UNIVERSITY OF CALIFORNIA, SAN DIEGO

Discovering motifs in DNA and protein sequences: The approximate common substring problem

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Timothy Lawrence Bailey

Committee in charge:

Professor Charles P. Elkan, Chair
Professor Paul R. Kube
Professor Christos H. Papadimitriou
Professor Edward A. Bender
Professor Douglas W. Smith
Professor Michael Gribskov

1995

The dissertation of Timothy Lawrence Bailey is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

1995

To my parents

TABLE OF CONTENTS

ACKNOWLEDGMENTS

It was my good fortune to have Charles Elkan as my advisor while at UCSD. He taught me how to be a fruitful researcher, write good papers and, above all, have a good attitude. Charles was always there when I needed someone to bounce ideas off of and often steered me away from dead-end paths. I only wish I had listened to his advice more often!

I would like to thank the rest of my committee for providing valuable feedback and guidance for my work. Ed Bender and Doug Smith were valuable sources of inspiration especially in the early phases. Paul Kube made statistics seem easy (finally) and helped me understand EM. Christos Papadimitriou taught me how to think about algorithms and has long been the example I bear in mind whenever I set out to explain my ideas to others. Michael Gribskov provided many key suggestions which greatly improved my research and was my primary sanity check for things biological.

I greatly appreciate my collaboration with biologists Michael Baker and Julia Bailey-Serres. Their input helped me understand the needs and point of view of the biologist in the trenches. The data they provided led directly to many improvements in my research. Working with them was always a pleasure.

I am grateful to all the other students at UCSD with whom I was fortunate enough to do research. The work I did together Bill Hart, Bill Grundy, Karan Bhatia and Paul Tucker was among the most rewarding I did at UCSD.

I owe a debt of gratitude to many colleagues around the country. In particular, I would like to thank Kimmen Sjölander for kindly providing me with the Dirichlet mixture priors for proteins and Jorja Henikoff for answering all my questions about BLOCKMAKER.

My friends in the AI lab at UCSD are the best people I know. Without their help getting things done and letting off steam I would never have made it. Thanks to Filippo Menzer, Karin Högstedt, Markus Jakobsson, Alvaro Monge,

Curtis Padgett, Dan Clouse and Tom Kammeyer.

To my family, mom, dad, Maren, Teresa, Julia, Nick, Veronica and Alexander—thanks for being there. You guys are my inspiration. Now we have three doctors in the house.

Finally, thanks to Elissa Mendoza for all the moral support and for creating the artwork which embellishes this dissertation.

VITA

March 11, 1955	Born, Berkeley, California
1977	B.S., Stanford University
1991	M.S., University of California, San Diego
1995	Doctor of Philosophy University of California, San Diego

PUBLICATIONS

Timothy L. Bailey and Charles Elkan, Unsupervised Learning of Multiple Motifs in Biopolymers using EM, *Machine Learning*, in press.

Timothy L. Bailey and Charles Elkan, "The value of prior knowledge in Discovering Motifs with MEME", *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, (in press), July, 1995.

Timothy L. Bailey and Charles Elkan, Chapter 18, "Cross-validation and Modal Theories", *Computational Learning Theory and Natural Learning Systems*, (345-359), Vol. 3, MIT Press, 1995.

Timothy L. Bailey and Charles Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers", UCSD Technical Report, CS94-351, March, 1994, University of California at San Diego.

Timothy L. Bailey and Charles Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers", *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, (28-36), AAAI Press, 1994.

Timothy L. Bailey and Charles Elkan, "Estimating the Accuracy of Learned Concepts", *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, (895-900), AAAI Press, 1993.

Timothy L. Bailey and Charles Elkan, "Unsupervised Learning of Multiple Motifs in Biopolymers using Expectation Maximization", UCSD Technical Report, CS93-302, August, 1993, University of California at San Diego.

Timothy L. Bailey, "Likelihood vs. Information in Aligning Biopolymer Sequences", UCSD Technical Report, CS93-318, February, 1993, University of California at San Diego.

ABSTRACT OF THE DISSERTATION

Discovering motifs in DNA and protein sequences:
The approximate common substring problem

by

Timothy Lawrence Bailey
Doctor of Philosophy in Computer Science
University of California, San Diego, 1995
Professor Charles P. Elkan, Chair

Discovering patterns called motifs that are repeated in groups of protein or nucleic acid sequences is an important and challenging problem in computational biology. This dissertation formulates the sequence motif discovery problem as a family of approximate common substring (ACS) problems and presents an algorithm (MEME) which solves them. The algorithm is based on stochastic models of sequences and a Bayesian variant of the expectation maximization (EM) algorithm.

MEME is unique among published motif discovery methods in its consistent use of statistically-motivated techniques to solve a wide variety of motif discovery problems. MEME motifs characterize membership in a group of sequences, and may split the group into subfamilies or reveal repeated subsequences within individual sequences in the group. The statistical nature of MEME allows background information from the problem domain to be translated into a Bayesian prior distribution or into model constraints, and used to improve the solution. A novel modification to the EM algorithm allows multiple motifs to be found, and a novel heuristic function based on the maximum likelihood ratio test (LRT) performs well at optimizing the width of motifs. A technique for mining the dataset for starting points for EM reduces the problem of local optima.

The results of extensive experiments on over 80 difficult protein and DNA datasets and a case study conducted in conjunction with a biologist show that MEME attains its goals.

- MEME selectively and sensitively discovers biologically relevant motifs in groups of related DNA or protein sequences.
- The motifs shed light on the structure and functions of different portions of the molecules represented in the group of sequences and reveal relationships among them.
- The motifs discovered by MEME have high recall, precision, and receiver operating characteristic (ROC). They correctly classify new sequences as belonging or not to the same family as those from which the motif was discovered.
- The biological relevance and classification performance of the motifs found by MEME is on a par with that of human experts and with that of other motif-finding algorithms which use more *ad hoc* approaches or require more background information than MEME.

The success of MEME at solving the protein and DNA sequence motif discovery problem makes it likely that we can solve other ACS problems involving discrete-, real-, or vector-valued sequences using similar techniques.

Chapter I

Introduction

I.A The approximate common substring (ACS) problem

Consider the following problem. We are given a set X of strings of symbols from some alphabet and told that the strings are essentially random except that embedded in each of them is a possibly different *approximate* copy of some *unknown* string S of length W. The approximate copies of S differ from S in that zero or more of the symbols in S have been changed to some other symbol by some random process which allows for mutations but does not include insertions or deletions of symbols. The approximate common substring problem is to discover the original string S and locate the substrings in each of the strings in the given set X which are the approximate copies of S.[1]

Figure I.1 shows an instance of the ACS problem and a possible solution. Each of the ten strings in the problem instance contains an approximate copy of the string "noisysubstring" inserted at a random place in what is otherwise a random

[1]The ACS problem differs from the well-known longest common subsequence problem (LCS) [Chin and Poon, 1994] in that approximate common substrings are *contiguous* sequences in the original strings and may differ from each other, whereas common subsequences need not be contiguous but must be identical.

sample problem	possible solution
krrfwluledjknolsymubstringodzj	krrfwluledjkNOLSYMUBSTRINGodzj
yffxlltunoisysufstringtgvdefub	yffxlltuNOISYSUFSTRINGtgvdefub
noboysubsprenguswksntnaixdxrgk	NOBOYSUBSPRENGuswksntnaixdxrgk
slzlgbenoioysuastringbgtzxgzhf	slzlgbeNOIOYSUASTRINGbgtzxgzhf
vnoisynubstringeoycndluabforqm	vNOISYNUBSTRINGeoycndluabforqm
etttnoisysubsqringdoozgkegboje	etttNOISYSUBSQRINGdoozgkegboje
kyjsbaxtmypjonoisysubstrungdms	kyjsbaxtmypjoNOISYSUBSTRUNGdms
mbbjyngwefzqzpjnoisysubstringh	mbbjyngwefzqzpjNOISYSUBSTRINGh
qgmtmmmfhwfynoishsubstranghhaq	qgmtmmmfhwfyNOISHSUBSTRANGhhaq
uerjjsnoisystbsmringoqindvvger	uerjjsNOISYSTBSMRINGoqindvvger
$S = ?$	$S =$ "noisysubstring"
$W = ?$	$W = 14$

Figure I.1: **Sample instance and solution of an approximate common substring problem.**

series of letters. In the possible solution, the approximate copies of S have been displayed in upper case.

The problem described above is the simplest form of the ACS problem. In practice, more difficult variants are often of interest. For example, the length W of the string S may not be known in advance and may need to be discovered as part of the solution. Another possibility is that we might be told that only *some* of the strings contain approximate copies of string S and the other strings in the given set are purely random. I call this the zero-one approximate common substring problem. More difficult yet is if we are informed that each string in the given set X can contain any number of non-overlapping approximate copies of the string S. This I call the general approximate common substring problem. One can imagine that in such a case it might be possible to detect the pattern and solve the problem even if the individual occurrences of S were extremely noisy.

Each of these variants of the ACS problem can be further complicated by adding the possibility that there is not just one common string S but a whole

multiple zero-one ACS	general ACS
irvSUBSTRKNGdjkpbfNMISYojtbbnn	kqqfvlsSAHHQTcfRLPEATadndyiiqp
vyffxlltuNOISYgSUBSTRINGiacsxm	lwarcnaiwschtnrgtAPVEAIgwcnlma
hxcplnapfwrdhdnkpggpiixzpkqmqy	qsqvirmsmahvdwqfIEPEQTwaydcfsc
SUBBTRINGpuagufbgtzxgzhfkxsjea	kEEPEVVpllqsapIEPEAHeaelteenyc
lveSUBSTRGNGjapbhztswgsczsavit	dkREAGWTqplreREPEANwvnqqkydrsc
wldoozgkegbojeuxkyjsbaxtmypjox	femMEHRATfiqcnAEPRAEykqhfwkhkc
pwhnvNOISYxgamatsjlgwhoysqdvtv	dlrclLAREATgvefypypiREPESTtwrr
uxstxNOILYzuksfuwzrSUBHTRINGcz	pgNEPRIQfsvyqmALNEAARETHSTsteq
haquuerjjsupyaubyNOISYindvvger	CEPEMTiREAEACmRGPECStgeqadrwpl
SSXSTRILGroedpkvvpapxjxugbdswh	viRREEAAfqyREPEHTfprRNPSFKitkp
S_1 = "noisy"	S = "repeat"
$W_1 = 5$	$W = 6$
S_2 = "substring"	
$W_2 = 9$	

Figure I.2: **Possible solutions to two instances of approximate common substring problems.**

set $\{S_1, S_2, \ldots, S_m\}$ of them which we must find. I call these variants of the ACS problem multiple approximate common substring problems. A solved instance of a multiple ACS problem is given in the left half of Figure I.2. Each string in the set of strings comprising the problem instance contains either zero or one approximate copies of each of the common strings S_1 and S_2. In the solution, the approximate copies are highlighted in uppercase and the common strings and their widths are indicated beneath. The right side of Figure I.2 gives a solved instance of the general ACS problem. Each string in the set contains a random number of approximate copies of of a single string S. These have been written in upper case in the solution shown in the figure.

This dissertation provides a uniform solution to all the variants of the multiple approximate common substring problem described above. Although the probabilistic nature of the ACS problem prevents the solution to any instance from being known with certainty, extensive experimentation shows that the algorithm

presented here produces highly useful solutions when applied to real ACS problems from the domain of molecular biology. Because it makes only very general assumptions about the form of the probability distribution of the approximate common substring, I expect the algorithm presented in this dissertation to do well on ACS problems from many other domains.

The remainder of this introductory chapter describes the motivating instances of ACS problems from biology which have guided this research and outlines my solution to the problem.

I.B Motifs and the ACS problem in molecular biology

Instances of approximate common substring problems arise in molecular biology in the context of analyzing DNA and protein sequences.[2] Biologists are able to determine the linear order of the building blocks of DNA and protein molecules in a procedure known as sequencing. The output of this process is referred to as a sequence in the field of biology. In the case of DNA, sequences are over a four letter alphabet; protein sequences use a twenty letter alphabet. Biological sequences are thus the strings in an ACS problem.

Since many proteins and genes within the same species and among related species have similar functions or physical structures, their sequences tend to be similar. In particular, those regions of the DNA or protein molecules most vital to maintaining a particular structure or function will tend to be most similar to their counterparts in related molecules. Other parts of the related molecules less constrained by such requirements will have tended over evolutionary time to have become more dissimilar. These islands of similarity afloat in seas of (relative) randomness, often referred to as *motifs* in the biological literature, are the approx-

[2] A brief introduction to the terminology of molecular biology used in this dissertation in given in Chapter II.

imate common substrings in an ACS problem. Locating them by computer in a set of sequences can help biologists understand the structure, function and evolutionary relationships of the genes or proteins represented by the sequences [Stormo and Hartzell, III, 1989], [Lawrence and Reilly, 1990], [Towell *et al.*, 1990], [Altschul and Lipman, 1990], [Henikoff and Henikoff, 1991], [Cardon and Stormo, 1992], [Bairoch, 1992], [Lawrence *et al.*, 1993].

I.B.1 DNA sequence analysis

A representative instance of an ACS problem in molecular biology is provided by protein binding sites in DNA molecules. The sequences in this problem are DNA and are portions of chromosomes. The approximate common substrings being sought represent sites on the chromosome where a particular protein binds in order to cause some nearby gene to be activated or deactivated. The physical geometry of a DNA molecule is roughly a long strand. The binding protein bonds selectively to places along the molecule which have the right nucleic acid building blocks in the right order. The protein binds optimally to places with some specific sequence but can still bind effectively even if one or more positions in the binding site sequence deviate from this ideal binding site sequence [Berg and von Hippel, 1988], [Li *et al.*, 1985]. Most of the interaction between the protein and the DNA molecule occurs in a short region along the DNA strand, generally about twenty DNA nucleic acid building blocks long. DNA on either side of this binding site tends to have little effect on binding and appears to be much more random in composition. Since several genes tend to be activated or deactivated in concert by a single binding protein, groups of DNA sequences can be assembled which constitute an instance of an ACS problem.

An example of a solution of a multiple ACS problem from molecular biology is shown in Figure I.3. The strings in the example are DNA sequences from the genome of *Escherichia coli* bacteria which contain binding sites for two different

multiple general ACS

TTTTGTGGCATCGGGCGAGAatagcgcgtggtgtgaaagactgtTTTTTTGATCGTTTTCACAAa
aacAAAAGTGTCTATAATCACGGcagaaaagtccacattgaTTATTTGCACGGCGTCACACtttg
tataactttataaattcctaaaattacacaaagttaatAACTGTGAGCATGGTCATATttttatc
acagtaatacattgatgtactgcatGTATGCAAAGGACGTCACATtaccgtgcagtacagttgat
ggtcaatcagcaAGGTGTTAAATTGATCACGTtttagaccatttttcgtcgtgaaactaaaaaa
aacacttgaTACTGTATGAGCATACAGTAtaattgcttcaacagaacatattgactatccggtat
tcttTACTGTATATAAAACCAGTTtaTACTGTACACAATAACAGTAATGGTTTTTCATACAGGAa

S_1 = "ttatttgaacgtcgtcacat"

$W_1 = 20$

S_2 = "tactgtatatatatacagta"

$W_2 = 20$

Figure I.3: **Possible solution of an approximate common substring problem from *E. coli* bacteria—crp and LexA protein binding sites.**

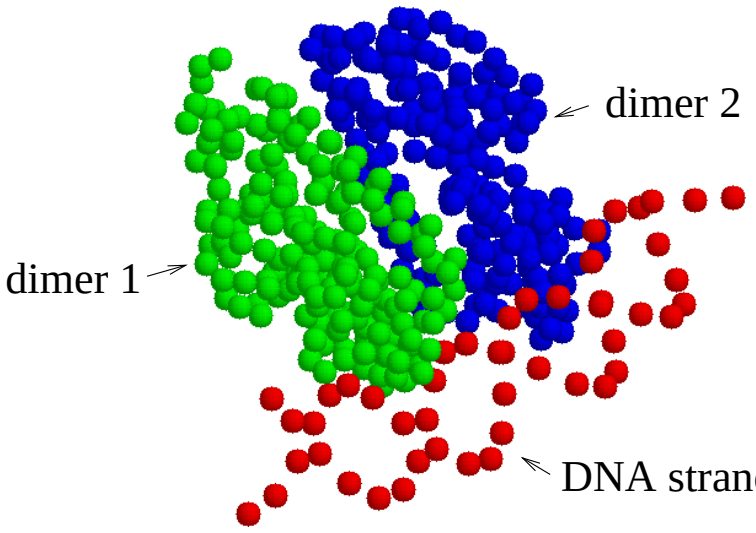


Figure I.4: **Theoretical model of catabolite activator protein (cap) bound to twenty base pair strand of DNA.** (Source: Brookhaven Protein Database entry 2gap 15-OCT-94.) The two identical halves (dimers) of the binding protein are shown above the double-helical DNA strand.

proteins referred to as cyclic-AMP receptor protein (crp) and LexA protein, respectively [Hertz *et al.*, 1990], [Stormo and Hartzell, III, 1989]. The DNA alphabet contains only the four letters A, C, G and T. The two hidden substrings and their widths are shown at the bottom of the figure. The approximate occurrences of each of the substrings are shown in uppercase in this solved instance. The example is a multiple general ACS problem instance because string S_1 occurs twice in the first two sequences, and string S_2 occurs three times in the the last sequence.[3]

A theoretical model of a protein that binds to DNA in a way similar to that of crp is shown in Figure I.4 bound to a short DNA strand [Weber and Steitz, 1984]. The protein consists of two identical parts which bind to the helical DNA strand, interacting over a region corresponding to a sequence of width approximately twenty positions, just as do crp and LexA protein.

I.B.2 Protein sequence analysis

Perhaps the largest number of ACS problems of interest to molecular biologists arise in the context of protein sequence analysis. The three dimensional structures and functions of proteins are highly correlated with their amino acid sequences [Weber, 1990], [Schiffer *et al.*, 1990]. Thousands of groups of related sequences have been assembled by biologists where the sequences share one or more approximate substrings [Bairoch, 1992]. Proteins, like DNA, are linear molecules.[4] The approximate substrings often correspond to the portions of protein strands of major interest to biologists. These include critical regions responsible for the structure and function of the protein such as the active sites of enzymes, DNA binding sites and structure-stabilizing amino acids. Characterizing and locating approximate substrings in groups of protein sequences is thus of great use in understanding proteins.

[3] This example is taken from a larger set of sequences which is used in testing my solution to ACS problems as described in Chapter V.

[4] As is described in the next chapter, each protein folds into a three dimensional configuration which to a large degree determines its biological activity.

Groups of proteins most often contain multiple approximate common substrings, so they are instances of multiple ACS problems. Related proteins can usually be thought of as consisting of sequences of motifs (approximate substrings) separated by varying length random strings. The motifs correspond to the critical portions of the protein which must be preserved from excessive mutation while the relatively random strings separating them correspond to regions of the protein less subject to evolutionary constraints. This property of proteins accounts for the large number of multiple sequence alignment algorithms which have been developed [Feng and Doolittle, 1987], [Carillo and Lipman, 1988], [Gribskov *et al.*, 1990], [Krogh *et al.*, 1994]. The order of the motifs is usually characteristic of the protein family, though deletions of entire motifs and cyclic permutations can and do occur. Such high-level patterns cannot be handled by multiple sequence alignment.

It is estimated that 15% to 20% of protein sequences contain repeats [Gribskov and Devereux, 1991], so the general ACS problem is also of interest in protein sequence analysis.

I.C Solution to the ACS problem

My solution to the approximate common substring problem is to model the approximate substring and the strings in which it appears stochastically. I propose a slightly different model for each variant of the ACS problem discussed above—one occurrence per string, zero or one occurrence per string and the general ACS problem. Each of the variants is then solved in the same way using an algorithm which combines a Bayesian adaptation of the expectation maximization (EM) [Dempster *et al.*, 1977], [Redner and Walker, 1984] algorithm for finite mixture models with a statistically motivated objective function for choosing the length W of the approximate string S. The multiple ACS problem is solved by fitting a series of stochastic string models to the set of strings, each of which models the strings as though they contained only one approximate substring. As a

aligned substrings		T	T	T	T	T	T	G	A	T	C	G	T	T	T	T	C	A	C	A	A
		T	T	A	T	T	T	G	C	A	C	G	G	C	G	T	C	A	C	A	C
		A	A	C	T	G	T	G	A	G	C	A	T	G	G	T	C	A	T	A	T
		G	T	A	T	G	C	A	A	A	G	G	A	C	G	T	C	A	C	A	T
		C	C	A	T	T	T	T	T	T	C	G	T	C	G	T	G	A	A	A	C
probability matrix	A	2	2	6	1	0	0	2	6	4	0	2	2	1	1	1	1	9	2	9	2
	C	2	2	2	0	0	3	0	2	0	7	0	0	5	0	0	7	0	5	0	4
	G	2	0	0	0	4	0	6	0	2	2	7	2	2	7	0	2	0	0	0	0
	T	4	6	2	9	6	7	2	2	4	1	1	6	2	2	9	1	1	2	1	4
consensus string		T	T	A	T	T	T	G	A	A	C	G	T	C	G	T	C	A	C	A	T

Figure I.5: **The occurrences, probability matrix representation, and inferred consensus string of the crp binding sites (approximate common substring S_1 in Figure I.3).** All probabilities have been multiplied by ten and rounded to the nearest integer. The probabilities in the matrix are *not* just the observed letter frequencies in the aligned substrings because a Bayesian prior has been used in the estimation of the matrix for reasons which will be explained later.

model for each different approximate substring is discovered, it is incorporated into each subsequent model in the series of stochastic models as a prior distribution. My solution to the ACS problem, although heuristic, has a uniformly statistical motivation. I call my algorithm MEME for Multiple EM for Motif Elicitation.

The key idea behind the approach is to model the the probability distribution of the approximate common substring rather than trying to discover the original substring itself. The substring model we use for all variants of the ACS is the same—a sequence of independent discrete random variables, each of which is responsible for generating one position in each occurrence of the approximate substring. This type of model can be fully specified by a probability matrix that specifies the probability of each letter in the alphabet appearing at each position in an occurrence of the approximate substring S. We can use the *consensus sequence*, i.e., sequence of highest-probability letters, as an estimate of S, but this throws away the information contained in the probability matrix about the different distributions of letters at each position of the approximate substring. We can also

use the probability matrix to make predictions about where the occurrences of the approximate substring are within the strings in the set X based on how well each substring in the strings matches the probabilistic substring model.

Figure I.5 illustrates the probability matrix representation of an approximate substring as well as its motivation. Assume that the aligned substrings in the figure are the actual occurrences of the approximate substring S_1 (the crp binding sites in Figure I.3). The observed frequencies of the letters in each column of the alignment can be shown to be the maximum likelihood estimate (MLE) of the parameters of the stochastic model for S given the observed occurrences of S. The stochastic substring model with the probability matrix representation shown beneath the alignment can easily be seen to have high chance of producing the observations in the alignment.

In an instance of an ACS problem we are not told the locations of the occurrences of the approximate substring S within the sequences in X, so we cannot use them to estimate the parameters of the stochastic model described above. Nor are we told what the approximate substring looks like, i.e., the parameters of the stochastic model, in any variant of the ACS problem of interest here. My solution to the ACS problem uses the expectation maximization algorithm to *simultaneously* estimate the locations of the occurrences *and* the parameters of the substring model. In a nutshell, EM works by first using a guess at the parameters of the substring model to predict where the occurrences are and then updates the model parameters based on assuming those predictions are observations of occurrences of S. Using the updated guess at the parameters of the model, the process begins anew. This cycle of estimating the positions of occurrences using the string model and then using those estimates to reestimate the parameters of the model is repeated until the parameters of the model converge.

The EM algorithm has good convergence properties but can easily get trapped by local optima of the likelihood function which it optimizes. To avoid

this, my solution uses several additional search heuristics. I also use Bayesian priors on the parameters of the substring model [Krogh *et al.*, 1994], to incorporate background knowledge from the problem domain. The solution also uses a novel objective function for discovering the correct width of the common substring. As mentioned previously, multiple ACS problems are solved by greedy search with different common substrings being found one-at-a-time.

I.D Overview of the dissertation

Chapter II describes related work and provides a simple overview of some of the aspects of molecular biology which bear upon this dissertation. The first section of that chapter can be omitted by those already familiar with molecular biology and sequence analysis. The complete solution to the ACS problem is given in Chapter III. This chapter is self-contained and explains the algorithms involved in the solution in detail. Additional mathematical and technical details such as complete derivations of the probability equations used by EM with each of the string models for each of the different variants of the ACS problem are given in Chapter IV. That chapter can be safely skipped by those not interested in the mathematical details without loss of continuity. I apply and demonstrate the robustness of my solution to the ACS problem in Chapter V. This chapter demonstrates the utility of the solution by measuring its ability to solve numerous ACS problems from molecular biology. For example, I demonstrate that it displays expert-level performance at discovering motifs in 75 distinct protein-family sequence datasets. Chapter V also compares the performance of my solution with that of other algorithms. A discussion of the contributions of this research and future directions concludes the dissertation in Chapter VI.

Chapter II

Background and related work

The problem of discovering sequence motifs that characterize protein families and can be used to predict the family membership, structure and functionality of new proteins is an important one in molecular biology. Protein sequence databases currently contain tens of thousands of individual proteins and are increasing in size daily. Thousands of protein families have been identified and sequence motifs for them have often been discovered by humans with limited help from automatic motif discovery systems. Many of these families contain multiple subfamilies with different properties, so a motif discovery system should be able to discover multiple different motifs in a set of sequences. Finding motifs in DNA sequences presents different challenges and opportunities for illuminating biological functions such as gene expression and metabolism.

Two problems related to the ACS problem have been heavily studied but do not appear to lead to solutions to the ACS problem. The longest common subsequence (LCS) problem is to determine the longest subsequence that can be obtained by deleting zero or more symbols from each string in a pair of strings [Apostolico *et al.*, 1992], [Chin and Poon, 1994]. Although variants of this problem with more than two strings have been studied, it differs from the ACS in that the subsequences need not be contiguous in the original sequences and must match

each other exactly. The approximate string matching problem (ASM) looks for all the positions in a string which match a pattern string with at most k mismatches [Ukkonen and Wood, 1993], [Chang and Lawler, 1994], so it involves only one string and an already known pattern. Because of these differences, the approaches which have been used to solve these related problems do not appear to be of any direct use in solving the ACS problem. Formulating the problem formally as the ACS problem has not been done previously and has several advantages including its generality of approach and statistical motivation.

This chapter begins with a brief overview of relevant concepts and terminology in molecular biology and biological sequence analysis. A more comprehensive introduction to sequence analysis and artificial intelligence applications in molecular biology can be found in Hunter [1983] and Gribskov and Devereux [1991]. The chapter concludes with a discussion of various approaches to the motif discovery problem and related problems in computational biology.

II.A Basic molecular biology

Biologists deal with sequence data in the form of strings of letters which represent the chemical formulas of proteins and nucleic acid (DNA and RNA) molecules. Sequencing projects involving thousands of biologists are currently underway around the world and are producing vast amounts of sequence data faster than it is possible to perform detailed studies on the proteins and nucleic acids to determine their three dimensional structures and biological functions. Much effort has gone into developing computer algorithms to analyze raw biological sequence data and reduce the amount of human biological expertise and laboratory experimentation necessary in order to understand the properties of the molecules they represent.

II.A.1 Proteins, DNA and RNA

The three most important molecules of life—proteins, DNA and RNA—are all linear. Proteins are all assembled from twenty different *amino acids* hooked together in long chains by chemical bonds known as *peptide bonds*. The average protein is a chain of approximately 300 to 400 amino acids. Most proteins contain few if any other bonds connecting their constituent amino acids together, so they are truly linear molecules. Every DNA molecule consists of varying numbers of four different *nucleic acids* assembled into a linear molecule which is most often found bonded to its biological "mirror image" and twisted into the famous double helix. The mirror image property refers to the fact that the nucleic acids in one single-stranded DNA molecule can bond to those in another strand forming what are known as *Watson-Crick pairs*—the rungs of the ladder in the double-helix. Of the four different nucleic acids composing DNA molecules, *adenine* (A) bonds preferentially with *thymine* (T) and *cytosine* (C) with *guanine* (G). This complementary-pairing property is crucial in the replication of DNA as well as in protein synthesis. Each *chromosome* in the cell is a single DNA molecule hundreds of thousands (or millions, depending on the organism) of amino acids long. The strongest bonds in DNA molecules are those along the linear backbone; slight heating is often enough to cause the two strands in the double helix to pull apart. RNA molecules are composed of chains of the same four nucleic acids as DNA, except that thymine is replaced by *uracil* (U). They tend to fold upon themselves and form relatively weak bonds between Watson-Crick pairs of nucleic acids.

One of the key scientific breakthroughs of this century was the discovery that DNA molecules duplicate themselves and create templates upon which all of the proteins needed by life are made. Chromosomes store the genetic information necessary for cells to create proteins. A *gene* is a contiguous portion of a chromosome that "codes" for a particular protein. In a multi-step process, the cell manufactures the proteins it needs by first copying the DNA in a gene

into an RNA molecule that is a mirror image of the gene. Then, the protein is assembled by a process which "reads" the RNA and creates a chain of amino acids. Which of the twenty standard amino acids appears at each position in the protein is determined by the three nucleic acids in the corresponding position in the RNA molecule according to the so-called "genetic code". This code assigns each of the $4^3 = 64$ possible nucleic acid triplets one of the twenty amino acids (or "stop", which causes the protein chain to terminate). The fact that proteins, DNA and RNA molecules are all linear polymers is thus not coincidental, but is a direct result of the way in which the cell stores and uses information about how proteins are made.

II.A.2 Gene regulation

In addition to genes, chromosomes contain stretches of DNA which do not code for protein but contain information that is used to regulate when a protein is produced and in what quantities. These "regulatory sites" perform their functions by temporarily forming bonds with messenger proteins which either enhance or inhibit the process of protein synthesis. Whether or not a messenger bonds to a regulatory site depends on the particular sequence of nucleic acids in the DNA at the site.

II.A.3 Sequence databases

Recent advances in biology have made it possible to rapidly "sequence" large amounts of protein, DNA and RNA. Sequencing gives the primary, linear structures of the molecules, but it does not directly reveal their three dimensional structure nor their function. Determining detailed structure and functional information is expensive and time consuming and the genomes of humans and other organisms each contain millions of amino acids and code for thousands of different proteins. This accounts for the interest in understanding proteins and the

genome directly from relatively inexpensive primary sequence data. Knowledge of the primary sequence of biological molecules is interesting, but what is really desired is deeper knowledge about how these long chains of amino or nucleic look (three dimensionally) and behave within the cell. The challenge is to discover this information automatically from the steadily growing sequence databases such as Genbank [Moore *et al.*, 1990], the Protein Identification Resource (PIR) [George *et al.*, 1986] and SWISS-PROT [Bairoch, 1994]. These databases contain DNA and protein sequences and annotation such as references to pertinent articles in the literature. They can be accessed via anonymous ftp from the NCBI server at ncbi.nih.nlm.gov.

II.B Discovering Patterns in Biological Sequences

Supervised learning problems using biological sequences as input have been approached using neural nets, genetic algorithms and hidden Markov models (HMM). Greedy algorithms and statistical modeling techniques such as expectation maximization (EM) have also been used. By far the most widely used technique, however, have been simple nearest-neighbor algorithms which classify a new sequence by computing a distance between it and every known sequence, and reporting the closest neighbors. In many cases the "learner" has been the biologist herself. Ever since it became possible to sequence DNA and proteins, biologists have visually compared sequences and looked for patterns.

II.B.1 Sequence alignment

It was quickly observed that many protein sequences could be *aligned* so that the same letters would line up in each of the sequences if a few gaps were inserted in one or the other sequence. In cases where different letters lined up, it was often the case that the amino acids they represented had similar chemical

properties, so substituting one for the other was less likely to change the properties of the protein than a random substitution. Sequence alignment of two or more sequences forms the basis of most biological sequence learning algorithms proposed to date, including nearest-neighbor, genetic algorithm, greedy search, HMM and EM techniques.

Very closely related proteins and genes can be aligned *globally* by inserting at most a few gaps in each of the sequences. More distant protein relationships and many DNA and RNA relationships cannot be easily detected by global sequence alignment. Even in these cases, however, sequence alignment can be used *locally* because short stretches of protein or DNA are often highly *conserved* (i.e., similar). As I mentioned in the introduction, this happens because certain portions of a protein may be more critical to its 3-D geometry or chemical function and are therefore *constrained* by evolutionary forces to remain relatively unchanged even in distant relatives. In DNA, these similar stretches of sequence are often regulatory sites which must conserve their nucleic acid sequence in order to continue to perform their function in the cell.

II.B.2 Motifs

Linear biological molecules can thus be thought of as being composed of *motifs*—short stretches of molecule with highly constrained sequence—interspersed in relatively unconstrained sequence. This view has been highly successful in yielding insights about proteins and DNA. It leads directly to the concept of a motif pattern language for describing biological sequences. Many motif pattern languages are possible.

- An actual sequence can be used as a motif description. This is sometimes referred to as a *consensus* sequence since it is often composed of the consensus letter in each column of an alignment of the known examples of a motif.

- A slightly more complicated motif language allows several choices of letter at each position of the motif. This is a direct generalization of a consensus sequence. An example of such a pattern is

  D-[SGN]-D-P-[LIVM]-D-[LIVMC],

  of which the string "DGDPLDC" would be one possible occurrence.

- A more biologically plausible representation of a motif is a probability matrix which assigns a different probability to each possible letter at each position in the motif [Schneider *et al.*, 1986]. An example of this type of motif is:

A	2	2	6	1	0	0	2	6	4	0	2	2	1	1	1	1	9	2	9	2
C	2	2	2	0	0	3	0	2	0	7	0	0	5	0	0	7	0	5	0	4
G	2	0	0	0	4	0	6	0	2	2	7	2	2	7	0	2	0	0	0	0
T	4	6	2	9	6	7	2	2	4	1	1	6	2	2	9	1	1	2	1	4

II.C Related work

II.C.1 Finding motifs by hand

The work of discovering biological motifs has often been done by hand. The set of sequences X which constitutes an instance of the ACS problem must be assembled, usually by a biologist expert in the particular family of proteins or genes comprising the set of sequences [Bairoch, 1992]. Traditionally this is done from prior knowledge of what sequences are likely to contain common motifs or by searches of sequence databases using a single sequence as a probe. These searches do pairwise global [Needleman and Wunsch, 1970] or local [Smith and Waterman, 1981] alignments between the probe sequence and each sequence in the database. The methods used in such pairwise alignments, mostly different forms

of dynamic programming algorithms, do not scale well to multiple alignments [Altschul and Lipman, 1990].

The next step in searching for motifs, multiple sequence alignment, has traditionally been done using clustering [Feng and Doolittle, 1987], branch and bound [Carillo and Lipman, 1988], or greedy algorithms. Recently, hidden Markov models (HMM) which use EM to learn their parameters have been introduced for doing multiple sequence alignments [Krogh *et al.*, 1994], [Baldi *et al.*, 1994]. Potential motifs identified by eye from multiple alignments are then usually screened using prior knowledge for biological relevance and then tuned by hand to maximize their selectivity for the family or characteristic they are intended to capture. Our intention is that most of this manual and subjective labor be replaced by the automatic discovery of motifs using MEME.

II.C.2 Automatic motif discovery

Several variants of the ACS problem have been studied in the guise of discovering motifs in biological sequences. Turning sequence alignments into motifs was pioneered by Gribskov *et al.* [1990]. This work introduced a type of scoring matrix called a *profile* which is very similar to the kind of motif learned by MEME except that it allows for gaps and insertions in sequences matching the motif. A greedy algorithm for learning a stochastic model of approximate common substrings was developed by Stormo and Hartzell, III [1989]. This approach was improved upon by the replacing the greedy algorithm with EM [Lawrence and Reilly, 1990], but only the one-occurrence per sequence ACS problem was addressed. This work also did not address the questions of avoiding local optima, finding multiple motifs or incorporating background information into the model. A different but related approach to finding motifs using iterative Gibbs sampling [Lawrence *et al.*, 1993] has good time complexity and convergence properties and can discover multiple motifs, but requires the number of occurrences of each motif

in each sequence in the dataset be specified as input to the algorithm. The algorithm presented in this dissertation, on the other hand, is capable of discovering this information on its own. I compare the performance of the Gibbs sampling algorithm with MEME in Chapter V.

The first algorithm for discovering multiple motifs in sets of biological sequences was probably BLOCKMAKER [Henikoff and Henikoff, 1991]. It differs from MEME in that it is not based on a statistical model and outputs only motifs which are present in the same order in all of the sequences in the input set. The algorithm underlying BLOCKMAKER is called MOTIF [Smith *et al.*, 1990], and searches for words consisting of three letters separated by two distances in a dataset. Adjacent words with the highest frequency of occurrence are then combined into motifs and used by BLOCKMAKER to find patterns that occur in each sequence in some subset of the training set.

Recently an algorithm [Saqi and Sternberg, 1994] similar in flavor to [Smith *et al.*, 1990] was published which discovers multiple motifs by clustering words in a dataset of length k which have at least $r < k$ matches. Clustering is applied to the most frequently occurring patterns in order to combine related ones and arrive at a reduced set of motifs. A closely related approach is that of looking for pairs of approximately matching words which can be separated by any distance [Wang *et al.*, 1994]. The approximate match criterion used there is edit distance, i.e., a fixed number of insertions, deletions or mutations are allowed between two approximately matching words. In that work, the motifs discovered by their DISCOVER program were used as classifiers and the results found to compare favorably to and complement those of BLOCKMAKER in the sense that if the two systems agree on a classification, it is most likely correct.

II.C.3 Using motifs for searching databases

Once motifs have been found, a primary use for them is to search databases for other sequences (not contained in the training set) which contain occurrences of them. This invariably involves sliding some kind of scoring matrix along each sequence in the databases to score each (overlapping) sequence. Recent work verifies that the Bayesian approach used in this dissertation is at least as good as any of the other methods which have been tried [Tatusov *et al.*, 1994]. Like Tatusov *et al.*, I convert the probability matrix representing the motif into a log-odds scoring matrix and choose a threshold for deciding if a subsequence matches a motif or not. An objective criterion for choosing the the threshold and at the same time determining the statistical significance of each possible score is also given in Tatusov *et al.* [1994]. That work also discusses the idea of iteratively adding new subsequences from a large database to the motif (actually to the alignment of motif occurrences) which score above some threshold. This requires knowing at least one occurrence of the motif in advance, so it does not address the more general problem of completely automatic motif discovery addressed by MEME.

Chapter III

Algorithms

My solution to the problem of finding motifs in biological sequences is presented in this chapter. The solution is a computer program which I call MEME which uses statistical modeling and artificial intelligence heuristics to discover motifs by fitting a statistical model to a dataset of sequences. This chapter discusses and motivates the statistical models, objective functions, heuristics and algorithms which MEME uses. It also discusses how motifs learned by MEME can be used as classifiers and how two utility programs PROBER and NOTE apply them as such. Chapter IV gives detailed derivations of the functions and algorithms.

