of co regulation. We now face the same problem we had in Section **??**: In order to handle the motif finding task, we need some way to measure the statistical significance of specific locations in long promoter regions and decide weather to report them as putative binding sites. As in Section **??**, our probabilistic, generative, framework assumed sequences are generated either from a background model $BG$ or the motif model $M$. It is therefore appealing to use the same *random generation* approach. According to this, we use the *log odds* to score any $k$-mer:

$$\mathcal{S}(S^{(i,i+k-1,\ldots,)}) = \log(\frac{P_M(S^{(i,i+k-1,\ldots,)})}{P_{BG}(S^{(i,i+k-1,\ldots,)})} \tag{3.12}$$

And the statistical significance of a score $\mathcal{S}$ is measured by the $p$-value of $\mathcal{S}$ *i.e.*, the probability of achieving this score or higher according to the background distribution over $k$-mers.

$$\boldsymbol{E}_{P_{BG}(X)}[1\{\mathit{Score}(X) \geq \mathcal{S}\}] \tag{3.13}$$

Where $1\{\}$ is the indicator function, $P_{BG}(X)$ is the background distribution over a random variable $X$ that ranges over possible DNA motifs, and $\mathit{Score}()$ is the scoring function as defined above. In our case $X$ is a subsequence of a fixed length $k$.

Although the description up to this point is totally identical to the one we had in Section **??**, it is important to note that there are two main differences. First, the motif model $\mathcal{G}$ is not necessarily a PSSM but can be any probability model. The only practical restriction we set is that we could easily sample from it and perform inference on full/partial assignments to it. The second difference is that now we have *explicit* representation of both the background and the binding site model $\mathcal{G}$. In principal, we could still use the branch and bound approach of *TestPSSM* to estimate the $p$-value of scores given by Eq. (**??**). However, the fact in current setting we have explicit probabilistic models for both background and motif enabled us to develop a much more efficient algorithm for this task. With the *CIS* algorithm we will now describe we were able to handle motifs 22bp long with ease, an infeasible task for *TestPSSM*, even with simple PSSM motif models.

The idea behind the *CIS* algorithm is efficient sampling. Let us first revisit the sampling approach for approximating $p$-values. We hinted before that the formulation of Eq. (**??**) suggests a simple procedure for estimating the $p$-value of a score $S$: We sample i.i.d samples from $P_{BG}(X)$, and then compute the fraction of samples whose scores are as high as $S$. Such a *naive sampling* procedure can reliably estimate $p$-values that are at least two orders of magnitude larger than the inverse of the number of samples. However, typical application involves scanning long sequences for putative binding sites. In this scenario, we treat each subsequence as a putative binding site, and then correct for multiple testing (*e.g.*, **?**). Assuming a typical promoter length of size 500bp, these corrections result in the need for estimating $p$-values in the order of $10^{-5}$ or even lower, rendering the naive sampling approach impractical as it requires the order of $10^7$ samples.

To solve the inefficiency problem of the naive sampling approach, we suggest here a novel method to estimate $p$-values we term *CIS* (for Compound Importance Sampling). The algorithm uses the *importance sampling* approach (**?**), and allows us to conceptually mimic the naive sampling approximation but with a significantly smaller number of samples. As we show the result is also a much more efficient and accurate estimation of the scores $p$-values.

We start with reviewing the importance sampling technique and then describe how we create the mixture probabilities we use in the *CIS* algorithm. We finish with a short evaluation section, demonstrating both the accuracy and efficiency of the algorithm.

## 3.3.1 The Compound Importance Sampling Approach

Importance sampling (**?**) is a general method for estimating $\boldsymbol{E}_{P(X)}[f(X)]$, where $f(X)$ is some function over the random variable $X$, and $P(X)$ is a distribution measure defined over $X$. Instead of using samples from $P(X)$, it uses samples from a *proposal distribution* $Q(X)$. It is especially useful in cases where sampling directly from $P(X)$ is not possible. The method relies on the following equalities

$$\begin{aligned}\boldsymbol{E}_{P(X)}[f(X)] &= \sum_x P(x)f(x) = \sum_x P(x)f(x)\frac{Q(x)}{Q(x)} \\ &= \sum_x Q(x)\left[f(x)\frac{P(x)}{Q(x)}\right] = \boldsymbol{E}_{Q(X)}[f(X)w(X)] \qquad (3.14)\end{aligned}$$

where the weight $w(x) = \frac{P(x)}{Q(x)}$ compensates for the bias introduced by sampling from $Q(X)$ rather then $P(X)$. Estimating $\boldsymbol{E}_{P(X)}[f(X)]$ using samples from $Q(X)$ is therefore basically the same procedure as naive sampling, but here each sample $x$ is re-weighted according to $w(x)$. The total weight of the samples in this case is not the number of samples ($N$) but rather $\sum_i^N w(x_i)$.

To apply this general framework to estimate $p$-values, we use the formulation of Eq. (**??**):

$$p\text{-}value(\mathcal{S}) = \boldsymbol{E}_{P_{BG}(X)}[1\{Score(X) \geq \mathcal{S}\}]$$

where $f(X)$ for this case is the indicator function $1\{Score(x) \geq S\}$. The main decision in applying importance sampling, is the choice of the proposal distribution $Q$. For the log-odds score, values of $X$ sampled from the binding site model are more likely to receive high scores. Naively, we can set $Q$ to be the distribution of the binding site model $P_M(X)$, and directly sample from the region of high scores. This approach, however, is problematic since we need a fair amount of low scoring samples, even when estimating the $p$-values of high scoring samples. Without such samples the empirical histogram of observed scores would be biased towards high scores, and the empirical estimation of the $p$-values would be poor. This problem is illustrated by Figure **??**, where the distribution over scores by $P_M(X)$ is plotted in the full brown line. As we can see, This distribution hardly covers the low to mid ranges of scores assigned by $P_{BG}(X)$ (the full black line).

One possible solution is to sample a mixture of $n_1$ samples from the model distribution $P_M$ and $n_2$ samples from the background $P_{BG}$. This is equivalent to sampling from

$$Q(X) = \frac{n_1}{n_1 + n_2} P_M(X) + \frac{n_2}{n_1 + n_2} P_{BG}(X) \tag{3.15}$$

While this solution takes into account both extremes, in practice it still suffers from poor estimation of the "middle-ground" scores. In Figure **??**, we see an illustration of this where sampling from just the background and motif model (black and brown full lines, respectively), leave the mid range scores hardly covered. Thus, we refine the above approach and consider a combination of richer set of models. We define *Compound Importance Sampling* (CIS) as:

$$Q(X) = \sum_i m_{\alpha_i} Q_{\alpha_i}(X) \tag{3.16}$$

where $\mathcal{A}$ is an index set and $m_{\alpha_i}$ is the fraction of samples generated from the model $Q_{\alpha_i}$. The models $\{Q_{\alpha_i}\}$ are basically "smoothed" versions of $P_M(X)$ that bias it in different degrees toward $P_{BG}(X)$. This set of distributions is illustrated by the set of colored and dashed lines representing the various distributions used by CIS.

### 3.3.2 Characterizing *CIS*'s Mixture Distributions

The main question we now face is how to define the set of mixture component's $\{Q_{\alpha_i}\}$, that will effectively "smooth" $P_M(X)$ towards $P_{BG}(X)$. Specifically, we want to create distributions $\{Q_{\alpha_i}\}$ that will enable us to use a relatively small number of samples and, at the same time, provide a good approximation for the distribution of $Score(X)$ over the entire range of scores.

The method we suggest is based on the following formulation: let $X = X_1 \ldots X_k$ denote the $k$ binding site's positions. Using the basic chain rule for any multi variable distribution we can write:

$$P(X) = \prod_{i=1}^{k} P(X_i \mid X_1 \ldots X_{i-1}) \tag{3.17}$$

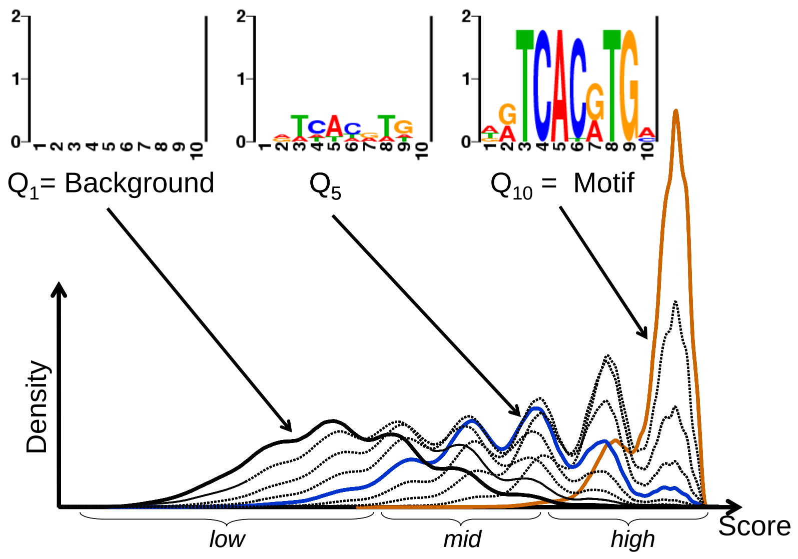


Figure 3.3: Illustration of the CIS approach. At the top there are three sequence logos of components of the proposal distribution, ranging from the binding site model (right), to the background model (left) where the information content of each position is low. At the bottom, the distributions of log-odd scores for the components of the proposal distribution are plotted, with the 3 matching the sequence logos at the top colored. The green ovals mark the score ranges which are under sampled if you use just $Q_1$ or a linear combination of $Q_1$ and $Q_{10}$.

where P is either $P_{BG}(X)$ or $P_M(X)$. We can now define $Q_\alpha$ as:

$$Q_\alpha(X) = \prod_{i=1}^{k}[\alpha P_m(X_i \mid X_1 \ldots X_{i-1}) + (1-\alpha)P_{BG}(X_i \mid X_1 \ldots X_{i-1})] \qquad (3.18)$$

It can easily be shown that $Q_\alpha$ is indeed a proper distribution.

The basic difference in the above formulation from Eq. (**??**) is that it mixes the conditional probability of each position in the binding site separately. As an example, consider a simple case in which the the background distribution is a 0-order Markov model favoring Guanine ($G$) with high probability for each position, and the binding site model is a 5bp long PSSM favoring Adenine ($A$) with high probability for each position. In this case sampling from a distribution defined as Eq. (**??**) would result in many high scoring $AAAAA$ as well as many low scoring $GGGGG$ samples. "Mid range" samples such as $GAGAG$ would be very rare, degrading the evaluation of the overall $p$-value range. When using Eq. (**??**) with $\alpha = 0.5$, on the other hand, $GAGAG$ is as likely as the two extremes when assuming that positions are independent of each other.

The idea of *CIS* is illustrated in Figure **??**. The top part shows the sequence logo of the background ($Q_{\alpha_0}$) and the binding site ($Q_{\alpha_{10}}$) defining the extreme distributions as well as one of the middle ground mixture distributions ($Q_{\alpha_5}$). The effect of sampling from them as well as from other "in between" distributions is illustrated by the bottom part where the corresponding score distributions are scores. It is easy to see to using just one or two of the extreme distribution does not cover will the whole range of scores while the CIS approach gives a much more tight cover of all the score range which in turn lead to better efficiency too, as much less samples are needed to get an accurate estimation of the $p$-values. We demonstrate this in the following section.

We emphasize that the formulation is not limited to any specific form of distribution. The only assumption we make in Eq. (**??**) about $P_M(X)$ or $P_{BG}(X)$ is that we can compute the conditional probability of $X_i$ given a specific value assignments to the preceding variables. In using $Q_\alpha$, we either sample an instance or compute the probability of a given instances. In both cases, these are the only queries we need. Also note, that we choose the order of variables in which to perform this expansion. In some cases, a specific order (*e.g.*, one that conforms to the topological order in a Bayesian network) can lead to more efficient computations.

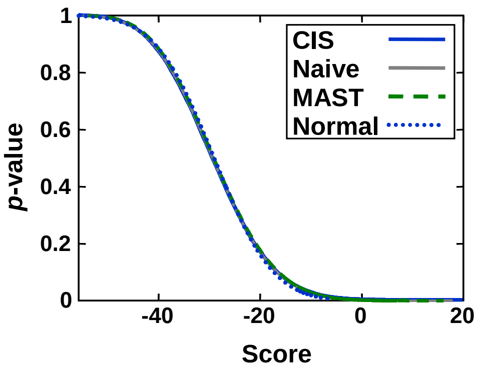
In the case of Bayesian networks, Eq. (**??**) and consequently Eq. (**??**), decompose according to the dependency model where each $X_i$ is dependent on a (typically small) number of additional positions. In the case of PSSM for example, each position is independent of all other positions, while in $K$-order Markov model each position depends only on $K$ others. Therefore using Eq. (**??**) rather than the naive mixture of Eq. (**??**) results in an inherently different set of samples. Similarly to the above illustrative example, this "smoothed" mixing results in better coverage of the middle ground score range and leads to an overall accurate estimation of $p$-values.

Finally, when using *CIS* we have to decide on the number of components as well as the number of samples and degree of smoothing ($\alpha$) for each component. In this work we use 10 components for which the $\alpha$ coefficient vary linearly from 0 to 1. The number of samples drawn from each component decreases exponentially, starting from 10,000 samples from $P_{BG}(X)$ to 1,000 samples from $P_M(X)$. It should be noted that the CIS method was proved to be robust in a wide range of settings.

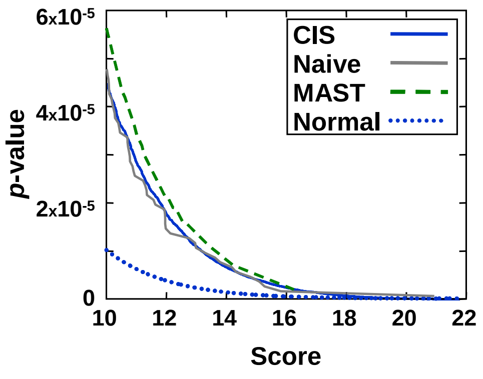
### 3.3.3 Evaluation

As a case study, we examine the binding site model of RAP1 in *Saccharomyces cerevisiae* from TRANSFAC 7.3 (**?**), which is 14bp-long. We estimated the $p$-value of each score using the following methods: CIS algorithm using $40,000$ samples from a proposal distribution as illustrated in Figure **??**; MAST (**?**); functional approximation by normal distribution, where we estimate the mean and variance of $Score(X)$ according to $P_{BG}(X)$, and then use the tail probability of normal distribution as the $p$-value estimate.

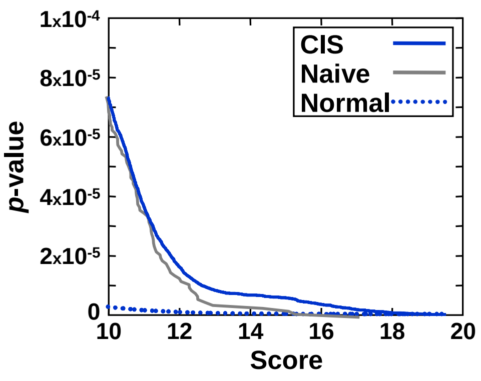
As a proxy to the truth, we computed the $p$-values using the naive sampling procedure with $10^9$ samples as described above. Figure **??**(a) compares the $p$-value estimates by the different methods.



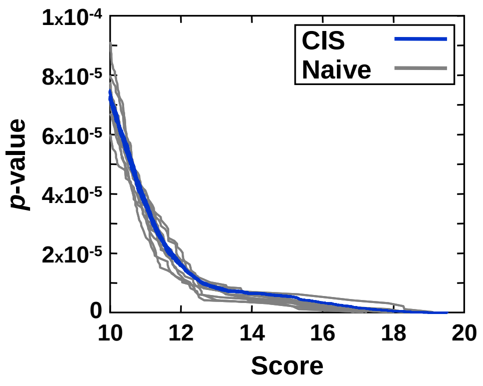
(a)



(b)



(c)



(d)

Figure 3.4: $p$-value estimations for putative binding sites. Compared are the Naive sampling approach using $10^6$ samples, the normal approximation, MAST, and our CIS method (with $40,000$ samples). Shown is the $p$-value (y-axis) as a function of the log-odds score (x-axis). (a) For the position independent model of TRANSFAC's RAP1; (b) Same as (a), zoomed on $p$-values $< 10^{-4}$; (c) For the dependency model of PHO4 learned by **?** (here MAST is not applicable); (d) Test for robustness with 10 repeats of (c).

While initially all methods appear the same, zooming into the region of interest in Figure **??**(b) reveals significant discrepancies. It is evident that the normal approximation is inaccurate in this region. Both CIS and MAST provide accurate estimations, with a slight advantage to the CIS method.

Figure **??**(c) shows evaluation of $p$-values for a binding site model with dependencies between positions learned by **?** for the PHO4 transcription factor. For models such as this, MAST is not applicable. As we can see, the estimations of CIS are similar to the direct sample estimate. One might suspect that the slight deviation observed between the two curves is due to the smaller number of samples used by CIS. However, when comparing ten repetitions of each procedure shown in Figure **??**(d), we see that CIS estimates are more robust then the ones by naive sampling while using two orders of magnitude less samples.

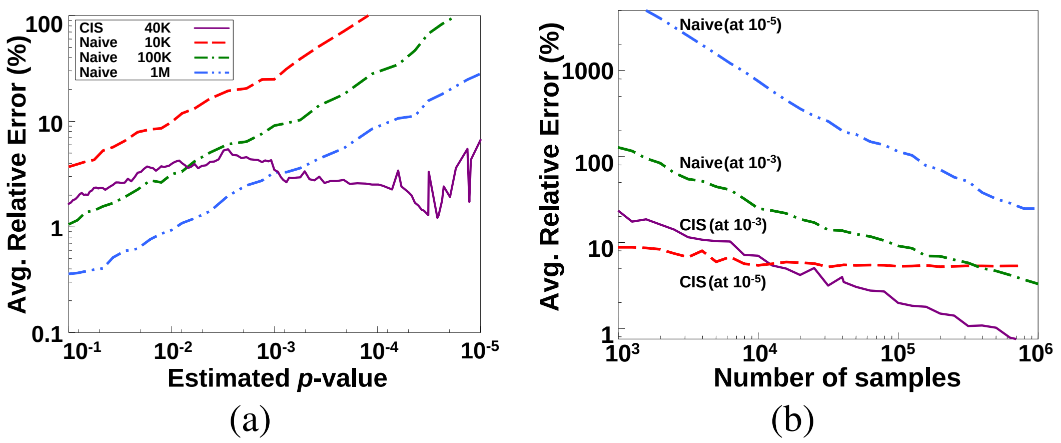


Figure 3.5: Robustness of $p$-value estimation. $10^9$ samples from a $3^{rd}$-order Markov background model were used to approximate the scores' $p$-values to the order of $10^{-5}$. The binding site model is TRANSFAC's F_CBF1_B. For each setting 50 runs were made to compute the average relative error in $p$-value estimation as a function of the estimated $p$-value (a) and as a function of the sample size at $p$-value $10^{-3}$ and $10^{-5}$ (b).

The robustness of the CIS estimator is furthered illustrated in Figure **??**. Here the relative error in $p$-value estimation is plotted as a function of the estimated $p$-value (a), and of the sample size for two fixed $p$-values (b). In both graphs the advantage of using CIS with only 40,000 samples over using naive sampling with $10^6$ samples is clear, particularly for $p$-values lower than $10^{-3}$. As we can see, CIS with 40,000 samples provides a good compromise between efficiency and accuracy.

So far we demonstrated the effectiveness of CIS with respect to the background distribution directly. We conclude by demonstrating our approach on a real-life genome-wide scan. Using the *Chromatin immunoprecipitation* location analyses of **?**, we excluded all the promoter regions of *S. cerevisiae* genes, that were found to be targets of the ZAP1 transcription factor. We then used the dependency model for ZAP1's binding sites by **?** to scan the promoter regions of remaining genes of *S. cerevisiae*. Given that we removed ZAP1 targets, we expect that the remaining promoters will contain only few real binding sites of ZAP1. We used a $3^{rd}$-order Markov model as a background distribution, and plotted the empirical frequencies of high scoring subsequence.

We note that in such a "real life" test, the accuracy of the results can be affected by factors other then the sampling technique used. These include the accuracy of the binding site and background models, and the fact we use i.i.d subsequence samples instead of long sequences. Figure **??** shows that once again the normal approximation results in poor $p$-value estimates. More importantly, using the models described above and 40,000 i.i.d samples, the CIS method provides accurate estimations over a wide range of $p$-values.

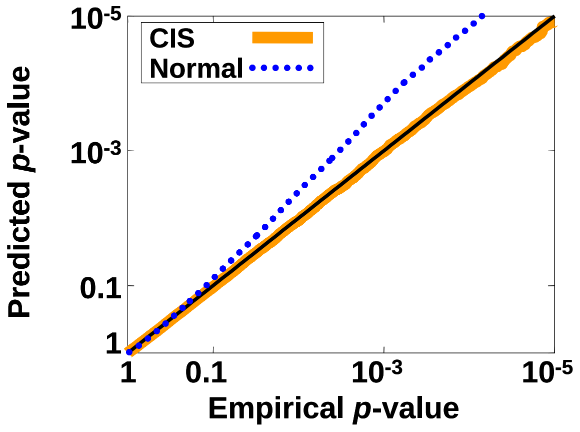


Figure 3.6: $p$-value estimation for a genome-wide scan of *S. cerevisiae* using the ZAP1 dependency model learned by **?** with a $3^{rd}$-order Markov background model. Shown are the normal approximation of the $p$-values and the CIS estimates compared to the empirical frequency of these scores.

## 3.4 Discussion

In this chapter we expanded the probabilistic representation of DNA motifs using the language of Bayesian network. Our framework allows any model spanning the range from the position independent PSSM to the full dependency model. Specifically, the model classes we used included trees, mixture of PSSMs, and mixture of trees. We described methods to learn these models from limited data in both the *aligned* sequences case, were our input is made of known binding site samples, and for the *unaligned* sequences case. In the later case we perform *de novo* discovery of motifs in unaligned genomic sequences suspected of co-regulation. We also presented an efficient algorithm based on importance sampling (*CIS*) for discovering binding sites given any Bayesian network motif model. *CIS* enabled us to assess the statistical significance of each putative site and control the false discovery rate.

We are not the first to model dependencies between positions in biological sequence motifs. **?** suggested the tree network model, and discussed algorithms for learning it. In a related problem of modeling splice junctions, recent works examined $k$-order Markov models (**?**) and tree Bayesian networks (**?**). These works learned models from aligned binding sites and used them to detect splice junctions in new sequences. Bayesian networks were also used to model dependencies between positions in protein motifs that were aligned according to 3-D structure (**?**). This domain introduces another layer of complications due to the large alphabet size of amino acids. However, our work was the first one that presented a general framework for learning *any* Bayesian network motif model in *de novo* discovery of transcription factors binding sites. As we will show in the following chapter, this ability can lead to dramatic improvements in the learned motifs.

The work we presented here can be extended in several directions. First, the advantage of being able to model a motif using any Bayesian network suggests further exploration of different types of models as well as general unrestricted models. This can include representational extensions that are geared toward the complexity vs. expressiveness issue such as *context specific* dependency models (**??**). Second, as our framework made no particular assumption on the type of binding sites, it can be readily adapted to discover other sequence motifs such as those of splicing and histone remodeling factors.

The models we developed here may also have interesting implications about protein-DNA interactions. The challenge is how to relate the identified dependencies between positions to protein structure and function. For this purpose, we need to be able to estimate our confidence in the discovered dependencies (*e.g.*, using bootstrap (**??**) or Bayesian methods (**?**)) and relate these dependencies with three dimensional conformations of Protein-DNA complexes.

The biggest immediate challenge posed by the experimental results we got using this modeling technique (see Section **??**) is how to automatically select between different dependency models (including the PSSM model). A natural measure for predicting performance of probabilistic models on test data is the *Bayesian score* (Section **??**). Unfortunately, in our experiments the *Bayesian score* was not successful in selecting one of the best models.

Comparing the *LearnMotif* and *LearnPSSM* algorithm presented in previous chapter we note both algorithms use a similar probabilistic framework. This includes a motif model, a background model and an implicit generative assumption that the observed sequence is generated solely from the background model or that it also contains a (single) binding site sampled from the motif model. The difference between the two algorithms is mainly in the function they optimize and the probabilistic models used. *LearnPSSM* implicitly assumes a Markov order 0 probabilistic model for the background but avoids the need to model it implicitly. The motif model is a PSSM and the algorithm optimizes the conditional posterior of regulation events given the observed sequence, using gradient ascent. *LearnMotif* uses an explicit (fixed) model for the background which can be any Bayesian network for that matter. The motif model is also represented as a Bayesian network, allowing the algorithm to capture dependencies between biding site positions. Unlike *LearnPSSM*, *LearnMotif* optimize a generative function using EM, namely the likelihood of the observed data. As we explained in Chapter **??**, algorithms that follow this generative approach may tend to converge to motifs that are indeed common to the genes suspected to be co-regulated, but are not necessarily unique to them. However, both algorithms we presented use the same discriminative motif as a starting point for motif refinement, based on the *SeedSearcher* results. An immediate question that follows from this analysis regards the relative effect of optimizing the different scoring function used by the two algorithms and the effect that modeling dependencies between biding site positions has. We will focus on these in the next chapter were we do systematic evaluation of the motif finding algorithms we developed.

The common probabilistic framework used by the two algorithms also implies about the common limitations of both and directions of extending this framework. For example, in both cases we assume a fixed length of the motif model so a binding site model with a gap made of a variable number of positions in the middle is not well represented. In the case of *LearnMotif* a partial compensation for this may be achieved by learning a motif model with direct dependency between binding site positions reflecting the variable gap length. Another solution may be to learn a mixture of PSSMs model where each motif with a different gap length is represented by a separate mixture component. For both *LearnMotif* and *LearnPSSM* more accurate motif models may be achieved by combining more sources of data as prior information about binding site location. Both algorithms share the probabilistic framework that includes only one motif and a fixed background model. Therefore even if several different motifs are present in a promoter (as is often the case in transcriptional regulatory programs) the learning task only involves one motif at a time (rather then optimizing several simultaneously).

In any case, the motif finding problem as we defined it is set to a specific group of genes given to us as input. The context by which this group is defined (like the cellular conditions in which we suspect these genes are co-regulated) is not part of the learning process. To move beyond this, in later chapter we will show how we combine more sources of data, like expression and ChIP, to learn the actual regulatory programs while utilizing the techniques developed here for motif learning and binding sites predictions. But before we go into these higher level models, we turn to perform a systematic evaluation of the motif finding algorithms we presented.

# Chapter 4

# Motif Finding Evaluation

In the last two chapters we described novel solutions to the motif finding problem. Given a set of sequences ($G$) suspected of co regulation, we defined this problem as finding a sequence pattern that can serve as a possible explanation for co regulation. The output are binding site locations for the assumed, unknown, regulator. The solutions we presented were all model based *i.e.*, we first learn a model for the binding site representation, and then, using a probabilistic framework, evaluate the statistical significance of putative sites. We suggested taking a *discriminative* approach, searching for a motif which was not only *common* to the sequences in $G$, but also relatively *unique* to them. This was in contrast to other computational methods (*e.g.*, **??**) that use a generative approach and search for the most *common* motif possible for sequences in $G$. The first step in our approach was to identify statistically significant crude motifs using the *SeedSearcher* (Section **??**). We then defined a generative probabilistic framework in which sequences can be generated either by a motif model $M$ or a background model $BG$. We used this framework to learn finer representation of the crude motifs found by the *SeedSearcher*. In Section **??**, we used this framework in the *LearnPSSM* algorithm to learn a PSSM motif model while using a discriminative scoring function. In Section **??** we used the same framework and the same crude motifs as starting points for the *LearnMotif* algorithm. In this case we learned not just PSSM models but also more complex motif representations modeling interdependent positions in the binding site. We also used a different target function in the learning process which was more generative in nature, maximizing the overall likelihood of the observed data.

When we approach evaluating this set of algorithms, there are several issues we should aim to check. The first thing that probably comes to mind is the affect our discriminative approach has compared to the most commonly used solutions. A related question that follows is the effect of switching from the discriminative target function of *LearnPSSM* to that of *LearnMotif* when refining motifs. Finally, we need to address the question we posed at the beginning of Section **??**.

Our main focus there was on learning motif representations that relax the independence between positions assumption. The question we started with and supplied us with our original motivation concerned the added value of learning more complex models compared to standard PSSM. We will try and answer this by a systematic evaluation of various performance criteria we now turn to define.

## 4.1 Evaluation Procedures

The first thing we need to define is the evaluation criteria we use. Turns out this is quite problematic for algorithms handling the motif finding task. Probably the most obvious bottom line test would simply be to compare binding site predictions to known ones using a procedure like the one illustrated in Figure **??**. Here we have the exact binding site location in a long promoter that was not used to as a train sample for the motif model. We scan the promoter and assign each position a $p$-value score using the *CIS* algorithm (Section **??**). Setting a threshold $t$ on the Bonferroni corrected $p$-value , we report as putative binding sites all positions with a better score (*i.e.*, a lower $p$-value) and compare them to the known site. As we can see in this case when $t = 0.01$, scanning the promoter with the learned mixture of trees motif model (blue bars) enabled us to correctly predict the known site. Scanning with the PSSM motif model (red bars) learned using exactly the same input , does not indicate any position as a putative binding site (all $p$-values are higher then $0.01$) and the best scoring one is actually a false one. Lamentably, this kind of test data is relatively rare. We usually do not have the exact binding site locations of transcription factors within promoters, verified experimentally. Moreover, in the biological world there are usually no definite negative samples verified *not* to bind a regulator. As it happens, regulators usually bind to gene promoters under specific cellular conditions. Even if an experiment was to indicate non binding at a specific location, it might always be the case there is a binding in some other conditions different from the ones used in the experiment.

Recently, several researchers joined in an effort to define a test set for algorithms handling the motif finding problem (**?**). They defined several criteria to test performance (*e.g.*, the ratio of positions correctly reported as binding sites from to the total amount of positions reported) and tested all algorithms publicly available on the web. However, their datasets contains only very few positively labeled examples to train on (around 4 in some cases). Moreover, since all the algorithms they used were generative in nature, no negative train set was supplied. These limitations rendered the use of their dataset impractical for our algorithms.

We were interested in defining test procedures which will enable us to evaluate our methods on large datasets. For this, ChIP experiments proved to be a valuable asset. Recall these assays measure the binding affinities of a target transcription factor to promoter regions *in vivo*. The experimental protocol of **?** assigns a $p$-value to each promoter sequence. A sequence with $p$-value less than $0.001$

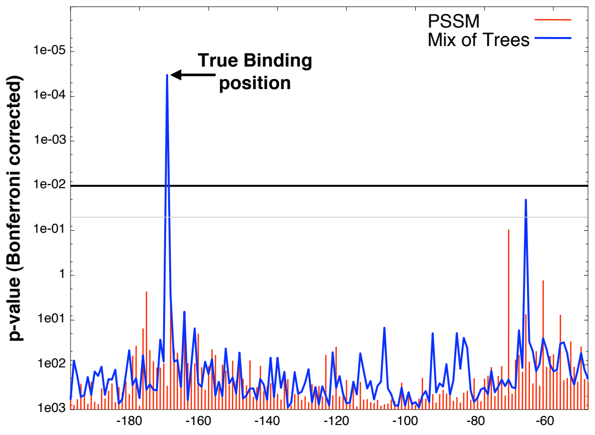


Figure 4.1: Example of detection of a motif in an upstream region. For each position ($x$-axis) we show the Bonferroni corrected $p$-value ($y$-axis) assigned by two different motif models learned by *LearnMotif.* Shown is the promoter region of Gal80 scanned with motif models learned from location assay of Gal4 (Galactose)

is considered to be bound by the factor. The stringent threshold of 0.001 is aimed to reduce the false positive identifications in a genome wide screening (**?**). This means that although these experiments do not pinpoint the exact binding location, they can supply us with an objective evaluation of the ability of our learned motifs to detect the sequences that the transcription factor binds to. We use these datasets by scanning promoter sequences just as in Figure **??**, and report as targets all genes found to have significant sites (*i.e.*, sites with $p$-value below some threshold $t$). Comparing these predictions to the ChIP experimental results, we get a measure of the sensitivity and specificity of our target gene predictions. [1] Formally, if we denote the predicted target genes with respect to the true labeling as either $TP$ (true positive, *i.e.*, true targets identified), $FP$ (false positive), $TN$ (true negative) or $TN$ (true negative) we have:

$$\begin{aligned} Sens(t \mid M) &= \frac{TP}{TP + FN} \\ Spec(t \mid M) &= \frac{TP}{TP + FP} \end{aligned} \tag{4.1}$$

Where $M$ is the model we used to perform the genomic scan and compute $p$-values as in Section **??** and $t$ is the threshold on the $p$-values used to make the target predictions. Using different thresholds for the target calls results in a different sensitivity specificity tradeoffs. Higher threshold results in more genes found to have significant sites and labeled as targets. As we can see from the above formulation this increases the sensitivity measure on the expense of lowering the specificity. This tradeoff can be represented graphically in an ROC curve as the one in Figure **??**. In the ROC

[1]Of course if we use the ChIP data also to train our models as described in Section **??**, then to keep the evaluation objective we test performance on sequences that were not seen during training. To achieve this, we perform a 5-fold cross validation test. In each of the 5 runs we use 80% of the yeast genes as training data, and tested the learned motif on the remaining 20% of the genes.

representation any threshold $t$ corresponds to some point along the ROC curve so its $y$-axis is the sensitivity and the $x$ value is the *false discovery rate* ($FDR = \frac{FP}{FP+TN}$) at this threshold $t$. For example, In the case presented in Figure **??** we see a comparison between 4 different motif models learned for the same transcription factor. In this case it is evident that whatever the threshold $t$ the mixture of trees model outperforms the rest.

In general, the answer which curve is better is not always that clear. We need some measure to quantitate performance over many such ChIP datasets. One possible solution is to compare performance by the point along the ROC that maximize the *hyper-geometric* score of the predictions with respect to the targets labeling by the ChIP experiment. Formally, if $G$ is the group labeled as targets by the experiment, we search for a threshold $t^*$, such that the induced detections in the group $G$ will be most significant.

$$t^* = \arg\min_t \textit{p-value}(G, I_{(M,t)})$$

Where $I_{(M,t)}$ is the group of predicted targets by using $t$ as the $p$-values threshold. This means we adjust the threshold $t$ over the models scores so that the event defined by $(M, t)$ has the smallest *hyper-geometric* $p$-value with respect to $G$. The correlation of the motif $M$ to the group $G$, also termed *specificity score* by **?**, is defined as:

$$\textit{p-value}(G, I_{(M,t^*)}) \tag{4.2}$$

The above measure is good for quantitating and comparing prediction performance. However, for practical usage of motif finding algorithms we usually need a different measure. Performing genomic wide scans in high amounts of false positive identifications. To control the false positive error rate, we select in advance some threshold $t$ on the (Bonferroni corrected) $p$-value of putative sites and report only sites with $p$-value below this point. Note this corresponds to a specific point along the $x$ axis of the ROC curve, matching the false positive rate we are willing to tolerate. If we need to compare different algorithms or motifs performance we then compare their sensitivity and specificity for this predefined $t$.

Similar to ChIP, high throughput gene expression measurements can also be facilitated to evaluate the performance of motif finding algorithms. One way is to use them is by first finding clusters of co-expressed genes and treating these cluster tags just as we did the target/non-targets labels by the ChIP experiments. The idea behind this is that if co-expression is the result of co-regulation then a more accurate motif would better correlate with the clustering tag. However, in practice this method is limited since the correlation between co-clustering and co-regulation is not so direct. We therefore used this approach only when we were limited to compare to previously published results by other motif finding algorithms.

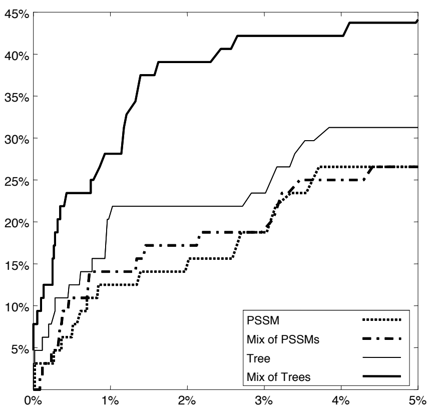


Figure 4.2: ROC performance analysis using ChIP data for the various motif models learned for MSN2 ($H_2O_2$). The $x$-axis shows the *false positive rate*, FP/(FP+TN), the $y$-axis shows the *true positive rate* (sensitivity) TP/(TP+FN).

