

Computer Sciences Department

Learning Expressive Computational Models of Gene
Regulatory Sequences and Responses

Keith Noto

Technical Report #1615

September 2007

**LEARNING EXPRESSIVE COMPUTATIONAL MODELS
OF GENE REGULATORY SEQUENCES AND RESPONSES**

by

Keith Noto

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2007

© Copyright by Keith Noto 2007
All Rights Reserved

To Mom and Dad.

Abstract

The regulation and responses of genes involve complex systems of relationships between genes, proteins, DNA, and a host of other molecules that are involved in every aspect of cellular activity. I present algorithms that learn expressive computational models of *cis*-regulatory modules (CRMs) and gene-regulatory networks. These models are *expressive* because they are able to represent key aspects of interest to biologists, often involving unobserved underlying phenomena. The algorithms presented in this thesis are designed specifically to learn in these expressive model spaces.

I have developed a learning approach based on models of CRMs that represent not only the standard set of transcription factor binding sites, but also logical and spatial relationships between them. I show that my expressive models learn more accurate representations of CRMs in genomic data sets than current state-of-the-art learners and several less expressive baseline models.

I have developed a probabilistic version of these CRM models which is closely related to hidden Markov models. I show how these models can perform inference and learn parameters efficiently when processing long promoter sequences, and that these expressive probabilistic models are also more accurate than several baselines.

Another contribution presented in this thesis is the development of a general-purpose regression learner for sequential data. This approach is used to discover mappings from sequence features in DNA (*e.g.* transcription or sigma factor binding sites) to real-valued responses (*e.g.* transcription rates). The key contribution of this approach is its ability to use the real values directly to discover

the relevant sequence features, as opposed to choosing the features beforehand or learning them from sequence alone, and without losing information in a discretization process.

Finally, I present and evaluate a gene-regulatory network that learns the hidden underlying *state* of regulators from expression data and a set of cellular conditions under which expression is measured. I show that using sequence data to estimate the *role* of regulators (activator or repressor) increases the accuracy of the learned models.

Acknowledgments

I am happy to have this opportunity to acknowledge and thank the many people who have helped to make my education a positive and successful experience, and without whom this thesis would not exist.

First and foremost, I owe everything to my advisor, Mark Craven. I could not have hoped for a better mentor during my graduate career. In addition to being one of the most clever and insightful people as well as one of the best teachers I have ever known, Mark has continually shown energy, respect and patience as I developed in my study of machine learning, research and writing skills. For everything I have learned in the last five years about the world of research and academia, I have Mark to thank.¹

I would like to thank professors Jude Shavlik, David Page, and Chuck Dyer for helping me with my decision to go into the area of artificial intelligence, and for helping me with advice and guidance along the way. I would especially like to thank Jude and David for frank advice as I plan my career.

I am extremely appreciative of professors Alan Attie, Audrey Gasch, James Thomson, and their groups for providing data and for helping me to understand the processes that created those data. The meetings with these groups that I have taken part in have helped me understand how people in other fields use language and do science, and provided me with a wealth of background knowledge.

I would like to thank Mark Craven, Jude Shavlik, David Page, Alan Attie, Jamie Thomson, and Colin Dewey for agreeing to be on my preliminary exam and PhD defense committees.

I would like to acknowledge and thank students in the machine learning and AI group, Louis Oliphant, Joe Bockhorst, Adam Smith, Jesse Davis, Yue Pan, Irene Ong, Frank Dimaio, Burr Settles, Mark Goadrich, Aaron Darling, Soumya Ray, Sean McIlwain, Michael Molla, Beverly Seavey, Lisa Torrey, Trevor Walker, Eric Lantz, Ameet Soni, David Andrzejewski, Mike Waddell, and Ted Wild. If it were not for Adam Smith, I am quite sure I would never have passed the computer science PhD qualifying exam. I would like to thank Louis Oliphant, Joe Bockhorst, and Jesse Davis, who helped us with some of our paper drafts. Several students on this list helped to organize the AI seminars and reading group, and I am thankful for the fact that all of them are very cooperative and wicked smart. I would also like to thank professors Rich Maclin and Jerry Zhu for their insights and helpful feedback at talks.

¹I also have Mark to thank for any fitness I might still possess. At Wisconsin, it has come to be known that Mark's students must also run a marathon before they can graduate.

I would like to acknowledge the professors that influenced my attitude toward teaching with their courses. These teachers are Jude Shavlik, David Page, Mark Craven, Chuck Dyer, Charlie Fischer, Olvi Mangasarian, Nichole Perna, Tom Reps, and Ras Bodik, to name just a few.

For much of my graduate career, I was on the CIBM (Computational Informatics in Biology and Medicine) training grant. I would like to thank the National Library of Medicine for funding me and other students at Wisconsin. I would like to thank Louise Pape for coordinating the seminar series and my fellow fellows for their feedback and presentations. I owe a special acknowledgment to the NLM and to the CIBM training grant for it was through this grant that I was introduced to my fiancé, Dalia.

I would not be writing this thesis without Dalia. She has been an unwavering source of support and encouragement. I want to thank my parents and sister Shanna for their constant support and interest in my education. I hope this thesis will help answer their questions about what it is that I do.

The research presented in this thesis was supported in part by NIH/NLM training grant 5T15LM005359, NSF grant IIS-0093016, and NIH/NLM grant R01-LM07050-01.

Table of Contents

	Page
Abstract	ii
List of Tables	ix
List of Figures	xi
1 Introduction	1
1.1 Supervised Learning	1
1.2 Model Space	2
1.3 Expressive Models for Gene Regulation	5
1.4 Thesis Statement	8
1.5 Outline	8
2 Background	9
2.1 Biology	9
2.2 Probabilistic Graphical Models	12
2.2.1 Bayesian Networks	12
2.2.2 Hidden Markov Models	12
2.3 Evaluating Models	19
3 Learning the Logical and Spatial Aspects of a cis-Regulatory Module	22
3.1 Introduction	22
3.2 Related Work	26
3.3 Approach	30
3.3.1 Learning a Model	30
3.3.2 Controlling the Expressivity of a Model	33
3.4 Empirical Evaluation	35
3.5 Conclusion	43
4 Learning Probabilistic Models of cis-Regulatory Modules	44
4.1 Introduction	44

	Page
4.2	Model Representation 47
4.3	Learning a Model 52
4.3.1	Structure Learning 53
4.3.2	Parameter Learning 53
4.4	Efficient Computation 56
4.5	Results 58
4.5.1	Evaluating Predictive Accuracy 60
4.5.2	Evaluating the Effectiveness of Logical and Spatial Aspects 60
4.5.3	CRMs in Human 65
4.5.4	Incorporating Background Knowledge 66
4.6	Related Work 66
4.7	Conclusion 67
5	Learning Hidden Markov Models for Regression using Path Aggregation 71
5.1	Introduction 71
5.2	Related Work 73
5.3	Approach 74
5.3.1	Model Representation 74
5.3.2	Parameter Learning 77
5.3.3	Structure Learning 82
5.4	Empirical Evaluation 82
5.5	Conclusion 84
6	Learning Models of Gene-Regulatory Networks that Represent States and Roles . . 87
6.1	Introduction 87
6.2	Related Work 88
6.3	Approach 89
6.3.1	Network Architecture 89
6.3.2	Representing Gene Expression States 91
6.3.3	Representing Conditional Probability Distributions 92
6.3.4	Learning Network Parameters 93
6.3.5	Initializing Network Parameters 95
6.4	Empirical Evaluation 96
6.4.1	Experimental Data and Methodology 96
6.4.2	Experiment 1: The Value of Representing Regulator States 97
6.4.3	Experiment 2: Discovering Missing Regulators 99
6.4.4	Experiment 3: The Value of Initializing Regulator Roles 100
6.5	Conclusion 100

	Page
7 Conclusions	102
7.1 Learning the Logical and Spatial Aspects of <i>cis</i> -Regulatory Modules	102
7.2 Learning Probabilistic Models of <i>cis</i> -Regulatory Modules	103
7.3 Learning Hidden Markov Models for Regression	105
7.4 Learning Models of Gene-Regulatory Networks that Represent States and Roles . .	106
7.5 Future Directions	107
7.6 Final Thoughts	108
 APPENDICES	
Appendix A: SCRM1 Results on Lee etal Data Sets	120
Appendix B: Path Aggregate Learning: Results on Simulated Data Sets	123

List of Tables

Table	Page
1.1 Data for Toy Supervised Learning Task #1.	2
1.2 Data for Toy Supervised Learning Task #2.	4
1.3 Data for Toy Supervised Learning Task #3.	5
2.1 Example Full Joint Distribution.	13
2.2 Example Parameters for a 3rd-order Markov Chain Over {a, c, g, t}.	14
2.3 Forward Algorithm DP Matrix.	17
2.4 Binary Confusion Matrix Definitions.	20
2.5 Pseudocode for Test Set N -Fold Cross-Validation.	21
2.6 Pseudocode for Learning With a Tuning Set.	21
3.1 Pseudocode for the SCRM1 TRAIN Function	31
3.2 Pseudocode for the SCRM1 SELECT-TRAIN Function	32
3.3 Summary of Data Sets for Evaluating SCRM1.	36
3.4 SCRM1: Results on the Lee <i>et al.</i> Data Sets.	37
3.5 SCRM1: Results on the Gasch <i>et al.</i> Data Sets.	38
3.6 SCRM1: Results on the Sinha <i>et al.</i> Data Sets.	38
4.1 Pseudocode for the SCRM2 Learning Algorithm.	54
4.2 SCRM2 Results on Lee <i>et al.</i> Yeast Data Sets.	61

Table	Page
4.3 Comparison Between SCRM2 and Bag-of-Motifs on Yeast Data.	61
4.4 SCRM2 Classification Margins of Four Data Sets.	62
4.5 Probabilistic Grammar Productions that Explain a Single Motif.	68
4.6 A Comparison of CRM Learning Approaches.	69
6.1 Predictive Accuracy for Gene Regulatory Network and Baselines.	98
6.2 Predictive Accuracy for Gene Regulatory Networks with Added Hidden Nodes.	99
6.3 Predictive Accuracy of Promoter-Based Parameter Initialization.	100
Appendix	
Table	
A.1 SCRM1: Results on the Lee <i>et al.</i> Data Sets.	121
A.2 SCRM1: Results on the Gasch <i>et al.</i> Data Sets.	122
A.3 SCRM1: Results on the Sinha <i>et al.</i> Data Sets.	122
B.1 Path Aggregate Learning: Results on Simulated Data Sets	124
B.2 Two-Phase Baseline: Results on Simulated Data Sets	124
B.3 Constant Baseline: Results on Simulated Data Sets	125

List of Figures

Figure	Page
1.1 A Decision Tree Model for the Data Set in Table 1.1.	3
1.2 A Regulatory Module in the Sea Urchin Put Forth by Yuh <i>et al.</i>	7
2.1 Illustration of Transcription and Translation.	10
2.2 Illustration of the <i>E. coli</i> Core Promoter.	10
2.3 Illustration of a Transcription Factor Binding to DNA.	11
2.4 Example Position Weight Matrix.	11
2.5 A Simple Bayesian Network Model.	13
2.6 A Simple HMM Model.	15
3.1 Example CRM Learning Task.	23
3.2 Example CRM Model Corresponding to the Task in Figure 3.1.	23
3.3 Example CRM Model.	25
3.4 CRM Model of Segal and Sharan.	27
3.5 CRM Model of Zhou and Wong.	27
3.6 The HMM CRM Models of Sinha <i>et al.</i>	28
3.7 Logic Regression CRM Model of Keleş <i>et al.</i>	29
3.8 SCRM1: Recovered Motifs from the Lee <i>et al.</i> Data Sets.	39
3.9 SCRM1: F1 Measure Scatterplots.	41

Figure	Page
3.10 Hypothesized CRM for the PHD1, YAP6 Data Set	42
3.11 Hypothesized CRM for the rESR_RPcluster Data Set.	42
4.1 Two Probability Distributions over Distance.	45
4.2 SCRM2: Example CRM Model.	46
4.3 HMM of a Sequence with a Single Motif.	47
4.4 Example SCRM2 CRM Model.	49
4.5 SCRM2: Model Represented as an HMM.	50
4.6 SCRM2: Model with Negated Binding Site Represented as an HMM.	51
4.7 SCRM2 HMM Model Space Operators.	55
4.8 SCRM2 Dynamic Programming.	57
4.9 SCRM2 Efficiency Gain.	59
4.10 SCRM2: P-R Curves for Four Data Sets.	63
4.11 SCRM2 Hypothesis for Yeast PAC/RRPE.	64
4.12 SCRM2 P-R Curve for Human Data Set.	65
5.1 A Sequence-Based Regression Task.	72
5.2 An HMM for Regression and the Corresponding Graphical Model.	75
5.3 Example Gaussian Mixture Model.	78
5.4 Error Rate on Simulated Data.	83
5.5 Error Rate over 15 Yeast Microarray Data Sets.	86
5.6 HMM Structure Used with Yeast Data Sets.	86
6.1 Illustration of Gene Regulatory Network Model.	90
6.2 Expression States for the Gene metE	90

Figure	Page
6.3 Example Promoter Configuration.	96
6.4 Examples of Baseline Regulatory Networks.	98

Chapter 1

Introduction

Machine learning is the area of artificial intelligence focused on developing computer programs that improve with experience. They improve by becoming better able to explain observations, make decisions, or predict the future. Machines have learned to drive cars [Pomerleau, 1989; Pomerleau, 1993], play games and sports [Kitano, 1998; Tesauro and Sejnowski, 1989], diagnose illness [Mangasarian *et al.*, 1995], interpret human language and speech [Franzini, 1987; Manning and Schütze, 1999; Mooney and DeJong, 1985; Rabiner, 1989], recognize faces [Guo and Dyer, 2007], and hypothesize models to explain a wide variety of observations, from protein interactions to airplane ticket prices [Blaschke *et al.*, 1999; Bunescu *et al.*, 2005; Darnell *et al.*, 2007; Etzioni *et al.*, 2003].













The nature of the learning task determines the form of input to these programs, as well as the algorithm used to learn. A car that is to learn how to drive by itself might continuously receive input in the form of digital camera images from behind the windshield, along with the “correct” steering direction. A backgammon-playing learner might receive as “input” its own moves, as well as those of its opponent, and would have to wait until the end of the game to begin to make judgments about those moves, when it finds out how many points are won or lost. A disease predictor might receive, all up front and at once, a large database of medical records along with the disease status (*e.g.* sick or healthy) of each patient.

1.1 Supervised Learning

These tasks where the machine is given the “right answers,” called *labels*, for each chunk of input are called *supervised learning* tasks. The input to the learner is called the *training set*, and consists of *examples*, $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, where \mathbf{x}_i is some encoding of the *features* of the *i*th example, and y_i is its label. The learner is to find a *model*¹ of the form:

¹I use the term *model* throughout this thesis to be synonymous with a supervised learning *hypothesis*. It is something with an interpretation, under which input examples are mapped to output labels. Similarly, the concept of model space discussed in Section 1.2 is commonly referred to as *hypothesis space* in the machine learning community.

Table 1.1 Data for toy supervised learning task #1. The symbols \oplus and \ominus indicate a positive and negative labels.

	Example	Feature Set			Label		Example	Feature Set			Label
		Style	Color	Shape				Style	Color	Shape	
1		Solid	Green	Square	\oplus	7		Solid	Blue	Square	\ominus
2		Open	Blue	Circle	\ominus	8		Open	Red	Square	\oplus
3		Solid	Blue	Circle	\ominus	9		Solid	Blue	Triangle	\ominus
4		Open	Blue	Square	\ominus	10		Open	Red	Circle	\ominus
5		Solid	Green	Circle	\oplus	11		Open	Green	Square	\ominus
6		Solid	Blue	Triangle	\ominus	12		Solid	Red	Square	\oplus

$$y = f(\mathbf{x}). \quad [1.1]$$

There are two reasons such a model is valuable. First, an agent is able to use the model to predict the label of an unlabeled example. This prediction helps an agent make the right move in a game or decide ahead of time if a patient has a disease (or just how likely it is). In the latter case, the cost of mislabeling the example could mean waiting until it is too late to begin an effective treatment.

The model is also valuable because it can lend some insight into the nature of the concept in question. A doctor may wish to know which features (medical test results, *etc.*) are good predictors of a patient's disease because this suggests treatments or helps to guide further research. In any case, it can shed light on underlying causes and lead to a deeper understanding of the concept in question. The emphasis in this thesis will be on this second reason. My research involves developing learning algorithms for biological tasks when there may be no future examples, but for which we use machine learning to come to a better understanding of the underlying biological mechanisms that result in observable data.

1.2 Model Space

It is difficult for a learning algorithm to generalize if it is capable of producing every conceivable model [Geman *et al.*, 1992; Mitchell, 1997; Quine, 1960]. Instead, each learning algorithm finds its model by searching through its own *model space*. In other words, each learning algorithm has its own *form* of model, which biases the learner toward certain (hopefully good) models.

Consider the toy supervised learning task in Table 1.1. Here, the features are encoded as vectors, and the labels are either positive or negative, *e.g.* (\mathbf{x}_1, y_1) is $(\langle \text{Solid Green Square} \rangle, \oplus)$. The positively labeled examples are instances of the concept to be learned, and the model should distinguish them from the negatively labeled examples, which are often present to keep the learner from overgeneralizing, *e.g.* “since everything in the training set is positive, *everything* is positive.”

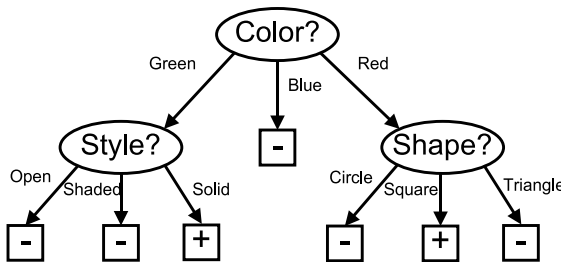


Figure 1.1 A decision tree model for the data set in Table 1.1.

There are several ubiquitous learning algorithms for classifying tabular data like these, such as nearest neighbor [Cover and Hart, 1967; Shepard, 1968], decision trees [Breiman *et al.*, 1984; Friedman, 1977; Quinlan, 1986], naïve Bayes [Mitchell, 1997] and artificial neural networks [Bryson and Yu-Chi, 1969].

For instance, decision tree learners produce a disjunctive model like the one shown in Figure 1.1, which is interpreted as the following set of rules:

$$f(\mathbf{x}) = \begin{cases} \oplus & \text{if } \text{Color} = \text{Green and Style} = \text{Solid} \\ \text{or} & \text{Color} = \text{Red and Shape} = \text{Square} \\ \ominus & \text{otherwise} \end{cases} \quad [1.2]$$

Because of the way decision trees divide up the model space, there are models that decision trees cannot represent.² For instance, they cannot represent the target model exactly for the learning task in Table 1.2 (right triangles), although they can find models that correctly classify all the data, even if there are hundreds of examples.

There are machine learning algorithms that can take advantage of *relationships* between objects (if we consider each side of a triangle an object, then the target relationship is $A^2 + B^2 = C^2$ or something equivalent), such as inductive logic programming [Muggleton and DeRaedt, 1994; Muggleton, 1995] and more recently, statistical relational learning [Getoor and Taskar, 2006] but even these might have trouble choosing exactly the right mathematical operations. It would help to have a model space that includes aspects more closely related to “triangleness,” without including all possible mathematical relationships.

The notion of model space is central to this thesis. The main contribution is the development of algorithms that are able to learn expressive representations for certain biological tasks. In this context, *expressive* means that the model space includes more than standard features, but those specific to the task at hand. It also means, as we will see, that the learning algorithms need to be specially designed to work well with expressive models.

Of course, the nature of model space depends on the way the features and labels are encoded, as well as the learning algorithm. The binary labeling (\oplus or \ominus) in Tables 1.1 and 1.2 is the simplest case of having *class labels*, where labels come from a discrete set of classes. Models that learn

²Decision trees are well-studied, and there are advancements that involve looking a few steps ahead to make the best split [Murthy and Salzberg, 1995] combining features [Murphy and Pazzani, 1991], handling continuous-valued features [Quinlan, 1993; Quinlan, 1996], and making non axis-parallel splits [Murthy *et al.*, 1994].

Table 1.2 Data for toy supervised learning task #2.

	Example	Feature Set					Label
		Style	Color	A	B	C	
1		Shaded	Red	20	21	29	\oplus
2		Shaded	Blue	4	4	4	\ominus
3		Open	Blue	3	5	6	\ominus
4		Solid	Red	5	12	13	\oplus
5		Open	Green	3	4	5	\oplus
6		Solid	Green	2	3	4	\ominus
7		Open	Blue	6	9	9	\ominus

Table 1.3 Toy learning task involving sequence data. Common sequence features have been highlighted. There are two types of labels for each example, a real-valued response, and a discrete class label.

	Sequence	Value	Class
1	gttcct <u>cg</u> ttagtc <u>cg</u> gctata <u>cg</u> gaatagtaagctata <u>cg</u> ggg	3.0	⊕
2	ctatgagctgataaagttagattctcgggcttgccaatcatacatggga	-0.1	⊖
3	aaactgctata <u>cg</u> ggcgat <u>cg</u> ttagtc <u>at</u> gcctaccacactgc	0.9	⊕
4	gt <u>cg</u> ttagtc <u>ct</u> <u>cg</u> ttagtc <u>ct</u> agctata <u>cg</u> ggcgtt <u>cg</u> ttagtc <u>at</u> ca	-0.9	⊖
5	aggagcggcacacggccgtaag <u>cg</u> ttagtc <u>at</u> ta	-1.1	⊖
6	tcgcgtctgatttgcccggtcgtagtc <u>tg</u> gacg	-1.0	⊖
7	gccacccatctaagaaccacatgcgctgatctacgtc <u>cg</u> ctata <u>cg</u> ggccg	1.9	⊕
8	<u>cg</u> ttagtc <u>cat</u> cagctata <u>cg</u> gactggtc <u>cg</u> ttagtc <u>aa</u>	-0.1	⊖

from these labels are aptly called *classifiers*, but labels can be continuous-valued as well, making the learned models *regression models*, which might learn a mapping from input examples to a predicted value or to a probability density function. Also, not all data can be represented as a set of feature vectors. Input data could be in the form of a relational database consisting of an unordered set of object-relations. In these cases, a learned model might be a set of rules expressed in first-order logic that classifies objects.

Other learning tasks, such as the DNA-promoter analysis task described in depth in this thesis, involve learning from examples encoded as variable-length sequences. Table 1.3 shows an example of a learning task involving variable-length sequence data. Two sequence features are highlighted—the recurring short subsequences cgttagtc and gctatacg. In general, however, the relevant sequence features are hidden, and their discovery is part of the learning task. Table 1.3 shows two types of labels, a real value and a discrete class label. A learner would typically be given one or the other. The value label (*i.e.* for a regression task) is roughly $2m_2 - m_1$ where m_1 is the number of times cgttagtc occurs in the sequence, and m_2 is the number of times gctatacg occurs. The class label is a discretization of this value, *i.e.* for a classifier that requires discrete labels. Hidden Markov models [Krogh, 1994; Rabiner, 1989], stochastic context-free grammar models [Lari and Young, 1990; Grate *et al.*, 1994], and conditional random fields [Lafferty *et al.*, 2001] are often used to represent the models based on sequence data.

1.3 Expressive Models for Gene Regulation

Every living cell sustains its own needs, in part by producing proteins and other products, such as ribonucleic acid (RNA) molecules. The task that a given gene product is able to do largely depends on its structure, which is encoded by a gene. When a certain gene product is required, the gene that encodes it becomes *expressed*, and the product is created. If the wrong genes are

expressed (or the appropriate ones are not expressed) at a given time, the results can be harmful to the cell, so the system of *gene regulation* has evolved to be responsive, robust, and complex.

The work presented in this thesis involves machine learning methods for uncovering gene regulation. Most of it focuses on the *mechanisms* of gene regulation: What is it about the *DNA itself* that determines exactly when a gene will be expressed? There are training examples from which to learn such concepts, but the answer can be quite complicated, involving specific properties of multiple locations on the DNA.

For instance, Figure 1.2 shows a map of several *sites* (locations on DNA), near a gene called *endo16* in *S. purpuratus* (the purple sea urchin), and the interactions between those sites [Yuh *et al.*, 2001]. Each site is associated with its own protein, which may or may not be bound to the DNA there. These proteins help control the expression of the gene. This model consists of sites on DNA and *relationships* between those sites. In part, these relationships are spatial, since they have to be close enough for the proteins to interact, and in part, they are logical, since sometimes one site, or another, or both need to be present. The hypothesized interactions that are drawn below the DNA in Figure 1.2 consist entirely of unobserved interactions between molecules, but their behavior is key to understanding the system.

For a machine to learn a model like this, the model space through which it searches would have to be made up of these relational aspects, and represent the unobserved phenomena as well. These are the kinds of aspects that I have tried to capture in the learning algorithms I present in later chapters. Of course, supervised learners do not learn from just one example. They come up with models generalized over multiple examples, so an appropriate learning task would be to find out what parts of a regulatory mechanism like the *endo16* modules are common to a set of genes.

Much of what is known specifically about molecular biology, like the map shown in Figure 1.2, comes from years of study. The *endo16* module comes from Yuh *et al.* [2001], who have studied this particular gene for the past decade. Yet every gene in every species has some sort of regulatory mechanism, and the vast majority of these are unknown. The National Center for Biotechnology Information website contains database entries for the genomes of 3,245 archaea, bacteria and eukaryotes, and the database is increasing by over ten billion base pairs per year.³ As sequence data continue to become available, and high-throughput data-gathering technologies such as microarrays [Schena *et al.*, 1995] and chromatin immunoprecipitation on microarrays (ChIP-on-chip) assays [Ren *et al.*, 2000; Iyer *et al.*, 2001; Lieb *et al.*, 2001] continue to be used, methods for automatic analysis of these data become increasingly important.

My project has been to extend state-of-the-art machine learning approaches so that models are able to represent additional biological concerns, *e.g.* the relational aspects described above, which comprise a key part of understanding the biological systems at hand. These extensions are grounded in biology, and therefore I hypothesize that they will aid the learner in finding more accurate models.

³www.ncbi.nlm.nih.gov/Genbank/genbankstats.html.

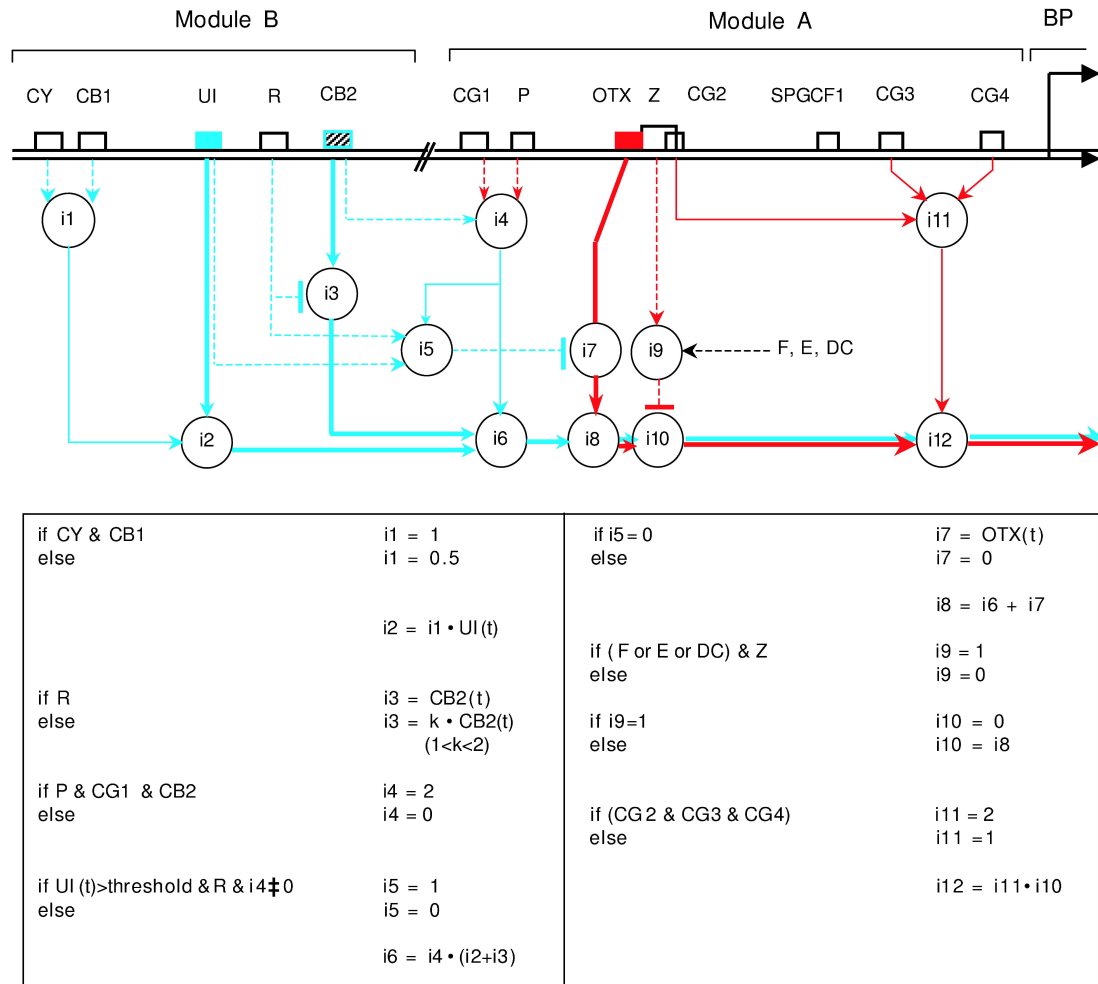


Figure 1.2 Two regulatory modules, A and B, for a gene called *endo16* in the sea urchin. The black double-line represents DNA upstream from the gene. Each protein-binding site is represented with a box. The “logic” schematic at the bottom represents the hypothesized interaction between these sites. Input to the schematic is whether or not regulatory proteins are bound to certain sites, and output at i_{12} represents the strength of the signal to the transcription apparatus. This figure is from Yuh *et al.* [2001]. The details are also summarized elsewhere [Howard and Davidson, 2004].

1.4 Thesis Statement

In the chapters that follow, I will describe in detail the approaches I have taken to various biological tasks, and show (i) how I choose the model space for various tasks so that the computational models are able to represent the biological aspects that we interested in learning, (ii) how I design models and learning algorithms for these tasks to search model space, and (iii) that the expressive, biologically-informative models that I have developed explain observed biological data more accurately than state-of-the-art models which are less expressive.

Specifically, I test the following hypotheses:

- i Models that represent the logical and spatial relationships between binding-site motifs in gene regulatory modules are more accurate than state-of-the-art models which do not.
- ii Models that learn binding-site motifs (*i.e.* DNA letters) in the context of their logical and spatial relationships are more accurate than state-of-the-art models which do not.
- iii Regression models that learn from actual gene expression level directly are more accurate than classifier models that learn from discretized expression patterns.
- iv Models that learn hidden regulator states (*i.e.* activated, inactivated) and roles (*i.e.* activator, repressor) are more accurate than state-of-the-art models which do not.

1.5 Outline

This thesis is organized as follows:

- Chapter 2 gives background information about the relevant biology as well as computational models and algorithms.
- Chapter 3 describes an algorithm for accurately selecting the important *logical* and *spatial aspects* of a *cis*-regulatory module model as well as the motifs.
- Chapter 4 describes an extension to the learning algorithm presented in Chapter 3 which learns spatial aspects motifs *de novo* along with logical and spatial aspects.
- Chapter 5 describes a general-purpose algorithm for learning a mapping from features present in sequential data to a real-valued responses.
- Chapter 6 describes a Bayesian network model for gene regulatory networks that represents the hidden states and roles of regulators.
- Chapter 7 summarizes the key contributions of this work to computer science and bioinformatics, and proposes future work.

Chapter 2

Background

2.1 Biology

The “central dogma of molecular biology” [Crick, 1958; Crick, 1970] characterizes the primary information flows in a cell. Part of this is the way that DNA is transcribed into RNA, which is translated into a protein. This process is illustrated in Figure 2.1. RNA Polymerase (RNAP) “knows” where to start transcribing a gene by recognizing the gene’s *promoter* region. The particular DNA *bases* (letters from the alphabet {a, c, g, t}) that make up the promoter are part of the reason that RNAP is able to recognize it. For example, Figure 2.2 illustrates part of the promoter sequence in an *E. coli* gene. Two hexamers (sequences that are six bases wide), called the -35 and -10 regions because of their relative position to the transcription start site, are known to be important in determining the ability of RNAP to recognize the promoter. These exact hexamers, *ttgaca* and *tataat* in Figure 2.2, are certainly not present in all *E. coli* genes. Different variations of the -35 and -10 hexamers (as well as many other factors) result in different transcription rates.

One way genes are regulated is by controlling the affinity to which RNA polymerase binds to this promoter. Proteins and other molecules called *transcription factors* (TFs) help to control whether or not this happens by binding to DNA in the promoter region.¹ *Activators* are transcription factors that bind to promoters and recruit RNAP to locate them. *Repressors* bind to DNA to prevent RNAP from transcribing a gene. Figure 2.3 shows an illustration of an activator. The general idea is that activators are present in a cell when the products of the genes they activate are needed by the cell.

Many proteins have been identified as transcription factors. What is generally unknown are the transcription factor *binding sites* (TFBS). For instance, Figure 2.3 illustrates a transcription factor binding to the sequence, *gcgatgag*. Transcription factor binding sites tend to be short DNA subsequences, *e.g.* 8-13 bases wide, which vary from instance to instance. Because they vary, they are often represented in computational models by *position weight matrices* (PWMs) [Stormo and Hartzell, 1989]. A PWM represents a binding site as a fixed-length sequence of bases, where each base comes from an independent distribution. An example of a PWM is shown in Figure 2.4.

¹Transcription factors are often, but not necessarily, proteins. RNA molecules can also perform transcriptional and post-transcriptional gene regulation as well, but transcription factors that recognize regulatory binding sites in the promoter are of particular importance in this thesis.

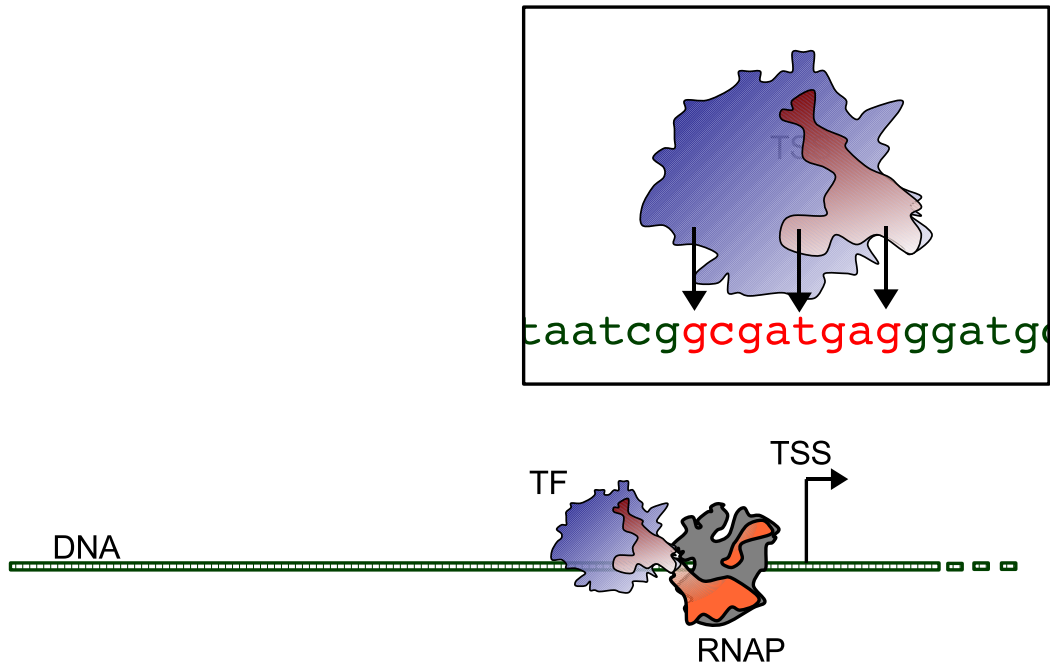


Figure 2.3 Illustration of a transcription factor (TF) protein recognizing and binding to a specific subsequence of characters `gcgatgag` (inset), helping to facilitate the transcription by RNAP.

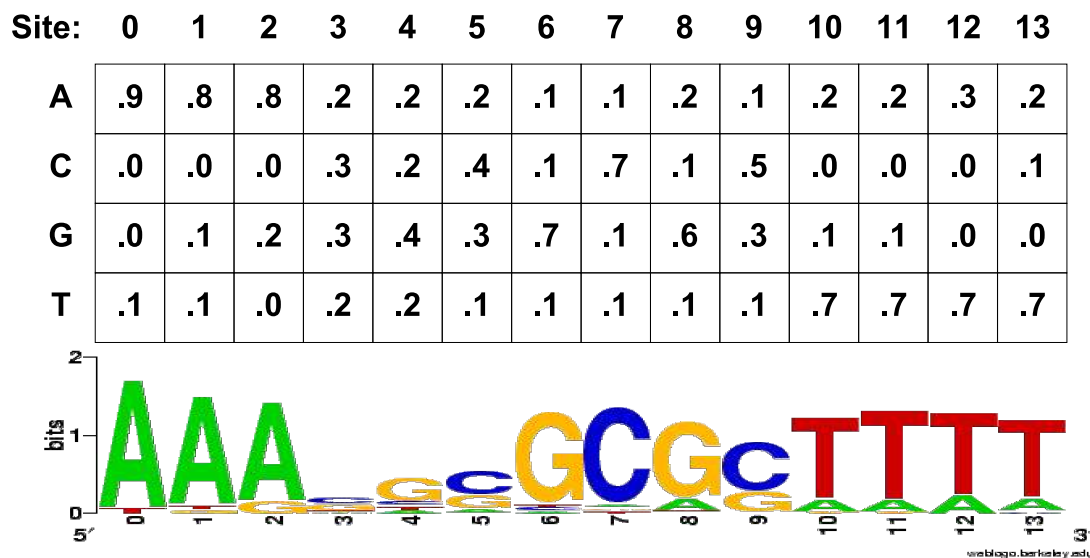


Figure 2.4 An example of a 14-base-wide position weight matrix (PWM). The base at each site, 0-13, is assumed to come from an independent distribution over $\{A, C, G, T\}$. Below the matrix is the *logo* representation of the same distribution [Crooks *et al.*, 2004]. Here, the information content of each site (measured in bits) is represented by the height of the letters, *e.g.* the taller the letter, the more likely it is at that site.