III.A Overview of MEME

The principal input to MEME is a set of DNA or protein sequences. Its principal output is a series of probabilistic sequence models, each corresponding to one motif, whose parameters have been estimated by expectation maximization [Dempster *et al.*, 1977]. In a nutshell, MEME's algorithm is a combination of

- expectation maximization (EM),
- an EM-based heuristic for choosing the starting point for EM,

- a maximum likelihood ratio-based (LRT-based) heuristic for determining the best number of model free parameters,
- multistart for searching over possible motif widths, and
- greedy search for finding multiple motifs.

The objective of MEME is to discover the occurrences of motifs in a dataset of sequences and to output the positions of the motif occurrences and descriptions of the motifs. The user of MEME provides the dataset of sequences and specifies a type of sequence model from among three different types which MEME supports. Each of these sequence model types incorporates different assumptions about the number and distribution of occurrences of motifs in the dataset. In particular, they assume either exactly one motif occurrence per sequence, zero or one occurrence per sequence, or any number of (non-overlapping) motif occurrences per sequence.

The models used by MEME are all finite mixture models, one component of which describes the motif. The sequences in the dataset are assumed to be independent samples from some model of the type specified by the user. MEME fits the parameters of the model to the observed data (the sequences), and outputs the parameters of the motif component of the sequence model. To discover multiple different, non-overlapping motifs, MEME repeats this process using the estimated positions of motif occurrences already found as a statistical prior during parameter fitting.

MEME discovers a motif by considering alternative models of the type specified by the user. The models considered differ only in the width of the motif which they assume. For each model, MEME uses a Bayesian variant of EM to find the best values of its free parameters given the dataset of sequences. Through the use of different Bayesian priors, background knowledge about the properties of the molecules which the sequences represent can be used to inform the search

for motifs. Background information about certain types of motifs motifs such as DNA palindromes can be incorporated into the model by constraining some of the model free parameters in certain ways. The use of Bayesian priors also alleviates to some extent the problem of getting stuck at local optima discussed below.

EM suffers from a tendency to get stuck at local optima. One way of overcoming this problem is to rerun EM repeatedly from different random starting points—initial values of the model free parameters—and choose the model with the highest likelihood. A faster method is to find one good starting point and run EM to convergence from it. Because EM usually converges quickly from good starting points, the likelihood of the model after one iteration of EM is a useful measure of starting point goodness. MEME uses a method based on this idea to choose a good starting point for EM. This is done via a dynamic programming algorithm which simultaneously estimates the goodness of many possible starting points.

In addition, the starting points tested are not random. Some of the subsequences in the dataset are presumed to be motif occurrences. MEME generates potential starting points by mapping subsequences to model parameters. It systematically tests all starting points which can be generated in this way from the actual subsequences in the dataset. EM is only run to convergence from the best of these starting points.

As mentioned above, MEME considers alternative models of the type selected by the user with differing motif widths. It also considers certain biologically plausible constraints on the free parameters of the models. These models have differing numbers of free parameters, so their likelihoods cannot be used directly to choose the best among them. To choose the best model, MEME uses a heuristic function based on the maximum likelihood ratio test. This function computes a score for a model from its likelihood and number of free parameters. The value of this function is computed for each of the final models delivered by EM, and

the one with the best value of this function is chosen as the final model. The motif component of this model is output and the estimated positions of its occurrences are used during the discovery of succeeding motifs to avoid rediscovering the same motif. Finding one motif at a time avoids the combinatorial explosion in possible numbers of different motifs of different widths with different numbers of occurrences per sequence.

III.B Models

III.B.1 OOPS, ZOOPS, and TCM models

The different types of sequence model supported by MEME make differing assumptions about how and where motif occurrences appear in the dataset. I call the simplest model type OOPS since it assumes that there is exactly one occurrence per sequence of the motif in the dataset. This type of model was introduced by Lawrence and Reilly [1990]. The ZOOPS sequence model type is a generalization of the OOPS type and assumes zero or one motif occurrences per dataset sequence. Finally, TCM (two-component mixture) models assume that there are zero or more non-overlapping occurrences of the motif in each sequence in the dataset.

Each of these types of sequence model consists of two components which model, respectively, the motif and non-motif ("background") positions in sequences. A motif is modeled by a sequence of discrete random variables whose parameters give the probabilities of each of the different letters (4 in the case of DNA, 20 in the case of proteins) occurring in each of the different positions in an occurrence of the motif. The background positions in the sequences are modeled by a single discrete random variable. If the width of the motif is W, and the alphabet for sequences is $\mathcal{L} = \{a, \ldots, z\}$, we can describe the parameters of the two components of each of the three model types in the same way as

$$\theta = [\ \theta_0 \quad \theta_1\ ] = [\ \mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_2 \quad \ldots \quad \mathbf{p}_W\ ]$$

$$= \begin{bmatrix} P_{a,0} & P_{a,1} & P_{a,2} & \ldots & P_{a,W} \\ P_{b,0} & P_{b,1} & P_{b,2} & \ldots & P_{b,W} \\ \vdots & \vdots & \vdots & & \vdots \\ P_{z,0} & P_{z,1} & P_{z,2} & \ldots & P_{z,W} \end{bmatrix}.$$

Here, $P_{x,j}$ is the probability of letter x occurring at either a background position ($j = 0$) or at position j of a motif occurrence ($1 \leq j \leq W$), θ_0 is the parameters of the background component of the sequence model, and θ_1 is the parameters of the motif component.

Formally, the parameters of an OOPS model are the letter frequencies θ for the background and each column of the motif, and the width W of the motif. The ZOOPS model type adds a new parameter, γ, which is the prior probability of a sequence containing a motif occurrence. A TCM model, which allows any number of (non-overlapping) motif occurrences to exist within a sequence, replaces γ with λ, where λ is the prior probability that any position in a sequence is the start of a motif occurrence.

III.B.2 DNA palindromes

A DNA palindrome is a sequence whose inverse complement is the same as the original sequence. DNA binding sites for proteins are often palindromes. MEME models a DNA palindrome by constraining the parameters of corresponding columns of a motif to be the same:

$$\theta_1 = \begin{bmatrix} P_{a,1} & P_{a,2} & \ldots & P_{t,2} & P_{t,1} \\ P_{c,1} & P_{c,2} & \ldots & P_{g,2} & P_{g,1} \\ P_{g,1} & P_{g,2} & \ldots & P_{c,2} & P_{c,1} \\ P_{t,1} & P_{t,2} & \ldots & P_{a,2} & P_{a,1} \end{bmatrix}.$$

That is,

$$P_{a,i} = P_{t,W+1-i},$$

$$
\begin{aligned}
P_{c,i} &= P_{g,W+1-i}, \\
P_{g,i} &= P_{c,W+1-i}, \\
P_{t,i} &= P_{a,W+1-i}
\end{aligned}
$$

for $i = 1, \ldots, \lfloor W/2 \rfloor$. The last column is an inverted version of the first column, the second to last column is an inverted version of the second column, and so on. Notice also that, although corresponding columns in the palindromic motif model have the same parameter values, the columns in the motif are still independent. In other words, in a motif occurrence (i.e., a sample from the distribution over sequences of length W defined by the motif), the probability of a particular letter occurring in any position is not affected by knowledge of which letter occurred at any other position. As will be described below, MEME automatically chooses whether or not to enforce the palindrome constraint, doing so only if it improves the value of the LRT-based objective function.

III.C Expectation maximization

Consider searching for a single motif in a set of sequences by fitting one of the three sequence model types to it. The dataset of n sequences, each of length L, will be referred to as $X = \{X_1, X_2, \ldots, X_n\}$.[1] There are $m = L - W + 1$ possible starting positions for a motif occurrence in each sequence. The starting point(s) of the occurrence(s) of the motif, if any, in each of the sequences are unknown and are represented by the the variables (called the "missing information") $Z = \{Z_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m\}$ where $Z_{i,j} = 1$ if a motif occurrence starts in position j in sequence X_i, and $Z_{i,j} = 0$ otherwise. The user selects one of the three types of model and MEME attempts to maximize the likelihood function of a model of that type given the data, $Pr(X|\phi)$, where ϕ is a vector containing all the parameters of

[1] It is not necessary that all of the sequences be of the same length, but this assumption will be made in what follows in order to simplify the exposition of the algorithm. In particular, under this assumption, $\lambda = \gamma/m$.

the model. MEME does this by using EM to maximize the expectation of the joint likelihood of the model given the data and the missing information, $Pr(X, Z|\phi)$. This is done by selecting an initial value $\phi^{(0)}$ for the model parameters and then repeating the following two steps, in order, until a convergence criterion is met.

- E-step: compute

$$Z^{(t)} = \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z]$$

- M-step: solve

$$\phi^{(t+1)} = \mathop{\mathrm{argmax}}_{\phi} \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[\log Pr(X, Z|\phi)].$$

This process is known to converge [Dempster *et al.*, 1977] to a local maximum of the likelihood function $Pr(X|\phi)$.

III.C.1 Joint likelihood functions

MEME assumes each sequence in the training set is an independent sample from a member of either the OOPS, ZOOPS or TCM model families and uses EM to maximize one of the following likelihood functions. (Refer to Tables IV.1, IV.2 and IV.3 for a summary of the notation used here.) The logarithm of the joint likelihood for models of each of the three model types is as follows. For an OOPS model, the joint log likelihood is

$$\log Pr(X, Z|\theta, W) = \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j} \log Pr(X_i|Z_{i,j} = 1, \theta) + n \log \frac{1}{m}. \quad \text{(III.1)}$$

The first term in Equation III.1 is the logarithm of the probability of the sequences given that we know where the motif occurrences start, $Pr(X|Z, \theta)$, and the second term is the logarithm of the prior probability of those starting points, $Pr(Z|\theta)$. For a ZOOPS model, the joint log likelihood is

$$\log Pr(X, Z|\theta, \gamma, W) = \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j} \log Pr(X_i|Z_{i,j} = 1, \theta)$$

$$+\sum_{i=1}^{n}(1-Q_i)\log Pr(X_i|Q_i=0,\theta)$$
$$+\sum_{i=1}^{n}(1-Q_i)\log(1-\gamma)+\sum_{i=1}^{n}Q_i\log\lambda. \quad \text{(III.2)}$$

The variable Q_i used above is defined as $Q_i = \sum_{j=1}^{m} Z_{i,j}$. Thus, $Q_i = 1$ if sequence X_i contains a motif occurrence, and $Q_i = 0$ otherwise. Also, $\lambda = \gamma/m$ in Equation III.2. The first two terms in Equation III.2 are the logarithm of the probability of the sequences given that we know where the motif occurrences start, $Pr(X|Z,\theta)$, and the third and fourth terms are the logarithm of the prior probability of those starting points, $Pr(Z|\theta)$. Four terms are necessary because the cases where there is a motif occurrence in sequence X_i, $Q_i = 1$, and where there is not, $Q_i = 0$, are covered separately. For a TCM model, the joint log likelihood is

$$\begin{aligned}\log Pr(X,Z|\theta,\lambda,W) &= \sum_{i=1}^{n}\sum_{j=1}^{m}(\ (1-Z_{i,j})\log Pr(X_{i,j}|\theta_0)+Z_{i,j}\log Pr(X_{i,j}|\theta_1)\\ &\quad +(1-Z_{i,j})\log(1-\lambda)+Z_{i,j}\log\lambda\ ). \quad \text{(III.3)}\end{aligned}$$

The first two terms in Equation III.3 are the logarithm of the probability of the sequences given that we know where the motif occurrences start, $Pr(X|Z,\theta)$, and the third and fourth terms are the logarithm of the prior probability of those starting points, $Pr(Z|\theta)$. The first and third terms cover the case were $X_{i,j}$ is not the start of a motif occurrence, and the second and fourth terms cover the case were it is.

The conditional sequence probabilities for sequences containing a motif used by OOPS and ZOOPS models are defined as

$$\log Pr(X_i|Z_{i,j}=1,\theta) = \sum_{k=0}^{W-1}\mathbf{I}(i,j+k)^T\log\mathbf{p}_k+\sum_{k\in\Delta_{i,j}}\mathbf{I}(i,k)^T\log\mathbf{p}_0,$$

where $\mathbf{I}(i,j)$ is a vector-valued indicator variable of length $A = |\mathcal{L}|$, whose entries are all zero except the one corresponding to the letter in sequence X_i at position j, $X_{i,j}$. $\Delta_{i,j} = \{1,2,\ldots,j-1,j+W,\ldots,L\}$ is the set of positions in sequence X_i which lie outside the occurrence of the motif when the motif starts at position

j. The conditional probability of a sequence without a motif occurrence under a ZOOPS model is defined as

$$Pr(X_i|Q_i = 0, \theta) = \prod_{k=1}^{L} P_{X_{i,k},0}.$$

The conditional probability of a length-W subsequence generated according to the background or motif component of a TCM model is defined to be

$$\log Pr(X_{i,j}|\theta_c) \quad = \quad \sum_{k=0}^{W-1} \mathbf{I}(i, j+k)^T \log \mathbf{p}_{k'},$$

where $k' = 0$ if $c = 0$ (background), and $k' = k + 1$ if $c = 1$ (motif).

III.C.2 The E-step

The E-step of EM calculates the expected value of the missing information—the probability that a motif occurrence starts in position j of sequence X_i. The formulas used by MEME for the three types of model are given below. Derivations are given in Chapter IV. For an OOPS model,

$$Z_{i,j}^{(t)} \quad = \quad \frac{Pr(X_i|Z_{i,j} = 1, \theta^{(t)})}{\sum_{j=1}^{m} Pr(X_i|Z_{i,j} = 1, \theta^{(t)})}.$$

For a ZOOPS model,

$$Z_{i,j}^{(t)} \quad = \quad \frac{f_j}{f_0 + \sum_{k=1}^{m} f_k}, \text{ where}$$

$$\begin{aligned} f_0 &= Pr(X_i|Q_i = 0, \theta^{(t)})(1 - \gamma^{(t)}), \text{ and} \\ f_j &= Pr(X_i|Z_{i,j} = 1, \theta^{(t)})\lambda^{(t)}, \ 1 \le j \le m. \end{aligned}$$

For a TCM model,

$$Z_{i,j}^{(t)} \quad = \quad \frac{Pr(X_{i,j}|\theta_1^{(t)})\lambda^{(t)}}{Pr(X_{i,j}|\theta_0^{(t)})(1 - \lambda^{(t)}) + Pr(X_{i,j}|\theta_1^{(t)})\lambda^{(t)}}.$$

III.C.3 The M-step

The M-step of EM in MEME reestimates θ using the following formula for models of all three types:

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k + \mathbf{d}(\mathbf{c}_k)}{|\mathbf{c}_k + \mathbf{d}(\mathbf{c}_k)|},\ 0 \le k \le W,\ \text{where}$$

$$\mathbf{c}_k \quad = \quad \begin{cases} \mathbf{t} - \sum_{j=1}^{W} \mathbf{c}_j & \text{if } k = 0,\ \text{OOPS or ZOOPS model} \\ \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{l=0}^{W-1} (1 - Z_{i,j}^{(t)}) \mathbf{I}(i, j+l) & \text{if } k = 0,\ \text{TCM model} \\ \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)} \mathbf{I}(i, j+k-1) & \text{otherwise.} \end{cases}$$

Here $\mathbf{d}(\mathbf{c}_k)$ is a function of the estimated letter counts $\mathbf{c}_k$ that yields a vector of pseudo-counts which is used to incorporate background information into EM as will be described later, $\mathbf{t}$ is the length-A vector of total counts of each letter the dataset, and $|\mathbf{x}|$ is the sum of the components of vector $\mathbf{x}$. For ZOOPS and TCM models, parameters γ and λ are reestimated during the M-step by the formula

$$\lambda^{(t+1)} = \frac{\gamma^{(t+1)}}{m} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)}.$$

III.D Finding multiple motifs

All three sequence model types supported by MEME model sequences containing a single motif (albeit a TCM model can describe sequences with multiple *occurrences* of the *same* motif). To find multiple, non-overlapping, different motifs in a single dataset, MEME uses greedy search. Motifs are found one at a time, with information about the motifs already discovered incorporated into the current model to avoid rediscovering the same motif.

The process of discovering one motif is called a pass of MEME. During pass i, MEME effectively uses a mixture model with $i + 1$ components but only updates the parameters for the components representing the background and the current motif. Model components representing the previously discovered motifs

are left unchanged during pass i and subsequent passes. Thus, the search greedily discovers a new motif on each pass.

The procedure for discovering multiple motifs used by MEME can be visualized as one of erasing the occurrences of motifs already discovered according to their probabilities given their motif models. It is implemented by changing the assumption that each position in the dataset is equally likely to be the start of a (new) motif occurrence and reestimating that probability based on the motifs found on previous passes. This is done in such a way that it is a natural extension of the EM algorithm.

III.D.1 Simulating multiple component mixture models

The three sequence model types used by MEME assume, *a priori*, that motif occurrences are equally likely at each position j in sequence X_i. This translates into a uniform prior probability distribution on the missing data variables $Z_{i,j}$. That is, initially, MEME assumes that $Pr(Z_{i,j} = 1) = \lambda$ for all $Z_{i,j}$.[2] On the second and subsequent passes, MEME changes this assumption to approximate a multiple-motif sequence model. A new prior on each $Z_{i,j}$ is used during the E-step that takes into account the probability that a new width-W motif occurrence starting at position $X_{i,j}$ might overlap occurrences of the motifs found on previous passes of MEME.

To help compute the new prior on $Z_{i,j}$ I introduce variables $V_{i,j}$ where $V_{i,j} = 1$ if a width-W motif occurrence could start at position j in sequence X_i without overlapping an occurrence of a motif found on a previous pass. Otherwise $V_{i,j} = 0$.

$$V_{i,j} = \begin{cases} 1, \text{ if no old motifs in } [X_{i,j}, \ldots, X_{i,j+W-1}] \\ 0, \text{ otherwise} \end{cases}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, L$.

[2] For an OOPS model, $\lambda = 1/m$. For a ZOOPS model, $\lambda = \gamma/m$.

To compute $V_{i,j}$ I use another set of binary variables $U_{i,j}$ which encode which positions in the dataset are *not* contained in occurrences of previously found motifs. So, $U_{i,j}$ is defined as

$$U_{i,j} = \begin{cases} 1, \text{ if } X_{i,j} \notin \text{ previous motif occurrence} \\ 0, \text{ otherwise} \end{cases}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Note that if we view V and U as sets of Boolean variables then

$$V_{i,j} = \bigwedge_{k=j}^{j+W-1} U_{i,k}. \tag{III.4}$$

As with the missing information variables $Z_{i,j}$, MEME computes and stores the *expected values* of the variables $U_{i,j}$. Before the first pass of MEME, the probability that $X_{i,j}$ is *not* already contained in a motif, the expected value of $U_{i,j}$, is set to one: $U_{i,j}^{(0)} = 1$ for $i = 1, \ldots, n$ and $j = 1, \ldots, L$. These values are updated after each pass according to the formula

$$U_{i,j}^{[p]} = U_{i,j}^{[p-1]}(1 - \max_{k=j-W+1,\ldots,j} Z_{i,k}^{[p]}) \tag{III.5}$$

where $Z_{i,j}^{[p]}$ is the *final* estimate of the missing information at the end of the current pass, p. Intuitively, we change the estimate of $X_{i,j}$ not being part of some motif by multiplying it by the probability of it not being contained in an occurrence of the current motif. This we estimate using the most probable motif occurrence of the current width that would overlap it. We use the maximum of $Z_{i,j}^{(t)}$ because occurrences of the current motif cannot overlap themselves, hence the values of $Z_{i,j}^{(t)}$ are not independent, so the upper bound on the probability used here is appropriate. The value of $U_{i,j}^{[p]}$ is then used as the value for $Pr(U_{i,j} = 1)$ in equation (III.6) below during the next pass, $p + 1$.

MEME estimates the probability of a width-W motif occurrence *not* overlapping an occurrence of *any* previous motif as the minimum of the probability of each position within the new motif occurrence *not* being part of an occurrence

found on a previous pass. In other words, MEME estimates $Pr(V_{i,j} = 1)$ as

$$Pr(V_{i,j} = 1) \quad = \quad \min_{k=j,\ldots,j+W-1} Pr(U_{i,k} = 1). \tag{III.6}$$

The minimum of $Pr(U_{i,k})$ is used because the probability of adjacent positions in sequence X_i not being contained in motif occurrences found on previous passes is clearly not independent.

MEME uses $\hat{Z}_{i,j}^{(t)}$ in place of $Z_{i,j}^{(t)}$ in the M-step of EM where

$$\hat{Z}_{i,j}^{(t)} \quad = \quad (\underset{(Z|X,\phi^{(t)})}{\mathrm{E}}[Z_{i,j}])\ Pr(V_{i,j} = 1).$$

This formula for reestimating $Z_{i,j}$ in the E-step of EM takes motifs found on previous passes into account and thus approximates a multiple-motif model. $\hat{Z}_{i,j}^{(t)}$ is an estimate of the probability that a motif occurrence starting at position j in X_i and no motifs found on previous passes have occurrences that would overlap it.

III.D.2 Soft versus hard erasing

The rationale for how the $U_{i,j}$ variables are updated is presented below. The difference between soft and hard erasing is discussed in the context of avoiding a problem with periodic motifs.

The variable $U_{i,j}^{[p-1]}$ stores the probability that position $X_{i,j}$ is *not* contained in any motif found on passes 1 through $p-1$. Let C_k be the event that $X_{i,j}$ *is* contained in a motif found on pass k. Then

$$U_{i,j}^{[p-1]} = Pr(\overline{C_1} \wedge \overline{C_2} \wedge \ldots \wedge \overline{C_{p-1}}).$$

After pass p, we want $U_{i,j}^{[p]}$ to be

$$U_{i,j}^{[p]} = Pr(\overline{C_1} \wedge \overline{C_2} \wedge \ldots \wedge \overline{C_{p-1}} \wedge \overline{C_p}).$$

If we assume that event C_p is independent from the other events C_i, we can write

$$U_{i,j}^{[p]} = U_{i,j}^{[p-1]} Pr(\overline{C_p}). \tag{III.7}$$

This assumption is not strictly justified, but multiplying probabilities causes them to approach zero, so any new evidence that $X_{i,j}$ is contained in the motif of pass p provided by $Pr(C_p)$ will tend to reduce our belief that $X_{i,j}$ is *not* contained in any motif.

To compute $Pr(\overline{C_p})$, let M_k be the event that a motif discovered on pass p starts at position $X_{i,k}$. Then the event C_p is the union of the series of events consisting of motifs starting to the left of or just at $X_{i,j}$:

$$C_p = M_{j-W+1} \vee M_{j-W+2} \vee \ldots \vee M_j.$$

Using deMorgan's law, the negation of this event is

$$\overline{C_p} = \overline{M_{j-W+1}} \wedge \overline{M_{j-W+2}} \wedge \ldots \wedge \overline{M_j}.$$

Since the probability of an intersection of events is always less than that of its least probable event, we can bound the probability of $X_{i,j}$ not being contained in any motif found on pass p by

$$Pr(\overline{C_p}) = Pr(\overline{M_{j-W+1}} \wedge \overline{M_{j-W+2}} \wedge \ldots \wedge \overline{M_j}) \leq \min_{k=j-W+1,\ldots,j} Pr(\overline{M_k}).$$

The value of $Z_{i,k}^{(t)}$ at convergence of EM on pass p, $Z_{i,k}^{[p]}$, gives an estimate of $Pr(M_k)$, so we can write

$$\begin{aligned} Pr(\overline{C_p}) &\leq \min_{k=j-W+1,\ldots,j} Pr(\overline{M_k}) \\ &= 1 - \max_{k=j-W+1,\ldots,j} Pr(M_k) \\ &= 1 - \max_{k=j-W+1,\ldots,j} Z_{i,k}^{[p]}. \end{aligned} \tag{III.8}$$

The updating of the values of U according to Equation III.7 can be thought of as "erasing" the occurrences of the motif just discovered. Since MEME uses the upper bound for $Pr(\overline{C_p})$ in computing $U_{i,j}^{[p]}$, it will tend to err by predicting that $X_{i,j}$ is not contained in a motif when it actually is. The erasing of motifs used by MEME is thus somewhat "soft". We could making the erasing "hard"

by multiplying probabilities instead of taking the maximum in equation (III.8), but experiments showed that this causes problems with certain types of motifs with periodic structure. For example, a motif representing the sequence pattern "AxxAxxAxxA" (where "x" means any letter) has a period of three. When such motifs match position $X_{i,j}$, they also tend to match positions $X_{i,j+s}$, $X_{i,j+2s}$, etc., where s is the period of the motif. Hard erasing tends to (erroneously) erase many positions which match weakly to such motifs because the weak matches are *not* independent. Soft erasing, which does not assume independence of the events M_k, prevents this.

III.E Using prior knowledge about motif columns

Applied to models of the forms described above, the EM method suffers from two problems. First, if any letter frequency parameter is ever estimated to be zero during EM, it remains zero. Second, if the dataset size is small, the maximum likelihood estimates of the letter frequency parameters tend to have high variance. Both these problems can be avoided by incorporating prior information about the possible values which the letter frequency parameters can take. The modified EM algorithm used by MEME actually performs Bayesian estimation as opposed to maximum likelihood estimation: it maximizes the mean posterior probability of the data assuming some prior distribution over the parameters of the model. As long as the prior distribution over the parameters of the model gives zero probability to any letter frequency parameter being equal to zero, the first problem is prevented. The second problem is reduced in severity because the influence of the prior on the posterior probability estimate increases as the size of the dataset decreases.

Using a mixture of Dirichlet densities as a prior in the estimation of the parameters of a model of biopolymer sequences has been proposed by Brown

et al. [1993]. This approach makes sense especially for proteins where many of the 20 letters in the sequence alphabet have similar chemical properties. Motif columns which give high probability to two (or more) letters representing similar amino acids are *a priori* more likely. A Dirichlet mixture density has the form $\rho = q_1\rho_1 + \ldots + q_R\rho_R$ where ρ_i is a Dirichlet probability density function with parameter $\beta^{(i)} = (\beta_a^{(i)}, \ldots, \beta_z^{(i)})$. A simple Dirichlet prior is the special case of a Dirichlet mixture prior where $R = 1$.

MEME uses Dirichlet mixture priors as follows. In the M-step, the mean posterior estimates of the parameter vectors $\mathbf{p}_i$, $i = 1$ to W, are computed instead of their maximum likelihood estimates. Let $\mathbf{c} = [c_a, \ldots, c_z]^T$ be the vector of expected counts of letters $a, \ldots, z$ in a particular column of the motif. We will consider this to be the "observed" letter counts in this column of the motif. The probability of component j in the Dirichlet mixture having generated the observed counts for this column is calculated using Bayes rule,

$$Pr(\beta^{(j)}|\mathbf{c}) = \frac{q_j Pr(\mathbf{c}|\beta^{(j)})}{\sum_{i=1}^{R} q_i Pr(\mathbf{c}|\beta^{(i)})}.$$

If we define $c = |\mathbf{c}| = \sum_{x\in\mathcal{L}} c_x$ and $b_j = |\beta^{(j)}| = \sum_{x\in\mathcal{L}} \beta_x^{(j)}$, then

$$Pr(\mathbf{c}|\beta^{(j)}) = \frac{\Gamma(c+1)\Gamma(b_j)}{\Gamma(c+b_j)} \prod_{x\in\mathcal{L}} \frac{\Gamma(c_x+b_j)}{\Gamma(b_j)}$$

where $\Gamma(\cdot)$ is the gamma function. We estimate the vector of pseudo-counts as a function of the observed counts as $\mathbf{d}(\mathbf{c}) = [d_a, d_b, \ldots, d_z]^T$ where

$$d_x = \sum_{j=1}^{R} Pr(\beta^{(j)}|\mathbf{c})\beta_x^{(j)},$$

for each $x \in \mathcal{L}$. The mean posterior estimate of the letter probabilities $\mathbf{p}_k$ in column k of the motif is then

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k + \mathbf{d}(\mathbf{c}_k)}{|\mathbf{c}_k + \mathbf{d}(\mathbf{c}_k)|}$$

for $k = 1$ to W. This gives the Bayes estimate of the letter probabilities for column k of the motif and is used to reestimate θ in the M-step.

j	1	2	3	4	5	6	7	8	9	10
q_j	55	198	43	60	65	67	80	51	103	62
b_j	5	0	1	2	3	2	2	3	1	8
A	85	56	14	45	36	19	42	54	315	86
C	22	32	8	5	4	7	15	0	38	6
D	11	69	7	169	13	1	4	144	11	60
E	20	38	8	75	46	3	5	460	13	107
F	50	40	255	7	5	84	19	2	11	15
G	25	143	11	84	20	4	5	19	107	37
H	15	42	50	36	30	4	1	10	7	32
I	132	22	23	6	16	152	307	9	15	23
K	24	44	12	54	307	4	5	56	11	113
L	150	66	56	11	33	456	115	16	23	41
M	57	11	15	3	13	106	28	7	11	19
N	20	44	19	212	36	3	5	24	22	62
P	16	83	8	20	12	5	4	17	30	26
Q	22	39	16	41	87	9	3	74	16	78
R	24	61	16	28	222	6	3	19	15	70
S	49	58	19	117	36	5	8	32	186	88
T	70	51	15	53	35	16	34	25	95	66
V	157	31	25	9	22	86	380	18	54	36
W	10	23	50	2	4	7	1	0	3	4
Y	30	36	364	13	12	15	6	4	7	19

j	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
q_j	12	4	13	8	8	32	5	7	9	17	2	5	22	5	11	8	19	7	3	4
b_j	38	381	90	404	83	32	35	101	150	31	201	108	15	79	93	47	14	76	150	30
A	732	5	8	2	8	23	10	10	6	11	3	6	41	14	7	28	24	44	5	34
C	14	959	1	0	4	2	3	2	1	3	1	2	3	2	3	6	7	7	2	8
D	6	1	859	12	2	11	13	4	2	2	1	14	11	5	2	7	8	5	3	12
E	9	2	36	963	3	8	10	4	3	5	1	4	15	32	3	6	7	8	3	17
F	7	1	2	0	850	2	12	8	1	19	2	3	5	2	1	5	6	7	18	156
G	28	1	6	1	3	868	7	3	3	3	1	6	12	4	4	9	7	5	4	11
H	3	0	4	1	1	3	722	1	2	3	1	14	6	25	7	3	4	4	2	43
I	12	3	4	1	13	3	5	681	4	40	9	7	9	5	4	6	17	68	4	20
K	8	1	4	1	3	8	6	5	903	6	2	7	16	10	27	10	12	7	4	16
L	18	3	5	1	35	4	15	77	3	790	21	6	16	16	6	9	12	25	17	36
M	8	1	2	0	7	2	6	18	1	31	928	2	4	9	2	4	10	8	3	12
N	6	1	29	2	2	12	36	4	4	3	1	858	9	10	5	19	23	5	3	13
P	3	1	2	0	3	4	6	2	2	5	1	3	755	6	4	7	6	6	2	10
Q	6	1	4	3	2	4	42	3	3	7	1	7	13	807	9	6	11	4	3	11
R	8	1	4	1	2	7	23	4	41	7	1	5	12	14	889	9	11	8	5	15
S	58	2	8	1	8	16	18	7	3	7	2	23	28	11	6	707	107	10	4	28
T	20	3	5	1	4	6	12	10	4	10	3	11	18	8	5	139	690	24	4	17
V	39	2	5	1	12	5	8	140	2	31	7	5	14	6	2	6	24	745	6	24
W	2	0	1	0	4	1	3	1	0	2	1	1	1	2	2	1	1	1	888	13
Y	4	0	2	0	25	2	33	4	1	5	3	3	3	2	2	4	3	2	12	494

Table III.1: **The 30-component Dirichlet mixture prior for proteins.**

Brown *et al.* [1993] have published several Dirichlet mixture densities that model well the underlying probability distribution of the letter frequencies observed in multiple alignments of protein sequences. The experiments reported in this thesis use either their 30-component Dirichlet mixture prior or a 1-component prior where $\beta^{(1)} = \boldsymbol{\mu}$ is the vector of average letter frequencies in the dataset. The (approximate) parameters of the 30-component Dirichlet mixture prior (multiplied by 1000 and rounded to integers) are given Table III.1.[3] Each column of the table contains the parameters of one component of the mixture. The component number j, mixing parameter q_j and magnitude b_j are shown above the mean $\beta^{(j)}/b_j$ of the component. The magnitude b_j gives an idea of the relative strength of the prior since the variance of a Dirichlet distribution [Santner and Duffy, 1989] with parameters $\beta^{(j)}$ is inversely proportional to b_j,

$$Var(\mathbf{p}) = \frac{(\beta^{(j)}/|\beta^{(j)}|)(1 - (\beta^{(j)}/|\beta^{(j)}|))}{|\beta^{(j)}|}.$$

The Dirichlet mixture distribution thus captures the idea that a column of a motif $\mathbf{p}$ is a random sample generated by a process which first selects a particular Dirichlet distribution j with probability q_j and then generates a probability vector $\mathbf{p}$ in a process with mean $\beta^{(j)}/b_j$ and variance given above.

It is easy to see that the right half of Table III.1 includes one Dirichlet prior that favors each of the twenty amino acids. This captures the knowledge that biologically relevant columns in motifs often tend to place the most weight on a single amino acid. These components have large values of b_j indicating that the variance of the component is small. The left side of the table contains ten other priors which capture the amino acid distributions of motif columns that correspond to other biological properties of proteins such as the ability to form secondary structures like alpha helices and beta sheets. These components have higher variance than the single amino acid components.

[3]The table shows only the approximate values of the mixture prior since precision has been lost in order to present the numbers compactly. The letters in the table amino acids according to the standard IUPAC one-letter code.

III.F Determining the right number of parameters

The number of free parameters in a model of any of the MEME sequence model types depends on the width of the motif and on whether or not the DNA palindrome constraints are in force. When the width of the motifs is not specified by the user and/or when MEME is asked to check for DNA palindromes, MEME chooses the number of free parameters to use by optimizing a heuristic function based on the maximum likelihood ratio test (LRT) [Seber, 1984], [Kendall *et al.*, 1983]. The optimum width of a motif depends on how many consecutive positions in the biopolymer sequences of a family are constrained by physical, chemical or biological considerations. The likelihood function cannot be used directly for comparing models with different motif widths, because its maximum value always increases with increasing W, as this adds more free parameters to the model. Likewise models of a given width with the palindrome constraints in force will have lower maximum likelihood values than unconstrained models.

The LRT is based upon the following fact [Kendall *et al.*, 1983]. Suppose we successively apply constraints $C_1, \ldots, C_s$ to a model with parameters ϕ and let $\phi_{(s)}$ be the maximum likelihood estimator of ϕ when all constraints $C_1, \ldots, C_s$ have been applied. Then, under certain conditions, the asymptotic distribution of the statistic

$$\chi^2 = 2 \log \frac{Pr(X|\phi)}{Pr(X|\phi_{(s)})}$$

is central χ^2 with degrees of freedom equal to the number of independent constraints upon parameters imposed by $C_1, \ldots, C_s$.

MEME uses the LRT in an unusual way to compute a measure of statistical significance for a single model by comparing it (and all other models of its type) to a "universal" null model. The null model is designed to be the simplest possible model of a given type. Let ϕ be the parameters of a model discovered

by MEME using EM. Then, ϕ is the maximum likelihood estimate (MLE) for the parameters of the model.[4] Likewise, let ϕ_0 be the maximum likelihood estimate for the parameters of the null model. Since both ϕ and ϕ_0 are maximum likelihood estimates, the LRT can be applied to these two models. At some significance level between 0 and 1, the LRT would reject the null model in favor of the more complicated model. We define $LRT(\phi)$ to be this significance level, so

$$LRT(\phi) = Q(\chi^2|\nu), \text{ where}$$
$$Q(\chi^2|\nu) \approx Q(x_2), \quad x_2 = \frac{(\chi^2/\nu)^{1/3} - (1 - \frac{2}{9\nu})}{\sqrt{2/(9\nu)}}$$

[Abramowitz and Stegun, 1972]. $Q(x_2)$ is the Q function for the standard normal distribution (i.e., size of the right tail), and ν is the difference between the number of free parameters in the model used with EM and the null model. There are $A-1$ free parameters per column of θ, so the difference in free parameters is $\nu = W(A-1)$ for all three model types. If the DNA palindrome constraints are in force, half the parameters in θ_1 are no longer free and $\nu = (W/2)(A-1)$.

To compute the value of $LRT(\phi)$ we need values of the likelihood functions for the given and null models and the difference in the number of free parameters between them. For the likelihood of the given model, MEME uses the value of the joint likelihood function maximized by EM. For the null model, it is easy to show that the maximum likelihood estimate has all columns describing motif and background positions equal to $\boldsymbol{\mu}$ where $\boldsymbol{\mu} = [\mu_a, \ldots, \mu_z]^T$ is the vector of average letter frequencies in the dataset. The log likelihood of the null model is

$$\log Pr(X|\phi_0) \quad = \quad nL \sum_{x \in \mathcal{L}} \mu_x \log \mu_x.$$

The criterion function which MEME minimizes is

$$G(\phi) = LRT(\phi)^{1/\nu}.$$

[4] I overlook the possibility that EM converged to a local maximum of the likelihood function. Note also that ϕ is actually the mean posterior estimate of the parameters, not the MLE, when a prior is used. In practice, the value of the likelihood function at ϕ is close to the value at the MLE.

This criterion is related to the Bonferroni heuristic [Seber, 1984] for correcting significance levels when multiple hypotheses are tested together. Suppose we only want to accept the hypothesis that ϕ is superior if it is superior to every model with fewer degrees of freedom. There are ν such models so the Bonferroni adjustment heuristic suggests to replace $LRT(\phi)$ by $LRT(\phi)\nu$. The function $G(\cdot)$ applies a much higher penalty for additional free parameters and yields motif widths much closer to those chosen by human experts than either $LRT(\phi)$ or $LRT(\phi)\nu$.

III.G Searching the space of models

The goal of MEME is to find the model of a given type which minimizes the objective function $G(\phi)$ for the current dataset and taking into consideration any motifs already found. If the width of the motif is not fixed, MEME searches for the best model of each of a series of different widths and chooses the one with the minimum value of $G(\phi)$. For each width W_i that MEME considers, if the model type is not OOPS so the mixing parameter λ is not determined, MEME considers a series of values of λ and chooses a value of θ for each (λ, W) pair. If λ is fixed, only one value of θ is chosen. The EM algorithm is run to convergence from each starting point triple $\phi^{(0)} = (\theta, \lambda, W)$. The resulting model is then checked to see if any flanking columns should be removed or if the palindrome constraints should be applied in order to further minimize $G(\phi)$. After this has been done with all chosen values of $\phi^{(0)}$ as starting points for EM, the best model is chosen and its motif component is output. The following sections describe how a good value of θ is chosen given values of W and λ and how the removing of unnecessary flanking motif columns and application of the palindrome constraints is done.