Another possible way to use gene expression measurements is to test the correlation between target prediction to expression profiles in relevant experiments. For example, when learning the motif for the heat shock transcription factor HSF1 we can test whether the expression of predicted targets differs from the rest under heat shock treatment. Again, remember that our predictions for target genes are *based solely on sites identifications* of *de novo* learned motifs. The logic behind this kind of a test is that predicted targets by a more accurate motif model would have a more distinct expression profile. To measure the distinction between the expression profile of target/non-target genes we used the *Kolmogorov Smirnov* $p$-value (**?**). The concept of this measure is illustrated in Figure **??**. We compute the empirical c.d.f over expression of targets and non-targets genes. The *Kolmogorov Smirnov* test then finds the value for which the two c.d.f are most different $l$ (marked with a black arrow) and attach a $p$-value to this, taking into account the number of samples used in each c.d.f . The $p$-value corresponds to the *null hypothesis* that the two c.d.f were actually sampled from the same distribution but still got a distance as large as $l$ between the two empirical c.d.fs . We state that a motif $M$ has high correlation with the expression in some experiment $E$ when we get a low KS $p$-value ($< 10^{-5}$) when comparing the expression profiles of targets/non-targets genes as predicted by $M$. Figure **??** demonstrates this analysis approach for the RAP1 transcription factor and expression measurements with Histone $H4$ depletion. This was a time affect experiment, so we have 7 different experiments along the $x$ axis. The $y$ axis gives the $p$-value for the distinction of expression distribution between target/non-target genes. As we see in this case the predictions based on the mixture of trees motif model demonstrate a far better correlation with expression then the predictions based on the PSSM motif model. As a reference, we can also plot how well the target predictions based on ChIP experiments do, when available. In general, we would expect them to do better in most cases, since these are biological experiments directly testing for target genes. Surprisingly, we see that in this specific case the targets predicted by the mixture of trees model do

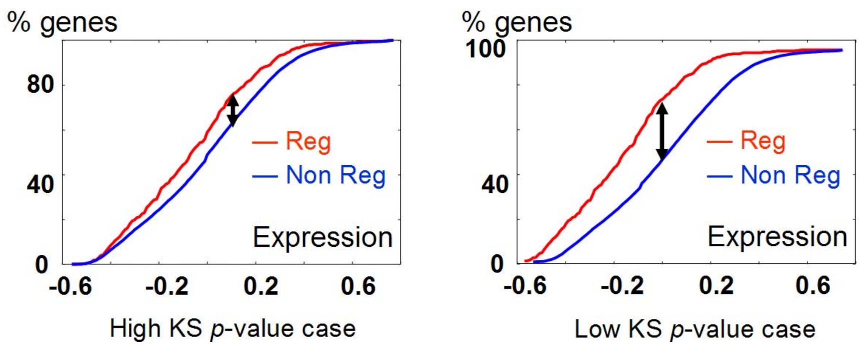


Figure 4.3: Illustration of the *Kolmagorov Smirnov* (KS) $p$-value for distinction between gene expression of target and non-target genes.

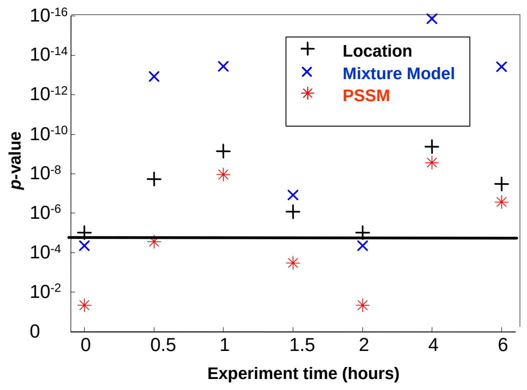


Figure 4.4: Correlation between target prediction for the RAP1 factor and gene expression measurements in histone $H4$ depletion (**?**). The $y$ axis is the computed KS $p$-value for the expression profiles of the target/non-target groups. In black are results from using targets identified by the ChIP experiments of **?**. In blue and red results from using target prediction based on motif binding sites identifications. The blue crosses where obtained using the mixture of trees motif, red by using the matching PSSM motif model. The vertical line corresponds to a $10^{-5}$ threshold over $p$-values for determining significance.

even better.

The last evaluation measure we define is the one usually used in machine learning with probabilistic models: the (log) likelihood when using (unseen) test data. In our case the likelihood tested was the one a learned motif model assigns to unseen samples of biologically verified binding sites. This capture the notion of how well our model was able to generalize the concept learned - that of a binding site motif.Specifically, for each dataset containing the binding sites of a known factor we performed *10-fold cross validation* tests and used the average log-probability per instance as a performance measure.

After defining the evaluation procedures, we now move to the actual evaluation of the issues we listed at the beginning of this chapter. We start with the effect the *discriminative* approach has.

| Source/ Cluster | Trans. Factor | Consensus | Seed rank | Seed p-value | PSSM rank | PSSM p-value | MEME $\leq 8$ rank | MEME $\leq 8$ e-value | MEME $\leq 50$ rank | MEME $\leq 50$ e-value |
|---|---|---|---|---|---|---|---|---|---|---|
| Spellman et al. (?) | | | | | | | | | | |
| CLN2 | MBF | ACGCGT | 1 | 4e-26 | 1 | 3e-42 | 1 | 1e-18 | 1 | 7e-31 |
| SIC1 | SWI5p | CCAGCA | 1 | 1e-07 | 1 | 1e-12 | 1 | 8e-00 | 8 | 5e+02 |
| Tavazoie et al. (?) | | | | | | | | | | |
| 3 | putative | GATGAG | 2 | 9e-07 | 5 | 6e-09 | 4 | 1e+06 | 2 | 1e-14 |
| | putative | GAAAAatT | 3 | 4e-07 | 2 | 1e-11 | 23 | 8e+07 | 3 | 7e-10 |
| 8 | STRE | aAGGgG | 1 | 6e-07 | 3 | 4e-06 | 20 | 1e+08 | – | – |
| 14 | putative | TTCGCGT | 1 | 2e-09 | 2 | 7e-11 | 13 | 1e+07 | – | – |
| | putative | TGTTTgTT | 3 | 2e-07 | – | – | – | – | 13 | 4e+05 |
| 30 | MET31/32p | gCCACAgT | 1 | 2e-11 | 1 | 2e-11 | 2 | 5e+02 | 8 | 1e+03 |
| Iyer et al. (?) | | | | | | | | | | |
| MBF | MBF | ACGCGT | 1 | 1e-12 | 1 | 3e-18 | 3 | 1e+04 | 19 | 1e-03 |
| SBF | SBF | CGCGAAA | 1 | 1e-32 | 1 | 1e-37 | 2 | 1e-17 | – | – |

Table 4.1: Selected results on binding site motifs in several yeast datasets, comparing our findings with those of MEME.

## 4.2 The Discriminative Approach Effect

The first test we did to get a sense of the effect of our discriminative approach was to compare our results on several previously published datasets to the ones obtained by the MEME algorithm. Using a variant of our *LearnPSSM* algorithm (**?**), we compared to MEME's results, run in two configurations. The first restricted MEME to retrieve only short patterns of width 6–8, corresponding to initial *SeedSearcher* search setting we used. The second configuration used MEME's own defaults for pattern retrieval resembling our end product PSSMs. The datasets we used were all yeast related. They included two of Spellman's cell cycle related clusters (**?**), gene clusters made by **?** based on $k$-means clustering of yeast cell cycle expression data, groups of genes defined by **?** and identified by experimental methods to be regulated by the MBF/SBF transcription factors complexes, and finally the clusters 'M", and "I/J" reported by **?** as related to yeast stress response.

The results of this evaluation are summarized in Table **??**. In general we were able to find the motif previously reported for these gene groups in all cases. In some cases, the best seeds according to the *SeedSearcher* yield the best scoring PSSMs. More often, the best scoring PSSM corresponds to a seed lower in the best scoring seeds list [2]. In most cases the PSSM learned to recognize regions flanking the seed sequence. In some cases more conserved regions were discovered. A few of the learned PSSMs are presented in Figure **??**.

Comparing our results to those of MEME we see it also identified the shorter patterns. However, there are two marked differences. First, MEME often gave top scores to spurious patterns that are clear artifacts of the sequence distributions in the promoter regions (such as poly A's). That is the result of using a generative model that optimizes a motif *common* to the clusters tested, but not necessarily *unique* to them. Second, there was a major difference in running time. Running the

[2]we took into account only seeds that had $p$-value passing the FDR threshold, as described in Section **??**. These are listed by the *SeedSearcher* according to their $p$-value score.

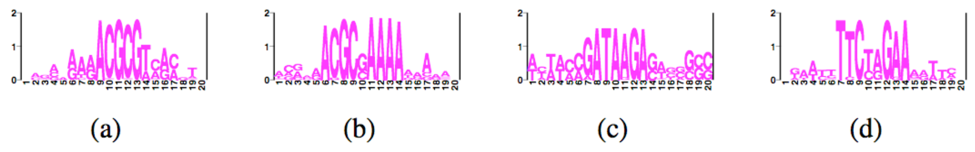
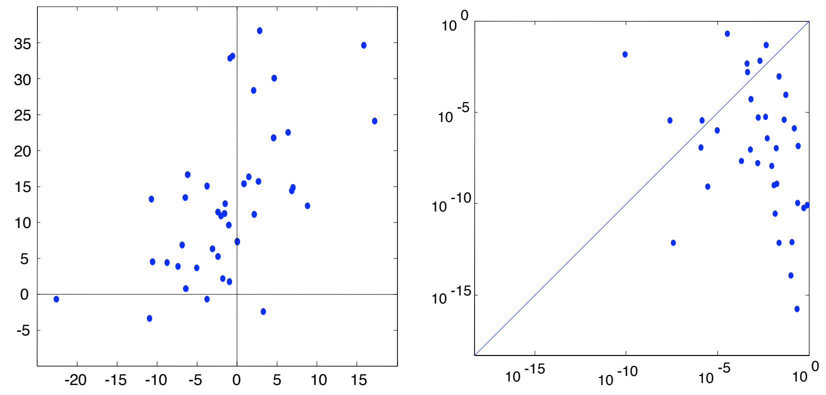


Figure 4.5: Examples of PSSMs learned by our procedure. (a) CLN2 cluster. (b) SBF cluster. (c) Gasch *et al.* Cluster M. (d) Gasch *et al.* Cluster I/J.

*SeedSearcher* took a few minutes and optimizing the PSSM can take an hour or so. The shortest MEME run on the same datasets took about an hour, while longer runs took days when asked to return only the top thirty patterns.

In a more systematic evaluation of the discriminative approach effect, we did a second test using a rich collection of datasets of gene clusters collected by the Church lab (**??**). These clusters are based on functional annotations, co-expression, and known targets of transcription factors. They were originally analyzed using AlignAce. AlignAce is another commonly uses program for motif finding using an MCMC-based procedure for finding common PSSM patterns in a given set of sequences. Since AlignAce suffers in general from a similar problem as MEME, *i.e.*, convergence to spurious patterns such as poly A/T, the results reported by **??** included multiple runs of AlignAce, followed by filtering of the motifs based on various quality criteria they defined. Only the best PSSMs were reported for each cluster. Our learning procedure on this dataset was based on the *LearnMotif* algorithm (Section **??**). To avoid adding the effect of complex motif models, we set *LearnMotif* to learn only PSSM motifs, just as AlignAce does. This means we first learned crude motifs using the *SeedSearcher* and then refined them to PSSM motifs, using a generative probabilistic model similar to MEME, (see Section **??**). As an evaluation criteria, we used the procedure we described in Section **??** Where we treat the cluster tags as the "true" labeling and evaluated the ability of the motifs learned to predict the labeling. [3] For the Evaluation of sensitivity and specificity of the predictions we set $t = 0.01$ as the $p$-value threshold for reporting putative sites. Figure **??** (a) plots the *difference* in sensitivity and specificity for each dataset. Each dataset (*i.e.*, a cluster of genes) appears as a dot with axis value being the difference in sensitivity ($y$) and specificity ($x$) between our model's predictions and that of AlignAce's. This means points in the upper right quadrant represent datasets where our model performed better in terms of both sensitivity and specificity, points in the bottom left quadrant are ones where the AlignAce algorithm performed better, while points in the other two quadrants are ones where the two methods achieve different tradeoffs between sensitivity and specificity. As we can see, our method had improved the PSSM's performance in 14 cases and reduced it in 3. The main observation is the different tradeoff between sensitivity and specificity of the two learning techniques. However, when we use the best hyper

[3]Note that since the original papers used all cluster members for learning and we only compare models of the same complexity, no cross validation was done in this case.



(a) (b)

Figure 4.6: Comparison of PSSMs learned by our method to the ones learned by AlignAce on the gene clusters of **?**. (a) Difference between the PSSMs learned by method and by AlignAce in sensitivity ($x$-axis) and specificity ($y$-axis). (b) Comparison of the hypergeometric $p$-value of PSSMs learned by AlignAce ($x$-axis) and the ones by our method ($y$-axis).

geometric $p$-value to asses motif's prediction performance as in Eq. (**??**), We see (Figure **??** (b)) that in most examples (32 out of 41) our PSSM obtained a more significant hypergeometric $p$-value.

To summarize, we see that the discriminative approach we presented proves to be beneficial both in terms of efficiency and in prediction performance. There are two more important observations to be made, regarding the evaluations we just described. First, note that in the first evaluation tests we used a variant of *LearnPSSM* and in the later we used a variant of *LearnMotif* set to learn only PSSM motifs. The two algorithms share the initial starting point for motif learning but use a different target function and optimization techniques during learning. Moreover, the target function used by *LearnMotif* when learning PSSMs is very similar to the one used by MEME algorithm, the reference algorithm in the first test. These two observations makes the question of comparing *LearnMotif* and *LearnPSSM* algorithms even more intriguing. We handle this in the following section.

## 4.3 *LearnPSSM* vs. *LearnMotif*

Both *LearnPSSM* and *LearnMotif* algorithms presented in previous chapters use a similar probabilistic framework. The two major differences between them is the target function they optimize during motif learning, and the fact *LearnMotif* is not limited to PSSM motif models. In the following test we used the PSSM motif model in both algorithms and the same initial starting point (a crude motif found by the *SeedSearcher*). Under these conditions difference in results can be attributed to the difference in target function optimization techniques used. Recall that *LearnPSSM* optimizes a discriminative function based on the conditional probability of the regulation events using gradient ascent, while *LearnMotif* optimizes the overall likelihood of the observed data, and

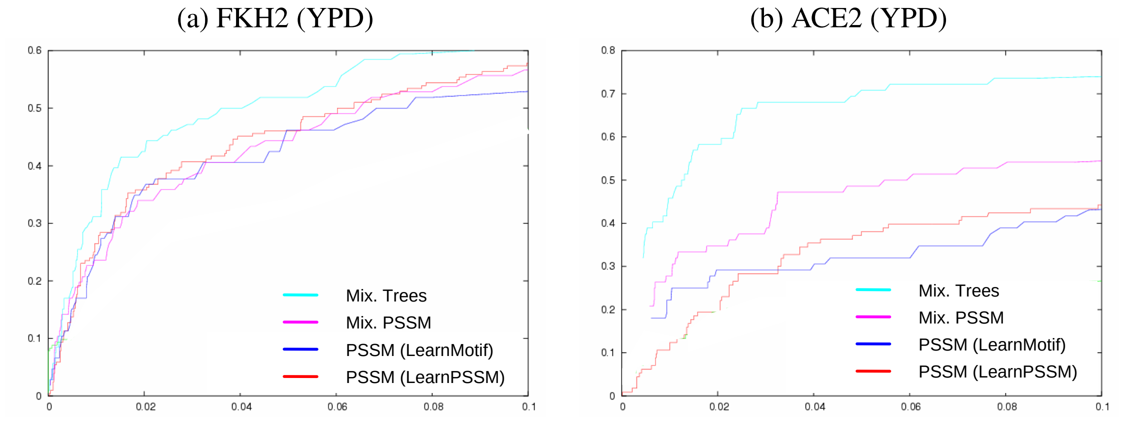


Figure 4.7: ROC curves showing target prediction accuracy using ChIP data for FKH2 (in YPD) and ACE2 (in YPD) by **?**. The different motif models are marked in different colors.

uses EM.

To evaluate the success of the different methods, we tested them using ChIP data as described in Section **??**. We selected 15 datasets by **?** for targets of known transcription factor in yeast. [4] Figures **??**(a) and (b) show ROC curves for two examples: FKH2 (YPD) and ACE2 (YPD). As we can see in these two examples, there are no dramatic differences in performance between the two methods when learning PSSM motif models. Over all the 15 datasets tested, similar motifs were learned. In some cases there was an advantage to the *LearnPSSM* procedure of learning PSSMs and in others *LearnMotif*'s PSSM proved to be better. In all cases learning the PSSM motifs using *LearnMotif* was much less time consuming.

From this and the previous section we conclude that both *LearnMotif* and *LearnPSSM* have comparable performance when learning PSSM motifs from the crude motif supplied by the *SeedSearcher*. As we saw earlier, the discriminative approach proved to be beneficial both in terms of time and prediction accuracy, compared to generative algorithms commonly used. Since *LearnMotif* uses a similar framework as the generative MEME algorithm does when learning PSSMs, we conclude that the difference, at least for PSSM motif learning, can be heavily attributed to the discriminative starting motif supplied by the *SeedSearcher*.

Finally, it is clear from Figure **??** that the difference in performance between different technique to optimize PSSM's is minor compared to the difference in performance we get by using more complex motif models that capture inter dependency between binding site positions. We test this much more throughly in the next section.

[4]The 15 datasets were selected so as to have a representative set of *LearnMotif* performance over the original set of more then 100 ChIP assays of **?**.

## 4.4 The Effect of Modeling Interdependent Positions

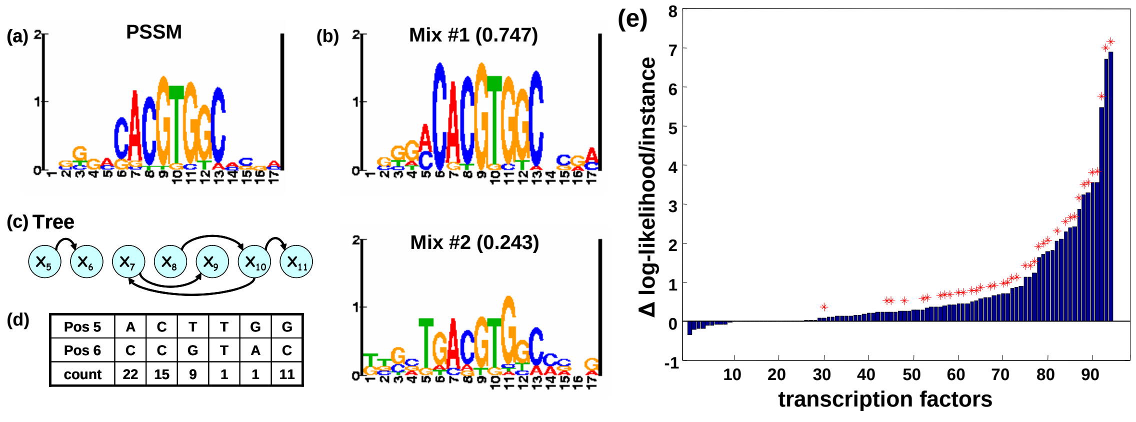
The most extensive evaluation we did was on the effect of modeling dependencies between binding site positions. We tried to gain a broad and objective perspective by exploiting high throughput data as described in the introduction to this chapter (Section **??**). In all the evaluation procedures we now review, the PSSM motif model was used as a baseline to gauge the relative performance of other representations such as mixture of trees, mixture of PSSMs and trees. In all cases, the *LearnMotif* algorithm was used to learn the motif model (PSSM or other), starting from the same crude motif supplied by the *SeedSearcher*.

### 4.4.1 Generalization from Known Binding Sites Samples

When we set out to develop the *LearnMotif* algorithm we started with the pragmatic question "Would modeling dependencies between binding site positions be beneficial?". Our main aim was to improve the accuracy of biding site predictions. We defined two distinct motif finding tasks. The *second* one involved learning a motif from "unaligned sequences" *i.e.*, given long promoters as input. The *first* and easier one was when we already had the exact binding sites positions at hand. We called that the "aligned sequences" motif finding problem. In the probabilistic framework we defined this simply boiled down to learning a Bayesian Network subclass of models (*e.g.*, mixture of trees, mixture of PSSMs) from given samples.

Since the first task involves learning an explicit probabilistic model for the binding site from given samples, we can test model performance by testing the (log) likelihood it assigns to unseen test data (*i.e.*, new binding site samples), as explained in Section **??**. This gives us a notion how well the motif model generalizes and captures the binding site representation. We examined 95 transcription factors for which there were 20 or more sites in the the TRANSFAC database (**?**). For each transcription factor, we evaluated the ability of each of the representations (mixture of trees, mixture of PSSMs, tree and PSSM models) to describe the distribution of sequences at the binding site of that factor.

As a specific example, consider the sites of TRANSFAC identifier `P$ABF_Q2` (*Arabidopsis* ABA responsive element binding factor). The associated dataset consists of 49 binding sites, each 17bp long. A PSSM learned from this dataset (see Figure **??**(a)), shows position 5 as uninformative, and position 6 as weakly informative. When we learn a mixture of PSSMs, we get a mixture between the two PSSMs shown in Figure **??**(b). As we can see, these two PSSMs differ mostly in positions 5 and 6. When we learn a tree Bayesian network, we get the dependency structure shown in Figure **??**(c). The strongest dependency is between position 5 and 6. We can see this dependency by examining the occurrence table for both positions, Figure **??**(d). As we can see, when position 5 is 'T', position 6 is 'G' with high probability, and when position 5 is not 'T',



|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Pos 5 | A | C | T | T | G | G |
| Pos 6 | C | C | G | T | A | C |
| count | 22 | 15 | 9 | 1 | 1 | 11 |

Figure 4.8: Comparison of dependency models and PSSM, learned on aligned binding sites from the TRANSFAC database (**?**). (a)–(d) One example of evident dependency in the bindings sites for TRANSFAC `P$ABF_Q2` motifs. (a) PSSM model; (b) mixture of PSSMs model (2 components); (c) the dependency structure tree model (only middle positions are shown); (d) the occurrence table for positions 5 and 6; (e) difference of average log-likelihood per instance on test data between the best dependency model and the PSSM model (y-axis) for 95 transcription factors (x-axis). Asterisks mark statistically significant results (paired t-test with $p < 0.05$).

position 6 is 'C' with high probability. In our cross validation test, the three methods achieves log-probability of $-19.93$, $-18.70$, $-18.47$ bits per instance for PSSMs, mixture of PSSMs, and trees, respectively. Qualitatively, we can say that by using a dependency model we were able to detect a real phenomenon in the data that was "smoothed out" by the over simplistic PSSM model.

Figure **??**(e) summarize our results on the 95 TRANSFAC datasets by comparing the best dependency model with the PSSM model. We evaluated the statistical significance of these differences using a paired t-test on log-probability of particular instances. For 12 cases, the PSSM model performed better. In all 12 cases, the differences were not statistically significant. For 14 cases, there are no noticeable differences, and indeed the learned dependency models in these cases show weak correlation. For 69 cases, the dependency models were better than PSSMs. For 51 of these case, the improvement was statistically significant. When we consider individual methods, the tree networks were better in 33 cases (22 of these were significant), mixtures of 2 PSSMs were better in 59 cases (36 significant), and mixtures of 2 trees were better in 57 cases (35 significant).

To summarize, these results give strong support to the claim that in many cases modeling dependencies between binding sites positions can significantly improve generalization performance on new unseen binding sites. We now move to see how this translates to actual binding site predictions.

| | PSSM Generated | | | | | | Tree Generated | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Learned Model | TP = 100; FP = 0 | | TP = 50; FP = 50 | | TP = 25; FP = 75 | | TP = 100; FP = 0 | | TP = 50; FP = 50 | | TP = 25; FP = 75 | |
| PSSM | 68%,56% | 2e-95 | 65%,51% | 1e-86 | 64%,57% | 3e-89 | 68%,65% | 1e-101 | 68%,65% | 1e-101 | 67%,63% | 7e-99 |
| Tree | 68%,56% | 2e-95 | 65%,52% | 5e-87 | 66%,56% | 3e-92 | 85%,66% | 6e-135 | 82%,70% | 1e-132 | 80%,66% | 1e-124 |
| Mix of PSSMs | 67%,55% | 9e-93 | 61%,48% | 1e-78 | 53%,46% | 3e-65 | 70%,64% | 2e-104 | 66%,59% | 1e-94 | 67%,57% | 1e-94 |
| Mix of Trees | 60%,48% | 4e-77 | 43%,38% | 2e-47 | 40%,48% | 3e-49 | 72%,62% | 9e-107 | 67%,54% | 2e-92 | 64%,56% | 2e-88 |
| True | 67%,54% | 2e-92 | 67%,54% | 2e-92 | 67%,54% | 2e-92 | 86%,68% | 1e-138 | 86%,68% | 1e-138 | 86%,68% | 1e-138 |

Table 4.2: Summary table for performance evaluation on synthetic data. For each method we report the sensitivity, specificity, and statistical significant of the set of sequences predicted to contain the motif. These were compared to the known planted sequences. The left hand side of the table is for datasets created using a PSSM model (no dependencies), while the right hand side for datasets created using a tree network. Each column reports a different setting of training data parameters: Number of true positive sequences (TP) vs. false positive ones (FP) in the training data. The line with method True reports the performance of the model from which we sampled the binding sites.

### 4.4.2 Prediction Accuracy

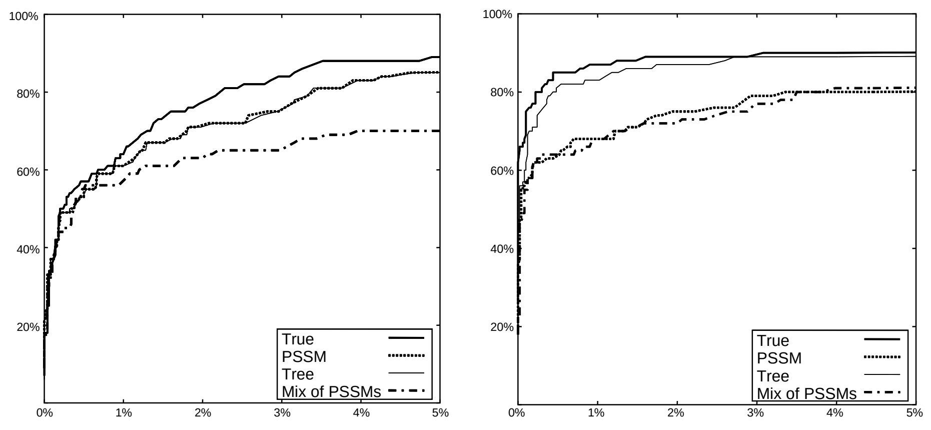
#### Synthetic Data

In Section **??** we explained how genome-wide Chromatin Immunoprecipitation (ChIP) location measurements can be used as a proxy to biological "truth" of transcription factors target genes and how this in turn can be used to evaluate the accuracy of algorithms handling the motif finding problem with "unaligned sequences". Although this serves as an objective test on real data, ChIP experiments and genomic sequence data introduces a lot of noise. We therefore wanted to first gauge the performance of our algorithm in a more "sterile" environment, using *Synthetic* data.

For this task,we built several datasets, each consisting of both "positive" promoters, (*i.e.*, sequences in which we planted binding site motifs), and "negative" ones. To simulate the underlying biological problem as accurately as possible, the motifs themselves were sampled from models trained on known binding sites of the Human LUN transcription factor from the TRANSFAC database (`V$LUN1_01`). In each setting, we created two parallel sets, one sampled from a tree network that contains position dependencies, and the other from a PSSM model. The promoter sequences were sampled from a 3-order Markov model background distribution, trained on Human promoter regions. To simulate noise, we have contaminated our datasets with another group of "false positive" promoters, where no motif was planted.

We set the observation model such that all "positive" sequences had $P(r^t \mid O) = 0.99$, while the "negative" ones had $P(r^t \mid O) = 0.01$. We tested our methods on a variety of settings, changing both the promoters lengths (from 250 to 500bp), and the composition of the "positive" promoters: from 100 true positives without false ones, down to 25 true positives with 75 false ones. After applying our methods to the synthetic training data, we tested them on unseen test data that was similarly generated. To measure performance, we used exactly the same testing procedures used with real ChIP data, as described in Section **??**.

The results on one dataset are shown in Figure **??**, as ROC curves where we view the *false positive rate* ($x$-axis) and the *true positive rate* ($y$-axis) as a function of the $p$-value threshold $t$,



(a) PSSM (b) Tree

Figure 4.9: ROC evaluation of performance with synthetic data. The ROC curves show ability to identify binding sites in synthetic test data. (a) Binding sites generated from a PSSM model. (b) Binding sites generated from a tree model. The $x$-axis shows the *false positive rate*, FP/(FP+TN), the $y$-axis shows the *true positive rate*, TP/(TP+FN). Each curve shows the performance for one model: True– the model that generated the data; PSSM– the learned PSSM model; Tree – the learned tree model; Mix of PSSMs– the learned mixture of PSSMs. In both graphs, the training data consisting of 100 true positive sequences of length 500bp.

used to make the target/non-target calls. It is evident that all methods perform similarly on the data generated from a PSSM, and are comparable to the *true* model that generated the data. The data generated from a tree network, shows a difference between the performances of the learned tree network to both the PSSM and the mixture of PSSMs models that are incapable of modeling the underlying dependencies and therefore perform worse. The learned tree network closely tracks the performance of the true model.

Table **??** summarize the results on all synthetic datasets comparing sensitivity and specificity for $p$-value threshold $t = 0.01$ and for the best hyper geometric $p$-value score defined in Eq. (**??**). As we can see, when the data does not contain false positives sequences, all models perform roughly equivalently on data generated from PSSM. On the tree generated data, we see that the PSSM does the worse, the mixture of PSSMs is slightly better, and finally the tree network is roughly equivalent to the true model. As we increase the noise ratio, the problem becomes harder. The PSSM model and to a large extent the tree model are fairly robust, even in high degrees of noise. On the other hand, the mixture models are more susceptible to noise, and their performance decays. When learning from shorter sequences of length 250bp, the same qualitative conclusions reappear.

### ChIP Data

To evaluate our methods' accuracy on large scale real life data, genome-wide Chromatin Immunoprecipitation (ChIP) location measurements served as the main tool. The ChIP dataset we used

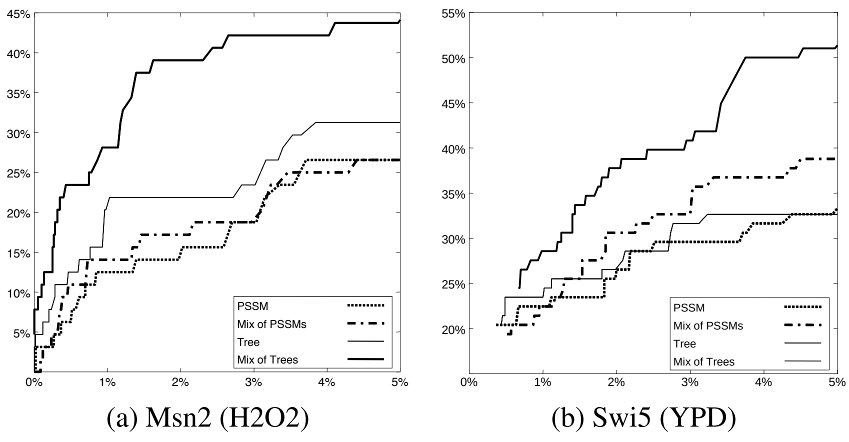
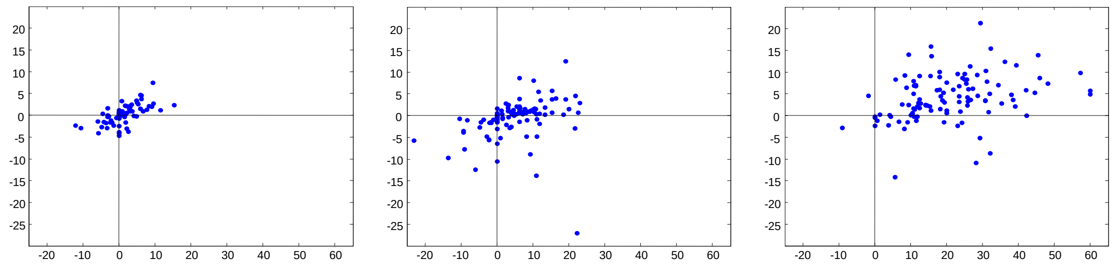


Figure 4.10: Example of ROC curves for two motifs learned and tested using ChIP location data by **?**.

involved 106 yeast transcription factors in 146 experiments (**???**). We focused on 109 experiments for which there where at least 10 genes with localization $p$-value $\leq 0.001$ and at least 50 genes with $p$-value $\leq 0.01$.

Using the evaluation procedure described in Section **??**, We got ROC curves for each motif model on each ChIP dataset. Figures **??**(a) and (b) show two of these: Msn2 (H2O2) and Swi5 (YPD). As we can see in these two examples, models that capture dependencies are clearly superior to the PSSM model. The differences in performances are due to the increased expressiveness of the richer models. Clearly, it is hard to quantitate comparison of more then 100 ROC curves. To get a more general quantitative measure of performance over all datasets, we set a threshold $t = 0.01$ on the $p$-value used to make target gene calls, and compared the sensitivity/specificity measures for this value (see Section **??** for details). Recall that in practice, $t$ allows us to controls the amount of false positives we are willing to tolerate in genome wide scans, and therefore has a practical implication. A summary of these results appears in Figure **??**(a)–(c) that show the *differences* in sensitivity and specificity between each of our methods and PSSM models. Points in the upper right quadrant represent experiments where the richer model performed better in terms of both sensitivity and specificity, points in the bottom left quadrant are ones where the PSSM model performed better, while points in the other two quadrants are ones where the two methods achieve different tradeoffs between sensitivity and specificity. As we can see, in most experiments all three methods perform better than PSSM models. The tree network shows a modest improvement in 32 experiments, and a slight decrease in 15. The mixture of PSSMs does somewhat better (55 better, 19 worse), and the mixture of trees is significantly better in virtually all experiments (87 better, 2 worse).

To evaluate the learned models with respect to what is known about the underlying biological context, we compared the PSSMs we learned with known yeast transcription factor binding sites



(a) Tree (b) Mixture of PSSMs (c) Mixture of trees

Figure 4.11: Results on ChIP localization data of **?**. Shown are the relative difference in sensitivity and specificity using different models, all compared to PSSM motif model. Each point represents results from one localization experiment date set where the difference in sensitivity is the $x$-axis value and the difference in specificity is the $y$-axis value.

from TRANSFAC (**?**) In 23 experiments we found a TRANSFAC PSSM for the tested transcription factor. Out of these, in 15 experiments the PSSM we learn matches the known one.

### 4.4.3 Correlation to Gene Expression Profiles

Another way to evaluate target predictions is by testing the correlation to gene expression measurements. The idea is that a motif that better predicts target genes defines a targets group that has a distinct expression profile in the relevant experiments. To quantitate this, we used the Kolmagorov Smirnov $p$-value described in Section **??**. As we showed there low KS $p$-value means there is high confidence that the expression distributions of target and non-targets genes are not the same.

The dataset we used included 53 yeast related transcription factors for which we identified a motif correlating reasonably with ChIP experiments (we choose ones with sensitivity or specificity ratios of at least 40%). We used a compendium of over 1000 gene expression experiments (**?**) done in various stress conditions, cell cycle phases, and with knockout mutants. As a baseline we measured the KS $p$-value we get using target predictions based on ChIP results. This gave us a good reference since these experiments measure DNA binding directly.

An example of the kind of correlations we get are given in Figure **??** and Figure **??**. In both cases we have time series experiments (hence several points along the $x$-axis), were there are biological reasons to expect a distinct expression profile for targets of the transcription factor involved. In both cases we see the dependency models perform much better then the PSSM model giving lower KS $p$-values ($y$ axis). Note that in some cases the KS $p$-value based on the motif model with interdependent positions is even better then the one achieved by using the ChIP target calls.