Of course, there can be, and often are, multiple transcription factors that can affect a gene at the same time. The interaction of transcription factors, the variance of the binding sites, and the fact that promoter regions can be quite large (mammalian genes can be regulated by sites hundreds of thousands of base pairs away) make learning models of regulatory systems very difficult.

2.2 Probabilistic Graphical Models

In this section, I give background on two types of probabilistic graphical models which I use in my research, Bayesian networks and hidden Markov models.

2.2.1 Bayesian Networks

Consider a set of events, where each event is described by a set of random variables. In general, these variables are not independent of each other, and each combination of values for these variables has its own probability. The probability distribution over all possible combinations of values for a set of variables is called the *full joint distribution*. Consider four Boolean variables, A, B, C, D . One possible full joint distribution is shown in Table 2.1.

A *Bayesian network* [Pearl, 1988] is encoded as a directed acyclic graph where each node represents one random variable, and edge between nodes represents some dependence between the variables. Each variable \mathcal{X} is associated with a *conditional probability distribution* (CPD), that is, a probability distribution over the values of \mathcal{X} , given the values of the *parents* of \mathcal{X} (the variables represented by the nodes that have edges pointing in to \mathcal{X}). The example Bayesian network in Figure 2.5 represents the same four Boolean variables A, B, C, D in the full joint distribution of Table 2.1. However, the structure of the network conveys information about the conditional independence of the variables, *e.g.* the fact that there is no edge from A to D means that D is *conditionally independent* of A , given the parents of D , B and C :

$$P(D|A, B, C) = P(D|B, C) \quad [2.1]$$

In other words, if you know the value of B and C , information about the value of A will not change your belief about the value of D . This means that Bayesian networks are often a substantially more compact representation of the full joint probability distribution.

2.2.2 Hidden Markov Models

Consider a sequence of events, denoted x_1, x_2, \dots , where each x_i is a character from an alphabet Σ . A *grammar* is a set of rules about which character sequences are legal in a certain language, and which character sequences are not. A *probabilistic grammar* is a model that specifies not only which sequences are legal, but how *probable* a sequence is. Probabilistic grammars are frequently used to explain sequence data.

Table 2.1 Example full joint distribution over four Boolean variables, A, B, C, D .

		A		$\neg A$	
		B	$\neg B$	B	$\neg B$
C	D	0.040	0.047	0.151	0.010
	$\neg D$	0.015	0.039	0.055	0.008
$\neg C$	D	0.105	0.003	0.048	0.000
	$\neg D$	0.101	0.320	0.046	0.008

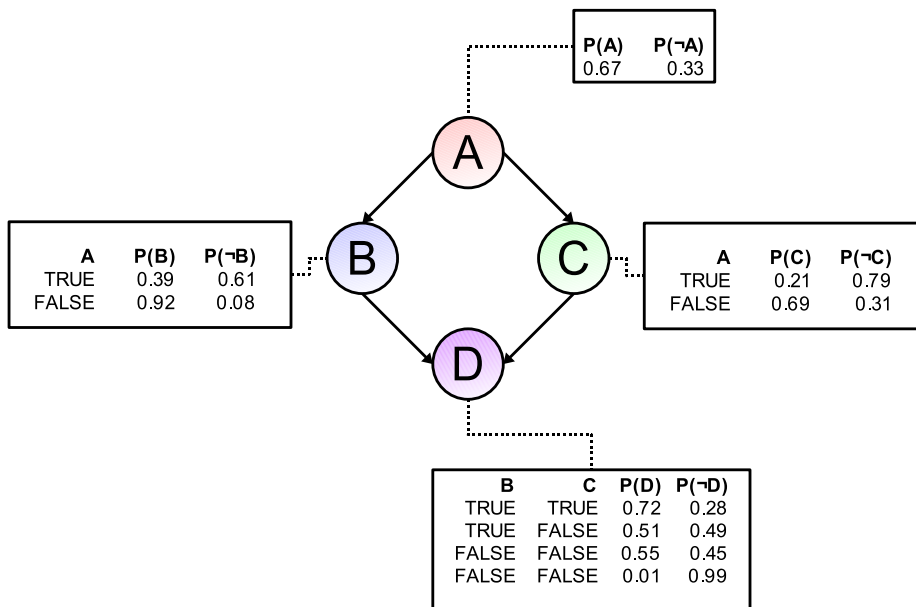
Figure 2.5 A simple Bayesian network model. Each node represents a random variable. In this case, variables are Boolean. A conditional probability table (CPT) shows a distribution over values for each variable, given its *parents* (the nodes that point to it).

Table 2.2 Example parameters for a 3rd-order Markov chain over the alphabet $\Sigma = \{a, c, g, t\}$. There is a probability distribution over Σ for each 3-character history in Σ^3 .

$x_{t-3}, x_{t-2}, x_{t-1}$	$P(x_t = a)$	$P(x_t = c)$	$P(x_t = g)$	$P(x_t = t)$
a,a,a	0.639	0.221	0.102	0.026
a,a,c	0.317	0.488	0.164	0.002
\vdots	\vdots	\vdots	\vdots	\vdots
t,t,t	0.241	0.148	0.008	0.600

The *Markov assumption*, named for Russian mathematician Andrei Markov, is that the value for x_{t+1} depends only on a finite history of the previous n events, where n is called the *order* of the Markov assumption, *i.e.*

$$P(x_t | x_1, x_2, \dots, x_{t-1}) = P(x_t | x_{t-n}, x_{t-n+1}, \dots, x_{t-1}). \quad [2.2]$$

A straightforward type of model that makes a Markov assumption is a *Markov chain model*. An n th-order Markov chain model keeps a separate probability distribution over characters from Σ for each of the $|\Sigma|^n$ possible histories of n characters. Table 2.2 shows the parameters of a 3rd-order Markov chain model. The standard way to learn the parameters Θ of such a model is to set them to maximize the likelihood of the sequences in a training set \mathbf{D} :

$$\hat{\Theta} = \arg \max_{\Theta} \prod_{\mathbf{x} \in \mathbf{D}} P(\mathbf{x} : \Theta) \quad [2.3]$$

Doing this is straightforward. One just scans the training set and counts all the relevant events:

$$P(x_t = \sigma | x_{[t-n..t-1]}) = \frac{n(x_{[t-n..t-1, \sigma]})}{n(x_{[t-n..t-1]})} \quad [2.4]$$

where $x_{[i..j]}$ represents the subsequence $x_i, x_{i+1}, \dots, x_{j-1}, x_j$ and $n(\mathbf{x}_{[i..j]})$ is the number of times a given subsequence occurs in the training set.

A hidden Markov model (HMM) is encoded as a set of nodes $\mathbf{Q} = \{q_1, q_2, \dots, q_N\}$ and directed edges. Each node represents a *state*, which “emits” exactly one character from Σ , according to its own probability distribution. The edges represent probabilistic transitions between nodes. This makes an HMM a probabilistic finite state automaton. An example of an HMM is shown in Figure 2.6. Consider how a sequence would be generated by such a model: Start in the **Start** state, choose the next state probabilistically from the outgoing transitions, generate a single character from the new state’s distribution, and repeat, eventually ending up in the **End** state.

The series of states visited in the process is called the *path* taken through the model. A character sequence is explained by a path through the model. Since HMMs can generate data from the distribution they represent, they belong to a class of models called *generative* models. It is useful to think of an HMM as stochastically generating sequence data, but the typical use of HMMs is

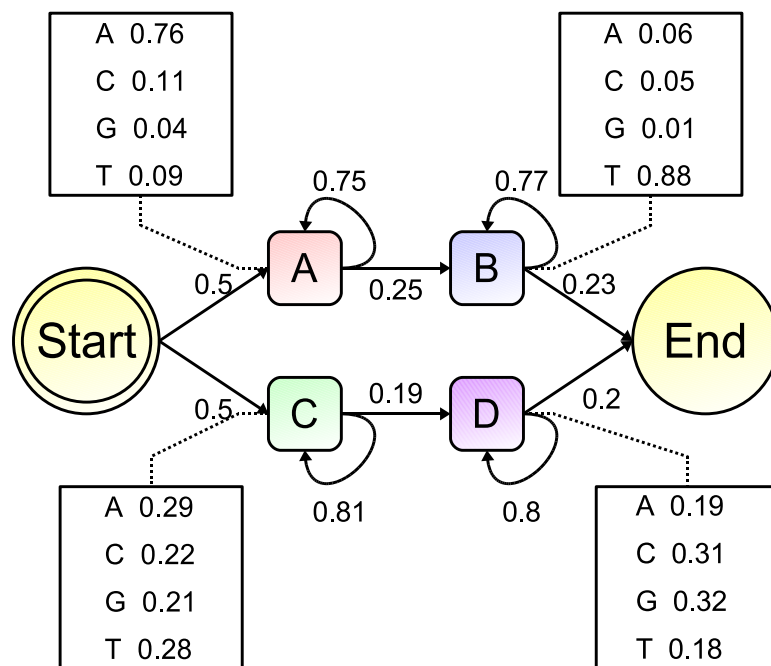


Figure 2.6 A simple hidden Markov model (HMM). Each *state* (nodes labeled A, B, C, and D) emits one character at a time according to the given probability distributions, and then transitions to a state (possibly itself), according to the given transition probabilities. A sequence of characters is explained by a *path* from the Start state to the End state.

to *explain* data that are already observed, not to generate new data. We consider the likelihood that a sequence was generated by a certain path through the model as a measure of how well the path explains the sequence. Consider the sequence, `aacaattttt`, as being explained by the HMM in Figure 2.6. Because of the emission parameters, it is clearly more likely to have been generated by the states **A** and **B**, but it could have been generated by states **C** and **D**. The fact that in many cases we do not know which path is the “correct” one is what is hidden about HMMs. What is Markovian about HMMs is that the probability of a transition between two states, $q_k \rightarrow q_l$, is independent of the states visited previous to q_k .

Since HMMs involve hidden information, using them and training them is more complicated than in the case of Markov chain models. There are three key questions we want to answer with our models:

1. What is the likelihood of a given sequence?
2. What is the most likely path that explains a given sequence?
3. What are the optimal parameters for a model, given a training set of sequences?

There are HMM algorithms to answer these questions, and I will address each in turn. To calculate the likelihood of a given sequence, we use the *forward* algorithm. This is explained very well elsewhere [Durbin *et al.*, 1998], but I will outline it here, because I will describe modifications to it in later chapters. Given a path π through the HMM, it is easy to calculate the likelihood of a sequence simply by multiplying the relevant parameters:

$$P(\mathbf{x}|\pi) = \prod_{i=1}^L a_{\pi_{i-1}, \pi_i} \times e_{\pi_i}(x_i) \quad [2.5]$$

where π_i is the state that explains the i th character, L is the length of \mathbf{x} , a_{kl} is the probability of the transition from state k to state l , and $e_l(x_i)$ is the likelihood of state l emitting the character x_i . However, in general, there are exponentially many paths through a model, so they cannot be examined individually. The main idea of the forward algorithm is to sum up the probabilities over all paths, using *dynamic programming* (DP). The forward algorithm fills in a DP matrix f , shown in Table 2.3. Each cell is indexed by an HMM state (row heading), and sequence index (column heading). Each cell $f_k(i)$ represents the likelihood of the sequence in question \mathbf{x} , up to the index i , given that the path that explains \mathbf{x} goes through state k at time i , *i.e.* $f_k(i) = P(\mathbf{x}_{[1..i]}|\pi_i = q_k)$, where π is the path through the model, and π_i represents the state in the path that explains character x_i . The value in each cell is calculated one column at a time, and it is updated from the previous column.

$$f_{\text{Start}}(0) = 1 \quad [2.6]$$

$$f_{l \neq \text{Start}}(0) = 0 \quad [2.7]$$

$$f_l(i) = \sum_k f_k(i-1) \times a_{kl} \times e_l(x_i) \quad [2.8]$$

Table 2.3 A forward algorithm dynamic programming matrix f for sequence of length L . Each cell, $f_k(i)$ represents the probability of being in state q_k after explaining i characters.

	$0 \dots i - 2$	$i - 1$	i	$i + 1 \dots L$
q_1				
q_2				
\vdots				
q_k		$P(\mathbf{x}_{[1..i-1]} \pi_{i-1} = q_k)$		
\vdots				
q_l			$P(\mathbf{x}_{[1..i]} \pi_i = q_l)$	
\vdots				
q_N				

The DP calculations effectively sum up the likelihood of \mathbf{x} over all possible paths, taking advantage of the Markovian nature of the model: Once the path has reached state k , it does not matter how it got there. Once the matrix is completely filled in, the likelihood of the sequence is given by $f_{\text{End}}(L)$, where L is the length of the sequence.

Equation 2.8 assumes each HMM state q_l emits exactly one character. This is a common assumption used for many applications, but it can be relaxed in favor of *generalized* Markov models [Rabiner, 1989; Burge and Karlin, 1997], where each state emits zero or more characters, and the choice of the number of characters can come from an arbitrary probability distribution. These models are sometimes called hidden *semi*-Markov models, because the probability of transiting out of a state depends on the history (number) of characters already emitted by the state, but not on the path taken through the model before entering the state. Doing the forward calculations for hidden semi-Markov models is slightly more complex, and the recurrence relation is

$$f_l(j) = \sum_k \sum_i f_k(i) \times a_{kl} \times e_l(x_{[i\dots j]}) \times d_l(j - i) \quad [2.9]$$

where $e_l(x_{[i\dots j]})$ is the probability of emitting the sequence $x_{[i\dots j]}$, and $d_l(n)$ is the probability of emitting exactly n characters in state q_l . Doing inference using Equation 2.8 is $O(Q^2L)$ for Q states and a sequence length L . Note that doing inference with Equation 2.9 is more costly, $O(Q^2L^2)$. In Chapter 4, I discuss ideas for speeding up these dynamic programming calculations.

To answer the next question, to find the single most likely path, we use a similar DP algorithm called the Viterbi algorithm [Viterbi, 1967], which calculates the most likely path, $\hat{\pi}$

$$\hat{\pi} = \arg \max_{\pi} P(\mathbf{x}|\pi). \quad [2.10]$$

The update step for Viterbi is similar to forward, except instead of summing up probabilities, it finds the most likely path:

$$v_{\text{Start}}(0) = 1 \quad [2.11]$$

$$v_{l \neq \text{Start}}(0) = 0 \quad [2.12]$$

$$v_l(i) = \max_k v_k(i - 1) \times a_{kl} \times e_l(x_i). \quad [2.13]$$

Of course, we need to keep a back-pointer to remember the most likely π_{i-1} for each π_i . We use these back-pointers to trace the most likely path from $v_{\text{End}}(L)$ to $v_{\text{Start}}(0)$.

When there is hidden information (*i.e.* which state emits which character in a path through the model), there is no closed form expression to answer the third question, to find the parameters $\hat{\Theta}$, that maximize the likelihood of a set of sequences \mathbf{D} :

$$\hat{\Theta} = \arg \max_{\Theta} \prod_{\mathbf{x} \in \mathbf{D}} P(\mathbf{x} : \Theta). \quad [2.14]$$

(Equation 2.4 shows a closed-form expression when there are no hidden data.) The Baum-Welch algorithm [Baum, 1972] is an expectation-maximization (EM) algorithm [Dempster *et al.*, 1977]

that iteratively updates the parameters in an attempt to maximize the likelihood of the data. Krogh [1994] proposed an EM algorithm to set the parameters in an attempt to maximize the likelihood of the *labels* associated with sequences:

$$\hat{\Theta} = \arg \max_{\Theta} \prod_{(x,y) \in \mathcal{D}} P(y|x : \Theta). \quad [2.15]$$

I use variants of both of these algorithms in the models presented in Chapters 3, 4, and 5.

2.3 Evaluating Models

After a machine learning algorithm has trained a model, how do we show that the model is a good one? That is, how do we measure the accuracy of a learned model? We can always measure the error rate on the training set, but this might not reflect what we really want to measure, which is how accurate model will be on *future* examples. In other words, how well does the model generalize?

To measure the model's accuracy on future examples, we hold aside a *test set* before we begin training. The learner does not use the test set during the training process. Once the model is learned, we use the test to evaluate the accuracy of the model. We do this by letting the model look at the features of each test example and predict its label, and then comparing the predicted label to the actual label.

Often data are scarce, and it is expensive to hold aside data for the purpose of testing. To get the most out of the data, one technique is called *N-fold cross-validation*. Here, we run the learning algorithm *N* times. Each time, we hold aside a different test set. This gives us *N* different accuracy measures, which can be added together. The most extreme case is called *leave-one-out* cross-validation where each test set is a single example. This maximizes the training set size each time and is therefore used for small data sets.

Pseudocode illustrating the process of test-set cross-validation is shown in Table 2.5. The accuracy metric given to the function takes a set of (*class*, *prediction*) pairs and returns a numerical score. Table 2.4 defines the four possible (*class*, *prediction*) pairs for a binary classification problem: True positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). A simple example of a possible metric function would be classification accuracy:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad [2.16]$$

where *TP* (*FP*, *etc.*) denotes the number of true positives (false positives, *etc.*) in the set of (*class*, *prediction*) pairs.

However, simple test set accuracy may not reflect how well the learned model is really making predictions. Consider the case when a vast majority of the test set examples are negative. If a model were to predict *everything* is negative, it would have a high accuracy, without really learning anything about the examples. *Precision* and *recall* are two metrics that are often used in place of accuracy for situations such as the one given above. *Precision* is a measure of the accuracy of the positive predictions,

Table 2.4 Binary Confusion Matrix Definitions

		Actual	
		\oplus	\ominus
Predicted	\oplus	TP	FP
	\ominus	FN	TN

$$P = \frac{TP}{TP + FP} \quad [2.17]$$

and *recall* is a measure of how often the positive examples classified as such by the model,

$$R = \frac{TP}{TP + FN}. \quad [2.18]$$

Note that there is a tradeoff between precision and recall, which is especially clear if the model is able to vary a “confidence threshold” above which an example is predicted to be positive.

It is often useful to sum up a score in a single number. A common metric which combines precision and recall is *F1*, which is the harmonic average of precision and recall:²

$$F1 = \frac{2 \times P \times R}{P + R}. \quad [2.19]$$

There are cases (*e.g.* disease diagnosis or intruder detection) when the cost of a false positive far outweighs the cost of a false negative. In these cases, we may wish to use a more task-specific scoring metric.

Evaluating the accuracy of a model is something that is often of use to a learner (say, for choosing a key parameter setting), but the accuracy of test-set statistics would be sacrificed if the learner made decisions after looking at the test set. Some algorithms therefore hold aside a further *tuning set* from the training data set, which is used to gauge accuracy using variants of the learned models. Table 2.6 illustrates this process. A learning algorithm uses both a tuning and testing set by calling TUNE-LEARN in place of LEARN on line 5 of CROSS-FOLD-VALIDATION.

²Harmonic average is the reciprocal of the average reciprocal. This means that if either precision or recall get close to zero, so does the F1 score. The “1” in “F1” represents the fact that precision and recall are weighted equally (*i.e.* a ratio of 1). If the user wishes to add more weight to precision (for instance because not all positive example sequences are expected to contain the CRM), then she may choose to use the harmonic average of precision times β and recall.

Table 2.5 Pseudocode for test set N -fold cross-validation. The CROSS-FOLD-VALIDATION procedure takes *Algorithm*: A learning algorithm, \mathbf{D} : A data set, N : The number of folds, and METRIC: A metric for scoring a set of (*class, prediction*) pairs, and returns the accuracy of the test set predictions according to the given metric.

```

CROSS-FOLD-VALIDATION(Algorithm,  $\mathbf{D}$ ,  $N$ , METRIC )
1   $P \leftarrow \{ \}$ 
2  for  $i \in [1 \dots N]$ 
3     $\mathbf{D}_{\text{Test}} \leftarrow$  Every  $N$ th example in  $\mathbf{D}$ , starting with  $i$ 
4     $\mathbf{D}_{\text{Train}} \leftarrow \mathbf{D} - \mathbf{D}_{\text{Test}}$ 
5     $model \leftarrow$  LEARN(Algorithm,  $\mathbf{D}_{\text{Train}}$ )
6    for  $(\mathbf{x}, y) \in \mathbf{D}_{\text{Test}}$ 
7       $y' \leftarrow$  PREDICT( $model, \mathbf{x}$ )
8       $P \leftarrow P \cup (y, y')$ 
9  return METRIC( $P$ )

```

Table 2.6 Pseudocode for the procedure to select an algorithm variant using a *tuning set*. TUNE-LEARN takes *Algorithm*: A learning algorithm with a few variants, $\mathbf{D}_{\text{Train}}$: A training data set (not including any testing set), M : The number of folds, and METRIC, a scoring metric, and returns a model trained with the best variant to use on the data set, according to the metric.

```

TUNE-LEARN(Algorithm,  $\mathbf{D}_{\text{Train}}$ ,  $M$ , METRIC )
1   $best\_score \leftarrow -\infty$ 
2   $best\_variant \leftarrow \mathbf{NULL}$ 
3  for  $variant \in \text{VARIANTS-LIST}(\textit{Algorithm})$ 
4    for  $i \in [1 \dots M]$ 
5       $P \leftarrow \{ \}$ 
6       $\mathbf{D}_{\text{Tune}} \leftarrow$  Every  $M$ th example in  $\mathbf{D}_{\text{Train}}$ , starting with  $i$ 
7       $\mathbf{D}_{\text{Train}'} \leftarrow \mathbf{D} - \mathbf{D}_{\text{Test}}$ 
8       $model \leftarrow$  LEARN(Algorithm,  $\mathbf{D}_{\text{Train}'}$ )
9      for  $(\mathbf{x}, y) \in \mathbf{D}_{\text{Tune}}$ 
10        $y' \leftarrow$  PREDICT( $model, \mathbf{x}$ )
11        $P \leftarrow P \cup (y, y')$ 
12     if METRIC( $P$ ) >  $best\_score$ 
13       then  $best\_score \leftarrow$  METRIC( $P$ )
14          $best\_variant \leftarrow variant$ 
15  return LEARN( $best\_variant, \mathbf{D}_{\text{Train}}$ )

```

Chapter 3

Learning the Logical and Spatial Aspects of a *cis*-Regulatory Module

3.1 Introduction

As described in Chapter 2, gene transcription is controlled by multiple factors, often proteins, whose sole purpose is to act as regulators. These proteins may need to bind to DNA in a specific arrangement in a gene's transcriptional control region. The DNA motifs to which these factors bind are often unknown, and may appear anywhere in a large region in the neighborhood of a gene. In eukaryotes, this region typically extends several thousand base pairs upstream of the transcription start site, and may also include DNA between the transcription start site and the start codon, and within introns of the transcribed gene. Even in simpler eukaryotes, such as yeast, this region cannot be limited to fewer than hundreds of base pairs. It is often the case that a set of genes are transcribed or expressed together under certain conditions, but the mechanisms underlying this co-expression are unknown. We would like a method that can aid in verifying that these genes are indeed transcribed by a common mechanism, and, more importantly, to explain this fact by finding the *cis*-regulatory module (CRM) that promotes transcription.

A module consists of binding sites corresponding to multiple interacting transcription factors, and the hypothesis motivating this research is that the *relationships* between binding sites (relative locations, distance between adjacent binding sites, *etc.*) are important considerations for a CRM model. These are considerations that current state-of-the-art models are unable to adequately represent.

Consider the data set shown in Figure 3.1. The task is to discover a model that distinguishes the positive examples (p_1 through p_8) from the negative (n_1 through n_8). A rule that does this is:

A sequence is positive if and only if it contains two binding sites relatively close together (within about 35 base pairs). The first is upstream, represented by a sequence close to `gcgatgag` or `gcgattgag`. The second is downstream, represented by a sequence close to `gccatggc`.

Note that this rule is true of all the positive instances (p_1 through p_8), but none of the negative instances because they do not contain all the required binding sites (n_1, n_2, n_5, n_7, n_8), the binding sites are in the wrong order (n_3 and n_4), or they are too far apart (n_6).

P_1 ...accgcgctgaaaaaatttgcgatgagtttagaagagtcaccagccatatggcatcagcctactgggttctctgtgtgtgctaccgatcattgacccgctgttgtta...
 P_2 ...aaagaaaaaaaaaaaaagaaaaagaaaaagagattgagaaaaatatgaaaagccatatcgccttggttotgaaagaagcgatgagatgactcatagtaacata...
 P_3 ...caaaattattcaagaaaaaaagaaatgttacaatgaatgcaaaagatggcgatgagataaaagcgagagatgccatatggcagcttaaatgatctggctcaagcagt...
 P_4 ...ggcgcgagaaactgaacggggtgacgcactgcaaatttttcatgttttacatactcaagcattacagactccgagattgagatgagccatatggccagggaaacca...
 P_5 ...gagctggaaaaaaatttcaaaagaaaaagcgatgagataactaatgctgccatatggcaggtataaagtaacgaattggggaaaggccatcaatccaaagtgc...
 P_6 ...aaccatcattaagcgatgagccttgaaaaaagccatatggccatgagaagaaaaagaaatagttaggcttaagtgtctgtattgatcaattaattactttaccact...
 P_7 ...ccatttttctctctttatcacacattcaaaagaaagaaaaaatataccccagctagttaaagaaaaatcattgagcgattgagagaaagccatatggccagag...
 P_8 ...cgatgagttcactaaaaagccatatggcctttccaactgagaagatcattgattattacggttttgatgagatgagatgagatgtaattatcatcctctattgcc...

n_1 ...ggaaactcgtgaatataaaacccgatgaatgtagttgtagcagatgttcggttacagttattattatctgtttcttggccatatggcactagtcacaccgocgag...
 n_2 ...aaaaacaaacaaaataaaacaaacaaaataaatgcagctttaaatacaatattcggcgaagcaagtaacggttgctagcggttttcccatcactactattacactc...
 n_3 ...gatcattgataagaagtgttaagtatcggctgcagcaagtgcataaaaaagccatttggcacttagtttagatgagatcaggtagatcaagaaccttcggtgagccaa...
 n_4 ...cgtttgctttgtttgcttttagttttcttccctgccatatggcgtttacatatggaacgagatgagacacctaaattcaagatcattgataaatataaaga...
 n_5 ...atttacaccacatgtaaaaaaacgtacaaaaagccaaatatttccaactcaatgcatgaattataggtattttaagttatattgctgacgtaaaattcaga...
 n_6 ...cgtttaaatcttcgcatagatcagggagcgatgagacgttataagattgagtggaagaacttccattcttcaggtgctgttattctttgccatatggccttagc...
 n_7 ...attgggaagcaataaattgcatacccaaaacaaaccagacatcatttaattgtttgaaacataaaatggtggaacttaaaaatatataaaaggagagattgaaatcagc...
 n_8 ...acgctataaaaaaatgcacagataggaccttaaggactaataaaaccgacgatcattgagtaatgctcgtgggttttcaaccttagtagttttggttaatttta...

Figure 3.1 An example of a CRM learning task. The sequences labeled p_i are positive sequences, and the sequences labeled n_i are negative sequences. Instances of identified motifs are highlighted.

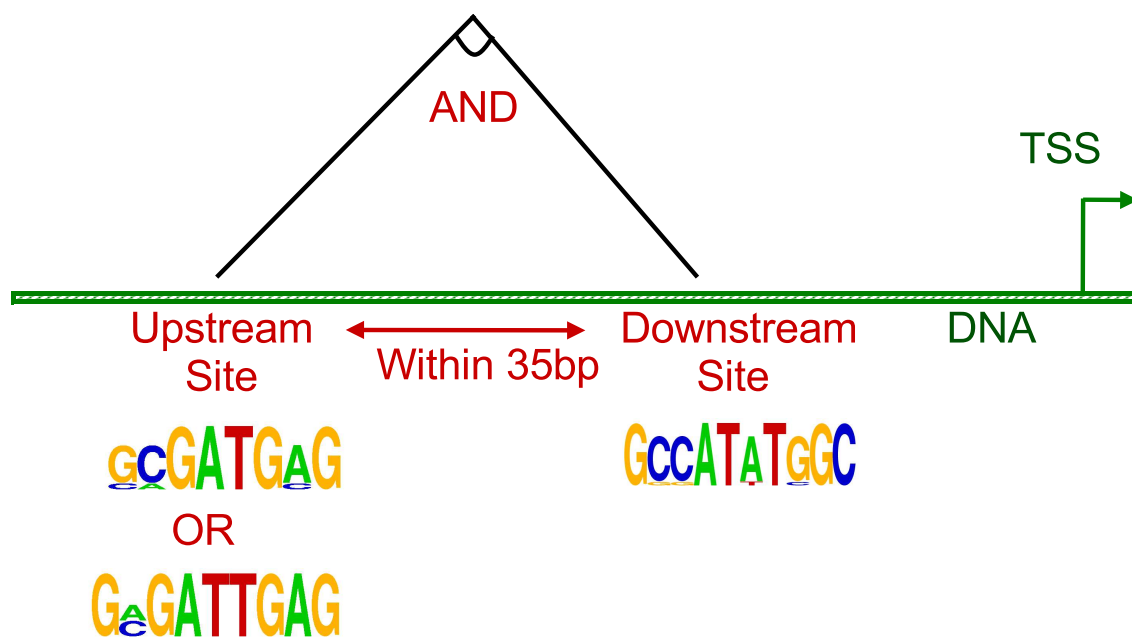


Figure 3.2 An example CRM model that corresponds to the task in Figure 3.1. The double-line represents DNA and the location of the transcription start site (TSS) is indicated. Motifs are represented here by sequence logos [Crooks *et al.*, 2004] (recall that the height of each letter indicates its prevalence in the motif). The structure represents the logical aspects of the model ($(\text{gcgatgag} \vee \text{gcgattgag}) \wedge \text{gccatatggc}$), and there are also order constraints (upstream, downstream specifications) and distance constraints between the two binding sites.

This rule is represented by the model shown in Figure 3.2. This model consists of three motifs. The first two (gcgatgag, gcgattgag) are connected by the logical OR operator, and represent a single binding site. One other motif (gccatatggc) represents a second binding site. These two binding sites are connected by the logical AND operator (which means both are necessary for the module to be present). Also, the order between the two binding sites is constrained (the disjunction must be upstream) and the distance is constrained (both binding sites must be within 35 base pairs of each other). Note that I am making a distinction between *motifs* and *binding sites*; a disjunction of multiple motifs can represent one binding site.

The learning task we consider in this chapter is to learn such a CRM model from data:

Given:

- (i) Positive sequences believed to share a CRM,
- (ii) Negative sequences believed not to contain the CRM,
- (iii) A set of candidate binding-site motifs.

Learn: A model of the CRM, including the logical and spatial relationships that characterize it.

I refer to the binding-site relationships considered in this chapter as structural *aspects*, which are divided into two categories, as follows.

Logical aspects:

- AND: Multiple required binding sites. The basic concept of a CRM.
- OR: A binding site can be represented by multiple motifs. My representation includes this because different transcription factors may play the same role in a CRM, and because a single transcription factor may have multiple or varying binding sites.
- NOT: A binding site that must *not* appear in the target CRM (*e.g.* it is a repressor).

Spatial aspects:

- Order: A certain binding site must appear upstream of another (or *not* appear in the case of negated binding sites).
- Distance: Two binding sites must appear within a certain distance (measured in base pairs). Also, the CRM must appear within a certain distance of the transcription start site (which may be known or estimated).
- Strand: A binding site must appear in a certain orientation (*i.e.* on a certain DNA strand, sense or antisense).

All of these aspects are represented in the example model shown in Figure 3.3.

In order to make learning possible in such an expressive model space, I have developed a specialized learner which has two important distinctions. First, the search process is specifically

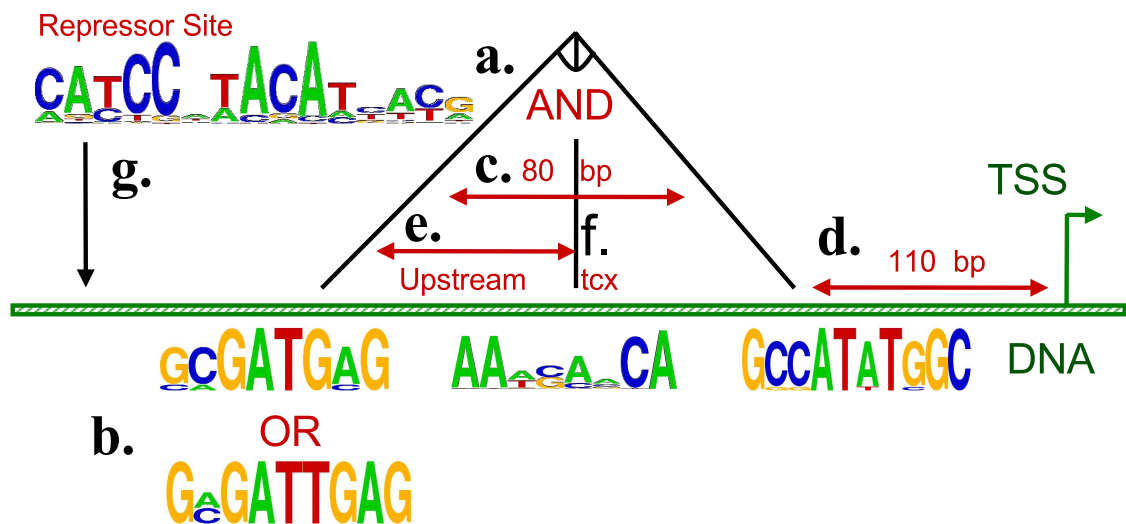


Figure 3.3 An example CRM model, which shows all the logical and spatial aspects considered in my approach. **a.** This model has three necessary binding sites. **b.** One of the binding sites can be satisfied by matching either of two motifs. **c.** One binding site must be within 80bp of another, and **d.** the CRM must be within 110bp of the start of transcription (TSS). **e.** One binding site must appear upstream of another. **f.** One of the binding sites must appear on the antisense DNA strand (*tcx*). **g.** One motif is *negated* (labeled “Repressor Site”) and therefore must *not* appear upstream of the other binding sites.

tailored for the context of *cis*-regulatory modules. Second, I have developed a method for learning the appropriate level of model expressiveness for the task in question. Although the expressivity capable of capturing all these physical aspects of a CRM is a major strength of my approach, only a few of these aspects may actually be needed to describe a given CRM. Therefore, there is a risk of overfitting due to this high-variance model space. For this reason, I have developed an expressivity selection method in which each aspect of the model space must be statistically justified by the data.

3.2 Related Work

There are several current approaches for finding CRMs from sequence data, *e.g.* [Aerts *et al.*, 2003; Sharan *et al.*, 2003; Sinha *et al.*, 2003; Beer and Tavazoie, 2004; Segal and Sharan, 2004; Zhou and Wong, 2004; Gupta and Liu, 2005; Mahony *et al.*, 2005; Philippakis *et al.*, 2005; Sidharthan *et al.*, 2005; Macisaac *et al.*, 2006]. These all represent CRMs as a set of over-represented sequence motifs. A few of them use models that characterize a limited set of relationships between motifs like the ones described in the previous section.

For instance, some previous methods have characterized CRMs as a set of motifs that appear within a window of predetermined size [Aerts *et al.*, 2003; Segal and Sharan, 2004; Zhou and Wong, 2004]. The models of Segal and Sharan [2004], shown in Figure 3.4, learn CRMs using a Bayesian network. The motifs must all appear within the same window of space (a few hundred base pairs). Similarly, the models of Zhou and Wong [2004], shown in Figure 3.5, represent CRMs as a set of motifs that appear within a fixed-size window. In both of these cases, CRM models are able to capture the *distance* (more specifically, an upper-bound on the distance) between motifs, although not the relative ordering or orientation. Zhou and Wong do suggest a method for guessing this window size before learning the CRM motifs, but neither of these cases involves selecting a window size based on the actual observed distances between motif occurrences. This means that the window size parameter is not an informative part of the learned models, even if it does increase predictive accuracy.

The models of Sinha *et al.* [2003], shown in Figure 3.6, use what they call a “history-conscious” hidden Markov model (hcHMM). This model learns the probability that a given motif j will appear in a sequence, given that the previous motif was i . In this way, their models are able to represent the *order* of motifs. That is, if $P(A, B)$ represents the probability that motif B will directly follow motif A , then learning that $P(A, B) > P(B, A)$ represents the fact that the order, motif A followed by motif B , is more representative of a CRM than the opposite order. Similarly, the models of Gupta and Liu [2005] keep track of the likelihood of adjacent motifs by keeping a square matrix of pairwise dependencies. These models do not, however, capture order dependencies beyond adjacent motifs. That is, if motif A is very likely to appear upstream of C , this information is lost if motif B appears between them. Also, since these approaches use hidden Markov models (not hidden *semi*-Markov models), the distance between them is represented implicitly as a geometric distribution, which may or may not be able to represent this relationship accurately.