III.G.1 Mining the dataset for EM starting points

MEME uses a dynamic programming algorithm based on a single EM iteration to simultaneously choose a good initial value for θ for each of several initial values of λ.[5] The algorithm for finding good starting points for EM evolved from a straightforward multi-start approach. Multi-start with EM means running EM to convergence from a number of different starting points and choosing the model with the highest likelihood as the final model. The most obvious way to choose the starting points is to randomly or systematically sample from the space of (θ, λ, W) triples. MEME samples systematically over λ, but uses information in the dataset to provide good candidate values for θ. This is done by generating candidate values of θ by mapping each width-W subsequence in the dataset, in turn, to a θ matrix. Some of these subsequences will be the actual motif occurrences and the mapping function described below insures that the corresponding θ values are likely to be good starting points. Because EM tends to converge quickly from good starting points, the likelihood of the model after one iteration of EM turns out to be an excellent predictor of starting point goodness. MEME scores each potential starting point for EM using an algorithm that estimates the what the likelihood of the model would be after one iteration of EM. This scoring algorithm is optimized to simultaneously compute scores for any number of (θ, λ, W) triples for given values of θ and W. Additional speed is achieved through the use of dynamic programming techniques. These reuse the computations done in scoring the starting points generated by the subsequence starting at position $X_{i,j}$ in the dataset when scoring the starting points generated by the next overlapping subsequence starting at position $X_{i,j+1}$.

We would like to find good initial values for θ to use as starting points for EM. Rather than using random or systematic sampling from the space Θ of possible values for θ, MEME uses the average frequencies $\boldsymbol{\mu}$ of letters in the dataset

[5] The goodness of an initial (θ, λ, W) triple is how likely EM is to converge from it to the globally optimal model.

as the initial estimate for the background component of the model θ_0, and estimates the initial value of the motif component θ_1 by assuming that each subsequence of length W in the dataset, in turn, is an occurrence of the motif. This is identical in principle to the overall Bayesian approach MEME uses to discover motifs. If a motif occurrence starts at position j in sequence X_i of the dataset, then the mean posterior estimate of the motif component of the model θ_1 *based on this sample of size one* is the same as in the M-step of EM,

$$\mathbf{p}_k = \frac{\mathbf{c}_k + \mathbf{d}(\mathbf{c}_k)}{|\mathbf{c}_k + \mathbf{d}(\mathbf{c}_k)|}, \ 1 \leq k \leq W,$$

where $\mathbf{c}_k = \mathbf{I}(i, j+k-1)$ is the vector of "observed" letter counts. Since all the entries in $\mathbf{c}_k$ are zero except the one corresponding to the letter in position k of the string starting at $X_{i,j}$ in the dataset, there are only $A = |\mathcal{L}|$ possible values for $\mathbf{c}_k$—one for each letter in the alphabet. So only A values of $c + \mathbf{d}(c)$ will ever be needed for any choice of prior on $\mathbf{p}$. MEME computes these and stores them as an $A \times A$ sequence-to-theta mapping table

$$\mathbf{p}^{(0)} = [\mathbf{p}_{\mathbf{a}}^{(0)} \ \mathbf{p}_{\mathbf{b}}^{(0)} \ \cdots \ \mathbf{p}_{\mathbf{z}}^{(0)}]$$

where $\mathbf{p}_{\mathbf{x}}^{(0)}$ is the initial estimate of $\mathbf{p}$ to use for a column of the motif when the observed letter is $x \in \mathcal{L}$. As will be described below, MEME allows the user to specify which the size and type of prior to be used in computing $\mathbf{p}^{(0)}$.

To illustrate using the standard DNA alphabet $\mathcal{L} = \{a, c, g, t\}$, suppose the string starting at $X_{i,j}$ is "tgtcat". If the uniform Dirichlet prior with $\beta = [1111]^T$ is used, then the sequence-to-theta mapping table is

$$\mathbf{p}^{(0)} = [\mathbf{p}_{\mathbf{a}}^{(0)} \mathbf{p}_{\mathbf{c}}^{(0)} \mathbf{p}_{\mathbf{g}}^{(0)} \mathbf{p}_{\mathbf{t}}^{(0)}] \ = \ \begin{bmatrix} 2/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 2/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 2/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 2/5 \end{bmatrix},$$

and the initial estimate for the motif component of the sequence model is

$$\theta_1 = [\mathbf{p_t^{(0)}} \mathbf{p_g^{(0)}} \mathbf{p_t^{(0)}} \mathbf{p_c^{(0)}} \mathbf{p_a^{(0)}} \mathbf{p_t^{(0)}}] = \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 2/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 2/5 & 1/5 & 1/5 \\ 1/5 & 2/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 2/5 & 1/5 & 2/5 & 1/5 & 1/5 & 2/5 \end{bmatrix}.$$

Observe that column k of the θ_1 matrix is just the column from the mapping table corresponding to the letter at position $X_{i,j+k-1}$.

MEME supports sequence-to-theta mapping tables based on a uniform Dirichlet prior or based on a prior that incorporates knowledge about likely motif columns. The user can also choose the "size" of the prior which determines the "fuzziness" of the initial estimate of θ_1. With the uniform Dirichlet prior $\beta = [s\ s\ \ldots\ s]^T$, the size of the prior is determined by s. Experiments reported in this thesis which use a sequence-to-theta mapping table based on the uniform prior use $s = 0.52$ for DNA datasets and $s = 0.15$ for protein datasets. The other type of sequence-to-theta mapping table supported by MEME is called a mutation probability matrices (MPA) [Dayhoff *et al.*, 1983]. Column x of an MPA matrix gives the estimated probability of the amino- or nucleic-acid represented by letter x having mutated to each other amino- or nucleic-acid letter in the alphabet (after some specified evolutionary distance) based on statistics gathered from biological databases. Evolutionary distance is measured in terms of percent accepted mutation (PAM) units. One PAM is the distance between two related sequences such that 1% of the positions in the sequences are different. The user can specify the size of the prior by specifying how many PAM units it should represent. Larger PAM values yield "fuzzier" mapping matrices. Starting from the 1-PAM MPA matrix M, the n-PAM MPA matrix can be computed by raising the 1-PAM MPA matrix M to the n-th power, i.e. $M^n = M \times M \times \ldots \times M$. Experiments reported here using MPA sequence-to-theta mapping matrices typically use the 120-PAM MPA matrix shown in Table III.G.1. Sequence-to-theta mapping tables can easily

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	265	32	73	82	22	111	34	58	37	36	46	72	112	58	33	127	130	85	8	20
C	11	726	3	3	4	4	8	10	3	2	3	6	11	3	9	26	12	15	1	23
D	41	5	257	154	4	43	40	15	33	7	10	107	22	59	14	41	32	15	3	7
E	48	5	163	269	5	36	38	20	33	13	16	64	30	116	18	36	29	20	2	9
F	10	5	4	4	550	9	17	43	4	45	30	11	4	5	9	13	12	14	27	180
G	112	19	78	64	13	490	25	22	30	16	21	77	47	38	22	107	57	43	5	8
H	13	8	29	25	16	7	370	7	20	10	8	59	23	80	44	17	14	9	10	26
I	22	15	11	13	38	8	9	261	16	55	60	16	9	12	14	15	34	118	3	15
K	35	8	59	56	8	28	46	34	453	19	92	102	36	77	188	60	67	22	14	13
L	31	6	12	18	96	15	34	134	26	560	218	25	27	40	19	21	34	107	34	32
M	7	1	3	4	9	2	4	25	18	38	233	5	3	10	9	7	11	21	2	2
N	34	8	92	52	11	37	70	18	52	10	14	162	24	37	25	56	43	15	9	20
P	65	14	22	30	11	28	36	17	23	18	17	29	430	47	36	60	41	23	5	6
Q	24	3	48	89	5	15	92	13	37	19	23	35	34	260	47	21	19	13	4	5
R	15	10	12	15	9	7	55	19	96	11	29	25	29	51	379	30	19	12	53	5
S	102	57	60	51	21	83	35	30	52	18	31	96	84	41	54	208	122	35	29	20
T	88	19	40	33	15	38	22	54	49	23	39	62	49	30	30	103	258	54	7	17
V	61	26	19	23	26	28	22	205	19	79	92	23	29	23	20	32	59	361	4	19
W	1	0	0	0	8	0	1	0	1	0	0	1	1	1	13	5	1	0	750	9
Y	8	23	5	7	135	3	25	13	3	13	7	15	3	5	3	9	9	9	18	540

Table III.2: **The 120-PAM MPA matrix for proteins.** All entries have been multiplied by 1000. The letters are the standard one-letter codes for amino acids.

be generated from any prior, including Dirichlet mixture priors, but this is not currently implemented in MEME since it would probably perform no better than MPA tables.

III.G.2 Simultaneously testing multiple starting points

The following algorithm TEST is used by MEME to test all of the possible starting points (θ, λ) generated by mapping subsequences in the dataset to values of θ_1, using the overall letter frequencies as θ_0, and a fixed value of λ for an OOPS model or a geometric series of λ values for a ZOOPS or a TCM model. TEST is motivated by a single iteration of EM but is much faster because it simplifies the E- and M-steps, uses dynamic programming in the E-step, and simultaneously performs the E-step for any number of values of λ. For a given value of θ_1, TEST

```
procedure TEST(W, list of λ values, model type, dataset)
  Set θ_0 = μ , average letter frequencies in dataset.
  for each sequence X_k in dataset do
    for each width-W subsequence starting at position l in X_k do
      Map subsequence X_{k,l} to θ_1.
      for each sequence X_i in dataset do
        for each width-W subsequence starting at position j in X_i do
          Compute Pr(X_{i,j}|θ_1).
        end
      end
      if (model type is OOPS or ZOOPS)
        Make list of single subsequence in each sequence
        with maximum value of Pr(X_{i,j}|θ_1).
      else
        Make list of non-overlapping subsequences with locally
        maximum value of Pr(X_{i,j}|θ_1).
      end
      if (model type is ZOOPS or TCM)
        Sort positions of maximum Pr(X_{i,j}|θ_1).
      end
      for each value of λ in list do
        Calculate c_k, 1 ≤ k ≤ W, for top nmλ subsequences in list.
        Estimate θ_1 after one pass of EM as θ̂_1 = [c_1 c_2 ... c_W].
        Compute likelihood of model (θ_0, θ̂_1, λ).
        Save θ_1 if likelihood of (θ_0, θ̂_1, λ) best for this value of λ.
      end
    end
  end
end
```

Figure III.1: **The TEST algorithm.**

finds the most likely positions for motif occurrences in the dataset (subject to the constraints imposed by the type of model such as one occurrence per sequence). It sorts these positions according to their likelihood given θ_1 if more than one value of λ is to be tested. Then it computes the observed letter frequences in each column of the motif given the most likely $nm\lambda_i$ subsequences in the dataset for each λ_i to be tested. These observed frequencies are used as the estimate of θ_1 after one iteration of EM and the likelihood of the new model is used as the score of the potential starting point. The TEST algorithm is sketched in Figure III.1.

Approximating EM. The main differences between TEST and a single iteration of EM are that it approximates the expected value of the missing information with $Pr(X_{i,j}|\theta_1)$ and uses the positions with the maximum likelihood of being motif occurrences to compute the observed letter counts rather than taking the expectation of the joint likelihood of the sequences and the missing information. This works well in practice because $Pr(X_{i,j}|\theta_1)$ tends to reach its maxima at motif occurrences. Using these positions as the observed data gives a good approximation of the letter counts which EM would estimate after one iteration. As a result, the new estimate of θ_1 tends to be close to what EM would find when the initial θ_1 is a good starting point. This causes the likelihood of the new model to be close to that which EM would discover.

Testing several values of λ at once. When several starting values of λ are to be tested for a ZOOPS or TCM model, TEST sorts the subsequences $X_{i,j}$ in the dataset by $Pr(X_{i,j}|\theta_1)$. For a ZOOPS model, only a single subsequence with maximal $Pr(X_{i,j}|\theta_1)$ from each sequence X_i is put in the list. For a TCM model, each sequence may contribute more than one motif occurrence, so the non-overlapping subsequences with locally-maximal $Pr(X_{i,j}|\theta_1)$ are all put in the list before it is sorted. Calculating the observed counts $\mathbf{c}$ from this list is extremely efficient since the counts are always computed for increasingly larger values of λ. If $(\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(r)})$ is the list of λ values to be tested in increasing order, then

the observed letter counts for $\lambda^{(1)}$ are computed by summing over the letters in the first $nm\lambda^{(1)}$ subsequences in the sorted subsequence list. (These subsequences are the most likely motif occurrences.) The counts for the next larger value of λ involve the top $nm\lambda^{(2)}$ subsequences, so the letters in the next $nm(\lambda^{(2)} - \lambda^{(1)})$ subsequences must be added to the previous counts. This procedure can continue for all values of λ in the list, optimizing the computation of the observed counts for any number of λ values.

Dynamic programming. TEST uses dynamic programming to optimize the calculation of $Pr(X_{i,j}|\theta_1)$. Let $\theta_1^{(k,l,w)}$ be the value of θ_1 gotten from the length-w subsequence at position $X_{k,l}$ in the dataset. TEST reuses the computations for $Pr(X_{i,j}|\theta_1^{(k,l,W)})$ when calculating $Pr(X_{i,j+1}|\theta_1^{(k,l+1,W)})$, saving a large amount of computation. The recursion relation used is

$$Pr(X_{i,j+1}|\theta_1^{(k,l+1,W)}) = \frac{Pr(X_{i,j}|\theta_1^{(k,l,W)})Pr(X_{i,j+W}|\theta_1^{(k,l+W,1)})}{Pr(X_{i,j}|\theta_1^{(k,l,1)})} \qquad \text{(III.9)}$$

This calculation takes only two floating point operations rather than the $(W-1)$ which would be required to compute $Pr(X_{i,j+1}|\theta_1^{(k,l+1,W)})$ by multiplying together the W probabilities in $\theta_1^{(k,l+1,W)})$ corresponding to the letters in the length-W subsequence at position $X_{i,j}$. The recursion works because $\theta_1^{(k,l+1,W)}$ is a shifted version of $Pr(X_{i,j}|\theta_1^{(k,l,W)}$ with new column $\theta_1^{(k,l+W,1)}$ on the right and column $\theta_1^{(k,l,1)}$ removed on the left. So the same probabilities are selected for each letter when $Pr(X_{i,j+1}|\theta_1^{(k,l+1,W)})$ is calculated as when $Pr(X_{i,j}|\theta_1^{(k,l,W)})$ was calculated except for the two letters on either end.

To illustrate, the initial θ matrices derived from consecutive subsequences starting at $X_{k,l}$ and $X_{k,l+1}$ are composed of column vectors corresponding to the letters in X_k as shown below.

$$
\begin{array}{rcccccccccccccc}
 & & & & & X_{k,l} & & & & & & X_{k,l+W} & & & \\
 & & & & & \downarrow & & & & & & \downarrow & & & \\
X_k & = & . & . & . & \mathrm{a} & \mathrm{c} & \mathrm{t} & \mathrm{g} & \mathrm{t} & \mathrm{a} & \mathrm{a} & \mathrm{t} & . & . & . \\
\theta_1^{(k,l,W)} & = & & & & [\mathbf{p_a^{(0)}} & \mathbf{p_c^{(0)}} & \mathbf{p_t^{(0)}} & \mathbf{p_g^{(0)}}] & & & & & & & \\
\theta_1^{(k,l+1,W)} & = & & & & & [\mathbf{p_c^{(0)}} & \mathbf{p_t^{(0)}} & \mathbf{p_g^{(0)}} & \mathbf{p_t^{(0)}}] & & & & & &
\end{array}
$$

The probabilities of a subsequence starting at position i or $i+1$, respectively, in any sequence X_i given the two initial values of θ shown above are related as shown below. X_i given each of these values of θ_1 can then be visualized as

$$
\begin{array}{rcccccccccccccc}
 & & & & & & X_{i,j} & & & & & X_{i,j+W} & & & \\
 & & & & & & \downarrow & & & & & \downarrow & & & \\
X_i & = & . & . & . & \mathrm{t} & \mathrm{a} & \mathrm{t} & \mathrm{a} & \mathrm{c} & \mathrm{c} & \mathrm{t} & . & . & . \\
Pr(X_{i,j}|\theta_1^{(k,l,W)}) & = & & & & \mathbf{p_{a,t}^{(0)}} & \mathbf{p_{c,a}^{(0)}} & \mathbf{p_{t,t}^{(0)}} & \mathbf{p_{g,a}^{(0)}} & & & & & & \\
Pr(X_{i,j+1}|\theta_1^{(k,l+1,W)}) & = & & & & & \mathbf{p_{c,a}^{(0)}} & \mathbf{p_{t,t}^{(0)}} & \mathbf{p_{g,a}^{(0)}} & \mathbf{p_{t,c}^{(0)}} & & & & & &
\end{array}
$$

where $\mathbf{p_{a,x}^{(0)}}$ means the entry in the $\mathbf{p_a^{(0)}}$ vector corresponding to letter x. Hence the value of $Pr(X_{i,j+1}|\theta_1^{(k,l+1,W)})$ can be calculated recursively from $Pr(X_{i,j}|\theta_1^{(k,l,W)})$ by dividing by $\mathbf{p_{a,x0}^{(0)}}$ and multiplying by $\mathbf{p_{t,x1}^{(0)}}$ where $x0 = X_{i,j}$ and $x1 = X_{i,j+W}$, as shown in the recursion formula (III.9). Algorithm TEST computes $Pr(X_{i,j}|\theta_1^{(k,0,W)})$ directly each time it enters its second **for** loop. On successive passes through that loop it uses the recursion relation shown above to save time.

Avoiding round-off errors. To avoid round-off errors, the recursion is done with integer arithmetic and logarithms. All of the values involved in computing the scores as well as the scores themselves are always between 0 and 1. So their logarithms lie between 0 and minus infinity. To perform integer logarithm arithmetic, each value $0 \leq x \leq 1$ is converted to a scaled logarithm using the formula

$$i_x = \lceil a log_2(x + \epsilon) \rceil.$$

This formula with appropriate choice of a and ϵ yields integral values between

0 and the machine-dependent smallest integer value. For 32-bit machines, good choices are $\epsilon = 10^{-200}$ and $a = 10^3$. For $x \geq 2^{(-1/a)} = .999307$, $i_x = 0$. For $x = 0$, $i_x = -664385$, well within the range of 32-bit integers. Thus, probabilities above .999307 are rounded up to 1, and the (approximate) range $[10^{-200}, .999]$ is divided into 665385 parts each separated by a factor of $2^{1/1000} = 1.00069$ from their neighbors. This scaling provides enough precision and dynamic range to capture the range of interesting probability values. Since all calculations of S are done using the integer values i_x, no *cumulative* roundoff errors occur as a result of doing the calculations recursively.

III.G.3 Shortening the motif

The strategy for sampling W only tests a subset of possible motif widths. To allow MEME to discover motifs of intermediate widths (without sampling all possible widths), an algorithm which shortens the motif component of a model is employed. I call this algorithm SHORTEN. The idea behind shortening motifs is that fitting a model that is a little bit too wide to a dataset will tend to find the same motif occurrences as would the model of optimum width. However, some outer columns in the motif will not be important, and this should be detectable using likelihood ratio test. After the superfluous columns of the motif have been removed, rerunning EM on the remaining columns (plus the rest of the model, i.e., ${\theta^{(t)}}_0$, $\lambda^{(t)}$ and the new value of W) may increase the chances of finding the optimum model.

The SHORTEN algorithm applies the LRT-based heuristic function $G(\phi)$ to the model gotten by removing some of the outer columns of the motif matrix θ_1. The is done just as though EM had been run to convergence with a model of the reduced width and yielded the value of $\theta^{(t)}$ gotten by just removing certain outer columns of the matrix. The the value of $G(\phi)$ is calculated for all possible versions of the motif gotten by removing outer columns such that the shortened width is at

```
procedure SHORTEN ( φ^(t) = (θ^(t), λ^(t), W) )
  for W' = ⌊W/√2⌋ to W do
    for i = 1 to W − W + 1' DO
      Create new model φ' by removing all columns except
      [p_i, ..., p_{i+W'−1}] from θ_1^(t).
      Calculate S = G(φ').
      Save φ' if S is best so far.
      If DNA, create new model φ'_pal from φ' by
      applying palindrome constraints.
      Calculate S = G(φ'_pal).
      Save φ'_pal if S is best so far.
    end
  end
  Run EM to convergence from starting point φ'.
  Return result of EM: the new model φ'^(t).
end
```

Figure III.2: **The** SHORTEN **algorithm.**

least $W/\sqrt{2}$, where W is the width of the motif before shortening. The model with the minimal value of G returned after after rerunning the EM algorithm using the shortened model as starting point.

The SHORTEN algorithm is sketched in Figure III.2. In the inner loop, a subset of the columns of the motif matrix $\theta_1^{(t)}$ are selected, and a new model ϕ' is created. This involves adjusting λ since its value depends on the number of possible motif occurrence starts in the dataset, which in turn depends on W. W is set to its new, smaller value W' and the value of the LRT-based heuristic function of the resulting model is computed. If the dataset consists of DNA sequences and the user has requested that palindromes be searched for, the DNA palindrome constraints are applied to the columns of ϕ' to create ϕ'_{pal} and the value of the LRT-based heuristic function of the resulting model is computed. The best model

```
procedure MEME ( X: dataset of sequences )
  for pass = 1 to pass_max do
    for W = W_min to W_max by ×√2 do
      for λ^(0) = λ_min to λ_max by ×2 do
        Choose good θ^(0) given W and λ^(0).
        Run EM to convergence from chosen
        value of φ^(0) = (θ^(0), λ^(0), W).
        Remove outer columns of motif
        and/or apply palindrome constraints
        to maximize G(φ).
      end
    end
    Report model which maximizes G(φ).
    Update prior probabilities U_{i,j} to
    approximate multiple-motif model.
  end
end
```

Figure III.3: **The** MEME **algorithm.**

derived in these ways from the original model ϕ is kept at each step. Finally, SHORTEN runs EM is using the best (shortened) model as a starting point, with palindrome constraints enforced if the best model found is a DNA palindrome.

III.H The MEME algorithm

This section describes the complete MEME algorithm and analyzes its time complexity.

III.H.1 Algorithm sketch

The complete MEME algorithm is sketched in Figure III.3. The type of sequence model to use, the number of passes and the maximum and minimum values of motif widths to try are input by the user. The minimum and maximum

model	λ_{min}	λ_{max}
OOPS	$\frac{1}{m}$	$\frac{1}{m}$
ZOOPS	$\frac{\sqrt{n}}{nm}$	$\frac{1}{m}$
TCM	$\frac{\sqrt{n}}{nm}$	$\frac{1}{2w}$

Table III.3: **The minimum and maximum values of $\lambda^{(0)}$ tried as starting points for EM.**

values for λ are determined by the data and the type of sequence model specified.

The spacing factor of $\sqrt{2}$ for widths to be tried by MEME was determined by experimentation. Using a larger spacing between widths such as a factor of 2 seemed to reduce the performance of MEME whereas smaller spacings did not appear to make a difference. The spacing factor of 2 for values of λ to try with ZOOPS and TCM models was also determined by experimentation to give a good balance between execution speed and quality of motifs found. (Results not shown.)

If the model type being used is OOPS, the inner loop is iterated only once since λ is not relevant. For a ZOOPS model, $\lambda_{min} = 1/(m\sqrt{n})$ and $\lambda_{max} = 1/m$. For a TCM model, $\lambda_{min} = 1/(m\sqrt{n})$ and $\lambda_{max} = 1/(W+1)$.Since there are n sequences, these values of λ correspond to there being on average at least one motif occurrence for every $\sqrt{n}$-th sequence, and at most one occurrence per sequence in a ZOOPS model, and at most half of the total positions in the dataset being part of motif occurrences in a TCM model. The minimum and maximum values which are tried as starting values for λ, λ_{min} and λ_{max} are are summarized in table III.H.1.

Sample output from the MEME algorithm is given in Appendix A.

III.H.2 Time and space complexity of MEME

The space of MEME is linear in the size of the dataset since the only storage required is for the data itself and various arrays (Z, V, U) which have the

same size and shape as the dataset. The storage required for the sequence model is constant for a given motif width and is usually much less than that for the dataset.

The theoretical time complexity of a single pass of MEME is a combination of the time complexity of the algorithm which chooses good starting points for EM, TEST, and that of the EM algorithm itself. Depending on the type of sequence model being used, EM may be repeated a number of times with different values of the mixing parameter λ. If MEME is searching for the best motif width, this increases the time complexity further. I discuss the contribution of these factors to the time complexity of MEME in what follows and show that the time complexity of MEME will tend to be quadratic in the size of the dataset.

The time complexity of a single iteration of EM is dominated by the E-step since that step must update the Z matrix which has size proportional to the dataset. The amount of work done is proportional to the width of the motif times the size of the dataset which is $O(WnL)$. The M-step only requires work proportional to the size of the motif matrix θ, which is typically much smaller than the dataset. If EM converges quickly, then its time complexity will be that of the E-step or $O(WnL)$. This is usually the case in practice.

The total time required for computing probabilities using dynamic programming in algorithm TEST is proportional to

$$\begin{aligned} T(W) &= n(\text{base case} + \text{recursive part}) \\ &= O(n(n(L-W+1)(W-1) + (L-W)(L-W+1)2)) \\ &= O(n^2(L-W+1)(W-1) + 2n(L-W)(L-W+1)), \end{aligned}$$

so the time complexity of TEST is at worst quadratic in the size of the dataset. If n is large compared to L, then the first term dominates and the time complexity is approximately $O(n^2L^2)$ when $W = L/2$. The sensitivity of the theoretical time complexity of TEST to the two parameters n and L is plotted in Figure III.4. The approximately quadratic increase in execution time can be seen in parts A and B of the figure, where the number of sequences and sequence length, respectively, are

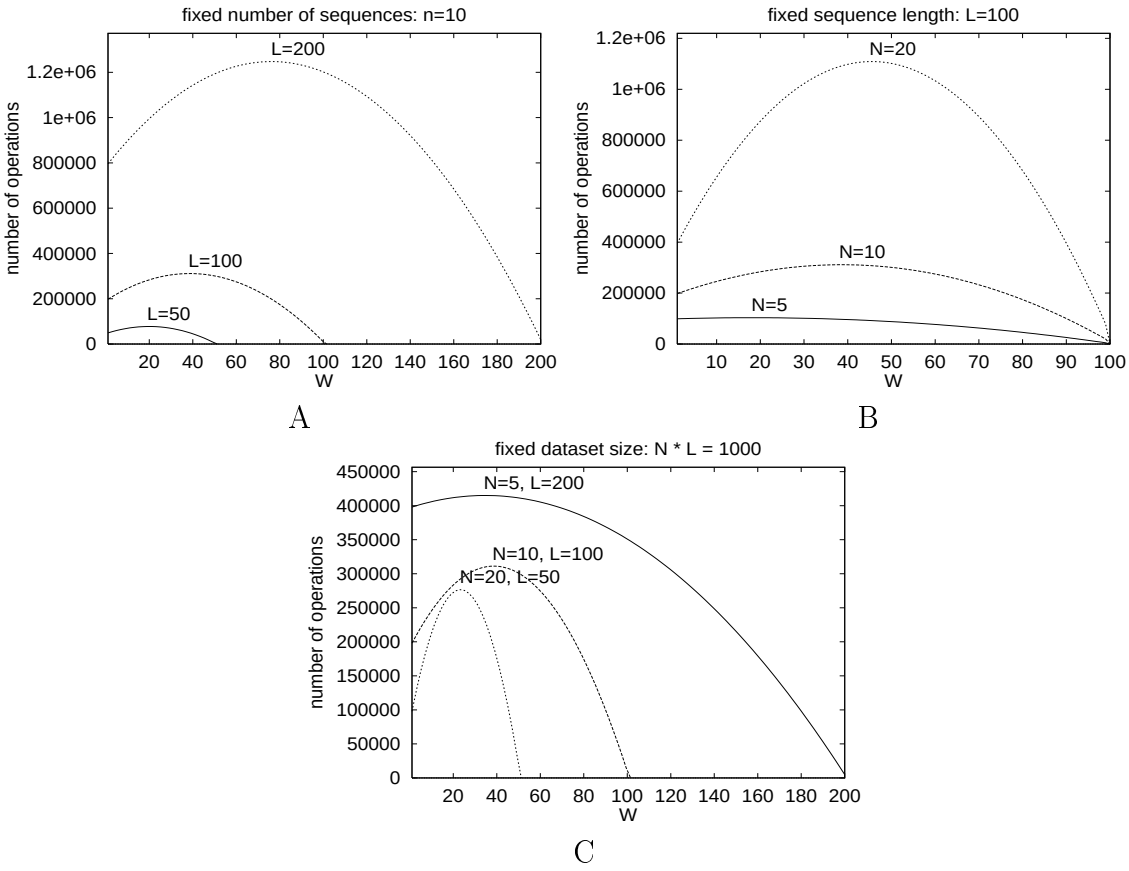


Figure III.4: **Time complexity of the search for starting points for EM.** This figure plots the time complexity of the TEST algorithm for searching for good starting values of θ_1. Part A shows the theoretical time complexity as a function of W for datasets with a fixed number of sequences ($n = 10$) as the length of the sequences is varied. Part B shows the theoretical time complexity as a function of W for datasets with a fixed number of sequences ($L = 100$) as the length of the sequences is varied. Part C shows the time complexity for a fixed dataset size ($nL = 1000$). It can be seen from the first two figures that if L or n is held fixed, the worst case time complexity is aproximately quadratic in dataset size. The third figure shows that, for a given total size, maximum execution time increases with sequence length.

kept constant. Part C shows that datasets with fewer, longer sequences will take longer than datasets with more, shorter sequences. I expect that the length of the sequences will usually exceed the number of sequences, and when this happens the second term will tend to dominate and the time complexity will be $O(nL^2)$. In practice, as long as W is not too large the time complexity of the TEST algorithm will dominate that of EM and the time complexity of MEME will approach $O(n^2L^2)$, i.e. be quadratic in the size of the dataset.

Searching over different values of λ and W does not affect the order of MEME's time complexity. A geometric series of W values is tried by MEME causing TEST and EM to be repeated at most $2 \log_2 W_{max}$ where, in practice, $W_{max} \leq 100$. This represents a constant increase in the time required by MEME. Testing multiple values of λ with ZOOPS and TCM models requires extra work in the TEST algorithm. As mentioned above, a geometric series of initial values of λ are considered. For a ZOOPS model of a given width, MEME finds and runs EM to convergence from $\log_2 \sqrt{n}$ starting points. For a TCM model, $\log_2 ((L - W + 1)(W + 1)\sqrt{n})$ starting points are found and EM is run to convergence from each of them. In the worst case this adds a factor of only $\log_2(L^2\sqrt{n})$ so the time complexity of MEME is essentially no worse than quadratic in the size of the dataset. Experiments reported in Chapter V show that quadratic behavior is the case in practice.

III.I Using motifs

This section explains how the motifs discovered by MEME can be used as classifiers

- to search for sequences containing occurrences of one or more motifs, which thus may be related to the training set family, and
- to annotate sequences with the positions where they contain occurrences of one or more motifs, and

- to display proteins as block structure diagrams illustrating the order and spacing of the motifs they contain.

Displaying motif occurrences in these ways is useful to biologists for visualizing the relationships among sequences. Searching for occurrences of more than one motif in a single sequence provides a more sensitive way of looking for distantly related protein sequences.

I developed two tools for performing these tasks, PROBER and NOTE which are discussed below following an explanation of how motifs can be used as classifiers.

III.I.1 Motifs as classifiers

A classifier with the form of a Bayes-optimal classifier [Duda and Hart, 1973] can be made from any of the motif models learned by MEME. Such a classifier requires assuming prior distributions for the motif and a loss function which assigns a cost for making each of the four possible types of decisions.[6] For the "zero-one" loss function, which assigns a loss of 0 for correct classification and 1 for incorrect, the Bayes-optimal decision rule says to decide that a subsequence x is a motif occurrence whenever

$$\frac{Pr(x|\text{motif model})}{Pr(x|\text{background model})} > \frac{Pr(\text{background})}{Pr(\text{motif})}.$$

A classifier based on this decision rule can be implemented by creating a log-odds matrix and a calculating a threshold. The log-odds matrix can then be used to calculate the left side of the decision rule for a given subsequence x. The result is then compared to the threshold, which is equal to the right side of the decision rule above.

[6] If we call motif occurrences class 1 and background sequence class 0, then the four possible costs are $r_{0,0}, r_{0,1}, r_{1,0}$ and $r_{1,1}$ where $r_{i,j}$ is the cost associated with predicting class i when the correct class is j. Thus, $r_{0,0}$ is the cost of true negatives, $r_{0,1}$ is the cost of false negatives, etc.

The log-odds matrix has L rows and W columns and is calculated as

$$LO_{a,j} = \log(P_{a,j}/P_{a,0})$$

for $j = 1, \ldots, W$ and $a \in \mathcal{L}$. Each entry thus corresponds to the logarithm of the relative frequencies of a given letter in a given position in the motif. The prior probability of the motif in the dataset is one possible choice for prior on the motif, $Pr(\text{motif})$. With this choice, the threshold t for the zero-one loss function is $t = \log((1-\lambda)/\lambda)$, which is the logarithm of the right side of the decision rule, $Pr(\text{background})/Pr(\text{motif})$.

To use LO and t as a classifier with a new dataset, each (overlapping) subsequence $x = \langle x_1, x_2, \ldots, x_w \rangle$ is given score

$$s(x) = \sum_{j=1}^{W} LO_{x_j,j}.$$

This is referred to as the information content (IC) score of the subsequence with respect to the motif.[7] It can be shown that

$$\begin{aligned} s(x) &= \log \frac{Pr(x|\theta_1)}{Pr(x|\theta_0)} \\ &= \log \frac{Pr(x|\text{motif model})}{Pr(x|\text{background model})} \end{aligned}$$

The decision rule for the classifier is, therefore, to classify sample x as being an occurrence of the motif if and only if $s(x) > t$. The threshold for any other loss function [Duda and Hart, 1973] can easily be found by scaling t. The scaled threshold t' should be

$$t' = t + \log \frac{(r_{1,0} - r_{0,0})}{(r_{0,1} - r_{1,1})},$$

where r_{ij} is the loss incurred for deciding class i when the correct class is j, and class 1 is the motif, class 0 the background.

[7] The score $s(x)$ is called the information content score because it is related to the information content of the motif relative to the background distribution. The information content of column j of a motif is defined as

$$IC_j = \sum_{a \in \mathcal{L}} P_{a,j} \log(P_{a,j}/Pa, 0) = \sum_{a \in \mathcal{L}} P_{a,j} LO_{a,j}.$$

III.I.2 Search and histogram tool: PROBER

The software tool PROBER searches a sequence database using MEME motifs. The search is done by computing the information content score of each subsequence in the database using a position-dependent log-odds scoring matrix derived from the MEME motif. The IC score of a subsequence is computed by adding the positions in the log-odds matrix corresponding to the letters in the subsequence. Hits are defined as IC scores above a threshold. A default threshold is provided by MEME; a different threshold may be input by the user.

PROBER produces a list of the sequences with hits showing

- the sequence identifier,
- the starting position of the hit within the sequence,
- the information content score of the hit, and
- the hit subsequence with ten positions of flanking sequence on either side.

PROBER also produces a score histogram which can be used to judge the sensitivity and selectivity of the motif and choose the optimal threshold for future searches using PROBER or NOTE.

An example of the output of software tool PROBER is shown in Figure III.5. The data show the output of PROBER when a group of motifs found by MEME in a small set of protein sequences was used to search the SWISS-PROT release 30 protein sequence database. The first few lines identify

- the program (`prober`),
- the name of the file containing the motifs discovered by MEME (`test002`), and
- the information content of the first motif found (`36.1 bits`).

```
program=    prober
experiment= test002
dataset=    swissprot30

Motif  1: IC =   36.1 bits

There were    12 scores above the threshold of 21
Alignment of sites with IC scores over 21:
sequence         start     IC        pre site           post
--------         -----     --        --- ----           ----
B61_HUMAN          101  21.02 QCNRPSAKHG PEKLSEKFQRFTPF TLGKEFKEGH
DPOB_RAT           138  22.18 EDKLNHHQRI GLKYFEDFEKRIPR EEMLQMQDIV
EBR_ECOLI           20  42.59 AIVGEVIATS ALKSSEGFTKLAPS AVVIIGYGIA
EBR_STAAU           19  41.03 AISTEVIGSA FLKSSEGFSKFIPS LGTIISFGIC
EMRE_ECOLI          20  46.28 AILAEVIGTT LMKFSEGFTRLWPS VGTIICYCAS
LGB2_SESRO          45  22.08 KAPAAKGMFS FLKDSDGVPQNNPS LQAHAEKVFG
LYSH_CHICK          36  27.09 QEENLLVLTV ATKQTEGFRRFRRS AQFFNYKIQV
LYSH_HUMAN          34  26.47 PEDNLLVLTV ATKETEGFRRFKRS AQFFNYKIQA
PT11_YEAST         434  21.43 FEGLFKYYRS VVKTTDGKTRLRPA FIQQLWSCAV
SUGE_ECOLI          92  50.36 AGLLEVVWAV GLKYTHGFSRLTPS VITVTAMIVS
SUGE_PROVU          19  52.50 AGLLEIVWAV GLKYTHGFTRLTPS IITISAMIVS
VP39_NPVAC         134  23.82 NENAVNTICD NLKYTEGFTSNTQR VIHSVYATTK

Histogram of information content scores on dataset swissprot30:
================================================================================
                    0         10        20        30        40        50
  center (  count  )+         +         +         +         +         +
================================================================================
      52 (        1)|*
      50 (        1)|*
-----------------------------------------  9 sigma -----------------------------
      48 (        0)|
      46 (        1)|*
      44 (        0)|
      42 (        2)|**
-----------------------------------------  8 sigma -----------------------------
      40 (        0)|
      38 (        0)|
      36 (        0)|
      34 (        0)|
-----------------------------------------  7 sigma -----------------------------
      32 (        0)|
      30 (        0)|
      28 (        1)|*
      26 (        1)|*
      24 (        1)|*
-----------------------------------------  6 sigma -----------------------------
      22 (        4)|****
      20 (        2)|**
      18 (       15)|***************
      16 (       43)|*******************************************
-----------------------------------------  5 sigma -----------------------------
      14 (      121)|************************************************~
      12 (      331)|************************************************~
      10 (      606)|************************************************~
       8 (     1252)|************************************************~
-----------------------------------------  4 sigma -----------------------------
```

Figure III.5: **Sample** PROBER **output.**

The next lines show the threshold being used to determine hits (`21`) and the number of with hits (`12`).

The alignment of the positions where hits were scored is shown next. Each line in the alignment shows

- the sequence identifier (i.e., `B61_HUMAN`),
- starting position of the hit (i.e., `101`),
- IC score of the hit (i.e., `21.02 bits`),
- the actual sequence matching the motif (i.e., `PEKLSEKFQRFTPF`),
- the left flanking sequence (i.e., `QCNRPSAKHG`), and
- the right flanking sequence (i.e., `TLGKEFKEGH`).

The score histogram comes after the alignment of hits. The IC score of the center of each bin is shown in the leftmost column. (The histogram has been truncated at scores of 8 bits in the example to save space.) The next column shows the number of hits in the bin. Horizontal dividing lines show the approximate locations of scores that are n standard deviations from the mean score of this motif for all subsequences in the dataset. Each "*" represents one hit. Off-scale columns are indicated by " ".