When we wanted to perform this analysis for all the 53 transcription factors learned from ChIP experiments and the compendium of more then thousand gene expression experiments we needed

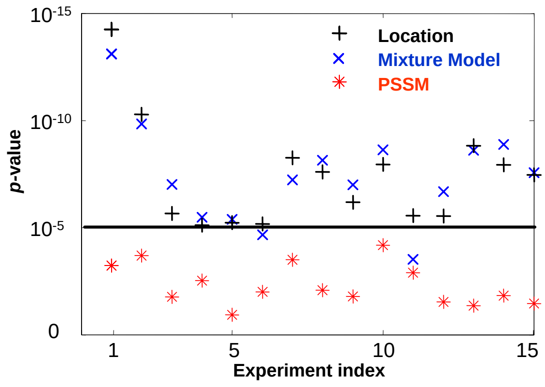


Figure 4.12: Correlation between target prediction for the heat shock factor HSF1 and 15 gene expression measurements after heat shock treatment from the compendium of **?**. The $y$ axis is the computed KS $p$-value for the expression profiles of the target/non-target groups. In black are results from using targets identified by the ChIP experiments of **?**. In blue and red results from using target prediction based on motif binding sites identifications. The blue crosses where obtained using the mixture of trees motif, red by using the matching PSSM motif model. The vertical line corresponds to a $10^{-5}$ threshold over $p$-values for determining significance.

some base line to compare to. Note that in most cases we simply do not know whether we should expect a low $p$-value since we have little knowledge about the underlying biology. Moreover, we need a systematic way to quantitate our results. Since the ChIP experiments directly test for target genes, we decided to use the KS $p$-value obtained by using ChIP target identifications as a reference. This means that for each original ChIP experiment we had target calls based on the original experimental results and on binding site identification using motif models learned by our algorithm. We defined *low* correlation confidence between gene expression and target calls when the KS $p$-value was above $10^{-3}$. High correlation confidence was defined as below $10^{-5}$ and we ignored ambiguous results with KS $p$-value $10^{-5} \leq p \leq 10{-}3$.

The results of this analysis are are summarized in Figure **??**. The scatter plot has a point for each combination of (ChIP,Expression) experiments. The $x$ value of each point represents the correlation between expression and target calls based on ChIP results while the $y$ is the correlation of the expression to target calls based on motif's binding site predictions. The red dots are for using the PSSM motif and the blue ones are for using mixture of tree motif models. Since the data itself is very noisy by nature, the scatter plots are very blurred. However, there are several phenomenon that comes out of it when we take a closer look. The region marked "A" contain points with coordinates $x, y \geq 5$ above the $45^o$ diagonal line (*i.e.*, $y > x$). This means this region contains (ChIP,Expression) combinations were target predictions based on ChIP results proved to have high correlation to observed expression ($x$ coordinate) but the motif models were able to *improve* this correlation ($y > x$). We see that we have 181 such cases when we use the mixture of trees model, compared to only 136 such cases when using PSSMs. The region marked "B" contains (ChIP,Expression) combinations for which ChIP results indicated high correlation between target calls and expression ($x > 5$), yet

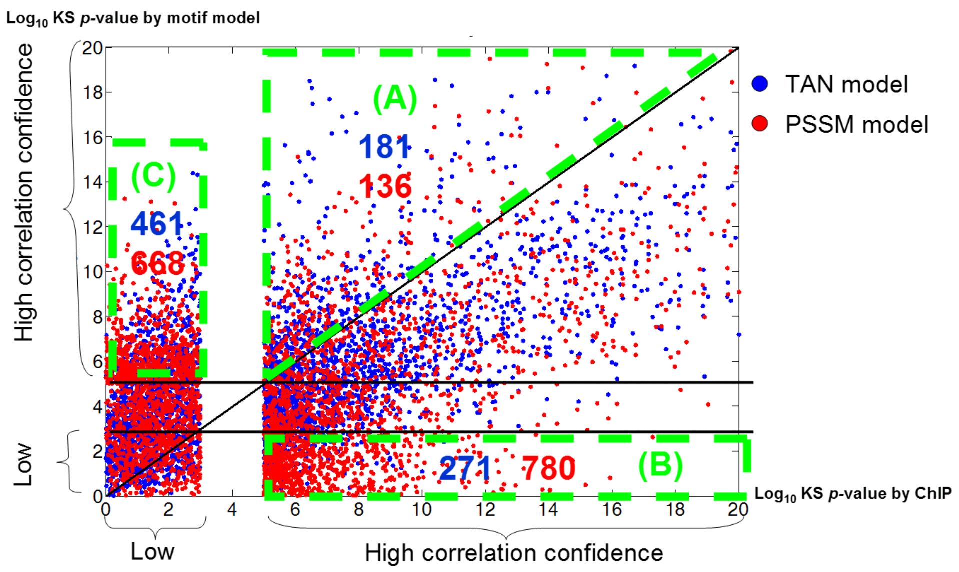


Figure 4.13: Correlation between target prediction and gene expression. Each point is a the correlation between a motif learned from a ChIP experiment and the expression profile in some experiment, measured using KS $p$-value. The $x$-axis is the correlation achieved by target calls based on the actual ChIP results. the $y$-axis is the correlation achieved by target calls based on the motif learned using the matching ChIP experiment. The red dots are when using PSSM motif models and the blue ones are for mixture of trees motifs.

the motif models failed to find a strong correlation ($y < 3$). We see much more cases of this kind for PSSM models (780) compared to mixture of trees models (271). Finally, in the region marked "C" we have (ChIP,Expression) combinations were ChIP experiments indicated no correlation between target prediction and expression ($x < 3$), but targets based on motif finding were found to suggest strong correlation to expression ($y > 5$). Again, we see that in this area to the PSSM based motifs tend to have much more cases of these kind (668), compared to mixture of trees based motifs (461).

To summarize this part we conclude that although measuring correlation between target prediction to gene expression is hard to quantitate and very noisy, we were able to show, using large scale analysis techniques, that mixture of trees based motif models gave target predictions that correlated better to observed expression measurements.

## 4.5 Discussion

In this chapter we performed an extensive evaluation of the motif finding algorithms presented in previous chapters. We used high throughput data, ChIP assays and expression measurements, to evaluate the effect of our discriminative motif finding approach and the added value in modeling Interdependent positions in binding site motifs. Although ChIP based target predictions are inherently noisy (Section **??**) and so are gene expression measurements and their relation to transcription factor's targets, we were able to show the overall distinct advantage of our approach. The *SeedSearcher* discriminative crude motif, used as a starting point for motif refinement by both *LearnPSSM* and *LearnMotif*, proved to be a major contributer to their success. The difference between these two algorithms when optimizing PSSM motifs was shown to be minor relative to the effect using more complex motif models has. The flexible framework we developed for learning motif models with interdependent positions proved to have a significant effect on improving binding site prediction accuracy.

The experimental results we got in these evaluation tests and in numerous research collaborations during the past years also indicate directions for further improvements of our algorithms. Probably the most burning issue is to be able to identify how many motif signals are actually present in the data, relate them to known motifs (when available), and be able to learn them simultaneously. For example, when performing *de novo* motif finding based on ChIP assays for the DIG1 transcription factor, we can identify sequence motif for two different transcription factors, namely TEC1 and STE12, that cooperatively bind with DIG1 to target genes in the filamentation pathway (**?**). This analysis is currently done as a post processing stage, manually, and each motif is optimized separately.

When it comes to understanding regulatory mechanisms, just handling the motif finding task by it own is an inherently limited approach. As we discussed in previous chapters, there is a limit for what we can hope to extract from plain promoter sequence data and general gene group definitions. There is maybe no better way to make this point clear then by simply reviewing the high false positive ratios in target predictions when using even the more accurate motif models such as mixture of trees. The key to improve on this and have a better understanding of what is a real binding site or when a transcription factor is active in regulating a gene, lies in our ability to combine more sources of data, genomic as well as genetic, in a synergistic manner. Given more evidence from various sources we can identify more accurately for example, that under nutrient limitation condition the yeast cell activates STE12, TEC1 and DIG1 to contorl a specific set of genes as part of the filamentation pathway. Model that incorporate the motif finding tools we developed here for exactly this kind of tasks is the subject of our next chapter.

# Chapter 5

# Learning Regulatory Mechanisms

In this thesis we aim to improve our understanding of transcriptional regulatory mechanisms in the living cell. In many senses, the wiring of these regulatory mechanisms DNA code is given by the transcription factors binding sites. This led us to devote the first chapters for improving our ability to represent binding sites and predict them. We have shown how we can incorporate various sources of genomic data such as ChIP assays to improve our predictions. The basic way in which these sources were used can be termed the *two stages approach*: At the *first* stage gene expression or ChIP measurements are used to define group of genes suspected to share a regulatory program. Then, in the *second* stage these groups are used to guide the search for regulatory elements related to these groups. In some cases we can guide the search further by defining a *fixed* prior over regulation events, binding site location etc.

We would like to extend our models at this point and integrate all sources of data together in a unified probabilistic framework that combines binding sites identification with learning the actual regulatory programs. This includes learning the combinations of regulators that make the regulatory programs, the experimental conditions in which they operate, and their target genes. Moving to a unified probabilistic model is naturally appealing, given the data's noisy and partial nature. The basic idea is that by combining the sources of data together we could clear out some of the noise and extract the features we are more certain of. For example, the first algorithm we introduce, *NBClust*, characterizes groups of genes that share not just a similar expression profile (as usually done in the first stage of the two stages approach described above) but also have a common set of regulators in their promoters. Each such gene group is characterized by only a *subset* of the possible regulators. This way, spurious identifications of other regulators sites (which often occur) can be ignored as they do not match the patterns found in genes with both similar expression profile and similar sequence motifs.

There are of course many ways in which one can use probabilistic graphical models to infer regulators programs. They may differ in their input, the structure imposed, model assumptions, and the target function used. One common characteristic of all models we develop here is they involve genomic sequence as input. This is a direct consequence of our primary interest in the DNA coding of the regulatory programs. However, the models we present here reflect a wide range of modeling approaches, affecting in turn the implicit modeling assumptions, target functions for learning and optimization techniques. Even the formal language used to describe the models vary.

We start with the more simple *NBClust* algorithm. The model learned by this algorithm can be categorized, by the Bayesian Network language, as having a *Naive Bayes* structure. We can think of this model's structure as grouping or clustering the genes into sub groups that share similar characteristics. The attributes that characterize each gene groups are expression profiles in some (not necessarily all) experiment and certain transcription factors binding sites found in their promoters. In *NBClust* the genomic sequence is not modeled directly. Instead we assumes biding sites existence in promoters is already known and given as input. This model reflects a *generative* approach towards regulatory programs modeling. The learning process in this case tries to learn a model (*i.e.*, the gene groups and their characteristic profiles) that best explains observed data. One of the benefits of *NBClust*, besides its relative simplicity, is its applicable to a wide range of problems, not confined to regulatory programs or even the biological domain. We demonstrate this as part of the algorithms evaluation.

In the second part of this chapter we describe the *DiscRegProg* algorithm that represent a very different approach for inferring regulatory mechanisms. *DiscRegProg* uses a more *discriminative* approach for learning regulatory programs using the modeling language of PRMs (**P**robabilistic **R**elational **M**odels) (Section **??**). The model learned combines sequence, expression and location (ChIP) measurements into a unified, probabilistic framework. Unlike *NBClust*, it involves *ab initio* motif learning, facilitating the *LearnPSSM* algorithm we developed in Section **??**. As we shell see, the dependency structure between the basic entities in the model (expression,sequence, regulation) is very different from that of *NBClust* and as a consequence so are the target function and optimization techniques used for learning. We evaluate each of the algorithms in turn and finish with a discussion in Section **??**.

## 5.1 The *CSI-NBClust* algorithm

Our aim in this algorithm is to characterize groups of genes based both on their expression profiles as well as the presence of putative binding sites within their promoter regions. The basic biological hypothesis suggests that genes within a functional group will be similar with respect to both types of attributes. The algorithm treats expression level measurements and information on binding sites in a symmetric fashion, and cluster genes based on both types of data. We note this algorithm does not involve discovery of *new* binding sites, but rather tries to learn which of the identified ones are relevant for different gene groups.

*CSI-NBClust* probabilistic model includes random variables of two types. Random variables of the first type describe the expression level of the gene in a specific experiment. Each experiment is denoted by a different random variable whose value is the expression level of a gene in that particular experiment. Random variables of the second type describe occurrences of putative binding sites in the promoter region of the gene. Each binding site is denoted by a random variable, whose value is the number of times the binding site was detected in the gene's promoter region.

The result of the *CSI-NBClust* algorithm is not just grouping of genes. The probabilistic model learned provides insight on the regulation of genes within each cluster. The key features of this approach are: (1) automatic detection of the number of clusters; (2) automatic detection of random variables that are irrelevant to the clustering; (3) robust clustering in the presence of many such random variables, (4) context-depended representation that describes which clusters each attribute depends on. This allows us to discover the attributes (random variables) that characterize each cluster and distinguish it from the rest.

In Section **??** we introduce the class of probabilistic models that we call *Context-Specific Clustering* models. In Section **??** we discuss how to *score* such models based on data. In Section **??** we describe our approach for finding a high-scoring clustering model. In Section **??** we evaluate the learning procedure on synthetic and real-life data.

### 5.1.1 Context-Specific Clustering: the CSI Naive Bayes Model

In this section we describe the class of probabilistic models we learn from data. We develop the models in a sequence of steps starting from a fairly well known model for Bayesian clustering, and refine the representation to explicitly capture the structures we want to learn. We stress that at this stage we focus on what can be represented by the class of models, and we examine how to learn them in subsequent sections.

#### Naive Bayesian Clustering

Let $X_1, \ldots, X_N$ be random variables. In our main application, these random variables denote the attribute of a particular gene: the expression level of this gene in each of the experiments, and the numbers of occurrences of each binding sites in the promoter region. Suppose we receive a dataset $D$ that consists of $M$ joint instances of the random variables. The $m$'th instance is a joint assignment $x_1[m], \ldots, x_N[m]$ to $X_1, \ldots, X_N$. In our application, instances correspond to genes: each gene is described by the values of the random variables.

In modeling such data we assume that there is an underlying joint distribution $P(X_1, \ldots, X_N)$

from which the training instances were sampled. The *estimation* task is to approximate this joint distribution based on the data set $D$. Such an estimate can help us understand the interactions between the variables. A typical approach for estimating such a joint distribution is to define a *probabilistic model* that defines a set of distributions that can be described in a *parametric* form, and then find the particular parameters for the model that "best fit" the data in some sense.

A simple model that is often used in data analysis is the *naive Bayes* model. In this model we assume that there is an unobserved random variable $C$ that takes values $1, \ldots, K$, and describes which "cluster" the example belongs to. We then assume that if we know the value of $C$, all the observed variables become independent of each another. That is, the form of the distribution is:

$$P(X_1, \ldots, X_N) = \sum_k P(C = k)P(X_1 \mid C = k) \cdots P(X_N \mid C = k) \tag{5.1}$$

In other words, we estimate a *mixture* of product distributions.

The naive Bayes model is illustrated in Figure **??** for the case were we cluster genes. Here each gene is a sample that provides an assignment to $X_1, \ldots, X_N$ that are the expression measurements in the experiments $E_1, \ldots, E_M$ ($E_i \in R$) and the transcription factor binding sites within the genes promoter $R_1, \ldots, R_T$ ($R_j \in N$). We can think of learning such a model as trying to identify *modules*, each such module has a characteristic expression profile and a characteristic profile of the binding sites involved in controlling it. Learning the model means learning the number of modules, each module's characteristics and the gene assignments to each module. However, one must also bare in mind that such models are not necessarily a representation of exact biological structure but rather a mathematical model that can provide insights into the biological connections between variables. For example, this model makes several *conditional* independence assumptions. First, we assume that *given* the model, the genes are independent. That is, after we know the model, observing the expression levels of a single gene does not help predict better the expression levels of another gene. Similarly, we assume that expression level of the same gene in different condition are independent *given* the cluster the gene belongs to. Although we do not expect this to be exactly the case for all experiments we model (*e.g.*, a set of time series experiments), this assumption simply states that the cluster captures the "first order" description of the gene's behavior, and we treat (in the model) all other fluctuations as noise that is independent in each measurement. [1]

We found the naive Bayes model to be attractive for several reasons. First, from estimation point of view we need to estimate relatively few parameters: the mixture coefficients $P(C = k)$, and the parameters of the conditional distributions $P(X_i \mid C = k)$. Second, the estimated model can be

[1] We attempt to be precise and explicit about the independence assumptions we make. However, we note that most clustering approaches we know of treat (explicitly or implicitly) genes as being independent of each other, and quite often also treat different measurement of the same gene as independent observations of the cluster.

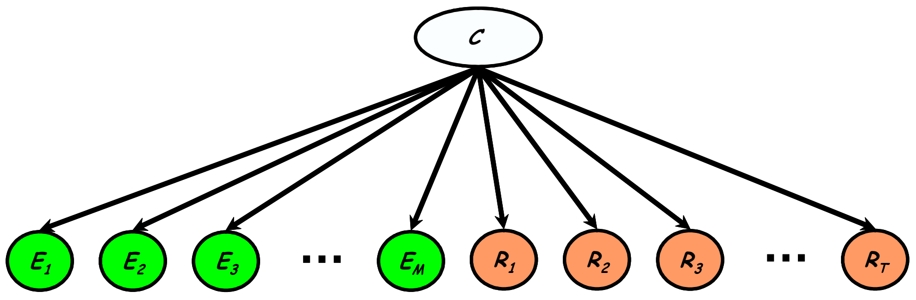


Figure 5.1: Illustration of the Naive Bayes model for gene clustering based on expression measurements and transcription factor biding sites. The hidden cluster/module variable ($C$) is colored white.

interpreted as modeling the data by $K$ clusters (one for each value $k = 1, \ldots, K$), such that the distribution of different variables within each cluster are independent. Thus, dependencies between the observed variables are represented by the cluster variable. Finally, this model allows us to use fairly efficient learning algorithms, such as *expectation maximization* (EM) (**?**).

The distribution form in Eq. (**??**) and its matching representation in Figure **??** describe the global *structure* of the naive Bayesian distribution. In addition, we also have to specify how to represent the conditional distributions. For this purpose we use *parametric* families. There are several of families of conditional distributions we can use for modeling $P(X_i \mid C = k)$. In this work, we focus on two such families.

If $X_i$ is a discrete variable that takes a finite number of values (*e.g.*, a variable that denotes number of binding sites in a promoter region), we represent the conditional probability as a *multinomial* distribution

$$P(X_i \mid C = k) \sim \text{Multinomial}(\{\theta_{x_i|k} : x_i \in \mathit{Val}(X_i)\}$$

For each value $x_i$ of $X_i$ we have a parameter $\theta_{x_i|k}$ that denotes the probability that $X_i = x_i$ when $C = k$. [2] These parameters must be non-negative, and satisfy $\sum_{x_i} \theta_{x_i|k} = 1$, for each $k$.

If $X_i$ is a continuous variable (*e.g.*, a variable that denotes the expression level of a gene in a particular experiment), we use a *Gaussian* distribution

$$P(X_i \mid C = k) \sim N(\mu_{X_i|k}, \sigma^2_{X_i|k})$$

such that

$$P(x_i \mid C = k) = \frac{1}{\sqrt{2\pi}\sigma_{X_i|k}} \exp\left\{-\frac{(x_i - \mu_{X_i|k})^2}{2\sigma^2_{X_i|k}}\right\}.$$

We use the Gaussian model in this situation for two reasons. First, as usual, the Gaussian

[2]We deviate here a little from the general notation we defined for Bayesian network (Chapter **??**). $\theta_{x_i|k}$ is actually $\theta_{x_i|\mathbf{pa}_i}$ since in the Bayesian network we define $\mathbf{Pa}_i = C$. We prefer the short hand notation defined for this specific case to make the following equations more clear.

distribution is one of the simplest continuous density models and allow efficient estimation. Second, when we use as observations the logarithm of the expression level (or logarithms of ratios of expression between a sample and a common control sample), gene expression has roughly noise characteristics. We note however, that most of the developments in this work can be achieved with more detailed (and realistic) noise models for gene expression.

Once we have estimated the conditional probabilities, we can compute the probability of an example belonging to a cluster:

$$P(C = k \mid x_1, \ldots, x_N) \propto P(C = k)P(x_1 \mid C = k) \cdots P(x_N \mid C = k)$$

If the clusters are well-separated, then this conditional probability will assign each example to one cluster with high probability. However, it is possible that clusters overlap, and some examples are assigned to several clusters. If we compare the probability of two clusters, then

$$\log \frac{P(C = k \mid x_1, \ldots, x_N)}{P(C = k' \mid x_1, \ldots, x_N)} = \log \frac{P(C = k)}{P(C = k')} + \sum_i \log \frac{P(x_i \mid C = k)}{P(x_i \mid C = k')}$$

Thus, we can view the decision boundary between any two clusters as the sum of terms that represent the contribution of each attribute to this decision. The ratio $P(x_i \mid C = k)/P(x_i \mid C = k')$ is the relative *support* that $x_i$ gives to $k$ versus $k'$.

### Selective Naive Bayesian Models

The naive Bayes model gives all variables equal status. This is a potential source of problems for two reasons. First, some variables should be considered as "noise" since they have no real interactions with the other variables. Suppose that $X_1$ is independent from rest of the variables. By learning $K$ conditional probability models $P(X_1 \mid C = 1), \ldots, P(X_1 \mid C = K)$, we are increasing the variability of the estimated model. Second, since we are dealing with a relatively small number of training examples, if we fail to recognize that $X_1$ is independent of the rest, the observations of $X_1$ can bias our choice of clusters. Thus, a combination of many irrelevant variables might lead us to overlook the relevant ones. As a consequence, the learned model discriminate clusters by the values of the irrelevant variables. Such clusters suffer from high variability (because of their "noisy" character).

If we know that $X_1$ is independent from the rest, we can use the fact that $P(X_1 \mid C) = P(X_1)$ and rewrite the model in a simpler form:

$$P(X_1, \ldots, X_N) = P(X_1) \sum_k P(C = k)P(X_2 \mid C = k) \cdots P(X_N \mid C = k).$$

This representation of the joint probability requires less parameters and thus the estimation of these parameters is more robust. More importantly, the structure of this model explicitly captures the fact that $X_1$ is independent of the other variables—its distribution does not depend on the cluster variable. Note that in this model, as expected, the value of $X_1$ does not impact the probability of the class $C$.

In our biological domain, we expect to see many variables that are independent (or almost independent) of the classification. For example, not all binding sites of transcription factors play an active role in the conditions in which expression levels were measured. Another example, is a putative binding site (suggested by some search method or other) that does not correspond to a biological function. Thus, learning that these sites are independent of the measured expression levels is an important aspect of the data analysis process.

Based on this discussion, we want to consider models where several of the variables do not depend on the hidden class. Formally, we can describe these dependencies by specifying a set $G \subseteq \{X_1, \ldots, X_N\}$ that represents the set of variables that depend on the cluster variable $C$. The joint distribution then takes the form of

$$P(X_1, \ldots, X_N \mid G) = \left( \prod_{X_i \notin G} P(X_i) \right) \sum_k \left( P(C = k) \prod_{X_i \in G} P(X_i \mid C = k) \right)$$

We note that this class of models is essentially a special subclass of *Bayesian networks* (**?**). Similar models were considered for a somewhat different application in *supervised* learning by **?**

We note again, that when we compare the posterior probability of two clusters, as in Eq. (**??**), we only need to consider variables that are not independent of $C$. That is,

$$\log \frac{P(C = k \mid x_1, \ldots, x_N)}{P(C = k' \mid x_1, \ldots, x_N)} = \log \frac{P(C = k)}{P(C = k')} + \sum_{X_i \in G} \log \frac{P(x_i \mid C = k)}{P(x_i \mid C = k')}.$$

This formally demonstrates the intuition that variables outside of $G$ do not influence the choice of clusters.

### CSI: Context-Specific Independence

Suppose that a certain binding site, whose presence is denoted by the variable $X_1$, is regulating genes in two functional categories. We would then expect this site to be present with high probability in promoter regions of genes in these two categories, and to have low probability of appearing in the promoter region of all other genes. Since $X_1$ is relevant to the expression level of (some) genes, it is not independent of the other variables, and so we would prefer models where $X_1 \in G$. In such a model, we need to specify $P(X_1 \mid C = k)$ for $k = 1, \ldots, K$. That is, for each functional category,

| | $X = 0$ | $X = 1$ | $X = 2$ | $X = 3$ |
|---|---|---|---|---|
| $C = 1$ | 0.1 | 0.1 | 0.5 | 0.3 |
| $C = 2$ | 0.1 | 0.1 | 0.2 | 0.6 |
| $C = 3$ | 0.7 | 0.2 | 0.05 | 0.05 |
| $C = 4$ | 0.7 | 0.2 | 0.05 | 0.05 |
| $C = 5$ | 0.7 | 0.2 | 0.05 | 0.05 |
| $C = 6$ | 0.7 | 0.2 | 0.05 | 0.05 |

(a) explicit table representation

| | $X = 0$ | $X = 1$ | $X = 2$ | $X = 3$ |
|---|---|---|---|---|
| $C = 1$ | 0.1 | 0.1 | 0.5 | 0.3 |
| $C = 2$ | 0.1 | 0.1 | 0.2 | 0.6 |
| $C = *$ | 0.7 | 0.2 | 0.05 | 0.05 |

(b) default table

Figure 5.2: Example of two representations of the same conditional distribution $P(X \mid C)$.

we learn a different probability distribution over $X_1$. However, since $X_1$ is relevant only for two classes, say 1 and 2, this introduces unnecessary complexity: once we know that $C$ is not one of the two "relevant" function classes (*i.e.*, $C > 2$), we can predict $P(X_1 \mid C)$ using a single distribution.

To capture such distinctions, we need to introduce a language that refines the ideas of selective naive Bayesian models. More precisely, we want to describe additional structure within the conditional distribution $P(X_1 \mid C)$. The intuition here is that we need to specify *context-specific independencies* (CSI): once we know that $C \notin \{1, 2\}$, then $X_1$ is independent of $C$. This issue has received much attention in the probabilistic reasoning community (**???**).

Here, we choose a fairly simple representation of CSI that **?** term *default tables*. This representation is as follows. The structure of the distribution $P(X_i \mid C)$ is represented by an object $\mathcal{L}_i = \{k_1, \ldots, k_l\}$ where $k_j \in \{1, \ldots, K\}$. Each $k_j$ represents a case that has an explicit conditional probability. All other cases are treated by a special *default* conditional probability. Formally, the conditional probability has the form:

$$P(X_i \mid C = k) = \begin{cases} P(X_i \mid C = k_j) & k = k_j \in \mathcal{L}_i \\ P(X_i \mid k \notin \mathcal{L}_i) & \text{otherwise} \end{cases}$$

It will be convenient for us to think of $\mathcal{L}_i$ as defining a random variable, which we will denote $L_i$, with $l + 1$ values. This random variable is the characteristic function of $C$, such that $L_i = j$ if $C = k_j \in \mathcal{L}_i$, and $L_i = l + 1$ if $C = k \notin \mathcal{L}_i$. Then, $P(X_i \mid C)$ is replaced by $P(X_i \mid L_i)$. This representation requires $l + 1$ different distributions rather than $K$ different ones. Note that each of these conditional distributions can be multinomial, Gaussian, or any other parametric family we might choose to use.

Returning to our example above, Instead of representing the probability $P(X_1 \mid C)$ as a complete table, as in Figure **??**(a), we can represent it using a more succinct table with the cases 1, 2 and the default $\{3, \ldots, K\}$ as shown in Figure **??**(b). This requires estimating a different probability of $X_1$ in each of the first two clusters, and one probability of $X_1$ in the remaining clusters.

We note that in the extreme case, when $\mathcal{L}_i$ is empty, then we are rendering $X_i$ independent of

$C$. To see this, note that $L_i$ has a single value in this situation, and thus $P(X_i \mid C)$ is the same for all values $C$. Thus, since CSI is a refinement of selective Bayesian models, it suffices to specify the choice $\mathcal{L}_i$ for each variable.

Finally, we consider classifying a gene given a model. As in Eq. (**??**), the decision between two clusters is a sum of terms of the form $P(x_i \mid C = k)/P(x_i \mid C = k')$. Now, if both $k$ and $k'$ fall in the "default" category of $\mathcal{L}_i$, then they map to the same value of $L_i$, and thus define the same conditional probability over $X_i$. In such a situation, the observation $x_i$ does not contribute to the distinction between $k$ and $k'$. On the other hand We will say that $X_i$ *distinguishes* a cluster $k_j$, if $k_j \in \mathcal{L}_i$ indicating a unique conditional distribution for $X_i$ given the cluster $k_j$.

### 5.1.2 Scoring CSI Clustering

We want to *learn* CSI Clustering from data. By learning, we mean selecting the number of clusters $K$, the set of dependent random variables $G$, the corresponding local structures $\mathcal{L}_i$, and in addition, estimating the parameters of the conditional distributions in the model. We reiterate that CSI clustering is a special sub-class of *Bayesian networks* with default tables. Thus, we adopt standard learning approaches for Bayesian networks (Section **??**) and specialize them for this class of models. In particular, we use a Bayesian approach where learning probabilistic models can be viewed as an optimization problem of some scoring function.

In this section we review the scoring functions over different choices of clustering models (including both structure and parameters). In the next section, we consider methods for finding the high-scoring clustering models. That is, we describe computational procedures for searching the vast space of possible structures efficiently.

#### The Bayesian Score

We assume that the set of variables $X_1, \ldots, X_N$ is fixed. We define a CSI Clustering *model* to be a tuple $\mathcal{G} = \langle K, \{\mathcal{L}_i\}\rangle$, where $K$ specifies the number of values of the latent class and $\mathcal{L}_i$ specifies the choice of *local structure* for $X_i$ (Recall that $X_i$ does not depend on $C$ if $\mathcal{L}_i = \emptyset$). We note that the definition of the CSI Clustering *model* corresponds to the *structure component* of the Bayesian network (Definition **??**), along with the cardinality of the hidden cluster variable. This is also the reason why we use the same notation ($\mathcal{G}$) for both. A model $\mathcal{G}$ is *parameterized* by a vector $\theta$ of parameters. These include the mixture parameters $\theta_k = P(C = k)$, and the parameters $\theta_{X_i|l}$ of $P(X_i \mid L_i = l)$.

As input for the learning problem, we are given a dataset $D$ that consists of $M$ samples, the $m$'th sample specifies a joint assignment $x_1[m], \ldots, x_N[m]$ to $X_1, \ldots, X_N$. Recall that in the *Bayesian* approach, we compute the *posterior* probability of a model, given the particular data set $D$:

$$P(\mathcal{G} \mid D) \propto P(D \mid \mathcal{G})P(\mathcal{G})$$

Where $P(\mathcal{G})$ is the *prior* probability of the model $\mathcal{G}$, and $P(D \mid \mathcal{G})$ is the *marginal likelihood* of the data, given the model $\mathcal{G}$. In this work, we use a fairly simple class of priors over models, in which the model prior decomposes into several independent components, as suggested by **?**)

$$P(\mathcal{G}) \propto P(K)P(G)\prod_i P(\mathcal{L}_i)$$

We assume that $P(K) \propto \lambda^K$ is a geometric distribution with parameter $\lambda$ which is fairly close to 1. The prior over $G$ is designed to penalize dependencies. Thus $P(G) \propto \alpha^{|G|}$ for some parameter $\alpha < 1$. (Recall that $G = \{i : \mathcal{L}_i \neq \emptyset\}$.) Finally, the prior distribution over local models is set to $P(\mathcal{L}_i) \propto \frac{1}{K}\binom{K}{|\mathcal{L}_i|}^{-1}$. Thus, we set a uniform prior over the number of cases in $\mathcal{L}_i$, and then put a uniform prior over all local structures with this cardinality. We choose these priors for their mathematical simplicity (which makes some of the computations below easier) and since they slightly favor simpler models.

We now consider the marginal likelihood term. This term evaluates the probability of generating the data set $D$ from the model $\mathcal{G}$. This probability requires averaging over all possible parameterizations of $\mathcal{G}$:

$$P(D \mid \mathcal{G}) = \int P(D \mid \mathcal{G}, \theta)P(\theta \mid \mathcal{G})d\theta \tag{5.2}$$

where $P(\theta \mid \mathcal{G})$ is the *prior* density over the parameters $\theta$, and $P(D \mid \mathcal{G}, \theta)$ is the *likelihood* of the data

$$P(D \mid \mathcal{G}, \theta) = \prod_m \sum_k \left( P(C = k \mid \mathcal{G}, \theta) \prod_i P(x_i[m] \mid l_i(k), \mathcal{G}, \theta) \right) \tag{5.3}$$

where $l_i(k)$ is the value of $L_i$ when $C = k$.

In this work we follow a standard approach to learning graphical models and use *decomposable priors* for a given model parameters $\theta$ that have the form

$$P(\theta \mid \mathcal{G}) = P(\theta_C)\prod_i \prod_{l \in \mathcal{L}_i} P(\theta_{X_i|l})$$

For multinomial $X_i$ and for $C$, we use a *Dirichlet* prior over the parameters, and for normal $X_i$, we use a *normal-gamma* prior (Section **??**).

The usage of the Bayesian method (compared to using the *maximum likelihood* estimators) is extremely important in our setting since it helps the model avoid over-fitting by averaging over all possible parameterizations. It can be shown that in the limit for large sample size (*i.e.*, $M \rightarrow \infty$) the Bayesian score behaves like a penalized maximum likelihood score, where the penalty depends on the complexity of the model (Section **??**).

### Complete Data

We briefly discuss the evaluation of the marginal likelihood in the case where the data is complete. This setting is easier than the setting we need to deal with, however, the developments here are needed for the ones below. In the *complete data* case we assume that we are learning from a data set $D_c$ that contains $M$ samples, each of these specifies values $x_1[m], \ldots, x_N[m], c[m]$ for $X_1, \ldots, X_N$ and $C$ (In this case, we also fix in advance the number of values of $C$). For such data sets, the likelihood term $P(D_c \mid \mathcal{G}, \theta)$ can be decomposed into a product of local terms:

$$P(D_c \mid \mathcal{G}, \theta) = L_{\text{local}}(C, \mathcal{S}_C, \theta_C) \prod_i \prod_{l \in \mathcal{L}_i} L_{\text{local}}(X_i, \mathcal{S}_{X_i|l}, \theta_{X_i|l})$$

where the $L_{\text{local}}$ terms denote the likelihood that depends on each conditional probability distribution and the associated *sufficient statistics* vectors $\mathcal{S}_C$ and $\mathcal{S}_{X_i|l}$. These statistics are cumulative functions over the training samples. These include *counts* of the number of times a certain event occurred, or sum of the values of $X_i$, or $X_i^2$ in the samples where $L_i = l$. (see Section **??** for details). We also note that the above equation is simply a special case of the likelihood decomposition in Bayesian networks according to the local family structure (Eq. (**??**)).

An important property of the sufficient statistics is that once we compute the statistics for the case in which $|\mathcal{L}_i| = |C|$ *i.e.*, we have a separate conditional distribution for each cluster in a node $X_i$, we can easily get statistics for other local structures, as a sum over the relevant statistics for each $l \in \mathcal{L}_i$ :

$$\mathcal{S}_{X_i|l} = \sum_k P(L_i = l \mid C = k) \mathcal{S}_{X_i|c_k}$$

(Note that since $L_i$ is a deterministic function of $C$, $P(L_i = l \mid C = c)$ is either 0 or 1.)

The important consequence of the decomposition of Eq. **??** and the corresponding decomposition of the prior, is that the marginal likelihood term also decomposes (Section **??**).

$$P(D_c \mid \mathcal{G}) = S_{\text{local}}(C, \mathcal{S}_C) \prod_i \prod_{l \in \text{Val}(L_i)} S_{\text{local}}(X_i, \mathcal{S}_{X_i|l}) \tag{5.4}$$

where

$$S_{\text{local}}(X_i, \mathcal{S}_{X_i|l}) = \int L_{\text{local}}(X_i, \mathcal{S}_{X_i|l}, \theta_{X_i|l}) P(\theta_{X_i|l} \mid \mathcal{G}) d\theta_{X_i|l}$$

The decomposition of marginal likelihood suggests that we can easily find the best model in the case of complete data. The intuition is that the observation of $C$ *decouples* the modeling choices for each $X_i$ from the other variables. Formally, we can easily see that changing $L_i$ for $X_i$ changes only the prior associated with that $L_i$ and the marginal likelihood term $\prod_{l \in \text{Val}(L_i)} S_{\text{local}}(X_i, \mathcal{S}_{X_i|l})$. Thus, we can optimize the choice of each $L_i$ separately of the others.

Note that there are $2^K$ possible choices of $\mathcal{L}_i$. For each such choice we compute the sufficient statistics, and evaluate the score of the model. When $K$ is small we can exhaustively evaluate all these choices. In such a situation we are find the optimal model given the data. In most learning scenarios, however, $K$ is large enough to make such enumeration infeasible. Thus, instead, we construct $L_i$ by a greedy procedure (**?**) that at each iteration finds the best $k$ to separate from the default case, until no improvement is made to the score.

To summarize, when we have complete data the problem of learning a CSI clustering model is straightforward: We collect the sufficient statistics $\mathcal{S}_{X_i|c_k}$ for every $X_i$ and $k = 1, \ldots, K$, and then we can efficiently evaluate every possible model. Moreover, we can choose the one with the highest posterior without explicitly enumerating all possible models. Instead, we simply decide what is the best $L_i$ for each $X_i$, independently of the decisions made for the other variables.