The models of Keleş *et al.* [2004], shown in Figure 3.7, can represent *logical* relationships between motifs. Their approach searches for a set of logical sentences (the trees in Figure 3.7) that

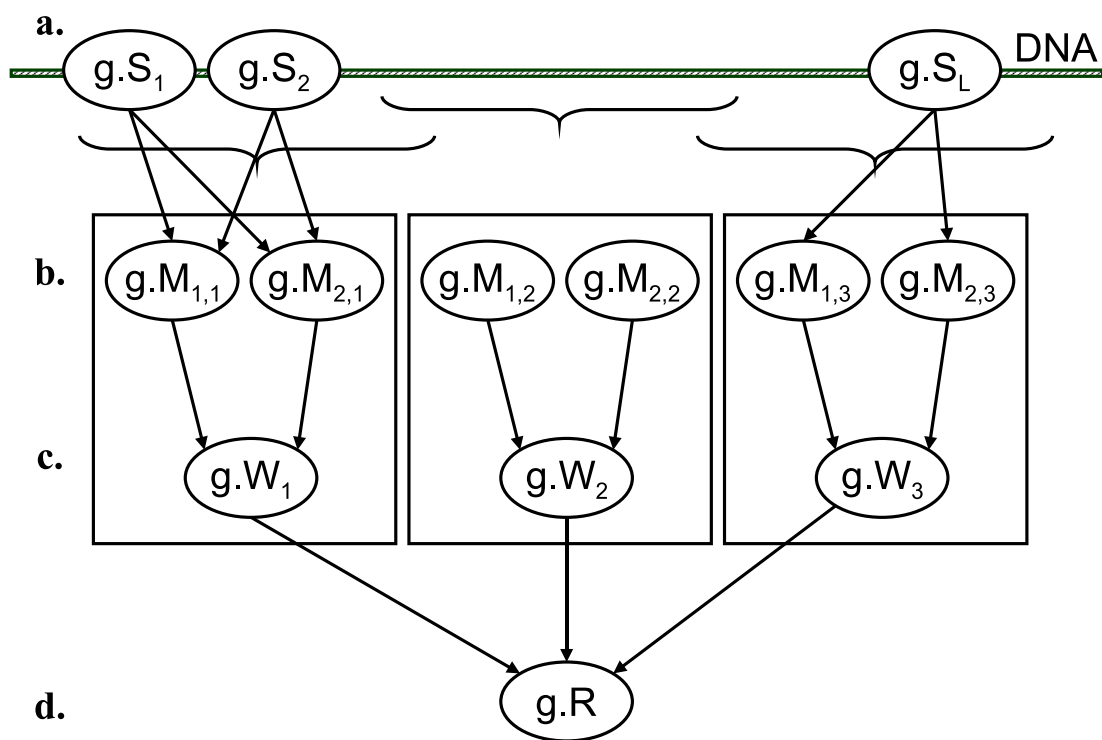


Figure 3.4 The Bayesian network CRM model of Segal and Sharan [2004]. The input DNA sequence is divided into a set of overlapping windows of a fixed size. **a.** Observed sequence variable $g.S_i$ takes on a value from $\{A, C, G, T\}$ corresponding to sequence position i . **b.** Motif occurrence variable $g.M_{m,w}$ is true if and only if motif m appears in window w . **c.** CRM occurrence variable $g.W_w$ is true if and only if the CRM appears in window w . **d.** Regulated variable $g.R$ is true if and only if the CRM appears in any window, *i.e.* the gene is regulated if and only if the CRM appears in the sequence. Note that not all variables and edges are shown here.

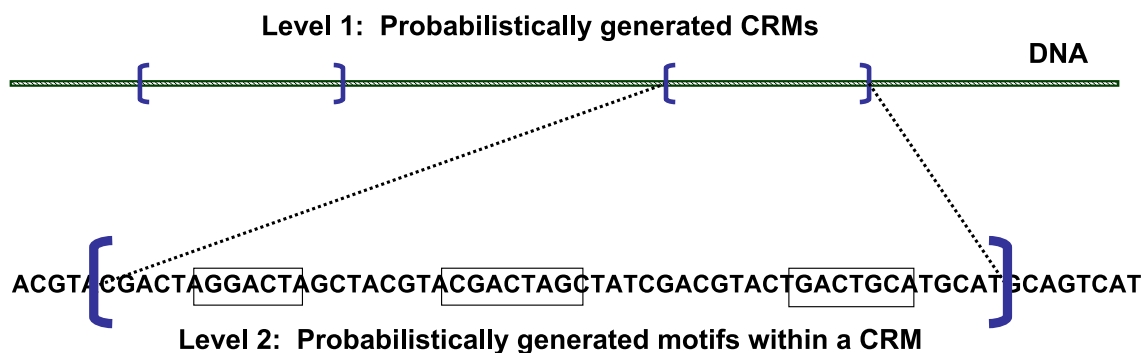


Figure 3.5 The hierarchical CRM model of Zhou and Wong [2004]. At the top level, a CRM is a fixed-length subsequence of DNA interspersed with “background” DNA. At the second level, a CRM is a set of over-represented motifs interspersed with background DNA.

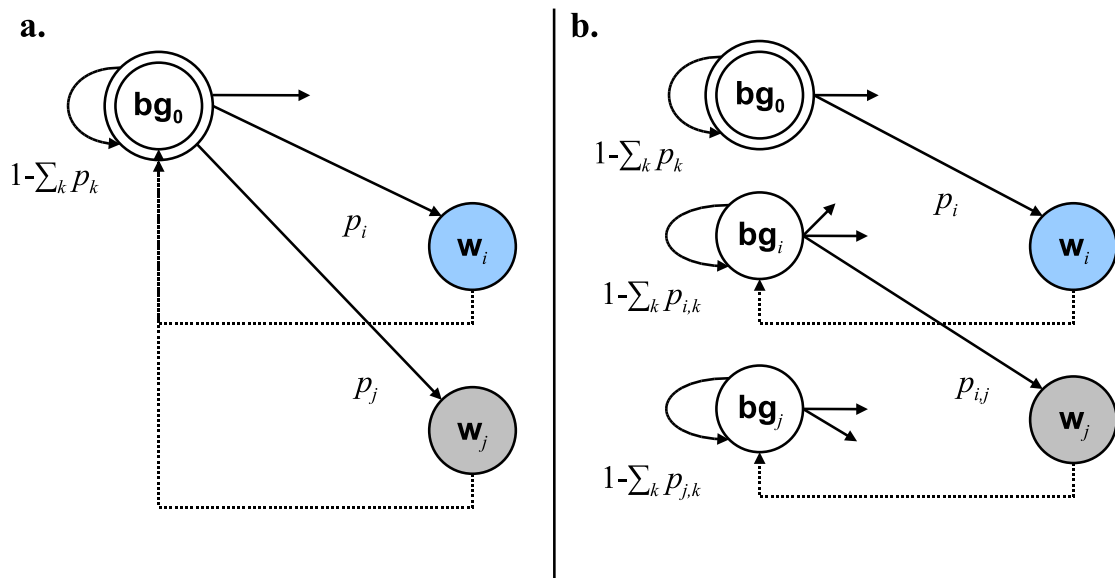


Figure 3.6 The hidden Markov CRM models of Sinha *et al.* [2003]. Motifs are explained by states labeled w_i and non-motif DNA is explained by “background” states labeled bg . Probabilistic transitions are represented by solid lines and fixed (*i.e.* probability=1) transitions are represented by dashed lines. **a.** The HMM model from Rajewsky *et al.* [2002] which is called HMM0 by Sinha *et al.* The probability of motif i is p_i . **b.** The history-conscious HMM of Sinha *et al.* The model has parameters $p_{i,j}$ for each possible pairwise combination of motifs. In this way, it represents the order between two motifs. The bg_k states emit background sequence from the same distribution, but “remember” the most recently encountered motif.

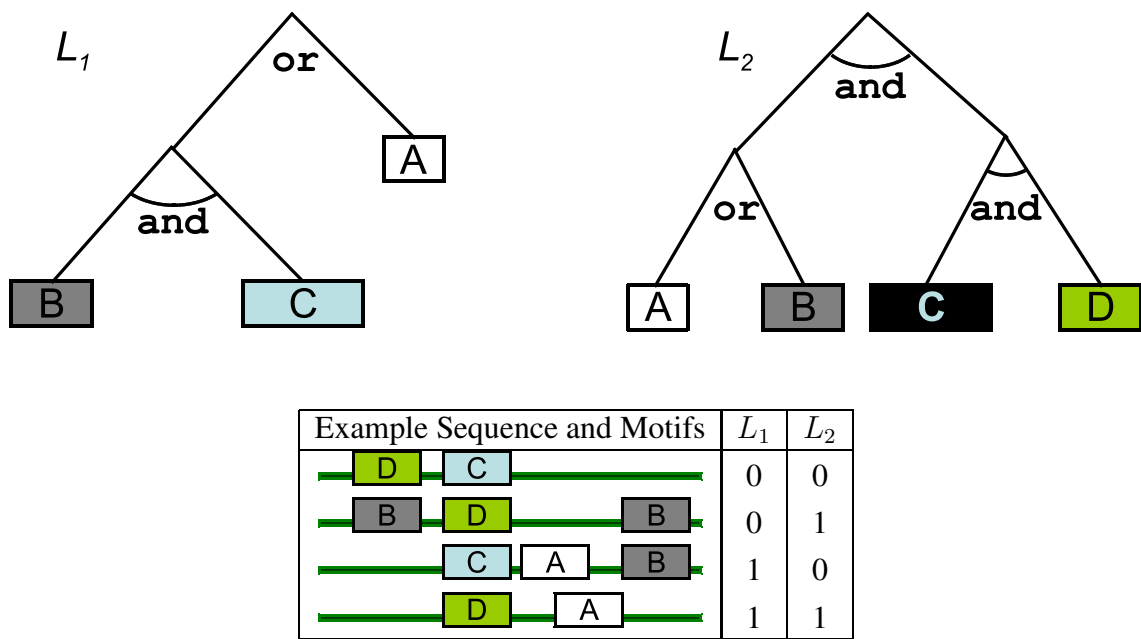


Figure 3.7 The Logic Regression CRM Model of Keleş *et al.* [2004]. L_1 and L_2 represent logical sentences $((B \wedge C) \vee A$ and $(A \vee B) \wedge (\neg C \wedge D)$, respectively). The table shows the truth values of L_1 and L_2 for a few example sequences which include labeled motifs occurrences. The algorithm searches for candidate logical sentences and uses a novel approach called “logic regression” to learn their relevance.

characterize the CRM of a given set of sequences using a novel approach called logic regression. Although these models are able to represent all the same logical aspects as described above, they do not represent any of the spatial aspects.

The approach of Beer and Tavazoie [2004] does capture motif orientation, and the relative order and distance between pairs of motifs. However, I would argue that the models presented in this chapter are more comprehensible than the models of Beer and Tavazoie.

3.3 Approach

My algorithm learns a CRM model from positive and negative example sequences and a set of potential binding site motifs. The evaluation function, which measures the accuracy of a given model on the training/tuning set, is also part of the input.

Positive examples are those believed to contain a shared CRM (*i.e.* a set of particular binding sites and structural aspects). These may be, for instance, the set of promoter sequences from a set of genes that are co-expressed under certain conditions and suspected to be co-regulated.

Negative examples are those believed not to contain the target CRM, although they certainly may contain other arrangements of motifs. The purpose of these sequences is to make the learned CRM model *discriminative*—so that it captures something specific to the given set of positive examples instead of something that is trivially or generally true of promoter sequences. The regulation of these negative examples may be related to the positive set in some interesting way (*e.g.* they are co-expressed under some other conditions), or they may simply be promoter sequences believed not to be regulated along with the positive examples.

The set of potential binding sites is specified by indicating the location of each occurrence in the positive and negative example sequences. These potential binding sites may come from a set of known or postulated transcription factor binding sites, *e.g.* from TRANSFAC or a BioCyc database [Wingender *et al.*, 2001; Krummenacker *et al.*, 2005], or they may come from a standard motif-finding algorithm, such as MEME [Bailey and Elkan, 1994].

3.3.1 Learning a Model

I refer to the approach presented here as SCRM1 (Structured CRM model learner version 1).¹ Pseudocode illustrating the learning algorithm is shown in Table 3.1.

Given sets of positive and negative DNA sequences, a set of potential binding site motifs, and an evaluation function, the TRAIN function searches through the space of possible models in an attempt to optimize the score given by the evaluation function. The ideal model would be satisfied by all the positive examples and none of the negative examples, so the evaluation function should be some measure of how well a given CRM model distinguishes between the positive and negative examples.

¹SCRM2 is presented in Chapter 4.

Table 3.1 The TRAIN function takes: *trainset*, a set of labeled DNA sequences; *aspects*, a list of CRM aspects which can be included (*i.e.* the maximum number of binding sites, whether or not distance constraints are allowed, *etc.*); *phases*, a list of phases, specifying the set of model changes allowed in each; *metric*, a CRM model scoring metric; and *K*, a maximum queue size (beam width). For each phase, TRAIN searches from the current list of solutions by making the changes allowed in that phase. It returns the best CRM model it finds.

```

TRAIN(trainset, aspects, phases, metric, K)
1  queue ← {NULL_SOLUTION }
2  CRM ← NULL_SOLUTION
3  for phase ∈ phases
4    while queue is not empty
5      current ← POP(queue)
6      for each applicable CRM change in aspects allowed in phase
7        alt ← APPLY(change, current)
8        if there is a sufficiently low  $\chi^2$  test probability that the trainset
9          predictions of current, alt are from the same distribution
10       then insert alt into queue
11         sort queue by metric
12         limit queue to K solutions.
13       if current has a better score than CRM given trainset, metric
14         then CRM ← current
15       repopulate queue with the best K solutions from phase
16 return CRM

```

Table 3.2 The SELECT-TRAIN algorithm takes: *trainset*, a set of labeled DNA sequences; *tuneset*, held-aside evaluation data; *aspects*, a list of CRM aspects to consider; *operators*, a set of model space reductions; as well as *phases*, *metric*, and *K*, which are arguments to the TRAIN algorithm. It removes aspects from the original list which are statistically shown (using the *tuning* set) not to contribute. Finally, it returns a CRM trained with all the data, using the CRM aspects chosen.

```

SELECT-TRAIN(trainset, tuneset, aspects, operators, phases, metric, K)
1  CRM ← TRAIN(trainset, aspects, phases, metric, K)
2  repeat
3      justified_aspect_sets ← { } # Aspect sets that are better justified than the current
4      for operator ∈ operators
5          alt_aspects ← APPLY(operator, aspects)
6          alt_CRM ← TRAIN(trainset, alt_aspects, phases, metric, K)
7          if there is not a sufficiently low  $\chi^2$  test probability that the tuneset
8              predictions of CRM, alt_CRM are from the same distribution
9              or alt_CRM scores better on tuneset than CRM
10             then justified_aspect_sets ← justified_aspect_sets ∪ alt_aspects
11             aspects ← highest scoring aspect set in justified_aspect_sets
12             CRM ← alt_CRM associated with these aspects
13 until justified_aspect_sets is empty
14 final_CRM ← TRAIN(trainset + tuneset, aspects, phases, metric, K)
15 return final_CRM

```


The search process is a best-first beam search [Mitchell, 1997] that starts with the null solution (an unconstrained model with zero binding sites) and searches in phases, modifying the best available solution and keeping only a queue of the best K models. Once the queue becomes empty, the best K solutions found are carried over to the initial queue for the next phase. In each phase, the algorithm applies a subset of the following operators:

- A new binding site is added (making use of the AND logic aspect).
- For a given binding site, a new motif is added (OR logic).
- A repressor motif is added between the CRM and the transcription start site (NOT logic).
- A repressor motif is added upstream of the CRM.
- A repressor motif is added between a pair of binding sites.
- The distance from the CRM to the transcription start site is constrained (to the best distance smaller than the current distance, according to the data and the scoring metric).
- For a given pair of binding sites, the distance between them is constrained.
- For a given pair of binding sites, their relative order is constrained.
- For a given binding site, a strand constraint is imposed.

There are user-defined limits on the maximum number of binding sites, motifs that can represent a binding site, and repressor motifs in a set.

Many of these operations will not affect the score of a solution (*e.g.* if a motif that does not appear in any sequence were added to the disjunction of motifs for a particular binding site, the model would match exactly the same sequences). For this reason, the algorithm will insist on some statistical difference between the set of sequences predicted by any of these changes. The TRAIN function uses a χ^2 test to decide whether it can reject the null hypothesis that two sets of sequence predictions by two different models come from the same distribution. It is not necessary to insist on near certainty when selecting the test's level of confidence; we mean only to avoid filling up the queue with multiple copies of essentially the same solution. If the test indicates that they come from different distributions, the new solution is added to the queue. Otherwise, it is discarded.

In my experiments, I use two phases for the TRAIN procedure, making most of the above changes during the first phase, but adding repressor motifs in the second phase. The argument for this choice is that these repressors can only be correctly added within the context of a CRM structure that has already been developed.

3.3.2 Controlling the Expressivity of a Model

Since the model space is expressive enough to represent many aspects of a CRM, the algorithm must address the potential for overfitting. This is done by first identifying the CRM *model space* appropriate for the data, and then searching through this space for the correct CRM. The model space is determined by the CRM aspects described above, so the goal is to find the appropriate set of aspects.

Spatial aspects (order, distance, and strand) can either be left in or out of a set, but the logical aspects can be limited without removing them altogether. This is done by reducing the maximum number of binding sites, the maximum number of motif disjuncts that can represent a binding site, and the maximum number of negated binding sites, until they are at a minimum (there must be at least one binding site, and at least one motif to represent it).

For the algorithm that does the model space selection procedure, we assume that set of aspects describes the limits placed on the logical aspects as well as whether or not each spatial aspect is included. Pseudocode illustrating the selection procedure is given in Table 3.2. Note that SELECT-TRAIN is the main procedure, and it calls the TRAIN procedure as a subroutine.

This search space is too large to consider all possible aspect sets, so SELECT-TRAIN uses greedy selection. Since the inclusion of one model aspect may depend on another (*e.g.* distance constraints are only effective once the affected binding sites are identified), it uses backward selection instead of forward. That is, we start with the full set of CRM aspects (with user-defined initial limits on the logical aspects), and prune away from this set as long as doing so is determined to be an improvement. This is opposed to starting with an empty set and adding to it.

The list of search space operators is:

- Reduce the maximum number of binding sites by one (*i.e.* reduce the AND logic maximum conjunction size).
- Reduce the maximum number of motifs per binding site by one (*i.e.* reduce the OR logic maximum disjunction size).
- Reduce the maximum number of motifs in a set of repressor motifs by one (*i.e.* reduce the NOT logic set size).
- Disallow distance constraints.
- Disallow order constraints.
- Disallow strand constraints.

To compare two aspect sets, SELECT-TRAIN compares the scoring metric's evaluation of a model learned using each of the aspect sets on a *tuning set* of sequences, held-aside from the training set. During the selection process, SELECT-TRAIN keeps the larger model space (*i.e.* with an additional spatial aspect or a higher limit on a logical aspect) over a smaller model space if and only if the evaluation score in the former case show both an improvement and a *statistically significant difference* in the scoring metric's evaluation. In other words, the search process selects an aspect of CRM expressivity if and only if doing so is statistically justified by the data. This way, the model uses only the expressiveness required by a specific CRM, and can then retrain the model by searching through the appropriate model space.

Unless leaving an aspect in the model space (or a higher size limit) produces a statistically significant improvement as determined by a χ^2 test,² it is removed. SELECT-TRAIN makes only

²The χ^2 test threshold does not need to be the same in both the TRAIN and SELECT-TRAIN algorithms.

one model space restriction per iteration. If more than one restriction in the list above is being considered, SELECT-TRAIN makes the restriction that leads to the best tuning set score. This process is repeated on the more restricted model space until no more restrictions should be considered (*i.e.* all structural aspects are statistically justified). This approach is similar to backward feature selection [Devijver and Kittler, 1982; Miller, 1990]. However, it does not decide on whether or not to include specific *features* (*e.g.* what is the distance between motif *A* and motif *B* in each DNA sequence), but rather, it decides whether or not to include (or limit) entire *aspects*. In this way, it is similar to *predicate selection*, wherein the learner decides whether or not to use logic predicates (which have a multitude of instantiations using different objects) when searching for a model in relational learning [Cohen, 1995].

3.4 Empirical Evaluation

I test my approach on several data sets, summarized in Table 3.3. Three of these data sets have been used in previous studies of computational CRM finding. The Gasch *et al.* data set, however, is novel. In each case, I obtain upstream/promoter sequences from the University of California Santa Cruz Genome Browser [Karolchik *et al.*, 2004] and perform cross-validation to evaluate my algorithm. I obtain a set of candidate motifs from running MEME [Bailey and Elkan, 1994] on the positive examples (not including any test sequences held-aside for evaluation) and from running MEME on upstream/promoter regions randomly sampled from the appropriate organism. For the fly data set, I also evaluate my approach when it is provided with a set of known motifs [Rajewsky *et al.*, 2002; Sinha *et al.*, 2003].

These motifs are represented by position weight matrices (PWMs as described in Section 2.1). Note that this is a probabilistic representation, so finding the locations of motifs is more complicated than simply matching a search string. To find the locations in a sequence S of length L of a motif M of width W , I consider all the $L - W + 1$ possible starting locations of M in S . For each of these locations i , I compare the likelihood of the subsequence $S_{i...i+W}$ being generated by the PWM representation of M to the likelihood of the subsequence being generated by a 5th-order Markov chain model [Thijs *et al.*, 2001; Marchal *et al.*, 2003] that is trained on the promoter regions of an entire genome. I consider a motif to be present if the ratio exceeds a threshold.

I use the $F1$ metric, discussed in section 2.3, as a measure of how well the model predicts all, and only, the sequences that contain the target CRM.

I set the maximum number of binding sites to three,³ the maximum number of motifs per binding site to three and the maximum number of repressor motifs in a set to one. I evaluate the learned model by using cross-fold-validation, as described in Section 2.3. If the held-aside sequence contains the hypothesized CRM, the model predicts that it is a positive example. The results from each fold are summed together.

³The Lee *et al.* data sets are believed to have binding sites for two specific proteins. However, limiting the maximum number of sites to two does not improve the results, as the aspect selection procedure restricts the number of binding sites to less than three in all but one case.

Table 3.3 Summary of the data sets on which I test my algorithms.

Data Set	Organism	Description
Lee <i>et al.</i>	<i>S.cerevisiae</i>	Twenty-five sets of genes with strong evidence (p -value ≤ 0.01) from the genome-wide location analysis of Lee <i>et al.</i> [2002] that a specific pair of regulators binds to their upstream regions. This is a recreation of the data sets used by Segal <i>et al.</i> [Segal and Sharan, 2004]. For each data set, I use 100 yeast promoters chosen at random as negative examples.
Gasch <i>et al.</i>	<i>S. cerevisiae</i>	Three sets of genes associated with environmental stress response (ESR) in Yeast, described in [Gasch <i>et al.</i> , 2000]. I use promoter sequences from non-ESR yeast genes as negative examples.
Sinha <i>et al.</i> –Yeast	<i>S. cerevisiae</i>	A set of six yeast sequences where MCM1 and MAT α 2 are known to bind, described in Sinha <i>et al.</i> [2003]. For negative examples, I use nine promoter sequences which contain binding sites for <i>either</i> MCM1 or MAT α 2, but not both.
Sinha <i>et al.</i> –Fly	<i>D. melanogaster</i>	A set of eight fly genes associated with the gap gene system, described in Sinha <i>et al.</i> [2003]. I use 10kb promoter sequences, and 100 promoter sequences selected randomly from the fly genome to use as negative examples.

Table 3.4 Results of running SCRM1 on the Lee *et al.* data sets described in Table 3.3. Significant CRMs (p -value < 0.01) are indicated with bold text.

Data set	F1	p -value	Data set	F1	p -value
FHL1, RAP1	0.745	3.33-e10	CIN5, NRG1	0.435	1.53e-03
GAT3, YAP5	0.667	8.75-e10	NRG1, YAP6	0.448	3.72-e03
MBP1, SWI6	0.645	6.34-e09	MBP1, NDD1	0.419	7.52-e03
RAP1, YAP5	0.588	3.68-e07	CIN5, YAP6	0.486	8.47-e03
MCM1, NDD1	0.545	2.37-e06	FKH2, SWI4	0.4	0.0113
GAT3, PDR1	0.556	1.138e-05	RGM1, YAP5	0.368	0.0137
FKH2, MCM1	0.513	2.45-e05	GAL4, YAP5	0.359	0.0169
NDD1, SWI4	0.5	8.51e-05	PDR1, YAP5	0.361	0.0585
FHL1, YAP5	0.508	1.07-e04	SKN7, SWI4	0.333	0.118
PHD1, YAP6	0.5	1.52-e04	GAT3, RGM1	0.238	0.253
MBP1, SWI4	0.54	2.00-e04	SWI4, SWI6	0.37	0.506
FKH2, MBP1	0.494	2.60-e04	FKH2, NDD1	0.417	0.978
ACE2, SWI5	0.472	7.97-e04			

For each data set, I calculate an F1 score, \mathcal{F} (the same statistic as the algorithm’s scoring metric), and use Fisher’s exact test [Agresti, 1992] to calculate a p -value. If the positive predictions (true positives plus false positives) were made simply by randomly sampling without replacement from the data set, this p -value represents the probability of an F1 score of \mathcal{F} or higher. If this p -value is sufficiently low (less than 0.01, following Segal and Sharan [2004]), I consider the learned CRM for this data set to be significant.

Results are shown in Tables 3.4, 3.5, and 3.6.⁴ I find a significant CRM in 17 of the 25 Lee *et al.* data sets. In their similar experiments, Segal *et al.* found significant CRMs in only 12 of the 25 data sets. The p -value calculations of Segal *et al.* are not identical to mine; as their CRM model makes probabilistic predictions, they are able to calculate a p -value using a *classification margin* [Segal and Sharan, 2004]. We are comparing against the same null hypothesis in each case, however, and calculating the probability of test set accuracy greater than or equal to what is observed, given chance alone.

A few of the binding sites associated with the proteins in the Lee *et al.* data sets are known. In these cases, my algorithm often recovers these motifs. Some examples of this are shown in Figure 3.8. However, I do not wish to focus too much attention on recovered motifs here, because my approach does not define these (they are found by MEME), it only selects them from a set of candidates.

I find significant CRMs in the three Gasch *et al.* data sets, which suggests that this method can be used to find novel CRMs corresponding to genes clustered by expression analysis.

⁴More complete tables of results on these datasets are shown in Appendix A.

Table 3.5 Results of running SCRM1 on the Gasch *et al.* data sets described in Table 3.3. Significant CRMs (p -value < 0.01) are indicated with bold text.

Data set	F1	p -value
iESR	0.219	~ 0
rESR_PACcluster	0.402	~ 0
rESR_RPcluster	0.392	~ 0

Table 3.6 Results of running SCRM1 on the Sinha *et al.* data sets described in Table 3.3. Significant CRMs (p -value < 0.01) are indicated with bold text.

Data set	F1	p -value
Yeast	0.857	5.59-e03
Fly	0.429	4.92-e03
Fly, known PWMs	0.560	5.10e-06

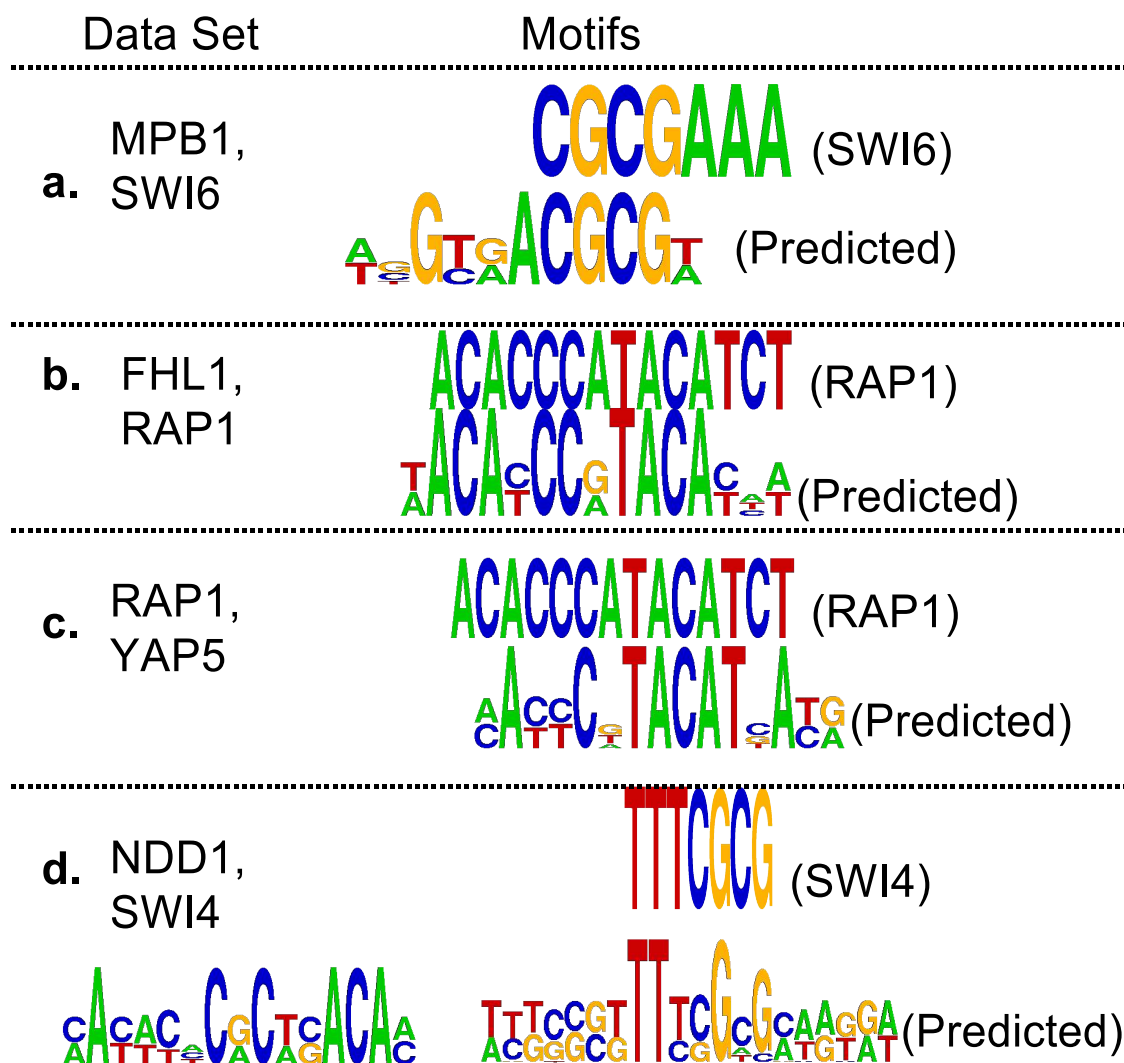


Figure 3.8 Some of the recovered motifs from the Lee *et al.* data sets. In each case (a. through d.), the data set is believed to contain the binding sites for two proteins, one of which has a known consensus sequence in TRANSFAC [Wingender *et al.*, 2001]. That motif is written on the top along with the protein name in parentheses. The motif recovered by my approach is written beneath it. In the last case (d, proteins NDD1 and SWI4), my approach predicts that there are two relevant motifs, one of which is a novel binding site motif prediction for NDD1.

I find a significant CRM in the Sinha *et al.* fly data set as well. For this data set, using motifs suggested by MEME, SCRM1 finds three true positives and three false positives. Although this result is statistically significant, I hypothesize that the reason it is unable to recover more of the positive examples is because the training set size is too small for MEME to find good candidate motifs. To test this, I use the PWMs from Rajewsky *et al.* [2002] and Sinha *et al.* [2003] and locate positions where these motifs are most likely to occur. In this case, SCRM1 recovers seven of eight positive examples. Note that I do not compare my results to those of Sinha *et al.* because I use this data set to evaluate predictive accuracy on held-aside data, whereas they do not.

I wish to determine whether or not the inclusion of structural aspects increases the accuracy of the learned models. I do this by comparing the results of my approach to those obtained when the set of *aspects* given to the TRAIN function is limited. I do this in two ways: First, I measure this accuracy by the F1 score of the predictions on held-aside data, and compare these scores to those obtained by a restricted version of the algorithm, for which the only *aspect* given to the TRAIN function is multiple binding sites. This experiment is designed to compare against the model space of several previous methods in which a CRM model is characterized simply by the presence of a set of motifs anywhere in an input sequence [Aerts *et al.*, 2003; Sharan *et al.*, 2003; Sinha *et al.*, 2003; Gupta and Liu, 2005; Mahony *et al.*, 2005; Philippakis *et al.*, 2005; Macisaac *et al.*, 2006]. I refer to this as the “bag-of-motifs” approach. Second, I compare the F1 scores of my approach to those of running the algorithm with a single structural aspect left out of the set given to the TRAIN function, for each aspect/dataset pair. This is designed to determine whether each structural aspect by itself makes a positive contribution to the learned models. I refer to these experiments as “lesion tests.”

These comparisons are illustrated in Figure 3.9. Note that sometimes the inclusion of a structural aspect can lead to overfitting (a point slightly above the diagonal line), but often it is essential (a point well below the line). Indeed, considering all data sets, the F1 score is more often higher with all aspects included than it is when any single structural aspect is removed.

On the 25 yeast data sets from Lee *et al.* (see Table 3.3), the bag-of-motifs approach is often about as accurate as SCRM1. One exception is shown in Figure 3.10. Here, my algorithm discovers that the order of binding sites is important. Compare the test set F1 score of SCRM1 (0.500) to that of the bag-of-motifs approach (0.205). On the other data sets, SCRM1 scores much higher than the bag-of-motifs approach. For instance, Figure 3.11 shows the hypothesis CRM model for the data set, rESR_RPcluster, which involves distance and strand constraints. The bag-of-motifs hypothesis (not shown) also includes two copies of the same motif, but without structural constraints. This model accepts eight additional true positives, but 265 additional false positives. Applying SCRM1 to the Sinha *et al.* fly data set, we find three true positives and three false positives (compared to two true positives and 34 false positives using the bag-of-motifs approach). Using PWMs from the literature, SCRM1 recovers seven of eight positive examples, with 10 false negatives (compared to six true positives and 16 false negatives using the bag-of-motifs approach).

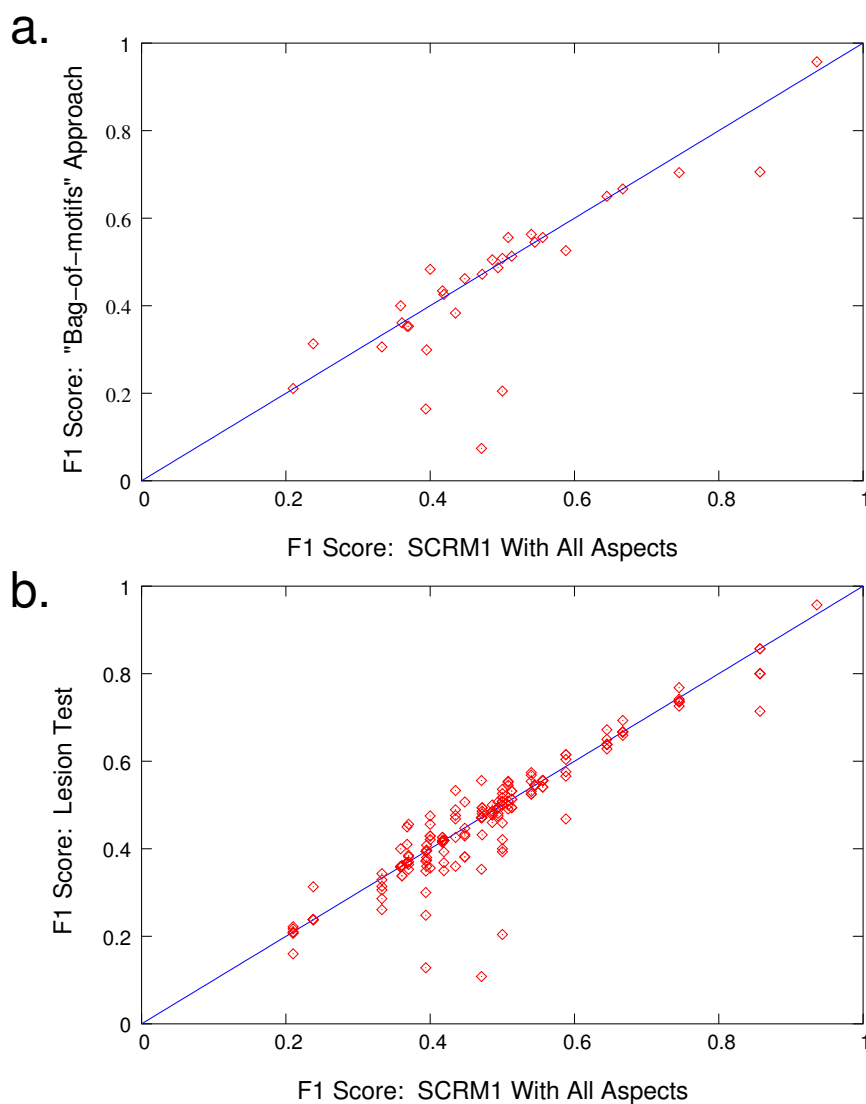


Figure 3.9 Each point in a scatter plot compares the F1 score of a data set using SCRM1 (x-axis) to the F1 score of a data set using an aspect-restricted version (y-axis). Higher F1 scores are better, so a point below the diagonal represents a data set for which using all aspects improves the accuracy of the learned models. **a.** The F1 score of SCRM1 (x-axis) compared to the F1 score of the bag-of-motifs approach (y-axis). **b.** The F1 score of SCRM1 (x-axis) compared to the F1 score of a lesion-test (y-axis) wherein a model was trained with one structural aspects left out of the set given to the TRAIN function (this experiment is run for each aspect, for each data set).

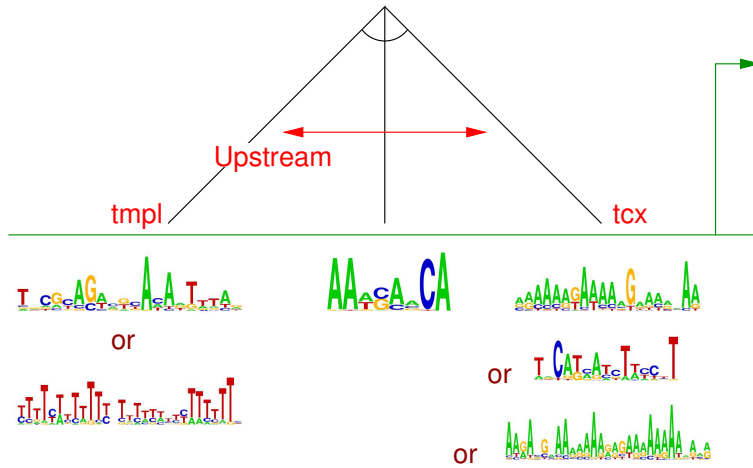


Figure 3.10 Hypothesized CRM for the PHD1, YAP6 (Lee *et al.* data set. The relative order of two binding sites (characterized by a set of possible motifs) is constrained. *tmpl* indicates the motif appears on the template DNA strand, and *tcx* indicates the transcribed strand.