III.I.3 Annotation and block diagram tool: NOTE

Software tool NOTE uses the motifs discovered by MEME in a group of related sequences to annotate other sequences. Each sequence in a database of sequences is searched for matches with each of the motifs in the same way as described above for PROBER. Each sequence with one or more hits is printed with all the hits for all the motifs highlighted so that the relative positioning of the motifs is visible.

```
program:   note
memefile:  test001
searching: adh.s
args: test001 -T 18 -c 5

name            pos/type         MAXSUM  diagram/annotation

S20554          hitsA:  4 score:    96   3-1-2-4-<4>
S20554          hitsB:  4 score:    96   (33)-3-(9)-1-(1)-2-(6)-4-<4>-(13)
S20554             1                                3333333333333333         1
S20554             1 IIAEGRAIGHRIGAGPVKVIHDISEMNRIEPGDVLVTDMTDPDWEPIMKKASAIVTNRG
S20554            60 11111 222222222222222      44444444444
S20554            60 GRTCHAAIIARELGIPAVVGCGDATERMKDGENVTVSCAEGDTGYVYAELL
```

Figure III.6: **Sample** NOTE **output.**

NOTE also produces two schematic block diagrams of each sequence containing one or more hits. The block diagrams show, respectively,

- just the order of the motifs, or
- the order of the motifs and their spacings.

These block diagrams show the strong hits as well as weak hits to aid in detecting distantly related sequences where not all motifs have been well conserved. These are defined as

- strong hits—IC scores above the default or user-provided threshold, and
- weak hits—IC scores above zero.

The block diagrams are sorted by the sum of the maximum score for each motif in a given sequence which gives a measure of the overall degree of match between the sequence and the original group of sequences given as input to MEME.

Figure III.6 shows the first entry in the output file produced by NOTE when it was used to search for occurrences of a group of MEME-discovered motifs. The first four lines indicate

- the program (`note`),
- the name of the file containing the MEME-discovered motifs (`test001`),

- the name of the database being searched (`adh.s`)
- arguments to the program (`-T 18` means the threshold for hits being used is 18 bits; `-c 5` means only the first five motifs found by MEME are used in the search).

The five motifs found by MEME are referred to in the following lines of Figure III.6 by their numbers.

After the header line describing the content of each column, two block diagrams are shown in the output of NOTE in Figure III.6. The "`hitsA:`" line shows

- the sequence identifier of the sequence with motif hit(s) (`20554`),
- the number of *different* motifs with hits in the sequence,
- the sum of best scores of the sequence versus each of the five motifs,
- the motif block diagram for the sequence showing strong hits (`-3-`) and weak hits (`-<4>-`). The "`hitsB:`" line shows the same things plus the inter-motif spacings (`-9-`).

The annotated sequence is shown following the two motif block diagrams in Figure III.6. Each sequence line shows the sequence identifier and offset within the sequence. Printed numbers above each sequence line show the positions and extents of strong motif hits.

Chapter IV

Algorithmic details and derivations

Patterns can be discovered in a sequence dataset by modeling the sequences statistically. The fitted statistical model of a given dataset can be examined to shed light on the particular pattern it contains. The model can also be used as a classifier to predict the probability that other sequences contain the patterns. Statistical models of sequence datasets thus both describe the patterns in them in a way that is meaningful to humans, and provide a way to search for new occurrences of the patterns.

This chapter gives detailed derivations for the functions and algorithms described in Chapter III. The notation used is summarized in Tables IV.1 and IV.2. The statistical models of sequences used by MEME are described in detail in the first section. The second section describes the EM algorithm in detail. The next three sections give derivations of the joint likelihood functions and describes the analysis necessary for implementing the EM algorithm for each of the three sequence model types. The last section discusses the likelihood of the null model used in computing the LRT-based heuristic function $G(\phi)$.

A	the length of the sequence alphabet
$\mathcal{L} = \{a_1, a_2, \ldots, a_A\}$	the sequence alphabet
n	the number of sequences in the dataset
L	the length of each sequence in the dataset
$X = \{X_1, X_2, \ldots, X_n\}$	a dataset of sequences
X_i	a sequence
$I(i, j, a)$	scalar indicator variable, 1 if $X_{i,j} = a$, 0 otherwise
$X_{i,j}$	the letter at position j in sequence X_i
$t_{i,a} = \sum_{k=1}^{L} I(i, k, a), \;\; a \in \mathcal{L}$	the total number of times letter a occurs in sequence X_i
$\mathbf{t}_i = [t_{i,a_1} \;\; t_{i,a_2} \;\; \ldots \;\; t_{i,a_A}]^T$	the length-A vector of total counts of each letter in sequence X_i
$\mathbf{t} = \sum_{i=1}^{n} \mathbf{t}_i$	the length-A vector of total counts of each letter in dataset X
$\mathbf{I}(i, j)$	kth column from identity matrix when $X_{i,j} = a_k$
$\lvert\mathbf{x}\rvert$	the sum of the components of any vector $\mathbf{x}$

Table IV.1: **Notation for describing datasets.**

$\phi = (\theta, W)$	the parameters of an OOPS sequence model
$\phi = (\theta, \gamma, W)$	the parameters of a ZOOPS sequence model
$\phi = (\theta, \lambda, W)$	the parameters of a TCM sequence model
$\theta = (\theta_0, \theta_1)$	the parameters of the mixture model components
$\theta_0 = \mathbf{p}_0$	the background component
$\theta_1 = [\ \mathbf{p}_1 \quad \mathbf{p}_2 \quad \ldots \quad \mathbf{p}_W\ ]$	the motif component
γ	probability of a sequence having a motif occurrence in a ZOOPS model
λ	probability of starting a motif in a TCM model
W	the width of the motif
$M_i \sim discrete(\mathbf{p}_i)$	a discrete random variable with vector of parameters $\mathbf{p}_i = [P_{a,i}, P_{b,i}, \ldots, P_{z,i}]^T$
$P_{a,0}$	the probability of letter a occurring outside a motif occurrence
$P_{a,j},\ 1 \leq j \leq W$	the probability of letter a occurring at position j of a motif occurrence
$m = L - W + 1$	the number of possible starts for a motif in a sequence

Table IV.2: **Notation for describing models.**

$Z = \{Z_{i,j}\}$	set of indicator variables showing the starting points of motif occurrences where, for $1 \leq i \leq n$ and $1 \leq j \leq m$, $Z_{i,j} = \begin{cases} 1, & \text{if a motif occurrence starts at position } j \text{ in } X_i \\ 0, & \text{otherwise} \end{cases}$
$Q = \{Q_i\}$	set of indicator variables showing if sequence X_i contains a motif occurrence where, for $1 \leq i \leq n$, $Q_i = \begin{cases} 1, & X_i \text{ contains a motif occurrence} \\ 0, & \text{otherwise} \end{cases}$
$U = \{U_{i,j}\}$	set of indicator variables showing positions not contained in occurrences of motifs found on previous passes where, for $1 \leq i \leq n$ and $1 \leq j \leq m$, $U_{i,j} = \begin{cases} 1, & X_{i,j} \notin \text{previous motif occurrence} \\ 0, & \text{otherwise} \end{cases}$
$V = \{V_{i,j}\}$	set of indicator variables showing possible positions for a new motif occurrence where, for $1 \leq i \leq n$ and $1 \leq j \leq m$, $V_{i,j} = \begin{cases} 1, & \text{if no old motif occurrences in } [X_{i,j}, \ldots, X_{i,j+W-1}] \\ 0, & \text{otherwise} \end{cases}$

Table IV.3: **Notation for EM.**

IV.A Statistical models of sequences

Protein or DNA sequence datasets are modeled statistically by MEME by assuming that each sequence in the dataset is an independent sample from some probability distribution over all possible sequences. This distribution models a random process which is assumed to have generated the sequences.. Each of the three sequence model types used by MEME assumes a slightly different underlying random process. They have in common the assumption that a sequence consists of one or more occurrences of a motif embedded within a background sequence. This section describes the concepts of background and motif sequence and the random processes underlying the types of models used by MEME.

IV.A.1 Motifs and background

In all the model types used by MEME, motifs are described by a simple statistical model. The model is a sequence of independent, discrete random variables. If $M_i,\ i = 1, \ldots, W$, are discrete random variables with parameter vectors $\mathbf{p}_i,\ i = 1, \ldots, W$, respectively, then

$$M = (M_1, M_2, \ldots, M_W)$$

is a motif of width W. The motif M can be considered to be a random variable whose instances are sequences of length W. An occurrence of motif M is a sample taken according to the distribution of M. In other words, an occurrence of motif M is a sequence $b_1 b_2 \ldots b_W$, where b_i is a sample from the discrete random variable $M_i \sim \mathit{discrete}(\mathbf{p}_i)$. Thus, each discrete random variable making up the motif defines the probability of seeing each possible letter of the sequence alphabet at that position in an actual occurrence of the motif.

Since the letter b_i at position i in a motif occurrence is an independent sample from discrete random variable M_i, the fact that $b_i = a$ has no effect on the letter b_j at some other position j in the motif. More precisely, for $1 \leq i \leq W$ and

for all $a, b \in \mathcal{L}$,[1]

$$Pr(b_j = b | b_i = a) = Pr(b_j = b),\ j \neq i,\ 1 \leq j \leq W.$$

As stated earlier, positions in a sequence which are not occurrences of a motif are called background positions. In all three model types used by MEME, background positions are generated by repeated, independent sampling from a single discrete random variable M_0 with parameter vector $\mathbf{p}_0$. Each position in a sequence which is not a motif occurrence is thus an independent sample from $M_0 \sim discrete(\mathbf{p}_0)$.

For convenience, I will refer to the parameter vector of the background distribution M_0 as $\theta_0 = [\mathbf{p}_0]$, and the sequence of parameter vectors for the motif distribution M as $\theta_1 = [\mathbf{p}_1\ \mathbf{p}_2\ \ldots\ \mathbf{p}_W]$. To refer to both distributions I will use $\theta = (\theta_0, \theta_1)$.

IV.A.2 The OOPS random process

An OOPS model generates a sequence of length L from left to right by a four-step process. First, a position j is randomly selected for the start of the motif occurrence. Second, the background distribution M_0 is sampled $j-1$ times. Third, the motif is generated. Fourth, the background distribution is sampled $L-W-j+1$ times. This creates a sequence consisting of a single motif occurrence embedded in background.

The probability of a motif occurrence starting at position j in a sequence of length L is assumed to be uniform for all $1 \leq j \leq L - W + 1$. To express this mathematically, it is convenient to introduce some more notation. First, let

[1] In biological terms this implies that having a particular amino acid or nucleotide at a position in a motif occurrence does not affect the probabilities of the amino acids or nucleotides at other positions in the motif occurrence. This precludes the type of motif used by MEME from being able to describe *some* motifs where interactions between the amino or nucleic acids in a particular occurrence are important. However, many biologically important motifs can be described by the class of motifs used by MEME, as will be evident from the success of MEME at discovering motifs in actual biopolymer sequence datasets.

- Sample $j \leftarrow S \sim U(1, m)$.
- Sample $j - 1$ times from M_0.
- Sample, in order, once each from $M_1, M_2, \ldots, M_W$.
- Sample $m - j$ times from M_0.

Figure IV.1: **The OOPS random process.**

$m = L - W + 1$. Then, let U(i,j) represent the uniform distribution over integers on the interval $[i, j]$. The assumption of uniformly distributed motif occurrence starts can then written as

$$S \sim U(1, m)$$

where S is a random variable. Now we can write the OOPS random process as shown in Figure IV.1.

The length of the sequence, L, is assumed to be fixed, so an OOPS model can be described by the values of the parameters

$$\phi = (\theta, W).$$

IV.A.3 The ZOOPS random process

A ZOOPS model generates sequences in much the same way as an OOPS model, with the exception that the sequence may contain no motif occurrence. One way of envisioning this is as a five-step process, where step zero is to randomly choose whether or not the sequence being generated will have a motif occurrence. In the case that it will, the rest of the process is just like for an OOPS model. Otherwise, a ZOOPS model just generates a sequence by sampling L times from the background distribution M_0.

- Sample $q \leftarrow Q \sim bernoulli(\gamma)$.
- If $q = 1$ then
  - Sample $j \leftarrow S \sim U(1, m)$.
  - Sample $j - 1$ times from M_0.
  - Sample, in order, once each from $M_1, M_2, \ldots, M_W$.
  - Sample $m - j$ times from M_0.
- otherwise $(q = 0)$
  - Sample L times from M_0.

Figure IV.2: **The ZOOPS random process.**

The ZOOPS random process introduces an additional (hidden) random variable Q which has the value 1 if the sequence has a motif occurrence, and 0 otherwise. We define the probability that $Q = 1$ to be γ,

$$Pr(Q = 1) = \gamma.$$

The ZOOPS random process is shown in Figure IV.2

A ZOOPS model is parameterized by the width of the motif, W, by the letter frequency distributions of the background θ_0 and the motif θ_1 by the probability of a sequence containing an occurrence of the motif, γ, and by the length of the sequences, L. As before, L is assumed to be fixed, so the parameters of the model are

$$\phi = (\theta, \gamma, W).$$

While the length of the generated sequence is less than L

- Sample $c \leftarrow C \sim bernoulli(\lambda)$.
- If $c = 1$ then
  - Sample, in order, once each from $M_1, M_2, \ldots, M_W$.
- otherwise ($c = 0$)
  - Sample once from M_0.

Figure IV.3: **The TCM random process.**

IV.A.4 The TCM random process

A TCM model can be visualized as generating sequences using a two-step process which repeatedly chooses randomly whether to generate a motif occurrence or a background position and then samples from the appropriate distribution, M or M_0, respectively.

The TCM random process generates a sequence by repeatedly sampling from a two-component mixture density. One component, the background, generates a single letter. The other component, the motif, generates a string of letters of length W. This process can be thought of as first choosing which component to use by sampling from a (hidden) random variable C and then sampling from the appropriate component. The probability of choosing the motif component is defined to be λ and the probability of choosing the background component is $1-\lambda$. The TCM random process can thus be represented as shown in Figure IV.3.

The length of the sequence, L, is again assumed fixed so the parameters of a TCM model are

$$\phi = (\theta, \lambda, W).$$

IV.B Expectation Maximization

Given a set of sequences assumed to have been generated by a random process which can be approximated by one of the model types discussed above, we would like to discover the values of the parameters of the model. Knowing the parameters of the model will completely describe the process which generated the sequences.

IV.B.1 Maximum likelihood estimation

Because the set of sequences is assumed to be a random sample from a random process, we cannot hope to definitively discover the actual underlying parameter values. However, we can use statistical methods to arrive at good estimates. One way of doing this is maximum likelihood estimation (MLE). MLE consists of finding the parameter values which maximize the likelihood of the model given the data. This is equivalent to maximizing the probability of the data given the model. The maximum likelihood estimate of the parameters ϕ of a model given a dataset of sequences X is defined to be

$$\phi^* = \underset{\phi}{\operatorname{argmax}} Pr(X|\phi) \tag{IV.1}$$

if the argmax exists.

Expectation maximization is a method of solving the maximum likelihood estimation problem. EM is an iterative algorithm which is generally used to solve MLE problems where some of the data is missing. In the context of the sequence motif discovery problem we are discussing, the missing data is the starting point(s), if any, of motif occurrence(s) in each sequence in the dataset. EM starts with an initial estimate $\phi^{(t)}$ for the model parameters. At each iteration, EM maximizes the expected value of the conditional joint probability of the data, X, and the missing data, Z, given the model where the expectation is over the distribution of the missing data given the actual data and the current estimate $\phi^{(t)}$ of the model

parameters. We will call this objective function the "expected joint log likelihood". In other words, at each iteration, EM solves the equation

$$\phi^{(t+1)} = \underset{\phi}{\operatorname{argmax}} \underset{(Z|X,\phi^{(t)})}{\mathrm{E}}[\log Pr(X, Z|\phi)]. \tag{IV.2}$$

Theoretical results [Redner and Walker, 1984] show that EM converges to a local optimum of the likelihood function $Pr(X|\phi)$. Additional methods can be applied to increase the probability of finding the global optimum.

The EM algorithm solves the MLE problem by repeatedly performing what are referred to as the expectation- and maximization-steps (the E- and M-steps).

The EM algorithm

- E-step: compute $Z^{(t)} = \underset{(Z|X,\phi^{(t)})}{\mathrm{E}}[Z]$
- M-step: solve $\phi^{(t+1)} = \underset{\phi}{\operatorname{argmax}} \underset{(Z|X,\phi^{(t)})}{\mathrm{E}}[\log Pr(X, Z|\phi)]$

These two steps are repeated until a convergence criterion is met. MEME uses a criterion based on the change in $\theta^{(t)}$.

As mentioned above, the missing data Z in equation (IV.2) is a set of random variables specifying the starting points of the occurrences of the motif in the dataset. MEME uses one random variable $Z_{i,j}$ for each possible starting position j in each sequence X_i. So, for $1 \leq i \leq n$ and $1 \leq j \leq m$,

$$Z_{i,j} = \begin{cases} 1 & \text{if motif starts at position } j \text{ in sequence } X_i \\ 0 & \text{otherwise} \end{cases}$$

In the case of a ZOOPS model, the missing data also includes the binary hidden variable Q which specifies whether or not a sequence contains a motif occurrence. For Q I will use the notation Q_i, for $1 \leq i \leq n$, where

$$Q_i = \begin{cases} 1 & \text{if } X_i \text{ contains a motif occurrence} \\ 0 & \text{otherwise} \end{cases}$$

Observe that the definition of a ZOOPS model implies that

$$Q_i = \sum_{j=1}^{m} Z_{i,j} \ ,$$

so the Q_i are completely determined by the $Z_{i,j}$.

For convenience I define a vector-valued indicator variable $\mathbf{I}(i,j)$ of length A whose entries are all zero except the entry corresponding to the letter in sequence X_i at position j is 1. In other words, if $X_{i,j} = a_k$ is the kth letter in alphabet $\mathcal{L}$, then $\mathbf{I}(i,j)$ is the k-th column of the identity matrix. Likewise, I define a scalar indicator variable $I(i,j,a)$ whose value is 1 if and only if sequence X_i has letter a at position j. These variables will be convenient for writing down the probability functions needed in the deriving the steps of the EM algorithm for the three MEME sequence model types.

To represent the set of positions in sequence X_i which lie outside of the occurrence of the motif when the motif starts at position j, I use the notation $\Delta_{i,j} = \{1, 2, \ldots, j-1, j+w, \ldots, L\}$. This will be useful in the analysis of OOPS and ZOOPS models.

To specify the counts of letters appearing in each sequence I will use

$$\mathbf{t}_i = [t_{i,a_1} \ t_{i,a_2} \ \ldots \ t_{i,a_A}]^T,$$

where

$$t_{i,a} = \sum_{k=1}^{L} I(i,k,a), \ \ a \in \mathcal{L}$$

is the total number of times letter a occurs in sequence X_i. Then,

$$\mathbf{t} = \sum_{i=1}^{n} \mathbf{t}_i$$

is the vector of total counts of letters over all sequences in the dataset. The notation $f(\mathbf{M})$ will refer to the matrix (or vector) obtained by applying function $f(\cdot)$ to each element of matrix (or vector) $\mathbf{M}$.

IV.B.2 Bayesian estimation

As discussed in the previous chapter, it is desirable to modify EM slightly in order to avoid overfitting small datasets or getting stuck at the edges of the likelihood surface if any parameter ever is estimated to be zero. So, in MEME the maximization step of EM is changed so that the mean posterior estimates of the the letter probabilities $\mathbf{p}_k$ given a prior distribution on $\mathbf{p}_k$ are calculated instead of the maximum likelihood estimates. The equations for doing this have already been given in section III.E. For each model type, the maximization step becomes

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k + \mathbf{d}(\mathbf{c}_k)}{|\mathbf{c}_k + \mathbf{d}(\mathbf{c}_k)|}$$

for $k = 1$ to W. The reestimation formulas given in the following sections are each replaced by the above in the actual MEME implementation.

IV.C EM and the OOPS model

An OOPS model generates a sequence by choosing a starting position j, generating $j-1$ background positions preceding the motif, generating a motif occurrence of width W, and then generating $m-j$ additional background positions following the motif occurrence.

Although the formulas can become a bit dense, the EM algorithm as applied to an OOPS model has a very intuitive interpretation. Starting from an initial guess at the free parameters of the model, θ, EM reestimates the model by alternately estimating the probability of the motif occurrences starting at each possible position in the dataset, and then using those probabilities to estimate the expected letter frequencies in background positions and at each position in motif occurrences. The expected letter frequencies constitute the next guess for θ.

EM tries to maximize the likelihood of the model given the data over values of the OOPS model parameters

$$\phi = (\theta, W).$$

We hold W constant, so maximization is actually only over θ. In the E-step, EM will estimate the probability of the motif occurrence starting in position j of sequence X_i as

$$Z_{i,j}^{(t)} = \frac{Pr(X_i|Z_{i,j}=1,\phi^{(t)})}{\sum_{j=1}^{m} Pr(X_i|Z_{i,j}=1,\phi^{(t)})}$$

given the current estimate of ϕ, $\phi^{(t)}$. Then, the expected value of the number of times each letter appears at each background or motif position will be shown to be

$$\mathbf{c}_k = \begin{cases} \mathbf{t} - \sum_{j=1}^{W} \mathbf{c}_j & \text{if } k = 0 \\ \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)} \mathbf{I}(i, j+k-1) & \text{if } 1 \le k \le W \end{cases}.$$

For $\mathbf{c}_0$ this can be seen to be the result of just adding the probabilities of letters across the columns of the motif, and subtracting those counts from the total counts for the entire dataset. Finally, the M-step of EM will use the expected letter frequencies,

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k}{|\mathbf{c}_k|},$$

as the new estimate for θ.

IV.C.1 The E-step for OOPS

The E-step of EM requires that we find the value of $Z_{i,j}^{(t)}$ which we do using the definition of expectation and Bayes' rule yielding, for $i = 1, \ldots, n,\ j = 1, \ldots, m$,

$$\begin{aligned} Z_{i,j}^{(t)} &= \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z_{i,j}] \\ &= 0 \cdot Pr(Z_{i,j}=0|X_i,\phi^{(t)}) + 1 \cdot Pr(Z_{i,j}=1|X_i,\phi^{(t)}) \\ &= Pr(Z_{i,j}=1|X_i,\phi^{(t)}) \\ &= \frac{Pr(X_i|Z_{i,j}=1,\phi^{(t)})\ Pr(Z_{i,j}=1|\phi^{(t)})}{Pr(X_i|\phi^{(t)})} \text{ (by Bayes' rule)} \\ &= \frac{\frac{1}{m} Pr(X_i|Z_{i,j}=1,\phi^{(t)})}{\frac{1}{m}\sum_{j=1}^{m} Pr(X_i|Z_{i,j}=1,\phi^{(t)})} \end{aligned}$$

$$= \frac{Pr(X_i|Z_{i,j}=1,\phi^{(t)})}{\sum_{j=1}^{m} Pr(X_i|Z_{i,j}=1,\phi^{(t)})}.$$

Note that an OOPS model assumes that the prior probability of the event $Z_{i,j}=1$, in the absence of information about X_i, is constant:

$$Pr(Z_{i,j}=1|\phi^{(t)}) = \frac{1}{m},\ 1 \le i \le n,\ 1 \le j \le m,$$

and we do not reestimate it.

To calculate $Z_{i,j}^{(t)}$, we must know the probability of a sequence X_i given the starting point of the motif occurrence within it. This is easily calculated from the definition of the OOPS model random process. We form the product of the probabilities of each letter which actually occurs in the sequence given that the letter is in a background or motif occurrence position. In other words, we form the product of $P_{X_{i,k},k'}^{(t)}$for each position k in sequence X_i, where $k'=0$ if k is a background position, and $1 \le k' \le w$ if k is at position k' in the motif occurrence.

$$Pr(X_i|Z_{i,j}=1,\phi^{(t)}) = \prod_{k=1}^{L} P_{X_{i,k},k'}^{(t)}$$

where

$$k' = \begin{cases} k-j+1 & \text{if } k \ge j \text{ and } k < j+w \\ 0 & \text{otherwise} \end{cases}.$$

IV.C.2 The M-step for OOPS

To perform the M-step of EM for an OOPS model, we need to derive an analytical solution for equation (IV.2). The first task is to write down an expression for the joint probability of the observed data (the sequences X) and the missing data (the motif occurrence starting points Z). We use the fact that the values of all $Z_{i,j}$ are either 0 or 1 to put things in a form that makes taking the logarithm easier.

$$Pr(X,Z|\phi) = \prod_{i=1}^{n} Pr(X_i,Z_i|\phi)$$

$$
\begin{aligned}
&= \prod_{i=1}^{n} Pr(X_i|Z_i,\phi)Pr(Z_i|\phi) \\
&= \prod_{i=1}^{n}(\frac{1}{m}\prod_{j=1}^{m} Pr(X_i|Z_{i,j}=1,\phi)^{Z_{i,j}}).
\end{aligned}
$$

The logarithm of the joint likelihood is

$$
\log Pr(X,Z|\phi) = \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}\log Pr(X_i|Z_{i,j}=1,\theta) + n\log\frac{1}{m} \qquad \text{(IV.3)}
$$

Here I have replaced $Pr(X_i|Z_{i,j}=1,\phi)$ with $Pr(X_i|Z_{i,j}=1,\theta)$ since W is assumed constant.

Next we take the expectation of (IV.3) with respect to the probability of Z given the observed data and current estimate of ϕ.

$$
\begin{aligned}
&\mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[\log Pr(X,Z|\phi)] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z_{i,j}]\log Pr(X_i|Z_{i,j}=1,\theta) + n\log\frac{1}{m} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)}\log Pr(X_i|Z_{i,j}=1,\theta) + n\log\frac{1}{m}. \qquad \text{(IV.4)}
\end{aligned}
$$

To find the value of θ that maximizes (IV.4) we need to write down the logarithm of the probability of a sequence X_i given the starting point of the motif occurrence in X_i. Recall that $\Delta_{i,j}$ represents the positions in sequence X_i which are not contained in a motif occurrence. Then, the logarithm of the probability of sequence X_i given that the occurrence of the motif starts at position j within it can be written as

$$
\begin{aligned}
&\log Pr(X_i|Z_{i,j}=1,\theta) \\
&= \log(\prod_{a\in\mathcal{L}}\prod_{k=1}^{W} P_{a,k}^{I(i,j+k-1,a)} \prod_{a\in\mathcal{L}}\prod_{k\in\Delta_{i,j}} P_{a,0}^{I(i,k,a)}) \\
&= \sum_{a\in\mathcal{L}}\sum_{k=1}^{W} I(i,j+k-1,a)\log P_{a,k} + \sum_{a\in\mathcal{L}}\sum_{k\in\Delta_{i,j}} I(i,k,a)\log P_{a,0} \\
&= \sum_{k=1}^{W} \mathbf{I}(i,j+k-1)^T\log\mathbf{p}_k + \sum_{k\in\Delta_{i,j}} \mathbf{I}(i,k)^T\log\mathbf{p}_0. \qquad \text{(IV.5)}
\end{aligned}
$$

Note that $\log Pr(X_i|Z_{i,j}=1,\theta)$ is a linear function of $\log\theta$.

The explicit formula for the expected joint log likelihood can now be obtained by substituting (IV.5) into (IV.4) giving

$$\begin{aligned}
&\mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[\log Pr(X,Z|\phi)] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)} \log Pr(X_i|Z_{i,j}=1,\theta) + n\log\frac{1}{m} \\
&= \sum_{k=1}^{W}\sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)}\mathbf{I}(i,j+k-1)^T\log\mathbf{p}_k \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k\in\Delta_{i,j}} Z_{i,j}^{(t)}\mathbf{I}(i,k)^T\log\mathbf{p}_0 + n\log\frac{1}{m} \\
&= \sum_{k=0}^{W}\mathbf{c}_k^T\log\mathbf{p}_k + n\log\frac{1}{m} \qquad \text{(IV.6)}
\end{aligned}$$

where

$$\mathbf{c}_k = \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{b\in\Delta_{i,j}} Z_{i,j}^{(t)}\mathbf{I}(i,b) & \text{if } k=0 \\ \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)}\mathbf{I}(i,j+k-1) & \text{if } 1\le k\le W. \end{cases} \qquad \text{(IV.7)}$$

Notice that $\mathbf{c}_k$ are the expected counts of letters in the background and in positions in the motif.

The expression for the expected counts of letters in background positions can be simplified to be just the total counts minus the expected counts in the motif positions

$$\mathbf{c}_0 = \mathbf{t} - \sum_{k=1}^{W}\mathbf{c}_k.$$

Observe that

$$\sum_{b\in\Delta_{i,j}}\mathbf{I}(i,b) = t_i - \sum_{b=1}^{W}\mathbf{I}(i,j+b-1).$$

Substituting into equation (IV.7) and rearranging we find that

$$\begin{aligned}
\mathbf{c}_0 &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{b\in\Delta_{i,j}} Z_{i,j}^{(t)}\mathbf{I}(i,b) \\
&= \sum_{i=1}^{n}(\sum_{j=1}^{m} Z_{i,j}^{(t)}\mathbf{t}_i - Z_{i,j}^{(t)}\sum_{b=1}^{W}\mathbf{I}(i,j+b-1))
\end{aligned}$$

$$= \sum_{i=1}^{n} \mathbf{t}_i - \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)} \sum_{b=1}^{W} \mathbf{I}(i, j+b-1)$$
$$= \mathbf{t} - \sum_{k=1}^{W} \mathbf{c}_k.$$

So a simpler version of equation (IV.7) is

$$\mathbf{c}_k = \begin{cases} \mathbf{t} - \sum_{j=1}^{W} \mathbf{c}_j & \text{if } k = 0 \\ \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)} \mathbf{I}(i, j+k-1) & \text{if } 1 \le k \le W. \end{cases} \tag{IV.8}$$

It is easy to see that the expected joint log likelihood given in equation (IV.6) can be maximized over $\mathbf{p}$ by independently maximizing over each $\mathbf{p}_k$ and ignoring the constant last term. This is straightforward using a well-known result from information theory which says that

$$\underset{\mathbf{g}}{\operatorname{argmax}} \mathbf{f} \log \mathbf{g} = \mathbf{f} \tag{IV.9}$$

if both $\mathbf{f}$ and $\mathbf{g}$ are probability vectors. We can put (IV.12) into this form by simply normalizing each $\mathbf{c}_k$ to be a probability vector. That is, we set

$$\mathbf{f}_k = \frac{\mathbf{c}_k}{|\mathbf{c}_k|}$$

where $|\mathbf{c}_k|$ is just the sum of the components of the vector $\mathbf{c}_k$. Note that $\mathbf{f}_k$ is just the expected frequency of letters in position k of the motif (or in the background when $k = 0$.) In the M-step of EM, we can now maximize the expected value of the joint likelihood with respect to θ by setting

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k}{|\mathbf{c}_k|}, \ 0 \le k \le W.$$

IV.D EM and the ZOOPS model

A ZOOPS model generates a sequence by first choosing whether the sequence will contain a motif occurrence, with probability γ, or not, with probability $1 - \gamma$. In the first case, a ZOOPS model then behaves just like an OOPS model,

generating a sequence consisting of a single motif occurrence embedded somewhere within a string of background. In the second case, where the sequence does not contain an occurrence of the motif, all L letters are generated by sampling L times independently from the background distribution.

EM tries to maximize the likelihood of the model given the data over values of the OOPS model parameters

$$\phi = (\theta, \gamma, W).$$

We hold W constant, so maximization is actually only over θ and γ. In the E-step, EM will estimate the probability of the motif occurrence starting in position j of sequence X_i as

$$Z_{i,j}^{(t)} = \frac{Pr(X_i|Z_{i,j}=1,\phi^{(t)})\lambda^{(t)}}{Pr(X_i|Q_i=0,\phi^{(t)})(1-\gamma^{(t)}) + \sum_{j=1}^{m} Pr(X_i|Z_{i,j}=1,\phi^{(t)})\lambda^{(t)}}$$

given the current estimate of ϕ, $\phi^{(t)}$. Then, the expected value of the number of times each letter appears at each background or motif position will be shown to be

$$\mathbf{c}_k = \begin{cases} \mathbf{t} - \sum_{j=1}^{W} \mathbf{c}_j & \text{if } k = 0 \\ \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)} \mathbf{I}(i, j+k-1) & \text{if } 1 \le k \le W. \end{cases}$$

This equation has exactly the same form as for an OOPS model, and can be seen to be the result of just adding the probabilities of letters across the columns of the motif, and subtracting those counts from the total counts for the entire dataset. As with an OOPS model, the M-step of EM will use the expected letter frequencies,

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k}{|\mathbf{c}_k|},$$

as the new estimate for θ. Finally, the M-step will reestimate the probability of a sequence containing a motif occurrence, γ, with the intuitively pleasing value

$$\gamma^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} Q_i^{(t)},$$

which is the expected fraction of sequences which contain a motif occurrence.

IV.D.1 The E-step for ZOOPS

The E-step of EM requires that we find the value of $Z_{i,j}^{(t)}$ which we do using the definition of expectation and Bayes' rule yielding, for $i = 1, \ldots, n,\ j = 1, \ldots, m$,

$$\begin{aligned}
Z_{i,j}^{(t)} &= \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z_{i,j}] \\
&= 0 \cdot Pr(Z_{i,j} = 0|X_i, \phi^{(t)}) + 1 \cdot Pr(Z_{i,j} = 1|X_i, \phi^{(t)}) \\
&= Pr(Z_{i,j} = 1|X_i, \phi^{(t)}) \\
&= \frac{Pr(X_i|Z_{i,j} = 1, \phi^{(t)})Pr(Z_{i,j} = 1|\phi^{(t)})}{Pr(X_i|\phi^{(t)})} \text{ (by Bayes' rule)} \\
&= \frac{Pr(X_i|Z_{i,j} = 1, \phi^{(t)})\lambda^{(t)}}{Pr(X_i|Q_i = 0, \phi^{(t)})(1 - \gamma^{(t)}) + \sum_{j=1}^{m} Pr(X_i|Z_{i,j} = 1, \phi^{(t)})\lambda^{(t)}}
\end{aligned}$$

where $\lambda = \gamma/m$ is the fraction of positions in the sequences which are motif occurrence starts. In the above derivation, we have substituted $\lambda^{(t)}$ for $Pr(Z_{i,j} = 1|\phi^{(t)})$ since $Q_i = 0$ implies there is no motif occurrence in sequence X_i, i.e.,

$$\forall i(Q_i = 0) \Rightarrow \forall j\ Z_{i,j} = 0\ .$$

So

$$\begin{aligned}
Pr(Z_{i,j} = 1|\phi) &= Pr(Z_{i,j} = 1|Q_i = 0)Pr(Q_i = 0) + \\
&\quad Pr(Z_{i,j} = 1|Q_i = 1)Pr(Q_i = 1) \\
&= 0(1 - \gamma) + \frac{\gamma}{m} \\
&= \frac{\gamma}{m}, \\
&= \lambda,\ \ 1 \le i \le n, 1 \le j \le m.
\end{aligned}$$

To finish the calculation, the probability of the sequence given the position where the motif occurrence starts, $Pr(X_i|Z_{i,j} = 1, \phi^{(t)})$, is calculated exactly as for an OOPS model. The probability of the sequence given that it contains no motif

occurrence, $Pr(X_i|Q_i = 0, \phi^{(t)})$, is just

$$Pr(X_i|Q_i = 0, \phi^{(t)}) = \prod_{k=1}^{L} P^{(t)}_{X_{i,k},0}.$$

IV.D.2 The M-step for ZOOPS

The M-step requires that we solve equation (IV.2). We thus need to express

$$\mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[\log Pr(X, Z|\phi)]$$

in terms of

$$Z^{(t)} = \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z].$$

We first write $Pr(X, Z|\phi)$ in a way that makes its form simple when we take logarithms. Assuming the independence of samples in X,

$$\begin{aligned}
& Pr(X, Z|\phi) \\
&= \prod_{i=1}^{n} Pr(X_i, Z_i|\phi) \\
&= \prod_{i=1}^{n} Pr(X_i|Z_i, \phi) Pr(Z_i|\phi) \\
&= \prod_{i=1}^{n} (\prod_{j=1}^{m} Pr(X_i|Z_{i,j} = 1, \phi)^{Z_{i,j}}) Pr(X_i|Q_i = 0, \phi)^{(1-Q_i)} \lambda^{Q_i} (1-\gamma)^{(1-Q_i)} \\
&= \prod_{i=1}^{n} (\ (\prod_{j=1}^{m} Pr(X_i|Z_{i,j} = 1, \phi)^{Z_{i,j}}) \lambda^{Q_i} \ (Pr(X_i|Q_i = 0, \phi)(1-\gamma))^{(1-Q_i)} \ ).
\end{aligned}$$

Notice that we have rewritten $Pr(X_i|Z_i, \phi)$ as we did with the OOPS model taking advantage of the fact all $Z_{i,j}$ are either 0 or 1. So Q_i are all either 0 or 1 as well, and

$$Pr(Z_i|\phi) = \lambda^{Q_i}(1-\gamma)^{(1-Q_i)}$$

where the first term covers the case when there is a motif and the second term covers the case of no motif in sequence X_i.

The logarithm of the joint likelihood can be written as

$$\begin{aligned}
&\log Pr(X, Z|\phi) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j} \log Pr(X_i|Z_{i,j}=1,\theta) + \sum_{i=1}^{n}(1-Q_i)\log Pr(X_i|Q_i=0,\theta) \\
&\quad + \sum_{i=1}^{n}(1-Q_i)\log(1-\gamma) + \sum_{i=1}^{n} Q_i \log\lambda
\end{aligned}$$

Here we have replaced $Pr(X_i|Z_{i,j}=1,\phi)$ with $Pr(X_i|Z_{i,j}=1,\theta)$ because knowing $Z_{i,j}$ or Q_i makes the sequence X_i independent of γ, and we assume W is constant. We have also rearranged the terms to separate those which depend on θ from those which depend on γ (directly or via $\lambda = \gamma/m$).

We now work out the expectation of the joint log likelihood. In the final step of the derivation, we make the substitution

$$\begin{aligned}
Q_i^{(t)} &= \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Q_i] \\
&= \sum_{j=1}^{m} \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z_{i,j}] \\
&= \sum_{j=1}^{m} Z_{i,j}^{(t)}, \quad i = 1,\ldots,n,
\end{aligned}$$

as well as the substitution of $Z_{i,j}^{(t)}$ that we made in the case of an OOPS model in Equation IV.4. The expectation of the joint log likelihood is

$$\begin{aligned}
&\mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[\log Pr(X, Z|\phi)] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z_{i,j}] \log Pr(X_i|Z_{i,j}=1,\theta) \\
&\quad + \sum_{i=1}^{n}(1 - \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Q_i]) \log Pr(X_i|Q_i=0,\theta) \\
&\quad + \sum_{i=1}^{n} \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Q_i] \log\lambda + \sum_{i=1}^{n}(1 - \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Q_i]) \log(1-\gamma) \\
&= \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)} \log Pr(X_i|Z_{i,j}=1,\theta) + \sum_{i=1}^{n}(1-Q_i^{(t)})\log Pr(X_i|Q_i=0,\theta)}_{term_1}
\end{aligned}$$

$$\underbrace{+\sum_{i=1}^{n} Q_i^{(t)} \log \lambda + \sum_{i=1}^{n} (1 - Q_i^{(t)}) \log(1-\gamma)}_{term_2} \tag{IV.10}$$

Observe that the expected log likelihood can be viewed as the sum of two terms. The first term depends only on θ. The second term depends only on γ since $\lambda = \frac{\gamma}{m}$. We can thus maximize the expected log likelihood separately over θ and γ. In order to do this, however, we will first need to know the forms of the log conditional probabilities $\log Pr(X_i|Z_{i,j} = 1, \theta)$ and $\log Pr(X_i|Q_i = 0, \theta)$.