### Incomplete Data

We now return to the case of interest to us, where we do not observe the class labels. Such a learning problem is said to have *incomplete data*. In this learning scenario, the evaluation of the marginal likelihood Eq. (**??**) is problematic as we need to summarize over all completions of the missing data. We denote the missing part of the data as $D_H$. In our case, the missing data consist of assignment to clusters for the $M$ samples. Using this notation, we can write Eq. (**??**) as:

$$P(D \mid \mathcal{G}) = \int_{\theta} \sum_{D_H} P(D, D_H \mid \mathcal{G}, \theta) P(\theta \mid \mathcal{G}) d\theta$$

Although $P(D, D_H \mid \mathcal{G}, \theta)$ is a product of local terms, we cannot decompose the marginal likelihood. Moreover, unlike the complete data term, we cannot learn the structure of $P(X_i \mid C)$ independently of learning the structure of other conditional probabilities. Since we do not observe the values of the cluster variables, these choices interact. As a consequence, we cannot compute the marginal likelihood in an analytical form. Instead, we need to resort to approximations (Section **??**). Specifically, we use two such approximations to the logarithm of the marginal likelihood. The first is the *Bayesian Information Criterion* (BIC) approximation:

$$BIC(\mathcal{G}, \theta) = \log P(D \mid \mathcal{G}, \theta) - \frac{1}{2} \log M \dim(\mathcal{G})$$

To evaluate this score, we perform *expected maximization* (EM) iterations to find the MAP parameters (Section **??**). The benefit of this score is that once we find the MAP parameters, it is fairly easy to evaluate. Unfortunately, this score is only asymptotically correct, and can over-penalize models for complexity in practice.

Another possible approximation is the *Cheeseman-Stutz* (CS) score (**?**); see also **?**. This score approximates the marginal likelihood as:

$$\text{CS}(\mathcal{G}, \theta) = \log P(D \mid \mathcal{G}, \theta) - \log P(D_c^* \mid \mathcal{G}, \theta) + \log P(D_c^* \mid \mathcal{G})$$

where $D_c^*$ is a fictitious data set that is represented by a set of sufficient statistics. The computation of $P(D_c^* \mid \mathcal{G}, \hat{\theta})$ and $P(D_c^* \mid \mathcal{G})$ is then performed as though the data is complete. This simply amounts to evaluating Eq. (**??**) and Eq. (**??**) using the sufficient statistics for $D_c^*$.

The choice of $D_c^*$ is such that its sufficient statistics will match the *expected sufficient statistics* given $\mathcal{G}$ and $\theta$. These are defined by averaging over all possible completions $D_c$ of the data

$$E\left[\mathcal{S}_{X_i|c_k} \mid \mathcal{G}, \theta\right] = \sum_{D_c} \mathcal{S}_{X_i|c_k}^{D_c} P(D_c \mid D, \mathcal{G}, \theta) \tag{5.5}$$

where $D_c$ represents a potential *completion* of the data (*i.e.*, assignment of cluster value to each example) and $\mathcal{S}_{X_i|c_k}^{D_c}$ is the sufficient statistics for $X_i$ given $C = k$ evaluated on $D_c$. Using the linearity of expectation, this term can be efficiently computed (**??**). Thus, to compute $D_c^*$, we find the MAP parameters $\theta_{\mathcal{G}}$, and then compute the expected sufficient statistics given $\mathcal{G}, \theta_{\mathcal{G}}$. We then use these within Eq. (**??**) and Eq. (**??**) as the sufficient statistics of the fictional data set $D_c^*$.

### 5.1.3 Learning CSI Clustering

#### Structural EM

Once we set our prior probabilities, and decide on the type of approximation we use (either BIC or CS), we implicitly induce a score over all possible models. Our goal is to identify the model $\mathcal{G}$ that attains the highest score. Unfortunately, for a fixed $K$, there are $O(2^{NK})$ choices of models with $K$ clusters and $N$ variables, therefore we cannot exhaustively evaluate the score on all models.

The typical way to handle this difficulty is to resort to heuristic search procedures. *Local* search procedures traverse the space of models by performing local changes (*e.g.*, changing one of the $\mathcal{L}_i$ by adding or removing a case in the default table). The main computational cost of such a search is evaluating candidate models. Remember that since we have incomplete data, we cannot directly score a candidate models. Instead, for each candidate model we want to score, we perform another search in the parameter space (using techniques such as EM) to find the MAP parameters and then use these parameters for computing the score. Thus, the search procedure spends non-negligible computation per candidate. This severely limits the set of candidates that it can explore.

To avoid such expensive evaluations of candidates, we use the framework of *Bayesian structural EM* (Section **??**). In this framework, we use our current candidate to "complete" the missing values (*i.e.*, cluster assignments). We then perform structure learning as though we have complete data, searching (efficiently) for a better model. This results in a new "best" model (with its optimized parameters). This new model, forms the basis for the next iteration, and so on. This procedure has the benefit that structure selection is done in a situation that resembles complete data. In addition, each iteration can find a model that is quite different from the model at the beginning of the iteration. In this sense, the local moves of standard search procedure are replaced by global moves. Finally, the procedure is proven to improve the structure in each iteration.

More specifically, the Structural EM procedure consists of repeated iterations. We initialize the process with a model $\mathcal{G}^0, \theta^0$. We discuss below the choice of this starting point. Then at the $\ell + 1$'th iteration we start with the pair $\mathcal{G}^\ell, \theta^\ell$ of the previous iteration and construct a new pair $\mathcal{G}^{\ell+1}, \theta^{\ell+1}$. This iteration consists of three steps.

- **E-Step:** Compute expected sufficient statistics

$$\tilde{\mathcal{S}}^{\ell}_{X_i|c_j} = E\left[\mathcal{S}_{X_i|c_k} \mid \mathcal{G}^\ell, \theta^\ell\right]$$

  for each $i = 1, \ldots, N$ and each $k = 1, \ldots, K$ using Eq. (**??**).

- **M-Step:** Learn a model $\mathcal{G}^{\ell+1}$ and parameters $\theta^{\ell+1}$ using these expected sufficient statistics, as though they were observed in a complete data set. For each $X_i$ choose the scoring CSI model $\mathcal{L}_i$ that maximizes the score with respect to the sufficient statistics. This is done independently for each of the variables.

- **Post processing-Step:** Maximize the parameters for $\mathcal{G}^{\ell+1}$ by running parametric EM. This optimization is initialized by the MAP parameters given the expected sufficient statistics.

These iterations are reminiscent of the standard EM algorithm. The main difference is that in the standard approach the M-Step involves re estimating parameters, while in Structural EM we also re-learn the structure. More precisely, Structural EM enables us to evaluate each possible new $\mathcal{L}_i$ based on the sufficient statistics computed with the current $\mathcal{L}_i$ instead of doing an expensive EM procedure for each such candidate.

In applying this procedure, we can use different scores in choosing models at the M-Step. This depends on the approximation we set out to use on the incomplete data. Above we discussed 2 different scores. The first one is the BIC approximation. In this case, we simply evaluate structures in the M-step using BIC on complete data (the likelihood in this case decomposes, and the complexity penalty remains the same). The second one is the CS approximation. In this case, note that CS applied to complete data is simply the Bayesian score (since $\log P(D \mid \mathcal{G}, \hat{\theta})$ and $\log P(D_c^* \mid \mathcal{G}, \hat{\theta})$ cancel out). Thus, in this case we use the exact Bayesian score with respect to the expected sufficient statistics.

These iterations are guaranteed to improve the score in the following sense. Each iteration finds a candidate that has better score (with respect to the incomplete training data) than the previous one. More precisely, if we use the BIC score (with respect to the expected sufficient statistics) in the M-step, then results of **?** show that the BIC score of $\mathcal{G}^{\ell+1}, \theta^{\ell+1}$ is greater than the BIC score $\mathcal{G}^{\ell}, \theta^{\ell}$, unless the procedure converged in which case the two scores will be equal. Thus, each step improves the score we set out to maximize, and at some point the procedure will reach a (local) maxima.

When we use the CS score, the situation is more complicated. The results of **?** show that each iteration is an approximate version of a procedure that does improve the Bayesian score on the incomplete data. In practice, most iterations do improve the CS score.

We use two different methods for initializing the structural EM procedure. In the first one, we start with the *full* model (where $|\mathcal{L}_i| = |C|$ for every variable $X_i$ node). This model is the most expressive in the class we consider, and thus allows the starting point to capture any type of "trend" in the data. The second initialization method, is by using a random model, where $G$ (*i.e.*, the set of variables dependent on the hidden cluster variable) is chosen at random. In both cases, we apply aggressive parametric optimization to find the initial parameters. This is done by using 100 random starting points for parametric EM, and returning the parameter vector that achieves the highest score.

### Escaping Local Maxima

The structural EM procedure, as described above, can get trapped in "local" maxima. That is, it can reach sub-optimal convergence points. This can be a serious problem, since some of these convergence points are much worse than the optimal model, and thus lead to a poor clustering.

A naive way to avoid this problem is by multiple restarts. However, when the number of local maxima is large, such multiple restarts have limited utility. Instead, we want strategies for escaping local maxima that improve on the solution found by earlier iterations. We implemented two approaches for escaping local maxima.

In the first approach, we apply a directed search once the structural EM procedure converges. More specifically, assume that $\mathcal{G}^{\ell}$ is the convergence point of structural EM. Starting from this model, we apply a local search procedure that attempts to add and remove variables to the model. As explained above, such a procedure is costly since it has to separately evaluate each candidate it proposes. To avoid evaluating all moves from the current model, we apply randomized moves and evaluate each one. Once a candidate with a score higher than that of $\mathcal{G}^\ell$ is found, we restart structural EM from that point. If after a fixed amount of random trials no improvement was found, the procedure terminates the search and returns the best model found so far.

In the second approach, we use annealing-like procedure to introduce randomness at each step of the process. This randomness needs to serve two purposes. On the one hand, a certain amount of randomness will allow the procedure to escape convergence points of structural EM. On the other hand, we want our steps to exploit the sufficient statistic computed in the E-step to choose models that build on information learned in previous iterations.

We achieve this goal by using a variant of Structural EM recently suggested by **?**. The idea is simple: at each iteration of Structural EM, we perform a random *reweighting* of the training samples. More precisely, for each sample $m$, we sample a weight $w_m^\ell$ from a Gamma distribution with mean 1 and variance $\sigma^\ell$, where $\sigma^\ell$ is an additional parameter that controls the "temperature" of the search.

In the modified E-step we compute *weighted* sufficient statistics

$$E\left[\mathcal{S}_{X_i|c_k} \mid W^\ell, \mathcal{G}, \theta\right] = \sum_{D_c} w_m^\ell \mathcal{S}_{X_i|c_k,D_c} P(D_c \mid D, \mathcal{G}, \theta)$$

We then apply the M-Step with respect to these re-weighted expected sufficient statistics. Additionally, we set $\sigma_{\ell+1}$ to be $\gamma \cdot \sigma_\ell$ where $\gamma < 1$ is a decay factor. The search is terminated once $\sigma_\ell$ reaches a certain predetermined threshold. In our experiments, the annealed approach dominated in performance the approach described above.

### 5.1.4 Evaluation

#### Simulation Studies

To evaluate the applicability of our clustering method, we started by performing tests on synthetic data sets. These data sets were sampled from a known clustering model (which determined the number of clusters, which variables depend on which cluster value, and the conditional probabilities). Since we know the model that originated the data, we can measure the performance of our procedure. We examined two aspects. First, how well the procedure recovers the structure of the real model (number of clusters, false positive and false negative edges in the model). Second, how well the procedure recovers the original clustering. That is, how well the model classifies a new sample (gene). The aim of these tests is to understand how the performance of the method depends on various parameters of the learning problem. We will review our techniques for evaluating the learning process results and then turn to describe the details of our artificial data set generation, followed with a summary of the results.

| Noise | Score | N | Cluster # | | Likelihood (train) | | Likelihood (test) | | $\frac{I(C_t,C_e)}{H(C_t)}$ | | $\frac{\#\text{FalseNegatives}}{\#\text{TrueEdges}}$ | | $\frac{\#\text{FalsePositives}}{\#\text{LearnedEdges}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std | Mean | Std |
| 10% | BIC | 200 | 6.0 | 0.00 | 8078 | 240.8 | -3998 | 183.4 | 1.00 | 0.00 | 0.0187 | 0.012 | 0.1389 | 0.009 |
| | | 500 | 6.0 | 0.00 | 20168 | 417.9 | -1302 | 362.1 | 1.00 | 0.00 | 0.0063 | 0.005 | 0.1587 | 0.007 |
| | | 800 | 6.0 | 0.00 | 32422 | 524.0 | -673 | 336.2 | 1.00 | 0.00 | 0.0000 | 0.000 | 0.1608 | 0.006 |
| | CS | 200 | 6.4 | 0.49 | 08109 | 231.7 | -4074 | 265.0 | 1.00 | 0.00 | 0.0083 | 0.008 | 0.1545 | 0.008 |
| | | 500 | 6.6 | 0.49 | 20186 | 415.0 | -1356 | 404.2 | 1.00 | 0.00 | 0.0042 | 0.005 | 0.1643 | 0.005 |
| | | 800 | 6.6 | 0.49 | 32444 | 520.0 | -675 | 328.8 | 1.00 | 0.00 | 0.0000 | 0.000 | 0.1666 | 0.007 |
| 30% | BIC | 200 | 5.6 | 0.49 | 20255 | 234.9 | -6304 | 650.7 | 0.94 | 0.07 | 0.0042 | 0.005 | 0.1555 | 0.004 |
| | | 500 | 5.8 | 0.40 | 50738 | 642.4 | -2847 | 1081.7 | 0.97 | 0.06 | 0.0000 | 0.000 | 0.1666 | 0.005 |
| | | 800 | 5.8 | 0.40 | 81027 | 1415.7 | -2016 | 1224.7 | 0.97 | 0.06 | 0.0000 | 0.000 | 0.1738 | 0.003 |
| | CS | 200 | 6.2 | 0.75 | 20349 | 209.4 | -6187 | 413.2 | 0.97 | 0.06 | 0.0021 | 0.004 | 0.1626 | 0.006 |
| | | 500 | 6.4 | 0.49 | 50988 | 487.6 | -2275 | 126.0 | 1.00 | 0.00 | 0.0000 | 0.000 | 0.1695 | 0.007 |
| | | 800 | 6.6 | 0.49 | 81397 | 691.6 | -1399 | 88.9 | 1.00 | 0.00 | 0.0000 | 0.000 | 0.1738 | 0.003 |

Table 5.1: Summary of results on synthetic data. The results summarize performance of the procedure on data generated from a model with 5 "true" clusters and additional background noise. We report: the number of clusters learned, logarithm of the likelihood ratio between learned model and "true model" on training data (with noisy samples) and test data (unseen samples, without noise), the information the learned clusters contain about the original clusters (see text), the fraction of edges not recovered (# false negative edges / # edges in the true models), and the fraction of false edges recovered (# false positive edges / # edges in learned model). For each figure of merit, we report the mean value and the standard deviation from results from 10 datasets (see text).

We first address the issue of evaluating the classification success, which can be measured in many different techniques. We use the following criterion. A clustering model $\mathcal{G}$ defines a conditional probability distribution over clusters given a sample. Let $\mathcal{G}_t$ denote the true model, and let $\mathcal{G}_e$ denote the estimated model. Both define conditional distributions over clusters. We want to compare these two conditional distributions. We will denote the clusters of the true model as $C_t$ and the clusters of the estimated model as $C_e$. Then, we can define a joint distribution over these two clusterings:

$$P(C_t, C_e) = \sum_{\mathbf{x}} P(\mathbf{x} \mid \mathcal{G}_t) P(C_t \mid \mathbf{x}, \mathcal{G}_t) P(C_e \mid \mathbf{x}, \mathcal{G}_e)$$

where the sum is over all possible joint assignments to $\mathbf{X}$. In practice we cannot sum over all these joint assignments, and thus we estimate this distribution by sampling from $P(\mathbf{X} \mid \mathcal{G}_t)$. Once we have the joint distribution we can compute the *mutual information*,

$$I(C_t; C_e) = \sum_{c_t, c_e} P(c_t, c_e) \log \frac{P(c_t, c_e)}{P(c_t)P(c_e)}$$

between the two clustering variables (**?**). This term denotes the number of bits one clustering carries about the other. In the table below we report the *information ratio* $I(C_t, C_e)/H(C_t)$, which measures how much information $C_e$ provides about $C_t$ relative to the maximum possible (which is the entropy of $C_t$ as $I(C_t, C_t) = H(C_t)$).

We now turn to the second issue of evaluating the structure learning. We measure several aspects of the learned structure. To evaluate the selected number of clusters, we record both the number of clusters in the model as well as the number of "identified" clusters in the model. These are clusters for which there is at least one training sample that is assigned to it. For the CSI structure evaluation we recorded the number of false positive and false negative edges in the implied graph. Recall that an edge corresponds to an informative attribute in the discussion above.

We generated synthetic data from a model learned from Gasch *et al* dataset we describe below. This model had 5 clusters, 93 continuous variables, and 25 discrete nodes. As described in Section **??**, this model (as most of the ones we learned from real data) had several characteristics. The continuous attributes were mostly informative (usually about several clusters). On the other hand, most discrete attributes were uninformative and the remaining ones distinguished mostly one cluster. From this model, we sampled 5 training sets of sizes 200, 500, and 800 samples (15 training sets in total), and a test set of 1000 samples.

We expect that biological data sets to contain many samples that do not fit into clusters. Thus, we want to ensure that our procedure is robust to the presence of such "noise". To estimate this robustness, we "injected" additional noise into the training sets. This was done by adding samples, whose values were sampled uniformly from the range of values each attribute had in the "real" samples we already had at hand. These obscure the clustering in the original model. We ran our procedure on the sampled data sets that we obtained by adding 10% or 30% additional "noise" samples to the original training data.

The procedure was initialized with random starting points, and for each training data we searched for the best scoring model with the number of clusters in the range $K = 3, \ldots, 7$. We then chose the model with the best score among these. Table **??** summarizes the average performance over the 5 training sets in each parameter setting (200,500, or 800 samples with 10% or 30% "noise") using the learning procedure with two scoring methods. We briefly summarize the highlights of the results.

**Search procedure:** We compared the performance of the two variants of the search procedure. The directed approach applies structural EM iterations, and attempt to escape from local maxima by attempting stochastic moves and evaluating each one. The annealed approach applies structural EM iterations where in each iteration, samples are re weighted. In our experiments, we started the annealing procedure with initial temperature (variance of gamma distribution) 2, and each iteration cooled the temperature by a factor of 0.9. In particular, in Figure **??**(a) we see that the annealed search procedure clearly outperforms the directed search on a particular setting. This behavior was consistently observed in all settings, and we do not report it here.

**Cluster number:** In all runs, models learned with fewer clusters than the original model were sharply penalized. On the other hand, models learned with additional clusters got scores that were close to the score received when learning with 5/6 clusters; see Figure **??**(b). Most runs added

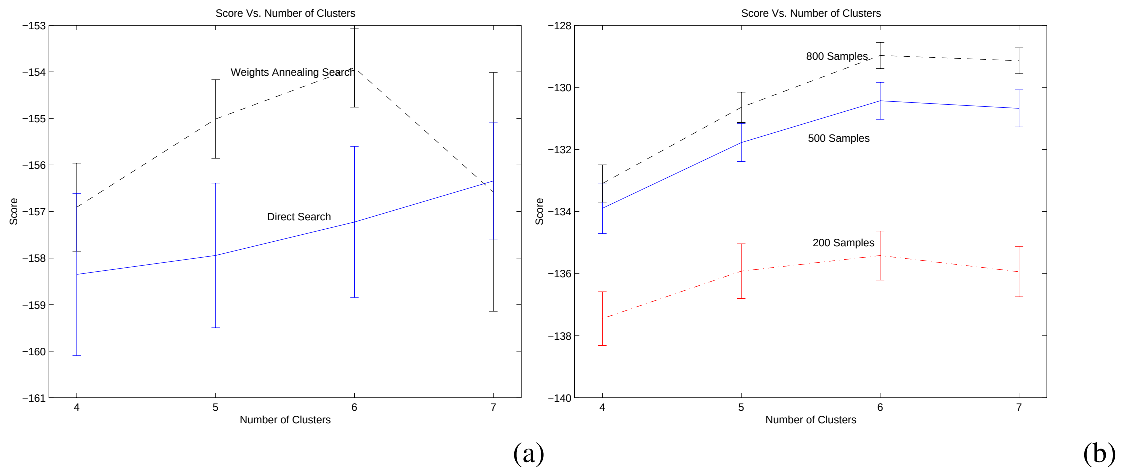


Figure 5.3: Graphs comparing the scores for different cluster numbers. The $x$-axis denotes the number of clusters, and the $y$-axis denote the score per sample (logarithm of BIC score divided by number of samples). (a) Comparison of directed search and weights annealing search on training data with 30% noise and 500 training samples. (b) Comparison of weight annealing search on training data with 10% noise, with 200, 500, and 800 training samples. Each point is the average of 5 data sets, and error bars denote one standard deviations.

another cluster that captured the "noise" samples we added in constructing the training data, and thus, most of the runs pick 6 or slightly more clusters (see Table **??**). In general, runs with the BIC score had stronger penalty for additional clusters, which resulted in choosing 6 clusters as the best scoring model more often. Runs with the CS score sometimes added more clusters. Additionally, as one might expect, the number of chosen clusters tends to grow with strong noise and with larger sample size.

**Likelihood:** As expected, training likelihood is higher than that of the true model. This occur both because the procedure "fits" better the training data, and because of the additional noisy samples in the training data. On the other hand, the learned models are always worse (as expected) on the test data. Additionally, the test data likelihood improves with number of training samples increase, even in noisy data that also has additional noise samples. As expected, models trained with noisier data are somewhat worse than models learned from cleaner data. As a general trend, the training data likelihood of models learned with the CS score are as good as or better than models learned with the BIC scores. This difference is significant mainly in the noisier data sets. The test set performance of both scores is roughly the same when learning with 10% noise (with BIC slightly better) and CS is better in 30% noise.

**Structure accuracy:** We measured the percentage of additional dependencies in the learned graph $G$ when compared to the true structure (false positives) and missing ones in the learned graph $G$ (false negatives). In general, the procedure (using both BIC and CS scores) tended to have very small ratio of false negatives which diminishes as more training samples are available. This shows the procedure is good on recognizing relevant attributes. On the other hand, the procedure had nontrivial number of false positives dependencies, about 13% - 17 % depending on the sample size, the scoring function, and the percentage of noise. In general, when using the CS score, the procedure has a slightly higher ratio of false positive. Similarly, the presence of higher noise levels, also increased the number of false positive dependencies.

**Mutual Information Ratio:** In this category all the runs with 800 training samples achieved the maximal information gain. Runs with 200 samples achieved information gain of 94% and above. Runs with 500 samples had various results that depended on the level of noises. For 10% noise we got maximal information gain, while results in the noisier data set got 97% information gain. As with the likelihood of the data, the CS score had slightly better results compared to the BIC score. These results show that the learned clusters were informative about the original clusters.

Clearly, these simulations only explore a small part of the space of possible parameters. However, they show that on a model that has statistical characteristics similar to real-life datasets, our procedure can perform in a robust manner and discover clusterings that are close to the original one, even in the presence of noise.

### Biological Data

We evaluated our procedure on two biological data sets of budding yeast gene expression. The first data set is from **?** who measured expression levels of genes during different cell-cycle stages. We examined the expression of the $\approx 800$ genes that Spellman *et al* identify as cell-cycle related in 77 experiments. The second data set is from **?** who measured expression levels of genes in response to different environmental changes. Gasch *et al* identified a cluster of genes that have "generic" response to stress conditions. In addition, they identified clusters of genes that responded to particular stress conditions, but not in a generic manner. Our data set consists of the 950 genes, selected by **?**, that responded to some stress conditions but are not part of the generic stress response. Both data sets are based on cDNA array technology, and the expression value of each gene is reported as the logarithm (base 2) of ratio of expression in the sample compared to the expression of the same gene in a common baseline ("control sample").

In addition to the expression levels from these two data sets, we recorded for each gene the number of putative binding sites in the 1000bp upstream of the ORF. These were generated by using the "fungi" matrices in the TRANSFAC 5.1 database (**??**). We used the MAST program (**?**) to scan the upstream regions. We used these matches to count the number of putative sites in the 1000bp upstream region. This generated discrete valued random variables (with values $0, 1, 2, > 2$) that correspond to each putative site (either a whole promoter region or a sub-region). For putative binding sites we also used motif occurrences based on *ab initio* motif search done by the *SeedSearcher*

(Section **??**). We compared the results we got using these motifs to the results we got using the known TRANSFAC motifs.

The *NBClust* algorithm as we described, needs several parameters to be specified. We applied the annealed search procedure with $\sigma_0 = 2, 4$ and $\gamma = 0.5, 0.75, 0.9, 0.95$. Best results were obtained with $\gamma = 0.9, 0.95$ with either $\sigma_0$ settings. As for other variations, such as the technique for choosing the initial model structure, these had no clear cut domination of one technique over the other.

We turn now to describe the results we got on the data sets we just described. In general, there were several common trends in the results on both expression data sets, when used with MAST TF binding sites. First, the expression measurements were considered informative by the clustering. Most of the expression variables had impact on many of the clusters. Second, most binding site measurements were considered non-informative. The learning procedure decided for most of these that they have no effect on any of the clusters. Those that were considered relevant, usually had only 1 or 2 distinct contexts in their local structure. This can be potentially due to the fact that some of these factors were truly irrelevant, or to a large number of errors made by the binding site prediction programs that mask the informative signal in these putative sites. In any case this means these attributes had relatively small influence on the clustering results and only the ones that seem to be correlated with a clear gene expression profile of one of the clusters were chosen by the model.

To illustrate the type of clusters we found, we show in Figures **??** and **??**(a) two of the clusterings we learned. These describe qualitative "cluster profiles" that helps see which experiments distinguish each cluster, and the general trend of expression at each cluster's experiments. Note the schematic illustration of the "masks" that denote the expression attributes that characterize each cluster. As we can see these capture quite well the experiments in which genes in the cluster deviate from the average expression.

Another clear observation is that clusters learned from the cell-cycle data all show periodic behavior. This can be expected since the 800 genes are all correlated with the cell-cycle. However, the clusters differ in their phase. Such clusters profiles are characteristic of many of the models we found for the cell-cycle data.

In the Gasch *et al* data, the best scoring models had twelve clusters. In the model shown in Figure **??**(a), we see two clusters of genes that are under-expressed in stress conditions (Clusters 1, and 2), seven clusters of genes that are over expressed in these conditions (Clusters 3, 5, 6, 7, 8, 10, and 12), two cluster of genes that are over expressed in some stress conditions and under-expressed in others (Cluster 4 and 9), and two cluster of genes with undetermined response to stress conditions (Cluster 8, and 11). These later clusters have high variance, while the others have relatively tight variance in most experiments.

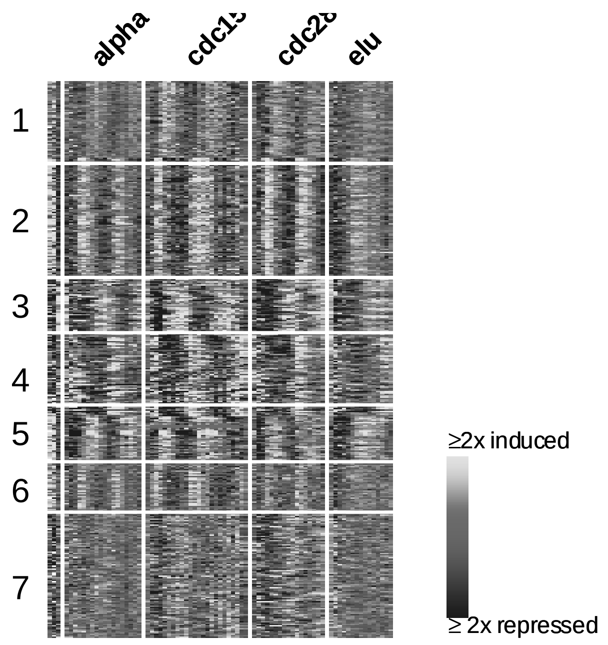


Figure 5.4: The clustering found for the cell-cycle data of Spellman *et al.*. Light pixels correspond to over expressed genes, and dark ones correspond to under-expressed genes. The clusters shown here, where also characterized by the existence of the following binding sites. Clusters 2 and 5: STUAP (Aspergillus Stunted protein), Cluster 3: QA1 (DNA-binding protein with repressor and activator activities, also involved in silencing at telomeres and silent mating type loci), Clusters 4 and 6: HSF (Heat shock transcription factor).

Some of the clusters correspond to clear biological function: For example, Cluster 7 , contains genes that are over-expressed in amino-acid starvation and nitrogen depletion. Examining the MIPS (**?**) functional annotation of these genes suggests that many of the genes involved amino-acid biosynthesis and in transport. Another example is Cluster 2 that contains genes that are under-expressed in late stages of nitrogen depletion, diauxic shift, and under YPD growth medium. This cluster is associated with frequent occurrences of the HAP234 binding site. This binding site (of the complex HAP2, HAP-3, and HAP-4) is associated with the control of gene expression under nonfermentative growth conditions. Many genes in this cluster are associated with mitochondrial organization and transport, respiration, and ATP transport. The association of the cluster with the HAP234 binding strengthens the hypothesis that genes with unknown function in this cluster might be related to these pathways.

**Incorporating *ab initio* motifs:**
We suspected that one of the reasons few clusters are associated with transcription factors binding site is the noisy prediction of these sites. To evaluate the effect of a more informative sequence motifs identification in the upstream region, we performed the following experiment. We applied our algorithm using expression values from the Gasch *et al* data set. Then, we applied the *SeedSearcher*

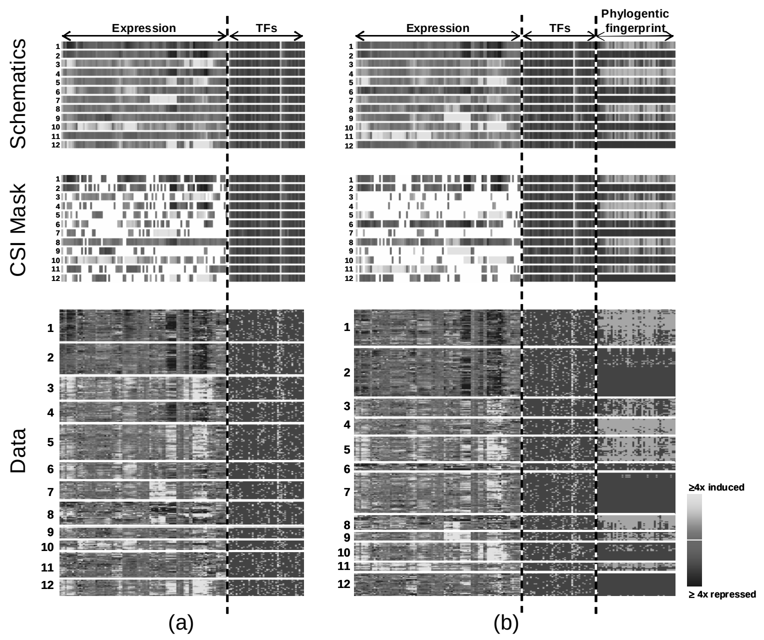


Figure 5.5: Representation of the clustering found in the stress data of Gasch *et al.* (a) clustering based on gene expression and TF putative binding sites. (b) clustering based also on phylogenetic profiles. The top row contains schematic representation of the clustering. The second row contains a "CSI mask" plot that hides all expression features that were considered uninformative by the model. The bottom row shows figures of all the genes, sorted by cluster identity. The following clusters were also characterized by putative binding sites: Cluster 6(a) and 8(b): GCN4 (Transcription factor of the basic leucine zipper (bZIP) family, regulates general control in response to amino acid or purine starvation) and CBF1, Cluster 2(a) HAP234, Cluster 7(b) GCN4.

(Section **??**) to the groups based on the hard assignment of genes to the clusters we identified. Recall that the *SeedSearcher* procedure searches for crude motifs that discriminatively appear in the upstream region of genes in particular clusters and are uncommon in other genes in the genome. We then annotated each gene with the set of motifs we found, and used these annotations as additional input to a new run of our algorithm. Although we only used the *SeedSearcher*'s crude motifs (and not some refined motifs as in Section **??**) the impact on the learning algorithm was clear. Several hundred genes have changed their hard assignment from the initial assignment made when clustering with only expression data, 26 out of 28 motifs were considered informative to the final clustering , and 2 motifs became relevant for 2 different clusters. When we ran the new hard assignments of genes to clusters in the *SeedSearcher* we got a general improvement in motifs identification in clusters in terms of their statistical significance.

**Incorporating phylogenetic profile data**

Next, in order to demonstrate the model's ability to facilitate relevant biological data from different sources, we considered adding additional attributes extracted from the COG database (**?**). This database associates each yeast gene with orthologous genes in 43 other genomes. Thus, we create for each gene a *phylogenetic pattern* that denotes whether there is an orthologous gene in each of the

43 genomes. When we include these additional features, the clusters learned changes. In general, we note that most of the phylogenetic patterns were considered informative by the model but still context specific. For example, we see pairs of clusters (*e.g.*, Clusters 5 and 10) that are similar in terms of expression, yet have distinct phylogenetic profiles. One cluster contains genes that do not have orthologs, while the other cluster contains genes that have orthologs in many bacterial genomes.

Phylogenetic patterns also allow us to gain additional insight into the functional aspects of the clusters. For example Cluster 8 contains genes that are highly over-expressed in amino-acid starvation and nitrogen depletion. It is characterized by occurrences of the binding sites of GCN4 and CBF1, and genes in it have the "typical" profile with orthologs in *C. jejuni*, *P. mutocide*, *Halobacterium sp. NRC-1*, *P. aeruginosa*, *M. tuberculosis*, *A. aeolicus*, *C. crescentus*, H. pylori J99, M. leprae, *D. radiodurans*, *T. volcanium*, and *T. acidophilum*, and no orthologs in *M. genitalium*, *B. burgdorferi*, *C. pneumoniae*, *C. trachomatis*, *S .pyogenes*, T. pallidum, *R. prowazekii*,, *U. urealyticum*, and *Buchnera sp. APS*. This cluster description suggests that this group of genes have common phylogenetic origins as well as common function and regulation.

**Clustering Experiments**

As we noted in the introduction, our method can be used for other clustering tasks. As an example, we clustered the 92 samples in the stress data. In this clustering, we reversed the roles of conditions and genes. Now we consider each condition as an (independent) sample, and each gene as a (continuous) attribute of the sample. The result of the clustering are groups of samples, for each cluster we have the list of informative genes. Not surprisingly, this clustering recovered quite well the groups of samples with the same treatments. Table **??** shows the composition of each cluster in a run with 10 clusters in terms of the original treatments. Each of the following treatments were recovered in a separate cluster: ddt, diamide, YP, and steady state. In addition, the nitrogen depletion time course was split into two clusters. The earlier samples (30 minutes to 4 hours) appeared in a cluster with the amino acid starvation samples, while the later samples (8 hours to 5 days) were clustered separately. This is consistent with clusters we learned over genes, that showed that some genes had distinct behavior in later parts of the nitrogen depletion time course. Similar phenomena occurs with H2O2 samples. Earlier samples (10 minutes - 50 minutes) are clusters with Menadion samples. Later H2O2 samples (60 minutes to 80 minutes, and also 40 minutes) were clustered with sorbitol samples. Finally, both heat shock time courses (fixed temperature, and variable temperature) were mostly clustered in one cluster, although some of the heat shock samples appear in other clusters. These results demonstrate that, as can be expected, different treatments have a clear signature at the mRNA level that can be easily picked by our algorithm. More important, these results demonstrate another possibly important application of the CSI clustering algorithm. As we cluster

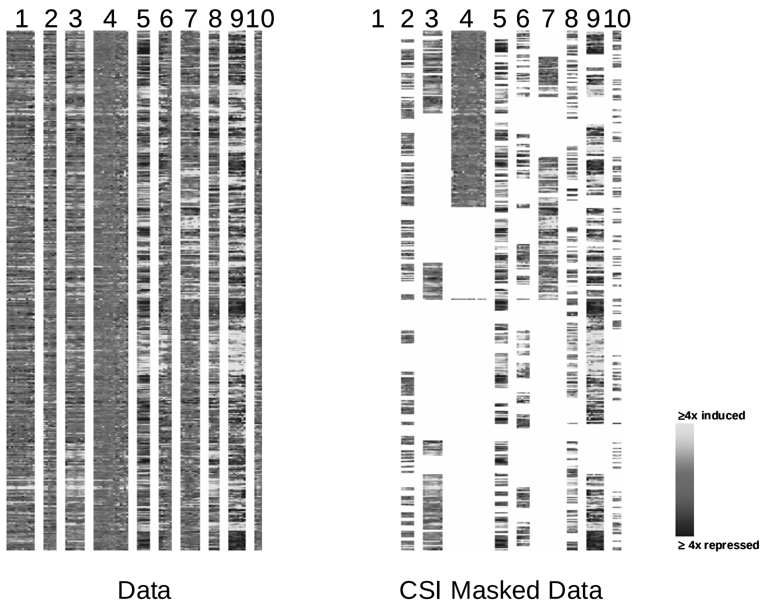


Figure 5.6: Clustering of arrays in the stress data set of Gasch *et al*. The left figure shows the data rearranged according to the clustering. The right figure shows only the positions that are informative the learned models (note that cluster 1 is totally masked in this model).

here experiments over gene as attributes this procedure not only cluster similar experiments but also automatically extracts for each such experiments cluster the genes that are differently expressed in it.