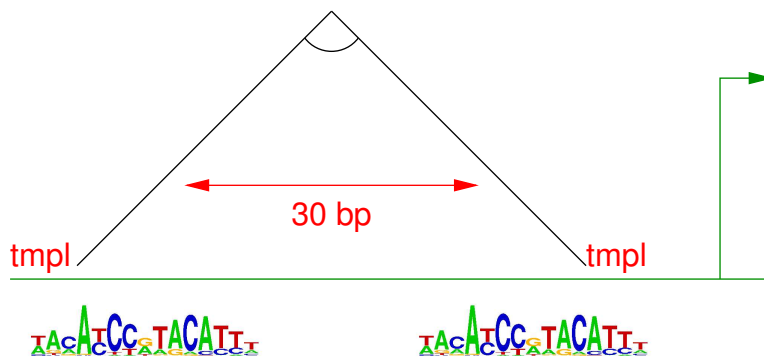


Figure 3.11 Hypothesized CRM for the rESR_RPcluster data set. The model consists of two copies of the same motif on the same strand, nearby one another.

3.5 Conclusion

One of the primary steps in gene regulation is transcription, and the ability to learn CRMs directly from data will be a crucial part of understanding how transcription is controlled. My experiments, as well as those of Beer and Tavazoie [2004] suggest that transcription is controlled not only by the presence of binding sites, but also by relationships between their locations. My models represent a step forward in this area because these aspects are represented in a model that is easy to inspect and understand, and the results show that each of them contributes to the identification of significant CRMs in real biological data.

With this increase in expressiveness, there is inevitably a risk of overfitting. I use data to identify the appropriate CRM aspects during the process of training the models. I believe that my novel approach of model space selection is an important and necessary step to facilitate the move toward more expressive models.

This work was originally published in Noto and Craven [2006]. A summary of the contributions is as follows:

- I have designed a comprehensible CRM model that includes biologically-relevant logical and spatial aspects that describe a *cis*-regulatory module in detail beyond the relevant binding site motifs.
- I have presented a specialized algorithm that selects these motifs and learns their relational aspects in a way that is data-driven.
- I have shown that each of these aspects improves the accuracy of learned models on real biological tasks.
- I have presented a method for selecting the appropriate, statistically justified *model space* through which to search.
- I have made the source code for SCRM1 publicly available at www.cs.wisc.edu/~noto/crm.

Chapter 4

Learning Probabilistic Models of *cis*-Regulatory Modules that Represent Logical and Spatial Aspects.

4.1 Introduction

In Chapter 3, I described a representation that was able to capture several logical and spatial aspects of a *cis*-regulatory module. These are:

1. AND logic; multiple required binding sites, *e.g.* A and B ,
2. OR logic; a set of motifs, any of which satisfies a binding site, *e.g.* B_1 or B_2 ,
3. NOT logic; a binding site that must not appear in a promoter sequence, *e.g.* A , but not B ,
4. Order; *e.g.* binding site A appears upstream of B ,
5. Distance; *e.g.* binding site A appears within 125 bp from B , or binding site A appears somewhere within 50 bp from the estimated start of transcription,
6. Strand; *e.g.* binding site A appears on the template DNA strand (as opposed to the transcribed strand).

In this chapter, I present and evaluate a second approach to the same task, but the model and learning algorithm are enhanced in two ways: First, the algorithm learns binding site motifs *de novo*, directly from sequence data, as opposed to selecting from a list of candidate motifs. Second, the spatial aspect constraints are replaced by probabilistic preferences.

This second enhancement requires some explanation. The models presented in Chapter 3, SCRM1, chose hard constraints, such as a specific DNA strand or upper-bound on distance, whenever doing so improved the training set score. This has the drawback of missing positive instances that are close to the CRM model. For instance, if a CRM consists of two binding sites A and B , and shows a distinct preference for A appearing upstream of B , SCRM1 will include that preference. It may be the case that some positive instances of the CRM, though rare, have the binding sites in the opposite order. The models presented in this chapter represent the fact that the order $A B$ has a high *probability*, but can still recognize the fact that some sequences are likely to be positive instances of a CRM without having the most likely value for every aspect of the model. Furthermore, the

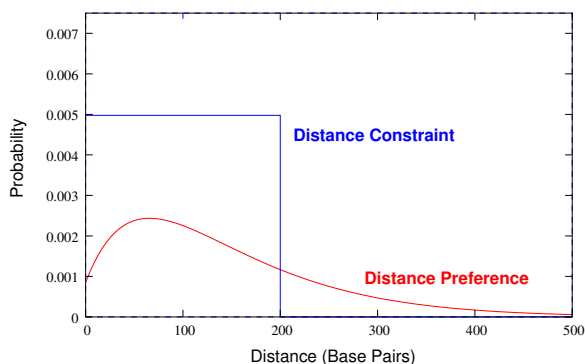


Figure 4.1 Two examples of a probability distribution over distance. One is representative of a distance upper-bound *constraint*, and one is representative of the smooth distance *preference* used by the models presented in this chapter.

SCRM1 models decide where motifs are using a threshold as part of a pre-processing step. The motifs are represented probabilistically, as position weight matrices (PWMs), but once the positions of candidate motifs are selected, the learning algorithm discards the probabilistic representation. Specifically, the shift to probabilistic preferences affects the following aspects:

- **Motif matching:** Each motif is represented as a PWM throughout the learning process, and the weights are adjusted as part of that process. The question of whether a motif matches a given position and strand is now a probabilistic one. This is opposed to using a PWM and a threshold to estimate motif occurrences as a pre-processing step.
- **OR logic:** Each binding site now consists of a probability distribution over independent position weight matrices that represent motifs. For instance, binding site *A* usually matches a PWM with consensus GCGATGAG closely, but 10% of the time, it matches the wider PWM with consensus GCGATTGAG.
- **Order:** Both possible orderings between pairs of binding sites (*i.e.* *A B* vs. *B A*) have probabilities represented by the model. These parameters are used to estimate the likelihood of a total ordering when there are more than two binding sites.
- **Distance:** The length of DNA between adjacent binding sites is represented by a probability distribution over the nonnegative integers. A comparison between distance constraints and distance preferences is shown in Figure 4.1.
- **Strand:** Each binding site has a probability representing whether or not it tends to appear on the transcribed (sense) DNA strand.

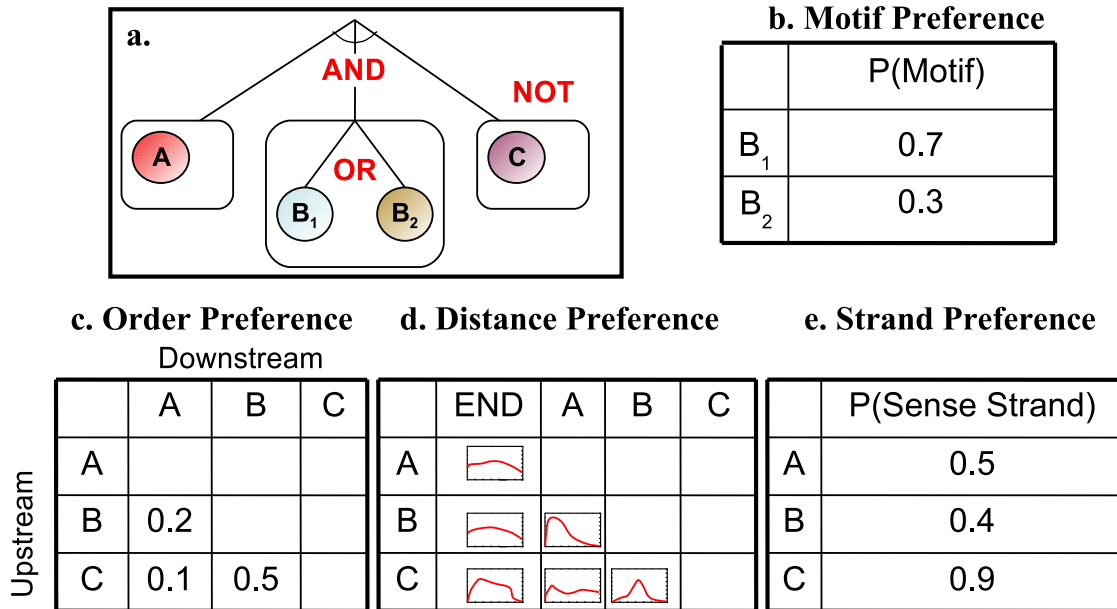


Figure 4.2 An instance of the CRM model presented in this chapter, which includes logical and spatial aspects of a set of motifs. **a.** A logical structure of motifs consisting of a conjunction of binding sites, each of which is represented by a disjunction of motifs (or a single motif), and may be *negated*, meaning that the CRM model explicitly represents the fact that one or more binding sites do not appear in positive instances of the CRM. **b.** Each disjunction of motifs is associated with a motif preference: A probability distribution over the motifs in the disjunction. **c.** Each pair of binding sites is associated with an order preference: One number indicating the probability of one site being upstream of the other. **d.** Each pair of binding sites is associated with a distance preference: A probability distribution over the distance between the motifs when they appear next to each other in a sequence. There is also a distribution over the distance from each motif, when it appears last in order, to the downstream end of the sequence. **e.** Each binding site is associated with a strand preference, indicating the probability that the binding site will appear on the template strand, as opposed to the the transcribed strand.

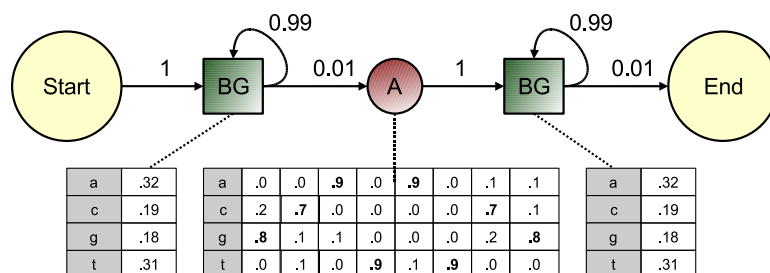


Figure 4.3 A hidden Markov model of a DNA sequence with a single motif. The motif state **A** emits characters from a position weight matrix, and is flanked on either side by state **BG**, which emits the rest of the sequence characters. Transitions between states are labeled with their probabilities.

4.2 Model Representation

An example model in my representation is shown in Figure 4.2. The model is composed of two parts. The first part is the logical structure, shown in Figure 4.2a,b. A sequence that contains the CRM represented by this example has an instance of motif *A*, one of either motif *B*₁ or *B*₂, and does *not* have an instance of motif *C*.¹ Since binding site *B* is represented as a disjunction, the model contains a probability distribution over its disjunct motifs, shown in Figure 4.2b. The second part is the spatial preference parameters, shown in Figure 4.2c-e. The order preferences (Figure 4.2c.) specify how likely it is to find *A* upstream of *B* (regardless of whether *B* is represented by motif *B*₁ or *B*₂). The distance preferences (Figure 4.2d.) specify how far apart each pair of adjacent binding sites are likely to be and how far from the downstream end of the sequence they are likely to be. The distance from the downstream end of the sequence determines the position of the CRM within the sequence. If the (known or estimated) position of the start of transcription is fixed relative to the end of each sequence, then these parameters represent the distance between the CRM and the transcription start site. The strand preferences (Figure 4.2e.) specify what orientation each binding site is likely to be. Note that the spatial preferences include negated binding site *C*, which apply when it is present.

The models presented in this chapter can be represented as hidden Markov models (HMMs, see Section 2.2.2), and the algorithm that learns the models uses an HMM parameter learning method. To explain the connection between a CRM model and a hidden Markov model, we will start with a simple example. Consider the case of a single motif of length *W* that appears on the template strand of a DNA sequence of length *L*. Such a sequence is explained by the HMM shown in Figure 4.3. More specifically, the sequence is explained by a particular *path* (sequence of states) through the model, starting with the **Start** state and ending with the **End** state. Such a path would enter the **BG** (“background”) state, explain one or more characters, transition to the motif state **A**,

¹Note the use of subscripts when there are multiple motifs that represent a single binding site. All motifs in the disjunction share spatial preferences because they have the same role in the CRM.

explain exactly W characters,² transition to the second BG state, explain the rest of the characters, and finally transition to the End state. The likelihood of such a path depends on the transition and emission parameters (*i.e.* how well do the characters explained by motif state A match the PWM?). Note that the *same background distribution* appears twice in the HMM. This is represented by the fact both the emission distribution and the transitions out of the background states have the same probabilities. This type of parameter sharing is important because it is a way of avoiding overfitting the training data. One can use standard HMM parameter learning algorithms to learn the PWM emission distribution probabilities of motif A . This is what some single-motif learners such as MEME do [Bailey and Elkan, 1994]. This model can also be used to estimate the likelihood that a given sequence contains a motif, if it is compared to an alternative model (*e.g.* the same HMM without motif state A).

There is also a hidden Markov model representation for the CRM models described above. Consider the CRM model shown in Figure 4.4. This model, with all its logical and spatial aspects, represented as an HMM, is shown in Figure 4.5.³

There are a few things to point out about this HMM model and how it corresponds to the CRM model in Figure 4.4. First, the paths through the HMM are divided into positive and negative groups. “Positive” paths explain instances of the CRM—sequences that contain the motifs as described by the CRM model. “Negative” paths explain sequences that do not contain the CRM. One negative path explains the entire sequence with the background state. Other negative paths contain some instances of motifs in the CRM model, but not the right combinations of them to count as an instance of the CRM.

Second, the BG background states have their own *duration distribution*. That is, they have an arbitrary probability distribution over the number of characters that are to be explained in that state. This is opposed to having a self-transition (which implies a geometric duration distribution). These duration distributions mean that these hidden Markov models are really *generalized* hidden Markov models [Burge and Karlin, 1997], also called hidden *semi*-Markov models [Rabiner, 1989] or segment models.

Third, there is extensive parameter sharing in this HMM. All the states that share the same name (A , BG , *etc.*) have identical emission distributions and are identical. That is, the PWM for state A is the same in the positive paths as in the negative paths. The PWMs for state A and for the reverse-complement of A are really the same distribution. All the BG states share character emission parameters. Several of the transition parameters are shared. For instance, according to the CRM model in Figure 4.4, the probability of binding site B being on the template strand is 0.9. This probability is repeated for every instance of motif state B_1 and also for every instance of B_2 , since the strand preference applies to binding sites, not each motif disjunct. The distance distribution between states A and B is the same regardless of order.

²I represent a motif with a single HMM state, but to be consistent with many typical formulations of HMMs where each state emits exactly one character, motif state A can be expanded to a chain of eight consecutive states, each generating a character from the corresponding column of the PWM.

³The HMM representation is too large to show in a figure when there are more than two binding sites (such as the model in Figure 4.2), but the topology of the HMM extends naturally with additional binding sites and other changes to the CRM model.

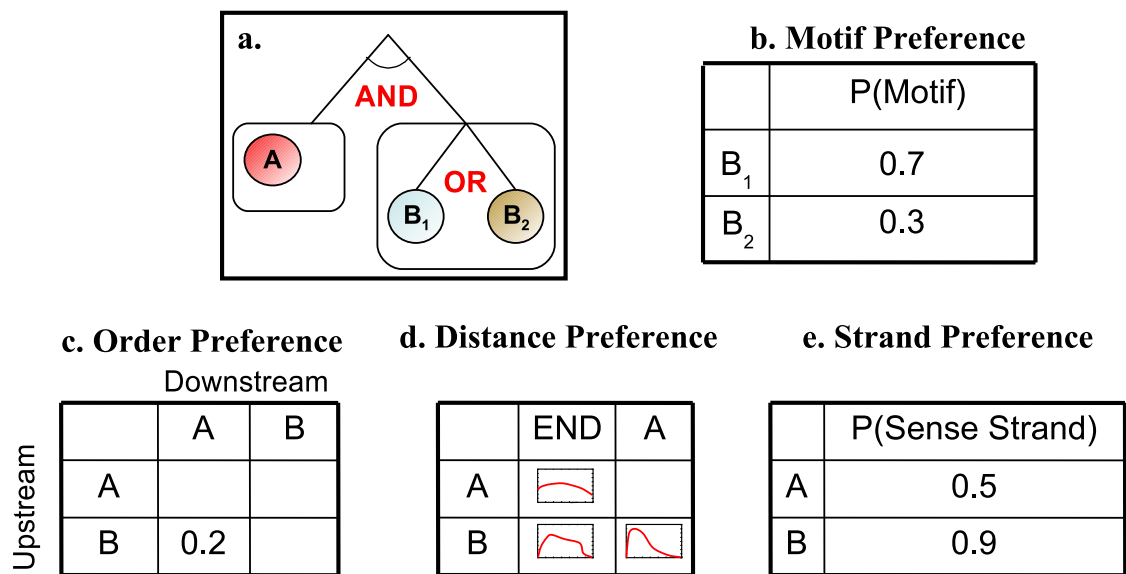


Figure 4.4 A CRM model with two binding sites, the second of which is a disjunction. **a.** The logical structure of the model, $A \wedge (B_1 \vee B_2)$. **b.** The motif preference for binding site B . **c.** Order preference: There is a 0.2 probability that B appears upstream of A . **d.** Distance preference: The distance between A and B , and between each motif and the end of the sequence is chosen from a probability distribution. **e.** Strand preference: A appears on either strand with equal probability, but B_1 or B_2 appears on the sense strand most of the time.

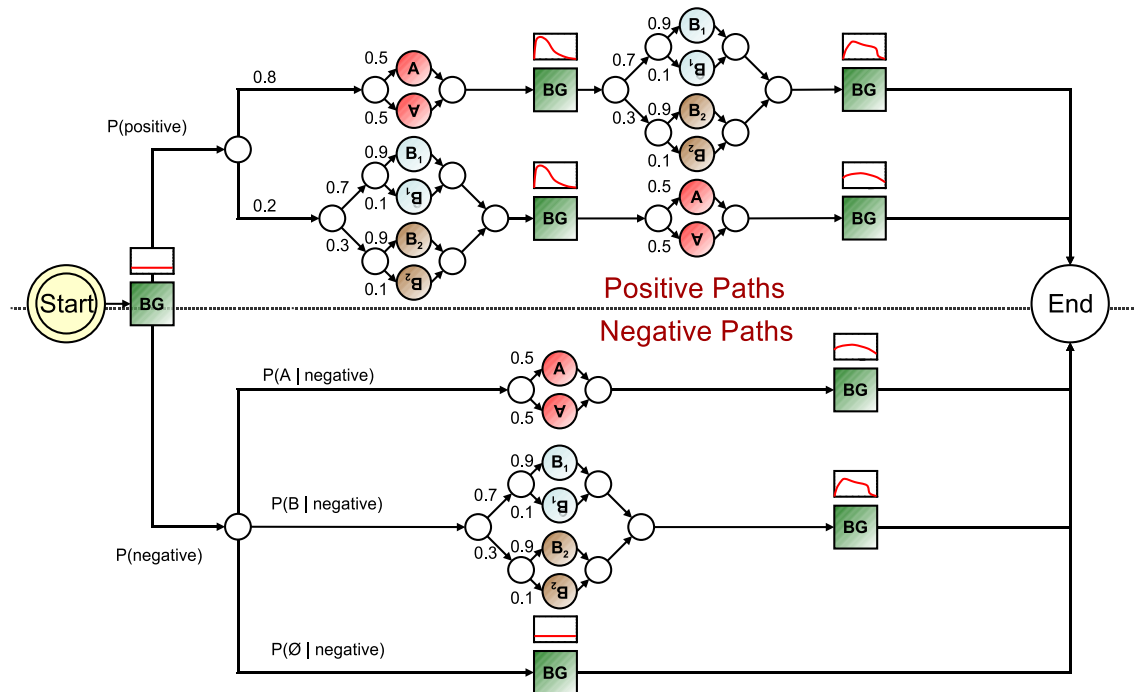


Figure 4.5 The HMM representation of the CRM model shown in Figure 4.4. Motif states (A , B_1 , B_2) are represented by labeled round states. The DNA reverse-complement is represented by upside-down states, which indicate that a motif may appear on either strand. The background emission distribution is represented by the square BG states. Silent (placeholder) states are represented by small white round states. All positive paths encounter motif A and either B_1 or B_2 , but either order is possible. Negative paths may encounter one of the two binding sites, but not both. Note that the state-transition choices made along a path from the Start state to the End state correspond to the order and strand probabilities in the model (Figure 4.4), and the duration distributions shown above the background states correspond to the distance preferences between motifs. Note that some transition probabilities (labeled “ $P(\dots)$ ”) are representative of the distribution in a data set and are not part of the CRM model.

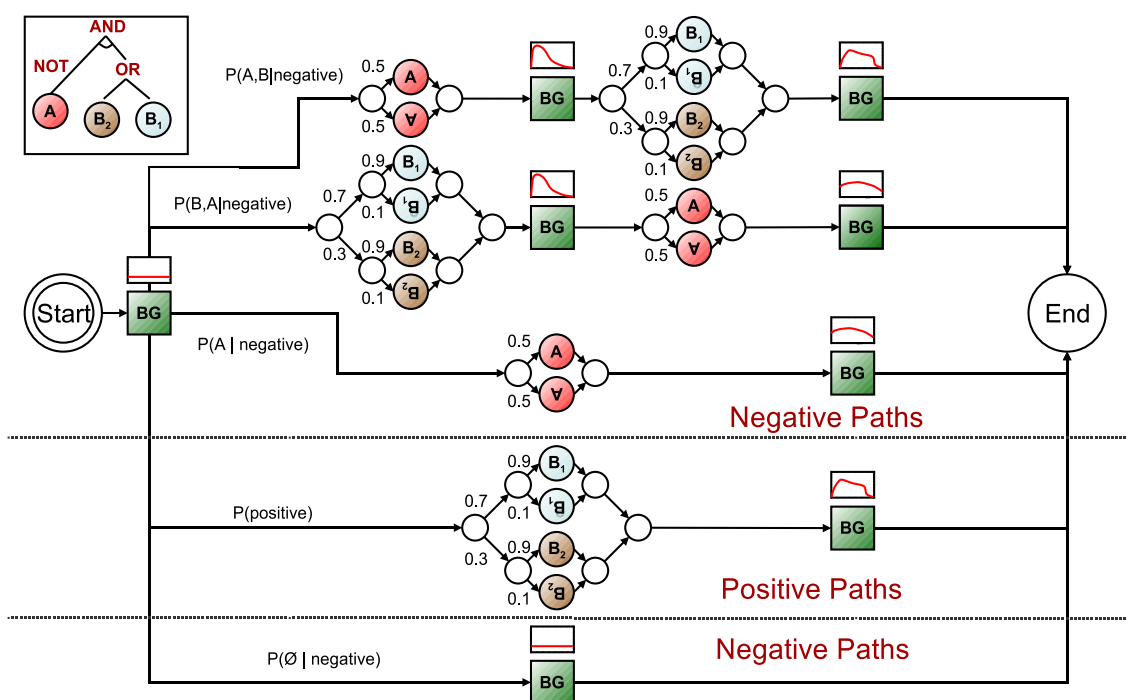


Figure 4.6 The HMM representation of the model shown in Figure 4.4, but with binding site A negated (the logical structure is shown in the inset). Note that the paths that encounter all and only the *unnegated* binding sites are marked as positive. (States are the same as described in Figure 4.5.)

Fourth, the HMM model includes parameters that are not shown as part of the CRM model in the figures. These are transitions labeled “P(...)” in Figures 4.5 and 4.6. These parameters depend on the training set distribution and not on the CRM itself.

Fifth, The purpose of a *negated* binding site is to explicitly represent that, if a sequence contains such a site, an otherwise positive example should be labeled negative. Therefore, the positive paths are the ones that contain *all* and *only* unnegated binding sites. If a path contains a negated binding site motif, then it is a negative example. This is illustrated in Figure 4.6, which shows the HMM corresponding to the CRM model in Figure 4.4 if binding site *A* were negated.

A sequence is explained by following a path through the model from the the start state to the end state. We do not know *which* path is correct, so we consider the likelihood of every path, given the sequence data. To make it clear how a path explains a sequence, consider a sequence with motif *A* appearing at position p_A (let W_A be the width of motif *A*, so state **A** explains characters p_A through $p_A + W_A$) and motif B_1 appearing at position p_B ($p_A < p_B$), both on the template strand. Exactly one path explains this case. This path involves: (i) Explaining all the sequence characters up to p_A with the background state, (ii) choosing a positive path, (iii) choosing the order *A B*, (iv) choosing the template strand of *A*, (v) explaining the characters from position p_A to $p_A + W_A$ with motif state **A**, (vi) explaining the characters from $p_A + W_A$ to p_B with the background state, (vii) choosing motif B_1 over B_2 , (viii) choosing the template strand for *B*, (ix) explaining the characters from position p_B to $p_B + W_B$ with motif state **B**, and finally (x) explaining the characters from $p_B + W_B$ to the end with the background state.

Each of these choices has a probability, and each character has a probability, given the state that explains it. The product of all these probabilities gives us the likelihood of the path.

And, just like the single-motif HMM in Figure 4.3, we can learn the parameters of our CRM models by training the HMM on our input data. However, we do not know *a priori* how many binding sites are in the CRM. That is, we must learn the logical structure of the CRM model, as well as the spatial parameters.

4.3 Learning a Model

Our learning task is the same as in Chapter 3, except we are not given a set of candidate motifs; instead we must learn them.

Given:

- (i) Positive sequences believed to share a CRM,
- (ii) Negative sequences believed not to contain the CRM.

Learn: A model of the CRM, including its binding site motifs, and the logical and spatial relationships that characterize it.

I refer to the learning algorithm presented in this chapter to solve this task as SCRM2. The task involves learning both the logical and spatial aspects of our CRM model. To do this, SCRM2 searches the space of possible CRM model logical structures, and learns the parameters by training

the corresponding hidden Markov model. The resulting model can then be evaluated, again using the corresponding HMM.

4.3.1 Structure Learning

Pseudocode illustrating the algorithm is shown in Table 4.1. It searches for the CRM logical structure using a best-first beam search [Mitchell, 1997]. That is, the algorithm keeps a finite-sized list of candidate solutions, sorted by their estimated accuracy on a held-aside tuning set. The CRM model search-space operators are shown in Figure 4.7. At each step in the search process, the algorithm removes the highest-scoring model from the list, applies each of the search space operators (*e.g.* adds a new binding site, adds a new motif to an existing binding site, *etc.*), then trains and scores the resulting models. The best model overall is returned as the solution. Note that each application of a search operator adds a new untrained motif that is represented by a position weight matrix (PWM). The learned model parameters will depend on the initial values in this PWM, so SCRM2 initializes it by sampling from the training data. The algorithm starts by generating a set of single-motif models and putting them in the list. This provides the simplest possible start for the the search, but it uses multiple “restarts,” since the training results depend on the initial values of the PWMs representing the motifs. There are user-defined limits on the size of the CRM model structure. These are a maximum number of binding sites, a maximum number of motif disjuncts per binding site, and a maximum number of negated binding sites. Note that some of the search operators do not apply once these limits have been reached, and in these cases, the operations are ignored.

4.3.2 Parameter Learning

When a given logical structure for a CRM model is considered, we must learn the rest of the model parameters and evaluate the model. The model parameters in question are the spatial parameters (order, distance and strand preferences), as well as the motif preferences, *i.e.* which motifs are most likely to account for each binding site. This is done by creating the hidden Markov model that corresponds to the CRM model (*e.g.* the model in Figure 4.4 corresponds to the HMM in Figure 4.5). The HMM parameters are then learned by training with the discriminative algorithm presented by Krogh [1994]. Krogh’s training algorithm sets the parameters $\hat{\Theta}$ in an attempt to optimize the likelihood of the *labels* given the sequences:

$$\hat{\Theta} = \arg \max_{\Theta} \prod_{(\mathbf{x}, y) \in \mathbf{D}} P(y | \mathbf{x} : \Theta) \quad [4.1]$$

where (\mathbf{x}, y) are the example sequences and labels (*i.e.* positive or negative), in a training set \mathbf{D} .

After learning the model parameters, SCRM2 does the appropriate normalization and smoothing. For most of the parameters, smoothing is done with pseudocounts. SCRM2 smooths distance distributions with a Gaussian-shaped kernel with standard deviation $\frac{1}{\sqrt{n}}$, where n is the training sample size [John and Langley, 1995].

Table 4.1 The SCRM2 function takes: *trainset*, a set of labeled DNA sequences; *tuneset*, a held-aside evaluation set; *operators*, a set of CRM model search operators; *metric*, a function for scoring model accuracy; *S*, a constant indicating the number of single-motif “seed” models; *K*, a constant indicating the beam width. The KROGH_TRAIN function takes a CRM model and training set, and selects the CRM model parameters by training the corresponding HMM using Krogh’s HMM training algorithm [Krogh, 1994].

```

SCRM2(trainset, tuneset, operators, metric, S, K)
1  queue ← A set of S single-motif models
2  best_model ← ∅
3  best_score ← 0
4  repeat
5      model ← POP_BEST(queue)
6      for operator ∈ operators
7          alt ← APPLY(operator, model)
8          KROGH_TRAIN(alt, trainset)
9          score ← EVALUATE(alt, tuneset, metric)
10         if score > best_score
11             then best_model ← alt
12                 best_score ← score
13         PUSH(queue, alt, score)
14         LIMIT_SIZE(queue, K) # Keep the best K models
15 until stopping criteria are met or queue is empty
16 return best_model

```

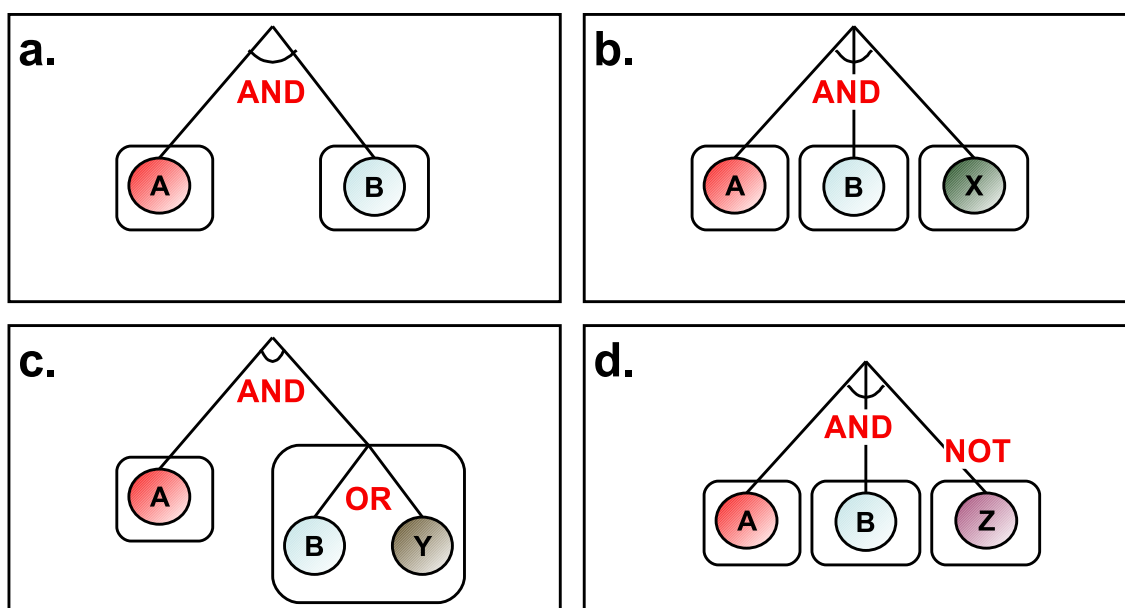


Figure 4.7 Illustration of our model search-space operators. **a.** An initial CRM logical structure. **b.** The result of applying the AND operator to the initial structure. This introduces a new binding site with an untrained PWM X . **c.** The result of the OR operator. The second binding site becomes a disjunction of the original PWM B and an untrained PWM Y . **d.** The result of the NOT operator. This is the same as the AND operator, except the new binding site is negated.

During the structure search procedure, SCRM2 must estimate the accuracy of each structure. To do this, SCRM2 calculates the accuracy of the trained model on a tuning set. Given a tuning set sequence, the HMM version of the model calculates the likelihood that the path that explains the sequence comes from the set of positive paths (those that contain all unnegated binding sites) and compares this to the likelihood summed over the the negative paths. This is translated to a probabilistic classification of each sequence. Let Π represent the set of possible paths through the HMM, and let Π^\oplus represent the set of *positive* paths through the HMM. The classification of an example (\mathbf{x}, y) is calculated as:⁴

$$P(y = \oplus | \mathbf{x} : \Theta) = \frac{\sum_{\pi \in \Pi^\oplus} P(\mathbf{x} | \pi) P(\pi : \Theta)}{\sum_{\pi \in \Pi} P(\mathbf{x} | \pi) P(\pi : \Theta)} \quad [4.2]$$

where Θ represents the parameters of the HMM. These predictions are evaluated on a tuning set to see how well the predictions match the sequence labels, depending on the user-defined scoring metric.

4.4 Efficient Computation

To classify an example (\mathbf{x}, y) we must calculate the probability that the sequence \mathbf{x} takes a positive path through the HMM. This is given by Equation 4.2. To understand how to efficiently compute $P(y = \oplus | \mathbf{x})$, consider Figure 4.8. Here, the CRM model has two binding sites, A and B , as shown in Figure 4.8a. SCRM2 considers each possible order separately, so assume the binding site order is fixed, with A upstream of B . Figure 4.8b shows two possible locations for A and B on sequence \mathbf{x} , which is of length L . A is at location i , which means the first character of motif A is the i th character in \mathbf{x} . Figure 4.8c shows a probability distribution over such locations for both motifs A and B . Figure 4.8d shows the forward dynamic programming (DP) matrices f for the locations of states A and B (recall our discussion of dynamic programming for hidden Markov models in Section 2.2.2). Although they are not shown in Figure 4.8, the same ideas are used for the backward DP calculations. f represents the likelihood of the sequence up to a point, given that the sequence at that point is explained by a certain state. These partial probability sums are used to effectively consider all paths through the model.

$$f_S(k) \equiv P(\mathbf{x}_{[1..k]} | \pi_k = S) \quad [4.3]$$

where $\mathbf{x}_{[1..k]}$ represents the subsequence of \mathbf{x} from position 1 to position k and π_i represents the state occupied at position i in the sequence. Since we consider *combinations* of motifs (*i.e.* the order and distance matter), we have to consider all pairs (i, j) for the positions of A and B , respectively. In this case, the update equation is

$$f_B(j) = \sum_{i=1}^{i=j-W_A} f_A(i) \times P(\mathbf{x}_{[i+1..j]} | \pi_i = A, \pi_j = B : \Theta) \quad [4.4]$$

⁴Recall that dynamic programming allows us to effectively sum over all possible paths through the HMM without explicitly enumerating them.

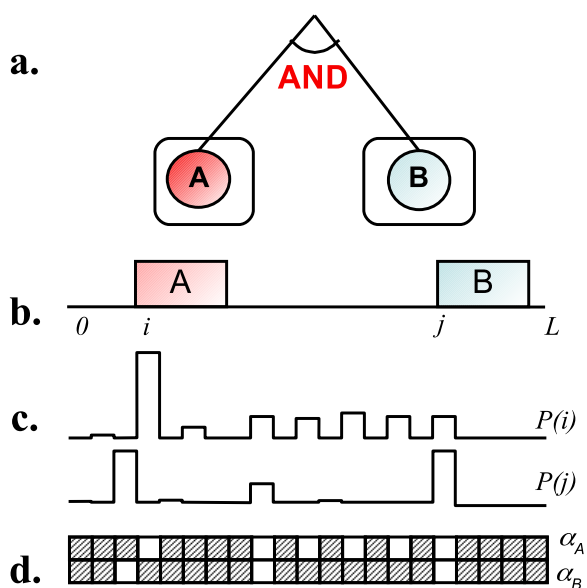


Figure 4.8 Illustration of efficient dynamic programming in SCRM2. **a.** A CRM logical structure. **b.** Possible binding site locations on a DNA sequence x . **c.** A probability distribution over the locations of binding sites A and B , respectively. These probabilities tend to be extreme (a motif is present at a location or it is not) and high probabilities are sparsely distributed. **d.** A forward dynamic programming matrix f , where $f_A(i)$ represents the likelihood of sequence x from location 1 to i when site A occurs at location i .

where W_A is the width of motif A and Θ represents the parameters of the HMM.

The number of (i, j) pairs, and therefore the run-time complexity of calculating inference for a single training sequence, is $O(L^2)$ for a sequence length L . To *update* the background states' emission distributions, we need $O(L)$ time to read the sequence data between each i and j . This makes the running time of the SCRM2 learning algorithm $O(L^3)$, which is prohibitively expensive for long sequences.⁵ However, since the background states are simple (represented by a low-order Markov chain), and explain most of the sequence data, it is a reasonable approximation to estimate their emission distributions as a pre-processing step and skip the update, reducing the running time back down to $O(L^2)$.

$O(L^2)$ is still too expensive for long sequences.⁶ However, the places where a motif occurs in a sequence will have multiple positions where the sequence character matches the PWM column preference and a relatively high likelihood. Places where motifs do not occur will have many poor matches and a relatively low probability. Most of the probability that effects the updates in (4.4) will come from these likely motif “hits.” Therefore, to speed up the process, SCRM2 first precomputes $P(\pi_i = A)$ for all i , $P(\pi_j = B)$ for all j , and so on for each motif in the CRM model. These computations take $O(L)$ time. SCRM2 then sorts these locations by likelihood, which is $O(L \log L)$. Finally, when updating the DP matrices, SCRM2 considers *only the most likely motif locations*. The running time of the efficient algorithm is still $O(L^2)$, but with a significantly reduced constant factor. Since most of the probability mass is contained in just a few of the most likely locations, SCRM2 can consider all the locations for each motif that make up almost all of the probability mass, and still greatly reduce the running time. This effect is shown in Figure 4.9.