The conditional sequence probabilities

Knowing the starting position of the motif makes the probability of X_i independent of γ. So the calculation of $\log Pr(X_i|Z_{i,j} = 1, \phi)$ is exactly as given for an OOPS model in equation (IV.5).

The logarithm of the probability of sequence X_i given that it contains no occurrence of the motif (i.e., given $Q_i = 0$) can be expressed as

$$\begin{aligned} \log Pr(X_i|Q_i = 0, \phi) &= \log Pr(X_i|Q_i = 0, \theta) \\ &= \log \prod_{a \in \mathcal{L}} (P_{a,0})^{t_{i,a}} \\ &= \sum_{a \in \mathcal{L}} t_{i,a} \log P_{a,0} \\ &= \mathbf{t}_i^T \log \mathbf{p}_0. \end{aligned} \tag{IV.11}$$

Like $Pr(X_i|Z_{i,j} = 1, \phi)$, $Pr(X_i|Q_i = 0, \phi)$ is a linear function of $\log \theta$.

We are now ready to solve for the values of θ and γ which maximize the expected joint log likelihood given in equation (IV.10).

Maximizing over θ

The first term of equation (IV.10) can be expressed as a linear combination of the logarithms of the $\mathbf{p}$ vectors making up θ since (IV.5) and (IV.11) are linear functions of the logarithms of the $\mathbf{p}$ vectors. This means that we will be

able to express the first term of (IV.10) as

$$term_1 = \sum_{k=0}^{W} \mathbf{c}_k^T \log \mathbf{p}_k$$

In particular, substituting (IV.5) into (IV.10),

$$\begin{aligned}
term_1 &= \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)} \log Pr(X_i | Z_{i,j} = 1, \theta) \\
&\quad + \sum_{i=1}^{n} (1 - Q_i^{(t)}) \log Pr(X_i | Q_i = 0, \theta) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)} (\sum_{k=1}^{W} \mathbf{I}(i, j+k-1)^T \log \mathbf{p}_k + \sum_{k \in \Delta_{i,j}} \mathbf{I}(i,k)^T \log \mathbf{p}_0) \\
&\quad + \sum_{i=1}^{n} (1 - Q_i^{(t)}) \mathbf{t}_i^T \log \mathbf{p}_0 \\
&= \sum_{k=1}^{W}\sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)} \mathbf{I}(i, j+k-1)^T \log \mathbf{p}_k \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k \in \Delta_{i,j}} Z_{i,j}^{(t)} \mathbf{I}(i,k)^T \log \mathbf{p}_0 \;+\; \sum_{i=1}^{n} (1 - Q_i^{(t)}) \mathbf{t}_i^T \log \mathbf{p}_0 \\
&= \sum_{k=1}^{W} [\sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)} \mathbf{I}(i, j+k-1)]^T \log \mathbf{p}_k \\
&\quad + \sum_{i=1}^{n} [\sum_{j=1}^{m}\sum_{k \in \Delta_{i,j}} Z_{i,j}^{(t)} \mathbf{I}(i,k) + (1 - Q_i^{(t)}) \mathbf{t}_i]^T \log \mathbf{p}_0 \\
&= \sum_{k=0}^{W} \mathbf{c}_k^T \log \mathbf{p}_k \qquad \text{(IV.12)}
\end{aligned}$$

where

$$\mathbf{c}_k = \begin{cases} \sum_{i=1}^{n} (\sum_{j=1}^{m} \sum_{b \in \Delta_{i,j}} Z_{i,j}^{(t)} \mathbf{I}(i,b) + (1 - Q_i^{(t)}) \mathbf{t}_i) & \text{if } k = 0 \\ \sum_{i=1}^{n} \sum_{j=1}^{m} Z_{i,j}^{(t)} \mathbf{I}(i, j+k-1) & \text{if } 1 \le k \le W. \end{cases}$$

Notice that $\mathbf{c}_k$ are the expected counts of letters in the background and positions in the motif.

We can simplify the computation of $\mathbf{c}_0$, the background expected letter counts, in the same way as for an OOPS model. The resulting formula turns out to be the same as well, with

$$\mathbf{c}_0 = \sum_{i=1}^{n} (\sum_{j=1}^{m} \sum_{b \in \Delta_{i,j}} Z_{i,j}^{(t)} \mathbf{I}(i,b) + (1 - Q_i^{(t)}) \mathbf{t}_i)$$

$$
\begin{aligned}
&= \sum_{i=1}^{n}(\sum_{j=1}^{m}(Z_{i,j}^{(t)}\mathbf{t}_i - Z_{i,j}^{(t)}\sum_{b=1}^{W}\mathbf{I}(i,j+b-1)) + (1-Q_i^{(t)})\mathbf{t}_i) \\
&= \sum_{i=1}^{n}(Q_i^{(t)}\mathbf{t}_i - \sum_{j=1}^{m}Z_{i,j}^{(t)}\sum_{b=1}^{W}\mathbf{I}(i,j+b-1) + \mathbf{t}_i - Q_i^{(t)}\mathbf{t}_i) \\
&= \sum_{i=1}^{n}(\mathbf{t}_i - \sum_{j=1}^{m}\sum_{b=1}^{W}Z_{i,j}^{(t)}\mathbf{I}(i,j+b-1)) \\
&= \mathbf{t} - \sum_{k=1}^{W}\mathbf{c}_k.
\end{aligned}
$$

So, the formula for $\mathbf{c}_k$ becomes exactly like equation (IV.8).

It is now be clear that maximization over θ for an ZOOPS model is exactly the same as for an OOPS model. As before with an OOPS model, we can maximize over $\mathbf{p}$ by independently maximizing over each $\mathbf{p}_k$. The new estimate of θ, $\theta^{(t)}$, is

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k}{|\mathbf{c}_k|},\ 0 \le k \le W.$$

Once again, these are just the expected frequencies of the letters in the background and in each position of the motif.

Maximizing over γ

The second term of (IV.10) makes it easy to see how to maximize the expected value of the joint likelihood with respect to γ. If we let

$$S = \frac{1}{n}\sum_{i=1}^{n}Q_i^{(t)}$$

we can rewrite $term_2$ as the product of a probability vector and the log of a probability vectors plus a constant term. We can use the same result from information theory as before, Equation IV.9, to arrive at the maximizing value of γ from

$$
\begin{aligned}
term_2 &= \sum_{i=1}^{n}(Q_i^{(t)}\log\lambda + (1-Q_i^{(t)})\log(1-\gamma)) \\
&= nS\log\lambda + n(1-S)\log(1-\gamma) \\
&= n(S\log\frac{\gamma}{m} + (1-S)\log(1-\gamma))
\end{aligned}
$$

$$
\begin{aligned}
&= n(S \log \gamma + (1 - S) \log(1 - \gamma) - S \log m). \\
&= n \begin{bmatrix} S \\ 1 - S \end{bmatrix}^T \log \begin{bmatrix} \gamma \\ 1 - \gamma \end{bmatrix} + nS \log m.
\end{aligned}
$$

It is apparent that the maximizing value for γ in equation (IV.10) is

$$
\begin{aligned}
\gamma^{(t+1)} &= S \\
&= \frac{1}{n} \sum_{i=1}^{n} Q_i^{(t)}.
\end{aligned}
$$

IV.E EM and the TCM model

The TCM random process models a sequence as the concatenation of samples from a two-component finite mixture. One component, the background, generates one letter at a time; the other component, the motif, generates a string of length W.

The parameters for the TCM model are $\phi = (\theta, \lambda, W)$ where θ determines the probabilities of each letter in the background and in each of the W positions in the motif, and λ is the probability of selecting the motif component at each step of the random process. We do not use EM to learn ϕ directly. Rather, we convert the dataset X into a new pseudo-dataset consisting of all the width W overlapping subsequences by running a window of width W along each sequence X_i and writing down the string contained in the window. We then model this new dataset as though it were generated by a two-component mixture model where *each component generates a string of width* W, and the mixing parameter is λ'. We constrain the first component, which represents the background positions in the dataset, to have only one independent column. And we apply a smoothing step after each E-step of EM to reduce the degree to which any two overlapping subsequences can both be assigned to the motif component. This smoothing step makes the model of the pseudo-dataset closely approximate a TCM model of the original dataset.

We use EM to fit the parameters of the modified TCM model, whose underlying random process generates a succession of strings of width W, to data consisting of all the overlapping width-W substrings in the original sequences. The fitted parameters of the modified TCM model are then used as the estimate of the parameters ϕ of the original TCM model except the estimate of λ is

$$\lambda = \frac{1}{(1/\lambda') - W + 1}.$$

This value of λ makes sense for the following reason. If a given sequence was generated by a TCM random process with mixing parameter λ in, say, a steps, the expected number of motif occurrences in the sequence is $a\lambda$. The other $a(1-\lambda)$ steps generated a single letter each, so the length of the sequence is $a\lambda W + a(1-\lambda)$. The fraction of positions in the sequence which are the start of a motif occurrence, λ' is

$$\lambda' = \frac{a\lambda}{a(\lambda(W-1)+1)}.$$

Solving this for λ gives the result stated above.

The following derivations are for the modified TCM model and the pseudo-dataset. For the sake of notational simplicity, we write λ instead of λ', with the understanding that the value of λ for the original TCM model can be computed as described in the previous paragraph. Also for notational convenience, in what follows $X_{i,j}$ will refer to the width-W subsequence starting at position j in sequence X_i in the original dataset, not the single letter at position j.

As with OOPS and ZOOPS models, we will hold the value of W constant and use EM to maximize the expected joint log likelihood of the observed and missing data. In the E-step, we will show that the probability that a (width-W pseudo-)sequence X_ij was generated by the motif, given the current estimate of ϕ, $\phi^{(t)} = (\theta^{(t)}, \lambda^{(t)}, W)$, is

$$Z_{i,j}^{(t)} = \frac{Pr(X_{i,j}|\theta_1^{(t)})\lambda^{(t)}}{Pr(X_{i,j}|\theta_0^{(t)})(1-\lambda^{(t)}) + Pr(X_{i,j}|\theta_1^{(t)})\lambda^{(t)}}.$$

```
procedure SMOOTH (W, Z)
  for i = 1 to n do
    for s = 1 to W do
      for k = s to m by W do
        Compute z = Σ_{j=k}^{k+W-1} Z_{i,j}, the sum of Z_{i,j} in the current window.
        Find z_max = max_{k≤j<k+W} Z_{i,j}, the largest Z_{i,j} in current window.
        if z > 1.0
          Compute scale = (1 − z_max)/(z − z_max).
          Multiply all but largest value of Z_{i,j} in the window by scale.
        end
      end
    end
  end
end
```

Figure IV.4: **The** SMOOTH **algorithm.**

Then, the expected value of the number of times each letter appears at each background or motif position will be shown to be

$$\mathbf{c}_k = \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{W}(1 - Z_{i,j}^{(t)})\mathbf{I}(i, j+l-1) & \text{if } k = 0 \\ \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)}\mathbf{I}(i, j+k-1) & \text{if } 1 \le k \le W. \end{cases}$$

Finally, the M-step of EM for the TCM model will use the expected letter frequencies,

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k}{|\mathbf{c}_k|},$$

as the new estimate for θ, and

$$\lambda^{(t+1)} = \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{Z_{i,j}^{(t)}}{nm}$$

as the new estimate for λ.

IV.E.1 The E-step for TCM

We make the simplifying assumption that all the substrings $X_{i,j}$ are independent of each other. Of course, this is not true since many substrings overlap.

The problems caused by this inaccurate assumption are dealt with by smoothing the values of $Z_{i,j}$. The main problem is that we do not want the model to predict that two overlapping substrings are both motif occurrences. To prevent this, the $Z_{i,j}$ values for overlapping $X_{i,j}$ substrings are reduced so that the sum of $Z_{i,j}$ in any window of width W is always less than or equal to 1. This is done in a winner-take-all fashion where the largest value of $Z_{i,j}$ in a window is never reduced. The smoothing algorithm is sketched in Figure IV.4.

For the E-step of EM we must find the expected value of the missing data, $Z_{i,j}^{(t)}$. This we do using the definition of expectation and Bayes' rule which gives,

$$\begin{aligned} Z_{i,j}^{(t)} &= \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z_{i,j}] \\ &= 0 \cdot Pr(Z_{i,j} = 0|X_{i,j}, \phi^{(t)}) + 1 \cdot Pr(Z_{i,j} = 1|X_{i,j}, \phi^{(t)}) \\ &= Pr(Z_{i,j} = 1|X_{i,j}, \phi^{(t)}) \\ &= \frac{Pr(X_{i,j}|Z_{i,j} = 1, \phi^{(t)})Pr(Z_{i,j} = 1|\phi^{(t)})}{Pr(X_{i,j}|\phi^{(t)})} \text{ (by Bayes' rule)} \\ &= \frac{Pr(X_{i,j}|\theta_1^{(t)})\lambda^{(t)}}{Pr(X_{i,j}|\theta_0^{(t)})(1-\lambda^{(t)}) + Pr(X_{i,j}|\theta_1^{(t)})\lambda^{(t)}}. \end{aligned} \tag{IV.13}$$

Equation (IV.13) has the nice intuitive interpretation that the probability of a subsequence $X_{i,j}$ having been generated by the motif is the ratio of its probability given the motif model to its probability given the complete model.

IV.E.2 The M-step for TCM

Once again we wish to solve

$$\phi^{(t+1)} = \operatorname*{argmax}_{\phi} \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[\log Pr(X, Z|\phi)].$$

We write down the joint likelihood, $Pr(X, Z|\phi)$, in a way that simplifies taking logarithms and separating the parts which depend on θ from those which depend

on λ. Assuming the independence of $X_{i,j}$, we can write

$$\begin{aligned} Pr(X,Z|\phi) &= \prod_{i=1}^{n}\prod_{j=1}^{m} Pr(X_{i,j},Z_{i,j}|\phi) \\ &= \prod_{i=1}^{n}\prod_{j=1}^{m} Pr(X_{i,j}|Z_{i,j},\phi)Pr(Z_{i,j}|\phi) \\ &= \prod_{i=1}^{n}\prod_{j=1}^{m} Pr(X_{i,j}|Z_{i,j}=0,\phi)^{1-Z_{i,j}}Pr(X_{i,j}|Z_{i,j}=1,\phi)^{Z_{i,j}} \\ &\qquad Pr(Z_{i,j}=0|\phi)^{1-Z_{i,j}}Pr(Z_{i,j}=1|\phi)^{Z_{i,j}} \\ &= \prod_{i=1}^{n}\prod_{j=1}^{m} Pr(X_{i,j}|\theta_0)^{1-Z_{i,j}}Pr(X_{i,j}|\theta_1)^{Z_{i,j}}(1-\lambda)^{1-Z_{i,j}}\lambda^{Z_{i,j}}. \end{aligned}$$

Notice that we have again used the trick of rewriting probabilities taking advantage of the fact that $Z_{i,j}$ is always either 0 or 1. Notice also that the probability of the subsequence $X_{i,j}$ depends only on θ once $Z_{i,j}$ is known.

Next we calculate the logarithm of the joint likelihood as

$$\begin{aligned} &\log Pr(X,Z|\phi) \\ &\quad= \sum_{i=1}^{n}\sum_{j=1}^{m}((1-Z_{i,j})\log Pr(X_{i,j}|\theta_0) + Z_{i,j}\log Pr(X_{i,j}|\theta_1)) \\ &\qquad+\sum_{i=1}^{n}\sum_{j=1}^{m}((1-Z_{i,j})\log(1-\lambda) + Z_{i,j}\log\lambda). \end{aligned} \tag{IV.14}$$

We finish the calculation of the expected joint log likelihood by taking the expectation of equation (IV.14) and substituting $Z^{(t)}$ giving

$$\begin{aligned} &\mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[\log Pr(X,Z|\phi)] \\ &\quad= \sum_{i=1}^{n}\sum_{j=1}^{m}(\mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[1-Z_{i,j}]\log Pr(X_{i,j}|\theta_0) + \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z_{i,j}]\log Pr(X_{i,j}|\theta_1)) \\ &\qquad+\sum_{i=1}^{n}\sum_{j=1}^{m}(\mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[1-Z_{i,j}]\log(1-\lambda) + \mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[Z_{i,j}]\log\lambda) \\ &\quad= \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{m}((1-Z_{i,j}^{(t)})\log Pr(X_{i,j}|\theta_0)) + \sum_{i=1}^{n}\sum_{j=1}^{m}(Z_{i,j}^{(t)}\log Pr(X_{i,j}|\theta_1))}_{term_1} \end{aligned}$$

$$\underbrace{+\sum_{i=1}^{n}\sum_{j=1}^{m}((1-Z_{i,j}^{(t)})\log(1-\lambda)+Z_{i,j}^{(t)}\log\lambda)}_{term_2}. \quad \text{(IV.15)}$$

To complete the M-step, we need to maximize equation (IV.15) with respect to λ and θ. The maximizing value of λ can be seen by inspection of $term_2$ to be

$$\lambda^{(t)} = \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{Z_{i,j}^{(t)}}{nm}.$$

The maximizing values of θ_0 and θ_1 depend on the conditional (sub)sequence probabilities $Pr(X_{i,j}|\theta_0)$ and $Pr(X_{i,j}|\theta_1)$. These have the forms

$$\begin{aligned} Pr(X_{i,j}|\theta_0) &= \sum_{a\in\mathcal{L}}\sum_{k=1}^{W} P_{a,0}^{I(i,j+k-1,a)} \text{ and} \\ Pr(X_{i,j}|\theta_1) &= \sum_{a\in\mathcal{L}}\sum_{k=1}^{W} P_{a,k}^{I(i,j+k-1,a)}, \end{aligned}$$

since the model assumes that subsequence $X_{i,j}$ was generated by sampling either W times from M_0 or once each (in order) from $M_1,\ldots,M_W$.

The logarithms of the conditional sequence probabilities can be simplified in the same way as was done for the other two models to

$$\log Pr(X_{i,j}|\theta_p) = \sum_{k=1}^{W}\mathbf{I}(i,j+k-1)^T\log\mathbf{p}_{k'}, \quad \text{(IV.16)}$$

where $k'=0$ if $p=0$, and $k'=k$ otherwise.

Substituting Equation IV.16 into $term_1$ of the expected joint log likelihood equation (IV.15) and rewriting, we observe that the new estimate for θ should be

$$\begin{aligned} \theta^{(t)} &= \underset{\theta}{\text{argmax}}\Big(\sum_{i=1}^{n}\sum_{j=1}^{m}(1-Z_{i,j}^{(t)})\sum_{k=1}^{W}\mathbf{I}(i,j+k-1)^T\log\mathbf{p}_0 \\ &\quad +\sum_{i=1}^{n}\sum_{j=1}^{m}Z_{i,j}^{(t)}\sum_{k=1}^{W}\mathbf{I}(i,j+k-1)^T\log\mathbf{p}_k\Big), \\ &= \underset{\theta}{\text{argmax}}\sum_{k=0}^{W}\mathbf{c}_k\log\mathbf{p}_k, \end{aligned}$$

where

$$\mathbf{c}_k = \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{W}(1 - Z_{i,j}^{(t)})\mathbf{I}(i, j+l-1) & \text{if } k = 0 \\ \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j}^{(t)}\mathbf{I}(i, j+k-1) & \text{if } 1 \le k \le W \end{cases}.$$

So, once again, the new estimate for θ is defined by the expected letter frequencies

$$\mathbf{p}_k^{(t+1)} = \frac{\mathbf{c}_k}{|\mathbf{c}_k|}.$$

IV.F Likelihood of the null model

This section shows the derivation of the formula for the likelihood of the null model that is used in computing the LRT-based heuristic function $G(\phi)$. The universal null model for each of the types of models supported by MEME predicts that the entire dataset contains no occurrences of any motif. If this were the case, then λ, the proportion of motif starts to possible motif starts in the dataset would be zero. As a result, the expected letter frequencies in the background, θ_0, would just be the average letter frequencies in the dataset, μ.

For the purposes of the LRT, we need to calculate the log likelihood of the null model for the model types supported by MEME. To do this, we reformulate the equations for the expected joint log likelihoods of the different models and substitute the appropriate values for θ, λ and W. It is easy to show that the expected joint log likelihoods given in equations (IV.6), (IV.10) and (IV.15) can be rewritten in terms of the entropy of each column in θ. This will simplify calculating the likelihoods for both the null model and the model found by EM. The value of $\mathop{\mathrm{E}}_{(Z|X,\phi^{(t)})}[\log Pr(X, Z|\phi)]$ when EM has converged (so that $\phi^{(t)} \approx \phi^{(t+1)}$) will be referred to as $ell_{TYPE}(\phi)$ where TYPE is the type of model. This is, for the three model types,

$$\begin{aligned} ell_{OOPS}(\phi) &= n((L-W)E_0 + (WE_1) - \log m) \\ ell_{ZOOPS}(\phi) &= (nL - sW)E_0 + sWE_1 + (n-s)\log(1-\gamma) + s\log\lambda \\ ell_{TCM}(\phi) &= (1-\lambda)(\log(1-\lambda) + WE_0) + \lambda(\log\lambda + WE_1), \end{aligned}$$

where E_i is the average entropy per column of θ_i, and s is the predicted number of motif occurrences. That is,

$$E_i = \begin{cases} \sum_{a \in \mathcal{L}} P_{a,0} \log P_{a,0} & \text{if } i = 0 \\ \frac{1}{W} \sum_{a \in \mathcal{L}} \sum_{j=1}^{W} P_{a,k} \log P_{a,k} & \text{if } i = 1 \end{cases}$$

and

$$s = nm\lambda.$$

Plugging in the assumptions that $W = 1$, $\theta_1 = \theta_0 = \mu$ and $\lambda = 0$, it is easy to show that the expected joint log likelihood for the null model will then be estimated by

$$\begin{aligned} ell_{OOPS}(\phi_0) &= nLH_0 - \log L \\ ell_{ZOOPS}(\phi_0) &= nLH_0 \\ ell_{TCM}(\phi_0) &= nLH_0 \end{aligned}$$

where H_0 is the total entropy of the dataset,

$$H_0 = \sum_{a \in \mathcal{L}} \mu_a \log \mu_a.$$

MEME actually uses the equation for ell_{ZOOPS} when computing the expected log likelihood of an OOPS model. This is done because the simulation of multiple component mixture models done by MEME (see Section III.D.1) in finding multiple motifs may cause an OOPS model to predict fewer than n occurrences of motifs found on the second and subsequent passes. That is, the sum of $Z_{i,j}$ for some sequence X_i may not be equal to 1. In this case, the assumptions of an OOPS model no longer strictly hold and the equation for ell_{OOPS} will tend to yield a value that is unreasonably large. In this case, the situation is similar to that of a ZOOPS model, so the formula for its expected log likelihood is used instead.

When the assumptions of the OOPS model are satisfied and the sum of $Z_{i,j}$ is 1 for each sequence X_i, it can be shown that $ell_{ZOOPS}(\phi) = ell_{OOPS}(\phi) + \log m$. In practice the width of the motif W is usually much less than the length of the sequences L so $m \approx L$. Using ell_{ZOOPS} instead of ell_{OOPS} has virtually no effect

on the outcome of the LRT which depends only on the difference in likelihoods since

$$\begin{aligned} ell_{OOPS}(\phi) - ell_{oops}(\phi_0) &= ell_{OOPS}(\phi) - (nLH_0 - \log L) \\ &\approx ell_{OOPS}(\phi) - (nLH_0 - \log m) \\ &= ell_{ZOOPS}(\phi) - ell_{ZOOPS}(\phi_0). \end{aligned}$$

As a result, MEME uses one equation for all three models in computing the log likelihood of the null model,

$$ell(\phi_0) = nLH_0.$$

Chapter V

Results

This chapter presents the results of experiments that demonstrate that MEME successfully solves a large number of biological motif discovery problems. It shows that MEME can indeed find multiple motifs in both DNA and protein datasets. Depending on the type of motif model used, the motifs characterize all of the sequences in the dataset (OOPS, ZOOPS, or TCM), or subfamilies of sequences within the dataset (ZOOPS or TCM), or elements which repeat within individual sequences and among sequences (TCM).

With both DNA and protein datasets, MEME discovers motifs which make sense biologically, and which generalize well.

- Known biological motifs are usually found on the first passes of MEME even when the dataset only contains half the known examples.
- The widths chosen by MEME for known motifs correspond well with human-determined widths and motif occurrences identified by MEME are usually centered on the known motif occurrences.
- With families of related protein sequences, the motif found by MEME on the first pass is almost always an excellent classifier for other (unseen) members of the family.

- In sets of DNA sequences, MEME correctly detects and reports motifs which are DNA palindromes.
- The execution time of MEME is quadratic in the size of the dataset, as predicted by the theoretical complexity analysis in Section III.H.2.

This chapter also describes experiments which show that MEME benefits from, but does not require, three sources of background information which can be provided by the user:

- knowledge of which sequence model type is appropriate;
- informative prior distributions on the model parameters; and
- a fixed motif width known in advance.

A study comparing MEME with another motif discovery method and a case study conducted in conjunction with a biologist which demonstrates the value of MEME, PROBER and NOTE for studying protein families show that MEME performs as well or better and is more flexible than the two best-known automatic motif discovery methods. This flexibility and power comes, in part, from the ability of MEME to incorporate sources of background knowledge from the problem domain.

The chapter is organized as follows. Section V.A describes the general methodology used in this thesis for testing MEME and other motif discovery methods. It also describes the datasets used in developing and testing MEME and the techniques used for measuring the different aspects of the performance of MEME motifs. The results of experiments to measure the biological relevance and generalization accuracy of motifs are discussed in Section V.B. Section V.C compares MEME with another motif discovery algorithm. Section V.D is a case study using MEME, PROBER and NOTE to shed new light on a biologically and medically important family of proteins.

V.A Experimental methods

The experiments described in this chapter fall into two categories. In the first category, the objective is to measure the biological relevance of motifs discovered by MEME.[1] To this end, I measure the agreement between the discovered motifs and the known motifs contained in the training set. This is done by measuring how well the discovered motifs predict the locations and widths of the known motif occurrences. I also consider the pass on which MEME discovers the known motif (hopefully the first pass) and whether or not the motif was correctly identified as a DNA palindrome. In the second category of experiment, the objective is to see how well motifs can be expected to generalize. In order to do this, the algorithm is allowed to discover only one motif and then the motif is used to classify sequences not in the training set in a form of cross-validation (CV) [Efron, 1983]. In this type of experiment it is of no concern whether the discovered motif resembles a known motif, only whether it correctly discriminates between members of the training set family of sequences and non-members. This kind of experiment provides an objective measure of the predictive accuracy of motifs.

The quantities measured in the experiments described in this chapter and how they are measured are described in Section V.A.2. For the biological relevance experiments, the measured quantities are:

- ROC, recall, precision
- motif width,
- motif shift,
- MEME pass number on which the known motif is found, and
- whether a DNA palindrome is correctly detected.

[1] The same methodology is used for measuring the performance of the other motif discovery system tested in Section V.C.

In the generalization experiments presented here, only ROC, recall and precision are measured. Because I am not concerned with whether or not a known motif was found, only one motif is searched for in the generalization experiments.

V.A.1 Sequence datasets

I studied the behavior of MEME with the help of two groups of sequence datasets. The algorithm was refined using a set of seven datasets referred to below as the development datasets. The development datasets were chosen to represent a variety of difficult DNA and protein motif discovery problems. The group includes datasets with strong and faint DNA palindromic motifs separately as well as in the same dataset. It also includes protein datasets with multiple, faint motifs, a single faint motif, or multiple faint motifs that repeat several times in each sequence. Further testing of the capabilities of MEME was carried out using 75 protein datasets each of which contains one or more known motifs. These datasets are derived from the Prosite dictionary of protein sites and patterns [Bairoch, 1993] and are referred to as the Prosite datasets in what follows. They were selected because the motifs they contain were known to be difficult to characterize. The development and Prosite datasets are described in detail below.

Development datasets

The protein datasets lip, hth, and farn, were created by Lawrence *et al.* [1993] and used to test their Gibbs sampling algorithm. Very briefly, the lip dataset contains the five most divergent lipocalins with known 3D structure. They contain two known motifs, each occurring once in each sequence. The positions of the two motifs in each of the sequences in the lip dataset are known from structural comparisons. The hth proteins contain DNA-binding features involved in gene regulation. The correct locations of occurrences of the hth motif are known from x-ray and nuclear magnetic resonance structures, or from substitution mutation

name	*type*	*N*	*L*	*W*	*sites*	
					proven	*total*
lip	protein	5	182	16	5	5
					5	5
hth	protein	30	239	18	30	30
farn	protein	5	380	12	0	30
					0	26
					0	28
crp	DNA	18	105	20	18	24
lex	DNA	16	200	20	11	21
crplex	DNA	34	150	20	18	25
					11	21
hrp	DNA	231	58	29	231	231

Table V.1: **Overview of the datasets used in developing** MEME**.** The table shows sequence type, number of sequences (N), average sequence length (L), and motif width (W). Proven sites have been shown to be occurrences of the motif by laboratory experiment (footprinting, mutagenesis, or structural analysis). Total sites include the proven sites and sites reported in the literature based primarily on sequence similarity with known sites.

quantity	*mean*	(*sd*)
sequences per dataset	34	(36)
dataset size	12945	(11922)
sequence length	386	(306)
shortest sequence	256	(180)
longest sequence	841	(585)
pattern width	12.45	(5.42)

Table V.2: **Overview of the 75 Prosite datasets.** Each dataset contains all protein sequences in SWISS-PROT annotated in the Prosite database as true positives or false negatives for the Prosite pattern characterizing a given family. Dataset size and sequence length count the total number of amino acids in the protein sequences.

experiments, or both. The farn dataset contains isoprenyl-protein transferases, each with multiple appearances of three motifs. No direct structural information is known for the proteins in the dataset, so I used the starting positions for the three motifs reported by Lawrence *et al.* [1993]. These starting positions agreed with the results of earlier sequence analysis work [Boguski *et al.*, 1992a], [Boguski *et al.*, 1992b].

The *E. coli* DNA datasets, crp, lex and crplex, contain known examples of two types of protein binding sites. The crp sequences contain binding sites for cyclic-AMP receptor protein (CRP) [Lawrence and Reilly, 1990], while the lex sequences contain binding sites for LexA; the crplex dataset is the union of the crp and lex datasets. The CRP dataset is taken from [Stormo and Hartzell, III, 1989] who, in turn, derived it from [Berg and von Hippel, 1988] and [de Crombrugghe *et al.*, 1984]. It contains 18 DNA fragments from *E. coli* each believed to contain one or more CRP binding sites. The dataset contains 18 CRP binding sites which had been verified by DNase protection experiments when the dataset was compiled. Some of the fragments contain putative CRP binding sites which have been determined by homology only. Each fragment in the dataset contains 105 bases. The LexA dataset is taken from Table I in [Hertz *et al.*, 1990]. It contains 16 DNA fragments each believed to contain one or more LexA binding sites. The dataset contains 11 LexA binding sites which had been verified by DNase protection experiments when the dataset was compiled. An additional 11 putative LexA binding sites, as determined by homology, are also present in the dataset. Most of the fragments contain 100 bases preceding and 99 bases following the transcription start position of a gene. Three of the fragments are shorter because 200 bases flanking the start position of the gene were not available. One of the samples in the LexA dataset overlaps a sample in the CRP dataset. The overlap includes the known CRP site.

The *E. coli* promoter dataset hrp [Harley and Reynolds, 1987] contains

a single motif which consists of two submotifs with a varying number of positions (usually about 17) between them. This dataset was originally compiled by Harley and Reynolds [1987], and contained 288 fragments, but Cardon and Stormo omitted a number of fragments that were from highly redundant sequences or known to be mutant promoters. All the fragments roughly comprise positions -50 to $+10$ with respect to the start of the known sites and consist of two highly conserved ends of width 6 separated by a "spacer" of variable length and highly variable sequence. The length of the spacer is usually 16, 17 or 18 base-pairs (bp), so the total width of the motif is about 29.

Prosite datasets

The 75 Prosite families described in general terms in Table V.2 correspond approximately to the 10% of fixed-width Prosite patterns with worst combined (summed) recall and precision. Prosite patterns are called *signatures*. Their interpretation is given in Appendix B. Fixed-width patterns such as

```
D-[SGN]-D-P-[LIVM]-D-[LIVMC]
```

are a proper subset of the patterns expressible by MEME motifs, and they form a majority in Prosite.[2] Recall and precision for Prosite signatures and for corresponding MEME motifs are calculated using information in the Prosite database about matches found when searching the large (36000 sequence) SWISS-PROT database of protein sequences [Bairoch, 1994]. Detailed descriptions of the Prosite families and their Prosite signatures are shown in Appendix B in Tables B.1, B.2, B.3, B.5 and B.6.

[2] I used Prosite release 11.1 (February 1994) which contains 1021 signatures of which 815 are of fixed width. The other signatures allow variable spacing between some or all positions in the signature.

V.A.2 Measuring performance

I conducted experiments to evaluate MEME in terms of how well it discovers motifs in groups of sequences. The two primary criteria for determining how well MEME discovers motifs are

- how well the motifs discovered correspond to known biological sequence motifs, and
- how well the motifs characterize membership in the family of sequences.

These criteria correspond to the two primary reasons for discovering motifs in groups of sequences—understanding the structural, functional and evolutionary relationships among sequences in a family, and searching for new family members. I measured the agreement of MEME motifs with known motifs by looking at how well the MEME motifs predicted the positions of the known motif occurrences in the training set and how well the motif width selected by MEME matched the known width. The degree to which a motif characterizes a family of sequences was measured by seeing how well the motif discriminated the sequences in the family it was learned on from other sequences known not to belong the to the family. Both types of performance measurement required using motifs as classifiers in the way described below.

Evaluating classification accuracy

I measured the performance of the motifs discovered by MEME by using the final sequence model output after each pass as a classifier. The parameters, ϕ, of the sequence model discovered on a particular pass are converted by MEME into a log-odds scoring matrix LO and a threshold t where $LO_{x,j} = \log(p_{x,j}/p_{x,0})$ for $j = 1, \ldots, W$ and $x \in \mathcal{L}$, and $t = \log((1-\lambda)/\lambda)$.[3] The scoring matrix and

[3] See Section III.B for an explanation of the notation used by MEME for sequence model parameters.

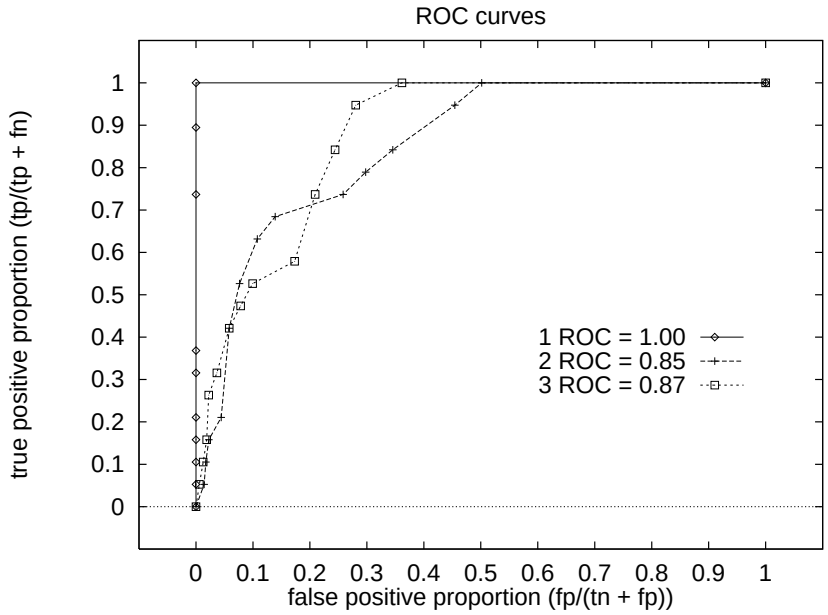


Figure V.1: **Example ROC curves.**

threshold was used to score the sequences in a test set of sequences for which the positions of motif occurrences are known. Each subsequence whose score using LO as a position-dependent scoring matrix exceeds the threshold t is considered a hit. For each known motif in the test set, the positions of the hits are compared to the positions of the known occurrences. The number of true positive (tp), false positive (fp), true negative (tn) and false negative (fn) hits was tallied. From these, recall $= tp/(tp + fn)$ and precision $= tp/(tp + fp)$ were computed.

I also calculated the receiver operating characteristic (ROC) [Swets, 1988] of the MEME motifs. The ROC statistic is the integral of the ROC curve, which plots the true positive proportion, $tpp = recall = tp/(tp + fn)$, versus the false positive proportion, $fpp = fp/(fp + tn)$. The ROC statistic was calculated by scoring all the positions in the test set using the log-odds matrix, LO, sorting the positions by score, and then numerically integrating tpp over fpp using the trapezoid rule. Sample ROC curves are shown in Figure V.1. The decimal numbers in the legend show the values of the ROC statistic for each curve.

MEME motifs which were shifted versions of a known motif were detected

by shifting all the known motif positions left or right the same number of positions and repeating the above calculations of recall, precision and ROC. All shifts such that all predicted occurrences overlap the known occurrences (by exactly the same amount) were tried. The performance values reported are those for the best shift. For datasets with multiple known motifs, recall, precision and ROC were calculated separately for each known motif using each of the sequence models discovered during the passes of MEME.

Evaluating motif width

The question of whether an algorithm such as MEME chooses the correct width for a motif is a difficult one since "correct" may not be well defined. In some cases, physical occurrences of a motif may have a fairly well defined length which defines the width of the motif. For example, a motif describing a protein-binding site on a DNA molecule may have a well defined width—the width of the DNA sequence which actually contacts the binding protein. Some protein motifs such as the active sites of enzymes may also have a fairly well defined width determined by the portion of the protein sequence (i.e., the enzyme) which actually makes contact with the substrate (i.e., the other molecules which are caused to react by the enzyme.) For motifs such as these it may be possible to define the meaning of correct motif width and measure it in the laboratory. The tests described in the following sections used published values for motif widths based on laboratory data for the development datasets and the width of a human generated motif for the Prosite datasets (see Section V.A.1).

V.B Discovering motifs

V.B.1 Biological relevance

This section describes experiments that show that MEME finds biologically relevant motifs. With the development datasets, MEME finds all the known motifs on the very first passes. That is, in each development dataset with n known motifs, the first n passes of MEME find them. The widths of the motifs found in the development datasets are, for the most part, close to the known widths, and the positions of the predicted motif occurrences are only slightly shifted from the known ones. MEME correctly identifies all of the DNA palindromic motifs without making any false predictions. With the Prosite datasets, MEME discovers 88% of the known motifs within its first five passes, discovering 84% within the first three passes. The average ROC (0.99) of these motifs at identifying the known motif occurrences is nearly perfect, and their widths are mostly within a factor of two of the human-determined widths. Finally, the ability of MEME to find the biologically relevant motifs in the Prosite protein datasets is virtually unaffected by removing half of the sequences from each dataset.

In all of the experiments in this section, MEME was run using the most appropriate model for five passes. The informative prior (30-component Dirichlet mixture) was used with the protein datasets; the simple Dirichlet prior was used with the DNA datasets. MEME was *not* told the width of the motif but chose widths in the range $5 \leq W \leq 100$.