### 5.1.5 Summary

In this work we developed a method for clustering genes based on a combination of genomic and genetic data. We presented an approach that learns from both gene expression data and putative binding sites (from various sources). This approach identifies and characterizes clusters of genes. An important feature of our method is its ability to handle data in which many attributes are irrelevant to the clustering and to tailor each cluster the attributes that it depends on. The experimental results on both synthetic and real data, suggest that this approach is robust to noise and recovers groups of genes with coherent behavior. Although we focused one a particular application — clustering genes based on expression levels and transcription factors binding sites we also demonstrated that the general techniques we develop here are applicable for many other forms of data analysis where one expect to find many irrelevant and "partially relevant" attributes.

We have demonstrated the usefulness of incorporating *ab initio* motifs using the methods we developed in previous chapters, namely the *SeedSearcher* of Section **??**. Most of the motifs identified by it were found to be useful in characterizing the clusters. On the other hand general PSSM

| Condition | Cluster | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Heatshock | | | | 1 | | | | | | |
| Heatshock (variable) | | | 1 | 1 | | | | 5 | | |
| H2O2 | 4 | | | 5 | | | | | | |
| Menadione | 9 | | | | | | | | | |
| DDT | | | | | | | | | | 4 |
| diamide | | | 8 | | | | | | | |
| sorbitol | | | | 7 | | | | | | |
| AA starvation | | | | | | | 5 | | | |
| Nitrogen starvation | | | | | 6 | | 4 | | | |
| Diauxic shift | | | | | | 1 | | | | |
| YPD | | | | 2 | | | | | 8 | |
| YP | | | | | | 5 | | | | |
| Steady state | | 6 | | | | | | | | |

Table 5.2: Confusion matrix comparing the learned clusters of experiments in the Gasch *et al* data set (in a one run of the procedure) to the original division of experiments according to treatment.

motifs from TRANSFAC did not yield high correlations to expression profiles. However, in both cases the *NBClust* does not incorporate the algorithms for motif search directly but rather uses them as a pre processing stage to find motif occurrences. The reason for this is that the unified probability model we described does not include genomic sequence, but just features related to it (like motifs or a phylogenetic profile). When using *ab initio* motifs, we had to cluster the genes, then look for *ab initio* motifs in the clusters, recluster the genes and so on. each stage was done separately. Instead, we would like to have an algorithm that incorporates all steps within a single unified probabilistic framework. This would allow both genomic and genetic information to flow and update both the model we learn over genes (*e.g.*, which groups of genes are controlled in what experimental conditions) and over sequence (*e.g.*, fine tune the relevant motifs in the sequence). We turn now to describe such an algorithm that uses a very different modeling approach.

## 5.2 *DiscRegProg*: Discriminative PRM for Regulatory Programs

In this section we present a probabilistic framework that models the process by which transcriptional binding explains the mRNA expression of different genes. Our joint probabilistic model unifies the two key components of this process: the prediction of gene regulation events from sequence motifs in the gene's promoter region, and the prediction of mRNA expression from combinations of gene regulation events in different settings. Unlike the model described in previous section, these two components are part of a single model and are trained together to achieve maximum predictiveness. This enables the algorithm, *DiscRegProg*, to simultaneously learn sequence motifs that are directly predictive of expression data and discovers clusters of genes whose behavior is well-explained by putative regulation events. Moreover, the general framework allows us to integrate additional data sources as ChIP location assays to improve inference of regulation events. As we shell see, the *DiscRegProg* algorithm relies on the discriminative motif learning algorithms we developed in Chapter **??**. It carries the same discriminative approach to identify co-regulated "modules" too. Each such module is characterized by a specific combination of regulators governing it in specific experimental conditions, and has a distinct expression profile under these conditions. This means the *DiscRegProg* is also able to identify *combinatorial* effects of regulations and relate them to the observed expression profiles.

We start by reviewing the model using the PRM modeling language (Section **??**). The next section describes the learning process of the different parts of the model. We finish this section with a detailed evaluation of the algorithm's results on the cell cycle data of **?** combined with binding location information of **?**.

### 5.2.1 The Unified Probabilistic Model

#### Model Overview

We start with a high-level description of our unified probabilistic model and then elaborate in subsequent sections on the details of its different components. Our model is based on the language of *probabilistic relational models* (PRMs) reviewed in Section **??** A simplified version of our model is presented in Figure **??**. As we see, the PRM framework represents the domain in terms of the different interacting biological entities. In particular, we have an *object* for every gene $g$. Each gene object is associated with several *attributes* that characterize it. Most simply, each gene has attributes $g.S_1, \ldots, g.S_n$ that represent the base pairs in its hypothesized promoter sequence. For example, we might have $g.S_1 = A$. More interestingly, for every transcription factor (TF) $t$, a gene has a *regulation variable* $R(t)$, whose value is $r^t$ if $t$ binds somewhere within $g$'s promoter region, indicating regulation (of some type) of $g$ by $t$. The regulation variables depend directly on the gene's promoter sequence, and the parameters governing this dependency are the TF's motif parameters. Here we use the framework developed for the *LearnPSSM* algorithm (Section **??**). However, note that now when we incorporate this as a component in our PRM model the regulation variables are *hidden* in the data; in fact, an important part of our task is to infer their values.

Our modeling approach also allows us to easily incorporate data from ChIP assays, measuring the extent to a particular transcription factor binds to a gene's promoter region. As we reviewed in previous chapters, these measurements are quite noisy. This means measurements with very high statistical significance indicates a strong likelihood that binding actually took place but we can not simply infer that binding did not take place otherwise. As we did for the *LearnPSSM*

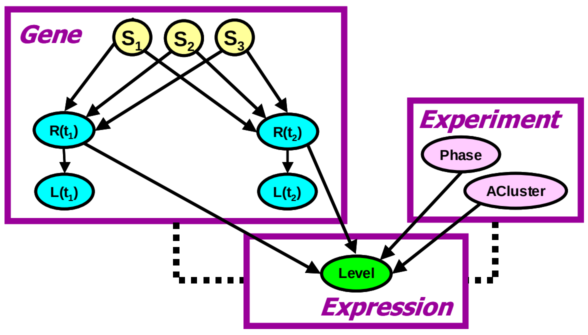


Figure 5.7: PRM model over the biological entities: Gene (including sequence), Experiment (with its characteristic attributes) and the Expression measurements.

and *LearnMotif* algorithms (Section **??**, Section **??**), a natural solution to this problem is to treat the ChIP measurements as *noisy sensors* of the actual regulation events. This solution is easily incorporated into model where the regulation events are represented as hidden variables and the location measurements are conditioned on their values as noisy indicators of their (hidden) values. Specifically, each gene $g$ is added a location variable $L(t)$ for each TF $t$. $L(t)$ is an observed variable, assigned a value according to the statistical test of the binding assay for $t$ and $g$. Our model for the values of this variable clearly depends on whether $t$ actually regulates $g$. For example, values associated with high-confidence binding are much more likely if $g.R(t)$ takes the value $r^t$. The actual conditional distribution are exactly those described in Section **??**.

The second main component of our model is the description of expression data. Thus, in addition to gene objects, we also have an object $a$ for every array, and an object $e$ for every expression measurement. Each expression $e$ is associated with a gene $e.\mathit{Gene} = g$, an array $e.\mathit{Array} = a$, and a real-valued attribute $e.\mathit{Level}$, denoting the mRNA expression level of the gene $g$ in the array $a$. Arrays also have attributes; for example, each array $a$ might be annotated with the cell-cycle phase at the point the experiment was performed, denoted by $a.\mathit{Phase}$. As the array attributes are not usually sufficient to explain the variability in the expression measurements, we often also introduce an additional hidden variable $a.\mathit{ACluster}$ for each array, which can capture other aspects of the array, allowing the algorithm both to explain the expression data better, and to generate more coherent and biologically relevant clusters of genes and experimental conditions.

Our model defines a probability distribution over each gene $g$'s expression level in each array $a$ as a (stochastic) function of both the different TFs that regulate $g$ and of the properties of the specific experiment used to produce the array $a$. Thus, we have a model that predicts $e.\mathit{Level}$ as a (stochastic) function of the values of its *parents* $g.R(t)$ and $a.Phase$ (where $g = e.Gene$ and $a = e.Array$). As we show when discussing the details of the expression model in Section **??** our model allows for combinatorial interactions between regulation events, as well as regulation that varies according to context, *e.g.*, the cell-cycle phase.

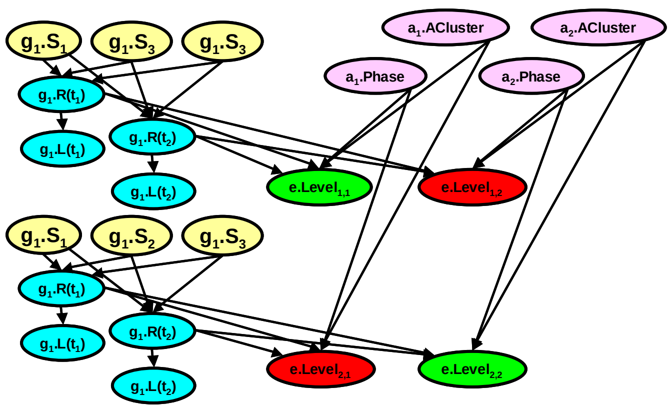


Figure 5.8: An instantiation of the PRM to a particular dataset with 2 genes each with a promoter sequence of length 3, 2 TFs, and 2 arrays.

The model that we learn has a very compact description. As we discuss below, we use the *LearnPSSM* algorithm from Section **??** to learn one PSSM model for each TF $t$, which is then used to predict $g.R(t)$ from the promoter sequence of $g$ for all genes $g$. Similarly, we learn a single model for $e.Level$ as a function of its parents, which is then applied to all expression measurements in our data set. However, the instantiation of the model to a data set is quite large. In a specific instantiation of the PRM model we might have 1000 gene objects, each with 1000 base pairs in its promoter region. We might be interested in modeling 9 TFs, and each gene would have a regulation variable for each of them. Thus, this specific instantiation would contain 9000 regulation variables. Our gene expression dataset might have 100 arrays, so that we have as many as $1000 \times 100$ expression objects (if no expressions are missing). Thus, an instantiation of our model to a particular dataset can contain a large number of objects and variables that interact probabilistically with each other. The resulting probabilistic model is a *Bayesian network*, where the local probability models governing the behavior of nodes of the same type (*e.g.*, all nodes $g.R(t_1)$ for different genes $g$) are shared. Figure **??** contains a small instantiation of such as network, for two genes with promoter sequence of length 3, two TFs, and two arrays.

After reviewing how our model represents the biological entities involved, we now move to describe the details of each of the models components.

### Model for Sequence Motifs

The first part of our model relates the promoter region sequence data to the *Regulates* variables. Our model associates, with each gene $g$, a binary variable $g.R(t) \in \{r^t, r^f\}$ which denotes whether the TF $t$ regulates the gene or not. To simplify notation in this subsection, we focus attention on the regulation for a particular TF $t$, and drop the explicit reference to $t$ from our notation. With these simplified notations, we use exactly model developed in Section **??**. Briefly, this means we have a probabilistic model in which promoter sequence can be generated either by a some Background model or from a motif model represented by a PSSM. We denote by $\theta_0$ the probability distribution over nucleotides according to the background model (a Markov process of order 0) and use $\psi_j$ to denote the distribution of characters in the $j$th position of the binding site, modeled as a PSSM (Section **??**). The model then assumes that if $t$ regulates $g$, then $g$'s promoter sequence has the background distribution for every position in the promoter sequence, except for a specific $k$-mer, $i+1, \ldots, i+k$, where $t$ binds to it. If $t$ does not regulate $g$, we have the background distribution for all positions in the sequence. This means we have:

$$P(S \mid R = r^f) \quad = \quad \prod_{\ell} \theta_0[S^{(l)}] \tag{5.6}$$

And similarly:

$$P(S \mid R = r^t) \quad = \quad \left(\prod_{\ell} \theta_0[S^{(l)}]\right) \sum_h P(H = h) \prod_{j=1}^{k} \frac{\psi_j[S^{(h+j-1)}]}{\theta_0[S^{(h+j-1)}]} \tag{5.7}$$

Which match exactly the equations Eq. (**??**) and Eq. (**??**), assuming a uniform prior over the binding position in case of regulation. [3]

As we have shown for *LearnPSSM* (Section **??**), instead of trying to learn the parameters $\psi_i[l]$ to maximize the probability of the training sequence (as in MEME's *generative* approach) we try find motifs that *discriminate* between promoter regions where the transcription factor binds and those where it does not. This allows us to avoid the common problem of motif convergence to repetitive low complexity patterns common in many promoters and focus instead on the classification task at hand. First, we define the parameters of the model to be:

$$w_j[c] = \log \frac{\psi_j[c]}{\theta_0[c]} \qquad v = \log \frac{P(R = r^t)}{P(R = r^f)} \tag{5.8}$$

As we showed, this definition carries the benefit that we do not need to learn the background

[3]we switch here to $R = r^t$ to mark regulation and $r^f$ to assign no regulation instead of $R \in \{1, -1\}$ we used in Chapter **??**, to make the notations more clear here.

distribution explicitly. Then the formal implication of our discriminative approach is we try to maximize the *conditional* probability of $g.R$ given the sequence $S = S^{(1,\ldots,L)}$

$$\max_{\psi} \sum_{g,g.R=r^t} \log P(g.R = r^t \mid g.S, \psi) + \sum_{g,g.R=r^f} \log(1 - P(g.R = r^t \mid g.S, \psi)) \tag{5.9}$$

Which is just as Eq. (**??**), with the notation adjusted to our current model. We discuss in Section **??**, we train these parameters directly, so as to best predict $g.R$.

### Location Model

Here also we rely on the modeling approach we developed in Section **??** to incorporate prior biological knowledge into our motif learning algorithm. We treat the location experiment as a *noisy sensor* of the *Regulates* variables $g.R(t)$. In current model's representation it means we have for each TF $t$ for which we have location measurements, a new variable $L(t)$, such that $g.L(t)$ represents the location evidence regarding the binding of $t$ to $g$. The value of $g.L(t)$ is set the p-value computed in the experimental assay. We then have in our model a direct probabilistic dependency of $g.L(t)$ on $g.R(t)$. when no regulation is involved *i.e.*, when $g.R(t) = r^f$, then $P(g.L(t) \mid g.R(t) = r^f)$ has a uniform distribution on $[0, 1]$ by design (this is exactly the definition of the p-value here). If on the other hand $g.R(t) = r^t$, then we expect $g.L(t)$ to be small. We choose to model this using a density $p(g.L(t) = \rho \mid g.R(t) = r^t) = c\exp(-w\rho)$, the exponential distribution with weight $w$, where $c = w/(1 - \exp(-w))$ is a normalization constant ensuring that the density function integrates to 1. Based on examination of p-values in the experiments of Simon *et al.*, we choose $w = 20$ in our experiments. Now, once we observe $g.L(t)$, the probability of this observation propagates to $g.R(t)$. If $g.L(t)$ is very small, then it is more likely that it was generated from $g.R(t) = r^t$. If it is larger, then it is more probable that it was generated from $g.R(t) = r^f$. This model allows us to use the location-specific binding data as guidance for inferring regulation relationships, without making overly strong assumptions about their accuracy.

### Model for Gene Expression

We now consider the second major component of our unified model: the dependence of the gene's expression profile on the transcriptional regulation mechanisms. More precisely, our model specifies how, in different experimental conditions, various TFs combine to cause up-regulation or down-regulation of a gene. From a technical perspective, we need to represent a predictive model for $e$.*Level* based on the attributes of the corresponding gene $g$ and array $a$. There are many possible choices of predictive models. Following earlier work by **?**, we choose to use the framework of tree-structured conditional distributions (Section **??**) formalism closely related to decision trees.

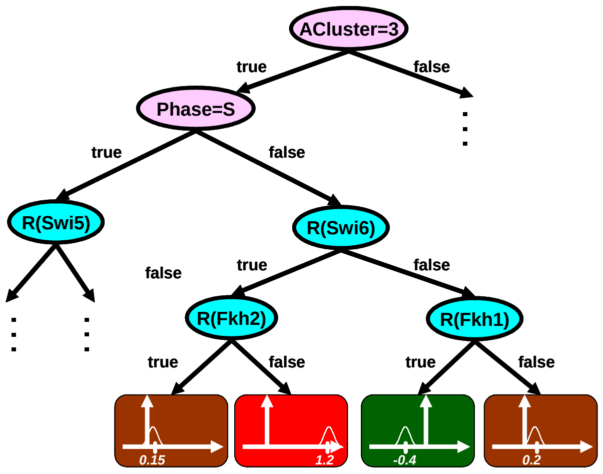


Figure 5.9: An example tree-CPD for the $e.Level$ attribute in terms of attributes of $e.Gene$ and $e.Array$.

This representation is attractive in this setting since it can capture multiple types of combinatorial interactions and context specific effects. Briefly, a *tree-structured CPD* $\mathcal{T}$ for a variable $X$ given some set of attributes $V_1, \ldots, V_n$ is a rooted tree; each *node* in the tree is either a *leaf* or an *interior node*. Each interior node is labeled with a test of the form $V = v$, for $V \in \{V_1, \ldots, V_n\}$ and $v$ one of its values. Each of these interior nodes has two outgoing *arcs* to its children, corresponding to the outcomes of the test (true or false). Each of the leaves is associated with a distribution over the possible values of $X$. In our case, $X$ is the expression level, and therefore takes real values. We therefore associate with each leaf $\ell$ a univariate Gaussian distribution, parameterized by a mean $\mu_\ell$ and variance $\sigma^2_\ell$. Figure **??** shows an example of a partial tree-CPD. We use $\mathcal{G}_\mathcal{T}$ to denote the qualitative structure of the tree, and $\theta_\mathcal{T}$ to denote the parameters at the leaves.

In our domain, the tests in the tree-CPD are about attributes of the gene, (*e.g.*, $g.R(\mathit{Swi6}) = r^t$) or attributes of the array (*e.g.*, $a.\mathit{Phase} = S$). Each leaf corresponds to a *grouping* of measurements. This is a “rectangle” in the expression data that contains the expression levels of a subset of genes (defined by the tests on the gene attributes) in a subset of the arrays (defined by the tests on array attributes).

It is important to realize that the tree-CPD representation can encode combinatorial interactions between different TFs. For example, as shown in the figure, in arrays in cluster 3, except for those in phase S of the cell cycle, genes that are regulated by Swi6 and not Fkh2 are highly over-expressed, whereas those that are also regulated by Fkh2 are only very slightly over-expressed. In addition, the tree can model context specific interactions, where different attributes predict the expression level in different branches of the tree. In our domain, this can capture variation of regulation mechanisms across different experimental conditions. For example, in phase S of the cell cycle, the set of relevant TFs is completely different.

### 5.2.2 Learning the Model

In the previous section, we described the different components of our unified probabilistic model. In this section, we consider how we learn this model from data: promoter sequence data, genomic expression data, and (if available) location 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 key to this joint training of the model are the regulation variables, that are common to the different components. It is important to remember that these variables are hidden, and part of the task of the learning algorithm is to hypothesize their values.

There are several nontrivial subtasks that our learning algorithm must deal with. First we need to learn the parameters of the discriminative motif models. Second, we need to learn both the qualitative tree structure and the parameters for our tree CPD. Finally, we need to deal with the fact that our model contains several hidden variables: the different *Regulates* variables, and the array cluster variables. We discuss each of these subtasks in turn.

#### Learning the Sequence Model

As we shell now show, the first task of learning the sequence model can be carried out by the *LearnPSSM* algorithm we described is Section **??**. The modular nature of our framework enables us to integrate it seamlessly into our model.

Our goal in this section is to learn a model for the binding sites of a TF $t$ that predicts well whether $t$ regulates a gene $g$ by binding somewhere in its promoter region. We defer the treatment of hidden variables to Section **??**; hence, we assume, for the moment, that we are given, as training data, a set of genes $g[1], \ldots, g[M]$ and their promoter sequences, and that we are told, for each gene $g[m]$, whether $t$ regulates $g[m]$ or not.

Given $M$ genes $g[1], \ldots, g[M]$, and the values of their regulation variable $g[m].R(t)$, we try to maximize the conditional log probability

$$\sum_m \log P(g[m].R(t) \mid g[m].S_1, \ldots, g[m].S_n).$$

Which is just a compact form of writing the optimization task giving by Eq. (**??**) and its equivalent Eq. (**??**). Our task therefore is to find the values of the parameters $w_j[c]$ and $v$ for the PSSM of $t$ that maximize this scoring function. Since this optimization problem has no closed form solution we use the solution described in Section **??**. Recall this algorithm is based on *conjugate gradient ascent*, to find a local optimum in the parameter space. Since this type of search typically convergences to a local optimum we need a good starting point to guide our search. Just as with the *LearnPSSM* algorithm, we use a discriminative crude motif found by the *SeedSearcher* (Section **??**) as a starting point.

### Learning the Expression Model

The second task is that of learning the model associated with the expression data, *i.e.*, the model of $e$.*Level* as a function of gene and array attributes: the regulation variables, the array cluster variables, and any other attributes we might have (*e.g.*, the array cell-cycle phase). Again, we temporarily assume that these attributes are all observed, deferring the treatment of hidden variables to Section **??**.

As we discussed, we use a tree-structured probabilistic model for $e$.*Level*, with a Gaussian distribution at each leaf; hence, our task is to learn both the qualitative structure of the tree and the parameters for the Gaussians. Our approach is based on the methods used by **?**. We review its details here.

There are two issues that need to be addressed: a *scoring function*, used to evaluate the "goodness" of different candidate structures relative to the data, and a *search algorithm* that finds a structure with a high score among the super-exponentially many structures.

We use *Bayesian model selection* techniques to score candidate structures (Section **??**). Recall that the *Bayesian score* of a structure is defined as the *posterior* probability of the structure $\mathcal{G}$ given a data set $D$:

$$P(\mathcal{G} \mid D) \propto P(\mathcal{G}) \int P(D \mid \mathcal{G}, \theta) P(\theta \mid \mathcal{G}) d\theta$$

where $P(\mathcal{G})$ is the prior over structures, and $P(\theta \mid \mathcal{G})$ is the prior over parameter values for each structure. This expression evaluates the fit of the model to the data by averaging the likelihood of the data over all possible parameterizations of the model. This averaging regularizes the score and avoids overfitting the data with complex models. When the training data is fully observed, and the likelihood function and parameter prior come from certain families, Bayesian score often has a simple analytic form as a function of the sufficient statistics of that model (Section **??**).

In our case, the set of possible structures are the tree structures $\mathcal{G}_\mathcal{T}$. Each parameterization $\theta_\mathcal{T}$ for the leaves of the tree defines a conditional distribution over $X$ given $V_1, \ldots, V_n$. Given a data set $D = \{V_1[m], \ldots, V_n[m], X[m]\}_{m=1}^{M}$, we want to find the structure $\mathcal{G}_\mathcal{T}$ that maximizes

$$\int \prod_m P(X[m] \mid V_1[m], \ldots, V_n[m], \mathcal{G}_\mathcal{T}, \theta_\mathcal{T}) P(\theta_\mathcal{T} \mid \mathcal{G}_\mathcal{T}) d\theta_\mathcal{T}.$$

We use an independent *normal-gamma* prior over the Gaussian parameters at each leaf in the tree, so this integral has a simple closed form solution (Section **??**).

Having defined a metric for evaluating different models, we need to search the space of possible models for one that has a high score. As is standard in both CPD-tree and Bayesian network learning (Section **??**) we use a greedy 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. The two operators we use are: *split* — replaces a leaf in a CPD-tree by an internal node, and labels it with some binary test $V = v$; and *trim* — replaces the entire subtree at an internal node by a single leaf.

We note that the Bayesian scoring function decomposes as a sum of scores for the individual leaves in the tree. Hence, each of these local transformations changes the score components only in the part of the tree that was modified. This property provides significant computational savings, as transformations to parts of the tree that were not modified continue to have exactly the same effect on the score, and thus their contribution does not need to be re-evaluated. More precisely, let $\tau$ be a transformation on one node $\nu$ in the current model $\mathcal{G}$ (*e.g.*, splitting $\nu$), and let $\delta_\tau$ be the change to the Bayesian score resulting from taking $\tau$. Assume that we, instead, chose to take a different transformation $\tau'$, that left the part of the tree relating to $\nu$ unchanged (*e.g.*, $\tau'$ splits another node $\nu'$). Then the change to the Bayesian score resulting from taking $\tau$ in the new model (the one obtained by applying $\tau'$ to $\mathcal{G}$) is also $\delta_\tau$.

The method we described to search the space of possible structures by simple local transformations is not guaranteed to converge to a global optimum. Hence, to avoid local maxima, we use a variant of simulated annealing: Rather than always taking the highest scoring move in each search step, we take a random step with some probability, which decays exponentially as the search progresses.

### Dealing with Hidden Variables

In this section, we present our overall learning algorithm, which learns a single model that predicts expression from promoter sequence. In a sense, our algorithm "puts together" the two learning algorithms described earlier in this section. Indeed, if the regulation (and cluster) variables were actually observed in the data, then that is all we would need to do: simply run both algorithms separately. Of course, these variables are not observed; indeed, inferring their values is an important part of our goal. Thus, we now have to learn a model in the presence of a large number of hidden variables — $g.R(t)$ for every $g, t$, as well as $a.\mathit{ACluster}$ for every $a$.

The main technique we use to address this issue is the *expectation maximization* (EM) algorithm (Section **??**). However, in applying EM in this setting, we have to deal with the large scale of our model. As we discussed, although our PRM model of Figure **??** is compact, it induces a complex set of interactions. In the experiments we describe below, we have 795 genes, 9 TFs, and 77 arrays, resulting in a Bayesian network model with 7242 hidden variables. Moreover, the nature of the data is such that we cannot treat genes (or arrays) as independent samples. Instead, any two hidden variables are dependent on each other given the observations A key simplification is based on the observation that the two parts of the model — the TF model and the expression model — decouple nicely, allowing us to deal with each separately, with only limited interaction via the *Regulates* variables.

We begin by learning the expression sub model. A good initialization is critical to avoid local maxima. Hence, we initialize the *Regulates* variables with some reasonable starting point. Possibilities include: direct inference from location data; a verified source of TFs such as the TRANSFAC (**?**) repository; or the results of applying a motif-discovery algorithm to the results of an expression clustering algorithm (as in **????**). We also initialize the *ACluster* attributes using the output of some standard clustering program. Treating these initial values as fixed, we then proceed with the first iteration of learning the expression model.

Using this hard assignment to all of the variables, we begin by learning a tree-CPD, as described in Section **??**. We then fix the tree structure, and run EM on the model to adapt the parameters. As usual, EM consists of: an E-step, where we run inference on the model to obtain a distribution over the values of the hidden variables; and an M-step, where we take the distribution obtained in the E-step and use the resulting soft assignment to each of the hidden variables to re-estimate the parameters using standard maximum likelihood estimation. However, computational reasons prevent us from executing this process simultaneously for all of the hidden variables — $g.R(t)$ and $a.\mathit{ACluster}$. We therefore perform an incremental EM update, treating the variables a group at a time. We leave most of the variables with a hard assignment of values; we “soften” the assignment to a subset of the variables, and adapt the model associated with them; we then compute a new hard assignment to this subset, and continue.

More precisely, we iterate over TFs $t$, for each one performing the following process:

- We “hide” the hard assignment to the values of the variables $g.R(t)$ for all $g$, leaving the other variables ($g.R(t')$ for $t' \neq t$, and $a.\mathit{ACluster}$) with their current hard assignment values.

- We perform an E-step, running inference on this model to obtain a posterior distribution for each variable $g.R(t)$. Note that, since $g.R(t)$ is hidden and is part of both the PSSM model and the expression model, inference is done jointly on both, and the posterior of $g.R(t)$ is determined by their combined probabilistic influence on $g.R(t)$ as propagated through the network. We also note that, with fixed assignments to all the other variables, the different $g.R(t)$ variables are all conditionally independent, so that this inference can be done exactly and very efficiently.

- We perform an M-step, adapting the parameters in the model appropriately. Specifically, the M-step may change the Gaussian parameters at the leaves in the tree-CPD, because genes now "end up" at different leaves in the tree, based on their new distribution for $g.R(t)$. Moreover, the M-step involves updating the parameters of the PSSM model. As discussed in Section **??**, there is no closed form solution for performing this update and we thus use conjugate gradient ascent, replacing the hard classification for $g.R(t)$ which we assumed exists in Section **??**, with a soft classification for $g.R(t)$, equal to the posterior probability of $g.R(t)$ as computed in the E-step.

- We pick a new hard assignment for $g.R(t)$ for every $g$, by choosing its most likely value from the distribution.

This process is executed in a round robin for every TF. A similar process is executed for the variables $a.\mathit{ACluster}$. Once we finish these EM updates, we either terminate, or repeat the whole sequence using the updated assignment to the hidden variables: learning the expression model, followed by EM updates. This process repeats until convergence.

From an intuitive perspective, our approach is executing a very natural process. As we mentioned, had the $g.R(t)$ variables been observed, we could have trained each part of the model separately. In our setting, we allow the expression data and the sequence data to simultaneously influence each other and together determine better estimates for the values of $R(t)$.

### 5.2.3 Experimental results

We evaluated our algorithm on a combined data set relating to yeast cell cycle. The data set involved all 795 genes in the cell cycle data set of **?**. For each gene, we had the sequence of the gene's promoter region (1000bp upstream of the ORF), the gene's expression profile on the 77 arrays of **?**, and the location data of **?** for nine transcription factors: Fkh1, Fkh2, Swi4, Swi5, Swi6, Mbp1, Ace2, Ndd1, Mcm1. To simplify the following discussion, we use $\mathit{Reg}_L(t)$ to refer to the set of genes where the statistical analysis of Simon *et al.* indicated binding by the TF $t$ with p-value $0.001$ or lower. We use $\mathit{Reg}(t)$ to refer to the set of genes $g$ where our algorithm assigned a posterior probability to $g.R(t)$ which is greater than $0.5$.

#### Prediction Tests

Recall that one of aims of this dissertation was to develop models able to predict expression levels based on genomic and genetic information. The first evaluation we did was therefore to test how well the learned models predicted genes expression based on the input data - genomic sequence, experiment attributes and location measurements. Since we use a probabilistic framework we were able to measure this by the average log likelihood over genes. To objectively test how each model generalizes to unobserved data, we used 5-fold cross validation. In each run, we trained using 596 genes, and then tested on the expression levels of the remaining 199 held back genes. The partition into training and test set was done randomly, and all models used exactly the same partitions. We report the average log-likelihood per gene. Differences in this measure correspond to multiplicative differences in the prediction probability for the test data. For example, When comparing models $M_A$ and $M_B$, an improvement of +1 in the average log-likelihood by $M_A$ represents that the expression level predictions by $M_A$ have twice the probability relative to the predictions by $M_B$.

The models and results are as follows:

- **Model $M_{cg}$ :**
  In this model we tried to predict the expression data using only the cell-cycle phase attribute for arrays and a corresponding *G-phase* attribute for genes, which represents the phase at which the gene is most active (see (**?**, Supp. data)); average log-likelihood: $-112.24 \pm 11.42$ (standard deviation).

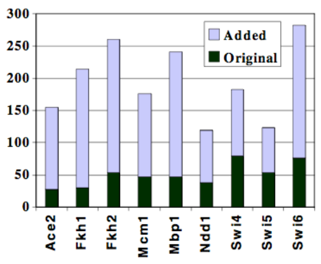
- **Model $M_{cl}$ :**
  Here we tried to predict the expression data using the cell-cycle phase attribute for arrays and *Regulates* variables for genes, whose values were simply set according to *Reg*$_L(t)$; average log-likelihood: $-134.87 \pm 15.29$.

- **Model $M_{cle}$ :**
  The same as $M_{cl}$, except that the location data is treated as a noisy sensor $g.L(t)$ of a hidden *Regulates* variable $g.R(t)$; average log-likelihood: $-121.48 \pm 11.96$.

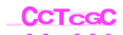
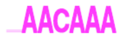
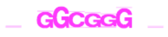
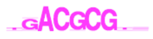
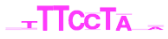
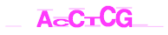
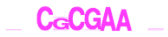
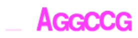
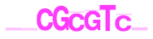
- **Model $M_{cla}$:**
  This model introduces the hidden $a$.*ACluster* attributes (with 5 possible values) into $M_{cle}$, and trains the model using EM; average log-likelihood: $-103.76 \pm 5.72$.

- **Model $M_{cls}$:**
  This model introduces the sequence data into the model of $M_{cla}$, giving us our full model; average log-likelihood: $-94.59 \pm 4.13$.

We see that our baseline model $M_{cg}$ does fairly well, which is not surprising: The *G-phase* attributes were chosen specifically to be an accurate description of the expression profile of the genes. More interesting is the comparison between the other four models. We can see that the location data alone can explain only a small part of the data, and achieves a fairly poor predictive performance. This is mainly due to the conservative approach of **?**, who selected the threshold defining *Reg*$_L(t)$ to minimize the number of false positives; thus, most of the genes in our data set had all the *Regulates* attributes set to false.



(a)

| TF | A | B | C | p-value | PSSM |
|---|---|---|---|---|---|
| **Ace2** | 10 | 9 | 1 | 1.4e-06 |  |
| **Fkh1** | 29 | 25 | 8 | 4.4e-10 |  |
| **Fkh2** | 29 | 29 | 10 | 5.4e-11 |  |
| **Mbp1** | 65 | 56 | 8 | 1.9e-45 |  |
| **Mcm1** | 28 | 24 | 2 | 4.2e-18 |  |
| **Ndd1** | 17 | 28 | 1 | 1.9e-24 |  |
| **Swi4** | 41 | 37 | 5 | 6.4e-26 |  |
| **Swi5** | 28 | 23 | 2 | 4.9e-15 |  |
| **Swi6** | 50 | 52 | 6 | 2.3e-48 |  |

(b)

Figure 5.10: (a) Changes in classification of *Regulates* variables from the initial to the final iteration. (b) Table showing the PSSMs found in the analysis and summarizing their specificity values. The table reports percent "hits" in 3 groups: A — genes in $Reg_L(t)$; B — genes in $Reg(t)$; C — not in $Reg(t)$. The table also reports specificity p-value for the regulated genes.

Treating the location data as a noisy sensor only and running EM improves the accuracy of the predictions. An examination of the data shows that the EM process substantially changes the values of the *Regulates* variables from their initial values according to $Reg_L$, primarily by adding new genes for which regulation is hypothesized, *i.e.*, where the most likely value of $g.R(t)$ is $r^t$. (We discuss this issue further below.) The new $R(t)$ variables are trained to be much more predictive of the expression data, so it is not surprising that the resulting model achieves a higher score. The addition of the *ACluster* attributes also improves the score substantially, as the five-valued cell cycle phase attribute is not enough to distinguish between qualitatively very different arrays. For example, some of the clusters captures distinctions such as "early" vs. "late" in the time series. These distinctions are important, as later measurements lost synchronization to a certain degree and thus are less "sharp". Finally, most interesting is the fact that, by introducing the sequence, we get a very substantial additional boost in predictive accuracy.

We attempted to isolate the source of this last improvement in accuracy. We tested the full model $M_{cls}$ on the test data, but giving it only the sequence information, rather than both the location and sequence information, as in our previous experiment. In other words, although the model was learned using location data, in the test data we are predicting the expression level using *only* the sequence data. In this case, the average log-likelihood is $-95.36 \pm 3.90$, which is almost indistinguishable from the result given the location data as well. This result is quite important, because it suggests that our model has, indeed, learned to predict the expression data directly from sequence data!