4.5 Results

I wish to test whether or not SCRM2 is able to learn accurate CRM models. To this end, I run it on several data sets, using cross-validation to measure the predictive accuracy of our learned models. For the data sets that I consider, I train a 5th-order HMM on promoter sequences in the same genome to use as the background distribution. I use F1 for the scoring metric given to the SCRM2 function. However, since the CRM models make *probabilistic* predictions, F1 is based on *expected* true positive (false positive, false negative) rates. Precision and recall are defined as

$$P = \frac{\sum_{(\mathbf{x}, y)} P(y = \oplus | \mathbf{x} : \Theta) \delta(y = \oplus)}{\sum_{(\mathbf{x}, y)} P(y = \oplus | \mathbf{x} : \Theta)} \quad [4.5]$$

$$R = \frac{\sum_{(\mathbf{x}, y)} P(y = \oplus | \mathbf{x} : \Theta) \delta(y = \oplus)}{\sum_{(\mathbf{x}, y)} \delta(y = \oplus)} \quad [4.6]$$

⁵The running time of the SCRM2 algorithm also depends on the number of binding sites, the order of the Markov-chain background distribution and the width of the motifs, but these are small bounded values. The rate-limiting factor is the L terms.

⁶Note that, when there are three binding sites, A, B, C (in that order), the running time is still $O(L^2)$, since the location of motif A is conditionally independent of the location of motif C , given the location of motif B .

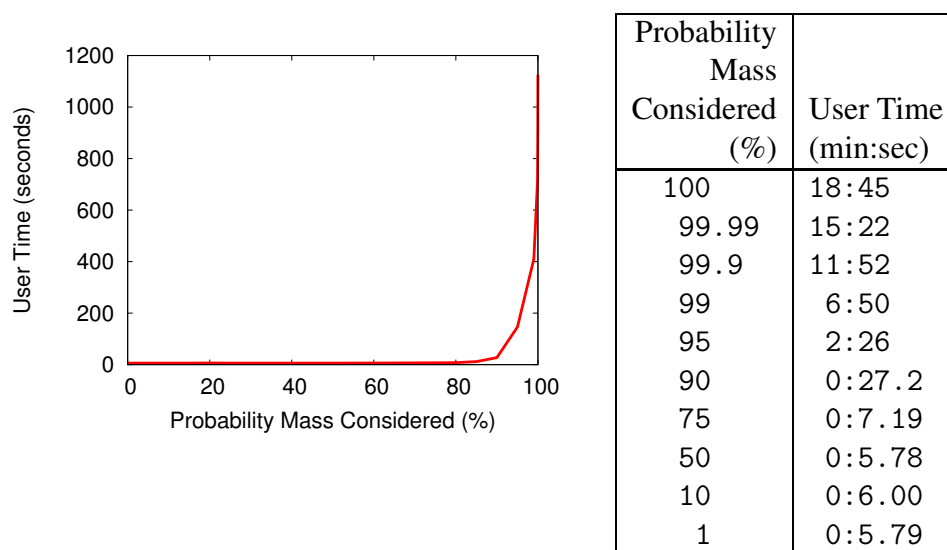


Figure 4.9 Time to train one two-binding site CRM model on 500 bp yeast sequences from [Lee *et al.*, 2002] as a function of the motif location probability mass examined during SCRM2's dynamic programming calculations.

where δ returns 1 if its argument is true, 0 otherwise, and is used to separate out the positive examples from the rest. F1 is defined as it was previously, $\frac{F1=2PR}{P+R}$.

4.5.1 Evaluating Predictive Accuracy

In order to test the algorithm’s effectiveness in identifying CRMs, I compare my approach to that of Segal and Sharan [2004] on the same data sets. I recreate 25 yeast data sets, as described by Segal and Sharan, where each gene in a given set has evidence (p -value < 0.001) from the genome-wide analysis of Lee *et al.* [2002] that two particular proteins bind somewhere in its upstream region. For each data set, I use 500 bp promoter sequences and choose 100 yeast promoter sequences at random to use as negative examples, following Segal and Sharan. The sequences are obtained from the University of California, Santa Cruz Genome Browser [Karolchik *et al.*, 2004].

To show that the predictions of the learned models on held-aside data are more accurate than could be obtained by chance, I compute a *classification margin*, following Segal and Sharan. To define the classification margin, let t be a probability threshold between zero and one. We predict that an example is positive if and only if the likelihood, according to the model, is above t , *i.e.* if $P(y = \oplus | \mathbf{x} : \Theta) > t$. The classification margin of a validation set is

$$margin = \max_t \sum_{(\mathbf{x}, y)} P(y = \oplus | \mathbf{x} : \Theta) \delta(y = \oplus) - \sum_{(\mathbf{x}, y)} P(y = \oplus | \mathbf{x} : \Theta) \delta(y = \ominus) \quad [4.7]$$

If there is less than a 1% chance that a randomly-labeled test set with the same cardinality of positive and negative examples would have the classification margin of one of the test sets (or a higher one), then this result is considered to be statistically significant.

The results are shown in Table 4.2. I find significant results in 21 of 25 data sets. This is compared to 17 of 25 found by SCRM1, and 12 of 25 found by the approach developed by Segal and Sharan.

Recall that the HMM models are trained using a discriminative approach. My experiments also show that SCRM2 is more accurate than when it uses a standard generative training approach. Of the 30 data sets mentioned in this section, the discriminative method finds more accurate models for 20 of them, especially on the yeast data sets described in the next section.

4.5.2 Evaluating the Effectiveness of Logical and Spatial Aspects

In order to evaluate whether the logical structure and spatial aspects of the representation improve the ability of SCRM2 to recover CRMs, I compare the SCRM2 algorithm to a restricted version wherein logical structure beyond AND is not included, and all the spatial probabilities are fixed by a uniform distribution. That is, the model space of this restricted version is simply a conjunction of motifs that may appear in any order, in any location, and so I refer to it as the “bag-of-motifs” approach. The classification margin is higher using SCRM2 than using the bag-of-motifs approach on 16 of the 25 Lee *et al.* data sets described above, as shown in Table 4.3

I test SCRM2 on four additional data sets from both yeast and fly, for which I obtain promoter sequences from genes known to be co-regulated. Table 4.4 describes these data sets and includes

Table 4.2 Results of finding CRMs in 25 yeast data sets from Lee *et al.* [2002]. Classification margins above the level of statistical significance (p -value < 0.01), which vary by data set size, are shown in bold.

Classification			Classification		
Data Set	Margin	p -value	Data Set	Margin	p -value
GAT3, PDR1	0.765	$< 1.0e-5$	GAT3, RGM1	0.467	1.9e-3
FHL1, RAP1	0.756	$< 1.0e-5$	FHL1, YAP5	0.450	2.2e-4
GAT3, YAP5	0.691	$< 1.0e-5$	MBP1, SWI4	0.440	2.0e-5
FKH2, SWI4	0.610	$< 1.0e-5$	SWI4, SWI6	0.429	1.0e-5
NDD1, SWI4	0.603	$< 1.0e-5$	MCM1, NDD1	0.424	1.8e-4
RAP1, YAP5	0.591	$< 1.0e-5$	SKN7, SWI4	0.395	2.4e-3
FKH2, MBP1	0.580	$< 1.0e-5$	FKH2, NDD1	0.350	2.5e-4
MBP1, SWI6	0.578	$< 1.0e-5$	NRG1, YAP6	0.323	5.8e-3
MBP1, NDD1	0.570	$< 1.0e-5$	GAL4, YAP5	0.313	0.053
FKH2, MCM1	0.540	1.0e-5	CIN5, NRG1	0.276	0.079
PDR1, YAP5	0.529	$< 1.0e-5$	PHD1, YAP6	0.190	0.22
ACE2, SWI5	0.509	3e-05	CIN5, YAP6	0.160	0.21
RGM1, YAP5	0.467	5.8e-4			

Table 4.3 A comparison of classification margin between SCRM2 and the bag-of-motifs approach on the yeast data from [Lee *et al.*, 2002]. Higher margins for each data set are shown in bold.

Data Set	Classification Margin		Data Set	Classification Margin	
	SCRM2	Bag-of-Motifs		SCRM2	Bag-of-Motifs
ACE2, SWI5	0.509	0.512	CIN5, NRG1	0.276	0.269
CIN5, YAP6	0.160	0.175	FHL1, RAP1	0.756	0.761
FHL1, YAP5	0.450	0.470	FKH2, MBP1	0.580	0.564
FKH2, MCM1	0.540	0.570	FKH2, NDD1	0.350	0.270
FKH2, SWI4	0.610	0.414	GAL4, YAP5	0.313	0.427
GAT3, PDR1	0.765	0.734	GAT3, RGM1	0.467	0.457
GAT3, YAP5	0.691	0.672	MBP1, NDD1	0.570	0.560
MBP1, SWI4	0.440	0.510	MBP1, SWI6	0.578	0.539
MCM1, NDD1	0.424	0.591	NDD1, SWI4	0.603	0.528
NRG1, YAP6	0.323	0.203	PDR1, YAP5	0.529	0.522
PHD1, YAP6	0.190	0.147	RAP1, YAP5	0.591	0.512
RGM1, YAP5	0.467	0.434	SKN7, SWI4	0.395	0.477
SWI4, SWI6	0.429	0.405			

Table 4.4 Descriptions and classification margins of four data sets we use to test the effectiveness of our representation’s logical structure and spatial aspects.

Data Set	Classification Margin	<i>p</i> -value
Yeast ESR induced: 270 <i>S. cerevisiae</i> genes induced under environmental stress response (ESR) [Gasch <i>et al.</i> , 2000] (1000 negatives)	0.305	< 1.0e-5
Yeast ESR PAC/RRPE cluster: 428 <i>S. cerevisiae</i> genes repressed under ESR. Promoters contain the PAC and RRPE elements [Gasch <i>et al.</i> , 2000] (1000 negatives)	0.338	< 1.0e-5
Yeast ESR ribosomal proteins: 121 <i>S. cerevisiae</i> ribosomal protein genes repressed under ESR [Gasch <i>et al.</i> , 2000] (1000 negatives)	0.495	< 1.0e-5
Fly gap system: 8 genes in the <i>D. melanogaster</i> gap system [Sinha <i>et al.</i> , 2003] (100 negatives)	0.730	8.5e-5

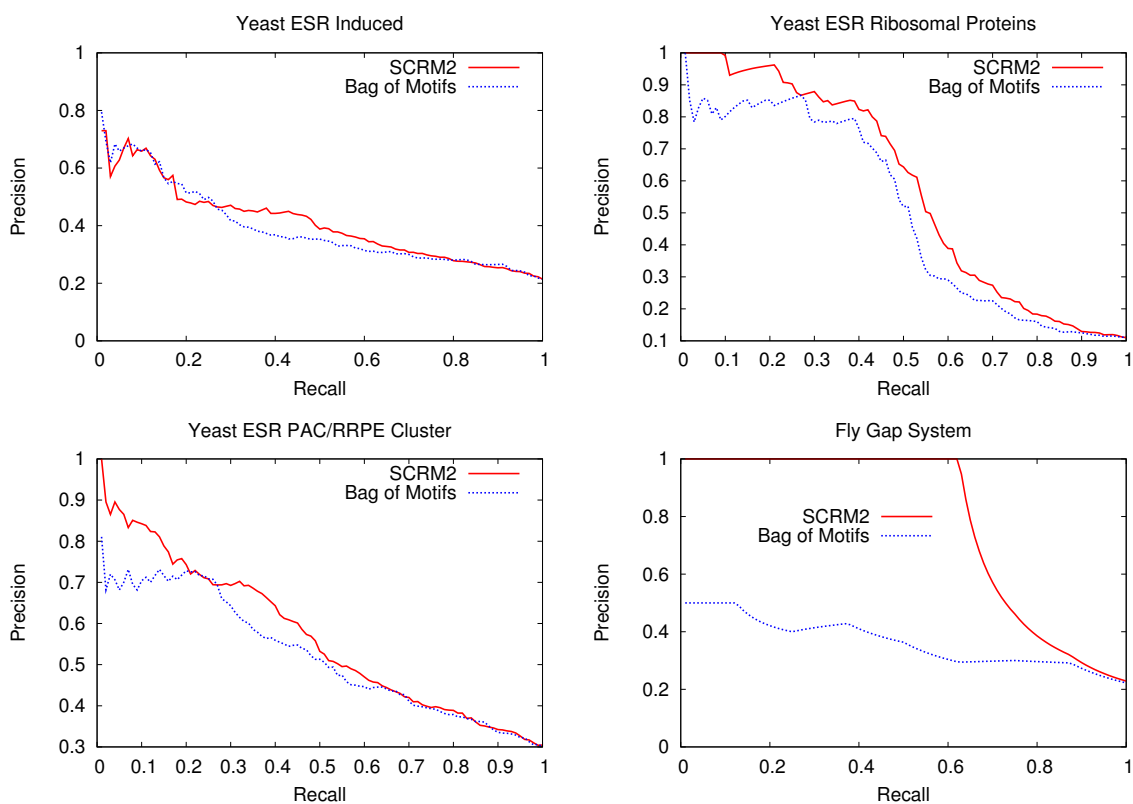


Figure 4.10 Precision-recall curves from four data sets described in Table 4.4. The accuracy of SCRM2 dominates that of the bag-of-motifs approach over almost all of the recall space in these data sets. The interpolation between points on the precision-recall curves have been calculated according to Davis and Goadrich [2006].

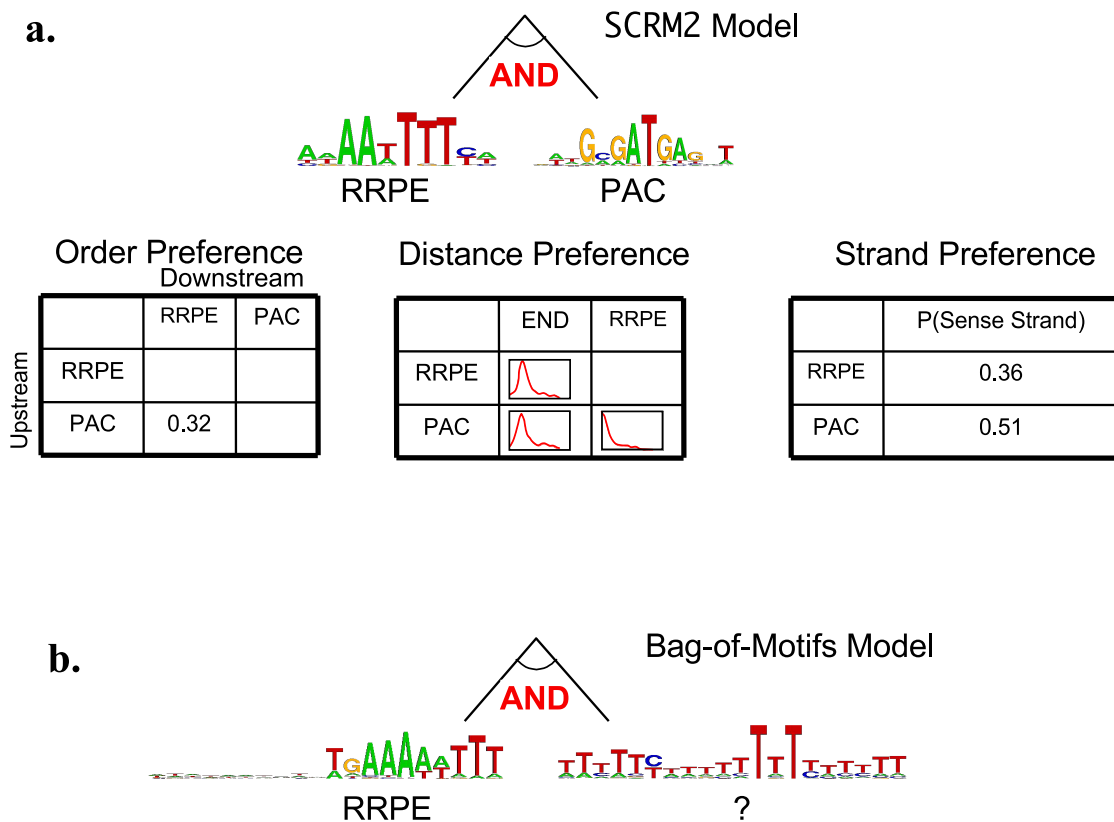


Figure 4.11 **a.** The hypothesis CRM model learned by SCRM2 for the yeast PAC/RRPE data set (Table 4.4). The model recovers both the PAC element (consensus sequence GCGATGAG) and the RRPE element (consensus sequence AAAAAwTTTTT) **b.** The model learned by the bag-of-motifs approach on the same data set. A consensus close to the RRPE element is recovered, but not the PAC element.

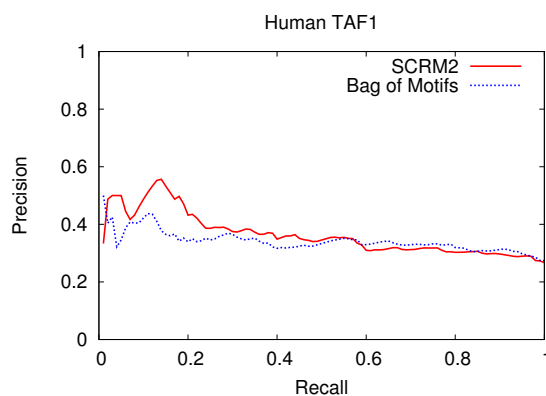


Figure 4.12 The precision-recall curve for SCRM2 compared to the bag-of-motifs approach on human DNA sequences.

a classification margin and p -value showing that SCRM2 finds statistically significant CRMs in all four data sets. Since there is a large discrepancy between the number of positive and negative examples in these data sets, I create *precision-recall* (PR) curves, which show the tradeoff between precision and recall over classification thresholds. These results are shown in Figure 4.10. In each case, the PR curve for SCRM2 dominates the PR curve for the bag-of-motifs model over all or almost all of the recall space.

The Yeast ESR PAC/RRPE genes described in Table 4.4 contain two known elements in their upstream regions, the PAC element (consensus sequence GCGATGAG) and the RRPE element (consensus sequence AAAAAA ω TTTTT). Figure 4.11 shows the hypothesis CRM model learned by SCRM2 (Figure 4.11a-d) and that of the bag-of-motifs approach (Figure 4.11e), when trained on the entire data set. The PWMs recovered by our algorithm are shown in Figure 4.11a as sequence logos [Crooks *et al.*, 2004], which show a high amount of overlap with the known consensus sequences. The bag-of-motifs approach does not recover the PAC element. This example illustrates how the inclusion of spatial preferences in the representation can aid the learner in finding better motif models. Moreover, the inclusion of these aspects leads to more accurate classifications even when the “right” motifs have been learned.

4.5.3 CRMs in Human

In order to determine whether or not SCRM2 can be effective in finding CRMs in DNA sequences in more complex organisms than yeast and fly, we test it on several human promoter sequences for genes annotated with Gene Ontology term 3677, DNA binding proteins [The Gene Ontology Consortium, 2000].

This set consists of 95 positive examples of 4000 bp regions that have evidence of being bound by a transcription factor called TAF1, and 284 negative examples with evidence of not having a TAF1 binding site⁷.

⁷These unpublished data were obtained from the J. Thomson lab at the University of Wisconsin.

The classification margin obtained from this data set is 0.220 (p -value = $8.9e-4$). The comparison of precision and recall with the bag-of-motifs approach is shown in Figure 4.12.

4.5.4 Incorporating Background Knowledge

It is worth noting that the SCRM2 algorithm described so far places an implicit uniform prior probability over the location of motifs. That is, the CRM motifs may appear anywhere in the input sequence with equal likelihood.

However, sources of background information about the input sequence include comparative genomics [Blanchette and Tompa, 2002; Sinha *et al.*, 2004] and hypersensitive regions [Noble *et al.*, 2005], and nucleosome occupancy [Segal *et al.*, 2006]. In the first case, one assumes that there is more selective pressure on conserving regulatory binding site motifs than the rest of the sequence and therefore one places more likelihood on sequence conserved across species as being locations for being binding site motifs. In the second and third cases, one predictively locates regions of DNA wrapped around proteins called histones. These DNA segments are more difficult for other proteins to interact with and are therefore less likely to be binding sites.

SCRM2 is able to account for prior distributions over motif locations. The probability is simply multiplied into the dynamic programming calculation (Equation 4.4).

I was not able to improve the results of SCRM2 using the yeast histone data from the Segal lab [Segal *et al.*, 2006] on the yeast data sets from Lee *et al.* [2002]. This is a bit surprising, but I would still argue that there is value in a probabilistic approach like the one presented in this chapter, which is better equipped to account for additional sources of evidence.

4.6 Related Work

The use of hidden Markov models to represent CRMs is not new. One example is the work of Sinha *et al.* [2003], who use HMMs to represent an order preference between adjacent motifs (see Figure 3.6). The models presented in this chapter represent a much richer set of spatial aspects, but even that is only part of the learning process. The other part is the structure search wherein SCRM2 selects the actual HMM topology to represent the CRM.

The task of learning the structure of a probabilistic grammar such as an HMM is more difficult than the task of learning the parameters of a fixed structure, which is well-studied and for which one can use maximum likelihood methods [Baum, 1972] or conditional maximum likelihood methods [Juang and Rabiner, 1991; Krogh, 1994] in an expectation-maximization framework.

There are approaches for learning HMM structure that involve starting with a standard or otherwise “reasonable” model for the task, and then merge states together [Seymore *et al.*, 1999] or split them apart [Freitag and McCallum, 2000]. One approach for learning structure that was proposed by Won *et al.* [2006] involves using a genetic algorithm for learning a fully-connected network of “blocks,” where each block is one of three types of short chains of character emitting states. These blocks are chosen because they represent common arrangements of DNA characters in a gene or promoter sequence. A more recent example was proposed by Galassi *et al.* [2007]. Their algorithm

iteratively detects and incorporates *sparse patterns* (motifs) into HMMs. What makes these examples interesting is that they do not learn arbitrary arrangements of single-character-emitting states and transitions between them, but bias the model space toward biologically-relevant submodels that appear in the literature, interspersed with short or long gaps of sequence.

Another type of probabilistic grammar is the Stochastic Context-Free Grammar (SCFG) [Rabiner, 1989; Lari and Young, 1990]. SCFGs are models of sequence data that have been used, for instance, to model the folding patterns of RNAs [Eddy and Durbin, 1994; Sakakibara *et al.*, 1994; Grate *et al.*, 1994]. They consist of *terminals* (sequence characters), *nonterminals* (variables), and *production rules* (ways of expanding nonterminals into terminals and other nonterminals). Table 4.5 shows a grammar that is equivalent to the single-motif HMM in Figure 4.3.⁸

Several methods for learning the structure (*i.e.* the set of nonterminals and production rules) of the grammar have been proposed. Stolcke and Omohundro [1994] suggest a method for introducing transformation rules in an *ad-hoc* fashion, then merging them, Kammeyer and Belew [1996] suggest a genetic algorithm for learning grammars, and Bockhorst and Craven [2001] suggest a heuristic method for learning new nonterminals and their productions by expanding existing ones.

What separates my approach from previous approaches is that it searches over an abstract grammar, which is specifically designed to represent the important aspects of the learning task. We do not add arbitrary production rules to the grammar. For instance, “motif” nonterminals always produce a sequence of single-letter nonterminals (*e.g.* the production rules for A and S_i in Table 4.5), “CRM” nonterminals always produce alternating (motif, background) nonterminals (we could add $C' \rightarrow BMC$ and $C' \rightarrow CMB$ rules for a new motif represented by M), *etc.* In other words, my approach searches over the space of logical structures for the CRM model. In this way, even large and complicated models are compact through parameter sharing.

The discussion of previous approaches to the CRM learning task from Chapter 3 is relevant here, as the models presented in this chapter capture all the same logical and spatial aspects, albeit in a more probabilistic way. SCRM2 is more comparable to previous approaches to the CRM learning task that learn binding site motifs *de novo*, however [Segal and Sharan, 2004; Zhou and Wong, 2004]. Table 4.6 shows how the representational ability of my models compares to others. To my knowledge, it is the first approach that learns the motifs of a CRM model that represents logical and spatial aspects.

4.7 Conclusion

I have presented a novel approach for learning CRM models *de novo* which performs better than a state-of-the-art approach on the 25 yeast data sets from Lee *et al.*. I have shown that learning information about the logical structure and spatial aspects of a CRM helps our learner find better models on five data sets, as measured by predictive accuracy.

⁸There is a hierarchy of grammars, known as the *Chomsky hierarchy of transformational grammars* [Durbin *et al.*, 1998]. Hidden Markov models represent what are known as *regular grammars*, which can be represented as a set of production rules of the form $P \rightarrow Qx$ or $P \rightarrow x$ (Nonterminals in upper-case, terminals in lower-case). Since Table 4.5 does not contain rules of the form $P \rightarrow xQy$, it does not show a not context-free grammar, except insofar as stochastic context free grammars are a superclass of regular grammars.

Table 4.5 The HMM in Figure 4.3, represented as a set of production rules. Nonterminals are upper-case and terminals are lower-case. The nonterminals represent an entire sequence with the motif (C), non-motif background DNA (B), a background DNA character (L), a motif (A), and all the sites within the motif ($S_1 - S_8$).

Operation	Probability
$C \rightarrow B A B$	1
$B \rightarrow L B$	0.99
$B \rightarrow L$	0.01
$L \rightarrow a$	0.32
$L \rightarrow c$	0.19
$L \rightarrow g$	0.18
$L \rightarrow t$	0.31
$A \rightarrow S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8$	1
$S_1 \rightarrow a$	0.0
$S_1 \rightarrow c$	0.2
$S_1 \rightarrow g$	0.8
$S_1 \rightarrow t$	0.0
$S_2 \rightarrow a$	0.0
\vdots	
$S_8 \rightarrow a$	0.1
$S_8 \rightarrow c$	0.1
$S_8 \rightarrow g$	0.8
$S_8 \rightarrow t$	0.0

This work was originally published in Noto and Craven [2007]. I summarize the contributions as follows:

- I have presented a novel approach to learning multiple binding site motifs and the logical and spatial relationships between them simultaneously.
- My CRM models can be thought of as generalized HMMs, but they are specifically designed to represent aspects of CRMs. My approach learns their structure by searching for the logical structure of the underlying CRM, and the representation is compact because of extensive parameter sharing.
- I have presented a learning algorithm to train these HMMs, which uses a heuristic approach to make it efficient enough to learn from long sequences such as mammalian promoter regions.
- I have made the source code for SCRM2 publicly available at www.cs.wisc.edu/~noto/crm.

Chapter 5

Learning Hidden Markov Models for Regression using Path Aggregation

5.1 Introduction

A wide array of problems in speech and language processing, biology, vision, and other application domains involve learning models that map sequences of inputs into some type of output. Common types of tasks include learning models that classify sequences (*e.g.* [Krogh *et al.*, 1994]), segment or parse them (*e.g.* [Lafferty *et al.*, 2001]), or map input sequences to output sequences (*e.g.* [Bengio and Frasconi, 1995]).

In this chapter, I consider the task of learning models that map input sequences to real-valued responses. I present an approach to this problem that involves simultaneously learning a hidden Markov model (HMM) and a function that maps paths through this model to real numbers. I empirically evaluate my approach using synthetic data sets and a large collection from a yeast genomics study.

The type of task that I consider is illustrated in Figure 5.1. This is a type of regression task in that the learner must induce a mapping from a given input sequence to a real-valued response. In particular, I assume that the real-valued responses can be represented as a function of the presence and arrangement of particular probabilistic *motifs* that occur in the sequences. I assume that these motifs are not given to the learner, but instead must be discovered during the learning process. The learner must also determine the extent to which each motif and its relationships to other motifs contribute to the response variable.

This research is motivated by a class of problems in computational biology that involve inferring the extent to which particular properties of genomic sequences determine certain responses in a cell, *e.g.* the presence of activator and repressor transcription factor binding sites determine, in part, the transcription rate of the gene. Moreover, the number of binding motifs, their arrangement in the sequence, and intrinsic properties of the motifs themselves may contribute to the response level of the gene. Thus, in order to explain the expression levels of genes in some condition, a model needs to be able to map these sequence properties into continuous values.

The approach that I present involves simultaneously learning (i) the structure and parameters of a hidden Markov model, and (ii) a function that maps paths through the model to real-valued responses. The hidden Markov model is able to represent the relevant sequence motifs and the

1	...acc ccgatgag aaaaattt ccgatgag tttagaaggta...	6.0
2	...aaagaaaaaaaaaaaaagaaaaagaaaaa gagatgag a...	3.2
3	...caaagg gcgatgag a taaaagcgat aaaaattttt gag...	9.1
4	...ggcgcgactcg gcgatgag atgagcagagataaagaca...	3.1
5	...gagctggatgct ataaatttctt aggtataaagtacga...	5.9
6	...aac aagaatttatt catcattaag gcgatgag actcact...	8.9
7	...ccattttttctctctttttataagaata aaatgttttt a...	6.1
8	... gcgatgag tt cactaaaa ataaatttctt tttttccaa...	9.2

Figure 5.1 An example of the sequence-based regression task. Each row in the figure represents a particular training instance. Each instance consists of a DNA sequence and an associated real-valued output. The sequences in this example contain two types of motifs; m_1 whose consensus sequence is **gcgatgag** and m_2 whose consensus sequence is **aaaaattttt**. In the tasks I consider, the motifs and their occurrences are hidden. The learning task involves discovering the motifs and their occurrences in the sequences, and inferring a function that maps motif occurrences to the real value associated with each sequence. In this example, $y \approx 3 \times v_1 + 6 \times v_2$, where v_1 represents the number of occurrences of m_1 and v_2 represents the number of occurrences of m_2 .

regression model is able to represent the mapping from occurrences of these motifs to the response variable. It is important to note that my models do not attempt to learn the sequence features that are merely overrepresented in a training set, but rather the ones that, together with a learned regression model, explain a set of response observations associated with the sequences. A key contribution of my approach is that the responses provide a training signal to the parameters of the HMM, and not just to those of the regression model.

5.2 Related Work

There is a wide variety of architectures and methods for learning HMMs [Rabiner, 1989], stochastic context-free grammars [Manning and Schütze, 1999], and related probabilistic sequence models, such as conditional random fields [Lafferty *et al.*, 2001; Sutton and McCallum, 2006]. For some types of problems, these models include continuous random variables. Typically these continuous variables depend only on a few other variables, and the dependencies are encoded at the outset. In my models, in contrast, the continuous response variable may depend on quite a few variables that characterize the input sequence, and these variables and their dependencies are determined during the learning process.

There is also large corpus of work on the topic of regression methods [Hastie *et al.*, 2001]. Most regression methods assume that each instance is represented using a fixed-size set of predefined variables. My approach, on the other hand, assumes that each instance is represented by a sequence of values, but these sequences may vary in their lengths and the positions of the relevant sequence elements may vary as well. Moreover, my method is designed to derive a set of variables, from the given sequences, that are predictive of the response variable.

There are kernels defined over sequences that provide a mapping from sequence features to real numbers. These string kernels can be used to map sequences to feature vectors which can then be used for regression or classification [Leslie *et al.*, 2002]. However, the kernels encode predefined sequence features. In contrast, my method is designed to learn which sequence features best provide input to the regression part of the model.

Several inductive logic programming (ILP) methods for learning regression models have been previously developed [Kramer, 1996; Karali and Bratko, 1997]. The algorithms are similar to mine in that they can handle variable-sized descriptions of instances and they employ an expressive representation for regression tasks. They differ from my approach in that they are not designed to discover sequence motifs and use properties of these motifs in the regression model. This aspect of my approach is essential for the problems I consider.

A variety of methods have been developed for discovering motifs in biological sequences [Lawrence *et al.*, 1993; Bailey and Elkan, 1995; Li and Tompa, 2006], and for identifying arrangements of motifs that are involved in particular biological processes [Segal and Sharan, 2004; Zhou and Wong, 2004]. These methods are designed for either unsupervised pattern discovery or supervised classification tasks. They either try to find motifs that are over-represented in a given set of sequences, or they try to find motif arrangements that distinguish two given sets of sequences.

My method, in contrast, is intended for regression tasks. My learned models are able to discover relevant motif arrangements and map them to quantitative gene responses.

Finally, the models and learning algorithm that I present in this chapter are focused on selecting features based on their performance on the training (or tuning) data set. In this respect, they are similar to nFOIL [Landwehr *et al.*, 2005], SAYU [Davis *et al.*, 2005], and MI-SAYU [Davis *et al.*, 2007], except that my approach learns sequence features.

5.3 Approach

The task that I consider is to learn a function that maps a given discrete character sequence $\mathbf{x} = \{x_1, x_2, \dots, x_L\}$ to a real-valued scalar y . In this section I describe the representation I employ and discuss the procedure I use for learning the models.

5.3.1 Model Representation

I assume that there are certain features, or *motifs*, present in each sequence \mathbf{x} that determine the associated y value. Thus, my approach involves learning the structure and parameters of a hidden Markov model that represents these motifs. The other key component of the learned model is a regression function that maps from occurrences of the motifs to y values.

In particular, I associate certain states in the HMM with motifs, and represent the putative occurrences of motifs in a given sequence by keeping track of the number of times that each of these states is visited. That is, a subset of the states $C = \{c_1, c_2, \dots, c_M\}$ in the HMM are designated as “counted” states, and a path through the model defines an integer vector $\mathbf{v} = \langle v_1, v_2, \dots, v_M \rangle$, where each v_k is the number of visits to state c_k . Roughly speaking, the real-valued response y is a function of these visit counts,

$$y = f(\mathbf{v}). \quad [5.1]$$

More generally, we have uncertainty about the “correct” path through the model, and therefore uncertainty about the number of visits to each state c_k . Consider the HMM shown in Figure 5.2a. There are two types of motifs, each two characters long. The motif occurrences are assumed to be interspersed with variable-length “background” sequence which is modeled by the BG state.¹ In this HMM, I count visits to each motif (*i.e.*, $c_1 = m_{12}$, $c_2 = m_{22}$). Figure 5.2b. shows the corresponding graphical model when processing a sequence of length L . Each circle represents a random variable in the model, and edges represent direct dependencies. Probabilistic dependencies are shown with solid lines and deterministic dependencies are shown with dashed lines. Figure 5.2c. shows the values taken on by the variables in the model for a case in which $\mathbf{x} = \text{ac-tacaacttg}$, $y = 9.0$, and we have assumed a particular path through the HMM. This path involves going through the top motif twice and the lower motif once. I discuss each of the random variables in turn.

¹Like the HMMs presented in Chapter 4, the BG states include a probability distribution over lengths of subsequence, (as an alternative to a self-transition) making these models *generalized* [Burge and Karlin, 1997] or hidden *semi*-Markov models [Rabiner, 1989]

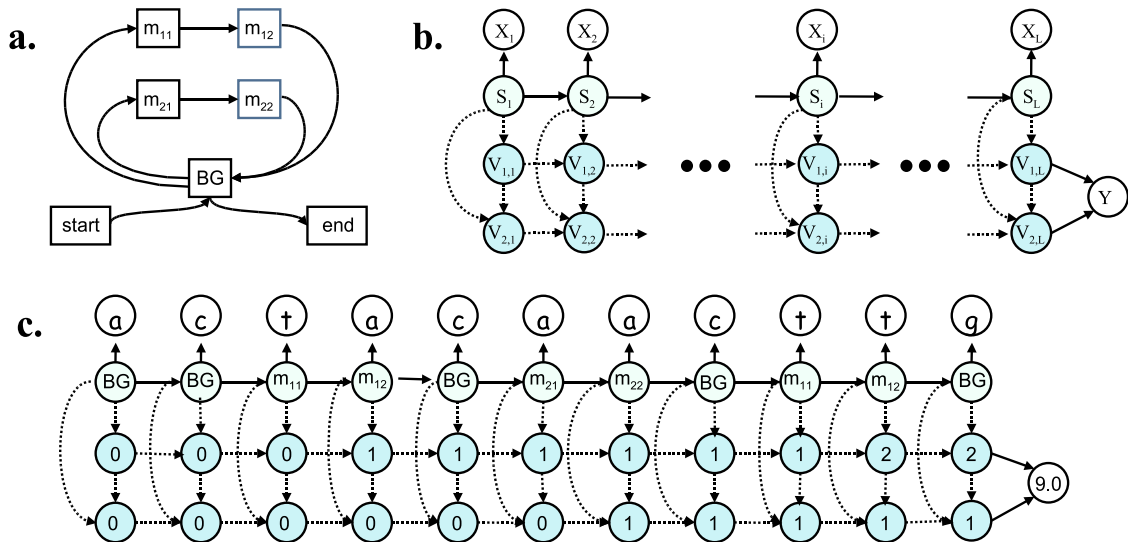


Figure 5.2 An HMM for regression and the corresponding graphical model. **a.** The state topology of a simple HMM that is able to represent occurrences of two types of motifs in given sequences. Each motif consists of exactly two DNA bases. For the k th motif, these bases are emitted by the states m_{k1} and m_{k2} . The state labeled BG emits the remaining “background” of the sequence. To calculate the distribution over the possible motif occurrences for each sequence, I count visits to states m_{12} and m_{22} . **b.** The structure of the corresponding graphical model when processing a sequence of length L . The X_i variables represent the observable sequence characters. The S_i variables represent the corresponding HMM state for each position in the input sequence. The $V_{1,i}$ ($V_{2,i}$) variables represent the number of visits to state m_{12} (m_{22}) at or before the i th character in the input sequence. The Y variable represents the real-valued response for the given instance. Probabilistic dependencies are illustrated using solid lines and deterministic dependencies are illustrated using dashed lines. **c.** The instantiation of variables in the graphical model for the instance (actcaacttg, 9.0) and a particular path through the HMM that visits m_{12} twice and m_{22} once.