MEME finds the known motifs in the development datasets

Table V.3 shows that MEME finds the eleven known motifs on the first passes and predicts their widths quite well. The lowest value of ROC for any motif is 0.92, and many motifs have ROC of 1.00. The relative width (discovered motif width divided by known motif width) is close to 1.00 for many motifs. The

	dataset										
	OOPS					ZOOPS			TCM		
	crp	lex	hth	lip		hrp	crplex		farn		
	1	1	1	1	2	1	1	2	1	2	3
ROC	0.98	1.00	1.00	1.00	1.00	0.92	0.97	1.00	0.97	1.00	0.99
recall	0.71	1.00	1.00	1.00	1.00	0.39	0.71	1.00	0.73	0.88	0.77
precision	0.94	0.91	0.97	1.00	1.00	0.50	0.63	0.95	0.37	0.91	0.65
rel. width	0.80	0.80	1.00	0.44	0.38	1.59	1.10	1.00	0.92	0.58	0.67
rel. shift	0.15	0.10	0.00	0.50	0.25	-0.24	0.00	0.00	0.17	0.00	0.50
pass	1	1	1	1	2	2	2	1	1	2	3
palindrome	yes	yes					yes	yes			

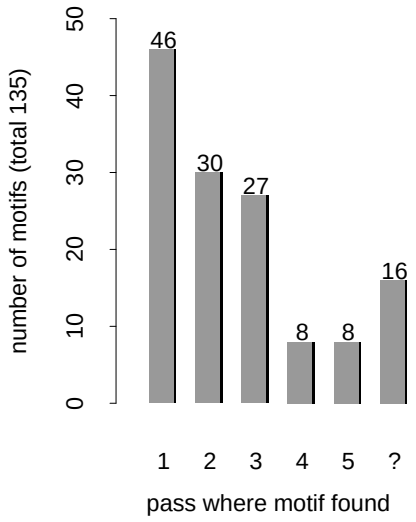
Table V.3: **Performance of** MEME **on the development datasets.** The table shows the ROC, relative width, relative shift, and pass when found of the best motif discovered by MEME for each known motif contained in the development datasets. The type of model used is shown above the dataset name. The columns beneath each dataset name show the results for each of the known motifs in that dataset. The simple Dirichlet prior was used with the DNA datasets and the 30-component Dirichlet mixture prior was used with the protein datasets. The palindrome constraints were tried with the DNA datasets and selected automatically if they improved the objective function $G(\phi)$.

discovered widths tend to be too small when the dataset is very small (i.e., datasets lip and farn which contain only five sequences each.) The relative shifts (shift of discovered motif divided by known motif width) are mostly close to zero. Most shifts are due simply to the discovered motif being shorter than the known motif. The three known DNA palindrome motifs are all detected by MEME and reported as such. No erroneous palindromic motifs are reported by MEME.

MEME finds the known motifs in the Prosite datasets

As can be seen in the histogram in Figure V.2, MEME using a ZOOPS model discovers 119 of the 135 motifs[4] contained in the datasets within five passes.

[4] A ZOOPS model is appropriate here because because 45 of the 75 datasets contain subfamilies. Including the all the subfamily motifs, there are 135 known motifs present in the 75 Prosite family datasets, many of which are only present in a small fraction of the sequences in a given dataset. The subfamily structure of the Prosite families is described in more detail in Table B.4.



model	*ROC*	*recall*	*precision*	*relative width*	*relative shift*
ZOOPS	0.99 (0.03)	0.82 (0.34)	0.77 (0.31)	1.22 (0.71)	-0.08 (0.46)

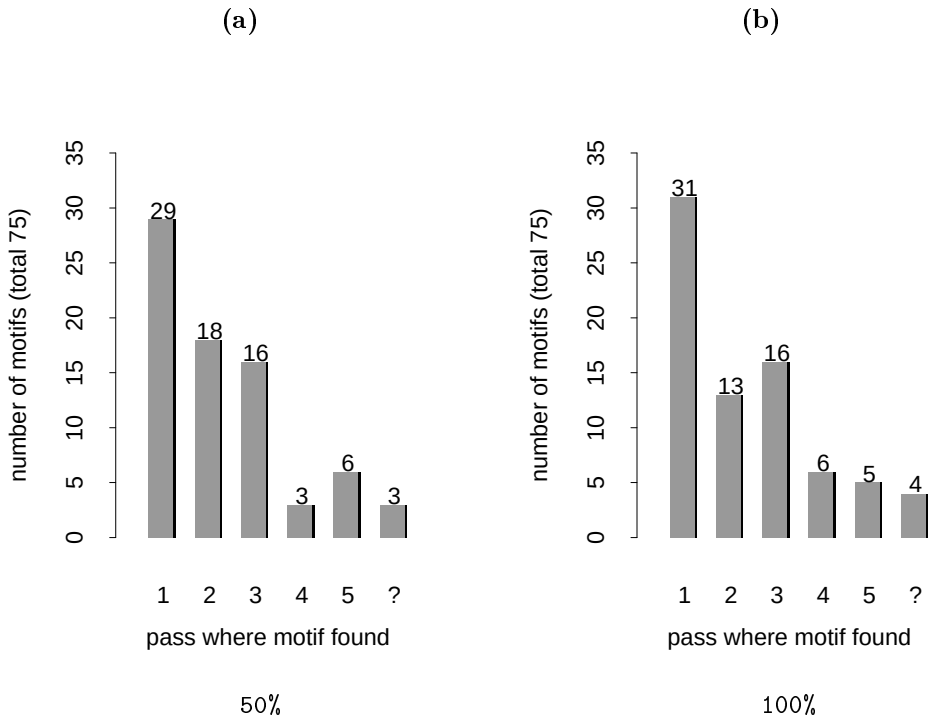
Figure V.2: **Histogram of the pass on which the known protein motif is found.** The number times of MEME finds a known Prosite motif on a given pass is shown in the histogram. A known motif if judged to have been found if the MEME motif has ROC over 0.99. The last column marked '?' represents known motifs which were not found by MEME within five passes. Average (standard deviation) performance of best motifs found by MEME in the 75 Prosite datasets versus all 135 known motifs present in the datasets is shown in the table beneath the histogram. The ZOOPS sequence model type was used along with the 30-component Dirichlet mixture prior.

The high average ROC of the MEME motifs can be seen in the table beneath the histogram in Figure V.2. The average recall and precision of these motifs using the MEME generated thresholds are both approximately 80%.[5]

The motifs found by MEME tend to be wider on average than the widths of the Prosite signatures. The signatures were specifically created to be as short as possible and still discriminate between members and non-members of a protein family. As can be seen from the average and standard deviation of the relative widths of the MEME motifs, most MEME motif widths are within a factor of two of the human-chosen signature widths. The MEME motifs are centered at or near the known motifs as indicted by the value of relative shift given in the table. The majority of MEME motifs are shifted no more than 50% of the width of the known motif.

Whether trained on half the members of a Prosite family or on all of them, MEME usually finds a motif which matches the known motif (i.e., with ROC over 0.99) on one of the first few passes. This is evident from the results of the first test shown in Figure V.3c and the histogram in Figure V.3(a). When trained on half the known sequences in a family using an OOPS sequence model, MEME fails to find the known motif only three times out of 75. Figure V.3(b) shows that when all the members of the family are present, the known motif in all but four training sets is discovered. A comparison of the two histograms reveals that the number of times that the known motif is found on the first pass is is virtually unaffected by removing half of the sequences from a Prosite family. The correlation between the number of sequences in the dataset and ROC is discussed Section V.B.4. The motif widths chosen by MEME with the smaller training sets do not correspond as closely to the widths of the Prosite signatures as seen by the higher standard deviation of relative width in line 1 as compared with line 2 of Table V.3c.

[5]The relative accuracies of MEME motifs and Prosite signatures at classifying sequence families will be discussed in Section V.B.2.



(c)

size	*ROC*		*recall*		*precision*		*relative width*		*relative shift*	
50%	1.00	(0.01)	0.86	(0.28)	0.82	(0.25)	1.24	(0.72)	-0.02	(0.41)
100%	1.00	(0.01)	0.85	(0.30)	0.79	(0.29)	1.23	(0.59)	-0.07	(0.40)

Figure V.3: **Effect of dataset size on finding known motifs in protein families.** The number of times MEME finds the known Prosite motif on a given pass is shown for Prosite training sets consisting of 50% (a) or 100% (b) of the sequences in one of the 75 Prosite families. A known motif if judged to have been found if the MEME motif has ROC over 0.99. The last column marked '?' represents known motifs which were not found by MEME within five passes. Average (standard deviation) performance of best motifs found by MEME in the 75 Prosite datasets versus the known motifs is shown in (c). MEME was run using the OOPS model type and the 30-component Dirichlet mixture prior.

V.B.2 Generalization accuracy

The cross-validation experiments described in this section show that MEME finds protein motifs which generalize well to the members of a family not present in the training set. The ROC of the first motif found by MEME using an OOPS model, averaged over the 75 Prosite datasets, 0.97, is only slightly worse than the *non-cross-validated* ROC (0.99) of the Prosite signatures. This is impressive considering that the signatures were created by human experts.

The tests described in this section consisted of performing stratified two-fold cross-validation on each Prosite dataset. This involved randomly splitting each dataset into two halves, training on one half and testing on the other half. The SWISS-PROT release 27 dataset (36000 sequences) was also split in half (after removing the known family member sequences) and a different half was combined with each of the Prosite testing sets during testing. MEME was run for one pass using an OOPS model, the Dirichlet mixture prior, and allowed to choose the motif width in the range $5 \leq W \leq 100$. Performance was measured against the known *sequences* belonging to the family buried in the large (18000 sequences) testing sets using a threshold of 18 bits.[6] The OOPS model was appropriate because, in each test, we want MEME to find a single motif that characterizes the entire Prosite family well. In essence I use MEME in this test as a supervised learning algorithm to discover a motif which describes a set of positive examples (the subset of family members making up the training set) and then validate the motif against the rest of the family members hidden among a large number of other sequences.

Table V.4 shows that MEME discovers motifs which accurately predict the members of the Prosite family which were not present in the training set. Average ROC is very high and the 18-bit threshold yields very respectable recall and

[6]The threshold of 18 bits was chosen based on there being 36000 sequences of average length 347 in SWISS-PROT release 27. There are 38 occurrences on average of each Prosite motif out of the approximately $347 \cdot 36000 \approx 10^7$ possible occurrences in SWISS-PROT. The average motif frequency is therefore $\lambda \approx 38/10^7$ and a reasonable threshold is $\log_2 \frac{1-\lambda}{\lambda} \approx 18$ bits.

algorithm	*ROC*		*recall*		*precision*	
MEME, OOPS model	0.97	(0.07)	0.74	(0.29)	0.73	(0.31)
human expert	0.99	(0.02)	0.92	(0.09)	0.72	(0.22)

Table V.4: **Performance comparison of** MEME **motifs and human generated motifs.** The two-fold cross-validated performance of motifs found by MEME in the 75 Prosite datasets at characterizing the protein families (as opposed to identifying the actual Prosite motifs) is shown. The average (standard deviation) performance values are for using the motif found by MEME to search SWISS-PROT release 27 with a threshold of 18 bits. MEME was run for one pass with the 30-component Dirichlet mixture prior and the OOPS model type. The *non-cross-validated* performance of the Prosite signatures is shown for comparison.

precision rates. The motifs MEME finds are are nearly as robust as those created by human experts. The performance figures for the Prosite signatures shown in Table V.4 are not cross-validated so they can be expected to over-estimate the performance of signatures on new sequences. Even so, these optimistic estimates of the the generalization accuracy of the Prosite signatures are not greatly different from the cross-validated estimates for the MEME motifs. It is reasonable to believe that, in practice, the MEME motifs will perform nearly as well as the Prosite signatures. This is impressive considering that the MEME motifs are generated automatically using only information contained in the sequences themselves. Later, in Section V.D, I discuss searching with multiple motifs discovered in a single training set to improve detection of distant family members.

V.B.3 Using background knowledge

The search for motifs can be aided by providing background knowledge of several types.

- Perhaps the most important type is knowledge of the appropriate type of sequence model to use. Knowing that each sequence contains exactly one occurrence of each motif greatly reduces the size of the space which needs to be searched with concomitant improvements in accuracy and speed compared

model type	*dataset*						
	OOPS-*like*				ZOOPS-*like*		TCM-*like*
	crp	lex	hth	lip	hrp	crplex	farn
OOPS	**0.98**	1.00	1.00	**1.00**	0.91	0.96	0.94
ZOOPS	**0.98**	**1.00**	**1.00**	**1.00**	**0.92**	**0.99**	0.91
TCM	0.92	0.99	0.99	0.98	0.88	0.97	**0.97**

Table V.5: **Performance of different sequence models on the development datasets.** Average ROC of the best motif discovered by MEME for all known motifs contained in each dataset is shown. The highest ROC for each dataset is printed in boldface type. The simple Dirichlet prior was used for both protein and DNA datasets. The palindrome constraint was not used. The datasets are divided according to the type of sequence model most appropriate to them based on background knowledge.

with a less constrained model type like TCM.

- Knowing the correct width of a motif in advance also restricts the search space and can improve the probability of finding a particular motif. It also improves the speed of the algorithm. When only one motif is to be used to characterize a family of sequences, specifying the width or at least constraining it to large values causes MEME to discover more selective motifs.

- Background knowledge can also be brought to bear via the prior on motif columns. The 30-component Dirichlet mixture prior biases the search for protein motifs towards motifs whose columns make sense biologically.

This is especially important with ZOOPS and TCM model types for preventing two or more separate motifs from being combined into one by overlaying one upon the other. Such "convex combination" motifs will tend to have columns that do not correspond to those found in the known biological motifs used in producing the Dirichlet mixture prior, so they will have lower posterior probability and thus be avoided by MEME.

Sequence model type

The advantage of using the appropriate model type when it is known is evident in Table V.5. This table shows the results of running MEME for five passes on each of the development datasets using each of the model types and then evaluating the MEME motifs against the known motifs. For datasets with multiple motifs, the ROC of the best MEME motif for each known motif is averaged together in the table. No other background knowledge was used in this test. MEME was left to find the appropriate motif width in the range $5 \leq W \leq 100$ and the simple Dirichlet prior was used. In the table, the datasets are grouped according to the type of sequence model that background knowledge dictates would best model them.

MEME finds the known motifs equally well using either an OOPS or ZOOPS model in the development datasets where each sequence contains one occurrence of each known motif. One can imagine cases where a ZOOPS model would be misled and discover motifs that only capture a subset of the occurrences of a known motif, but this does not happen with the OOPS-like development datasets. With the two development datasets whose sequences do not all contain occurrences of each motif, a ZOOPS model is clearly superior to either an OOPS or TCM model. Likewise, with the one development dataset where the motifs occur multiple times in each sequence the TCM model is clearly superior.

The OOPS and ZOOPS model types are about equally effective at discovering the known motifs in the 75 Prosite datasets. Table V.6 shows the results of running MEME for five passes using the simple Dirichlet prior and letting MEME choose the motif width in the range $5 \leq W \leq 100$. All of the known motifs, including ones present in only some of the sequences in a dataset, are included in the performance figures, so the background knowledge implies a ZOOPS model should perform better. A slight improvement in the recall and precision of the ZOOPS motifs (using the MEME-selected thresholds) can be noted in the table. A

model	*ROC*		*recall*		*precision*		*relative width*		*relative shift*	
OOPS	0.99	(0.02)	0.80	(0.36)	0.75	(0.33)	1.30	(0.75)	-0.10	(0.44)
ZOOPS	0.99	(0.02)	0.82	(0.34)	0.78	(0.31)	1.31	(0.77)	-0.08	(0.43)

	ROC		*recall*		*precision*	
	OOPS	ZOOPS	OOPS	ZOOPS	OOPS	ZOOPS
means	0.991	0.992	0.805	0.823	0.751	0.775
mean difference	-0.001		-0.018		-0.024	
t-test, df=134	-1.788		-1.385		-2.029	
significant? $t_{0.05,134} = 1.7$	YES		NO		YES	
significant? $t_{0.01,134} = 2.4$	NO		NO		NO	

Table V.6: **Performance comparison of different sequence models at finding known motifs in protein families.** The average (standard deviation) performance of the best motifs found by MEME using different sequence model types with the 75 Prosite datasets is shown in the upper table. The results of paired t-tests are shown in the lower table. Performance was measured versus the 135 known motifs contained in the datatsets. MEME was run for five passes and chose the motif widths in the range $5 \leq W \leq 100$ and used the simple Dirichlet prior.

paired t-test of the performance figures for the two model types shows that there is a (barely) statistically significant ($P = 0.05$) difference in the ROC and precision although the practical significance of this slight difference is dubious. I did not run MEME using the TCM model type on the entire set of 75 Prosite datasets because preliminary experiments showed that it tended to converge to motifs which were convex combinations of several motifs. The performance of TCM motifs would be much lower than that shown in Table V.6 for the OOPS and ZOOPS model types.

The first motif found using an OOPS model does slightly better than that found using a ZOOPS model at characterizing membership in the 75 Prosite families. I repeated the two-fold cross-validation experiment described in Table V.4 using a ZOOPS model and the results for both the OOPS and ZOOPS model types are shown in Table V.7. The difference in ROC is significant at the 5% level but not at the 1% level according to a paired t-test. The differences in recall and precision are not significant at the 5% level as shown in the table. Note that this

model	*ROC*		*recall*		*precision*	
OOPS	0.97	(0.07)	0.74	(0.29)	0.73	(0.31)
ZOOPS	0.96	(0.09)	0.73	(0.30)	0.70	(0.33)

	ROC		*recall*		*precision*	
	OOPS	ZOOPS	OOPS	ZOOPS	OOPS	ZOOPS
means	0.971	0.960	0.743	0.728	0.723	0.699
mean difference	0.011		0.015		0.025	
t-test, df=147	2.015		1.484		1.514	
significant? $t_{0.05,147} = 1.7$	YES		NO		NO	
significant? $t_{0.01,147} = 2.4$	NO		NO		NO	

Table V.7: **Comparison of the ability of different sequence models to characterize membership in protein families.** The average two-fold cross-validated performance of motifs found by MEME in the 75 Prosite datasets when MEME uses a motif width of 20 and when it is allowed to choose the motif width is shown in the upper table. The results of paired t-tests are shown in the lower table. The average (standard deviation) performance values are for using the motif found by MEME to search SWISS-PROT release 27 with a threshold of 18 bits. MEME was run with the 30-component Dirichlet mixture prior for one pass and chose the motif width in the range $5 \leq W \leq 100$.

	dataset										
	OOPS					ZOOPS			TCM		
motif	crp	lex	hth	lip		hrp	crplex		farn		
width	1	1	1	1	2	1	1	2	1	2	3
known width	0.97	1.00	1.00	1.00	1.00	0.92	0.96	1.00	0.96	0.99	0.98
$5 \leq W \leq 100$	0.98	1.00	1.00	1.00	1.00	0.92	0.98	1.00	0.97	0.96	0.98

Table V.8: The value of knowing the correct motif width with the development datasets. The ROC of the best motif discovered by MEME for each known motif contained in the development datasets when MEME is told the correct motif width or allowed to choose the width is shown. The type of model used is shown above the dataset name. The simple Dirichlet prior was used for both protein and DNA datasets and the palindrome constraint was not tried with DNA datasets.

experiment, in contrast to that described in Table V.6, only considered the ability of the motif found to detect unseen members of the family.

Considering the ability of the ZOOPS model type to discover motifs that do not occur in each sequence, the small decrease in performance observed indicates that the prior knowledge of whether the OOPS or ZOOPS sequence model is appropriate is of relatively little value. As a general rule, it is a good idea to use the ZOOPS model type instead of the OOPS model. The main drawback of this approach is that the execution time of MEME with a ZOOPS model is longer due to the need to try several values of the motif frequency λ.

Motif width

MEME discovers the known motifs in the development datasets equally well when it must find the motif widths on its own as when it is told the correct motif widths. The ROC of the best motif for each known motif found in five passes of MEME on each of the development datasets is shown in Table V.8. The first line shows the results when MEME is told the width of the known motifs and forced to search for motifs of exactly that width. The second line shows the results when MEME is forced to choose the motif widths. Optimizing the heuristic function $G(\phi)$

motif width	*ROC*		*recall*		*precision*	
fixed width, $W = 20$	0.99	(0.03)	0.82	(0.21)	0.84	(0.23)
MEME chooses, $5 \leq W \leq 100$	0.97	(0.07)	0.74	(0.29)	0.73	(0.31)

Table V.9: **The value of fixing the motif width when characterizing protein families.** The average two-fold cross-validated performance of motifs found by MEME in the 75 Prosite datasets when MEME uses a motif width of 20 and when it is allowed to choose the motif width is shown. The average (standard deviation) performance values are for using the motif found by MEME to search SWISS-PROT release 27 with a threshold of 18 bits. MEME was run for one pass with the 30-component Dirichlet mixture prior and the OOPS model type.

does an excellent job at picking motifs which correspond to the known motifs. The ROC results when MEME picks the width are essentially the same as when it is told the widths. The ROC value is higher when the known width is used for one motif, and lower for three motifs.

Forcing MEME to search for a single motif of width $W = 20$ in each of the 75 Prosite datasets yields motifs that characterize the individual families better than if MEME is allowed to choose the width. The explanation for this is that when the width is unconstrained MEME may choose a short motif on the first pass which is statistically surprising but not highly specific to the family. If the family can be characterized by multiple motifs, subsequent passes of MEME will detect these, some of which may be more specific to the family. This is the explanation for the fact that the performance figures in Table V.9 for motifs of width 20 are superior to those where MEME chose the width in the range $5 \leq W \leq 100$. This table summarizes the results of two-fold cross-validation of the single motif found by the first pass of MEME on the 75 Prosite datasets. As in Table V.7, the 30-component Dirichlet prior was used. The value of 20 for W was because it is slightly larger than the average fixed-width Prosite signature and at least as large as the majority of them as can be seen from the distribution of signature widths in Table V.2. Choosing a motif width that is slightly is less likely to cause problems than choosing one that is too short and consequently cannot capture the important

model type	*dataset*		
	OOPS-*like*		TCM-*like*
	hth	lip	farn
OOPS	0.9979	**1.0000**	0.9446
OOPS_DMIX	**1.0000**	**1.0000**	0.9336
ZOOPS	0.9992	**1.0000**	0.9112
ZOOPS_DMIX	**1.0000**	**1.0000**	0.9324
TCM	0.9901	0.9795	0.9693
TCM_DMIX	0.9827	0.9948	**0.9880**

Table V.10: **Comparison of simple Dirichlet and Dirichlet mixture priors using the development protein datasets.** The ROC of the best motif discovered by MEME (averaged over all known motifs contained in the dataset) when MEME was run for five passes on each of the development datasets containing protein sequences. Either the simple Dirichlet prior (OOPS, ZOOPS, TCM) or the 30-component Dirichlet mixture prior was used (OOPS_DMIX, ZOOPS_DMIX, TCM_DMIX). MEME chose the motif widths in the range $5 \leq W \leq 100$. The highest ROC for each dataset is printed in boldface type.

information in the motif occurrences.

Using a fixed motif width of 20, the motif MEME finds on the first pass has performance comparable to that of the Prosite signature for the family. The cross-validated average ROC is the same as the *non-cross-validated* average ROC of the Prosite signatures. This can be seen by comparing the performance figures for the Prosite signatures ("human expert" in Table V.4) and the figures for $W = 20$ in Table V.9. Using a threshold of 18 bits, the average precision of the MEME motifs is better but the recall is worse than Prosite signature. The threshold could be lowered to increase recall at the expense of precision, approximating the performance of the Prosite signatures at characterizing the protein families.

Informative column priors

Table V.10 shows that MEME finds the known motifs much better in the two small datasets lip and farn with a TCM sequence model if the mixture prior is

model	*ROC*	*recall*	*precision*	*relative width*	*shift*
OOPS	0.99 (0.02)	0.80 (0.36)	0.75 (0.33)	1.30 (0.75)	-0.98 (5.61)
OOPS_DMIX	0.99 (0.03)	0.81 (0.35)	0.76 (0.32)	1.21 (0.68)	-0.64 (5.34)
ZOOPS	0.99 (0.02)	0.82 (0.34)	0.78 (0.31)	1.31 (0.77)	-0.70 (5.57)
ZOOPS_DMIX	0.99 (0.03)	0.82 (0.34)	0.77 (0.31)	1.22 (0.71)	-0.59 (4.89)

Table V.11: **Comparison of simple Dirichlet and Dirichlet mixture priors for finding known motifs in protein families.** The average (standard deviation) performance of the best motifs found by MEME in the 75 Prosite datasets is shown. Performance was measured versus the 135 known motifs contained in the datasets. MEME chose the motif widths in the range $5 \leq W \leq 100$ and used the simple Dirichlet prior (OOPS, ZOOPS) or the 30-component Dirichlet mixture prior (OOPS_DMIX, ZOOPS_DMIX).

used instead of the simple Dirichlet prior. This is as one would expect—the effect of the prior is greatest for small sample sizes. The informative prior has little or no effect on the results with the larger hth dataset. The OOPS and ZOOPS sequence model types do not benefit substantially from the information in the 30-component mixture prior over columns in protein motifs.

The use of the 30-component Dirichlet mixture prior increases the agreement between the known motif widths and alignments and the motifs discovered by MEME, but virtually no improvement is visible in the ROC performance of MEME motifs discovered in the Prosite datasets when the mixture prior is used, as shown in Table V.11. There is, however, a significant improvement in ROC for the datasets containing fewer than 20 sequences. For the 36 Prosite datasets used which meet this criterion and would thus be most likely to benefit from the background information contained in the Dirichlet mixture prior, the ROC of motifs found with the mixture is significantly better at the 5% level for the OOPS model type according to a paired t-test. Table V.11 shows the average performance results on the Prosite datasets when MEME is run for five passes with various model types, with the simple Dirichlet prior or the 30-component Dirichlet mixture prior, and required to choose the motif width in the range $5 \leq W \leq 100$. The performance values are for all 135 known motifs contained in the 75 datasets.

V.B.4 Sensitivity

The ability of MEME to find subtle motifs is dependent on the amount of information in the motif relative to the size of the training set, and on the number of samples of the motif in the training set. Training sets on which MEME fails to find the Prosite motif generally tend to contain few sequences. This is not surprising since trying to discover a motif which has only a few examples is always difficult. This is related to the standard difficulties in estimating the parameters of a statistical model when the sample size is small. If the OOPS or ZOOPS models are being used, MEME can make use of at most n samples of the motif in a training set which contains n sequences. More directly, the statistical significance of the motif that MEME is expected to discover determines to a large extent whether or not it will be found. Thus, datasets with few sequences or even many sequences but very weak motifs may prevent MEME from discovering the known motifs.

The cases where MEME fails to find the known Prosite motif well tend to be when the training set contains few sequences (under 25) as shown in Figure V.12(a). This figure shows the relationships between the ROC of the best motif found by MEME and the number of sequences in the training set. The data is for the same experiment as shown in Figure V.3(b), where MEME was run for five passes using an OOPS model with each of the 75 Prosite datasets using the 30-component Dirichlet mixture prior. As seen in that figure, there were four training sets where MEME failed to find the known motif. Figure V.12(b) shows that two of those datasets contained known motifs whose statistical significance was extremely poor (the two data points with ROC near 0.99 and $\log_{10} G(\phi)$ near 0.0). The low statistical significance of the motifs in those datasets makes them difficult to detect. In only two instances did MEME fail to find known motifs with high significance. I measured the statistical significance using the same maximum likelihood ratio test used by MEME that compares the likelihood of the motif with the likelihood of the null model.

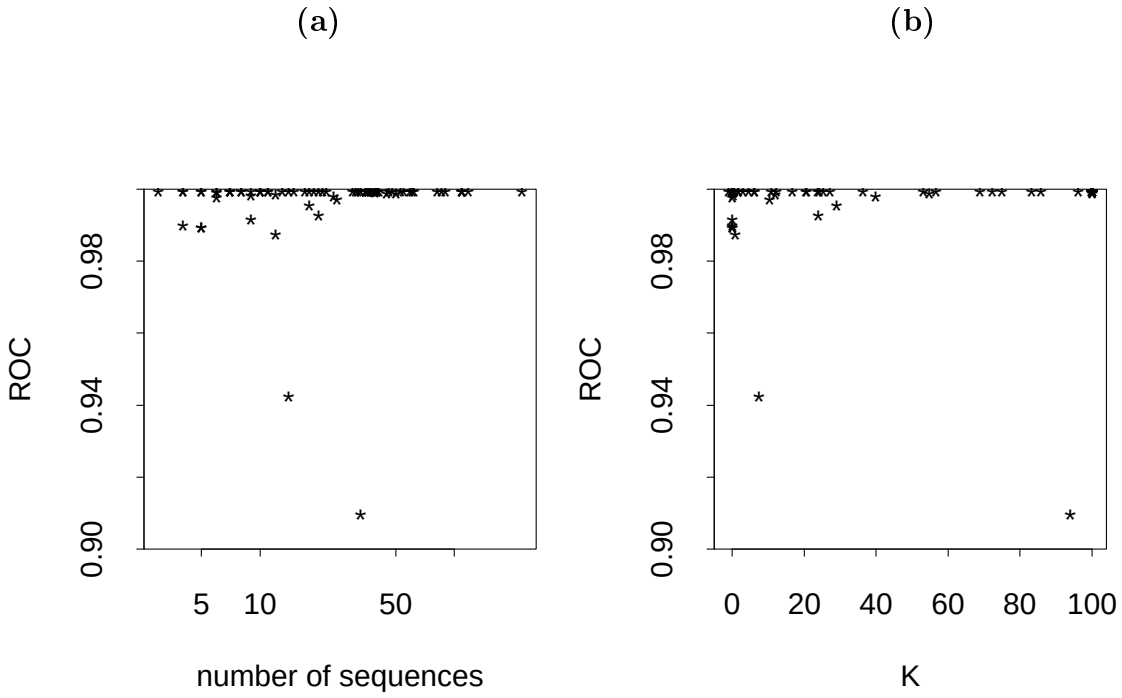


Table V.12: **Scatter plots of ROC versus (a) number of sequences in the training set, or (b) statistical significance $K = -\log_{10} G(\phi)$ of the known motif.**

A closer look at the amount of information contained in the *known* motifs in the Prosite database is given in Figure V.13. The histograms were created by specifying the actual positions of the known occurrences to MEME and letting MEME calculate the information content (IC) of the motif matrix and the significance level of the LRT ($G(\phi)$). The EM algorithm was *not* run, but the values of IC and $G(\phi)$ were computed as though EM had converged to a model predicting the known motif occurrences. Although the total information content of the known occurrences of the motifs in the 75 Prosite datasets shows a unimodal distribution centered around 20 bits in Figure V.13(a), the values of $G(\phi)$ form a bimodal distribution in Figure V.13(b). The two modes of the distribution correspond roughly to datasets for which the motif is statistically significant and those for which it is not. For 20 of 75 datasets, the value of $G(\phi)$ for the known motif occurrences is above 0.1, indicating that the dataset does not contain enough information to support the statistical hypothesis that the motif model is more likely than the null model at the 10% significance level. Any method of discovering motifs based a statistical likelihood approach (such as MEME) is bound to have difficulty with such datasets. Low values of $G(\phi)$ are caused in some cases by the motif having low information content relative to the size of the dataset, and, in other cases, by the dataset containing very few sequences.

Comparing Figures V.12 and V.13, it is apparent that MEME finds the known motifs even when they are not statistically significant according to the LRT. With most of the datasets whose known motif is not significant according to the LRT at the 0.01 significance level, MEME finds the known motif nonetheless. In particular, MEME finds the known motif in all but four datasets even though the known motif has statististical significance worse than 0.01 in 20 of the Prosite datasets. MEME is therefore useful for discovering motifs in datasets even when they cannot be said to be statistically significant. This also implies that the LRT should not be used to determine when to stop searching for motifs since useful

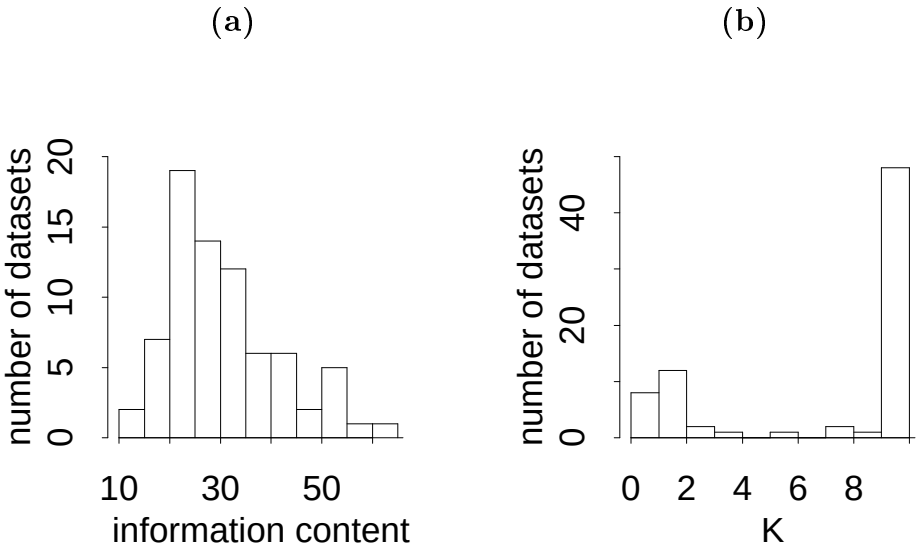


Table V.13: **Histograms of the number of Prosite datasets whose known motifs have (a) information content, or (b) $K = -\log_{10} G(\phi)$ in different ranges.**

or interesting motifs might be missed. Thus, although low information content or statistical significance can cause MEME to miss discovering a motif, it does not always prevent it from doing so.

V.B.5 Speed

The observed execution times for running MEME agree well with the theoretical time complexity calculations in Section III.H.2. The execution time is roughly quadratic in the size of the dataset due primarily to the time necessary for the TEST algorithm to test and choose a good starting point for EM. The OOPS sequence model type requires the least CPU time due to the fact that only one value of λ is tried. Running EM from starting points with a range of λ values increases the execution time by a constant factor when a ZOOPS sequence model is used. Use of a TCM sequence model requires even more time due to the larger range of λ values searched and the need to do a sort operation in the TEST algorithm for

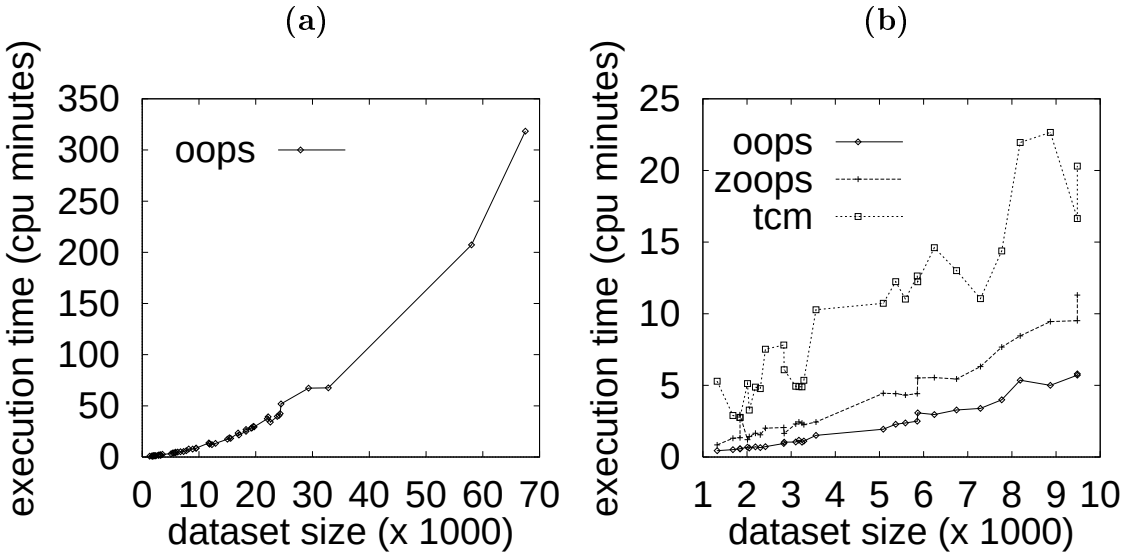


Figure V.4: **Speed of** MEME**.** The time required for MEME to discover one motif when run on protein datasets of varying sizes (thousands of characters) on a Sun SPARCstation 10. Figure (a) shows the execution time using an OOPS sequence model. Figure (b) shows the execution time using each of the three sequence models on small- to medium-sized datasets.

choosing EM starting points.

Figure V.4 shows the execution times for one pass of MEME run with different size protein datasets on a Sun SPARCstation 10. The execution time for datasets containing 1000 to approximately 70000 characters (number of sequences n times average sequence length L) is shown in Figure V.4(a) end can be seen to increase approximately as the square of the dataset size. Figure V.4(b) shows the CPU time required by MEME with each of the three sequence model types on datasets in the size range of approximately 500 to 10000 characters. The additional search required by the ZOOPS and TCM models can be seen to increase the execution time of the algorithm by approximately two and four times, respectively. The irregularity in the curves for each model type is due to the fact that execution time actually depends independently on the number of sequences n and their length L, not just on total dataset size nL, as was discussed in Section III.H.2.

	dataset										
	OOPS-*like*					ZOOPS-*like*			TCM-*like*		
	crp	lex	hth	lip		hrp	crplex		farn		
	1	1	1	1	2	1	1	2	1	2	3
Gibbs	**0.97**	**1.00**	**1.00**	**1.00**	**1.00**	**0.89**	**0.94**	**1.00**	**0.87**	**0.99**	**0.93**
OOPS	**0.98**	**1.00**	**1.00**	**1.00**	**1.00**	**0.91**	**0.91**	**1.00**	**0.87**	**0.99**	**0.94**
ZOOPS	**0.98**	**1.00**	**1.00**	**1.00**	**1.00**	**0.92**	**0.97**	**1.00**	**0.89**	**0.98**	**0.93**
TCM	**0.98**	**0.98**	**0.98**	**1.00**	**0.99**	**0.88**	**0.98**	**0.98**	**0.97**	**1.00**	**0.99**

Table V.14: **Performance comparison of** MEME **and the Gibbs sampler on the development datasets.** The ROC of the best motif discovered by the Gibbs sampler and MEME for each known motif contained in the development datasets. The type of dataset is shown above the dataset name. The columns beneath each dataset name show the results for each of the known motifs in that dataset. The Gibbs sampler was told the correct motif widths but MEME was not.

V.C Comparison with other methods

MEME compares favorably at finding multiple motifs with the Gibbs sampler algorithm [Lawrence *et al.*, 1993] and requires less prior knowledge. The published Gibbs sampler algorithm can find multiple motifs in DNA or protein sequence sets but requires that the number of occurrences of each motif be specified individually for each sequence in the dataset. Specifying that each sequence contains one occurrence of each motif is identical with assuming an OOPS sequence model. The first two lines in Table V.14 show that the ROC of the motifs found by the Gibbs sampler and by MEME with an OOPS model on the seven development datasets is essentially the same. This is not surprising since the Gibbs sampler is optimizing a likelihood function very similar to MEME's. The data in the table was generated by running the Gibbs sampler to simultaneously find five motifs with one occurrence per motif per sequence—essentially an OOPS model. Because the published program requires it, the Gibbs sampler was told the correct width for each motif. MEME was run *without* being told the correct width and left to discover it on its own. From the third line in the table, it can be seen that MEME with a ZOOPS model finds motifs with equal or better ROC than the Gibbs sam-

model	*ROC*		*recall*		*precision*	
Gibbs	0.98	(0.05)	0.78	(0.24)	0.88	(0.17)
MEME	0.99	(0.03)	0.82	(0.21)	0.84	(0.23)

Table V.15: **Comparison of** MEME **and the Gibbs sampler at characterizing protein families.** The average (standard deviation) two-fold cross-validated performance of best motifs found by MEME and the Gibbs sampler in the 75 Prosite datasets are shown. The motif width was fixed at $W = 20$ for both algorithms and only one motif was searched for.

pler in the development datasets (except for the second motif in farn) when the Gibbs sampler is not told in advance the correct number of motif occurrences in the datasets which do not fit the OOPS model. With the much less constrained TCM model, MEME still does an impressive job at finding the known motifs in the development datasets, as can be seen in line four of the table.