**Inferring Regulation**

A second important aspect of our model is its ability to predict regulation relations. We tested the effect the model had on inferred regulation compared to the known motifs, known regulation relations and the original ChIP readings. We note that for performance evaluation we cannot use here the techniques developed in Chapter **??**. Unlike the large scale evaluation of motif finding done there, here we have ChIP data as part of the input, along with other sources of data that the model combines. Moreover, we found that one of the main factors that contribute to the success of the learning algorithm is its ability to *change* the classification of the $R(t)$ variables from their original values as determined by location alone. The algorithm changes these variables to provide a better explanation of the training data. The only constraints on this transformation are those imposed by our choice of the probabilistic model for location, which makes it very likely that if the binding of $t$ to $g$ was associated with a low p-value, the value of $g.R(t)$ will remain $r^t$. Indeed, if we examine the genes for which $g.R(t)$ changed its value, we see that the value changed from $r^t$ to $r^f$ for at most 1 or 2 genes per TF. Thus, the procedure mainly introduced new genes into *Reg*$(t)$. Figure **??**(a) compares the original and final number of genes in *Reg*$(t)$. We see that the procedure increased the number of regulated genes for all the TFs. For some TFs (*e.g.*, Ndd1, Swi4), the change was fairly minor; others (*e.g.*, Fkh1, Fkh2, Swi6) increased by close to 5-fold.

There are several explanations for these changes. In some cases, the genes that we added are also truly regulated by the TF, but the signal was not visible in the location data. For example, our model predicts that Clb1 and Cdc5 are regulated by Ndd1. There is evidence that these genes are regulated by Mcm1 which works together in G2/M with Ndd1. The analysis of Simon *et al.* failed to find binding. Additionally, our model suggests that Fkh1 and Fkh2 are regulated by Fkh1 and not Mcm1. This conclusion fits the recent results of **?**.

A second explanation for this phenomenon is the fact that we gave the model only the nine *Regulates* variables in order to try and explain a complex expression matrix. As such, the model sometimes had to "stretch" the boundaries of these variables in order to improve the quality of its predictions. Thus, it is possible that the semantics of $R(t)$ may have changed in order to capture interactions for which no explanation was present in the data. Nevertheless, we believe that the sets *Reg*$(t)$ are meaningful, and almost certainly co-regulated by some combination of mechanisms.

One strong indicator in favor of this hypothesis is the demonstrated ability of the algorithm to predict expression directly from sequence data, suggesting that there are common features in the promoter region of the genes that our algorithm asserts are co-regulated. We tested this conjecture by looking at the motifs that were discovered by the algorithm. Figure **??**(b) lists, for each TF $t$, the percentage of genes for which the PSSM learned by our algorithm determined the existence of a motif, *i.e.*, those where $P(g.R(t) = r^t \mid g.S_1, \ldots, g.S_n) > 0.5$. We compute the percentage in three groups: the genes in $\mathit{Reg}_L(t)$, the genes in $\mathit{Reg}(t)$, and in the remaining genes (those where $g.R(t) = r^f$). In addition, the table lists the p-value (using a hypergeometric model (**?**)) of the motif and a pictorial representation of the learned PSSM.

This table shows several trends. We see that some motifs (*e.g.*, Mbp1) appear in a majority of the genes in $\mathit{Reg}(t)$. Moreover, in some cases (*e.g.*, Ace2, Mcm1, Swi5) the motif is very rare in the group of genes for which $g.R(t) = r^f$. Thus, these motifs are quite specific for the genes they were trained on, as we can see by the p-values. The significance of the p-values suggests that this is not an artifact. In addition, we see that the motifs have a similar concentration in the genes in $\mathit{Reg}_L(t)$. Thus, although these genes are often less than 25% of the training genes, the learned motif is quite common among them. This last result suggests that there is no difference, from the perspective of finding common motifs, between the co-regulated set of genes discovered by Simon *et al.* and those that were introduced into this set by our algorithm.

This conclusion is further validated by comparing the learned PSSMs to the known binding sites in the literature. The PSSMs for Mbp1 and Swi4 are similar to the ones found by **?**. The PSSM for Mcm1 is also similar to the one found by Tavazoie *et al.*, except that they discovered a homodimer site that consists of two roughly palindromic parts. Our PSSM for Mcm1 captures only one of these parts. (We note that since our model scans both strands, it suffices for the discriminative model to learn only half of the site.) Our PSSM for Fkh1 matches the model suggested by **?**. On the other hand, our PSSM for Swi5 and Ace2 are quite different from the known ones in the literature.

Finally, it is interesting to examine the PSSMs learned from Ndd1 and Swi6. The current hypothesis for Ndd1 is that it cannot bind to the promoter directly. Rather, it is recruited by either Fkh1/2 or by Mcm1. The PSSM we learned for Ndd1 is somewhat similar to PSSM learned by Simon *et al* for Fkh1. Similarly, Swi6 is recruited by either Swi4 or Mbp1; and, indeed, the PSSM we learn for Swi6 is similar (but not identical) to the published Mbp1 motif.

### Biological analysis

We end the discussion of our results by examining their biological plausibility. First, we tested the extent to which our set of potentially co-regulated genes were actually co-expressed. To do so, we computed, for each array $a$, the average of the expression levels for the genes in $\mathit{Reg}(t)$. Figure **??**(a) shows the expression of Swi5 and the average expression of Swi5-regulated genes. We see that the expression of Swi5 regulated genes follows a pronounced cyclic behavior that peaks in the M cell cycle phase. We also can see that the Swi5 gene is transcribed before its protein product is being used. This is consistent with knowledge that Swi5 itself is transcriptionally regulated (**??**). In addition, the fact that Swi5 is transcribed before the genes it activates fits nicely with biological understanding, where the delay corresponds to the time need for translation of the Swi5 transcript

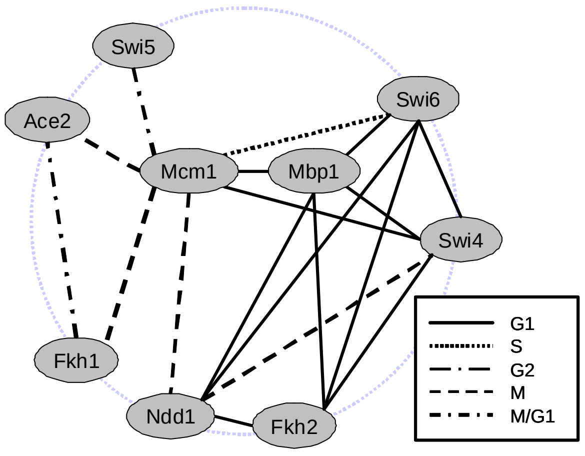
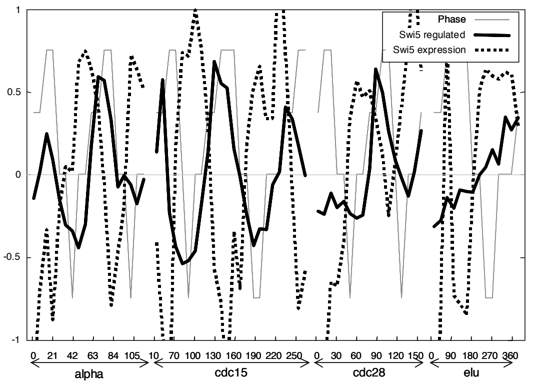


Figure 5.11: Figure of putative combinatorial interactions based on the $M_{cla}$ model.

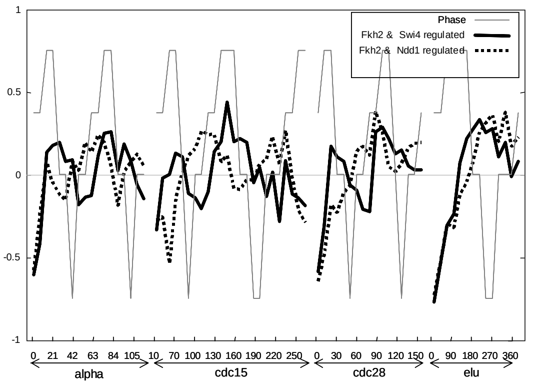
and then the time required for it to bind to its target sites and initiate transcription. As described above, we expect our model to capture effects of combinatorial regulation. These can be seen when we examine the expression of groups of genes that are regulated by two or more TFs. Figure **??**(b) shows that genes that are regulated by both Fkh2 and Swi4 peak in G1 and late G1, whereas genes regulated by Fkh2 and Ndd1 peak in M or M/G1. This behavior is exactly compatible with our current understanding of the role of these two transcription factors complexes that involve Fkh2. Figure **??**(c) shows the behavior of three complexes involving Mcm1: with Ndd1, Ace2, and Swi5. We can see that genes also co-regulated with Ndd1 peak earlier than the other two. Once again, this behavior is compatible with current biological understanding; see **?**.

We then tried to see if we can recover biological insights from our $M_{cla}$ model. Recall that the tree structure learned by our model defines a set of *groupings*, each one defined by one of the leaves of the tree. Each grouping is associated with a list of tests (*e.g.*, $g.R(\mathit{Swi4}) = r^t$) on attributes of the genes and attributes of the arrays that occur on the path to the leaf. It therefore also corresponds to a “rectangle” in the expression matrix (defined by the genes and the arrays that satisfy the tests), which has a similar expression value. The tests performed along the path indicate which aspects of genes and arrays are important for defining a uniform set of expressions. Thus, they can provide biological understanding of the processes.

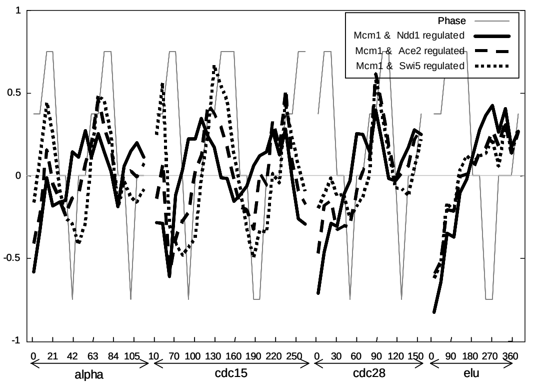
However, before imputing biological significance to these tests, it is important to realize that not all of them are truly relevant. While some of these tests are crucial to defining a coherent set of genes and conditions, others might simply be an artifact of our learning algorithm. We therefore performed significance analysis on each of the tests used to produce this grouping, using a $t$-test to compare the expression measurements in the rectangle with the expression measurements satisfying all other tests defining the grouping except the one in question. We then eliminated tests that appeared



(a)



(b)



(c)

Figure 5.12: Plots of the average expression for subsets of genes regulated by different combinations of TFs. The $x$-axis denotes arrays along the four time courses of Spellman *et al.*, and the $y$-axis denotes average expression level of the genes in each group. The cell cycle phase is shown by by the thin gray line that with peaks in the G1 phase and troughs in the G2 phase.

irrelevant, remaining with a set of overlapping rectangles, such that all of the tests used to define the rectangle were necessary, with a p-value of less than 5e-4.

We selected groups that were over-expressed, in that their average expression level was greater than $0.5$. (Recall that the expression levels of Spellman *et al.* are measured in units of log (base 2) of the ratio to a control.) We note that, in this data, coherent groups are always specific to a particular cell-cycle phase, as determined by the tests in the definition of the group. We then looked for indications of combinatorial regulation: groups which required regulation by two TFs. The resulting "interaction map" is shown in Figure **??**, with an arc between two TFs indicating joint regulation in at least one group. The different arcs indicate joint regulation in different cell-cycle phases.

Many interactions in this map correspond very well with known biology. For example, the interactions between Mbp1 and Swi6, between Swi4 and Swi6, between Ace2 and Mcm1, between Swi5 and Mcm1, between Ndd1 and Mcm1, and between Fkh2 and Mcm1. Other interactions that we would have expected are missing, such as the interaction between Ace2 and Swi5, between Fkh1 and Fkh2, and between Fkh1 and Ndd1. The latter two can perhaps be explained by the fact that Fkh1 and Fkh2 regulate very similar sets of genes, and are therefore somewhat interchangeable. As our learning algorithm looks for compact models that explain the data, it may choose not to introduce one test on a path if if the data is already explained well using some other test. Hence, somewhat redundant tests, such as those on Fkh1 and Fkh2, might never appear together on a path.

Other interactions in the map may suggest potentially interesting hypotheses. For example, finding Ndd1 in interaction with Mbp1 on G1 genes suggests that the Ndd1 protein may participate together with the Forkhead proteins in modulating the expression of Mbp1 targets in G1, as suggested also by the results of **?**. Other interactions also seem compatible with the results of Pilpel *et al.*, including the interaction between Fkh2 and Swi4, Swi6, and Mpb1.

Finally, we compare the genes in these coherent groups to known annotations of genes from the YPD server (**?**). Many of these groups contain a significant portion of genes annotated with a particular functional or cellular role. For example, there is a group of 43 genes that are regulated by Swi6 and Mbp1 but not by Swi4, which contain 8 DNA repair genes (out of 37 such genes in the data). Such a concentration has p-value of $2e - 4$. The same group of genes also contain 14/72 chromatin/chromosome structure genes (p-value = $3e - 6$), 5/8 DNA polymerase or subunit (p-value = $7e - 6$), and 10/36 DNA synthesis genes (p-value = $3e - 6$). These results are compatible with our biological understanding about the processes that occur in phase G1 of the cell cycle, when these TFs are active. Another group of 73 genes is defined as being regulated by Swi6, Mbp1, and Fkh2. It contains 17/77 DNA-binding protein genes (p-value = 9e-5) and 20/72 chromatin/chromosome structure genes (p-value = 4e-7). A final example is a set of 72 genes regulated by Ace2 and not by

Swi4, containing 11/37 amino-acid metabolism genes (p-value = 8e-5).

## 5.3 Discussion

In this final chapter we described two different approaches to create a unified probabilistic framework over genomic and genetic data to infer regulatory mechanisms. The first approach, as implemented in the *NBClust* algorithm, was based on context specific naive Bayes modeling. It tried to identify clusters of genes with similar expression profile and a similar profile of transcription factors binding sites in their promoters. We showed its ability to handle data in which many attributes are irrelevant to the clustering and to tailor each cluster the attributes that it depends on. Our experiments showed the importance of this ability in the case of co regulated "modules" that only appear as such in a subset of the experiments performed and for which only a subset of the transcription factors are relevant. However, as we reviewed in Section **??**, this first attempt with this generative approach also made it clear for us that using biding sites identification based on standard motif is not informative enough. This motivated the development of our discriminative approach for motif finding along with the algorithms described in Chapter **??**. These were all integrated in the much more complex model, *DiscRegProg*, we described in the second part of this chapter. There we used the PRM modeling language to describe a unified probabilistic framework that defines a (simplified) model of the "end-to-end" process of genomic expression: from transcriptional regulation, based on the binding of transcription factors to the gene's promoter region, to the expression data itself. We showed how we can learn a coherent model based on heterogeneous data: sequence data, expression data, and binding location data. We demonstrate the *DiscRegProg* algorithm on the yeast cell cycle process and showed how we were able to learn to predict expression from sequence. Our algorithm also simultaneously learned highly significant motifs in clusters that it asserts were co-regulated, providing a strong biological basis for this claim. We showed that the learned model provides valuable biological insight into the domain, including information about combinatorial regulation by complexes of transcription factors.

There have been many works in recent years aimed to elucidate the regulatory mechanisms at the transcriptional level using diverse sources of data as input. We can categorize them into several classes. One is the "step by step" approach, were each source of data is processes and the results of this is used to analyze another type of data in the next step. Some works first cluster genes based on gene expression and then try to identify motifs characteristic of each cluster (*e.g.*, **??**). Other works work in a different order of steps, first trying to find sequence motifs and then test if they correlate with gene expression. For example, **?** used *AlignAce* to search for PSSM motifs in groups of genes suspected to be co regulated and then tested the hypothesis that a combination of two motifs serve as a regulation "module". For this they compare the average expression profile of genes found to have both motifs compared to ones having only one of them. If there is a significant difference they report it as a putative regulation program. Several other works tried to use a similar approach with other sources of data too. For example, **?** used the same approach but this time defined the groups of genes regulated by two transcription factors based on ChIP results by **?**.

A more recent trend in computational search for such transcriptional regulatory modules involves yet another source of data, phlogenetic comparison of closely related species. The idea behind this approach is that by such a comparison we can find subsequences in the non coding promoter areas that are more conserved then expected. Such areas are good candidates for binding sites because this functional rule would put negative selective pressure that might explain their preservation. **?** used this approach comparing three related yeast species (S. paradoxus, S. mikatae and S. bayanus) to come up with a large collection of 72 motifs. Later **?** used also closely related yeast species and combined it with statistical tests over expression data and ChIP assays to come up with a large collection of motifs and some draft of how they combine to affect expression in all the yeast genome, including some tandem combinations of transcription factors.

All the works we just reviewed share a common characteristic. They do not propose an algorithm to learn regulatory modules automatically and/or fit some function/model to the observed data. Instead they use (fixed) results from one step (*e.g.*, motifs found) to guide the search at the following step (*e.g.*, combination of motifs affecting expression). Usually, at each stage they use statistical tests for specific hypothesis they want to evaluate. Another class of works is where the various sources of data are combined together, trying to fit some model or optimize a target function. As we have shown, the possible advantage of this approach is that information can flow and improve overall predictions. For example, the motifs learned in the *DiscRegProg* were optimized to match evidence from common expression profiles and vis versa. This allowed us to improve our predictions of regulation events and consequently the expression profile prediction, compared to using location data “as is” to infer regulation and predict expression. Another possible benefit of this kind of approach is its ability to capture certain relation between the entities modeled under specific conditions which might be otherwise overlooked. For example, during learning and fitting the *DiscRegProg* model to the data we might find that a combination of two motifs is crucial to explain co-expression. but then, since the algorithm keeps trying to fit the data better, we can identify that we need a third motif, but only under some specific cellular conditions. This would be hard to discover by undirected statistical tests since the combination of all experimental conditions and all triple motifs is hard to explore.

In the class of works that fit a general overall function/model we can distinguish between several sub types. Some of the algorithms that combine expression and sequence data together first parse sequence data into pre defined (fixed) features. The *NBClust* algorithm we described in the first part of this chapter is one. Another work of this kind was done by **?**. They used linear regression, where the hypothesis is that the expression level is normally distributed around a value, whose mean is a linear function of the presence or absence of the different sequence attributes. Comparing the two we note that the later can easily handle a very large set of sequence motifs but is limited to linear combinations of them. Also, it is not able to characterize subsets of experiments which are relevant to a modules definition.

Unlike these works, **?** describe *Naive Bayes* model similar to that of *NBClust* but incorporating both promoter sequence and expression in a unified framwork. However, compared to *NBClust*, their model assume only one binding site per cluster, and cannot detect multiple regulatory sites, nor detect that some experiments are irrelevant to the definition of a specific cluster. **?** describe an alternative unified framework for location and expression data. Their approach is based on the Bayesian network framework for pathway discovery (**??**). They use the location data to guide the discovery by limiting the set of models they consider. Their approach, however, only makes use of the small set of regulation predictions that received very low p-values in the analysis of the location data. As our results show, it is possible as well as beneficial to make use of all of the location results. More recently, **?** suggested another method based on Bayesian networks aimed to explain expression as a function of sequence motifs. One important addition in their proposed method is that sequence motifs were characterized by their positional preference, orientation, proximity and order (for pairs of motifs). They created clusters of genes based on co-expression and then used bayesian networks to learn such motif features and the dependency relation between them so as to best explain the observed expression. However, in contrast to both *NBClust* and *DiscRegProg* they do not suggest a unified model over expression and sequence or sequence features. As a result, they do not allow genes to change module assignment after the initial step.

Its Important to note that learning graphical models, although naturally appealing in this setting, is not the only machine learning approach that can be applied to infer regulatory programs. For example, recently **?** suggested an algorithm (MEDUSA) based on the discriminative *ada boost* technique. They use as input gene promoter sequence, gene expression data and a set of possible regulators to learn rules which are predictive for differential expression of genes. The rules learned by the algorithm are based on weighted majority voting in a structure of a decision tree. The weak rules in the tree include binary questions of two types. The first type is questions about the activity (*i.e.*, differential expression) of regulators from the given regulators set. The second type of questions in the rules learned are of presence of simple sequence motifs. The actual motifs chosen are part of the learning process. Compared to our *DiscRegProg* we first note their approach first quantitate gene expression to -1,0,1 in a pre process step. Because of computational efficiency limitations their model is restricted to handle only short motifs ( 7bp long), which is problematic

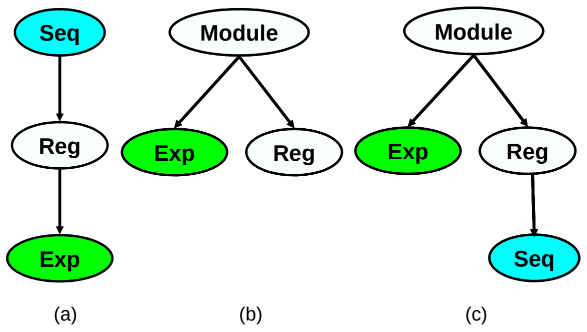


Figure 5.13: Illustration of modeling approach for regulatory programs in unified probabilistic models. Hidden variables are colored in white. The *DiscRegProg* discriminative approach (a), The *NBClust* generative approach (b), and a possible extension to the *NBClust* model including sequence data (c).

from a biological point of view. Also, the absence of a model makes interpreting the results less intuitive and incorporating prior biological knowledge (as ChIP results, not used by MEDUDA) less natural. Finally, we note that none of the works we reviewed offers integration of expression, location, and sequence into a unified probabilistic framework as the *DiscRegProg* model does.

The two algorithms we presented, *NBClust* and *DiscRegProg* can not be directly compared. The later uses a much more complex model and incorporates sequence data directly along with ChIP location assays. Of course the complexity of the model comes with a price - Learning this model is more computationally involved and may take several days on data set as the ones we used. *NBClust* is a much simpler utility for general clustering of heterogeneous data from diverse sources. It does not model sequence directly , only the presence of (fixed) binding site motifs. The motifs themselves are not part of the learning process. However, there are some good lessons to be made from comparing the different modeling approaches taken in both cases. In the case of *NBClust*, we took a generative approach illustrated in Figure **??**(b). Here we had the regulatory *module* modeled explicitly , by the hidden cluster/module variable to which each gene is assigned. In the *DiscRegProg* algorithm we used a more discriminative approach, trying to explain the observed differences in expression profiles via regulation events and in turn trying to find sequence motif that best match our suspected regulation events. This approach, illustrated in Figure **??**(a), did not try to *explain* the observed sequence data but rather condition on it to explain the observed expression profile, using the location data as noisy sensors of regulation events. Note the regulatory *modules* are not represented explicitly in this model by any variable. Rather, they are implicit in the dependency of the expression on regulation variables and experiments attributes (Section **??**). We showed that this context specific dependency is captured by a tree structure where each node can be thought of as a module of regulation, acting under the specific experimental conditions defining by the path along the tree to this specific leaf.

The above comparison points towards an obvious extension to the work presented in this chapter and illustrated in Figure **??**(c). This optional model uses the generative probabilistic model as in Figure **??**(a) but incorporates both sequence and location data as part of the model to infer about the (hidden) regulators that characterize each of the (hidden) modules of regulation. Such a model involves developing a probabilistic model for promoter sequence given the regulators involved in it.

Another possible extension to our models involve incorporating other sources of genomic or genetic data into our model. Obvious candidates are phlogenetic comparison of closely related species as we described above for the work of **?**, or protein interactions. **?** have already shown how the later can be combined with expression data to infer molecular pathways. Other extensions include incorporating more accurate motif models as the ones we described in Chapter **??** or sequence features as the ones used by **?**. Obviously, without modeling these sequence features and more accurate motifs our models might simply fail to identify regulation relations. These extensions point to a more fundamental possible change to our sequence modeling as well as learning approach. Instead of learning each motif separately we should consider modeling the entire promoter sequence as a whole, learning both sequence features as used by **?** as well as all binding site motif simultaneously. Another obvious utility the *DiscRegProg* algorithm lacks is the ability to include new motifs or exclude non relevant or redundant ones during the learning process. This was indeed added in a follow up work by **?** using a similar model to *DiscRegProg* with PSSM motifs but lacking the location data.

A whole different direction for extending our models is by adding more hidden variables that would expend the model's expressiveness and make it correspond more accurately to the underlying biology. Such variables might correspond for example to a regulator being active or not in specific conditions, express the fact it serves as a repressor or activator under these conditions, or designate the level of its activity. We might be able to use time series data to infer the values of these hidden variables, making our models more dynamic and time dependent. Promising work in that direction has already been done by **?** but it did not include sequence modeling.

# Chapter 6

# Discussion

## 6.1 Summary

In this dissertation we aimed to increase our ability to infer the transcriptional regulatory mechanisms used by the living cell. Our approach was to develop models for these regulatory programs along with algorithms to learn them from diverse sources of data. We used machine learning methods and probabilistic graphical models in particular for this. We argued that by combining various sources of genomic and genetic data we can improve on current computational approaches suggested for this task. Unlike several works that also utilized expression data to infer about biological pathways using probabilistic models (*e.g.*, **??**), our approach was *sequence oriented*: we were looking to understand the code of the regulatory programs as given by the DNA sequence. Specifically, we wanted to improve our transcription factors binding site predictions and to identify the regulatory programs they are involved in: what specific combinations of transcription factors make a regulatory program, at what cellular conditions are they activated, and what are their target genes. Furthermore, we wanted to exploit our sequence based models to improve gene expression profile predictions, thus bridging the gap from the observed DNA sequence to the observed mRNA levels.

The works we presented as part of this dissertation went a long way towards these goals. The first two chapters concentrated on improving our ability to identify the core of the DNA code for transcriptional regulatory programs - the binding sites of transcription factors. We first developed the *discriminative* approach for motif learning with the *SeedSearcher* and the *LearnPSSM* algorithms, and then extended it to learn binding site models with interdependent positions using *LearnMotif*. The motif models we developed, based on the Bayesian networks language, also served to demonstrate the flexibility of our approach: Just as we use Bayesian networks to model the regulatory programs, we can use them to model the binding sites themselves. To actually use the motif models for binding site predictions we developed algorithms (*TestPSSM* and later *CIS*) to efficiently assess the statistical significance of putative sites. To assess the usefulness of our motif finding algorithms we developed methods to evaluate their predictions using high throughput biological experiments and available motif data bases. With these, we were able to show the significant improvement in binding site predictions achieved by our algorithms.

Finally, we incorporated our motif finding algorithms into two different models which represent two very different approaches for modeling and learning regulatory modules. The first, *NBClust*, represents a generative approach using a naive Bayes model that tries to fit the observed data. The hidden cluster/module variable in this model cluster genes based on common expression profile as well as common binding sites found in their promoter regions. The model used *context specific independence* (CSI) in the form of *default tables* to overcome the fact that many binding sites or experiments may be non relevant for characterizing each cluster/module of genes. Using *NBClust* we were able to identify meaningful groups of genes with functional enrichment. By incorporating motif finding based on our *SeedSearcher* algorithm into the *NBClust* model we were able to illustrate their usefulness compared to standard motifs that were found to be much less informative by the algorithm.

The second algorithm for learning regulatory programs, *DiscRegProg*, represents a more discriminative approach for learning regulatory modules. It used the language of PRMs to build a model that included biding site motifs, regulation events (as hidden variables), and various gene and experiments attributes all of which used to explain the observed differences in expression measurements. This model used context specific independence too, in the form of tree CPDs, to learn combinatorial interactions of transcription factors, what genes they affect and in what cellular conditions. These tree CPDs implicitly coded the regulatory programs identified by this model. *DiscRegProg* also demonstrated the flexibility and modularity of our approach in several ways: It enabled us to incorporate ChIP assays into our model, as *noisy sensors* of the hidden regulation events. It also incorporated the *LearnPSSM* framework as part of its model over sequence. This enabled us to optimize motif learning as part of the model learning so as to better explain observed expression patterns. Using cell cycle data we were able to show how our model improve expression prediction over regular clustering technique. Moreover, we showed how incorporating sequence data improves expression profile predictions. Finally we related the inferred regulatory programs to known and novel mechanisms.

## 6.2 Limitations and Future Directions

In this dissertation we developed several algorithms and showed their usefulness for inferring regulatory mechanisms. However, it is also obvious that the scientific questions that motivated us are far from being resolved. We will now review the limitations of our current solutions and try and point for future directions that might help resolve these.

First, It is important to grasp the computational price that comes with the more complex models we developed. The *DiscRegProg* algorithm that learns PRM models for regulatory modules represents quite well the complexity of this approach. Since the PRM holds an object for each gene and each measurement in each experiment is represented by a separate random variable, just learning a model over the the cell cycle expression data of **?** for 800 genes with 500bp long promoters can take more then a day. The computational costs of these models also points to future direction of research, that of developing new and efficient techniques for performing inference and learning tasks in this type of models. Some of the other algorithms we developed also had a nontrivial computational price. The *SeedSearcher* algorithm for example, was highly optimized to quickly report statistically significant crude motifs in varying length. This optimization also came with a price - the prefix tree data structure we use takes more then 1 gigabyte of memory when analyzing the yeast promoter regions. We extended the algorithm to support other data structures, thus allowing to analyze bigger data sets, at the cost of additional computation time.

As we repeatedly pointed out along this dissertation, one of the major ways our model might improve and become more accurate is by incorporating more sources of data into them. This includes phylogenetic comparison of closely related species to help direct motif search (**?**) and exploiting knowledge about nucleosomes positions (**?**). Protein-protein interactions data was already employed in several works involving probabilistic graphical models (**??**). How to incorporate these sources of data into probabilistic models for regulatory programs involving DNA sequence remains open.

Specifically, a major future challenge is to employ such sources of data to move into more complex sequence models of regulatory programs. Such models may include constraints like position preference along the promoter, relative order of binding sites in a specific module, multiple motif occurrences, strand bias, location relative to nucleosomes positions, etc. Such models should also be able to learn the various binding site motifs simultaneously. This is a major change from the setting in *any* current motif learning algorithm, including ours. Currently, even when learning several motifs as in *DiscRegProg*, the sequence model always assumes just a background model and a single motif. One possible way to extend our models towards these ends in to use *generalized HMM* models (**?**). **?** used a similar model for *identifying* predefined CRMs (*Cis*-Regulatory Modules) over given sequences. Using this modeling approach to *learn* regulatory module using the diverse sources of data mentioned above remains as a future direction of research. Other possible models for such complex sequence features of regulatory programs may be based on sparse, undirected graphical models such as *Factor Graphs* (**?**)

On a more higher level, we should point out two main issues related to the machine learning approach we used. First, to successfully develop machine learning algorithms for real world problems, one must formalize effectively human prior knowledge about the domain in terms of what the machine learning task is and how the models are constrained. For example, if we do not confine our model for regulatory modules to select possible regulators from a reasonable set of genes, the resulting search space would be so vast and hard to explore that the algorithm might converge to much poorer models. Generally speaking, effective incorporation of all prior knowledge into the model and learning task definitions is much more easily said then done. It still remains more of an art then a computational procedure. Devising more principled approaches for these remains another open challenge.

The second issue, closely related to the first, is the constant tension between the level of details we would like to capture of the systems we model, and what we can hope to recover from the limited input data we have. The theoretical aspects of this problem are well formalized in machine learning theory (*e.g.*, concepts like VC dimension, efficient PAC learnability, over fitting etc. - see for example **?**). Specifically for our domain, as more data becomes available, we may hope to capture more of the systems *dynamics*. This includes both the *time* factor in regulatory processes, and complex dependencies over time that involve *feedback* effects. For example we might be able to model the *rates* of transcription as a function of the amounts of *active* transcription factor molecules while taking into account the amounts of induced factor, degradation rates etc. Preliminary promising work in this direction has been done by **?** and **?**, but did not involve modeling regulatory modules over DNA sequence. Modeling feedback effects would probably involve adopting our modeling language too. Bayesian networks and PRMs are unsuitable for this task as they are both based on directed acyclic graphs, while feedback involves structures with loops. Possible solutions for this in the framework of probabilistic graphical models may be found in DBNs (Dynamic Bayesian Network) or models based on undirected graphs such as Markov Random Fields (**?**).

Another important future direction for improving our algorithms is to supply them with complementary methods to assess the confidence in model features. As we have shown, structural features of the models imply biological hypotheses. However, we must also bare in mind that our machine learning model based approach will always result in *some* model being learned. We therefore need to be careful in how we use and interpret our models. For example, assume *LearnMotif* learns a binding site motif with two interdependent positions. If we just use it for binding site predictions (as we did in the work presented here), this may suffice for our needs. But suppose we want to use the learned model to relate it to the underlying structure of the DNA binding domain of the transcription factor's family. We would then need to supply some measure of confidence that two positions are indeed dependent. In another case say *DiscRegProg* learns transcription factor $t_1$ control gene $g_1$. An immediate question of any biologist who contemplates on whether to go and check this in a wet lab experiment is how sure we are of this model feature. Another question related to confidence in model features is about *causality* relations between the entities we model. As we reviewed in Section **??**, there is a distinct difference between identifying (in)dependence and causal relations. We note that analyzing the confidence in model features fits well within our Bayesian approach for learning graphical models: According to the Bayesian approach we answer this by averaging over all possible models to get a posterior probability over the specific feature we are interested in (like an arc between $t_1$ and $g_1$). Initial work towards this end has been done by **??**. However, coming up with more ways to efficiently estimate our confidence in various model features remains a challenge.

The computational complexity we described and the need to come up with techniques to validate model features are definitely a drawback of our approach, compared to the commonly used approach of direct statistical hypothesis testing. For example, **?** uses a statistical test over expression profile distributions to determine whether the presence of two sequence motifs in promoters act together as a regulatory program. In general, using this computational approach is usually much easier to implement, fast, and results in the hypothesis either been validated (with a “built in” confidence measure) or not. However, we note that there is a major difference between the two approaches: statistical hypothesis testing is usually based on some assumed *null* hypothesis which is not always easy to come up with or may not match our domain very well. More importantly, performing just a fixed set of statistical tests is inherently limited to discovering only what we *expect* to find. The machine learning approach on the other hand, is all about letting the machine *explore* and find out the domain’s infra structure, given our basic definitions of it. As a result, machine learning algorithms can come up with hypotheses which are otherwise hard to discover. For example when we try to discriminate between different expression profiles in the *DiscRegProg* algorithm, we may find that adding a certain transcription factors combination explains a difference in expression profiles in a certain *subset* of experiments for which we failed to learn a good “explanation” until this point. Direct statistical tests may fail to discover this since this phenomena is *context specific* and we only get to identify it during the model learning process. Both the model-learning based approach and direct hypothesis testing have proved valuable for analyzing this domain. As we have showed in the case of *DiscRegProg*, hypothesis testing can also be used to validate model features. Devising more principled approaches for this is another direction for improving our algorithms.

An important practical issue about the model based machine learning approach is how to present the models to the human eye. Although this may not be defined as an academic field of research, our ability to extract meaningful hypotheses from our models is closely related to this. For example a PSSM *sequence logo* is a very simple graphical representation for a motif model we can easily interpret. Coming up with a graphical representation for mixture of trees motif models that is as easily interpretable as a sequence logo remains a challenge. More generally, we need to develop more tools with graphical user interface that enable us to visualize our models and perform queries over them. This is strongly related to the previous point we discussed, of assessing the confidence in certain model structural feature. For example, a researcher might want to take a model learned for some regulatory module and after closely inspecting it decide to test the affect of adding some regulator to the model, based on his/her knowledge. The algorithm should then restart model learning from this particular point, with the added new constraint and be able to quantitate the effect of the change. Although in theory we have much of the learning machinery for this kind of tasks, in practice we are far from having these easy to use, user friendly tools. Future development of such algorithms, along with the matching graphical representation of the models is crucial for making our methods more approachable.

To summarize, we strongly believe that our approach of using machine learning in general and probabilistic graphical models in particular, carries a great promise for helping to shape biological and medical research in terms of information processing. We hope the algorithms we developed, besides their practical usage by our collaborators, have helped show this. As a final note, I would like to end with a comment about the future role of machine learning in biological research as I see it, based on a commonly known joke:

> A biologist and a mathematician are both given the same task: predicting the winner in a horse race. The biologist comes back after a year he spent in studying the ancestral trees of all known winners and examining their tissues under a microscope, and says he is now able to predict the winner 30% of the times. "That's a very poor result" comments the mathematician, who spent much less time on the task. "I can predict the winner with 100% accuracy" he claims. "But it only works for spherical horses with no friction".

As I see it, our task as practitioners of the machine learning approach, is exactly to bridge the gap between these two researchers. We need to come up with computational ways to analyze the biologist various sources of data so as to help him better understand what makes a horse a winner and better predict the winner. Maybe limited data, computational power or modeling techniques would only lead us to predict the winner in 70% accuracy. But at least in this case a predictive power of 70% is something many would be willing to settle for.

# Appendix A

# Bayesian Networks & Relational Models

The aim of this appendix is to give a formal review of the type of probabilistic graphical models we use in our work. The algorithms and models we develop demonstrate how this formal theory and related algorithms can be facilitated for our tasks.