Each variable X_i represents the i th character in the input sequence \mathbf{x} . The variable S_i represents the HMM state that we are in after explaining the first i characters of \mathbf{x} . This state depends directly on the previous state, and this dependency is encoded by the HMM transition parameters. The variable X_i depends on the corresponding state variable S_i , and this relationship is encoded via the HMM emission parameters. In the problems I consider, these state sequences are hidden during both training and testing.

Each $V_{k,i}$ represents the number of visits to state c_k in the paths through the HMM that explain the first i characters of \mathbf{x} . These variables are also hidden and depend on the HMM state S_i and the corresponding variable from the previous position, $V_{k,i-1}$. They are updated as follows:

$$P(V_{k,i} = v | S_i, V_{k,i-1}) = \begin{cases} P(V_{k,i-1} = v - 1) & \text{if } S_i = c_k \\ P(V_{k,i-1} = v) & \text{otherwise.} \end{cases} \quad [5.2]$$

Moreover, as illustrated by the edges between the bottom two nodes in each column of Figure 5.2b., we may represent dependencies among the $V_{k,i}$ variables at the i th position. Doing this enables us to model an arbitrary joint distribution characterizing the visits to the “counted” states.

Finally, the variable Y in Figure 5.2b. is the real-valued response associated with the sequence in question. Its value depends on the number of visits to all counted states after explaining the entire sequence. Thus, the last column of visit count variables in Figure 5.2b. determines the response value, $y = f(\langle V_{1,L}, \dots, V_{M,L} \rangle)$.

I represent Y using a linear Gaussian model. Let \mathbf{V}_L denote the vector of variables $\langle V_{1,L}, \dots, V_{M,L} \rangle$, and let \mathbf{v}_L denote a particular vector of visit counts $\langle v_{1,L}, \dots, v_{M,L} \rangle$. Given a specific \mathbf{v}_L , this model represents the probability distribution of Y as a Gaussian whose mean is a linear function of the visit-count vector:

$$p(Y | \mathbf{v}_L) \equiv N(\beta_1 v_{1,L} + \beta_2 v_{2,L} + \dots + \beta_M v_{M,L}, \sigma^2) \quad [5.3]$$

$$= N(\boldsymbol{\beta} \cdot \mathbf{v}_L, \sigma^2) \quad [5.4]$$

where $N(\mu, \sigma^2)$ denotes a Gaussian distribution with mean μ and variance σ^2 . The parameters of the regression model are $\Phi = \{\langle \beta_1, \beta_2, \dots, \beta_M \rangle, \sigma\}$. Each β_k represents the contribution to the response variable of each occurrence of the motif represented by state c_k . The standard deviation σ is also a model parameter to be learned.

To keep the regression model finite, we place a limit on the number of counted visits to each c_k , but this limit can be different for each counted state. Let B_k denote the maximum number of visits to state c_k . Then letting \mathcal{V} be the possible values of the vector \mathbf{V}_L , we have

$$\mathbf{V}_L = \langle V_{1,L}, V_{2,L}, \dots, V_{M,L} \rangle \quad [5.5]$$

$$\mathcal{V} = \{\langle 0, 0, \dots, 0 \rangle, \langle 0, 0, \dots, 1 \rangle, \dots, \langle B_1, B_2, \dots, B_M \rangle\} \quad [5.6]$$

The size of \mathcal{V} (*i.e.* the number of possible values of \mathbf{V}_L) depends on \mathbf{B} :

$$|\mathcal{V}| = \prod_{k=1}^M B_k + 1. \quad [5.7]$$

Since the \mathbf{V}_L variables are hidden, we infer a distribution for Y given a sequence \mathbf{x} by marginalizing out \mathbf{V}_L . For a particular \mathbf{x} , the distribution over Y is given by:

$$p(Y|\mathbf{x}) = \sum_{\mathbf{v}_L \in \mathcal{V}} p(Y|\mathbf{v}_L)P(\mathbf{v}_L|\mathbf{x} : \Theta). \quad [5.8]$$

where Θ represents the parameters of the HMM.

Since there is uncertainty about the value of \mathbf{V}_L , there is uncertainty about which Gaussian distribution is the one implied by the model. Thus, Equation 5.8 describes a mixture of Gaussians. Figure 5.3 shows such a Gaussian mixture for the model in Figure 5.2a., assuming a few parameter settings. Recall that there are two counted states, $c_1 = m_{12}$ and $c_2 = m_{22}$. Let the maximum number of counted visits be $B_1 = B_2 = 2$, and let $\Phi = \{\beta_1 = 1.0, \beta_2 = 2.0, \sigma = 1.0\}$. There are $\prod_k B_k + 1 = 9$ possible values for \mathbf{V}_L . The likelihood of each \mathbf{v}_L is shown, for a hypothetical HMM and input sequence \mathbf{x} .

The size of \mathcal{V} increases exponentially with the number of counted states. When we are considering a few sequence motifs of interest, such as is the case in Figure 5.2, it is tractable to consider all possible values for \mathbf{V}_L in \mathcal{V} . If, however, the number of counted states is too large to consider each possible \mathbf{v}_L in Equation 5.8, then we may instead calculate the *expected* value $\hat{v}_{k,L}$ of each $V_{k,L}$ in \mathbf{V}_L independently.

$$\hat{v}_{k,L} = \sum_{v_{k,L}=0}^{B_k} v_{k,L} \times P(V_{k,L} = v_{k,L}) \quad [5.9]$$

where B_k is the maximum considered visits to c_k . When using the expected value of each $V_{k,L}$, the distribution over Y is characterized by a single Gaussian distribution.

$$p(Y|\hat{\mathbf{v}}_L) = N(\beta \cdot \hat{\mathbf{v}}_L, \sigma^2) \quad [5.10]$$

where $\hat{\mathbf{v}}_L = \langle \hat{v}_{1,L}, \hat{v}_{2,L}, \dots, \hat{v}_{M,L} \rangle$. There is no closed-form expression for the likelihood distribution over $V_{k,L}$ given \mathbf{x} , y and the model parameters that does not involve the other variables in the vector \mathbf{V}_L , since y depends on a *combination* of visits to all counted states. In this case, I sample individual values of $v_{k,L}$ by considering various possibilities for the full vector \mathbf{v}_L , and calculating $P(\mathbf{v}_L|\mathbf{x}, y : \Theta, \Phi)$.

5.3.2 Parameter Learning

I have developed a few variations of parameter learning methods for path aggregate models. The first is based on standard Baum-Welch parameter learning for HMMs [Baum, 1972; Rabiner, 1989; Durbin *et al.*, 1998]. The second is based on the discriminative learning algorithm developed by Krogh [1994], which I use for the CRM learner described in Chapter 4. With either of these, we have a choice of whether or not to sum over \mathcal{V} (the possible values of \mathbf{V}_L), or whether to use expected values of all $V_{k,L}$ in \mathbf{V}_L .

$v_{1,L}$	$v_{2,L}$	$\beta \cdot \mathbf{v}_L$	$P(\mathbf{v}_L \mathbf{x})$
0	0	0	0.05
0	1	2	0.15
0	2	4	0.05
1	0	1	0.27
1	1	3	0.35
1	2	5	0.05
2	0	2	0.05
2	1	4	0.02
2	2	6	0.01

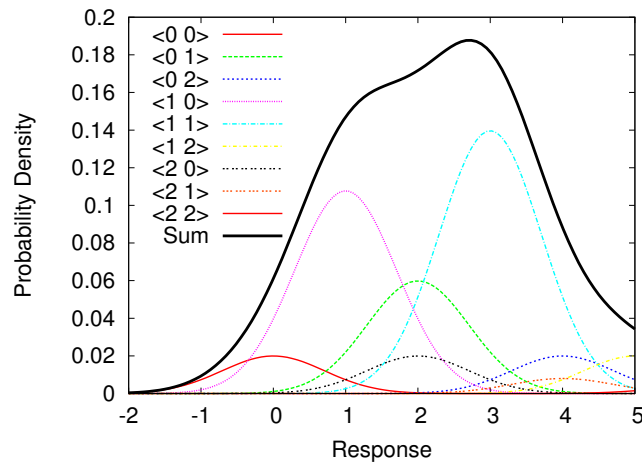


Figure 5.3 An example Gaussian mixture predicted by our model. Assume there are two counted states, and let the regression model be defined by $\Phi \equiv \{\beta_1 = 1.0, \beta_2 = 2.0, \sigma = 1.0\}$. The table shows possible values for $\mathbf{v}_L = \langle v_{1,L}, v_{2,L} \rangle$, the resulting value for $\beta \cdot \mathbf{v}_L$ and a likelihood $P(\mathbf{v}_L|\mathbf{x})$ as calculated by a hypothetical HMM and input sequence \mathbf{x} of length L . The graph shows the components of the mixture, one for each value of \mathbf{v}_L , along with the sum, which is the density function $p(y|\mathbf{x})$

5.3.2.1 Baum-Welch Style Parameter Learning

Given an HMM structure, I select parameter values to maximize the joint probability of the observed input sequences and their associated response values.

$$\hat{\Theta}, \hat{\Phi} = \arg \max_{\Theta, \Phi} \prod_{(\mathbf{x}, y)} P(\mathbf{x}, y : \Theta, \Phi) \quad [5.11]$$

Here the product ranges over all of the (\mathbf{x}, y) pairs in the training set, Θ represents the usual set of HMM state transition and character emission parameters, and Φ represents the parameters of the regression model described above, β and σ .

Taking into account uncertainty in the “correct” path for each given sequence, and the dependencies represented in the model, we can express the objective function as:

$$\arg \max_{\Theta, \Phi} \prod_{(\mathbf{x}, y)} \sum_{\pi} [P(\pi : \Theta) P(\mathbf{x} | \pi : \Theta) p(y | \mathbf{v}_L = \nu(\pi) : \Phi)] \quad [5.12]$$

where π is the path through the HMM that explains \mathbf{x} , and ν is the aggregation function that maps a path to the number of visits to each counted state. This product ranges over all of the (\mathbf{x}, y) pairs in the training set, Θ represents the usual set of HMM state transition and character emission parameters, and β and σ are the parameters of the regression model described above. There is no closed-form solution for this objective function, so I train the model using an expectation-maximization (EM) approach which is a slight modification of the standard Baum-Welch algorithm for HMMs [Baum, 1972; Rabiner, 1989].

E-step: The difference between standard Baum-Welch and my approach is that my models calculate the expected values for our hidden variables taking into account y as well as \mathbf{x} . To accomplish this, I calculate a probability distribution over $\mathbf{V}_L = \langle V_{1,L}, V_{2,L}, \dots, V_{M,L} \rangle$ given \mathbf{x} , y and our model parameters by considering each possible value for \mathbf{v}_L . The probability is given by

$$P(\mathbf{v}_L | \mathbf{x}, y : \Theta, \Phi) = \frac{1}{Z} p(y | \mathbf{v}_L : \Phi) P(\mathbf{v}_L | \mathbf{x} : \Theta) \quad [5.13]$$

where Z is a normalizing constant. The base-case initialization for the backward calculations, $P(\mathbf{V}_L | y)$, is calculated directly from the regression model, and the standard Baum-Welch E-step calculates $P(\mathbf{V}_L | x)$.

M-step: I choose the HMM parameters Θ using the standard M-step of Baum-Welch. I choose the regression model parameters $\Phi = \{\beta, \sigma\}$ using standard least-squares regression, except that the possible values for \mathbf{v}_L given a training example are weighted by their likelihood given \mathbf{x} and y . Thus, I minimize the total *expected* squared difference between the observed and predicted response values in the training set. This is calculated by marginalizing over the possible values for \mathbf{v}_L , according to their likelihood $P(\mathbf{v}_L | \mathbf{x}, y : \Theta, \Phi)$, which is calculated in the E-step:

$$\hat{\beta} = \arg \max_{\beta} \sum_{(\mathbf{x}, y) \in \mathbf{D}} \sum_{\mathbf{v}_L \in \mathcal{V}} P(\mathbf{v}_L | \mathbf{x}, y : \Theta, \beta, \sigma) (y - \beta \cdot \mathbf{v}_L)^2 \quad [5.14]$$

where \mathbf{D} is the training set, and \mathcal{V} is the number of possible values of \mathbf{v}_L . Equation 5.14 has the closed-form solution:

$$\hat{\boldsymbol{\beta}} = (\mathbf{A}^T \boldsymbol{\Gamma} \mathbf{A})^{-1} \mathbf{A}^T \boldsymbol{\Gamma} \mathbf{b}. \quad [5.15]$$

\mathbf{A} is a $|\mathcal{V}||\mathbf{D}| \times M$ matrix, where $|\mathcal{V}|$ is the number of possible values of \mathbf{v}_L , $|\mathbf{D}|$ is the size of the training set, and M is the number of counted states. It consists of all possible values of \mathbf{v}_L for each training example (*i.e.* each vector in \mathcal{V} repeated $|\mathbf{D}|$ times).

$\boldsymbol{\Gamma}$ is a $|\mathcal{V}||\mathbf{D}| \times |\mathcal{V}||\mathbf{D}|$ diagonal matrix, where each value $\gamma_{i,j}$ represents the likelihood of \mathbf{v}_L in row i of \mathbf{A} , given the j th training example, *i.e.* $\gamma_{1,1} = P(\mathbf{v}_L = \langle 0, 0, \dots, 0 \rangle | \mathbf{x}_1, y_1 : \boldsymbol{\Theta}, \boldsymbol{\Phi})$.

\mathbf{b} is a $|\mathcal{V}||\mathbf{D}| \times 1$ column vector of the y response values corresponding to each row of \mathbf{A} .

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ B_1 & B_2 & \cdots & B_M \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ B_1 & B_2 & \cdots & B_M \end{bmatrix} \boldsymbol{\Gamma} = \begin{bmatrix} \gamma_{1,1} & & & \cdots & 0 \\ & \gamma_{1,2} & & & \vdots \\ & & \ddots & & \\ & & & \gamma_{1,\mathcal{V}_L} & \\ & & & & \gamma_{2,1} \\ \vdots & & & & \ddots \\ 0 & \cdots & & & \gamma_{|\mathbf{D}|,|\mathcal{V}|} \end{bmatrix} \mathbf{b} = \begin{bmatrix} y_1 \\ y_1 \\ \vdots \\ y_1 \\ y_2 \\ y_2 \\ \vdots \\ \vdots \\ y_{|\mathbf{D}|} \end{bmatrix}. \quad [5.16]$$

The value for σ is estimated from the (minimized) expected difference between our best fit line and the data points. If \mathcal{V} is prohibitively large, we can use the expected number of visits \hat{v}_k to each c_k , and solve $\hat{\boldsymbol{\beta}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$, where \mathbf{A} is based on the expected number of visits to each counted state, in each example j ,

$$\hat{v}_{k,j} = \sum_{v_{k,j}=0}^{B_k} v_{k,j} \times P(V_{k,j} = v_{k,j} | \mathbf{x}_j, y_j : \boldsymbol{\Theta}, \boldsymbol{\Phi}). \quad [5.17]$$

$$\mathbf{A} = \begin{bmatrix} \hat{v}_{1,1} & \hat{v}_{2,1} & \cdots & \hat{v}_{M,1} \\ \hat{v}_{1,2} & \hat{v}_{2,2} & \cdots & \hat{v}_{M,2} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{v}_{1,j} & \hat{v}_{2,j} & \cdots & \hat{v}_{M,j} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{v}_{1,|\mathbf{D}|} & \hat{v}_{2,|\mathbf{D}|} & \cdots & \hat{v}_{M,|\mathbf{D}|} \end{bmatrix} \mathbf{b} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_j \\ \vdots \\ y_{|\mathbf{D}|} \end{bmatrix}. \quad [5.18]$$

5.3.2.2 Krogh-Style Parameter Learning

The key difference between the parameter learning approach described in the previous section and the approach described in this section is the objective function. Instead of attempting to maximize the joint likelihood of \mathbf{x} and y , we attempt to maximize the likelihood of y , conditioned

on \mathbf{x} .

$$\hat{\Theta}, \hat{\Phi} = \arg \max_{\Theta, \Phi} \prod_{(\mathbf{x}, y)} P(y|\mathbf{x} : \Theta, \Phi) \quad [5.19]$$

I use an approach based on the gradient descent parameter learning algorithm proposed by Krogh [1994]. The parameters of the HMM are updated as follows:

$$\Theta_k^{new} \leftarrow \text{Normalize}(\Theta_k^{old} + \alpha(n_{k,j}^{\text{correct}} - n_{k,j}^{\text{all}})) \quad [5.20]$$

where Θ_k is the k th parameter of the HMM, $n_{k,j}^{\text{correct}}$ is the expected number of times Θ_k was used in *permissible* paths through the HMM when explaining the sequence \mathbf{x}_j , $n_{k,j}^{\text{all}}$ is the expected number of times Θ_k was used in all paths, α is the learning rate, and “Normalize” is a normalization operator. A permissible path, or a “correct” path, is one where the label predicted by the path matches the example’s label. The HMMs described in Chapter 4 provide a good example. Some of the paths were considered “positive” (where the motifs emitted by the path matched the requirements of the CRM model), and some paths were considered “negative.”

For the regression task we consider here, however, each path has a certain *probability* of producing the observed response. Therefore, I weight each $n_{k,j}^{\text{correct}}$ in Equation 5.20 by the likelihood that the path π produces the response y , according to the parameters of the regression model. As described above, each path π determines a vector \mathbf{v}_L , which represents the counts to each state, and the regression model uses this path aggregate. If $\nu(\pi) \rightarrow \mathbf{v}_L$ represents the mapping from a path to a vector of occupancy counts, then the weight $w_{k,j}$ on the number of times parameter Θ_k is used in permissible paths is estimated as:

$$w_{k,j} = \sum_{\pi} P(\nu(\pi)|y_j : \Phi) \quad [5.21]$$

The parameters of the regression model are also updated using gradient descent. Recall our objective function,

$$f = \prod_{(\mathbf{x}, y)} P(y|\mathbf{x} : \Theta, \Phi). \quad [5.22]$$

The β_k parameters are updated according to a learning rate α and the derivative of the objective function:

$$\beta_k^{new} \leftarrow \beta_k^{old} + \alpha \frac{df}{d\beta_k} \quad [5.23]$$

$$\frac{df}{d\beta_k} = C \frac{e^{-\frac{(y - \beta \cdot \mathbf{v}_L)^2}{2\sigma^2}} v_{k,L} (y - \beta \cdot \mathbf{v}_L)}{\sigma^3} \quad [5.24]$$

where C is a constant associated with the equation for a Gaussian distribution. Similarly, σ may also be updated according to the derivative:

$$\frac{df}{d\sigma} = C \frac{e^{-\frac{(y - \beta \cdot \mathbf{v}_L)^2}{2\sigma^2}} (\sigma^2 - (y - \beta \cdot \mathbf{v}_L)^2)}{\sigma^4}. \quad [5.25]$$

Although this method of gradient descent for learning HMM and regression parameters is a recent development which has not yet been thoroughly empirically evaluated, initial experiments indicate that it works equally well or better to recalculate σ directly from the training set variance, as described in Section 5.3.2.1.

5.3.3 Structure Learning

Our task includes learning the underlying structure of the representative HMM as well as its parameters. This structure refers to the set of states and transitions that define the HMM topology.

Although my regression approach applies to arbitrary HMM structures, I am primarily interested in the occurrence and arrangement of motifs. Instead of searching through the space of arbitrary HMM topologies by adding and removing individual states and transitions, the search operators are the addition or rearrangement of these motifs. Although my regression approach applies to arbitrary HMM structures, for a structure like the one in Figure 5.2a., the models I consider here vary only in the number of motifs. The model begins with a single motif which is initialized by sampling from sequences in the training set. The model learns the parameters for this model and continues to introduce additional motifs until we reach some maximum structure size. The learner repeats this search process from different initial parameters some fixed number of times, and returns the model structure and parameters that perform the best on the training set or a tuning set of sequences held-aside for evaluation.

Complex arrangements of motifs can be encoded directly in the HMM topology. For instance, if we are interested in conjunctions of motifs or a specific left-to-right order of motif occurrences, we search over the space of possible structures by adding paths that explicitly contain these features.

5.4 Empirical Evaluation

The task is to learn the structure and parameters of an HMM, including the local sequence patterns that characterize each motif, as well as the parameters of the regression function. I hypothesize that an algorithm that uses the real-valued response associated with each input sequence to train HMM parameters is able to learn more accurate models than an approach that does not. To test this hypothesis, I compare the path aggregate learning approach to a slightly less sophisticated two-phase version, where we first learn the HMM parameters Θ (using standard Baum-Welch), and then learn the parameters of the regression model ($\Phi = \{\beta, \sigma\}$), from Θ and the observed sequence and response data. The key difference is that the regression model is just learned once in the two-phase approach, rather than iteratively refined as described in the previous section.

Given an input sequence \mathbf{x} , we would like to predict an output response \hat{y} , and compare that to the observed response y . However, my models predict a probability density function over y , such as the one shown in Figure 5.3. To predict a single \hat{y} , I choose to use the vector \mathbf{v}_L implied by the Viterbi (most likely) path through the HMM.

$$\hat{y} = \beta \cdot \nu(\pi_{\text{Viterbi}}) \quad [5.26]$$

where ν maps a path to a vector \mathbf{v}_L of visit counts. This choice is opposed to calculating the *expected* response based on the model parameters and the input sequence, which may not have a high probability density, and to using the density itself in our evaluation metric because one exceptionally large or small density may affect the metric over a large number of sequences. To measure the accuracy of the learned models, I calculate the average absolute error on held-aside

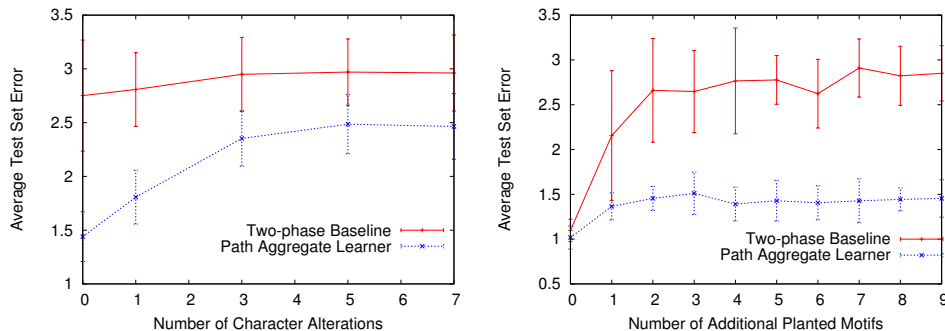


Figure 5.4 Test set average absolute error on simulated data comparing the path aggregation learning approach to a two-phase baseline. Left: Test set error as a function of mutation rate (using five additional motifs). Right: Test set error as a function of additional motifs (without mutations).

test sequences:

$$\text{error} = \frac{1}{|\mathbf{D}_{\text{Test}}|} \sum_{(\mathbf{x}, y) \in \mathbf{D}_{\text{Test}}} (y - \hat{y}) \quad [5.27]$$

I test my learner on both simulated data and real gene expression data from the species *S. cerevisiae* (baker’s yeast). For the yeast data, I hypothesize that the gene expression measurement is a function, in part, of a combination of short (*i.e.* 8-13 bases) DNA motifs in the gene’s promoter region, to which transcription factor proteins may bind to regulate the gene’s expression. For the simulated data, I create such a situation by planting known motifs in simulated DNA sequence data.

For each simulated data experiment, I generate 200-character sequences from the alphabet $\{a, c, g, t\}$. I then plant 10-character motifs in each sequence. The number of times each motif is planted comes from a Poisson distribution with $\lambda = 1$. Only two of the motifs affect the response value, which is set to $-2 + 7 \times v_1 + 3 \times v_2 + \varepsilon$, where v_k is the number of times motif k was planted in the sequence, and ε is random noise distributed normally from $N(0, 1)$. In my experiments, I vary the number of additional motifs (that do not affect response), and the “mutation rate,” where a rate of r means that r characters in each motif are changed at random before the motif is planted.

The HMM model that I use for this is similar to the one shown in Figure 5.2a., except that it varies in the number of motifs, and they are 15 characters wide. The learner searches through the space of appropriate structures by first learning the parameters for a model with one motif, then adding a second and learning the parameters of that model. This process is repeated ten times with different initial parameter settings, and the best overall model is kept, according to the accuracy on a held-aside tuning set. For each experiment, I generate 128 training sequences, 128 tuning sequences and 256 testing sequences, and I replicate each experiment several times.

Figure 5.4 shows how the accuracy of the learned models changes as a function of the mutation rate, and as a function of the number of additional planted motifs (apart from the two motifs that affect the response).² The error rate using the path aggregate learning approach is consistently

²More results on these simulated data sets are given in Appendix B.

less than that of the two-phase baseline, and tends to level off even as the number of mutations or additional motifs increases. Also, the recovery rate of the planted motifs is consistently higher using my integrated approach than that of the two-phase baseline. For instance, as I vary mutation rate and motif set size, I find that the approach returns the exact 10-character string of the motifs four times as often than the two-phase baseline. Also, with a mutation rate of three, and a decoy rate of five, my approach recovers the first motif 38% of the time (baseline model 2%), and with a mutation rate of zero, decoy rate of five, my approach recovers the first motif 72% of the time (baseline model 12%). I believe that the reason my approach learns more accurate models than the two-phase baseline is because it is able to pick out the motifs that affect the response value instead of over-represented motifs that do not.

To determine whether my approach can aid in the discovery of motifs in real genomic data, I use data from the yeast gene expression analysis of Gasch *et al.* [2000]. In these experiments, yeast cells are put in a variety of stress conditions, such as heat shock or amino acid starvation, and measurements of gene expression are taken using microarrays to determine which genes' activity is increased or decreased specifically in these conditions. I choose 15 of these experiments that have the highest degree of differential expression, but without repeating experimental conditions. From each of these, I select genes that are differentially expressed, and a control group. For each gene, I obtain 500 base pairs of promoter sequence from the University of California Santa Cruz genome browser [Karolchik *et al.*, 2004].

For these data sets, I use models similar to the HMM models described in Chapter 4, which I have previously shown to be well-suited to the task of identifying motifs in promoter data. An example of the HMM structure is shown in Figure 5.6. These models are able to represent conjunctions of motifs, and instead of counting motifs, the regression model is based on which *combination* of motifs occurs in a sequence. We search over the space of possible structures by adding new motifs to the existing model, which affects the topology of the entire HMM. Due to time constraints, I limit this search to a maximum of two motifs. As one additional baseline, I include a model that always predicts the average training set expression as the response:

$$\bar{y} = \frac{1}{|\mathbf{D}|} \sum_{(\mathbf{x}, y) \in \mathbf{D}} y. \quad [5.28]$$

The results are shown in Figure 5.5. The models learned by the path aggregate approach are more accurate than the two-phase approach for 13 of the 15 data sets. Eight of these 13 are measured as being statistically significant using a two-tailed, paired *t*-test over ten cross-validation folds at a *p*-value of 0.05. The models learned by my approach are more accurate than the training set average baseline on 12 of the 15 data sets (10 of these 12 are statistically significant).

5.5 Conclusion

I have presented a novel approach for learning HMM models for sequence-based regression tasks. My approach involves simultaneously learning the structure and parameters of an HMM, along with a linear regression model that maps occurrences of sequence motifs to the response

variable. My experiments indicate that integrating the processes of learning the HMM and the associated regression model yields more accurate models than a two-phase baseline regression approach that first learns the HMM and then subsequently learns a regression model. Note that this baseline is fairly sophisticated, compared to many methods for sequence-based regression, in that it learns sequence features instead of relying on a fixed, pre-defined set of features to represent each sequence being processed. Recall also that the model space that could be considered by my approach is richer than the model space used in the evaluation. For instance, the simulated data experiments use a relatively simple HMM topology, which involves any arrangement of position weight matrices. However, the algorithm can be applied to any hidden Markov model topology and is therefore capable of representing a much richer model space. I summarize my contributions as follows:

- I have developed an algorithm that learns the relevant features of sequential data using a mapping to real-valued responses as a training signal. This algorithm learns the sequence model and regression model simultaneously.
- I have developed a method with the objective of optimizing the joint probability of the sequences and responses, and a method with the objective of optimizing the conditional probability of the responses given the sequences.
- I have shown that simultaneously learning sequence features and a regression model results in more accurate models than training a sequence model and a regression model separately, both on synthetic and genomic data sets.
- I have made the source code publicly available at www.cs.wisc.edu/~noto/pub/software/mrf (the simple motif-counting model shown in Figure 5.2a.) and www.cs.wisc.edu/~noto/pub/software/scrm3 (the CRM learner shown in Figure 5.6).

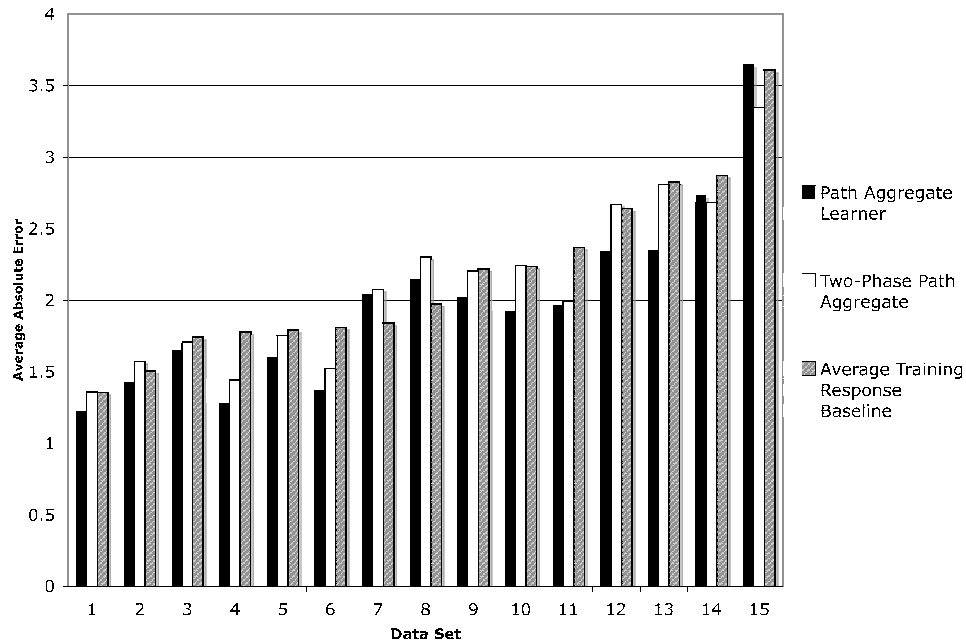


Figure 5.5 Test set average absolute error over 15 yeast microarray data sets. Results are shown for the path aggregation learning approach, a two-phase baseline, and a model that always predicts the average training set response.

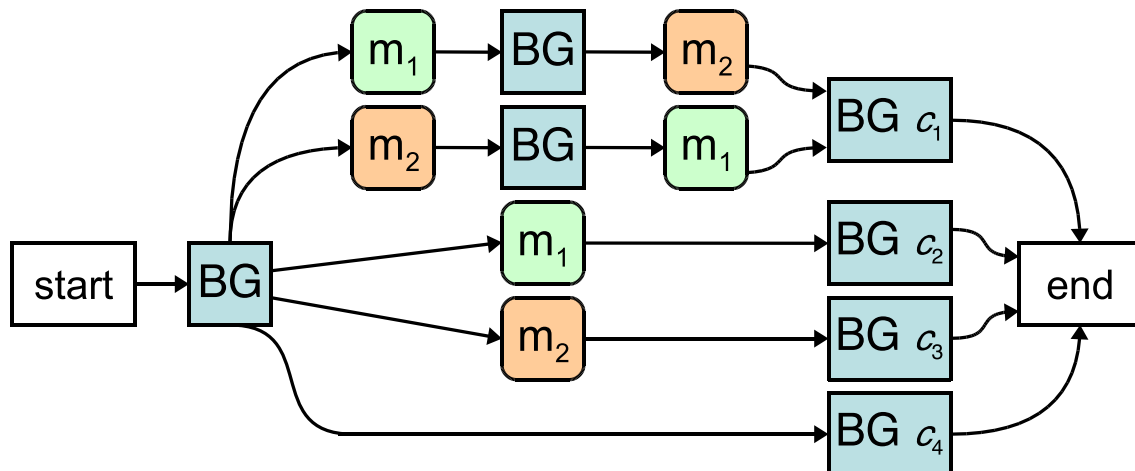


Figure 5.6 The type of HMM structure I use with the yeast data sets. Every combination of motifs is present in some path through the model. This example has two motifs, m_1 and m_2 , which are chains of states but encapsulated here as single states. Other characters are explained by the BG background states. Visits to states labeled c_k are counted.

Chapter 6

Learning Models of Gene-Regulatory Networks that Represent States and Roles

6.1 Introduction

In this chapter, I describe an approach concerning a different aspect of gene regulation. Namely, learning which regulators affect which genes, and how. In recent years, numerous research groups have developed methods that address the tasks of inferring regulatory [de Jong, 2002] and metabolic networks [King *et al.*, 2004] from data. Such models of biological networks can have both predictive and explanatory value. To achieve a high level of explanatory value, a model should represent the *mechanisms* of the network in as much detail as possible. In this chapter, I describe an approach to inferring regulatory networks from gene-expression and genomic sequence data. The approach incorporates innovations that attempt to provide a more mechanistic representation than those used in previous work in this area. My research in this area has focused on prokaryotic genomes, and thus I empirically evaluate the method using sequence and expression data from *E. coli* K-12 [Blattner *et al.*, 1997]. My experiments show that the models described in this chapter are able to provide expression predictions that are almost as accurate, and sometimes more accurate than baseline models with less explanatory value.

There are numerous factors that make the task of inferring networks from high-throughput data sources a difficult one. First, the available data characterizing states of cells, such as microarray data, are incomplete; they characterize the states of cells under a range of conditions that is usually quite limited. Second, there are typically high levels of noise in some of the available data sources, such as microarray and protein-protein interaction data. Third, measurements are not available for important aspects of the biological networks under study. For example, most efforts at network inference have employed only gene-expression measurements of protein-coding genes and genomic sequence data. However, in many cases gene regulation, even at the level of transcription regulation, is controlled in part by small molecules (*e.g.* IPTG inactivates the lac repressor), changes in protein states such as phosphorylation (*e.g.* arcA is activated through phosphorylation), or expression of small RNAs (*e.g.* 6S RNA associates with and regulates RNA polymerase).

Probabilistic models of gene regulation [Friedman *et al.*, 2000; Hartemink *et al.*, 2001; Pe'er *et al.*, 2001; Hartemink *et al.*, 2002; Ong *et al.*, 2002; Yoo and Cooper, 2002; Yoo *et al.*, 2002; Segal *et al.*, 2003b; Segal *et al.*, 2003a; Tamada *et al.*, 2003; Nachman *et al.*, 2004] are appealing because

they can, in part, account for the uncertainty inherent in available data, and the non-deterministic nature of many interactions in a cell. The method that I present here builds on work in learning probabilistic graphical models to characterize transcriptional regulation.

My approach involves learning Bayesian networks [Pearl, 1988] using both gene-expression data from microarrays and genomic sequence data, and incorporates a few innovations. First, the models include hidden nodes that can represent the *states* of transcription factors. It is often the case that expression levels of transcription factors alone are not sufficient to predict the expression levels of genes they regulate. Transcription factors may not bind to a particular DNA site unless (or except when) they have bound a specific small molecule or undergone some post-translational modification. By incorporating regulator states, My models are able to explain the effects that these small molecules, small RNAs, microRNAs, and other cellular conditions have on regulators. Given only microarray and genomic sequence data, we cannot directly measure these states. However, we can think of these states as latent variables and represent them using hidden nodes in Bayesian networks.

A second significant innovation in my approach is that it uses known and predicted transcription start sites to determine whether a given transcription factor is more likely to act as an activator or a repressor for a given gene. I refer to this distinction as the *role* of a regulator with respect to a gene. To estimate this role, I take advantage of a detailed probabilistic model of transcription units developed by Bockhorst *et al.* [2003]. Depending on the relative positions of a transcription factor binding site and a known or predicted promoter, we get an indication as to whether the transcription factor is acting as an activator or a repressor in a given case. My approach uses this information to guide the initialization of parameters associated with the hidden nodes discussed above.

6.2 Related Work

Two key aspects of my approach are the representation of regulator *states* and *roles*. My models learn what controls regulator states in terms of mRNA expression and other cellular conditions. This is crucial, because regulator *activity* is the real cause of transcription regulation, not just the expression of the genes that encode the regulators. Other groups have used latent variables to represent the underlying states of regulators [Hartemink *et al.*, 2001; Nachman *et al.*, 2004]. My approach differs from theirs in that it learns relationship between the regulator *state*, the regulator *expression*, and the cellular conditions under which the expression was measured. In other words, my approach involves learning the Bayesian network parents of the regulator state variables by selecting them from the regulator's expression and the set of cellular conditions. My approach learns the model parameters that determine the role of regulators with respect to genes. Sequence analysis has already been used to confirm regulatory relationships [Bannai *et al.*, 2004; Segal *et al.*, 2003b], by finding binding sites, but my approach uses sequence in a way that theirs does not, which is to initialize model parameters based on the location of these binding sites, relative to the (known or estimated) start of transcription.

Another important aspect of my approach is the addition of new regulator activity variables to the model. This means doing a *structure search*, which is more difficult than learning parameter

models, just as it is for hidden Markov models and other types of models. General approaches to Bayesian network structure learning have been suggested, *e.g.* [Friedman and Koller, 2000; Friedman and Koller, 2003], but my approach differs insofar as it adds new variables to the network. My approach is to add regulator state variables to explain gene expression that cannot be explained by the set of known or predicted regulators. Others have proposed a methods for learning Bayesian regulatory network connections with a similar bias, namely that a few regulators are connected to many regulatees [Pe'er *et al.*, 2001; Segal *et al.*, 2003a; Pe'er *et al.*, 2006].

6.3 Approach

A Bayesian network consists of two components: a qualitative one (the *structure*) in the form of a directed acyclic graph whose nodes correspond to the random variables, and a quantitative component consisting of a set of *conditional probability distributions* (CPDs). In this section, I first describe how I use Bayesian networks to represent various aspects of transcriptional regulation networks. I then discuss how to learn both the structure and the parameters of the networks.