I compare the performance of MEME with that of the Gibbs sampler on the Prosite datasets in Table V.15. Both algorithms were given the same information—one motif, one occurrence per sequence and $W = 20$ for the width of the motif. The Gibbs sampler was run from 250 independent, random starts to maximize its chances of discovering the globally optimum motif. The results for the two algorithms are quite similar, with MEME giving motifs with slightly higher ROC. The recall and precision were measured by using the motifs discovered by MEME and the Gibbs sampler with a threshold of 18 bits to classify the sequences in SWISS-PROT release 30.

In Section V.D.4 I compare the performance of an algorithm which uses the Gibbs sampler as a component with that of MEME. That algorithm, like, MEME chooses the width of motifs and is therefore more directly similar to MEME than the Gibbs sampler alone.

V.D Case study: a dehydrogenase family

Together with biologist Dr. Michael E. Baker of the UCSD Department of Medicine we used MEME, PROBER and NOTE to study a family of sec-alcohol dehydrogenases. Dr. Baker constructed a dataset containing a group of distantly related dehydrogenase proteins and we used MEME to discover motifs in it. Studying these motifs sheds light on which parts of the sequence are most conserved across large evolutionary distances and why. The motifs corresponded to one known biologically important motif and several novel motifs. Using PROBER and NOTE we were able to identify distant and hard to detect members of this family in the SWISS-PROT database of proteins by combining the scores from all of the motifs discovered by MEME. Our results using MEME suggest that the known motif, a Prosite signature, may be incorrect since it misses these distant but definite family members. I end this section with a comparison of MEME with other methods of analyzing this family of proteins that shows that MEME is more biologically informative.

V.D.1 MEME-generated motifs

We constructed a dataset for analysis by MEME by using the FASTA program to search version 28 of SWISS-PROT database with *E. coli* sorbitol-6-phosphate dehydrogenase, Klebsiella ribitol dehydrogenase, and the Candida albicans trifunctional enzyme. These three divergent sequences identified 32 nonredundant sequences that have less than 30% identity with any other sequence in the set, after adjustment with gaps and insertions; many alignments have 20% or fewer identical residues ([Baker, 1994], [Persson *et al.*, 1991]) in pairwise alignments. The SWISS-PROT identifiers and title lines for the sequences in the dataset are shown in Table V.16.

Figure V.5 shows the information content (IC) and the three-level consen-

identifier	*description*
2BHD_STREX	20-BETA-HYDROXYSTEROID DEHYDROGENASE (EC 1.1.1.53)
3BHD_COMTE	3-BETA-HYDROXYSTEROID DEHYDROGENASE (EC 1.1.1.51)
ACT3_STRCO	PUTATIVE KETOACYL REDUCTASE (EC 1.3.1.-)
ADH2_DROMO	ALCOHOL DEHYDROGENASE 2 (EC 1.1.1.1)
ADH2_DROWH	ALCOHOL DEHYDROGENASE 2 (EC 1.1.1.1)
ADH_DROSI	ALCOHOL DEHYDROGENASE (EC 1.1.1.1)
AP27_MOUSE	ADIPOCYTE P27 PROTEIN (AP27)
BA71_EUBSP	7-ALPHA-HYDROXYSTEROID DEHYDROGENASE (EC 1.1.1.159)
BA72_EUBSP	7-ALPHA-HYDROXYSTEROID DEHYDROGENASE (EC 1.1.1.159)
BDH_HUMAN	D-BETA-HYDROXYBUTYRATE DEHYDROGENASE PRECURSOR (EC 1.1.1.30) (BDH) (3-HYDROXYBUTYRATE DEHYDROGENASE) (FRAGMENT)
BDH_RAT	D-BETA-HYDROXYBUTYRATE DEHYDROGENASE PRECURSOR (EC 1.1.1.30) (BDH) (3-HYDROXYBUTYRATE DEHYDROGENASE)
BEND_ACICA	CIS-1,2-DIHYDROXY-3,4-CYCLOHEXADIENE-1-CARBOXYLATE DEHYDROGENASE (EC 1.3.1.-)
BNZE_PSEPU	CIS-1,2-DIHYDROBENZENE-1,2-DIOL DEHYDROGENASE (EC 1.3.1.19) (CIS- BENZENE GLYCOL DEHYDROGENASE)
BPHB_PSEPS	BIPHENYL-CIS-DIOL DEHYDROGENASE (EC 1.3.1.-)
BUDC_KLETE	ACETOIN(DIACETYL) REDUCTASE (EC 1.1.1.5) (ACETOIN DEHYDROGENASE)
DHES_HUMAN	ESTRADIOL 17 BETA-DEHYDROGENASE (EC 1.1.1.62) (20 ALPHA-HYDROXYSTEROID DEHYDROGENASE) (E2DH) (17-BETA-HSD)
DHGA_BACME	GLUCOSE 1-DEHYDROGENASE A (EC 1.1.1.47)
DHGB_BACME	GLUCOSE 1-DEHYDROGENASE B (EC 1.1.1.47)
DHG_BACSU	GLUCOSE 1-DEHYDROGENASE (EC 1.1.1.47)
DHII_HUMAN	CORTICOSTEROID 11-BETA-DEHYDROGENASE (EC 1.1.1.146) (11-DH) (11-BETA- HYDROXYSTEROID DEHYDROGENASE) (11-BETA-HSD)
DHII_RAT	CORTICOSTEROID 11-BETA-DEHYDROGENASE (EC 1.1.1.146) (11-DH) (11-BETA- HYDROXYSTEROID DEHYDROGENASE) (11-BETA-HSD)
DHK1_STRVN	GRANATICIN POLYKETIDE SYNTHASE PUTATIVE KETOACYL REDUCTASE 1 (EC 1.3.1.-) (ORF5)
DHK2_STRVN	GRANATICIN POLYKETIDE SYNTHASE PUTATIVE KETOACYL REDUCTASE 2 (ORF6)
DHMA_FLAS1	N-ACYLMANNOSAMINE 1-DEHYDROGENASE (EC 1.1.1.233) (NAM-DH)
ENTA_ECOLI	2,3-DIHYDRO-2,3-DIHYDROXYBENZOATE DEHYDROGENASE (EC 1.3.1.28)
FABG_ARATH	3-OXOACYL-[ACYL-CARRIER PROTEIN] REDUCTASE PRECURSOR (EC 1.1.1.100) (3-KETOACYL-ACYL CARRIER PROTEIN REDUCTASE)
FABG_CUPLA	3-OXOACYL-[ACYL-CARRIER PROTEIN] REDUCTASE PRECURSOR (EC 1.1.1.100) (3-KETOACYL-ACYL CARRIER PROTEIN REDUCTASE)
FABG_ECOLI	3-OXOACYL-[ACYL-CARRIER PROTEIN] REDUCTASE (EC 1.1.1.100) (3-KETOACYL- ACYL CARRIER PROTEIN REDUCTASE)
FIXR_BRAJA	FIXR PROTEIN
FOX2_YEAST	HYDRATASE-DEHYDROGENASE-EPIMERASE (HDE)
GUTD_ECOLI	SORBITOL-6-PHOSPHATE 2-DEHYDROGENASE (EC 1.1.1.140) (GLUCITOL-6- PHOSPHATE DEHYDROGENASE)
HDE_CANTR	HYDRATASE-DEHYDROGENASE-EPIMERASE (HDE)
HDHA_ECOLI	7-ALPHA-HYDROXYSTEROID DEHYDROGENASE (EC 1.1.1.159) (HSDH)
LIGD_PSEPA	C ALPHA-DEHYDROGENASE (EC -.-.-.-)
NODG_AZOBR	NODULATION PROTEIN G
NODG_RHIME	NODULATION PROTEIN G (HOST-SPECIFICITY OF NODULATION PROTEIN C)
NODG_RHIMS	NODULATION PROTEIN G (HOST-SPECIFICITY OF NODULATION PROTEIN C)
PGDH_HUMAN	15-HYDROXYPROSTAGLANDIN DEHYDROGENASE (NAD(+)) (EC 1.1.1.141)
PHBB_ALCEU	ACETOACETYL-COA REDUCTASE (EC 1.1.1.36)
PHBB_ZOORA	ACETOACETYL-COA REDUCTASE (EC 1.1.1.36)
RIDH_KLEAE	RIBITOL 2-DEHYDROGENASE (EC 1.1.1.56) (RDH)
SX19_YEAST	SPORULATION PROTEIN SPX19
TODD_PSEPU	CIS-TOLUENE DIHYDRODIOL DEHYDROGENASE (EC 1.3.1.19)
XYLL_PSEPU	CIS-1,2-DIHYDROXY-3,4-CYCLOHEXADIENE-1-CARBOXYLATE DEHYDROGENASE (EC 1.3.1.-)
YINL_LISMO	HYPOTHETICAL 26.8 KD PROTEIN IN INLA 5'REGION (ORFA)
YOHF_ECOLI	HYPOTHETICAL 20.2 KD PROTEIN IN DLD-CDD INTERGENIC REGION
YRTP_BACSU	HYPOTHETICAL 25.3 KD PROTEIN IN RTP 5'REGION (ORF238)

Table V.16: SWISS-PROT **identifiers and title lines for the 32 dehydrogenases in the dehydrogenase dataset.**

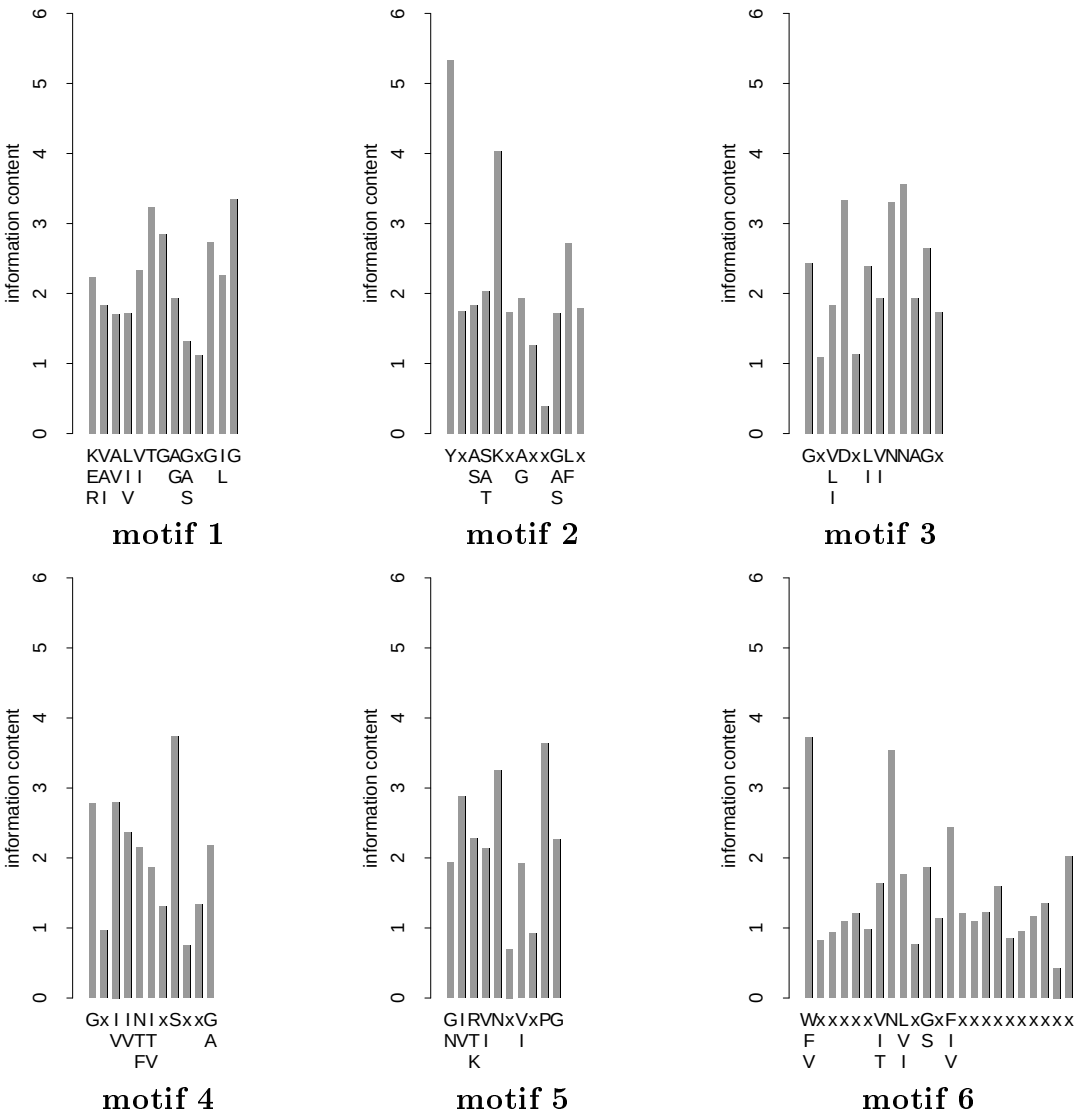


Figure V.5: **Consensus sequences and information content of the six** MEME **dehydrogenase motifs.**

sus of each column of the first six motifs identified by MEME in the dehydrogenase dataset. The three-level consensus shows the residues at sites where one to three amino acids are found in 80% of the dataset's sequences with the highest scoring amino acid first. Some sites with `x`'s have a preference for one or more residues in more than 50% of the sequences, and this could be functionally important for subsets of the dehydrogenase family.

V.D.2 Motif analysis

MEME motifs correspond to structural units and exon-intron boundaries. The six motifs show some alignment with β-strand and α-helix structures for the two enzymes that have had their 3D structure determined. They also correspond nicely to exon-intron boundaries for three animal proteins in which the genomic structure has been determined. Figure V.6 shows the motifs aligned onto the sequences of Streptomyces hydrogenans 20b-hydroxysteroid dehydrogenase and rat dihydropteridine reductase and their secondary structures as determined by X-ray crystallographic analysis. Schematic representations of the three dimensional structures of the proteins are also shown in the figure. The motifs found by MEME are shown aligned with the primary and secondary structure of 20 β-hydroxysteroid dehydrogenase, which was in the training set, and with the distantly related rat dihydropteridine reductase. MEME motif hits are indicated by motif number above the primary sequences. Alpha helices are indicated by "ah", beta sheets by "bs" and turns by "t" under the primary sequences. For the dehydrogenase, all MEME motif hits shown had scores above 18 bits. For the reductase, the hits shown have scores between 2.5 and 13.6 bits

Motif 1 is 13 residues long and corresponds to β-strand A and the turn that are part of a $\beta\alpha\beta$ fold that is the AMP binding domain at the amino terminus. The Gly-X-X-X-Gly-X-Gly part is considered to be a signature for NAD(P)(H) binding proteins and is used to locate the nucleotide binding domain on dehydro-

20β-hydroxysteroid dehydrogenase

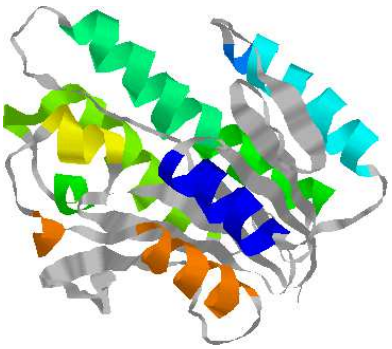
```
  1       1111111111111
  1 MNDLSGKTVIITGGARGLGAEAARQAVAAGARVVLADVLDEEGAATARELGDAARYQHL
        |bs-A__|t |ah-B________|bs-B___|ah-C_______|     |bs-C_

 60                    333333333333             666666666666666
 60 DVTIEEDWQRVVAYAREEFGSVDGLVNNAGISTGMFLETESVERFRKVVDINLTGVFIG
    ___| |ah-D________| |bs-D________|      |ah-E_________________

119 66666666     44444444444          222222222222          5555
119 MKTVIPAMKDAGGGSIVNISSAAGLMGLALTSSYGASKWGVRGLSKLAAVELGTDRIRV
    -------|    |bs-E___|           |ah-F--------------|       |--

178 555555
178 NSVHPGMTYTPMTAETGIRQGEGNYPNTPMGRVGNEPGEIAGAVVKLLSDTSSYVTGAE
    bs-F--|                              |ah-G------|        |---

237
237 LAVDGGWTTGPTVKYVMGQ
    bs-G|
```



rat dihydropteridine reductase

```
  1        1111111111111
  1 MAASGEARRVLVYGGRGALGSRCVQAFRARNWWVASIDVVENEEASASVIVKMTDSFTE
          |bs__|     |ah-B______|   |bs___|        |bs__|      |ah

 60                333333333333
 60 QADQVTAEVGKLLGDQKVDAILCVAGGWAGGNAKSKSLFKNCDLMWKQSIWTSTISSHL
    -D_________|       |bs_|          |ah-E_______________________

119                              222222222222
119 ATKHLKEGGLLTLAGAKAALDGTPGMIGYGMAKGAVHQLCQSLAGKNSGMPSGAAAIAV
    ___|     |bs__|   |ah|       |ah-F____________|          |bs___

178                                            44444444444
178 LPVTLDTPMNRKSMPEADFSSWTPLEFLVETFHDWITGNKRPNSGSLIQVVTTDGKTEL
    -------|ah__|_____|a|__||ah-G______|          |bs____|    |bs

237
237 TPAYF
    __|
```

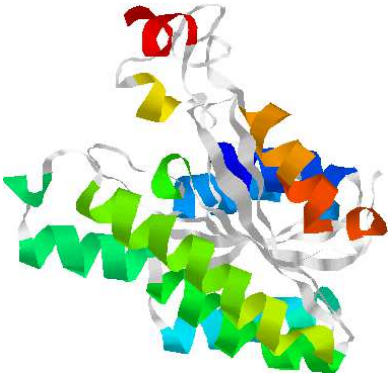


Figure V.6: **Three-dimensional structures and primary sequences annotated with secondary structure and locations of** MEME **motifs, for two dehydrogenases.**

genases. The three glycines receive high entropy scores. One glycine is found in all sequences; the other two glycines are found in 90% of the dataset. As seen in Figure V.5, the MEME motif contains other residues with high entropy scores. The MEME motif is Val-X-X-Val-Thr-Gly-X-X-X-Gly-Ile-Gly, which is a more informative description of this domain in sec-alcohol dehydrogenases. The information content of this motif is 28.1 bits.

A search of SWISS-PROT version 30 with motif 1 yields 58 sequences with scores that are 6 standard deviations ($P = 10^{-9}$) above the mean. Figure V.7 shows the bimodal output, indicating good selectivity for sec-alcohol dehydrogenases. Several new sequences were found. Also seen were mammalian fatty acid synthesis genes, which have much in common in function with bacterial antibiotic dehydrogenases. The figure shows the output of the PROBER utility for the first motif found by MEME. The hits for the first motif are shown along with *part* of the score histogram. Each star represents the score for one subsequence. The score at the center of each histogram is shown in the first column on the left, the number of subsequences in the bin is shown in the second column.

Motif 2 is 12 residues and corresponds to α-helix F. This motif is of interest because it contains a tyrosine and lysine that are highly conserved and which MEME scores with very high information content, in agreement with earlier sequence analyses and the results of site specific mutagenesis in Drosophila alcohol dehydrogenase, human 11β-hydroxysteroid dehydrogenase-type 1, human 17β-hydroxysteroid dehdyrogenase-type 2 and human 15-hydroxyprostaglandin dehydrogenase, all of which indicate that the tyrosine and lysine have a central role in catalysis. This motif is similar to the Prosite signature for the short chain alcohol dehydrogenase family. MEME identifies a more complex motif consisting of a consensus sequence of Y-X-A-S-K-X-A-X-X-G-L-X. The information content of this motif is 26.2 bits. A search of SWISS-PROT with the log-odds matrix for motif 2 yielded 90 scores that are 6 standard deviations above the mean. The output is

```
Motif  1: sig = 5.213000e-01

There were    69 scores above the threshold of 18
Alignment of sites with IC scores over 18:
sequence         start     IC        pre site          post
--------         -----     --        --- ----          ----
2BHD_STREX           7  30.14     MNDLSG KTVIITGGARGLG AEAARQAVAA
3BHD_COMTE           7  33.02     TNRLQG KVALVTGGASGVG LEVVKLLLGE
ACT3_STRCO           7  34.11     MATQDS EVALVTGATSGIG LEIARRLGKE
ADH1_EMENI         172  18.08 GLKEAGVRPG QTVAIVGAGGGLG SLAQQYAKAM
ADH3_EMENI         174  18.08 GLKESGARPG QTVAIVGAGGGLG SLAQQYAKAM
AP27_MOUSE           8  26.86    MKLNFSG LRALVTGAGKGIG RDTVKALHAS
BA71_EUBSP           7  29.52     MKLVQD KITIITGGTRGIG FAAAKLFIEN
BA72_EUBSP           7  32.64     MNLVQD KVTIITGGTRGIG FAAAKIFIDN
BDH_BOVIN           10  22.38  AASVDPVGS KAVLITGCDCGFG FCLAKHLHSE
BDH_HUMAN           56  27.73 YASAAEPVGS KAVLVTGCDSGFG FSLAKHLHSK
BDH_RAT             57  27.73 YTSQADAASG KAVLVTGCDSGFG FSLAKHLHSK
BEND_ACICA          10  36.70  MNSTQRFEH KVVIVTGAAQGIG RGVALRIAQE
... 50 sequences omitted ...
XYLL_PSEPU           8  35.33    MNKRFQG KVAVITGAAQGIG RRVAERMAAE
YB9K_YEAST           8  32.93    MKFTLED QVVLITGGSQGLG KEFAKKYYNE
YCIK_ECOLI          13  24.64 YQPKQDLLND RIILVTGASDGIG REAAMTYARY
YINL_LISMO           6  34.91      MTIKN KVIIITGASSGIG KATALLLAEK
YKF5_YEAST           5  32.08       MHYL PVAIVTGATRGIG KAICQKLFQK
YOHF_ECOLI           3  23.91         MA QVAIITASDSGIG KECALLLAQQ
YRTP_BACSU           7  33.39     MQSLQH KTALITGGGRGIG RATALALAKE

Histogram of information content scores on dataset swissprot30:
================================================================================
                   0         10        20        30        40        50
  center (  count  )+         +         +         +         +         +
================================================================================
      38 (        2)|**
----------------------------------------  8 sigma -----------------------------
      36 (        2)|**
      34 (       10)|**********
      32 (       14)|**************
      30 (        7)|*******
      28 (        6)|******
----------------------------------------  7 sigma -----------------------------
      26 (        5)|*****
      24 (       10)|**********
      22 (        5)|*****
      20 (        5)|*****
      18 (        6)|******
----------------------------------------  6 sigma -----------------------------
      16 (       12)|************
      14 (       23)|***********************
      12 (       49)|************************************************
      10 (       44)|********************************************
       8 (       51)|*************************************************~
----------------------------------------  5 sigma -----------------------------
       6 (      115)|*************************************************~
       4 (      223)|*************************************************~
       2 (      356)|*************************************************~
       0 (      739)|*************************************************~
      -2 (     1150)|*************************************************~
```

Figure V.7: **Distribution of scores of the first dehydrogenase motif on** SWISS-PROT **release 30.**

very bimodal, indicating good selectivity for this dataset.

Motif 3 is 12 residues long, matching well with β-strand D. Although the input was 32 sequences, because two sequences contain gene duplications the input was 34 dehydrogenase sequences. MEME identified the two gene duplications, both of which have Motif 3. MEME identifies 7 residues that are present in at least 70% of the sequences yielding a consensus motif of Gly-X-Val-Asp-X-Leu-Val-Asn-Asn-Ala-Gly-X for this dehydrogenase family. The information content of this motif is 26.9 bits. A search of the SWISS-PROT database with motif 3 yielded 90 sequences with scores that are 6 standard deviations above the mean. Thirty nine sequences are Drosophila alcohol dehydrogenases. Many others are close homologs of members of the dataset; that is enzymes in rat that are almost identical to those in humans. Nevertheless, the search identified several distantly related proteins, in particular Leishmania resistance protein, plant protochlorophyllide reductases and Myxobacteria CsgA.

Motif 4 is 11 residues long and corresponds to β-strand E. This motif contains a highly conserved serine residue. The information content of this motif is 21.8 bits. A search of the SWISS-PROT database yielded 60 scores that are 6 standard deviations above the mean, with a nice bimodal output. Motif 5 is 10 residues long and corresponds to β-strand F. The information content of this motif is 21.6 bits. A search of the SWISS-PROT database yielded 50 scores that are 6 standard deviations above the mean, again with a nice bimodal output. Motif 6 is 23 residues long and corresponds to α-helix E. The information content of this motif is 21.6 bits. A search of the SWISS-PROT database yielded 83 scores that are 6 standard deviations above the mean, with a nice bimodal output.

V.D.3 Dihydropteridine reductase and other distant homologs

Rat and human dihydropteridine reductase are sec-alcohol dehydrogenases although a search with BLAST does not reveal any high scoring homologous sequences. The crystal structure of the human and rat enzymes have been solved and found to closely resemble S. hydrogenans 20β-hydroxysteroid dehydrogenase.

We searched rat dihydropteridine reductase against the 6 motifs. None of the motifs scored above the 6 standard deviation cutoff score of 18 bits. However, motif 2 which contains the catalytic residues scores close to the cutoff (13.6 bits) and motifs 1 and 3 have positive scores. Moreover the order and spacing of the first three motifs was characteristic of the enzyme family as can be seen in Figure V.6.

For most of the proteins in the dehydrogenase dataset, the distances between the motifs are reasonably well conserved, a point that is noted in the multiple sequence alignments where the the first 190 residues have few gaps or insertions in the majority of sec-alcohol dehydrogenases. The order and spacing of the MEME motifs in the dehydrogenase dataset is shown in Table V.17 along with the total score for the sequence. The total score is the sum of the single best score for each motif.

We used this information about the order and spacing of motifs to create another criterion for sec-alcohol dehydrogenases. We could examine the output from a database search for proteins that contain more than one motif in the characteristic order and spacing. In this way, proteins with motifs that were below 6 standard deviations in significance, could be identified as homologs, if they had the proper motif order. We then added the scores for the six motifs to get a score for the protein that could be used to evaluate its relationship to other proteins. We could also extract homologs that may not conserve all motifs during evolution. Using this criterion, the dihydropteridine reductases (SWISS-PROT entries DHPR_RAT and DHPR_HUMAN) have a total score of 19 and 22 bits, respectively. These

sequence	*total score*	*block diagram*
2BHD_STREX	197	(6)-1-(59)-3-(13)-6-(5)-4-(9)-2-(10)-5-(72)
3BHD_COMTE	175	(6)-1-(59)-3-(13)-6-(4)-4-(9)-2-(12)-5-(69)
ACT3_STRCO	205	(6)-1-(62)-3-(13)-6-(7)-4-(9)-2-(10)-5-(73)
ADH_DROME	132	(83)-3-(5)-6-(8)-4-(9)-2-(92)
AP27_MOUSE	163	(7)-1-(54)-3-(13)-6-(6)-4-(9)-2-(10)-5-(64)
BA72_EUBSP	185	(6)-1-(64)-3-(13)-6-(5)-4-(9)-2-(10)-5-(61)
BDH_HUMAN	158	(55)-1-(67)-3-(13)-6-(4)-4-(9)-2-(10)-5-(104)
BEND_ACICA	182	(9)-1-(61)-3-(14)-6-(5)-4-(7)-2-(10)-5-(74)
BNZE_PSEPU	178	(5)-1-(59)-3-(18)-6-(4)-4-(9)-2-(9)-5-(90)
BPHB_PSEPS	156	(5)-1-(58)-3-(18)-6-(3)-4-(9)-2-(9)-5-(92)
BUDC_KLETE	187	(2)-1-(62)-3-(13)-6-(6)-4-(9)-2-(10)-5-(58)
DHES_HUMAN	161	(2)-1-(66)-3-(13)-6-(25)-2-(161)
DHGB_BACME	172	(7)-1-(64)-3-(13)-6-(6)-4-(10)-2-(10)-5-(71)
DHII_HUMAN	126	(34)-1-(88)-6-(4)-4-(9)-2-(98)
DHK1_STRVN	203	(17)-1-(62)-3-(13)-6-(7)-4-(9)-2-(10)-5-(73)
DHMA_FLAS1	151	(14)-1-(83)-6-(31)-2-(10)-3-(63)
ENTA_ECOLI	155	(5)-1-(52)-3-(13)-6-(25)-2-(10)-5-(73)
FABG_ECOLI	199	(5)-1-(59)-3-(13)-6-(5)-4-(9)-2-(10)-5-(62)
FIXR_BRAJA	167	(36)-1-(60)-3-(19)-6-(4)-4-(10)-2-(10)-5-(58)
FOX2_YEAST	191	(9)-1-(69)-3-(13)-6-(5)-4-(9)-2-(146)-1-(60)-3-(13)-6-(5)-4-(9)-2-(10)-5-(400)
GUTD_ECOLI	156	(2)-1-(64)-3-(13)-6-(6)-4-(9)-2-(94)
HDE_CANTR	197	(8)-1-(68)-3-(13)-6-(5)-4-(169)-1-(58)-3-(13)-6-(5)-4-(9)-2-(10)-5-(408)
HDHA_ECOLI	173	(11)-1-(62)-3-(12)-6-(5)-4-(9)-2-(10)-5-(65)
LIGD_PSEPA	119	(6)-1-(88)-6-(6)-4-(9)-2-(137)
NODG_RHIME	192	(6)-1-(59)-3-(13)-6-(5)-4-(9)-2-(10)-5-(62)
PGDH_HUMAN	204	(5)-1-(64)-3-(5)-6-(8)-4-(9)-2-(12)-5-(82)
PHBB_ZOORA	205	(2)-1-(58)-3-(13)-6-(5)-4-(9)-2-(10)-5-(63)
RIDH_KLEAE	161	(14)-1-(59)-3-(13)-6-(5)-4-(9)-2-(10)-5-(58)
TODD_PSEPU	178	(5)-1-(59)-3-(18)-6-(4)-4-(9)-2-(9)-5-(90)
XYLL_PSEPU	173	(7)-1-(60)-3-(14)-6-(5)-4-(7)-2-(10)-5-(85)
YINL_LISMO	160	(5)-1-(62)-3-(13)-6-(5)-4-(9)-2-(83)
YRTP_BACSU	198	(6)-1-(62)-3-(13)-6-(5)-4-(9)-2-(72)

Table V.17: MEME **motif block diagrams of the dehydrogenase dataset sequences.** Motifs are shown by number (i.e., -1-) and spacings are shown in parentheses (i.e., -(59)-). The total score for the sequence is the sum of the single best match scores for each motif.

scores rank them in the top 150 sequences (out of 40,292 in SWISS-PROT release 30) in terms of similarity to the dehydrogenase dataset.

V.D.4 Comparison with other methods

We compared the results of motif discovery using MEME with two other methods of pattern discovery in groups of related sequences. The PROTOMAT system [Henikoff and Henikoff, 1991] finds patterns in the form of "blocks". They define a block as an "ungapped region of aligned amino acids". In essence, a PROTOMAT block is a set of similar subsequences of a given width taken from a multiple alignment. Only the most similar subsequences in a given region of the multiple alignment are members of the block. This differs from the definition of a motif used by MEME. A MEME motif is a probability matrix which combines contributions from all of the examples of the motif occurring in subsequences in a group of related protein or DNA sequences. The information present in the distant examples of the motif is preserved in the MEME motif representation, whereas it is lost in the PROTOMAT block representation. PROTOMAT outputs the best set of blocks which occur in the same order in each of a critical number of sequences. This set of blocks is called the "best path". Only the sequences contained in the best path are saved in the blocks output by PROTOMAT. The best path idea is similar to the idea of using several MEME motifs in a given order as a means of detecting distant homologs. PROTOMAT thus solves a problem similar to using MEME to search for multiple motifs with a ZOOPS model except for the constraint that all motifs be present in the *same* subset of the sequences. So it is not designed, as MEME is, to be able to detect subfamilies in a group of sequences. Similarly, PROTOMAT is not designed to detect motifs which have multiple repeats in each of the sequences as MEME can do using a TCM model.

I used the PROTOMAT system program BLOCKMAKER to find the blocks in the dehydrogenase dataset. The program produces two alternative sets

```
BDH_HUM ( 56) KAVLVTGCDSGFG ( 26) GVKELDSLNSD ( 82) GRVVNISSMLGRMANPARSPYCITKFGVEAFSDCLRYEMYPLGVKVS
2BHD_ST (  7) KTVIITGGARGLG ( 59) GSVDGLVNNAG ( 42) GSIVNISSAAGLMGLALTSSYGASKWGVRGLSKLAAVELGTDRIRVN
3BHD_CO (  7) KVALVTGGASGVG ( 59) GTLNVLVNNAG ( 41) GSIINMASVSSWLPIEQYAGYSASKAAVSALTRAAALSCRKQGYAIR
ACT3_ST (  7) EVALVTGATSGIG ( 62) GPVDVLVNNAG ( 44) GRIVNIASTGGKQGVVHAAPYSASKHGVVGFTKALGLELARTGITVN
ADH_DRO (  7) KNVIFVAGLGGIG ( 64) KTVDVLINGAG ( 37) GIICNIGSVTGFNAIYQVPVYSGTKAAVVNFTSSLAKLAPITGVTAY
AP27_MO (  8) LRALVTGAGKGIG ( 54) GPVDLLVNNAA ( 43) GSIVNVSSMVAHVTFPNLITYSSTKGAMTMLTKAMAMELGPHKIRVN
BA72_EU (  7) KVTIITGGTRGIG ( 64) GRLDVMINNAG ( 42) GVIINTASVTGIFGSLSGVGYPASKASVIGLTHGLGREIIRKNIRVV
BEND_AC ( 10) KVVIVTGAAQGIG ( 61) GRIDVLINNVG ( 41) QQGTIVNVSSIATRGIHRIPYSACKGGVNALTASLAFEHAQHGIRVN
BNZE_PS (  6) EVALVTGGGAGLG ( 59) GKLDCLVGNAG ( 46) GSAIFTVSNAGFYPGGGGVLYTAGKHAVIGLIKQLAHEWGPRIRVNG
BPHB_PS (  6) EAVLITGGASGLG ( 58) GKIDTLIPNAG ( 45) GNVIFTISNAGFYPNGGGPLYTAAKQAIVGLVRELAFELAPYVRVNG
BUDC_KL (  3) KVALVTGAGQGIG ( 62) GGFNVIVNNAG ( 43) GKIVNACSQAGHVGNPELAVYSSSKFAVRGLTQTAARDLAPLGITVN
DHES_HU (  3) TVVLITGCSSGIG ( 66) GRVDVLVCNAG ( 42) GRVLVTGSVGGLMGLPFNDVYCASKFALEGLCESLAVLLLPFGVHLS
DHGB_BA (  8) KVVVITGSSTGLG ( 64) GKLDVMINNAG ( 44) TVINMSSVHEWKIPWPLFVHYAASKGGMKLMTETLALEYAPKGIRVN
DHII_HU ( 35) KKVIVTGASKGIG ( 63) GGLDMLILNHI ( 41) GSIVVVSSLAGKVAYPMVAAYSASKFALDGFFSSIRKEYSVSRVNVS
DHK1_ST ( 18) PVALVTGATSGIG ( 62) GTVDILVNNAG ( 44) GRIINIASTGGKQGVVHAVPYSASKHGVVGLTKALGLELARTGITVN
ENTA_EC (  6) KNVWVTGAGKGIG ( 67) GATDQLSKEDW ( 27) GAIVTVASDAAHTPRIGMSAYGASKAALKSLALSVGLELAGSGVRCN
FABG_EC (  6) KIALVTGASRGIG ( 59) GEVDILVNNAG ( 42) GRIITIGSVVGTMGNGGQANYAAAKAGLIGFSKSLAREVASRGITVN
FIXR_BR ( 37) KVMLLTGASRGIG (108) GLFDELRAASG (  0) SIVNVTSIAGSRVHPFAGSAYATSKAALASLTRELAHDYAPHGIRVN
GUTD_EC (  3) QVAVVIGGGQTLG ( 64) GRVDLLVYSAG ( 43) GRIIQINSKSGKVGSKHNSGYSAAKFGGVGLTQSLALDLAEYGITVH
HDHA_EC ( 12) KCAIITGAGAGIG ( 62) GKVDILVNNAG ( 41) GVILTITSMAAENKNINMTSYASSKAAASHLVRNMAFDLGEKNIRVN
LIGD_PS (  7) QVAFITGGASGAG ( 62) GQAPTLLSNTA ( 44) GHIVTVSSLGGFMGSALAGPYSAAKAASINLMEGYRQGLEKYGIGVS
NODG_RH (  7) RKALVTGASGAIG ( 59) EGVDILVNNAG ( 42) GRIINVTSVAGAIGNPGQTNYCASKAGMIGFSKSLAQEIATRNITVN
PGDH_HU (  6) KVALVTGAAQGIG ( 64) GRLDILVNNAG ( 37) GIIINMSSLAGLMPVAQQPVYCASKHGIVGFTRSAALAANLMNSGVR
PHBB_ZO (  3) RVALVTGGSRGIG ( 58) GPIDVLVNNAG ( 42) GRIVNISSINGQKGQMGQANYSAAKAGDLGFTKALAQEGAAKGITVN
RIDH_KL ( 15) KVAAITGAASGIG ( 59) GRLDIFHANAG ( 42) GDIIFTAVIAGVVPVIWEPVYTASKFAVQAFVHTTRRQVAQYGVRVG
TODD_PS (  6) EVALVTGGGAGLG ( 59) GKLDCLVGNAG ( 46) GSAIFTVSNAGFYPGGGGVLYTAGKHAVIGLIKQLAHEWGPRIRVNG
XYLL_PS (  8) KVAVITGAAQGIG ( 60) GRLDILINNVG ( 41) GSGAIVNVSSVATRGIHRVPYGAAKGGVNALTACLAFETAERGIRVN
YINL_LI (  6) KVIIITGASSGIG ( 62) GKVDAIFLNAG ( 42) GHIIATSSVAGLKAYPGGAVYGATKWAVRDLMEVLRMESAQEGTNIR
YRTP_BA (  7) KTALITGGGRGIG ( 62) GDIDILINNAG ( 42) GDIINISSTAGQRGAAVTSAYSASKFAVLGLTESLMQEVRKHNIRVS

              1111111111111       333333333333      4444444444          222222222222          55555
```

Figure V.8: **The blocks produced by** BLOCKMAKER **using the** MOTIF**-based algorithm.** The spacings between block elements are shown in parentheses. The relative location of the MEME motifs are shown in the last line.

of blocks using different algorithms. One of the algorithms is based on the MOTIF program [Smith *et al.*, 1990]. The other is based on a modification of the Gibbs sampler [Lawrence *et al.*, 1993].