The foundations of probabilistic graphical models as an active field of research in machine learning has been laid in a seminal book by Peal (**?**). Specifically, Pearl also introduced the *Bayesian Network* model, which we use extensively in this work. Many forms of probabilistic graphical models have been developed during these past years. They include *Hidden Markov Models* (HMMs) (**?**) which have been used extensively to in computational biology for tasks such as gene finding (**?**), *Decision Trees* (**??**), *Markov Random Fields* (**?**), and *Factor Graphs* (**?**) just to name a few. One of the generalizations of *Bayesian Network*, developed to handle relational data, is *PRM* (Probabilistic Relational Models). This is the second type of models we use in this work.

Probabilistic graphical models have been successfully applied to solve problems in numerous fields (see, for example, (**?**)). Their success can be attributed to the way they are able to compactly and intuitively represent uncertainty and stochasticity in the domain of interest, capturing some of its inherent structure. A key feature is how these models encode inherent independence relations between the entities in the domain into the graph structure they use. We will therefore start with formalizing the notion of *conditional independence* between random variables and then see how these applies to Bayesian networks models.

## A.1 The Bayesian Network Model

A Bayesian network is a (compact) representation of a probability distribution over a domain $\mathcal{X} = \{X_1, \ldots, X_N\}$ where each $X_i$ is a random variable. To set up our notations from here on, $X_i$ may be discrete, in which case it may take on any value $x_i$ from the domain $Val(X_i)$, or it may

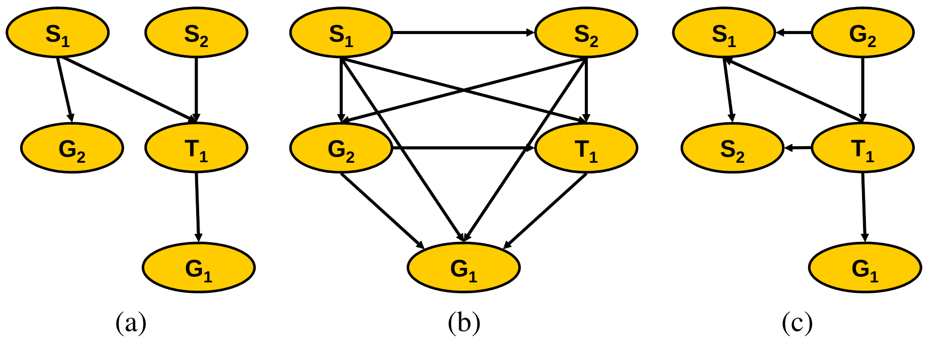


Figure A.1: (a) An example of a simple Bayesian network structure for a regulatory network. The network structure implies several conditional independence statements: $(S_1 \perp S_2)$,$(T_1 \perp G_2 \mid S_1, S_2)$,$(G_2 \perp T_1, S_2, G_1 \mid S_1)$, and $(G_2 \perp S_1, S_2, G_2 \mid T_1)$. The joint distribution has the product form $P(S_1, S_2, T_1, G_1, G_2) = P(S_1)P(S_2)P(T_1|S_1, S_2)P(G_2|S_1)P(G_1|T_1)$. (b) A graph representation over the same domain, but with no independencies encoded in it (a full graph). The matching Bayesian network qualifies as an *I-map* of the distribution defined by (a), but not an informative one. (c) An alternative *I-map* of the distribution defined by (a).

be continuous, in which case it might take a value from some real interval. We use capital letters, such as $X, Y, Z$, for variable names and lowercase letters $x, y, z$ to denote specific values taken by those variables. Sets of variables are denoted by boldface capital letters $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, and assignments of values to the variables in these sets are denoted by boldface lowercase letters $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

Given two random variables sets $\mathbf{X}, \mathbf{Y} \in \mathcal{X}$ we say that $\mathbf{X}$ is *independent* of $\mathbf{Y}$ if $P(\mathbf{X}, \mathbf{Y}) = P(\mathbf{X})P(\mathbf{Y})$. If this is the kind of distribution we have at hand then we can very compactly represent it. For example, if $X \in \{0, 1\}, \forall X \in \mathcal{X}$ and all are independent then instead of having to specify $2^N$ parameters (the probability of each possible assignment), we need to specify (or learn from observed data, as we shell see later) only $N$ such parameters to fully and accurately define the distribution. Of course in most cases this is not the case and such a strong independence assumption over our entire domain will probably result in a poor model for it. For example, if we make gene expression levels be binary variables (*i.e.*, going up/down), then assuming the expression of genes are all independent of each other will tell us very little about our domain. However, in many cases we can find a good representation of our domain that will still avoid the $2^N$ parameters. This is where the concept of *conditional independence* between random variables comes into play.

**Definition A.1.1:** We say that $\mathbf{X}$ is *conditionally independent* of $\mathbf{Y}$ given $\mathbf{Z}$ if

$$P(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) = P(\mathbf{X}|\mathbf{Z}) \quad \text{when } P(\mathbf{Z}) > 0$$

and we denote this statement by $P \models (\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$. ▮

We can get an intuition how this concept is going to serve us by viewing the example illustrated in Figure **??**(a). The graph describes a simple regulation network using binary variables to designate up/down expression levels or absent/present of external stimuli. In the system modeled here, the amounts of active transcription factor ($T_1$) can be turned up by either external stimuli $S_1$ or $S_2$ which are assumed to be independent. To account for the fact we do not model all biological entities involved in this system, there is also some chance of $T_1$ being up although $S_1, S_2$ are absent because of some other, unknown, reason. We know that if $T_1$ levels are up, it enhances the transcription of gene $G_1$, so we can expect $G_1$ to be up in much higher probability. In addition, the presence of $S_1$ usually decreases the amounts of $G_2$ in the cell. In this example we see since $T_1$ is the one responsible to directly control the transcription of $G_1$, than $G_1$ expression levels are independent of the presence of $S_1, S_2$ if we already know what are the levels of $T_1$. Similarly, if we do not observe the amounts of active $T_1$ molecules but are able to measure that $G_2$ amounts have decreased, then that would affect our belief about the levels of active $T_1$ and the result we should expect on $G_1$. But if we are able to measure $T_1$ directly, then the amounts of $G_2$ does not tell us anything new about the expected amounts of $G_1$. We now formalize the independence assumptions just described in the above example in the following way:

**Definition A.1.2:** Let $\mathcal{G}$ be a *Directed Acyclic Graph* (DAG) whose vertices correspond to random variables $\mathcal{X} = \{X_1, \ldots, X_N\}$. We say the $\mathcal{G}$ encodes a set of *Markov independence statements*: Each variable $X_i$ is independent of its non-descendants, given its parents in $\mathcal{G}$.

$$\forall X_i \ \ (X_i \perp \mathbf{NonDescendants}_{X_i} \mid \mathbf{Pa}_i)$$

and we denote the set of these statements as *Markov*$\mathcal{G}$. ▮

Figure **??** illustrates the concept of the Markov independence statements.

Using the rules of probability, it is important to note that we can derive *more* independence statements from *Markov*$\mathcal{G}$. For example, in Figure **??**(a), we can also say that $(T_1 \perp G_2 \mid S_1)$. This brings us to the question of when should we consider two graphs $\mathcal{G}_1, \mathcal{G}_2$ to be equivalent.

**Definition A.1.3:** We say that $\mathcal{G}_1$ and $\mathcal{G}_2$ are *independence equivalent* if

$$\mathit{Markov}\mathcal{G}_1 \Leftrightarrow \mathit{Markov}\mathcal{G}_2$$

That is, *Markov*$\mathcal{G}_1$ and *Markov*$\mathcal{G}_2$ imply they same set of independencies. ▮

(**?**) offers an efficient method for testing whether two structures belong to the same equivalence class. For us, the concept of equivalence class is specifically important because it shows why we have to be careful in how we constrain and interpret our models. It formalizes the fact that two very different graph structure can encode exactly the same set of independence assumptions.

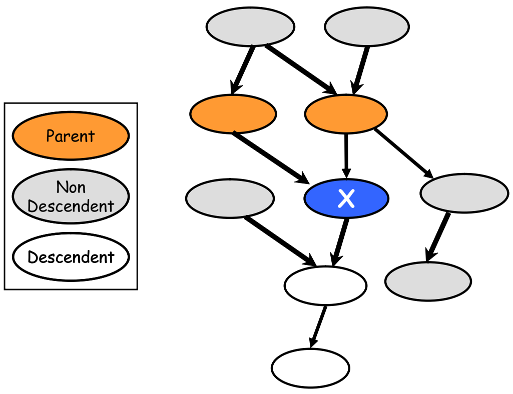


Figure A.2: Illustrative example of the Markov independence statements. $X$ is independent of all its non-descendents nodes in the graph $\mathcal{G}$, given its parent nodes. In contrast, $X$ is not unconditionally independent of any of its descendent nodes.

just as equivalence class defines the relations between two graph structures, we also need to define the relation, in terms of independence statements, between a graph structure $\mathcal{G}$ and the actual distribution we are trying to model.

**Definition A.1.4:** We say that the graph $\mathcal{G}$ is an *Independence map* (*I-map*) of the distribution $P$ over the random variables $\mathcal{X} = \{X_1, \ldots, X_N\}$ if

$$P \models \mathit{Markov}\mathcal{G}$$

▮

The *I-map* relation only implies that $P$ satisfies all of the independencies that can be derived from *Markov*$\mathcal{G}$. In a sense, $P$ does not "contradict" what we can conclude about it from reviewing the structure of $\mathcal{G}$. The *I-map*ness relation between $\mathcal{G}$ and $P$ is also important as it allows us to decompose $P$ into a product of terms. Formally, the *factorization theorem* states that

**Theorem A.1.5:** (**?**) $\mathcal{G}$ is an *I-map* of $P$ if and only if $P$ can be written as

$$P(X_1, \ldots, X_N) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}_i^{\mathcal{G}}) \tag{A.1}$$

where $\mathbf{Pa}_i^{\mathcal{G}}$ are the parents nodes of the variables $X_i$ in $\mathcal{G}$.

This theorem is a direct consequence of the chain rule of probabilities and properties of conditional independence.

Now,with all the definition given so far, we are ready to formally define a Bayesian network model.

**Definition A.1.6:** (**?**) A *Bayesian network* $\mathcal{B} = \langle \mathcal{G}, \theta \rangle$ is a representation of a joint probability distribution over a set of random variables $\mathcal{X} = \{X_1, \ldots, X_N\}$, consisting of two components: A *directed acyclic graph* $\mathcal{G}$ whose vertices correspond to the random variables and that encodes the *Markov independence assumptions Markov*$\mathcal{G}$; a set of parameters $\theta$ that describe a *conditional probability distribution* (CPD) $P(X_i \mid \mathbf{Pa}_i)$ for each each variable $X_i$ given parents in the graph $\mathbf{Pa}_i$. In addition, we require that $\mathcal{G}$ is an *I-map* of the distribution $P$ represented by the Bayesian network. ▮

$\mathcal{G}$ and $\theta$ define together a unique probability distribution that can be written in the form given by Eq. (**??**). This is called the *chain rule for Bayesian networks*.

Although we just defined our probabilistic graphical model, we are still missing a very important aspect of how the model $\mathcal{B}$ relates to the underlying distribution $P$ and how we can use the first to learn about the later. For this, let us return to the simple example of the regulatory network of Figure **??**(a). First, note that a full graph, as the one given by Figure **??**(b), is always an *I-map* of *any* probability distribution. However, such a graph would be of little use for us since it does not capture any interesting independencies in the domain we are interested in. Also note such a model would involve $2^N$ parameters we need to estimate, hence there is no real added value in our model. The fine point lies in the way we defined the *I-map* relation between $\mathcal{G}$ and $P$, which is not symmetrical. Any independence we can conclude from $\mathcal{G}$ has to be satisfied by $P$, but $P$ may include *additional* independencies not captured by $\mathcal{G}$. A full graph is simply the extreme case of this. A more symmetrical definition in terms of the implied independencies is the *P-map*. We say that $\mathcal{G}$ is a *Perfect Map* (*P-map*) of $P$ when $\mathcal{G}$ is an *I-map* of $P$ *and* $P$ does not satisfy any additional independencies that are not encoded by $\mathcal{G}$. In many cases, however, this definition turns out to be to stringent. Instead, we use the more relaxed notion of *minimal* I-map:

**Definition A.1.7:** Graph $\mathcal{G}$ is a *minimal* I-map of the distribution $P$ over the random variables $\mathcal{X} = \{X_1, \ldots, X_N\}$ if it is an *I-map* of $P$ and the removal of any edge from it renders it a non-*I-map* of $P$. ▮

When we learn a probabilistic graphical model for a given domain, we strive to learn the most compact representation of it, as captured by this formal definition of *minimal* I-map . Then the advantages of our model are, in a sense, maximized. As an example, consider the joint probability distribution $P(S_1, S_2, T_1, G_1, G_2)$ over our regulatory network example. If we had to use just the chain rule of probability, without any additional independence assumptions we would get:

$$P(S_1, S_2, T_1, G_1, G_2) = P(S_1)P(S_2|S_1)P(G_2|S_1, S_2)P(T_1|S_1, S_2, G_2)P(G_1|T_1, G_2, S_1, S_2)$$

This representation requires $1 + 2 + 4 + 8 + 16 = 31$ parameters. But when we encode relations between the entities into our graph representation and take into account the resulting conditional independencies we can write:

$$P(S_1, S_2, T_1, G_1, G_2) = P(S_2)P(S_1)P(G_2|S_1)P(T_1|S_1, S_2)P(G_1|T_1)$$

which only requires $1 + 1 + 2 + 4 + 2 = 10$ parameters. More generally, if $\mathcal{G}$ is defined over $N$ binary variables and its indegree (*i.e.*, maximal number of parents) is bounded by $K$, then instead of representing the joint distribution with $2^N - 1$ independent parameters we can represent it with at most $2^K N$ independent parameters.

The above discussion illustrated the power of Bayesian networks in terms of their ability to easily encode (in)dependency relations between the entities in the domain via the graph structure. This gives this type of models the power to efficiently represent complex distributions over domains that involve many random variables with a network of local interactions between them. We will later show how we can effectively learn such models for domains of interest from observed data. The resulting models enables us to infer about the expected values of entities in our domain (*e.g.*, the expression level of a gene given the external stimuli condition) and the relations between them via the graph structure we learn. However, a common mistake when first introduced with the concept of Bayesian networks is to mix the graph structure learned $\mathcal{G}$, with *causal* relations between the entities, and to treat this it as the *unique* structure that matches the underlying distribution. We therefore briefly review the concepts of causality and the *I-map* relations between $\mathcal{G}$ and $P$ in the next section.

## A.1.1 Bayesian Networks and Causality

To understand the relation between Bayesian networks and causality we first note that the formal definition we gave for a Bayes net only required $\mathcal{G}$ to be an *I-map* of the underlying distribution $P$ and we already showed that there can be many graphs that are *I-map* of $P$, including the full graph. Even the later definition we gave of a *minimal* I-map , aimed to formalize the concept of an efficient and compact representation of $P$, should not be confused with uniqueness or causality. In many cases the same distribution $P$ may have several, very different, *minimal* I-map representations of it. For example, Figure **??**(c) gives an alternative *minimal* I-map for the same distribution captured by Figure **??**(a). This illustrates that not only a *minimal* I-map is not necessarily *unique*, but that the dependency structure defined by $\mathcal{G}$ does not necessarily imply causality.

To treat the notion of causality more formally we say that given two random variables $(X, Y)$ $X$ *causes* $Y$, implies that manipulating the value of $X$ affects the value of $Y$. If the causal relation is reversed, *i.e.*, $Y$ is a cause of $X$, then manipulating $X$ will not affect $Y$. Thus, although $X \rightarrow Y$ and $X \leftarrow Y$ are equivalent Bayesian networks, they are not equivalent *causal networks* ((**?**)). A

| | $X = 0$ | $X = 1$ | $X = 2$ |
|---|---|---|---|
| $C = 1$ | 0.1 | 0.2 | 0.7 |
| $C = 2$ | 0.1 | 0.7 | 0.2 |
| $C = 3$ | 0.8 | 0.15 | 0.05 |
| $C = 4$ | 0.8 | 0.15 | 0.05 |
| $C = 5$ | 0.8 | 0.15 | 0.05 |

(a) full CPT

| | $X = 0$ | $X = 1$ | $X = 2$ |
|---|---|---|---|
| $C = 1$ | 0.1 | 0.2 | 0.7 |
| $C = 2$ | 0.1 | 0.7 | 0.2 |
| $C = *$ | 0.8 | 0.15 | 0.05 |

(b) default CPT

Figure A.3: Example of two representations of the same conditional distribution $P(X \mid C)$.

*causal network* is represented mathematically in a form similar to a Bayesian network *i.e.*, a model made of a directed acyclic graph (DAG) where each node represents a random variable along with a local probability model (CPD) for each node. The difference is in the stricter interpretation of the edges: the parents of a variable are its *immediate causes*. Causal interpretations for Bayesian Networks have been proposed in several works (**??**).

In this dissertation we will not attempt to use causal interpretations of our learned models. The models we develop are usually constrained so that the structure we learn for them matches our prior biological knowledge about the domain. For example, we can only learn that the expression profile of the gene is dependent on the experimental condition, not the other way around. Although this makes the interpretation of the model much simpler, given the above discussion and the fact we always learn from limited amount of data , we still need to be careful about our certainty of specific (in)dependency relations we identify. This means we need to develop methods to assess the statistical significance of our qualitative results (related to the graph structure) as well as quantitative results (*e.g.*, expression profile prediction). See (**?**) for more about this in the context of biological domains.

## A.2 Representing Dependencies: the CPD

We now turn to discuss the second component of the Bayesian network model, the parameterization component, $\theta$. As we saw in the definition of the Bayesian network model, this component is made of a set of conditional probability distribution (CPDs), that specify the probability distribution for each variable given the assignment to its direct parents in $\mathcal{G}$. We denote these by $P(X_i \mid \mathbf{Pa}_i)$. The actual probability distributions we use in the model can have any form, but we are interested in representations that will serve our aims well. We need to have conditional distributions that fit reasonably well the empirical distributions we observe in the data yet can be efficiently learned from data and used efficiently to perform inference tasks. We now review common representations for CPD's used in models we develop.

We start with the simple case where both the variable $X_i$ and its parents $\mathbf{Pa}_i$ are *discrete*. In this case the most general representation for a CPD is a *conditional probability table* (CPT), as the one we have in Figure **??**(a). The values in the table are the probability of $x_i$ given the assignment $\mathbf{pa}_{X_i}$ so that the row index corresponds to $x_i$ and the column to $\mathbf{pa}_{X_i}$ . This general representation can describe any discrete conditional distribution. If $X_i$ is a *continues* variable and $\mathbf{Pa}_{X_i}$ are discrete then we need the values in the CPT to describe the probability density function over $X_i$ given the (discrete) assignment to $\mathbf{Pa}_i$. common choice for the density function is the *Gaussian* distribution. In this case we have $X_i \sim N(\mu_{\mathbf{pa}_i}; \sigma^2_{\mathbf{pa}_i})$, so each entrance in the table specify the matching values of the parameters governing the Gaussian distribution, namely the mean and the variance of it.

There are of course numerous other representations for the CPDs, parametric as well as non parametric ones. The main problem with the table CPT we just described is the number of parameters involved and the fact we actually have to infer all of them from the data. Moreover, in our domain of interest many may simply be non relevant. For example, if $X$ is the expression levels of a gene, characterized by a gaussian distribution, and $\mathbf{Pa}_i$ include the experiment condition $E$ and the activity level of some transcription factor $\mathcal{T}$, then $\mathcal{T}$ would probably affect $X_i$ only under specific experimental conditions *i.e.*, only for specific assignments $e \in \mathit{Val}(E)$. This means that all other conditions belonging to $\mathit{Val}(E)$ can have the same $\mu, \sigma$ parameters, whether the activation levels of $\mathcal{T}$ are. This kind of phenomena can be captured with *context specific independence* (CSI) representations for CPDs. Two such CSI CPD we use are the *Tree CPD* and the *Default Table CPD*. We review these now.

### A.2.1 CSI - Tree CPD

A *tree-structured CPD* (**?**), (**?**) for a variable $X$ is a rooted tree where each *node* in the tree is either a *leaf* or an *interior node*. Each interior node is labeled with a test of the form $U = u$ where $U \in \mathbf{Pa}_i$. Each of these interior nodes has two outgoing *arcs* to its children, corresponding to the outcomes of the test (true or false). Each of the leaves is associated with a distribution over the possible values of $X$. For instance, in the example we just gave where $X$ is the expression level of a gene, an internal node of the tree might correspond to the question weather the experiment was conducted in the mitotic cell cycle phase *i.e.*, $E = \mathit{mitotic}$. Only in the branch that corresponds to the answer *true* we would have a following node in tree testing whether $\mathcal{T}$ is active in the cell.

### A.2.2 CSI - Default Tables

Another CSI representation form is that of *default tables* (**?**). As an example of their usage let's return to the CPT in Figure **??**(a). Assume now $X_i$ is the number of binding site of some transcription factor $\mathcal{T}$ that occur in a gene's promoter. Also assume that the parent variable $C = \mathbf{Pa}_i$ in this case is the index of the regulatory program that might use $\mathcal{T}$ to control expression of genes. Naturally, most regulatory programs do not facilitate $\mathcal{T}$. As a result, the distribution over the number of occurrences of $\mathcal{T}$'s binding site in genes controlled by these program tend to be around zero, with very similar values. The only regulatory programs that really use $\mathcal{T}$ are the two first ones, with one usually involving two of $\mathcal{T}$ binding sites and other only a single site. In such a case, we can much more compactly represent the conditional distribution with a default table as in Figure **??**(b). In general, default tables are suitable for cases where the probability distribution of $X_i$ given $\mathbf{Pa}_i$ has some default value excluding a small number of assignments $\mathbf{pa}_i$.

## A.3 Inference

A fundamental task in any graphical model is that of inference. That is, we want to be able to answer general queries of the form $P(\mathbf{X} \mid \mathbf{Z})$ where $\mathbf{X}, \mathbf{Z} \subset \mathcal{X}$, as efficiently as possible. Assume for example, that in the simple regulatory model of Figure **??**(a) we want to evaluate the overall probability of gene $G_2$ to be found highly expressed in the cell. With brute force, summing over all the possible assignments to all other variables in our domain we get:

$$P(g_1 = high) = \sum_{s_1, s_2, g_2, t_1} P(s_1, s_2, g_2, t_1, g_1)$$

We can get exactly the same result more efficiently by utilizing the decomposition of the joint probability which results in

$$P(c) = \sum_{t} P(g_1|t_1) \sum_{s_1} P(s_1) \sum_{s_2} P(s_2) P(t_1|s_1, s_2) \sum_{g_2} P(g_2|s_1)$$

which is significantly more efficient: Each internal summation typically goes over the different assignments of only few variables, eliminating one of them. The computed factors are then reused in the next outer summation. While the complexity of the brute force summation is $O(2^N)$ (in the case of binary valued variables), this decomposed summation costs only $O(N2^C)$, where $C$ is roughly the number of possible assignments to the largest family in the network. The larger the network is, the bigger the advantage of this procedure. Furthermore, the factors computed during this summation can be reused in the computation of other queries. This procedure of *variable elimination* (summation) is the basis of all exact inference methods.

There are methods that enable answering multiple queries at the cost of two variable elimination computations. These include *Bucket Elimination* (**?**) and *Junction Trees* (*e.g.*, (**?**)), both of which widely used as techniques to perform *exact* inference. The key to making these exact inference methods as efficient as possible lies in the *order* in which the variables are eliminated because this can have a crucial effect on the size of $C$ in the exponent of the above expression. Finding a technique that will find a good order for variable elimination has been the focus of numerous works. In any case, it is important to note that whatever the method you use for variable elimination, inference in Bayesian networks is in general (excluding tree structured networks) NP-hard (**?**).

In many cases, exact inference using variable elimination is simply not practical. For example, in our domain of modeling regulatory mechanisms, we might have many transcription factors and numerous experiment attributes all affecting the expression profile of the gene. This would cause the size of $C$ to be such that it would render the exact inference methods non feasible. In such cases we need to resort to *approximate* inference methods. These methods have been the subject of extensive research in the past few years. Roughly, we can divide approximate inference algorithms into three classes. The first class is usually termed *Particle based* methods. In a nutshell, these methods generate a set of instantiations, often called *particles*, to all or some of the variables in the domain $\mathcal{X}$. Then they use these instantiations to approximate the specific probability of interest $P(X_i \mid \mathbf{Pa}_i)$. This class includes methods such as *rejection sampling* and *Gibbs sampling* (see (**?**) for an overview of inference sampling techniques). The second class is known as *variational approximation* methods (**?**). These methods define a class of distribution $\mathbf{Q}$ that are simplifications of the real distribution $P$ we need to model (*e.g.*, a class of distributions were some variables are assumed independent thus reducing the effective size of $C$ when performing exact inference). They search for a particular distribution $P_Q \in \mathbf{Q}$ that is a good approximation of $P$ and then use it instead of $P$ to perform inference tasks. The last class of approximate inference methods are *Belief Propagation*. In general, this method involves first converting the original Bayesian network into an undirected graph (similar to the clique tree used for exact inference) and then performing local "message" passing between neighboring nodes in the resulting graph. The 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). However, in many cases albeit the fact the network is not singly connected this methods does converge and the results prove to be reasonable approximation to the correct posterior. see (**?**) and reference therein for more on this.

## A.4 Learning Bayesian Networks with Complete Data

A probabilistic graphical model represent a distribution $P$ over some domain $\mathcal{X}$ via its two components $\mathcal{G}, \theta$ . Given samples from an (unknown) distribution $P^*$, we want to *learn* a model that best describes $P^*$ *i.e.*, we want to learn a graph structure $\mathcal{G}$ which is a *minimal* I-map of $P^*$ along with the matching CPT parameters $\theta$. Of course in reality our ability to match $P^*$ by a learned model may be limited. In some cases $P^*$ cannot be captured by a Bayesian network, in others our initial choice of the CPT representation may be of limit (*e.g.*, we use a Gaussian distribution for some variable while the underlying distribution is not so). Moreover, when we handle problems from the real worlds we are almost always limited by the amount of samples we get from $P^*$. Formally, given a finite set of samples (the *training set*) $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$, assumed to be independently drawn from $P^*$, we need to learn a model $\mathcal{B} = \langle \mathcal{G}, \theta \rangle$ that best approximates $P^*$. As we shell see, because our data set is *finite* we need to resort to various techniques to avoid poor learning of $P^*$ arising because for reasons such as *over-fitting* (see below). If we succeed in this task we can then exploit the learned model to identify qualitative features in our domain via its structure (*e.g.*, which transcription factor affect which gene) and infer quantitative results (*e.g.*, predict gene expression profiles).

We start by assuming that the correct graph structure $\mathcal{G}$ is already known and given to us as input. This leaves us with the simplified task of just learning the CPT parameters. Of course this is not a realistic assumption, especially in our domain, but we shell see how the algorithms we discuss now for parameter learning are used when the structure is unknown too.

## A.4.1 Parameter Learning

### Maximum Likelihood Estimation

As we just stated, we now assume we know the correct model structure $\mathcal{G}$ that matches the underlying distribution $P^*$. For the task of learning the correct parameter $\theta$ the first thing we need to define is a "score" or a measure of quality for a candidate set of parameter. An intuitive and commonly used measure is the *likelihood function*:

**Definition A.4.1:** The *likelihood function*, $L(\theta : \mathcal{D})$, is the probability of the independently sampled instances of $\mathcal{D}$ given the parameterization $\theta$

$$L(\theta : \mathcal{D}) = \prod_{m=1}^{M} P(\mathbf{x}[m] \mid \theta) \tag{A.2}$$

▮

In the *Maximum Likelihood Estimator* (MLE) approach, we favor the parameters $\hat{\theta}$ that maximize the likelihood of the data:

$$\hat{\theta} = \max_{\theta} L(\theta : \mathcal{D}) \tag{A.3}$$

The intuition behind choosing the MLE as the estimator for $\theta$ is simple: We choose the parameters that make the observed data more likely to be generated. The theory behind this intuition is also sound: In many cases the MLE is both *consistent* and *unbiased.* Consistency means it converges (in probability) to the correct parameters of $P^*$ and unbiased suggests that the estimator mean over finite sample sets is equal to the true value.

But how do we find the MLE in the case of Bayesian networks? Turns out that the Bayesian network representation offers a decomposition of the optimization task we just set in Eq. (**??**):

$$\begin{aligned} L(\theta : \mathcal{D}) &= \prod_{m=1}^{M} P(\mathbf{x}[m] \mid \theta) = \prod_{m=1}^{M}\prod_{i=1}^{N} P(x_i[m] \mid \mathbf{pa}_i[m] : \theta_{X_i|\mathbf{Pa}_i}) = \\ &= \prod_{i=1}^{N}\left[\prod_{m=1}^{M} P(x_i[m] \mid \mathbf{pa}_i[m] : \theta_{X_i|\mathbf{Pa}_i})\right] = \prod_{i=1}^{N} L_i(\theta_{X_i|\mathbf{Pa}_i} : \mathcal{D}) \end{aligned}$$

Where we exploited the decomposition property of Eq. (**??**). We use here $\theta_{X_i|\mathbf{Pa}_i}$ to denote the parameters that encode the conditional probability distribution of $X_i$ given its parents $\mathbf{Pa}_i$. The term:

$$L_i(\theta_{X_i|\mathbf{Pa}_i} : \mathcal{D}) \equiv \prod_{m=1}^{M} P(x_i[m] \mid \mathbf{pa}_i[m] : \theta_{X_i|\mathbf{Pa}_i}) \tag{A.4}$$

is called the *local likelihood function* for $X_i$ and its exact form depends on our chose of CPD representation (Section **??**).

The MLE for each of the local likelihood function parameters can often be computed in closed form. For the table CPDs discussed in Section **??** we can further simplify the local likelihood by rearranging the order of the product in the above formula in the following way. For the variable $X$ with its parents $\mathbf{Pa}$ we have a parameter $\theta_{x|\mathbf{u}}$ for each combination of $x \in \mathit{Val}(X)$ and $\mathbf{pa} \in \mathit{Val}(\mathbf{Pa})$. Given the observed data, we then group together all the instances in which $X = x$ and $\mathbf{Pa} = \mathbf{pa}$. We denote by $S[x_i, \mathbf{pa}_i]$ the number of these instances, and by rearranging the order of the product we can write:

$$L_i(\theta_{X|U} : \mathcal{D}) = \prod_{\mathbf{pa}_i \in \mathit{Val}(\mathbf{Pa}_i)} \prod_{x \in \mathit{Val}(X)} \theta_{x_i|\mathbf{pa}_i}^{S[x_i,\mathbf{pa}_i]} \tag{A.5}$$

By optimizing this local likelihood function under normalization constraints, we obtain the intuitive result which is also the maximal likelihood estimators (MLE) for case of multinomial table CPD:

$$\hat{\theta}_{x|\mathbf{pa}_i} = \frac{S[x_i, \mathbf{pa}_i]}{\sum_{x_i} S[x_i, \mathbf{pa}_i]} \tag{A.6}$$

This is an intuitive result because it basically means we count the number of observed values and divide by the total number of observations, for each of the values $X_i = x_i$ given a specific value assignments of the parents $\mathbf{Pa}_i = \mathbf{pa}_i$.The counts $\{S[x_i, \mathbf{pa}_i]\}\forall(x_i, \mathbf{pa}_i)$ are what is termed the *sufficient statistics* for the case of a multinomial distribution. The sufficient statistics summarize all the relevant information from the observed data. As a result, any dataset with the same sufficient statistics will give rise to exactly the same model. For the case of Gaussian CPD the sufficient statistics are the number of samples ($M$) along with the first and second moments for the distribution over $X_i$ given its parents ($E[X_i \mid \mathbf{Pa}_i = \mathbf{pa}_i], E[X_i^2 \mid \mathbf{Pa}_i = \mathbf{pa}_i]$)

### Bayesian Estimation

The MLE approach, although very intuitive and easily implemented, has several drawbacks. First, it does not leave room for incorporating prior knowledge about the problem, which in our biological domain may be very significant. The second problem is the fact ML estimators tend to perform badly in the presence of limited and noisy data. This can be illustrated by a simple example. Say we have a fair dice and we want to estimate the probability of it falling on each side. If we get 3 rolls of the dice as samples, just computing the standard ML estimator would necessarily give us at least 3 values with probability zero. Not only is our estimator limited by the fact we had 3 samples and 6 possible outcomes, but given the fact we are actually rolling a dice (*i.e.*, a prior knowledge about the domain) we can argue that having a side with probability exactly zero is not very likely.

Many real life problems involve a lot of variables and limited, noisy, data. We therefore need a method that is more robust to data fluctuations and allows prior knowledge to be incorporated. Intuitively, we would like the method that incorporates prior knowledge to have the following characteristics. We could use it to define a prior that is "strong", meaning we have a strong belief about how the observations should look like (in our dice case - how the multinomial distribution over the six possible outcomes should look like) or define a "weaker" prior. The more strong our prior the more confident we would be about our estimators if indeed they seem to match the observations. Of course this would come with the price that if our prior belief is wrong, then the stronger our prior is the more it would take to convince us we are actually wrong. We would also want to have the ability to define priors which are "informative", meaning we believe certain type of outcomes are more likely then others (*e.g.*, we believe a multinomial distribution that favors high numbers as the result of rolling the dice is more plausible, because we know the dealer in the casino); or define "non informative" priors in which case we have no tendency to a priori believe some distributions are more favorable then others (but it also means we a-priori believe *any* outcome is plausible) . Last but not least, an important characteristic for any estimator is *consistency i.e.*, that given enough samples it would finely converge to the correct parameters. We turn to describe the *Bayesian approach* for parameter estimation that formalize the concept of prior as we just described.

In the Bayesian approach we have to define some *prior distribution* over all possible parameters of our model $P(\theta)$. Then, following our observations, we update our beliefs and compute the *posterior* distribution over $\theta$:

$$P(\theta \mid D) = \frac{P(\mathcal{D} \mid \theta)P(\theta)}{P(\mathcal{D})} \tag{A.7}$$

The term $P(\mathcal{D})$, named the *marginal likelihood*, averages the probability of the data over all possible parameter assignments:

$$P(\mathcal{D}) = \int P(\mathcal{D} \mid \theta) P(\theta) d\theta \tag{A.8}$$

To estimate the probability of a new sample $X[M+1]$ we average over all possible parameters of the distribution according to their posterior probability:

$$P(X[M+1] \mid \mathcal{D}) = \int P(X[M+1] \mid \mathcal{D}, \theta) P(\theta \mid \mathcal{D}) d\theta \tag{A.9}$$

Note how the above formulation catches the intuitive characteristics of prior we start from. The definition of $P(\theta)$ can either favor certain parameter settings (*i.e.*, an informative prior) or not (non informative). It can also be sharply picked around certain values (strong prior) or only mildly picked around the same values (weaker prior). Also, note that the only factor in Eq. (**??**) which is not constant is $P(\mathcal{D} \mid \theta)$ which corresponds to the ML estimator. This helps to see why the Bayesian estimator is also *consistent* in many cases (see (**?**) for more details on this).

The above description still left the following question open: What kind of representation should we choose for this prior? Obviously, it should be able to express our belief about the distribution's parameters formally in a wide range of settings, but it should also be easy to handle, in terms of computation efficiency. A common strategy is to choose a prior with a suitable form so the posterior belongs to the same functional family as the prior. These are called *conjugate* priors. For a multinomial distribution the conjugate prior has the form of a *Dirichlet* distribution (**?**). The Dirichlet distribution is parameterized by a set of *hyper-parameters* $\alpha_{x_i^1|\mathbf{pa}_i}, \dots \alpha_{x_i^K|\mathbf{pa}_i}$. Each such hyper-parameter corresponds to a matching $x_i^j \in \mathit{Val}(X_i)$. The Dirichlet distribution is specified by:

$$P(\theta_{X_i|\mathbf{pa}_i}) = \mathit{Dirichlet}(\alpha_{x_i^1|\mathbf{pa}_i}, \dots \alpha_{x^K|\mathbf{pa}_i}) \propto \prod_j \theta_{x_i^j|\mathbf{pa}_i}^{\alpha_{x^j|\mathbf{pa}_i}-1} \tag{A.10}$$

The posterior distribution has the same form:

**Proposition A.4.2:** *(**?**) If $P(\theta_{X_i|\mathbf{pa}_i})$ is $\mathit{Dirichlet}(\alpha_{x_i^1|\mathbf{pa}_i}, \dots, \alpha_{x_i^K|\mathbf{pa}_i})$ then the posterior $P(\theta_{X_i|\mathbf{pa}_i} \mid \mathcal{D})$ is $\mathit{Dirichlet}(\alpha_{x_i^1|\mathbf{pa}_i} + S[x_i^1, \mathbf{pa}_i], \dots, \alpha_{x_i^K|\mathbf{pa}_i} + S[x_i^K, \mathbf{pa}_i])$ where $S[x_i^j, \mathbf{pa}_i]$ is the sufficient statistics derived from $\mathcal{D}$.*

Thus, the hyper-parameters $\alpha_{x_i}$ play a similar role to the empirical counts and are often referred to as *imaginary counts* and their sum $M' \equiv \sum_{x_i} \alpha_{x_i|\mathbf{pa}_i}$ is called the *effective sample size* of the prior. That is, using a Dirichlet prior with the above hyper-parameters is equivalent to having seen, prior to $\mathcal{D}$, $M'$ other samples where in $\alpha_{x_i^j|\mathbf{pa}_i}$ of them are $X_i = x^j$. This intuition is also reflected in the resulting form for the likelihood of a new sample:

$$\hat{\theta}_{x_i|\mathbf{pa}_i} \equiv P(X_i[M+1] = x_i \mid \mathbf{Pa}_i[M+1] = \mathbf{pa}_i, \mathcal{D}) = \frac{\alpha_{x_i|\mathbf{pa}_i} + S[x_i, \mathbf{pa}_i]}{\sum_{x_i'} \alpha_{x_i'|\mathbf{pa}_i} + S[x_i', \mathbf{pa}_i]} \tag{A.11}$$

We also see how the notion of a “stronger” prior is formalized here, by increasing $M'$ and that an informative prior is such that assigns more imaginary counts on specific values.