6.3.1 Network Architecture

An example of the network models used in my approach is shown in Figure 6.1a. These models contain four distinct types of variables on three distinct levels: On the top level there are nodes that represent the expression of regulators (genes whose products regulate other genes), and also nodes that represent the cellular conditions under which various gene-expression measurements were collected. On the bottom level, there are nodes representing the expression of genes known or predicted to be influenced by the regulators on the top level (I refer to these genes as *regulatees*). On the middle level, there are hidden nodes, one paired with each regulator node. These hidden nodes represent the “states” of the corresponding regulators. The parents of each hidden node are selected from a set of candidates that includes both the corresponding regulator expression node and the cellular condition nodes. The parents of each regulatee node are the hidden nodes corresponding to the regulators known or predicted to have a regulatory influence over that gene.

Each hidden node has two possible values, which can be interpreted as “activated” and “inactivated.” As discussed in Section 6.1, regulators, such as transcription factors, are often activated or inactivated by *effectors*, such as small molecules. Although we do not have data that will allow us to directly detect the effectors for specific regulators, the network-learning algorithm can use cellular condition nodes as surrogates for these effectors. Consider for example, the transcription factor CAP which is activated by the small molecule cAMP. Our data do not contain cAMP measurements, but my method may learn that the absence of glucose in the growth medium is predictive of when CAP is activated. Thus the method has learned that glucose absence is a good surrogate for cAMP.

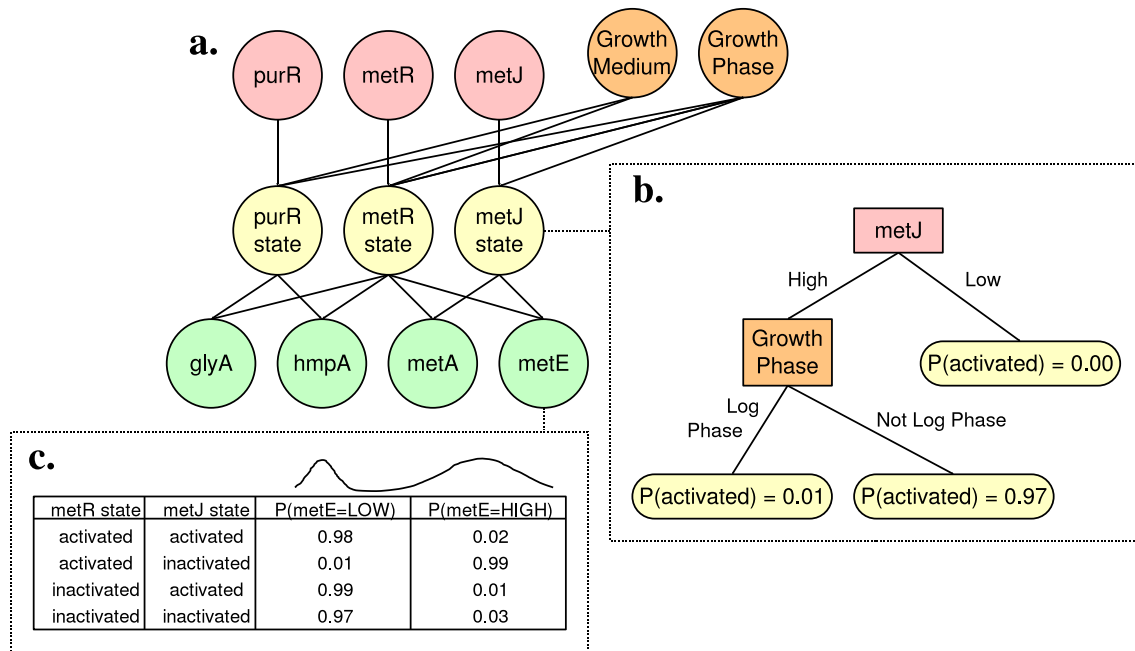


Figure 6.1 **a.** An example network with three regulators (*purR*, *metR*, and *metJ*), two cellular condition variables (Growth Medium and Growth Phase), and four regulated gene variables (*glyA*, *hmpA*, *metA*, and *metE*). **b.** A possible CPD-Tree for the hidden node *metJ*-state. **c.** A possible CPT for the regulatee node, *metE*, whose expression states are defined by a two-Gaussian mixture.

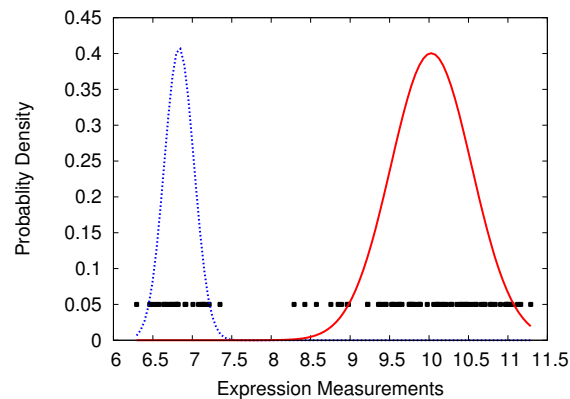


Figure 6.2 Expression measurements for the gene *metE*, and a two-Gaussian mixture that describes its states. Expression measurements are plotted near the x -axis.

6.3.2 Representing Gene Expression States

My approach represents the expression levels of genes using a Gaussian mixture model [Xing *et al.*, 2001]. I assume that most genes have multimodal expression-level distributions, with each mode corresponding to an “expression state” of the gene. Each Gaussian in the mixture represents the range of expression values for one state of the gene. Figure 6.2 shows the mixture model inferred by this method for the *metE* gene. In this case, there are two expression states, which might be referred to as the “low” and “high” expression states, indicating their relative expression levels.

I use cross-validation to choose the number of Gaussians in each mixture (see Section 2.3). Let $\mathbf{x} \equiv \{x_1, x_2, \dots\}$ be the set of expression values for a given gene. For each fold i of cross-validation, I divide \mathbf{x} into two subsets, training data \mathbf{x}'_i and held-aside data \mathbf{x}''_i . Let G be the number of Gaussians in the mixture. Let $\Phi \equiv \{(\mu_1, \sigma_1^2, w_1), (\mu_2, \sigma_2^2, w_2), \dots, (\mu_G, \sigma_G^2, w_G)\}$ be the parameters of the Gaussian mixture (the mean, standard deviation and weight of each Gaussian). Let $N(\mu, \sigma^2)$ denote a Gaussian distribution function, so that $p(x : N(\mu, \sigma^2))$ gives the density of $N(\mu, \sigma^2)$ at point x :

$$p(x : N(\mu, \sigma^2)) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \quad [6.1]$$

The probability density of x , according to a mixture Φ of G Gaussian distributions is

$$p(x : \Phi) = \sum_{g=1}^G w_g \times p(x : N(\mu_g, \sigma_g^2)). \quad [6.2]$$

I use an expectation-maximization (EM) algorithm to optimize the parameters, Φ_i of the mixture model. However, I constrain the parameters so that the Gaussians are sufficiently far apart to ensure they each cover a separate range of expression values. Specifically, each Gaussian must have the highest probability density at each expression level within two standard deviations of its mean.

The EM algorithm will attempt to optimize Φ_i as

$$\hat{\Phi}_i = \arg \max_{\Phi} \prod_{x \in \mathbf{x}'_i} p(x | \Phi) \quad [6.3]$$

subject to the constraint described above. Formally, this constraint holds that

$$\forall_{A,B} w_A \times p(\mu_A \pm 2\sigma_A : N(\mu_A, \sigma_A^2)) \geq w_B \times p(\mu_A \pm 2\sigma_A : N(\mu_B, \sigma_B^2)) \quad [6.4]$$

for all pairs of Gaussians (μ_A, σ_A^2, w_A) and (μ_B, σ_B^2, w_B) in Φ . Only a local optimum is guaranteed when using EM to optimize Equation 6.3. I then score the model for this fold as the likelihood of the held-aside data according to the learned Gaussian mixture.

$$score_i = \prod_{x \in \mathbf{x}''_i} p(x : \Phi_i) \quad [6.5]$$

I repeat this process for $G = 1, 2, 3$. I choose the number of Gaussians associated with the highest score, $\sum_i score_i$, provided that a pairwise, two-tailed t -test determines the improvement to be

statistically significant over the scores obtained from mixtures with fewer Gaussians. That is, the approach is biased toward fewer Gaussian distributions explaining the data (in accordance with Occam’s razor), and only settles on a larger mixture when doing so is statistically justified. Once the number of Gaussians has been settled, I use the EM algorithm to optimize the final parameters in the same way as was done for each fold, and consider each individual Gaussian to represent an expression state for that gene. If this method selects a mixture model with only one Gaussian for a given gene (*i.e.* there is only one expression state of this gene in the training set), then the gene is not included in the network model. In my experiments, this is the case for roughly half of the genes considered. About 90% of the remaining genes have two expression states, and 10% have three.

In order for the network to learn from data, these data must be expressed in terms of values of the variables in our network. For the regulator and regulatee nodes, these are expression states of the genes, but our evidence consists of expression measurements. Since the expression states are described by Gaussians over the expression values, it is straightforward to calculate a probability distribution over the expression states for each gene, given an expression measurement. These probability distributions are what I use as the evidence concerning the states of genes. Because of the constraints on the distance between Gaussians and the fact that they are positioned using the data, most such distributions will clearly favor a single Gaussian (gene expression state) over any other.

6.3.3 Representing Conditional Probability Distributions

As shown in Figure 6.1b., I use trees to represent the conditional probability distributions for the hidden nodes in the networks. Each tree represents the distribution over values (*i.e.* states) of the corresponding hidden node, conditioned on the values of the node’s parents. Recall that the candidate parents for each hidden node consist of the regulator expression level as well as the complete set of cellular condition nodes. I use trees to represent these CPDs for three key reasons. First, I assume that only a few of the candidate parents are relevant to modeling the regulator state, and thus I want the model to be able to select a small number of parents from a fairly large candidate pool. Second, trees provide descriptions of regulator states that are readily comprehensible and thus they can lend insight into the mechanisms that determine a regulator’s behavior. Third, the trees can account for cases in which there is context-sensitive independence in determining a hidden node’s probability distribution [Boutilier *et al.*, 1996]. In the sample tree shown in Figure 6.1b., note how **Growth Phase** is only relevant if the regulator **metJ** has expression **High**. Note also how **Growth Medium** is not chosen as a parent at all.

Using our current data set, each regulatee in the network has a relatively small number of parents (between one and four), and I expect each parent to be relevant, so I use conventional *conditional probability tables* (CPTs) for the regulatee nodes, as shown in Figure 6.1c. The CPT for each regulatee node represents the distribution over the possible expression states of the particular gene, conditioned on the possible states of its parents.

6.3.4 Learning Network Parameters

Recall that each hidden variable is binary, and I refer to the possible values as “activated” and “inactivated”. Since these states are unobserved, there is no closed-form expression from which to directly calculate the CPDs for the hidden and regulatee nodes. Instead, I set their parameters with an EM algorithm that iteratively refines the CPDs until they converge to a local optimum which is consistent with the observed training data. Let $s_{R,i}^e$ represent the observed expression state of the i th regulator in experiment e , and let $s_{r,i}^e$ represent the observed expression state of the i th regulatee. Similarly, let $s_{c,i}^e$ denote the state of the i th cellular-condition variable in experiment e . I use Θ to represent all of the parameters of a Bayesian network, including the parameters (and structure) of each CPD tree and of each CPT. The EM algorithm adjusts the parameters trying to maximize the joint probability of the expression states across all experiments, given the cellular conditions:

$$\hat{\Theta} = \arg \max_{\Theta} \prod_e P(s_{R,1}^e, \dots, s_{R,m}^e, s_{r,1}^e, \dots, s_{r,n}^e \mid s_{c,1}^e, \dots, s_{c,k}^e : \Theta). \quad [6.6]$$

Here m is the number of regulators in the network, n is the number of regulatees, and k is the number of cellular-condition variables. I assume that the learner is always given the values of the cellular-condition variables (*i.e.* we are never interested in predicting them), and thus the model represents the probability of the expression states conditioned on these values. The details of the E-step and M-step in this context are as follows.

6.3.4.1 E-Step

Let $S_{h,i}^e$ represent the (unobserved) state of hidden node i in experiment e . Let $s_{h,i}^e, s_{R,i}^e, s_{r,i}^e, s_{c,i}^e$ represent the particular values that the hidden nodes, regulator nodes, regulatee nodes, condition nodes can take (*i.e.* uppercase S represents a random variable in the Bayesian network, and lowercase s represents a particular value the variable can take). In the E-Step, the learner computes the expected distribution over values of $S_{h,i}^e$ for each e and each i , given the observed expression states of the regulators and regulatees, and the observed cellular conditions:

$$P(S_{h,i}^e \mid s_{R,1}^e, \dots, s_{R,m}^e, s_{r,1}^e, \dots, s_{r,n}^e, s_{c,1}^e, \dots, s_{c,k}^e : \Theta).$$

This is computed as a Bayesian network query using *variable elimination* [D’Ambrosio, 1999]. If the probability that a certain $S_{h,i}^e$ takes on the value “activated” is 0.7, then $S_{h,i}^e$ is treated as being 70% an “activated” data value and 30% an “inactivated” data value.

6.3.4.2 M-Step

Once these expected values are calculated, the learner uses the now complete set of data to recalculate the network parameters. Let $\Theta_{h,i}$ refer to the CPD parameters for the i th hidden node and let $\Theta_{r,i}$ refer to the CPT parameters for the i th regulatee node. In the M-step, the learner

attempts to maximize:

$$\prod_e \sum_{s_{h,1}^e, \dots, s_{h,m}^e} P(s_{h,1}^e, \dots, s_{h,m}^e) P(s_{R,1}^e, \dots, s_{R,m}^e, s_{r,1}^e, \dots, s_{r,n}^e \mid s_{h,1}^e, \dots, s_{h,m}^e, s_{c,1}^e, \dots, s_{c,k}^e : \Theta). \quad [6.7]$$

where $s_{h,1}^e, \dots, s_{h,k}^e$ denotes particular values (activated or inactivated) for the hidden nodes. Note that the likelihood of a given set of variable assignments to the hidden nodes, given by the term $P(s_{h,1}^e, \dots, s_{h,m}^e)$ in Equation 6.7, is calculated in the E-step.

Each CPD-tree for hidden variable i , represented by $\Theta_{h,i}$, is regrown by recursively selecting a variable to split on (regulator expression state or cellular condition variable) which separates the set of expected values for the hidden variable, $\{s_{h,i}^e\}$, over all experiments e , into two subsets such that the expected classification error is minimized.

The tree-growing process for a hidden node $S_{h,i}$ requires some detailed explanation. Each interior¹ node in the binary CPD-Tree represents a variable. The left subtree of each interior node represents one value of that variable, and the right subtree represents all other values. Thus, each node in the CPD-Tree is characterized by a “history” of decisions of the form (variable,value,left-or-right) that are made to reach the node from the root. Let $T \equiv \{(\mathcal{A}_1, v_1, d_1), (\mathcal{A}_2, v_2, d_2), \dots\}$ represent such a set, where d_i is either **left** or **right** indicating whether the subtree represents the set of experiments where \mathcal{A}_i has value v_i or does not have the value, respectively. At each point in the process, a leaf node becomes a new interior node by splitting on a (variable,value) pair. let (\mathcal{A}, v) represent a potential new split for a given node, T . \mathcal{A} is the split variable, and might be either $S_{R,i}$, or one of the cellular condition nodes, S_c . In either case, \mathcal{A} has some number of discrete values, one of which is v . This split creates two new subtrees. The set of experiments where \mathcal{A} does indeed have the value v are put into the *left* subtree, and the experiments where $\mathcal{A} \neq v$ go into the *right* subtree. If the model were to stop splitting at this point, and label both of the new leaf nodes either $S_{h,i} = \text{activated}$ or $S_{h,i} = \text{inactivated}$ (whichever matched more experiments in either node), there would be some hidden states misclassified, and thus a *classification error*. The learner chooses the $(\widehat{\mathcal{A}}, v)$ split to minimize this error:²

$$(\widehat{\mathcal{A}}, v) = \arg \min_{(\mathcal{A}, v)} \text{error}(T \cup (\mathcal{A}, v, \text{left})) + \text{error}(T \cup (\mathcal{A}, v, \text{right})) \quad [6.8]$$

where T is the node that we are splitting, and *error* computes the classification error of a node:

$$\text{error}(T) = \min_{\rho \in [\text{activated}, \text{inactivated}]} \sum_e \delta(\text{subset}(e, T)) \times \delta(S_{h,i}^e \neq \rho) \quad [6.9]$$

where $\delta(\text{true}) = 1$ and $\delta(\text{false}) = 0$. The term $\text{subset}(e, T)$ is true if the experiment e belongs in the set specified by node T :

$$\text{subset}(e, T) \iff \prod_{(\mathcal{A}, v, d) \in T} \delta((d = \text{left} \wedge \mathcal{A}^e = v) \vee (d = \text{right} \wedge \mathcal{A}^e \neq v)) = 1 \quad [6.10]$$

¹Interior nodes are the non-leaf nodes. In Figure 6.1b., these are **metJ** and **Growth Phase**. The interior nodes always split on a variable, and the leaf nodes always give a probability over [activated, inactivated] for the regulator state variable in question.

²The learner will not stop splitting at this point, but rather recursively split the new leaf nodes it creates. However, it calculates the classification error if it were to stop as the scoring metric for the potential split.

where \mathcal{A}^e is the value of the variable \mathcal{A} in experiment e . However, we do not have all the information we need to use (6.9) because we are using *probabilistic* data sets. We have expected values for our hidden states, and when \mathcal{A} is a regulator variable, we have a probability distribution over possible expression states. Therefore, our classification error in (6.9) is replaced with:

$$error(T) = \min_{\rho \in [\text{activated}, \text{inactivated}]} \sum_e P(\text{subset}(e, T)) \times P(S_{h,i}^e \neq \rho) \quad [6.11]$$

where $P(S_{h,i}^e \neq \rho)$ is given by the expected values we calculated in the E-step, and $P(\text{subset}(e, T))$ gives the probability that the example e belongs in the subtree:

$$P(\text{subset}(e, T)) = \prod_{(\mathcal{A}, v, d) \in T} \delta(d = \text{left}) \times P(\mathcal{A}^e = v) + \delta(d = \text{right}) \times P(\mathcal{A}^e \neq v) \quad [6.12]$$

If \mathcal{A} is a regulator variable, then $P(\mathcal{A}^e = v)$ is the probability that \mathcal{A} has the expression state v . If \mathcal{A} is a cellular condition value, then \mathcal{A}^e is known, and $P(\mathcal{A}^e = v)$ is either 0 or 1.

When growing a CPD-Tree, the process of variable-splitting, recurs on both subsets until no candidate split will further separate the data. A probability distribution over $S_{h,i}$ is then calculated for each leaf in the tree based on $\{s_{h,i}^e\}$ for the experiments e contained in that leaf's subset. Subtrees with a common ancestor that have nearly the same probability distributions over $S_{h,i}$ are pruned using an approach based on *minimum description length* (MDL), similar to one developed by Mehta *et al.* [1995]. These pruned trees may not maximize the probability of the hidden states exactly, as do their unpruned counterparts. In practice, however, pruning speeds up convergence without sacrificing accuracy. Each regulatee CPT, $\Theta_{r,i}$, is also recalculated in the standard way during the M-step using $\{s_{r,i}^e\}$ and $\{s_{\pi(i)}^e\}$, where $\pi(i)$ is the set of parent nodes of regulatee node i .

6.3.5 Initializing Network Parameters

The EM algorithm described above will converge to a local optimum. In order to guide the network to converge to a good solution, the learner initializes the CPT for each regulatee based on prior knowledge about the roles of each its regulators. Specifically, for each regulatee, I consider the relative location of the transcription start site, which is either known or predicted [Bockhorst *et al.*, 2003], and the binding sites for the regulators, which are also either known or predicted. The learner tentatively designates as *activators* those regulators that bind strictly upstream of the regulatee's RNA polymerase binding site (which is estimated to extend 35 nucleotides upstream of its transcription start site), and the learner tentatively designates all other regulators as *repressors*. In the CPT for each regulatee, the learner assigns a higher probability to the *highest* expression state when the putative activators are in the "activated" state, and it assigns a higher probability to the *lowest* expression state when the putative repressors are in the "activated" state. I put more weight on this effect for repressors, which are believed to have a more stringent control on expression, and I put more weight on this effect when the regulatee's transcription start site is known than when it is only predicted. This initialization process is illustrated in Figure 6.3.

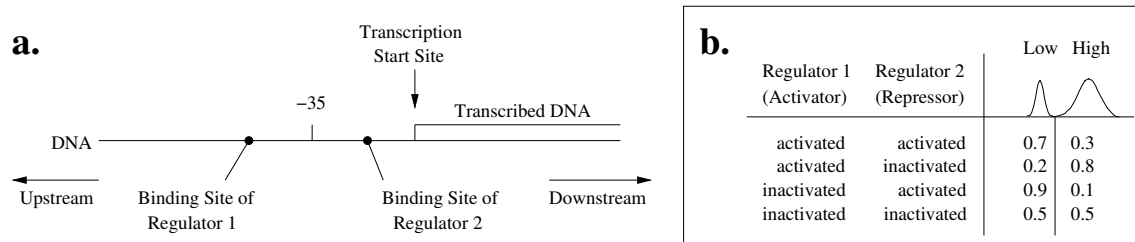


Figure 6.3 **a.** An example promoter configuration with one regulator binding site on each side of the -35 position. **b.** A CPT for the gene that has been initialized based on the configuration of these two regulator binding sites.

6.4 Empirical Evaluation

In this section I present experiments designed to evaluate Bayesian networks that (a) use hidden nodes to represent regulator states, (b) attempt to compensate for missing regulators, and (c) have the parameters associated with these nodes initialized to reflect their predicted roles (activator or repressor) with respect to individual genes. I hypothesize that these innovations will lead to more accurate models as well as models that better represent the mechanisms of gene regulation.

6.4.1 Experimental Data and Methodology

I initialize the topology of the networks using 64 known and predicted *E. coli* regulators and their 296 known and predicted regulatees. The known instances are from TRANSFAC [Wingender *et al.*, 2001] and EcoCyc [Karp *et al.*, 2002], and the predicted instances are based on binding sites predicted from cross-species comparison [McCue *et al.*, 2002]. The gene-expression data comes from a set of 90 Affymetrix microarray experiments [Glasner *et al.*, 2003]. Each array is annotated with experimental conditions, and the data are normalized using the *robust multiarray averaging* (RMA) technique [Irizarry *et al.*, 2002].

I divide the microarray experiments into sets for which all of the annotated cellular conditions are identical (I call these replicate sets). From the original 90 experiments, there are 42 of these sets. The largest contains five experiments, and the others are copied so that each replicate set contains exactly five experiments. To assess model accuracy, I use leave-one-out cross-validation on each replicate set. That is, I hold one set of five identical experiments aside, train the model on the remaining experiments, and then evaluate the network on the held-aside data. For each testing example, I provide the network with expression levels of the regulators and the values of the cellular conditions, and then calculate a probability distribution over the possible expression states for each regulatee.

I evaluate the accuracy of the models using three measures. First, I calculate *classification error* as the extent to which the network predicts the incorrect expression states for each regulatee and experiment. Instead of calculating this error using “hard predictions” of the expression state of each regulatee, I take into account the uncertainty in each predicted expression state as well as

the uncertainty in the discretization of gene expression values. In particular, I calculate classification error in predicting the regulatee expression state S_r for each of n regulatees in each of E experiments as follows:

$$\text{class error} = 100\% \times \left(1 - \frac{1}{E} \frac{1}{n} \sum_{e=1}^E \sum_{i=1}^n \sum_d P(S_{r,i}^e = d : \Theta) P(S_{r,i}^e = d | x_{r,i}^e : \Phi) \right). \quad [6.13]$$

where $P(S_{r,i}^e = d : \Theta)$ is the likelihood that $S_{r,i}^e$ is in expression state d according to the network parameters Θ , and $P(S_{r,i}^e = d | x_{r,i}^e : \Phi)$ is the likelihood that the observed expression $x_{r,i}^e$ belongs to expression state d , according to the Gaussian mixture, Φ .

The second measure is the *average squared error*, where the error is the difference between the actual expression value and the means of the Gaussians representing each expression state, weighted by the predicted distribution over these states:

$$\text{ASE} = \frac{1}{E} \frac{1}{n} \sum_{e=1}^E \sum_{i=1}^n \sum_d P_{\Theta}(S_{r,i}^e = d) (\mu_{r,i,d} - x_{r,i}^e)^2. \quad [6.14]$$

Here, $\mu_{r,i,d}$ is the mean of the Gaussian for state d in the mixture model for the i th regulatee.

Finally, the third measure is the joint *log probability* of all test-set expression values, again taking into account the uncertainty in each predicted expression state and in discretization of each gene:

$$\log \text{probability} = \sum_{e=1}^E \sum_{i=1}^n \log \left(\sum_d P_{\Theta}(S_{r,i}^e = d) P_{\Phi}(S_{r,i}^e = d | x_{r,i}^e) \right). \quad [6.15]$$

I apply pairwise, two-tailed t -tests to test the statistical significance of differences between methods.

6.4.2 Experiment 1: The Value of Representing Regulator States

Probabilistic models of gene regulation have already been developed wherein gene expression is modeled as a function of regulator activity [Pe'er *et al.*, 2001]. In order to test the value of including hidden nodes that represent regulator states, I compare against two baselines, examples of which are shown in Figure 6.4. The first baseline employs Bayesian networks that have nodes representing the expression levels of regulators and regulatees, but which do not have hidden nodes representing regulator states (Figure 6.4a.). The second baseline augments the first by also incorporating cellular condition nodes, but it too does not have hidden nodes (Figure 6.4b.).

Table 6.1 shows the totals over all test folds for all three measures. For each of these measures, my models are more accurate than the baseline models that do not have hidden or cellular condition nodes. The differences are statistically significant at a p -value of less than 0.01 for classification error and average squared error. The difference in overall probability is not statistically significant at a p -value of less than 0.05. The baseline networks that include cellular condition nodes provide slightly more accurate predictions than the models with hidden nodes. However, I argue that these

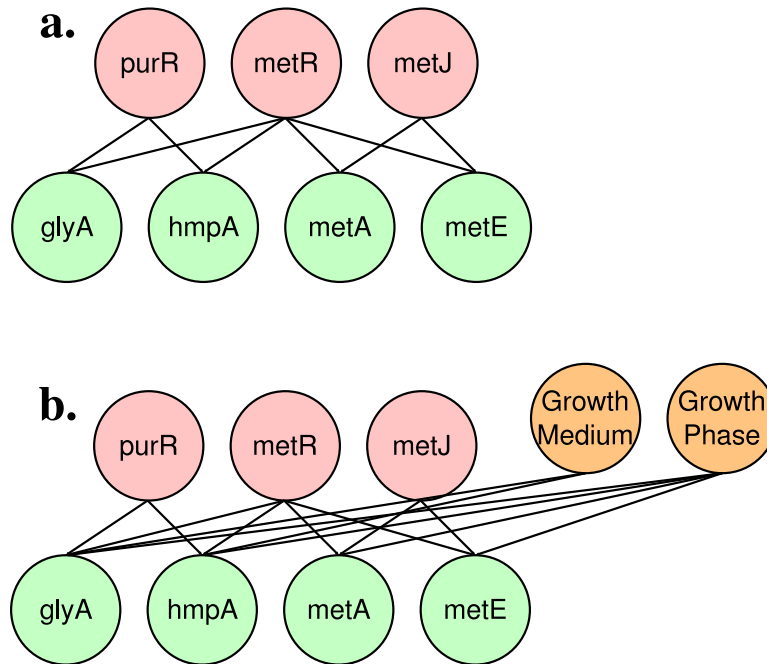


Figure 6.4 Examples of the two baseline networks used in Experiment 1. These are the counterpart baselines for the network shown in Figure 6.1. **a.** Baseline network without hidden states and without cellular conditions. **b.** Baseline network without hidden states but with cellular conditions.

Table 6.1 Predictive accuracy for the models with hidden nodes and the two baselines.

Model Variant	Classification Error	Average Squared Error	Log Probability
Full Model	16.59%	0.59	-12,066
Without Hidden Nodes	12.42%	0.51	-12,193
Without Hidden or Cellular Condition Nodes	22.16%	0.75	-13,363

Table 6.2 Predictive accuracy for the models with added hidden nodes.

Iterations	Classification Error	Average Squared Error	Log Probability
0 (Original model)	16.59%	0.59	-12,066
1	14.23%	0.53	-11,586
2	13.65%	0.51	-11,987
3	13.34%	0.51	-12,004

baseline models have a significant limitation in they do not provide a very mechanistic description of regulatee expression. That is, they do not directly represent the states of regulators and how these states govern the expression of regulatees. Thus, they have less explanatory power than my models.

Note also that my models show an improvement in overall log probability when compared to each of these baselines. Since the overall probability is a product of regulatee expression probabilities, an incorrect prediction with a probability very close to zero can have an unbounded effect on the final measurement. I hypothesize that my models make fewer of these extreme probability predictions because the regulatees are constrained by binary-valued parents.

6.4.3 Experiment 2: Discovering Missing Regulators

It is certainly the case that some relevant regulators are not represented in my networks. In this section, I consider a simple approach that dynamically adds hidden nodes to the networks. This approach tries to identify sets of regulatees for which a network makes incorrect predictions on many of the same training examples. After first training a network using the EM approach described earlier, I recursively cluster regulatees that share at least 50% of training examples incorrectly predicted by either regulatee (or cluster). A new hidden node is created for each cluster and this node becomes a parent of the regulatees in the cluster. The network is then re-trained using the EM approach. This procedure may be iterated a number of times.

Table 6.2 shows the resulting cross-validation accuracy with up to three iterations of this procedure. Each iteration decreases the classification error of the models, and each decrease is statistically significant at a p -value less than 0.03. For the average squared error measure, only the decrease in error from the original model to any of the other three is statistically significant (at a p -value of 0.05 or less). The application of this procedure improves the overall probability, but it does not continue to increase over multiple iterations. The differences in overall probability are not statistically significant.

Notice that this technique improves all three of the accuracy measurements, and that the classification error approaches that of the baseline without hidden nodes shown in Table 6.1, yet still provides models that explain relevant regulatory mechanisms.

Table 6.3 Predictive accuracy for models with promoter-based parameter initialization and random initialization.

Initialization	Classification Error	Average Squared Error	Log Probability
Using Promoter Data	13.34%	0.51	-12,004
Random	14.19	0.54	-11,893

6.4.4 Experiment 3: The Value of Initializing Regulator Roles

Recall that, before I train my network, I initialize the CPTs of the regulatee nodes based on the relative positions of known and predicted regulator binding sites and known or predicted promoters. I hypothesize that this initialization process will guide the EM algorithm toward a better solution. To evaluate the effectiveness of this technique, I compare the accuracy of my approach to a variant in which I initialize the parameters randomly. I also apply the technique of adding hidden nodes as described in the previous experiment because this increases the parameter space, and, one would expect, the importance of a good initialization.

The results of this experiment are shown in Table 6.3. My initialization technique improves both the classification error and the average squared error, and the improvement is statistically significant at a p -value less than 0.04 for both measures. Although the technique does not improve the overall probability, the decrease is not statistically significant. Repeating the experiment using random initialization many times on the same fold of cross-validation, I estimate the standard deviation of the classification error at about 0.63% and of the average squared error at about 0.028. Notice that my initialization technique is an improvement over random initialization of at least this much. The standard deviation for log probability is estimated at about 25.53.

6.5 Conclusion

In addressing the problem of inferring models of transcriptional regulation, I have developed an approach that is able to learn to represent the states of regulators (*i.e.* whether a transcription factor is activated or not) as well as their roles (*i.e.* whether a transcription factor acts as an activator or repressor for a given gene). I have empirically evaluated my approach using gene-expression and genomic-sequence data for *E. coli* K-12. My experiments show that both of these aspects of my approach result in models with a high level of predictive accuracy.

This work was originally published in Noto and Craven [2004]. I summarize my contributions as follows:

- I have designed a model that represents the hidden *states* of regulators.
- I have used sequence data to predict the *roles* of a regulators, and I have shown that using these predictions to initialize the network models increases their accuracy.

- I have demonstrated that models that represent regulator states are about as accurate as simpler models that only represent gene expression levels on *E. coli* data, but my models provide potentially useful insights into what regulators are doing after transcription and what cellular conditions affect them.
- I have demonstrated that the inclusion of hypothesized additional regulator states increases the accuracy of the learned models.
- I have used CPD-Trees to learn which cellular conditions are good predictors of regulator states, and I have developed a method for growing and pruning them that works with hidden and probabilistic data.
- I have made the source code publicly available at www.cs.wisc.edu/~noto/pub/software/grn.

Chapter 7

Conclusions

I have presented four major approaches to important and difficult biological model-learning tasks. Although all of these tasks are still very much open problems, I have shown that machine learning approaches, with a focus on designing, or indeed learning, the appropriate *model space* for a task, improve accuracy over current state-of-the-art models.

One thing that these approaches have in common is the use of expressive and comprehensible models. This means that they are able to capture and clearly represent important aspects of biological phenomena, often including those that are not, or cannot, be observed.

7.1 Learning the Logical and Spatial Aspects of *cis*-Regulatory Modules

Chapter 3 describes an approach for learning models of *cis*-regulatory modules that represent logical (*i.e.* AND, OR, NOT) and spatial (*i.e.* order, distance, strand) aspects. This work was originally published in *BMC Bioinformatics* [Noto and Craven, 2006].

The models learn both the *set* of CRM model aspects to represent (*e.g.* distance between binding sites, relative order of binding sites, *etc.*) and then learn the *values* for these aspects (*e.g.* 100 bp, *A* upstream of *B*, *etc.*) in a way that is data-driven. For instance, when a model represents the distance between two binding sites, it chooses that distance to maximize accuracy of the training data.

The empirical evaluation in Chapter 3 shows two things. First, that my learner which induces these expressive models learns more accurate models than other state-of-the-art approaches. This is shown by comparing the significance of the learned models on the same data sets with those of Segal and Sharan [2004]. In addition to this, the experiments show that my models learn more accurate models when compared to a “bag-of-motifs” model, which represents only logical AND and is meant to represent a variety of current CRM-learners.

A second result of the experiments is that each of the six aspects listed above generally improves the accuracy of the learned models. This was shown by doing “lesion tests.” If an aspect

was left out of the model space, the models learned to represent CRMs in yeast became less accurate more often than not.

Contributions

The main contribution of this work is the development of a CRM model which is more expressive than most state-of-the-art models. Another significant contribution is the method I use for avoiding overfitting by selecting the appropriate logical and spatial aspects of the task before training the final model.

Limitations

One limitation of this approach is that it makes decisive decisions. The set of motifs is not learned, but chosen from a list of candidates. The location of each motif occurrence is decided as a pre-processing step based on a threshold. The spatial aspects are represented by hard constraints. All of these limitations, however, are addressed in version two of the approach, described in Chapter 4. A further limitation of this approach related to making hard decisions is the discretization of the training data into positive and negative examples. This is standard for supervised classifiers, and it is relaxed in the regression models described in Chapter 5 and summarized in Section 7.3.

A limitation that is not addressed in Chapter 4 is that the model space increases exponentially with the number of binding sites, which limits its usefulness. Although it is true that CRMs may potentially consist of ten or more binding sites, this number of sites is generally not shared among the dozens of positive examples my algorithms use for training. In my experiments, I limited the search to CRMs with at most three binding sites. This decision was made for practical reasons, considering the number of data sets and cross-validation folds, and I believe that my method could be used now for CRMs with a few more than three binding sites.

7.2 Learning Probabilistic Models of *cis*-Regulatory Modules that Represent Logical and Spatial Aspects

I refer to the approach described in the previous section as SCRM1 (structured CRM version 1). Chapter 4 describes SCRM2, an approach for learning models which addresses some of the limitations of its predecessor. This work was originally presented at the European Conference on Computational Biology (ECCB) [Noto and Craven, 2007].

The main difference between the models used in SCRM1 and SCRM2 is that the latter models employ *probabilistic* representations. Specifically, the spatial aspects are represented by probabilistic preferences instead of hard constraints. These preferences allow the models to adjust the *confidence* of a test sequence that shows a degree of similarity to the CRM model instead of ruling it out completely.

SCRM1's models used position weight matrices (PWMs) to represent motifs, but once the decisions have been made as to where these motifs are most likely to be, the PWM representations are discarded. In contrast, SCRM2's models use PWM representations throughout the learning process, allowing them to learn binding site motifs *de novo*.

Empirical evaluation of the SCRM2 approach shows that it improves the accuracy of learned models compared to SCRM1 and compared to the approach of Segal and Sharan [2004]. Examination of the learned models in a case where the binding site motifs are known shows that two known motifs were recovered by the SCRM2 approach, but not by the same learning strategy that does not use structural aspects in its model space, suggesting that use of spatial aspects may aid the learner in locating the relevant motifs and learning their parameters.

Contributions

One contribution of this research is the use of hidden semi-Markov models as probabilistic models for representing *cis*-regulatory modules. The transition and state-duration parameters of the HMM represent probabilistic spatial preferences concerning the relative order and distance between binding sites, as well as the DNA strand preference of each binding site.

SCRM2 uses HMMs and HMM learning algorithms to learn the parameters of the CRM models. Learning the qualitative (logical) structure of the CRM model is closely related to structure learning in HMMs and other graphical models. However, SCRM2's search operators manipulate logical CRM structures instead of the structure of the HMM directly. In other words, SCRM2 represents a solution to a HMM structure-learning task, where the search is over a more abstract grammar than the one corresponding to the HMM itself.

Another contribution of this research is the design of code optimization for hidden semi-Markov model inference and parameter learning. In the general case, the dynamic programming and EM algorithms for these tasks are $O(L^2)$ and $O(L^3)$ respectively, to process sequences of L sequence characters. However, since position weight matrices have highly-biased preferences for certain subsequences of DNA, it is possible to scan the input once per motif, and only run inference for the very few positions where motifs are actually likely to occur. Furthermore, the general parameter learning EM approaches are $O(L^3)$ because it is necessary to scan the sequence for every beginning and ending position of variable-duration states. In SCRM2's models, however, only the background states have variable-duration, and it is not necessary to update them, as there is only a negligible difference between the background DNA distribution in the presence and in the absence of a CRM. This optimization is well-suited for CRM models, but will work in any domain with high-specificity, low-duration-variability sequence feature representations interspersed with low-specificity, variable-duration regions.