For continues attributes assumed to be normally distributed with mean $\mu$ and precision $\tau = \sigma^{-1}$ a common choice of prior over $\mu, \tau$ is the *Normal-Gamma* distribution. According to this distribution we have $\mu$ distributed normally given $\tau$:

$$P(\mu \mid \tau) \sim \mathcal{N}(\mu_0, \tau')$$

Where $\tau' = \lambda_0 \tau$ is the precision and $\mu_0, \lambda_0$ are two of the Normal-Gamma four parameters. The other two parameters are $(\alpha_0, \beta_0)$, determining the shape of the prior Gamma distribution over $\tau$:

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

Where $\Gamma(x)$ is the Gamma function defined as: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-x} dt$. The Normal Gamma is a conjugate prior for the Normal distribution so that the posterior also has a Normal Gamma form:

$$P(\theta_{X_i \mid \mathbf{pa}_i} \mid \mu_0, \lambda_0, \alpha_0, \beta_0, \mathcal{D}) = NG(\mu_1, \lambda_1, \alpha_1, \beta_1)$$

Where:

$$\begin{aligned}
\lambda_1 &= \lambda_0 + S[\mathbf{pa}_i] \\
\mu_1 &= \frac{\lambda_0 \mu_0 + S[\mathbf{pa}_i] E[X_i \mid \mathbf{pa}_i]}{\lambda_1} \\
\alpha_1 &= \alpha_0 + \frac{S[\mathbf{pa}_i]}{2} \\
\beta_1 &= \beta_0 + \frac{1}{2} S[\mathbf{pa}_i] E[X^2 \mid \mathbf{pa}_i] - \frac{1}{2} S[\mathbf{pa}_i] E[X \mid \mathbf{pa}_i] + \frac{S[\mathbf{pa}_i] \lambda_0 (E[X \mid \mathbf{pa}_i] - \mu_0)}{2\lambda_1}
\end{aligned}$$

As a final note, we should point out that although we used notation for random variables in a Bayesian network *i.e.*, $X_i, \mathbf{Pa}_i$ the above equations handled only *separate* random variables. To have the above equations correct for full Bayesian networks, we need to introduce a few more definitions.

**Definition A.4.3:** (**?**) A parameter prior $P(\theta)$ for a Bayesian network is said to satisfy *parameter independence* if it can be written as

$$P(\theta) = \prod_{i=1}^{n} \prod_{\mathbf{pa}_i} P(\theta_{x_i \mid \mathbf{pa}_i})$$

The decomposition according to the network structure is called *global parameter independence*.

The further decomposition according to the values $\mathbf{pa}_i$ is called *local parameter independence.* ▮

Now, assuming parameter independence, we can assign an independent prior distribution $\theta_{X_i|\mathbf{pa}_i} \sim \mathit{Dirichlet}(\alpha_{x_i^1|\mathbf{pa}_i}, \ldots, \alpha_{x_i^K|\mathbf{pa}_i})$ for each variable and each instantiation of its parents in the network and Eq. (**??**) holds for each of these.

## A.4.2 Structure Learning

In the previous section we assumed the structure of the Bayesian network represented by $\mathcal{G}$ is given to us and we just need to handle the task of learning the quantitative component of the network, $\theta$. In real life problems this is not usually the case. We now turn to focus on learning the structure of the Bayesian network model. There are two basic approaches for handle this task. In the *constrained based* approach we use independence tests over the variables in our domain. The results of these tests defines a set of independence statements that serve as constraints for the model structure. At the next step we learn a structure so as to best match this set of constraints. Although efficient, this approach is sensitive to mistakes done at the independence tests at the first stage. The second approach termed *score based*, is of common use and is the one we apply in the works presented here too. This approach formulates structure learning as an *optimization* problem. We first define a *score* over all possible structures and then search for the structure that maximize the score.

Intuitively, a score for evaluating the fit of a model structure to the observed data should have several properties to make it suitable for the task. First, it should follow the *Occam's Razor* principle: if two models achieve the same likelihood then the simpler one will receive a higher score. Second, it should give the same result for models that are *independence equivalent* as given by Definition **??**. It is also highly important to have the *decomposition* property: Given a complete dataset it should decompose according to the network structure. Without this property we will not be able to perform efficient computations for the cost of local structure changes. This is what stands in the hart of many algorithms for learning Bayesian networks hence its importance. Finally, we would like the score to be *consistent i.e.*, if $\mathcal{G}^*$ is the generating model, as the number of samples grows to infinity, the score should prefer $\mathcal{G}^*$ (or some equivalent model) to any other structure.

### Structure Scores

We start our review of structure scores with the same approach we used for parameter learning, the *Bayesian* approach. According to it, we do not use a specific structure but rather define some prior $P(\mathcal{G})$ over all possible structures, which we update according to data we get:

$$\mathrm{Score}(\mathcal{G} : \mathcal{D}) \quad = \quad \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

The above equations contains two elements. The first term $P(\mathcal{D} \mid \mathcal{G})$ is the the *marginal likelihood* of the structure given the data and it averages the probability over all possible parameterizations of $\mathcal{G}$:

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathcal{G}, \theta) P(\theta \mid \mathcal{G}) d\theta$$

It is important to note, in retrospect of the desired properties of structure scores we listed, that this term is responsible for regularizing the score for more complex models. A model which is unnecessarily complex given the sample data we have will assign prior over many (non relevant) parameterizations that will decrease its score while a model which is to simple to fit the data well will not be able to get a high likelihood for any parameterizations in the whole range of the integral.

An important property for parameters prior $P(\theta \mid \mathcal{G})$ is *parameter modularity*:

**Definition A.4.4:** (**?**) A parameter prior satisfies *parameter modularity* if for any two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, if $\mathbf{Pa}_i^{\mathcal{G}_1} = \mathbf{Pa}_i^{\mathcal{G}_2}$ then:

$$P(\theta_{X_i|\mathbf{Pa}_i} \mid \mathcal{G}_1) = P(\theta_{X_i|\mathbf{Pa}_i} \mid \mathcal{G}_2)$$

∎

Priors that satisfy parameter modularity are called *factorized* priors (**??**). The reason for requiring this property lies in the following proposition:

**Proposition A.4.5:** *If the prior $P(\theta \mid \mathcal{G})$ satisfies both global parameter independence and parameter modularity then:*

$$P(\mathcal{D} \mid \mathcal{G}) = \prod_i \int_{\theta_{X_i|\mathbf{Pa}_i}} \prod_m P(x_i[m] \mid \mathbf{pa}_i[m], \theta_{X_i|\mathbf{Pa}_i}) P(\theta_{X_i|\mathbf{Pa}_i}) d\theta_{X_i|\mathbf{Pa}_i}$$

The second term in Eq. (**??**) is the prior over all possible structures $P(\mathcal{G})$. Since $P(\mathcal{D} \mid \mathcal{G})$ already has the property of preferring simpler structures over unnecessarily complex ones, $P(\mathcal{G})$ is sometimes taken to be uniform and thus omitted as a constant from Eq. (**??**). If not uniform, we need it to satisfy *Structure modularity*:

**Definition A.4.6:** A prior $P(\mathcal{G})$ satisfies *Structure Modularity* if it can be written in the form

$$P(\mathcal{G}) \propto \prod_i \rho(X_i, \mathbf{Pa}_i^{\mathcal{G}});$$

∎

This means the prior decomposes according to the families in the graph structure. Combined with the properties of $P(\theta \mid \mathcal{G})$ we just described this enables us to decompose the Bayesian score of Eq. (**??**) into local contributions each depending only on the sufficient statistics of $X_i$ and its parents

$$\text{Score}(\mathcal{G} : \mathcal{D}) = \sum_i \text{FamScore}(X_i, \mathbf{Pa}_i : \mathcal{D}) \tag{A.12}$$

To summarize this part, we see now how the combination of the prior properties we listed enable us to facilitate the decomposition of the Bayesian score. As we mentioned in the beginning of this section, this is highly important in order to have efficient algorithms that search for a high scoring network structure by constantly testing local structure changes.

For the case of Dirichlet priors and full table CPDs (**?**) have showed how the parameter estimation can have a closed form:

$$\text{FamScore}_{BDe}(X_i, \mathbf{Pa}_i : \mathcal{D}) = \sum_{\mathbf{pa}_i} \left[ \log \frac{\Gamma(\alpha_{\mathbf{pa}_i})}{\Gamma(\alpha_{\mathbf{pa}_i} + S[\mathbf{pa}_i])} + \sum_{x_i} \log \frac{\Gamma(\alpha_{x_i|\mathbf{pa}_i} + S[x_i, \mathbf{pa}_i])}{\Gamma(\alpha_{x_i|\mathbf{pa}_i})} \right] \tag{A.13}$$

where $\Gamma$ is the Gamma function, $\alpha_{\mathbf{pa}_i} = \sum_{x_i} \alpha_{x_i|\mathbf{pa}_i}$ and $S[\mathbf{pa}_i] = \sum_{x_i} S[x_i, \mathbf{pa}_i]$. This score is called *BDe* for *Bayesian Dirichlet equivalence* score. As the "equivalence" part in the name suggests, this score has another desired property we listed in the beginning of this section: model that belong to the same equivalent class as defined by Definition **??** would get the same score.

In many cases however, we are not so fortunate as to have such a closed form expression for $P(\mathcal{D} \mid Graph)$, even if it can be decomposed into separate families. *Approximations* for the Bayesian scores usually involve the likelihood of the data given the model's structure $\mathcal{G}$, the maximum a-posteriori (MAP) or maximum likelihood (ML) parameters ($\hat{\theta}$) and some complexity penalty term. The penalty typically involves some representation of the model's complexity ($Dim(\hat{\theta})$) and the number of samples we saw ($M$). A commonly used approximation for the Bayesian score is the *Bayesian Information Criterion* (BIC) score (**?**):

$$\text{Score}_{BIC}(\mathcal{G} : \mathcal{D}) = \log L_{\hat{\theta},\mathcal{G}}(\mathcal{D} : -) \frac{\log M}{2} Dim(\hat{\theta}) \tag{A.14}$$

Note that this score also retain the important property that it can be decomposed according to family structure in $\mathcal{G}$. We note that the BIC score is closely related to the *MDL (Minimum Description Length)* principle (**?**): it approximates the number of bits needed to describe the data using the model. This follows a well known principle from learning theory, that models which compress the data better, also have better generalization performance, or alternatively are closer to the generating distribution (for more about the theoretical relation between compression and generalization see (**?**)). It can be shown that as the size of the dataset grows to infinity, the BIC score converges to the

Bayesian score.

**Structure Search**

In the last section we described the Bayesian approach to score possible structures. The questions we are interested in, when it comes to learning a network structure, are either a global one - "What is the model that best describes my data?", or more local ones like: "Does $X_i$ directly depend on $X_j$?". If we carry the Bayesian approach to answer local structure questions such as this one it means that given the *posterior* $P(\mathcal{G} \mid \mathcal{D})$ over graph structures we evaluate the probability of a certain feature $f(\mathcal{G})$:

$$E[f] = \sum_{\mathcal{G}} f(\mathcal{G}) P(\mathcal{G} \mid \mathcal{D})$$

In practice however the above summation is generally not feasible and some methods have been suggested to approximate this expectation (**??**).

In many cases we focus just on the "global" structure question and try to find the best scoring structure. If we use the Bayesian score or approximations to it we can interpret this as using the *Maximum a posteriori* (MAP) scoring structure for answering quantitative or qualitative questions. In our case this would include quantitative prediction of expression profiles or qualitative relations of which transcription factor control each gene.

In some cases finding the best scoring model can be done efficiently. Specifically, Chow and Liu (**?**) have shown that learning the ML tree can be represented as the problem of finding a *maximum spanning tree* for which an efficient algorithm is known ( (**?**)). (**?**) have shown how this can be extended to some more specific settings with any decomposable score. Lamentably, finding the best scoring model for the general case is NP-hard (**?**). Thus, we resort to a heuristic methods for finding the best scoring structure. Most methods explore the space of possible network structures by applying small changes to some given structure(s), exploiting the decomposability of the score by the network structure. It is easy to think of these methods in the following way: Each (legal) network structure is a *node* in the search space they explore. Two such nodes are connected by an edge if we can move from one structure to another by performing an operation on the structure such as *Adding*, *Removing* or *Reversing* an edge. The set of these pre defined operations are termed *operators*. These search methods start with some initial structure (an initial node in the search space graph *e.g.*, the empty graph) and use the operators to traverse the space, searching for high scoring structures.

The most simple method to traverse the search space is probably *greedy hill climbing*. The algorithm is outlined in Algorithm **??**. According to this method, we simply move to the highest scoring neighbor and continue on doing this until a (local) optimum is reached. Since the search

```
Input  : $\mathcal{D}$    // training set
         $\mathcal{G}_0$   // initial structure
Output : A final structure G

$\mathcal{G}_{best} \leftarrow \mathcal{G}_0$
repeat
    G$\leftarrow \mathcal{G}_{best}$
    foreach Add,Delete,Reverse edge in G do
        $\mathcal{G}' \leftarrow$ ApplyOperator($G$)
        if $\mathcal{G}'$ is cyclic then continue
        if $\text{Score}(\mathcal{G}' : \mathcal{D}) > \text{Score}(\mathcal{G}_{best} : \mathcal{D})$ then
            $\mathcal{G}_{best} \leftarrow \mathcal{G}'$
        end
    end foreach
until $\mathcal{G}_{best} == G$
return $\mathcal{G}_{best}$
```

**Algorithm 1**: Greedy Hill-Climbing Structure Search for Bayesian Networks

space is generally not convex, avoiding local maxima is a crucial issue and many different methods have been proposed to help the greedy approach overcome this problem. Some methods involve "memory" of previous states/operations or performing several of these in parallel. For example In the *tabu list* method we keep track of the most recent operators applied and allows only steps which do not reverse these recently applied operators. Other method try to introduce stochasticity into the search process to overcome local optimum. With the *random restart* method, when we reach a local maximum we take some number of random steps, and then we restart our search process. Another method that is stochastic in nature uses a *simulated annealing* approach. We define a parameter that controls the probability of taking a random step instead of the best scoring one and as the search progresses the parameter is "cooled down" so the search convergence to a regular greedy search. To summarize, Finding methods which are both computationally efficient and offer a robust solution for overcoming local optimum in the search space is an active field of study (*e.g.*, (**?**) and references therein).

## A.5 Learning with Missing Values and Hidden Variables

Up to this point we discussed how learning Bayesian networks is performed, assuming *complete* data. In real life problems and particularly in those from the biological domain we are usually not that fortunate. some samples might have missing values (*e.g.*, some genes in a gene expression microarray experiment were not measured due to technical problems), while in other cases the variable we defined is *hidden* by definition and we have no direct measurements of it (*e.g.*, whether a transcription factor regulates a certain gene). In our biological domain, because of the noisy way in which the data is produced, it is common practice to assume the missing values are *missing at random* (MAR) [1] We now turn to describe how the two learning tasks, namely parameter estimation and structure learning, can be adopted for the presence of missing values.

### A.5.1 Parameter Estimation

When it comes to incomplete data, even the task of parameter estimation with a given structure $\mathcal{G}$ becomes hard. Recall that our ability to efficiently estimate parameters relied on our ability to decompose the log likelihood into a sum of terms that are independent of each other, according to the network structure (Eq. (**??**)). When we have missing values, the likelihood becomes a sum over all possible completions of the missing values and therefore cannot be decomposed according to the structure in the network. The number of possible completions is exponential in the number of missing values so brute force summation is not practical. Moreover, the many possible completions of the data typically give rise to many global and local maxima in the space of possible parameterizations of the model.

Given this problematic setting we need to resort to some method that optimize the likelihood function and converge to a local optimum. In some cases we may use direct optimization techniques such as gradient ascent and its variants (see (**?**)). An alternative approach commonly used is that of *Expectation Maximization* (EM) algorithm (**??**). The idea behind the algorithm is both intuitive and, in many cases, easy to implement efficiently: Since we already have efficient methods for learning with complete data, let us first “complete” the data and then perform parameter estimation over the completed dataset. All we have to make sure is that the completion of the data is done so the parameters we learn give a better likelihood over the real, not complete, data. In practice, the EM iterates between two steps. At each iteration $t$ we have some parameters for the model $\theta^t$. In the expectation (E) step of stage $t$ we compute the posterior distribution $P(H \mid O, \theta_t)$ over the missing values $H$ given the observations $O$ and the current setting of the parameters $\theta^t$. We can then compute the *expected sufficient statistics*:

$$\boldsymbol{E}_{P(H|O,\theta_t)}[S[x_i, \mathbf{pa}_i]] = \sum_m P(X_i = x_i, \mathbf{Pa}_i = \mathbf{pa}_i \mid \mathbf{o}[m], \theta_t) \qquad \text{(A.15)}$$

where $\mathbf{o}[m]$ is the $m$'th partially observed instance. The expected sufficient statistics serve to give us sufficient statistics from the data, based on our “completion” using our current model parameters $\theta^t$. As we explained for the complete data case, the sufficient statistics summarize all the relevant

[1] If not then it practically means the “missing” value tells us something about the domain so a common practice is to treat this as another possible value in the variables range. Hidden variables are MAR be definition. Otherwise, a single “MISSING” value will be observed throughout which will render the distribution of the variable not useful in terms of its ability to affect our prediction abilities.

information from the dataset and can be used directly to compute the ML or MAP parameters $\theta^{t+1}$ of the next stage (Section **??**). This optimization of the parameters given the expected sufficient is called the maximization (M) step. (**?**) have shown that the EM is equivalent to lower-bounding the likelihood function at the current model parameters via a concave function and then taking the maximum of this function. As a result EM algorithm is guaranteed to improve the likelihood of the observed data $\mathcal{D}$ at each iteration, until a (typically local) optimum is reached.

In practice the EM converges to a local optimum in the search space and the quality of this is highly sensitive to the definition of the EM starting point $\theta^{t=0}$. A general solution commonly used is to try many initial $\theta^{t=0}$ by some form of randomization over the parameter space, and them picking the best solution we get for $\theta$. In some cases it is also crucial to develop specific methods, adjusted for the problem at hand, for finding "promising" starting points that will give rise to a good solution for $\theta$.

### A.5.2 Structure Learning

The task of learning the network structure $\mathcal{G}$ with incomplete data is even more challenging. Recall that when performing structure learning with complete data we used algorithms that given a current model are able to efficiently compute its score and the score of neighboring structures. But now, because of the missing values the structure score no longer decomposes according to the network structure, by families. Therefore making a small local change in structure involves recalculating the entire new structure score and as we have just seen, this can involve many EM iterations which are much more computationally intensive then the closed form solution for complete data we had before.

As a (partial) remedy to this high computational cost of evaluating scores for similar structures to the current model, (**?**) suggested the *Structural Expectation Maximization* (SEM) algorithm that generalizes the idea of EM to the scenario of structure learning. The E-Step is identical to parametric ("regular") EM where we use $\mathcal{G}^t, \theta^t$ to generate the distribution $P(H \mid O, \theta_t, \mathcal{G}^t)$, thus providing complete sufficient statistics as in Eq. (**??**). The difference is that now in the M-Step we need to optimize not just the parameters $\theta^{t+1}$ but also the structure $\mathcal{G}^{t+1}$. Given the "completed" dataset of the E-Step, this is identical in form to structure learning with complete data. The key benefit is that we can use the sufficient statistics based on current model $\mathcal{G}^t$ to evaluate the score of all candidate new structures.As (**?**) have shown, the structural EM algorithm is guaranteed to converge to a local maximum.

## A.6 Probabilistic Relational Models (PRM)

Although Bayesian networks have proved to be a valuable tool in a variety of real world applications, they do have some limitations in the expressiveness of it as a language to define models. Specifically, think of the setting were we have several databases, each composed of several relational tables, with some relations between the tables. In our domain this could be a database that holds expression measurements in one table, description of each gene (functional annotation etc.) in a second table and description of each experiment in a third one. One database might contain experiments done in different cell cycle phases while another might have measurements under various stress conditions. The Bayesian network language does not specify any “high level rules” of how to construct a probabilistic model from these tables or to formalize the relations between the different entities involved.

To answer this need *Probabilistic Relational Models* (PRM) were developed (**??**). They extend the Bayesian network language by introducing the concept of *objects*, their properties and the relations between them. A PRM can be thought of as a *template* to specify a probability distribution over a database. The database must comply with the first component of the PRM, the *relational schema* for the domain. Given that it does, then the PRM’s second component, together with the specific objects and relations defined in the database, define a probability distribution over the objects. This kind of modeling enable us to capture high level relational structure in our domain in a formal way. In practice it leads to a definition of a specific *ground* Bayesian network that match the relational structure and conditional probability dependencies specified by the PRM for the specific objects in a given database. Because objects of the same class are assumed to share the same probability dependency structure, the ground Bayesian network has CPD which are *shared* between random variables of the same type. Thus, the PRM captures the underlying structure while enabling us to still use all the extensive machinery developed for Bayesian networks learning. We now turn to define the concepts involved in more details, and then review PRM learning. As a running example we will use the PRM illustrated in Figure **??**.

### A.6.1 The Relational Language

The relational language enables us to formalize the definition of the various entities in the domain of interest and the relations between them. As we shell see, the concepts are highly correlated to those of relational databases which serve as a primary motivation. However, the framework defined for PRM is a general one and as we shell see this probabilistic modeling approach carries many benefits not necessarily related to relational databases.

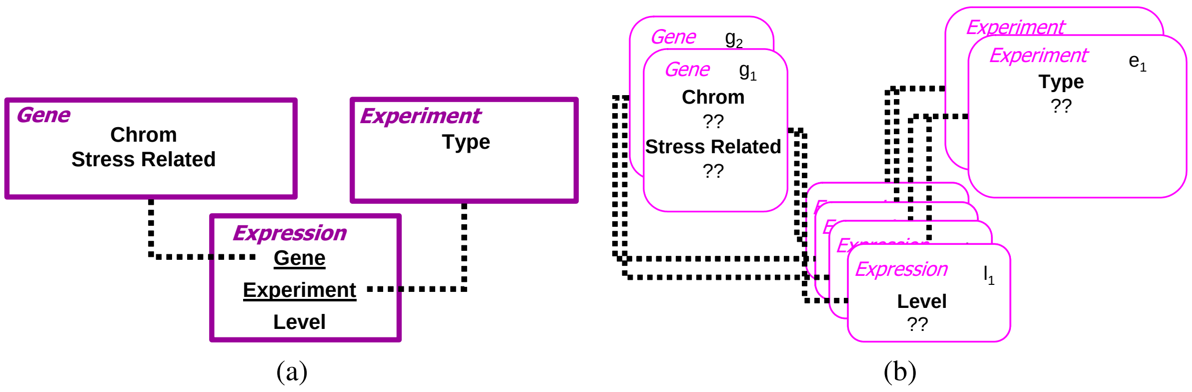


Figure A.4: (a) A relational schema for a simple gene expression domain . The underlined attributes are reference slots of the class and the dashed line indicates the types of objects referenced. (b) A relational skeleton $\sigma_\rho$ of (a) with only 2 experimented and 2 genes. Here we do not show the reference slots and used dashed lines to indicate the relationships that hold between objects.

## Relational Schema

A *schema* of a relational model describes a set of classes $\mathcal{C} = \{C_1, \ldots, C_n\}$. Each class has a set of *descriptive attributes* and a set of *reference slots*. For example, a descriptive attribute of a *Gene* class in our domain might be the functional annotation of this gene. The set of descriptive attributes of a class is denoted $\mathcal{A}(C)$. Attribute $A$ of class $C$ is denoted $C.A$ and its domain of values is denoted $Val(C.A)$. As we shell shortly see, specific databases might contain several objects (*e.g.*, genes) of the same class (*Gene*, in this case), and the values of their attributes may either be observed or *hidden*.

As we defined above, the schema also describes a set of *reference slots* for each class $C$, denoted $\mathcal{R}(C)$. We use $C.\rho$ to denote a specific reference slot $\rho$ in class $C$. A reference slot is what allows us to have one object refer to another object. In our example we have each expression measurement object of class *Expression*) refer to a specific gene object involved in that measurement. If we think of it in terms of a relational database, then a class can correspond to a specific table in the database and each entry in the table would be a specific object of that class. The reference slots in this case are the foreign keys for objects in other tables. Formally, a reference slot is a typed function so that a reference $C.\rho$ has a domain type $Dom(\rho) = C$ and a range type $Range(\rho) = C'$, where $C, C' \in \mathcal{C}$. Similarly, we can define the *inverse* slot $\rho^{-1}$ as the inverse function of $\rho$. Note that both $\rho, \rho^{-1}$ are not necessarily a one to one mapping from objects in their domain to objects in their range. For example, the inverse slot for a *Gene* slot in a class *Expression* can return a whole set of Expression objects that have this specific gene as the range value of their *Gene* slot. Finally, we can also define a *slot chain* in a similar way, as a function with a typed range and a typed domain. Formally, for a sequence of slot $\rho_1, \ldots, \rho_n$ (inverse or not) to be well defined we need to have:

$Range(\rho_i) = Domain(\rho_{i+1}), \forall i \in 1 \ldots, n-1.$

### Schema Instantiation

An *instance* $\mathcal{I}$ of a schema $\Sigma$ specifies for each class $C \in \mathcal{C}$ its set of objects $\mathcal{I}[C]$; a value for each attribute $c.A$ in each object $c$; and a value $y$ for each reference slot $c.\rho$ such that $y$ is some object in the matching class $Range[\rho] = Y, Y \in \mathcal{C}$. We can think of $\mathcal{I}$ as a specific relational database that match the relational schema $\Sigma$ with all the non missing values (the values in each row in each table) specified. A *relational skeleton* $\sigma_\rho$ of a schema $\Sigma$ is a partial specification of an instance $\mathcal{I}$ of $\Sigma$. It contains only the set of objects in each class, denoted $\sigma_\rho[C]$ and the value of the reference slots that define the relations between the objects. The value of the attributes are not specified. If we follow our relational database allegory, we can think of $\sigma_\rho$ as a database with all the tuples defined in each table along with all the foreign keys in them, but without the actual values of the different attributes in the tables.

### Model Definition

Given the above definition, we can finally formalize our PRM model. To put it in simple terms, given a schema $\Sigma$ and a relational skeleton $\sigma_\rho$ that complies with it, the PRM model defines a probability distribution over all possible (full) instantiations $\mathcal{I}$ that match $\sigma_\rho$. Note that the definition of $\sigma_\rho$ is *outside* of the probabilistic model. For example the PRM by itself *does not* specify the amount of objects in each class or any distribution over their relational structure.

Similar to the Bayesian network definition, the PRM model consists of two components, the first is a qualitative one related to the structure of the graphical model and the second is a quantitative one, defining the conditional probabilities that follow the structure definition. We start with the first component, the *qualitative* dependency structure, $\mathcal{S}$. This associate with each attribute $C.A$ a set of *formal parents* $\mathbf{Pa}_{C.A}$. Note that unlike Bayesian network this definition does not relate to specific *variables* but rather to specific *attributes* of classes. For example, a PRM might specify a dependency of $Expression.Val$ on the attribute $Experiement.Type$. Only when we combine the PRM with a specific relational skeleton $\sigma_\rho$ that specify all objects for each class we would get the matching random variables for expression measurement. Following our previous definitions, the formal set of parents $\mathbf{Pa}_{C.A}$ can contain attributes of the same class but also attributes of other classes (as in the example we just gave), via some slot chain. The second component of the PRM, denoted $\theta_{\mathcal{S}}$ is the *quantitative* one, that defines a CPD $P(C.A \mid \mathbf{Pa}_{C.A}), \forall C \in \mathcal{C}, \forall A \in \mathcal{A}[C]$. More specifically, if we denote the set of parent attributes $U = \mathbf{Pa}_{C.A}$ then this set has a range of values $Val(U)$ so that the CPD $P(C.A \mid \mathbf{Pa}_{C.A})$ specify a conditional probability distribution $P(C.A \mid \mathbf{u})$ for all $\mathbf{u} \in U$ . We can now give the formal definition of the PRM model:

**Definition A.6.1:** A probabilistic relational model (PRM) $\Pi$ for a relational schema $\Sigma$ defines for each class $C \in \mathcal{C}$ and each attribute $C.A \in \mathcal{A}[C]$ the following:

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

∎

Thus, for a given skeleton $\sigma_\rho$, the PRM $\Pi$ defines a *ground* Bayesian network. The *qualitative* structure of the network $\mathcal{G}_{\sigma_\rho}$ is the *instance dependency graph* where the nodes are the descriptive attributes of objects in the skeleton (like $c.A$) and there is an edge in the graph $b.B \rightarrow c.A$ if $b.B$ is the actual parent according to the PRM definitions. The *quantitative* component of the ground Bayesian network is defined by the matching CPDs of the PRM $\Pi$. Note that this also means the different variables that relate to the same attribute in the class definition *share* the same CPD in the ground Bayesian network. Applying the *chain rule* on the ground Bayesian network we get:

$$P(\mathcal{I} \mid \sigma_\rho, \Pi) = \prod_{C \in \mathcal{C}} \prod_{c \in C} \prod_{A \in \mathcal{A}[C]} P(c.A \mid \mathbf{pa}_{c.A}) \tag{A.16}$$

Of course in order for this to define a coherent probability distribution $\mathcal{G}_{\sigma_\rho}$ must be acyclic. (see more in (**?**) on how this can be verified efficiently for general cases).

## A.6.2 Learning and Inference with PRM

In the last section we have just seen how a PRM, coupled with a matching skeleton $\sigma_\rho$ defines a ground Bayesian network. This means that all the methods we reviewed in previous sections for learning and inference in Bayesian networks can be applied here directly. In this section we will therefore only highlight a few points that characterize PRM learning.

First and maybe most notably, the ground Bayesian network usually have *shared* CPDs. In terms of learning, this means the sufficient statistics collected to update the CPDs during learning (see Section **??**) may now come from several different variables in the network.

Second, unlike model definition via regular Bayesian network language, the PRM model holds the relations structure of the data. For example say we have a few measurements of a specific gene $g$ expression profile, that do not fit our regulation model. Given the PRM model it can be easier to identify that these samples are from the same gene and maybe introduce a distinct distribution for them which should be modeled differently (via a separate CPD or maybe by introducing a new attribute for *Gene* class, *Type*). If we were to use just the Bayesian network language all the relational data is lost and in this case we would simply treat all samples as i.i.d and fail to notice the discrepancy. In fact, Bayesian network modeling can be viewed as a simple case of PRM, where we only have one class involved.

The fact a PRM keeps the relational structure comes with a high price though, namely the number of variables in the ground Bayesian network. Following our example, if we measure the expression profile of $G$ genes in $M$ experiments then in a regular Bayesian network we might have a variable $E_1, \ldots E_M$ for gene expression in each experiment, and observe $G$ samples for each in the experiments performed. In the PRM model we would probably have a random variable for each genes in each experiment, *i.e.*, $M$ times more variables, with shared CPD between them. The *inference* task in this setting is therefore much more computationally demanding, and so is the physical representation of the model in terms of computer memory. In practice, this leads to the usage of many *approximate* inference algorithms as the ones described in Section **??**. Another common practice is to use the *EM* algorithm but with *hard assignments*. This means that in the expectation $E$-step of the algorithm we perform (approximate) inference over the missing values and then *complete* the data with the the most probable assignment. Given the full assignment and its resulting sufficient statistics, the $M$-step is carried in the regular way.

Finally we note that in the framework we reviewed here the relational structure is assumed to be given and not part of the probability model. The PRM modeling language can be expended to allow uncertainty in the relational structure and have the relational structure learned as part of the learning task (for more on this see (**?**)).

# Notation

| | |
|---|---|
| $X, Y, Z \ldots$ | random variables or their corresponding nodes in the network |
| $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \ldots$ | sets of random variables or their corresponding nodes in the network |
| $Val(X)$ | the set of possible values of a discrete variable $X$ |
| $x, y, z \ldots$ | assignments to random variables |
| $\mathbf{x}, \mathbf{y}, \mathbf{z} \ldots$ | joint assignments to sets of random variables |
| $(\mathbf{X} \perp \mathbf{Y})$ | independence of (sets of) random variables |
| $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$ | conditional independence of (sets of) random variables |
| $P(\mathbf{X})$ | probability density of (sets of) random variables ($P$ is used to denote $P(\mathcal{X})$) |
| $P(\mathbf{X}, \mathbf{Y}, \mathbf{Z} \ldots)$ | joint probability density of several (sets of) random variables |
| $P(\mathbf{X} \mid \mathbf{Y})$ | conditional probability density of $\mathbf{X}$ given $\mathbf{Y}$ |
| $\mathcal{B}$ | a Bayesian network |
| $\mathcal{X}$ | the set of all variables in the network $\{X_1, \ldots, X_N\}$ |
| $\mathcal{G}$ | the graph structure of a network |
| $\theta$ | the parameters of a network |
| $\theta_{X_i \mid \mathbf{Pa}_i}$ | parameter of the CPD of $X_i$ |
| $\theta_{x_i \mid \mathbf{pa}_i}$ | parameter corresponding to the values $x_i$ and $\mathbf{pa}_i$ in the CPD of $X_i$ |
| $\mathbf{Pa}_i^{\mathcal{G}}$ | the parents variables of the variable $X_i$ in $\mathcal{G}$ |
| $\mathbf{Ch}_i^{\mathcal{G}}$ | the children variables of the variable $X_i$ in $\mathcal{G}$ |
| $\mathbf{Fam}_i^{\mathcal{G}}$ | the family of $X_i$ in $\mathcal{G}$ (itself and its parents) |
| $\mathbf{Adj}_i^{\mathcal{G}}$ | the adjacent variables to $X_i$ in $\mathcal{G}$ (its parent and children) |
| $\mathbf{MB_i^{\mathcal{G}}}$ | the Markov blanket variables of $X_i$ in $\mathcal{G}$ (its parent, children and their parents) |
| $\mathbf{pa}_i^{\mathcal{G}}$ | an assignment to parents of $X_i$ |
| $N$ | number of variables in the network |
| $\mathcal{D}$ | the training data set |
| $M$ | number of instances in training set |
| $S[x_i, \mathbf{pa}_i]$ | the sufficient statistics count of $x_i$ and $\mathbf{pa}_i$ in $\mathcal{D}$ |
| $\mathbf{H}$ | hidden variables |
| $\mathbf{O}$ | observed variables |
| $\mathbf{x}[m]$ | the value of $\mathbf{x}$ in the $m$'th sample |
| $\mathbf{o}[m]$ | the $m$'th partially observed instance |
| $L(\theta, \mathcal{G} : \mathcal{D})$ | likelihood of the data given $\theta$ and $\mathcal{G}$ |
| $\ell(\theta, \mathcal{G} : \mathcal{D})$ | log-likelihood of the data given $\theta$ and $\mathcal{G}$ |
| $L_{X_i}(\theta, \mathcal{G} : \mathcal{D})$ | local likelihood of $X_i$'s family |
| $\ell_{X_i}(\theta, \mathcal{G} : \mathcal{D})$ | local log-likelihood of $X_i$'s family |
| $\hat{\theta}$ | estimated parameters |
| $\alpha^j$ | hyper-parameters corresponding to the $j$th value in a multinomial distribution |
| $\alpha_{x_i^j}$ | hyper-parameters corresponding to the $j$th value of $X_i$ |
| $\mathrm{Score}(\mathcal{G} : \mathcal{D})$ | score of $\mathcal{G}$ given $\mathcal{D}$ |