Limitations

The SCRM2 learning algorithms suffer from the same representational limitations as the SCRM1 approach described in the previous section. Specifically, the number of binding sites in the model is limited by the fact that the runtime complexity of the learning algorithm (and inference algorithm, for that matter) increases exponentially with the number of binding sites in the model. Again, I argue that the number of binding sites conserved across multiple promoter regions, which serve as input to the program, is typically quite small, so that limiting the search to three binding sites or so still considers a reasonable model space for the task at hand. This approach, however, will not scale up to allow its use in domains where there are a greater number of motifs of interest.

7.3 Learning Hidden Markov Models for Regression

Chapter 5 describes an approach for using hidden Markov models for regression.

The main idea behind the approach is a parameterized and learnable mapping from sequence features to real-valued responses. The model calculates an aggregate function which maps paths through the model to a vector of the number of times certain “counted” states were visited. The sequence features in my experiments have been position weight matrices (a fixed-length series of individual distributions), but sequence features can be arbitrarily complex, as long their use in a path through the HMM is represented by one of these counted states. The regression model maps vectors of visit counts to real numbers. In my experiments, I use a linear model, which learns a coefficient for each counted state. This is a simple model but again, more complex relationships between motifs can be captured by the topology of the HMM itself. For example, one counted state in Figure 5.6 represents two binding sites in the order m_1, m_2 , and the other counted state represents the order m_2, m_1 , even though m_1 and m_2 could be represented by distant parts of the input sequence.

Much like the HMM structure learning method described in the previous section, the structure search for the HMM regression models manipulates the structure at a higher level of abstraction. Instead of adding and subtracting individual states and transitions, my approach is to add entire new *motifs* to the model. Again, in my experiments, these have been PWMs, but conceptually, they could be any kind of sequence feature represented by a local structure of states and transitions.

Often, input to CRM learning algorithms is a set of promoter sequences from genes that share a common “expression profile” over a series of experiments. Two genes may be considered co-expressed if they are both up-regulated, but the actual expression measurement of the two genes may differ wildly. Classifiers which learn from the promoter sequences of co-expressed genes must “cut-off” expression similarity at some point, and inevitably lose information. This information may be quite important, however, as the degeneracy of binding site motifs is likely to affect transcription rate. A key contribution of my research is framing the CRM learning task as a regression problem, and the development of a method that not only models the real-valued responses for each example, but actually learns sequence features directly from them.

My experiments on simulated data show that the path aggregate learner is able to learn a more accurate model of the relevant sequence motifs and the relative contribution of each than a similar model that first learns features from sequence data alone, and then learns a regression model based on the learned features. Additionally, my experiments using genomic data from yeast show that the path aggregate learner is able to learn more accurate representations of CRMs than comparable models that do not learn sequence features and a regression model simultaneously.

Contributions

The contribution of this research to computer science is the development of an HMM learning algorithm which learns structure and parameters from a real-valued training signal. I have developed an EM learning algorithm for these models based on the Baum-Welch HMM parameter learning algorithm [Rabiner, 1989] which optimizes the joint likelihood of the sequence and the

real-valued responses. I have developed a gradient descent algorithm for these models which optimizes the conditional likelihood of the responses, given the sequence, based on the discriminative learner presented by Krogh [1994]. The gradient descent algorithm is a new development and has not yet been empirically evaluated.

Limitations

One drawback of the path aggregate learner, as presented, is the introduction of additional parameters to represent the regression model. This gives the model more parameter space through which to search, and the learning algorithms are only guaranteed to find a local optimum.

Another limitation is the issue of scalability. Much like the models described in Chapter 4, the algorithmic complexity increases exponentially with the number of counted states. Unlike the models described in Chapter 4, this is true even without parameterizing the relative order and distance of binding site motifs. The number of possible values for the hidden vector of visit counts makes the algorithm intractable for more than about ten counted states. In response to this limitation, in Chapter 5, I present a method for calculating and using the *expected* number of visits to each state instead of exact calculations. Unfortunately, this involves further approximation, and makes the learning task even more difficult than it already is.

7.4 Learning Models of Gene-Regulatory Networks that Represent States and Roles

Chapter 6 describes a Bayesian network learning approach for gene-regulatory networks. This work was originally published at the RECOMB workshop on Regulatory Genomics [Noto and Craven, 2004].

These regulatory networks represent the expression of genes, both regulators and regulatees, and learns the relationships between them. To do this, my networks explicitly represent the underlying *states* of regulators. I refer to the effect that these regulators have on regulatees (depending on their state) as the *roles* of the regulators. The value of these roles is initialized using an analysis of the promoter sequence and predictions of the binding site locations of regulators. Regulators that bind well upstream of the predicted start of transcription are predicted to be activators. Regulators that bind further downstream are predicted to be repressors and the CPDs for the regulatee genes are initialized accordingly. These predicted roles may be erroneous, and corrected during the learning process.

My learning algorithm has the ability to identify clusters of genes for which the expression states cannot be explained by the regulators present in the data. These regulators are hypothesized to be regulated by an additional regulator, whose state is then added to the model.

Empirical evaluation shows that my network models are about as accurate as baseline models which are less expressive. My models, in contrast to these baselines, are able to learn which cellular conditions affect regulator states, and how these states, instead of the expression of regulators, affect gene expression. My experiments show that using sequence predictions to initialize network

parameters results in more accurate network models than initializing the parameters randomly, and that the addition of regulator states tends to improve the accuracy of the learned network models.

Contributions

One contribution of this work is the inclusion of latent variables that represent the underlying states of regulators (*i.e.* activated or inactivated) with a hidden layer of network nodes, and the roles of regulators, represented by their parameters in the CPTs of the genes they regulate.

Another contribution is the development of a method to grow conditional probability distribution (CPD) trees from probabilistic data. These trees are grown for the regulator state variables, which are hidden and represented in data as a probability distribution over state values, and the expression states of the parent regulators are represented as a probability distribution over expression states.

Limitations

Although the addition of regulator states to the model increases test set accuracy, the usefulness of this approach is limited by the fact that these putative regulators are hypothetical and at least unidentified. It may be possible to identify these regulators by aligning their predicted activity with that of identified regulators in the model or with the expression of regulators not yet included in the network, or by considering other sources of data, *e.g.* a common sequence motif in the promoter region of regulatees.

7.5 Future Directions

There are a few interesting directions for the work presented in this thesis that I can foresee. For example, I would be interested in extending the framework for using HMMs for regression to other graphical models, such as conditional random fields or stochastic context-free grammars (SCFGs). The basic idea is to learn a mapping from sequence features to real-valued responses. For HMMs, these sequence features can be the number of visits to key states in a path through the model, but for SCFGs, they might be the frequency with which certain production rules are employed.

Another idea is to extend the regression CRM models described in Chapter 5 to explain multiple experiments. These CRM models currently take promoter sequences and gene expression input of the form: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$, where the \mathbf{x} 's are the promoter sequences, and the y 's are the expression measurements. The task is to learn the arrangement of binding sites that explain the expression measurements. However, there may be motifs present in the promoter sequences that do not affect the expression in some cases, but do affect the expression in others. There may be an advantage to sharing information about motif predictions across multiple experiments. That is, given input form $\langle \mathbf{x}_1, y_{1,1}, y_{1,2}, \dots, y_{1,M} \rangle, \langle \mathbf{x}_1, y_{2,1}, y_{2,2}, \dots, y_{2,M} \rangle, \dots, \langle \mathbf{x}_N, y_{N,1}, y_{N,2}, \dots, y_{N,M} \rangle$, where $y_{g,e}$ is the expression of gene g in experiment e , the task is to learn a model which explains why genes are differentially expressed in different situations. This idea is closely related to transfer learning [Thrun, 1996] and multitask learning [Caruana, 1997]. These models may benefit from the inclusion of cellular condition data, like the models presented in Chapter 6.

7.6 Final Thoughts

With the increasing availability of genomic sequence and expression data, computer scientists have turned their attention to automatic methods for learning models that explain these data. Machine learning methods that use probabilistic and statistical models are attractive because they can represent uncertainty about unobserved events. However, general-purpose methods often fail to learn comprehensible models or require more data than are available in order to generalize.

In contrast, my models are designed with a built-in task-dependent bias, putting the search for model structure on a higher level of abstraction than the underlying probabilistic graphical model. I believe that the comprehensible models and expressive model space that I have used in this thesis represent a significant and measurable contribution to computational biology in particular and to computer science in general.

Bibliography

- S. Aerts, P. Van Loo, G. Thijs, Y. Moreau, and B. De Moor. Computational detection of cis-regulatory modules. *Bioinformatics*, 19(2):5–14, 2003.
- A. Agresti. A survey of exact inference for contingency tables. *Statistical Science*, 7(1):131–177, 1992.
- T. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36. AAAI Press, 1994.
- T. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21:51–83, 1995.
- H. Bannai, S. Inenaga, A. Shinohara, M. Takeda, and S. Miyano. Efficiently finding regulatory elements using correlation with gene expression. *J Bioinform Comput Biol.*, 2(2):273–288, 2004.
- L. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.
- M. Beer and S. Tavazoie. Predicting gene expression from sequence. *Cell*, 117:185–198, 2004.
- Y. Bengio and P. Frasconi. An input-output HMM architecture. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, Cambridge, MA, 1995.
- M. Blanchette and M. Tompa. Discovery of regulatory elements by a computational method for phylogenetic footprinting. *Genome Research*, 12(5):739–748, 2002.
- C. Blaschke, M. Andrade, C. Ouzounis, and A. Valencia. Automatic extraction of biological information from scientific text: Protein-protein interactions. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 60–67. AAAI Press, 1999.
- F. Blattner, G. Plunkett III, C. Bloch, N. Perna, V. Burland, M. Riley, J. Collado-Vides, J. Glasner, C. Rode, G. Mayhew, J. Gregor, N. Davis, H. Kirkpatrick, M. Goeden, D. Rose, B. Mau, and Y. Shao. The complete genome sequence of *Escherichia coli* k-12. *Science*, 277(5331):1453–1462, 1997.

- J. Bockhorst and M. Craven. Refining the structure of a stochastic context-free grammar. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1315–1320. Morgan Kaufmann, 2001.
- J. Bockhorst, Y. Qiu, J. Glasner, M. Liu, F. Blattner, and M. Craven. Predicting bacterial transcription units using sequence and expression data. *Bioinformatics*, 19(Suppl. 1):i34–i43, 2003.
- C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pages 115–123, 1996.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.
- A. Bryson and H. Yu-Chi. *Applied Optimal Control*. Blaisdell, New York, 1969.
- R. Bunescu, R. Ge, R. Kate, E. Marcotte, R. Mooney, A. Ramani, and Y. Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–55, 2005.
- C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.
- R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.
- W. Cohen. Text categorization and relational learning. In Armand Prieditis and Stuart J. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 124–132. Morgan Kaufmann Publishers, San Francisco, US, 1995.
- T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.
- F. Crick. On protein synthesis. *Symp. Soc. Exp. Biol. XII*, pages 139–163, 1958.
- F. Crick. Central dogma of molecular biology. *Nature*, 227:561–563, 1970.
- G. Crooks, G. Hon, J. Chandonia, and S. Brenner. WebLogo: A sequence logo generator. *Genome Research*, 14:1188–1190, 2004.
- B. D’Ambrosio. Inference in Bayesian networks. *AI Magazine*, 20(2):21–36, 1999.
- S. Darnell, D. Page, and J. Mitchell. Automated decision-tree approach to predicting protein-protein interaction hot spots. *Proteins: Structure, Function, and Bioinformatics*, 2007.
- J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *ICML ’06: Proceedings of the 23rd international conference on Machine learning*, pages 233–240, New York, NY, USA, 2006. ACM Press.

- J. Davis, E. Burnside, I. Dutra, D. Page, and V. Santos Costa. An integrated approach to learning Bayesian networks of rules. In *Proceedings of the 16th European Conference on Machine Learning*, pages 84–95, 2005.
- J. Davis, V. Santos Costa, S. Ray, and D. Page. An integrated approach to feature construction and model building for drug activity prediction. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.
- H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.
- A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38, 1977.
- P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall International, 1982.
- R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK, 1998.
- S. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22:2079–2088, 1994.
- E. Eskin and P. Pevzner. Finding composite regulatory patterns in DNA sequences. *Bioinformatics*, 18(Suppl. 1):1367–4803, 2002.
- O. Etzioni, R. Tuchinda, C. Knoblock, and A. Yates. To buy or not to buy: Mining airfare data to minimize ticket purchase price. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 119–128. ACM Press, 2003.
- A. Favorov, M. Gelfand, A. Gerasimova, D. Ravcheev, A. Mironov, and V. Makeev. A Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length. *Bioinformatics*, 21(10):2240–2245, May 2005.
- M. Franzini. Speech recognition with back propagation. In *IEEE Ninth Annual Conference of the Engineering in Medicine and Biology Society*, pages 1702–1703, 1987.
- D. Freitag and A. McCallum. Information extraction with HMM structures learned by stochastic optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, TX, 2000. AAAI Press.
- N. Friedman and D. Koller. Being bayesian about network structure. In *Proceedings of the Sixteenth International Conference on Uncertainty in Artificial Intelligence*, 2000.
- N. Friedman and D. Koller. Being bayesian about network structure: A bayesian approach to structure discovery in bayesian networks. *Machine Learning*, 9:309–347, 2003.

- N. Friedman, M. Linial, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.
- J. Friedman. A recursive partitioning decision rule for non-parametric classification. *IEEE Transactions on Computers*, pages 404–408, 1977.
- M. Frith, U. Hansen, J. Spouge, and Z. Weng. Finding functional sequence elements by multiple local alignment. *Nucleic Acids Research*, 32(1):189–200, 2004.
- U. Galassi, A. Giordana, and L. Saitta. Incremental construction of structured hidden markov models. In M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 798–803, 2007.
- A. Gasch, P.T. Spellman, C. Kao, O. Carmel-Harel, M. Eisen, G. Storz, D. Botstein, and P. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11(12):4241–57, 2000.
- S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.
- L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2006.
- J.D. Glasner, P. Liss, G. Plunkett III, A. Darling, T. Prasad, M. Rusch, A. Byrnes, M. Gilson, B. Biehl, F.R. Blattner, and N.T. Perna. ASAP, a systematic annotation package for community analysis of genomes. *Nucleic Acids Research*, 31(1):147–151, 2003.
- L. Grate, M. Herbster, R. Hughey, D. Haussler, I. Mian, and H. Noller. RNA modeling using Gibbs sampling and stochastic context free grammars. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, 1994.
- G-D. Guo and C. Dyer. Face cyclographs for recognition. In *Proceedings of the Eighth International Conference on Computer Vision, Pattern Recognition and Image Processing*, 2007.
- M. Gupta and J. Liu. De novo cis-regulatory module elicitation for eukaryotic genomes. *Proceedings of the National Academy of Sciences*, 102(20):7079–7084, 2005.
- A. Hartemink, D. Gifford, T. Jaakkola, and R. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Proceedings of the Fifth Pacific Symposium on Biocomputing*, pages 422–433. World Scientific Press, 2001.
- A. Hartemink, D. Gifford, T. Jaakkola, and R. Young. Combining location and expression data for principled discovery of genetic regulatory networks. In *Proceedings of the Fifth Pacific Symposium on Biocomputing*, pages 437–449. World Scientific Press, 2002.
- T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

- M. Howard and E. Davidson. cis-regulatory control circuits in development. *Developmental Biology*, 271:109–118, 2004.
- J. Hughes, P. Estep, S. Tavazoie, and G. Church. Computational identification of cis-regulatory elements associated with functionally coherent groups of genes in *Saccharomyces cerevisiae*. *Journal of Molecular Biology*, 296(5):1205–1214, 2000.
- R.A. Irizarry, B. Hobbs, F. Collin, Y.D. Beazer-Barclay, K.J. Antonellis, U. Scherf, and T.P. Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4(2):249–264, 2002.
- V. Iyer, C. Horak, C. Scafe, D. Botstein, M. Snyder, and P. Brown. Genomic binding sites of the yeast cell-cycle transcription factors SBF and MBF. *Nature*, 409:533–538, 2001.
- G. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.
- B. Juang and L. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.
- T. Kammeyer and R. Belew. Stochastic context-free grammar induction with a genetic algorithm using local search. In Richard K. Belew and Michael Vose, editors, *Foundations of Genetic Algorithms IV*. Morgan Kaufmann, 3–5 1996.
- A. Karali and I. Bratko. First order regression. *Machine Learning*, 26(2-3):147–176, 1997.
- D. Karolchik, A. Hinrichs, T. Furey, K. Roskin, C. Sugnet, D. Haussler, and W. Kent. The UCSC table browser data retrieval tool. *Nucleic Acids Research*, 32(1):D493–D496, 2004.
- P. Karp, M. Riley, M. Saier, I. Paulsen, J. Collado-Vides, S. Paley, A. Pellegrini-Toole, C. Bonavides, and S. Gama-Castro. The EcoCyc database. *Nucleic Acids Research*, 30:56–58, 2002.
- S. Keleş, M. van der Lann, and C. Vulpe. Regulatory motif finding by logic regression. *Bioinformatics*, 20(16):2799–2811, 2004.
- R. King, K. Whelan, F. Jones, P. Reiser, C. Bryant, S. Muggleton, D. Kell, and S. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427:247–252, 2004.
- H. Kitano, editor. *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Computer Science*. Springer, 1998.
- S. Kramer. Structural regression trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 812–819. AAAI/MIT Press, 1996.

- A. Krogh, M. Brown, I. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications in protein modeling. *Journal of Molecular Biology*, 238:54–61, 1994.
- A. Krogh. Hidden Markov models for labeled sequences. In *Proceedings of the Twelfth IAPR International Conference on Pattern Recognition.*, pages 140–44. IEEE Computer Society Press, October 1994.
- M. Krummenacker, S. Paley, L. Mueller, T. Yan, and P. Karp. Querying and computing with BioCyc databases. *Bioinformatics*, 21:3454–3455, 2005.
- J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann, 2001.
- N. Landwehr, K. Kersting, and L. De Raedt. nFOIL: Integrating naïve bayes and FOIL. In M. Veloso and S. Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pages 795–800, 2005.
- K. Lari and S. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.
- C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.
- T. Lee, N. Rinaldi, F. Robert, D. Odom, Z. Bar-Joseph, G. Gerber, N. Hannett, C. Harbison, C. Thompson, I. Simon, J. Zeitlinger, E. Jennings, H. Murray, D. Gordon, B. Ren, J. Wyrick, J. Tagne, T. Volkert, E. Fraenkel, D. Gifford, and R. Young. Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science*, 298:799–804, 2002.
- C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for SVM protein classification. *Pacific Symposium on Biocomputing*, 7:566–575, 2002.
- N. Li and M. Tompa. Analysis of computational approaches for motif discovery. *Algorithms for Molecular Biology*, 1:8, 2006.
- J. Lieb, X. Liu, D. Botstein, and P. Brown. Promoter-specific binding of Rap1 revealed by genome-wide maps of protein-DNA association. *Nature Genetics*, 28:327–334, 2001.
- X Liu, D. Brutlag, and J. Liu. Bioprospector: Discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. In *Proceedings for the Pacific Symposium for Biocomputing*, pages 127–138, 2001.

- K. Macisaac, D. Gordon, L. Nekludova, D. Odom, J. Schreiber, D. Gifford, R. Young, and E. Fraenkel. A hypothesis-based approach for identifying the binding specificity of regulatory proteins from chromatin immunoprecipitation data. *Bioinformatics*, 22(4):423–429, 2006.
- S. Mahony, D. Hendrix, A. Golden, T. Smith, and D. Rokhasr. Transcription factor binding site identification using the self-organizing map. *Bioinformatics*, pages 1807–1814, 2005.
- O. Mangasarian, W. Nick Street, and W. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.
- C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge MA, 1999.
- K. Marchal, G. Thijs, S. De Keersmaecker, P. Monsieurs, B. De Moor, and J. Vanderleyden. Genome-specific higher-order background models to improve motif detection. *Trends in Microbiology*, 11, 2003.
- L. McCue, W. Thompson, C. Carmack, and C. Lawrence. Factors influencing the identification of transcription factor binding sites by cross-species comparison. *Genome Research*, 12:1523–1532, 2002.
- M. Mehta, J. Rissanen, and R. Agrawal. MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 216–221. AAAI Press, 1995.
- A. Miller. *Subset Selection in Regression*. Chapman and Hall, 1990.
- T. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- R. Mooney and G. DeJong. Learning schemata for natural language processing. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 681–687, August 1985.
- S. Muggleton and L. DeRaedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19:629–679, 1994.
- S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
- P. Murphy and M. Pazzani. ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees. In *Proceedings of the Eighth International Machine Learning Workshop*, pages 183–187. Morgan Kaufmann, 1991.
- S. Murthy and S. Salzberg. Lookahead and pathology in decision tree induction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1025–1031. Morgan Kaufmann, 1995.

- S. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.
- I. Nachman, A. Regev, and N. Friedman. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics*, 20(Suppl. 1):i248–i256, 2004.
- W. Noble, S. Kuehn, R. Thurman, M. Yu, and J. Stamatoyannopoulos. Predicting the *in vivo* signature of human gene regulatory sequences. *Bioinformatics*, 21(1):i338–i343, 2005.
- K. Noto and M. Craven. Learning regulatory network models that represent regulator states and roles. In *Regulatory Genomics: RECOMB 2004 International Workshop*. Springer-Verlag, New York, NY, 2004.
- K. Noto and M. Craven. A specialized learner for inferring structured cis-regulatory modules. *BMC Bioinformatics*, 7:528, 2006.
- K. Noto and M. Craven. Learning probabilistic models of cis-regulatory modules that represent logical and spatial aspects. *Bioinformatics*, 23(2):e156–e162, 2007.
- I. Ong, J. Glasner, and D. Page. Modeling regulatory pathways in *E. coli* from time series expression profiles. *Bioinformatics*, 18(Suppl. 1):S241–S248, 2002.
- G. Pavesi, P. Mereghetti, G. Mauri, and G. Pesole. Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Research*, 32:W199–W203, 2004.
- G. Pavesi, F. Zambelli, and G. Pesole. WeederH: An algorithm for finding conserved regulatory motifs and regions in homologous sequences. *BMC Bioinformatics*, 8(46), 2007.
- J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.
- D. Pe’er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17(Suppl. 1):215S–224S, 2001.
- D. Pe’er, A. Tanay, and A. Regev. MinReg: A scalable algorithm for learning parsimonious regulatory networks in yeast and mammals. *Journal of Machine Learning Research*, 7:167–189, 2006.
- A. Philippakis, F. He, and M. Bulyk. Modulefinder: A tool for computational discovery of cis-regulatory modules. In Russ B. Altman, Tiffany A. Jung, Teri E. Klein, A. Keith Dunker, and Lawrence Hunter, editors, *Proceedings for the Pacific Symposium for Biocomputing*, pages 519–530. World Scientific, 2005.
- D. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1, pages 305–313. Morgan Kaufmann, 1989.

- D. Pomerleau. Knowledge-based training of artificial neural networks for autonomous robot driving. In J. Connell and S. Mahadevan, editors, *Robot Learning*, pages 19–43. Boston: Kluwer Academic Publishers, 1993.
- W. Quine. *Word and Object*. Cambridge, MA: MIT Press, 1960.
- J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- J. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI/MIT Press, 1996.
- L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- N. Rajewsky, M. Vergassola, U. Gaul, and E. Siggia. Computational detection of genomic cis regulatory modules. *BMC Bioinformatics*, 2002.
- B. Ren, F. Robert, J.J. Wyrick, O. Aparicio, E.G. Jennings, I. Simon, J. Zeitlinger, J. Schreiber, N. Nannett, E. Kanin, T. Volkert, C. Wilson, S. Bell, and R. Young. Genome-wide location and function of DNA binding proteins. *Science*, 290:2306–2309, 2000.
- Y. Sakakibara, M. Brown, R. Hughey, I. Mian, K. Sjölander, R. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22:5112–5120, 1994.
- M. Schena, D. Shalon, R. Davis, and P. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.
- E. Segal and R. Sharan. A discriminative model for identifying spatial cis-regulatory modules. In *Proceedings of the Eighth Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 141–149. ACM Press, 2004.
- E. Segal, M. Shapira, A. Regev, D. Pe’er, D. Botstein, D. Koller, and N. Friedman. Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, 2003.
- E. Segal, R. Yelensky, and D. Koller. Genome-wide discovery of transcriptional modules from DNA sequence and gene expression. *Bioinformatics*, 19(Suppl. 1):i273–i282, 2003.
- E. Segal, Y. Fondufe-Mittendorf, L. Chen, A. Thåström, Y. Field, I. Moore, J.-P. Wang, and J. Widom. A genomic code for nucleosome positioning. *Nature*, 442:772–778, 2006.
- K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, 1999.

- R. Sharan, I. Ovcharenko, A. Ben-Hur, and R. Karp. Creme: a framework for identifying cis-regulatory modules in human-mouse conserved segments. *Bioinformatics*, 19(Suppl. 1):i283–i291, 2003.
- D. Shepard. A two-dimensional interpolation function for irregularly spaced data. In *Proceedings of the Twenty-Third National Conference of the ACM*, pages 517–523, 1968.
- R. Siddharthan, E. Siggia, and E. van Nimwegen. PhyloGibbs: A Gibbs sampling motif finder that incorporates phylogeny. *PLoS Computational Biology*, 1(7), 2005.
- S. Sinha and M. Tompa. YMF: a program for discovery of novel transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic Acids Research*, 31:3586–3588, 2003.
- S. Sinha, E. van Nimegen, and E. Siggia. A probabilistic method to detect regulatory modules. *Bioinformatics*, 19(1):292–301, 2003.
- S. Sinha, M. Blanchette, and M. Tompa. PhyME: A probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics*, 5(170), 2004.
- A. Stolcke and S. Omohundro. Inducing probabilistic grammars by Bayesian model merging. In *Proceedings of the Second International Colloquium on Grammatical Inference and Applications (ICGI)*, pages 106–118, London, UK, 1994. Springer-Verlag.
- G. Stormo and G. Hartzell. Identifying protein-binding sites from unaligned DNA fragments. *Proceedings of the National Academy of Sciences*, 86(4):1183–1187, 1989.
- C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.
- Y. Tamada, S. Kim, H. Bannai, S. Imoto, K. Tashiro, S. Kuhara, and S. Miyano. Estimating gene networks from gene expression data by combining Bayesian network model with promoter element detection. *Bioinformatics*, 19(Suppl. 2):ii227–ii236, 2003.
- G. Tesauro and T. Sejnowski. A parallel network that learns to play backgammon. *Artificial Intelligence*, 39(3):357–390, 1989.
- The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
- G. Thijs, M. Lescot, K. Marchal, S. Rombauts, B. De Moor, P. Rouze, and Y. Moreau. A higher order background model improves the detection of regulatory elements by Gibbs sampling. *Bioinformatics*, 17(12):1113–1122, 2001.
- G. Thijs, K. Marchal, M. Lescot, S. Rombauts, B. De Moor, P. Rouze, and Y. Moreau. A Gibbs sampling method to detect over-represented motifs in upstream regions of coexpressed genes. *Journal of Computational Biology*, 9(2):447–464, 2002.

- W. Thompson, L. Newberg, S. Conlan, L. McCue, and C. Lawrence. The Gibbs centroid sampler. *Nucleic Acid Research*, 35:W232–W237, July 2007.
- S. Thrun. Is learning the n -th thing any easier than learning the first? In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 640–646. The MIT Press, 1996.
- A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.
- E. Wingender, X. Chen, E. Fricke, R. Geffers, R. Hehl, I. Liebich, M. Krull, V. Matys, H. Michael, R. Ohnhäuser, M. Prüß, F. Schacherer, S. Thiele, and S. Urbach. The TRANSFAC system on gene expression regulation. *Nucleic Acids Research*, 29:281–283, 2001.
- K.-J. Won, A. Prügel-Bennett, and A. Krogh. Evolving the structure of hidden markov models. *IEEE Transactions on Evolutionary Computing*, 10(1):39–49, 2006.
- C. Workman and G. Stormo. ANN-Spec: A method for discovering transcription factor binding sites with improved specificity. In *Proceedings for the Pacific Symposium for Biocomputing*, pages 188–194, 2000.
- E.P. Xing, M.I. Jordan, and R.M. Karp. Feature selection for high-dimensional genomic microarray data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.
- C. Yoo and G. Cooper. Discovery of gene-regulation pathways using local causal search. In *Proceedings of the Annual Fall Symposium of the American Medical Informatics Association*, pages 914–918, 2002.
- C. Yoo, V. Thorsson, and G. Cooper. Discovery of causal relationships in a gene-regulation pathway from a mixture of experimental and observational DNA microarray data. In *Proceedings of the Fifth Pacific Symposium on Biocomputing*, pages 498–509. World Scientific Press, 2002.
- C.-H. Yuh, H. Bolouri, and E. Davidson. *cis*-regulatory logic in the *endo16* gene: Switching from a specification to a differentiation mode of control. *Development*, 128:617–629, 2001.
- Q. Zhou and W. Wong. CisModule: *De novo* discovery of *cis*-regulatory modules by hierarchical mixture modeling. *Proceedings of the National Academy of Sciences*, 101(33):12114–12119, 2004.

Appendix A: SCRM1 Results on Lee etal Data Sets

This appendix shows the complete results testing the SCRM1 algorithm on the data sets described in Table 3.3. The statistics calculated are: True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Precision (P), Recall (R), F1 Score, and statistical p -value. For convenience, the equations for precision, recall and F1 from Chapter 2 are reprinted here.

$$P = \frac{TP}{TP + FP} \quad [\text{A.1}]$$

$$R = \frac{TP}{TP + FN} \quad [\text{A.2}]$$

$$F1 = \frac{2 \times P \times R}{P + R} \quad [\text{A.3}]$$

Table A.1 Results of running SCRM1 on the Lee *et al.* data sets described in Table 3.3. Significant CRMs (p -value < 0.01) are indicated with bold text. Statistics shown are: True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Precision (Equation A.1), Recall (Equation A.2), F1 Score (Equation A.3), and statistical p -value.

Data set	TP	FP	TN	FN	P	R	F1	p -value
GAT3, RGM1	5	22	78	10	0.185	0.333	0.238	0.253
GAL4, YAP5	7	16	84	9	0.304	0.438	0.359	0.0169
GAT3, PDR1	10	9	91	7	0.526	0.588	0.556	1.138e-05
CIN5, NRG1	10	18	82	8	0.357	0.556	0.435	1.53e-03
RGM1, YAP5	7	13	87	11	0.35	0.389	0.368	0.0137
NDD1, SWI4	15	23	77	7	0.395	0.682	0.5	8.51e-05
SKN7, SWI4	12	38	62	10	0.24	0.545	0.333	0.118
PDR1, YAP5	13	36	64	10	0.265	0.565	0.361	0.0585
FKH2, SWI4	11	20	80	13	0.355	0.458	0.4	0.0113
PHD1, YAP6	14	18	82	10	0.438	0.583	0.5	1.52-e04
FHL1, YAP5	15	19	81	10	0.441	0.6	0.508	1.07-e04
FKH2, MCM1	20	33	67	5	0.377	0.8	0.513	2.45-e05
MBP1, NDD1	13	24	76	12	0.351	0.52	0.419	7.52-e03
ACE2, SWI5	17	29	71	9	0.37	0.654	0.472	7.97-e04
FKH2, MBP1	19	31	69	8	0.38	0.704	0.494	2.60-e04
MCM1, NDD1	24	36	64	4	0.4	0.857	0.545	2.37-e06
RAP1, YAP5	15	7	93	14	0.682	0.517	0.588	3.68-e07
NRG1, YAP6	15	22	78	15	0.405	0.5	0.448	3.72-e03
GAT3, YAP5	25	11	89	14	0.694	0.641	0.667	8.75-e10
CIN5, YAP6	26	41	59	14	0.388	0.65	0.486	8.47-e03
MBP1, SWI4	27	33	67	13	0.45	0.675	0.54	2.00-e04
SWI4, SWI6	20	45	55	23	0.308	0.465	0.37	0.506
MBP1, SWI6	39	38	62	5	0.506	0.886	0.645	6.34-e09
FKH2, NDD1	35	83	17	15	0.297	0.7	0.417	0.978
FHL1, RAP1	89	36	64	25	0.712	0.781	0.745	3.33-e10

Table A.2 Results of running SCRM1 on the Gasch *et al.* data sets described in Table 3.3. Significant CRMs (p -value < 0.01) are indicated with bold text. Statistics shown are: True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Precision (Equation A.1), Recall (Equation A.2), F1 Score (Equation A.3), and statistical p -value.

Data set	TP	FP	TN	FN	P	R	F1	p -value
iESR	89	453	4497	181	0.164	0.33	0.219	~ 0
rESR_PACcluster	173	260	4690	255	0.4	0.404	0.402	~ 0
rESR_RPcluster	50	84	4866	71	0.373	0.413	0.392	~ 0

Table A.3 Results of running SCRM1 on the Sinha *et al.* data sets described in Table 3.3. Significant CRMs (p -value < 0.01) are indicated with bold text. Statistics shown are: True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), Precision (Equation A.1), Recall (Equation A.2), F1 Score (Equation A.3), and statistical p -value.

Data set	TP	FP	TN	FN	P	R	F1	p -value
Yeast	6	2	7	0	0.75	1	0.857	5.59-e03
Fly	3	3	97	5	0.5	0.375	0.429	4.92-e03
Fly, known PWMs	7	10	90	1	0.412	0.875	0.560	5.10e-06

Appendix B: Path Aggregate Learning: Results on Simulated Data Sets

This appendix shows the results of running my path aggregate learner and baseline approaches described in Chapter 5 on the simulated data sets also described in that chapter. Note that the number of replicate experiments may differ for different values of the mutation and decoy rates (affecting the standard deviation) and that the results shown in Figure 5.4 are based on a subsets of the experiments described here.

Table B.1 Results of my Path Aggregate learning algorithm on the simulated data described in Chapter 5.

		Mutation Rate			
		0	1	3	5
Decoy Rate	0	0.97 ± 1.08	1.57 ± 2.47	2.20 ± 3.76	2.27 ± 3.95
	1	1.33 ± 1.76	1.48 ± 2.57	2.12 ± 3.62	2.43 ± 4.35
	2	1.40 ± 1.88	1.72 ± 2.60	2.31 ± 3.71	2.61 ± 4.57
	3	1.46 ± 2.08	1.67 ± 2.23	2.32 ± 3.72	2.51 ± 4.32
	4	1.35 ± 1.83	1.48 ± 2.13	2.32 ± 3.88	2.35 ± 4.10
	5	1.39 ± 1.89	1.68 ± 2.60	2.25 ± 3.68	2.40 ± 4.23
	6	1.37 ± 1.92	1.72 ± 2.79	2.27 ± 3.81	2.13 ± 3.58
	7	1.38 ± 1.86	1.57 ± 2.57	2.36 ± 3.94	2.65 ± 4.47
	8	1.37 ± 1.85	2.16 ± 3.48	2.11 ± 3.43	2.20 ± 3.84
	9	1.41 ± 1.86	1.48 ± 2.34	2.31 ± 3.90	2.44 ± 4.37

Table B.2 Results of the two-phase baseline described in Chapter 5 on the simulated data in Table B.1.

		Mutation Rate			
		0	1	3	5
Decoy Rate	0	1.08 ± 1.35	2.62 ± 3.79	2.70 ± 3.67	2.92 ± 3.48
	1	2.16 ± 3.06	2.49 ± 3.86	2.95 ± 3.55	2.60 ± 4.28
	2	2.62 ± 3.34	1.69 ± 2.46	2.97 ± 3.64	3.16 ± 3.91
	3	2.67 ± 3.45	3.39 ± 3.62	2.88 ± 3.71	3.33 ± 3.44
	4	2.74 ± 3.53	2.94 ± 3.06	2.82 ± 3.71	3.01 ± 3.55
	5	2.73 ± 3.46	2.79 ± 3.61	2.95 ± 3.64	2.99 ± 3.71
	6	2.60 ± 3.46	2.99 ± 3.87	2.81 ± 3.77	2.67 ± 3.00
	7	2.85 ± 3.72	3.19 ± 3.90	2.95 ± 3.79	3.32 ± 3.89
	8	2.79 ± 3.66	2.84 ± 3.69	2.78 ± 3.57	2.75 ± 3.41
	9	2.82 ± 3.64	3.04 ± 3.29	2.92 ± 3.77	2.75 ± 3.85

Table B.3 Results of the training-set-average baseline described in Chapter 5 on the simulated data in Table B.1.

		Mutation Rate			
		0	1	3	5
Decoy Rate	0	2.91 ± 3.77	2.72 ± 3.73	2.98 ± 3.72	2.93 ± 3.49
	1	3.03 ± 3.77	2.59 ± 3.77	2.91 ± 3.63	2.94 ± 3.97
	2	3.01 ± 3.70	3.19 ± 3.56	3.00 ± 3.67	3.31 ± 3.80
	3	2.97 ± 3.66	3.38 ± 3.68	3.03 ± 3.68	3.37 ± 3.46
	4	3.00 ± 3.75	2.66 ± 3.27	2.92 ± 3.68	2.94 ± 3.60
	5	2.92 ± 3.62	2.88 ± 3.68	2.98 ± 3.63	3.01 ± 3.71
	6	2.85 ± 3.60	3.11 ± 3.90	2.85 ± 3.79	2.36 ± 3.22
	7	3.00 ± 3.78	3.07 ± 4.00	2.98 ± 3.79	3.49 ± 3.73
	8	2.88 ± 3.73	2.89 ± 3.75	2.82 ± 3.55	2.73 ± 3.42
	9	2.87 ± 3.66	2.87 ± 3.42	2.97 ± 3.74	3.03 ± 3.63