The blocks chosen by the two algorithms are shown in Figures V.8 and V.9. The MOTIF blocks contain 29 of the 32 sequences in three blocks. The Gibbs sampler blocks contain 30 sequences in four blocks. It is surprising that BLOCKMAKER chooses to omit the sequence HDE_CANTR from both sets of blocks even though this sequence contains two copies of all the motifs (with the typical order and spacing) as a result of gene duplication.

The MEME motifs seem to capture the block structure better than the two BLOCKMAKER algorithms. MEME motifs 2, 4 and 5 are combined into one block

```
BDH_HUM ( 56) KAVLVTGCDSGFG ( 67) KGMWGLVNNAG ( 41) GRVVNISSMLGRMANPARSPYCITKFGVEAF ( 11) GVKVSVVEPG
2BHD_ST (  7) KTVIITGGARGLG ( 59) GSVDGLVNNAG ( 42) GSIVNISSAAGLMGLALTSSYGASKWGVRGL ( 11) RIRVNSVHPG
3BHD_CO (  7) KVALVTGGASGVG ( 59) GTLNVLVNNAG ( 41) GSIINMASVSSWLPIEQYAGYSASKAAVSAL ( 13) AIRVNSIHPD
ACT3_ST (  7) EVALVTGATSGIG ( 62) GPVDVLVNNAG ( 44) GRIVNIASTGGKQGVVHAAPYSASKHGVVGF ( 11) GITVNAVCPG
ADH_DRO (  7) KNVIFVAGLGGIG ( 64) KTVDVLINGAG ( 37) GIICNIGSVTGFNAIYQVPVYSGTKAAVVNF ( 11) GVTAYTVNPG
AP27_MO (  8) LRALVTGAGKGIG ( 54) GPVDLLVNNAA ( 43) GSIVNVSSMVAHVTFPNLITYSSTKGAMTML ( 11) KIRVNSVNPT
BA72_EU (  7) KVTIITGGTRGIG ( 64) GRLDVMINNAG ( 42) GVIINTASVTGIFGSLSGVGYPASKASVIGL ( 11) NIRVVGVAPG
BEND_AC ( 10) KVVIVTGAAQGIG ( 61) GRIDVLINNVG ( 43) GTIVNVSSIATRGIHRIPYSACKGGVNALTA (  9) GIRVNAVATG
BNZE_PS (  6) EVALVTGGGAGLG ( 59) GKLDCLVGNAG ( 46) GSAIFTVSNAGFYPGGGGVLYTAGKHAVIGL (  1) RIRVNGIAPG
BPHB_PS (  6) EAVLITGGASGLG ( 58) GKIDTLIPNAG ( 45) GNVIFTISNAGFYPNGGGPLYTAAKQAIVGL (  1) YVRVNGVGPG
BUDC_KL (  3) KVALVTGAGQGIG ( 62) GGFNVIVNNAG ( 43) GKIVNACSQAGHVGNPELAVYSSSKFAVRGL ( 11) GITVNGFCPG
DHES_HU (  3) TVVLITGCSSGIG ( 66) GRVDVLVCNAG ( 42) GRVLVTGSVGGLMGLPFNDVYCASKFALEGL ( 11) GVHLSLIECG
DHGB_BA (  8) KVVVITGSSTGLG ( 64) GKLDVMINNAG ( 43) GTVINMSSVHEWKIPWPLFVHYAASKGGMKL ( 12) GIRVNNIGPG
DHII_HU ( 35) KKVIVTGASKGIG ( 63) GGLDMLILNHI ( 41) GSIVVVSSLAGKVAYPMVAAYSASKFALDGF ( 11) RVNVSITLCV
DHK1_ST ( 18) PVALVTGATSGIG ( 62) GTVDILVNNAG ( 44) GRIINIASTGGKQGVVHAVPYSASKHGVVGL ( 11) GITVNAVCPG
DHMA_FL ( 15) KAAIVTGAAGGIG ( 57) GGLDILVAGGA ( 49) ARIITIGSVNSFMAEPEAAAYVAAKGGVAML ( 11) GILVNMIAPG
ENTA_EC (  6) KNVWVTGAGKGIG ( 52) ERLDALVNAAG ( 42) GAIVTVASDAAHTPRIGMSAYGASKAALKSL ( 11) GVRCNVVSPG
FABG_EC (  6) KIALVTGASRGIG ( 59) GEVDILVNNAG ( 42) GRIITIGSVVGTMGNGGQANYAAAKAGLIGF ( 11) GITVNVVAPG
FIXR_BR ( 37) KVMLLTGASRGIG ( 60) APLHALVNNAG ( 47) GSIVNVTSIAGSRVHPFAGSAYATSKAALAS ( 12) GIRVNAIAPG
FOX2_YE ( 10) RVVVITGAGGGLG ( 69) GRVDVLINNAG ( 42) GRIINTASPAGLFGNFGQANYSAAKMGLVGL ( 11) NINVNSIAPL
GUTD_EC (  3) QVAVVIGGGQTLG ( 64) GRVDLLVYSAG ( 43) GRIIQINSKSGKVGSKHNSGYSAAKFGGVGL ( 11) GITVHSLMLG
HDHA_EC ( 12) KCAIITGAGAGIG ( 62) GKVDILVNNAG ( 41) GVILTITSMAAENKNINMTSYASSKAAASHL ( 11) NIRVNGIAPG
NODG_RH (  7) RKALVTGASGAIG ( 59) EGVDILVNNAG ( 42) GRIINVTSVAGAIGNPGQTNYCASKAGMIGF ( 11) NITVNCVAPG
PGDH_HU (  6) KVALVTGAAQGIG ( 64) GRLDILVNNAG ( 37) GIIINMSSLAGLMPVAQQPVYCASKHGIVGF ( 13) GVRLNAICPG
PHBB_ZO (  3) RVALVTGGSRGIG ( 58) GPIDVLVNNAG ( 42) GRIVNISSINGQKGQMGQANYSAAKAGDLGF ( 11) GITVNAICPG
RIDH_KL ( 15) KVAAITGAASGIG ( 59) GRLDIFHANAG ( 42) GDIIFTAVIAGVVPVIWEPVYTASKFAVQAF ( 11) GVRVGAVLPG
TODD_PS (  6) EVALVTGGGAGLG ( 59) GKLDCLVGNAG ( 46) GSAIFTVSNAGFYPGGGGVLYTAGKHAVIGL (  1) RIRVNGIAPG
XYLL_PS (  8) KVAVITGAAQGIG ( 60) GRLDILINNVG ( 43) GAIVNVSSVATRGIHRVPYGAAKGGVNALTA (  9) GIRVNATAPG
YINL_LI (  6) KVIIITGASSGIG ( 62) GKVDAIFLNAG ( 42) GHIIATSSVAGLKAYPGGAVYGATKWAVRDL ( 70) NVNEFTVGPT
YRTP_BA (  7) KTALITGGGRGIG ( 62) GDIDILINNAG ( 42) GDIINISSTAGQRGAAVTSAYSASKFAVLGL ( 11) NIRVSALTPS

              1111111111111       333333333333      4444444444          222222222222      5555555555
```

Figure V.9: **The blocks produced by** BLOCKMAKER **using the Gibbs sampler-based algorithm.** The spacings between block elements are shown in parentheses. The relative location of the MEME motifs are shown in the last line.

by the MOTIFbased PROTOMAT algorithm. MEME motifs 2 and 4 are combined in the output of the Gibbs sampler-based BLOCKMAKER algorithm. Splitting out MEME motif 5 makes sense because the spacing between motifs 2 and 5 varies between nine and twelve residues as can be seen in Table V.17. This indicates that they are separate motifs whose exact distance from each other is not critical. Both MEME and the Gibbs sampler-based BLOCKMAKER algorithms make this split. MEME further splits the Gibbs sampler block into motifs 2 and 4. This is reasonable because their spacing varies between seven and ten residues. Splitting them allows them to be aligned independently when searching new sequences for the motifs. Neither BLOCKMAKER algorithm discovered MEME motif 6 which seems to be very typical of the family.

We searched the the BLOCKS version 8.0 (8/8/94) database with the distant homolog rat dihydroxipteridine reductase sequence. Ten possible hits were reported. The strongest hit was to two blocks labeled BL00061A and BL00061C in the BLOCKS database which were built from a family of short-chain alcohol dehydrogenases. The strongest match, scored in the 69.70 percentile for shuffled queries meaning such a hit would be expected by chance on about one query in a hundred according to the PROTOMAT documentation.

The BLOCKS database family BL00061 was produced the BLOCKMAKER program from the group of sequences identified by Prosite signature PS00061 which is the short-chain alcohol dehydrogenase family signature. The Prosite signature for the family is Y-[PSTAGCV]-[STAGCIV]-[STAGC]-K-x-[SAG]-[LIVMAG]-x(2)-[LIVMF] which does not detect the rat dihydropteridine reductase solely because it does not match the signature in the third position. The reductase has a methionine (M) in that position which is not among the residues allowed by the signature. The signature should probably be changed to allow for this family member. It is interesting to note that the Prosite signature for the short-chain alcohol dehydrogenase family corresponds almost exactly with the sec-

ond motif found by MEME in the dehydrogenase dataset. The MEME motif starts with the tyrosine (Y) and ends one residue past the end of the Prosite signature. The length and positioning of the motif is thus almost identical to that chosen by the Prosite authors.

Chapter VI

Conclusions and future directions

VI.A Conclusions

This dissertation has examined a pattern discovery problem which I call the approximate common substring problem and presented an algorithm, MEME, which solves it in several variations. Given only a set of strings, the algorithm finds one or more different motifs where a motif is an approximate string represented by a stochastic model. For each motif in the given set of strings, the algorithm discovers the width, the stochastic motif model and the location of its occurrences in the given strings. MEME is unique in its consistent use of clearly defined statistical models and statistically based heuristic functions to solving these variants of the ACS problem.

The contributions of this research include several techniques of general interest for increasing the robustness of the expectation maximization algorithm and the complexity of the models which it can learn. Experimental comparisons show that MEME is as sensitive and robust as other algorithms which solve more limited problems or are more *ad hoc*.

The solution to the ACS problem described in this dissertation postulates a stochastic model for the approximate substrings and the strings in which they

are embedded. Several different stochastic models are provided to match different background knowledge available from the domain providing the problem instance. The algorithm also allows complex prior distributions on the parameters of the string models to be specified in order to utilize background knowledge from the specific problem domain which generated the ACS instance.

Extensive experiments using instances from molecular biology show that MEME is robust and can solve ACS problem instances even with the least constrained string model, but also benefits from background knowledge about the best form for the string model and from informative prior distribution over its parameters. These experiments show that both DNA and protein problems can be solved. In both cases, the patterns found tend to be biologically significant and predictive of common structure and function. The common motif patterns and aligned occurrences of the motifs are informative to biologists in and of themselves and can also be used as probes for searching biological sequence databases. When used in this way, they provide a highly sensitive way of identifying distant structural, functional and evolutionary relationships among different proteins or genes. As part of this dissertation, I have developed tools for searching databases with substring patterns discovered by MEME and displaying the the block structure of and relationships among proteins.

Although the algorithm presented in this dissertation was primarily intended to solve ACS problems arising in molecular biology, it should work equally well with any problem for which the type of string models provided are reasonable. This conclusion is based upon the fact that the string models used, although well-motivated, are certainly not accurate models of DNA or protein sequences. Also, the sequence properties of DNA and protein sequences are quite diverse, yet MEME works well with both types. Furthermore, the primary method of incorporating background knowledge into the model—via prior distributions on the parameters of the model—is extremely general and can easily incorporate background knowledge

from a variety of problem domains.

Several techniques incorporated in the MEME algorithm are of independent interest. Statistical modeling in general, and in particular using mixture models for solving clustering problems, of which the ACS problem can be considered an example, is a powerful tool in the arsenal of machine learning approaches. Expectation maximization is a powerful paradigm for solving mixture model problems, and the technique of mining the training set for good starting points for EM should prove of general utility. The maximum likelihood ratio test-based heuristic which MEME uses as its objective function may also be useful in other contexts. Learning the parameters of multiple-component mixture models with EM is difficult at best because of a combinatoric explosion in the amount of computation required for the likelihood function. For this reason, the technique introduced here of using greedy search combined with merging previously found models with the current model in the E-step of EM should be useful with EM elsewhere.

VI.B Future directions

The research presented in this dissertation leads in several different directions. They include:

- Improvements to MEME in the areas of speed and versatility can be studied.
- Methods for assessing the statistical significance of the motifs discovered by MEME can be studied in more detail along the lines of Karlin and Altschul [1990].
- New uses for motifs discovered by MEME can be explored such as using them as building blocks in hidden Markov models (HMMs) [Rabiner, 1989].
- Last but not least, the techniques and algorithms developed for this dissertation can be extended to handle quite different kinds of ACS problems

involving continuous rather than discrete sequences or vector-valued discrete or continuous sequences.

Improving the speed of MEME is one area for future research. The quadratic time complexity of MEME puts practical limits on the size of sequence datasets it can handle. It might be possible to develop more efficient algorithms for searching for starting points for EM. The convergence of the EM algorithm might also be amenable to improvement. Improvements in these two areas could possibly reduce the time complexity to be linear in dataset size at some cost in terms of robustness and sensitivity to weak motifs. More efficient ways for searching the space of possible motif widths (W) and motif frequency (λ) might also be possible, but these could only produce constant factor speedups in the overall algorithm.

Multiple motifs discovered in sets of protein sequences could be used as building blocks in hidden Markov models of protein families. Such models would be much more sensitive than single motifs for detecting distant family members. They would also provide a new method for doing multiple sequence alignment. In addition, the problem of deciding how many motifs are enough for a given set of sequences might be eliminated since insignificant motifs would be on low probability paths in the HMM.

Building HMMs from MEME motifs would have three advantages over current techniques [Krogh *et al.*, 1994], [Baldi *et al.*, 1994] of producing HMMs of protein sequence families. First, the number of parameters which would need to be learned would be greatly reduced since long stretches of sequence between motifs could be modeled by a single state in the model instead of multiple states. Second, the size of the model could be automatically determined and could be initialized to very good starting parameter values directly from the motifs generated by MEME. This would reduce the training time and might improve the robustness of the final model. Third, because fewer parameters need to be learned and the initial values are good, the topology of the model could be more complicated, allowing

it to better model insertions, deletions, repetitions, cyclic permutations and other evolutionary events at the motif level which shape protein sequences [Gribskov and Devereux, 1991].

Motifs might be useful for secondary structure prediction directly from sequence information [Cohen *et al.*, 1986]. This problem might be approached by developing sequence models that model correlations between primary sequence and secondary structure. Such models could be learned using a variant of MEME. It might even be possible to use motifs to help predict the tertiary (three dimensional) structure of proteins since correlations exist between tertiary structure and sequence hydrophobicity which can be inferred from the primary sequence [Sweet and Eisenberg, 1983].

Sequential continuous valued data such as time series data can also be viewed as a string and the techniques used by MEME applied to discover noisy patterns. It should also be possible to discover patterns in strings whose elements are vectors of discrete and/or continuous values. An algorithm based on MEME could be developed to solve analogs of the noisy common substring problem which can be expressed in this way. Any developments in this direction will need to focus on creative ways to utilize background knowledge about the problem domain through priors on the model parameters.

Appendix A

Sample MEME output

Figure A.1 is the output for the fifth motif discovered by MEME in a group of sequences. The output format for earlier and subsequent motifs is the same.

The first six lines in Figure A.1 give various statistical measures of the strength of the motif and other motif data. The best measure of motif strength is probably its information content (IC), shown on the sixth line. The next lines show a default threshold which can be used for searches with the motif on small datasets.

The occurrences (hits greater than the default threshold) of the motif in the original dataset are shown next under a header line labeling the six columns. The data in the columns in each line are, from left to right,

- (`sequence`) the sequence identifier (i.e., `P23388`),
- (`start`) the starting position of the hit (i.e., `5`),
- (`IC`) the IC score of the hit, (i.e., `43.48`),
- (`pre`) the left flanking sequence, (i.e., `VLSG`), and
- (`site`) the actual sequence matching the motif (i.e., `RAIDLRDAGQRVLQHL`),

```
Motif   5

(best) e_ll_0  -7031 e_ll  -6814 ll  -6750 sig 9.980e-001
(best) lrt 5.396e-001 bonferroni 1.000e+000 root 9.980e-001

(best) RAADIRDVTKRVLAHLLGVTI --> RAADIRDVxxRVLxHL
(best) w  16 nsites   8.0 lambda 0.0052288 IC/col  2.269

(best) IC 36.298

Bayes optimal threshold for information content scores =   7.5718
Alignment of sites with IC scores over 7.57175:
sequence          start     IC        pre site             post
--------          -----     --        --- ----             ----
U12340                1  52.50            RAADIRDVTKRVLAHL LGVTISNPSL
P08838                1  46.94            RAADIRDVTKRVTGHL LGVEIPNPSM
L15191                1  51.32            RAADIRDVAKRVLAHL LGVELPNPAT
U15110                1  35.40            RSADIKDVSLRIISHI LGLEIHDLST
M21450                1  41.54            RAADVRDIGKRLLRNI LGLAIIDLSA
P32670                2  39.45              E RALDVRDVCFQLLQQI YGEQRFPAPG
P23388                5  43.48          VLSG RAIDLRDAGQRVLQHL GRVRTGETHL
Z37113                1  45.39            RAADLRDVGRRVLAQL DAAAAGAGLT
(consensus)      (   8)                    RAADIRDVxxRVLxHL

Information content of positions in motif:
                                    7.1
                                    6.4
                                    5.7
                                    5.0
                                    4.3
Info Content                        3.5 *
                                    2.8 *  * **   *   *
                                    2.1 *  * **   *   *
                                    1.4 ** ***** **** **
                                    0.7 ****************
                                    0.0 ----------------
(consensus)                             RAADIRDVxxRVLxHL
                                         SL V  I   LI QI
                                          I L      I  N

                                     A  0870000120000300
                                     C  0000000010000000
                                     D  0009009000000000
                                     E  0000000000000000
                                     F  0000000001000000
                                     G  0000000030000100
                                     H  0000000000000070
                                     I  0010600100011003
                                     K  0000000005000000
                                     L  0010100001027006
                                     M  0000000000000000
                                     N  0000000000000010
                                     P  0000000000000000
                                     Q  0000000001000210
                                     R  9000090001900100
                                     S  0100000010000100
                                     T  0000000020001000
                                     V  0000200700060000
                                     W  0000000000000000
                                     Y  0000000000000000
```

Figure A.1: **Sample** MEME **output.**

- (`post`) the right flanking sequence, (i.e., `GRVRTGETHL`).

The next lines in the output of MEME show the information content of the motif on a column by column basis. Information content depends on the frequencies of the letters in a given column compared to the overall frequencies of those letters in the group of sequences. The more conserved the position is and the more rare the conserved letters are, the higher information content is.

Beneath the motif information content plot is the three-level consensus sequence of the motif. This shows the (up to) three most common letters occurring in each column of the motif. The rules for printing letters are:

- If three or fewer letters have combined frequency of 0.8 they are printed, most frequent letter first, one above the other.
- If no three letters have combined probability above 0.8, "`x`" is printed.

This provides a simplified picture of the conserved letters in each column of the motif at a glance.

The parameters θ_1 of the motif are shown below the consensus sequence. They are shown rounded to one digit and multiplied by 10 to conserve space. MEME also prints the log-odds matrix corresponding to the motif. This is used for searching by programs PROBER and NOTE and is omitted from the sample output shown in Figure A.1.

Appendix B

Prosite protein families

The Prosite families used in the experiments in Chapter V and described in Section V.A.1 are shown in more detail in the following tables.

Table B.1 gives the actual Prosite signatures for the 75 families. Families are identified by their Prosite accession number. The meaning of the signature notation is:

- capital letters stand for the standard one-letter amino acid codes
- x means "any amino acid"
- [...] means "one of ..."
- {...} means "anything but ..."
- ⟨anything⟩(n) is shorthand for "⟨anything⟩⟨anything⟩...⟨anything⟩" repeated n times.

Tables B.2 and B.3 describe the 75 datasets in terms of how many known occurrences of the motif that characterizes them they contain as well as other statistics about the number of sequences in the datasets and their lengths.

Table B.4 describes the Prosite families among the 75 families which contain more than one known motif. Prosite families with at least five sequences

in common with other Prosite families are shown. Families in the set of 75 which overlap significantly with other Prosite families are listed in the leftmost column (family X). The degree of overlap with some other Prosite family (family Y), given as the number of sequences in common ($|I| = |X \cap Y|$) divided by the number of sequences in family X or family Y, is shown in succeeding columns.

Tables B.5 and B.6 give the performance of the Prosite motifs at characterizing their respective protein families in terms of the numbers of sequences in SWISS-PROT release 27 that they correctly or incorrectly classify.

accession number	*signature*
PS00030	[RK]-G-{EDRKHPCG}-[AGSCI]-[FY]-[LIVA]-x-[FYM].
PS00037	W-[ST]-x(2)-E-[DE]-x(2)-[LIV].
PS00038	K-[LIVMAG]-x-[IT]-[IL]-x(2)-[STAV]-x(2)-[YHV]-[LIVMA]-x(2)-[LIVM].
PS00043	E-x(2)-[LIVM]-x(3)-[LIVMF]-x-[LIVMF]-[NSTK]-R-x(2)-[LIVM]-x(3)-[LIVM]-x(2)-L.
PS00060	G-x(2)-H-x(2)-A-H-x(2)-G-x(5)-P-H-G.
PS00061	Y-[PSTAGCV]-[STAGCIV]-[STAGC]-K-x-[SAG]-[LIVMAG]-x(2)-[LIVMF].
PS00070	[FYV]-x(3)-G-[QE]-x-C-[LIVMGSTNC]-[AGCN]-x-[GSTDNE].
PS00075	[LIF]-G-x(4)-[LIVMF]-P-W.
PS00077	W-x-H-H-[LMF].
PS00079	G-x-[FYW]-x-[LIVMFYW]-x-[CST]-x(8)-G-[LM]-x(3)-[LIVMFYW].
PS00092	[LIVMAC]-[LIVMFYA]-x-[DN]-P-P-[FY].
PS00095	[RKQTF]-x(2)-G-N-[STAG]-[LIVM]-x(3)-[LIVM]-x(3)-[LIVM]-x(3)-[LIVM].
PS00099	[AG]-[LIVMA]-x-[STAG]-x-C-x-G-x-G-x-[AG].
PS00118	C-C-x(2)-H-x(2)-C.
PS00120	[LIV]-x-[LIVFY]-[LIVST]-G-[HYWV]-S-x-G-[GSTAC].
PS00133	H-[STAG]-x(3)-[LIVM]-x(2)-[LIVMFYW]-P-[FYW].
PS00141	[LIVFA]-D-T-G-[STA]-[STAPN].
PS00144	[LIVM]-x(2)-T-G-G-T-I-[AG].
PS00158	E-G-x-[LS]-L-K-P-N.
PS00180	[FYW]-D-G-S-S.
PS00185	[RK]-x-[STA]-x(2)-S-x-C-Y-[SL].
PS00188	[LIVM]-x-[AV]-M-K-[MA]-x(3)-[LIVM].
PS00190	C-{CPWHF}-{CPWR}-C-H-{CFYW}.
PS00194	[STA]-x-[WG]-C-[AGV]-[PH]-C.
PS00198	C-x(2)-C-x(2)-C-x(3)-C-[PEG].
PS00209	Y-[FYW]-x-E-D-[LIVM]-x(2)-N-x(6)-H-x(3)-P.
PS00211	[LIVMFY]-S-[SAG]-G-x(3)-[RKA]-[LIVMYA]-x-[LIVMF]-[SAG].
PS00215	P-x-[DE]-x-[LIVAT]-[RK]-x-[LRH]-[LIVMFY].
PS00217	[LIVMF]-x-G-[LIVMFA]-x(2)-G-x(8)-[LIFY]-x(2)-[EQ]-x(6)-[RK].
PS00225	[LIVMFYWA]-x-{DEHRKSTP}-[FY]-[DEQHKY]-x(3)-[FY]-x-G-x(4)-[LIVMFCST].
PS00281	C-x-[SAD]-[STA]-C-x(2)-C.
PS00283	[LIVM]-x-D-x-[EDNTY]-[DG]-[RKHDENQ]-x-[LIVM]-x(5)-Y-x-[LIVM].
PS00287	Q-[LIVT]-V-[SAG]-G-x(2)-[LIVMFY]-x-[LIVMFY]-x-[LIVMFY].
PS00301	D-x(4)-E-x(3)-[GC]-x-T-[IV].
PS00338	C-[LIVMFY]-x(2)-D-[LIVMFYSTA]-x(5)-[LIVMFY]-x(2)-[LIVMFY]-x(2)-C.
PS00339	[GSTALVF]-{DENQHRKP}-[GSTA]-[LIVMF]-[DE]-R-[LIVMF]-x-[LIVMSTAG]-[LIVMFY].
PS00340	[STGL]-x-W-[SG]-x-W-S.
PS00343	L-P-x-T-G-[STGAVDE].
PS00372	[LIVM]-P-H-G-T.
PS00399	[LIVMF]-G-H-A-G-A.
PS00401	K-x-[NQEK]-[GT]-G-[DQ]-x-[LIVM]-x(3)-Q-S.
PS00402	[LIVMFY]-x(8)-[EQR]-[STA]-[STAG]-x(3)-G-[LIVMFYSTAC]-x(5)-[LIVMFYSTA]-x(4)-[LIVMFY]-[PKR].
PS00422	[DE]-[SN]-L-[SAN]-x(2)-[DE]-x-E-L.
PS00435	[DET]-[LIVMTA]-x(2)-[LIVM]-[LIVMSTAG]-[SAG]-[LIVMSTAG]-H-[STA]-[LIVMFY].
PS00436	[SGATV]-x(3)-[LIVMA]-R-[LIVMA]-x-[FW]-H-x-[SAC].
PS00490	[STA]-x-[STAC](2)-x(2)-[STA]-D-[LIVM](2)-L-P-x-[STAC](2)-x(2)-E.
PS00548	[LIVMF]-[RE]-x-G-x(2)-[KRQ]-x(3)-[DNS]-x(2)-[FYW]-[SAV]-[NQDE].
PS00589	[GA]-[KR]-x(4)-[KR]-S-[LIVMF](2)-x-[LIVM]-x(2)-[LIVM]-[GA].
PS00599	T-[LIVMFYW]-[SAG]-K-[SAG]-[LIVMFYW]-[GA]-x(2)-[SAG].
PS00606	G-x(4)-[LIVMAP]-x(2)-[AGC]-C-[STA](2)-[STAG]-x(3)-[LIVMF].
PS00624	G-[STA]-x(2)-[ST]-P-x-[LIVM](2)-x(2)-S-G-[LIVM]-G.
PS00626	[LIVMFA]-[STAGC](2)-G-x(2)-H-[STAGLI]-[LIVMFA]-x-[LIVM].
PS00637	C-x(2)-C-x-G-x-G-[AGS]-x(2)-G.
PS00639	[LIVMGSTAN]-x-H-[GSA]-[LIVM]-x-[LIVMAT](2)-G-x-[GSNH].
PS00640	[FY]-[WI]-[LIVT]-x-[KRQAG]-N-[ST]-W-x(3)-[FYW]-G-x(2)-G-[FYW]-[LIVMFYG]-x-[LIVMF].
PS00643	R-C-[LIVM]-x-C-x-R-C-[LIVM]-x-F.
PS00656	[LIVMFY]-[LIV](3)-E-P-D-x-[LIV].
PS00659	[LIV]-[LIVMFYWGA](2)-[DNEG]-[LIVMGST]-x-N-E-[PV]-[RHDNSTLIVFY].
PS00675	[LIVMFY](3)-x-G-[DE]-[ST]-G-[ST]-G-K-x(2)-[LIVMFY].
PS00676	G-x-[LIVMF]-x(2)-A-[DNEQASH]-G-G-[STI]-[LIVMFY](3)-D-E-[LIVM].
PS00678	[LIVMSTAC]-[LIVMFYWSTAGC]-[LIMSTAG]-[LIVMSTAGC]-x(2)-[DN]-x(2)-[LIVMWSTAC]-x-[LIVMFSTAG]-W-[DEN]-[LIVMFSTAGCN].
PS00687	[LIVMFG]-E-[ILSTA]-[GS]-G-[KN]-[SAN]-[TAPF].
PS00697	[EDQH]-x-K-x-[DN]-G-x-R-[GACV].
PS00700	G-x-[DNS]-x-[QE]-x-[LIVM]-[GST]-[NQEDKR]-x-[AC]-A-x-[LIVM].
PS00716	[STN]-x(2)-[DEQ]-[LIVM]-[GAS]-x(4)-[LIVMF]-[STG]-x(3)-[LIVMA]-x-[NQR]-[LIVMA]-[EQH]-x(3)-[LIVM]-x(2)-[LIVM].
PS00741	L-x(2)-[LIVMFYW]-L-x(2)-P-[LIVM]-x(2)-[LIVM]-x-[KRS]-x(2)-L-x-[LIVM]-x-[DE]-[LIVM]-x(3)-[ST].
PS00760	K-R-[LIVMSTA](2)-G-[LIVM]-P-G-D-x-[LIVM].
PS00761	[LIVMFYW](2)-x(2)-G-D-N-x(3)-[SND]-x(2)-[SG].
PS00831	R-Q-R-G-T-K-x(3)-G-x-N-V-G-x-G-x-D.
PS00850	[STIV]-x-R-[VT]-[CSA]-G-Y-x-[GAV].
PS00867	[LIVMF]-[LIN]-E-[LIVMCA]-N-[PATLIVM]-[KR]-[LIVMSTAC].
PS00869	G-V-E-G-G-H-x-I-D.
PS00881	[LIVM](2)-V-H-N-[STC].
PS00904	[PSIAV]-x-[NDFV]-[NEQIY]-x-[LIVMAGP]-W-[NQSTHF]-[FYHQ]-[LIVMR].
PS00933	[LIVMFYGST]-x-[PST]-x(2)-K-[LIVMFYW]-x-W-[LIVMF]-x-[DENR]-[ENH].

Table B.1: **Prosite signatures of the 75 Prosite families.**

Prosite accession number	*width of sign.*	*number of sites*	*number of seqs.*	*minimum sequence length*	*maximum sequence length*	*mean sequence length*	*total dataset size*
PS00030	8	98	59	157	713	411.8	24297
PS00037	9	35	18	151	811	492.9	8872
PS00038	15	83	90	133	710	364.3	32786
PS00043	22	10	10	236	254	241.6	2416
PS00060	19	5	7	370	890	453.6	3175
PS00061	11	81	82	201	906	275.3	22577
PS00070	12	32	34	140	902	500.9	17029
PS00075	9	33	33	35	608	213.8	7054
PS00077	5	53	53	109	698	454.5	24088
PS00079	21	18	12	548	2351	978.2	11738
PS00092	7	35	35	228	997	429.0	15015
PS00095	19	29	33	309	1502	471.8	15571
PS00099	12	13	14	387	547	418.3	5856
PS00118	8	108	110	39	162	125.6	13821
PS00120	10	31	36	277	690	426.6	15357
PS00133	11	19	19	303	477	408.9	7769
PS00141	6	87	50	52	587	384.1	19204
PS00144	9	8	8	26	348	287.4	2299
PS00158	8	17	20	349	394	364.4	7287
PS00180	5	52	55	72	729	399.4	21966
PS00185	10	9	10	311	338	328.8	3288
PS00188	10	15	15	70	2345	951.3	14270
PS00190	6	265	223	35	596	142.4	31752
PS00194	7	63	48	45	645	230.5	11063
PS00198	12	148	109	38	793	144.8	15784
PS00209	20	11	14	623	759	677.2	9481
PS00211	12	122	119	163	1548	567.2	67499
PS00215	9	88	39	286	436	330.9	12906
PS00217	26	42	46	220	884	517.4	23800
PS00225	16	165	47	30	252	179.6	8441
PS00281	8	23	22	53	133	74.0	1629
PS00283	17	29	30	20	221	175.4	5263
PS00287	12	38	32	98	644	244.8	7834
PS00301	13	105	110	389	858	527.4	58015
PS00338	18	83	86	85	267	211.0	18145
PS00339	10	31	38	201	967	518.7	19710
PS00340	7	23	37	292	897	529.0	19574
PS00343	6	24	25	369	1902	887.1	22177

Table B.2: Characteristics of the individual Prosite datasets.

Prosite accession number	*width of sign.*	*number of sites*	*number of seqs.*	*minimum sequence length*	*maximum sequence length*	*mean sequence length*	*total dataset size*
PS00372	5	7	7	59	637	304.3	2130
PS00399	6	4	4	288	1100	502.5	2010
PS00401	13	5	5	311	352	336.0	1680
PS00402	29	32	39	147	514	303.6	11842
PS00422	10	11	12	446	677	562.2	6747
PS00435	11	38	41	158	933	471.8	19345
PS00436	12	31	40	292	933	479.7	19187
PS00490	18	7	9	684	1246	909.9	8189
PS00548	16	15	18	209	562	282.6	5086
PS00589	16	10	10	85	827	191.7	1917
PS00599	10	20	21	356	642	451.5	9482
PS00606	17	20	17	401	3567	1438.2	24449
PS00624	15	7	9	546	664	596.4	5368
PS00626	11	22	6	421	547	472.3	2834
PS00637	12	8	9	190	409	360.6	3245
PS00639	11	53	62	73	1597	402.3	24944
PS00640	20	52	62	73	1597	402.3	24944
PS00643	11	4	5	233	744	620.6	3103
PS00656	9	5	5	388	471	438.0	2190
PS00659	10	38	40	343	1331	550.7	22027
PS00675	14	30	36	302	938	508.1	18293
PS00676	16	29	36	302	938	508.1	18293
PS00678	15	77	26	317	788	457.0	11883
PS00687	8	30	33	430	902	511.8	16889
PS00697	9	11	11	346	919	567.6	6244
PS00700	14	12	13	101	193	177.0	2301
PS00716	27	31	36	125	708	341.3	12288
PS00741	26	5	6	736	1271	977.7	5866
PS00760	11	5	8	167	324	230.2	1842
PS00761	14	8	8	167	324	230.2	1842
PS00831	18	5	6	84	371	158.2	949
PS00850	9	4	4	605	765	710.2	2841
PS00867	8	28	20	398	2345	1465.0	29301
PS00869	9	5	5	409	411	410.0	2050
PS00881	6	3	3	790	1702	1187.7	3563
PS00904	10	20	4	290	377	330.8	1323
PS00933	13	11	11	454	709	507.9	5587
mean	12	38	34	256	841	463	12945
sd	5	44	36	180	585	270	11922

Table B.3: **Characteristics of the individual Prosite datasets (continued).**

family X	*family* Y	*overlap* $\lvert I\rvert$	*overlap* $\frac{\lvert I\rvert}{\lvert X\rvert}$	*overlap* $\frac{\lvert I\rvert}{\lvert Y\rvert}$	*family* Y	*overlap* $\lvert I\rvert$	*overlap* $\frac{\lvert I\rvert}{\lvert X\rvert}$	*overlap* $\frac{\lvert I\rvert}{\lvert Y\rvert}$	*family* Y	*overlap* $\lvert I\rvert$	*overlap* $\frac{\lvert I\rvert}{\lvert X\rvert}$	*overlap* $\frac{\lvert I\rvert}{\lvert Y\rvert}$
PS00037	PS00334	17	0.94	0.94								
PS00060	PS00913	7	1.00	1.00								
PS00070	PS00687	33	0.97	1.00								
PS00079	PS00080	9	0.75	1.00								
PS00095	PS00094	33	1.00	0.92								
PS00099	PS00737	14	1.00	1.00	PS00098	14	1.00	1.00				
PS00118	PS00119	109	0.99	1.00								
PS00133	PS00132	19	1.00	1.00								
PS00144	PS00917	7	0.88	1.00								
PS00180	PS00181	53	0.96	0.98	PS00182	17	0.31	1.00				
PS00185	PS00186	10	1.00	1.00								
PS00188	PS00867	8	0.53	0.40	PS00866	8	0.53	0.40				
PS00194	PS00014	12	0.25	0.26								
PS00198	PS00197	6	0.06	0.07								
PS00209	PS00210	14	1.00	1.00	PS00498	7	0.50	0.37				
PS00217	PS00216	46	1.00	1.00								
PS00338	PS00266	84	0.98	1.00								
PS00339	PS00179	38	1.00	1.00								
PS00340	PS00241	37	1.00	1.00								
PS00401	PS00757	5	1.00	1.00								
PS00422	PS00423	8	0.67	1.00								
PS00435	PS00436	40	0.98	1.00								
PS00436	PS00435	40	1.00	0.98								
PS00490	PS00551	9	1.00	1.00	PS00932	9	1.00	1.00				
PS00548	PS00734	18	1.00	1.00								
PS00589	PS00369	10	1.00	0.83								
PS00606	PS00012	8	0.47	0.20								
PS00624	PS00623	9	1.00	1.00								
PS00626	PS00625	6	1.00	1.00								
PS00637	PS00636	9	1.00	0.53								
PS00639	PS00139	59	0.95	0.91	PS00640	62	1.00	1.00	PS00018	6	0.10	0.03
PS00640	PS00139	59	0.95	0.91	PS00639	62	1.00	1.00	PS00018	6	0.10	0.03
PS00643	PS00641	5	1.00	1.00	PS00642	5	1.00	1.00				
PS00656	PS00655	5	1.00	1.00								
PS00659	PS00448	6	0.15	0.55								
PS00675	PS00676	36	1.00	1.00	PS00688	36	1.00	1.00				
PS00676	PS00675	36	1.00	1.00	PS00688	36	1.00	1.00				
PS00687	PS00070	33	1.00	0.97								
PS00697	PS00333	11	1.00	1.00								
PS00700	PS00525	13	1.00	0.93								
PS00716	PS00715	35	0.97	0.97								
PS00760	PS00501	8	1.00	1.00	PS00761	8	1.00	1.00				
PS00761	PS00760	8	1.00	1.00	PS00501	8	1.00	1.00				
PS00867	PS00188	8	0.40	0.53	PS00866	20	1.00	1.00				
PS00933	PS00445	11	1.00	1.00								

Table B.4: **Prosite families containing multiple motifs.**

Prosite accession number	*recall*	*precision*	*true positives (tp)*	*false positives (fp)*	*false negatives (fn)*
PS00030	0.9322	0.7639	55	17	4
PS00037	1.0000	0.4186	18	25	0
PS00038	0.9222	0.7905	83	22	7
PS00043	1.0000	0.7692	10	3	0
PS00060	0.7143	1.0000	5	0	2
PS00061	0.9634	0.7524	79	26	3
PS00070	0.9412	0.8205	32	7	2
PS00075	1.0000	0.6600	33	17	0
PS00077	1.0000	0.8154	53	12	0
PS00079	0.9167	0.6111	11	7	1
PS00092	0.9714	0.8095	34	8	1
PS00095	0.8788	0.8529	29	5	4
PS00099	0.9286	0.6842	13	6	1
PS00118	0.9818	0.9153	108	10	2
PS00120	0.8611	0.7561	31	10	5
PS00133	1.0000	0.6786	19	9	0
PS00141	0.9800	0.5904	49	34	1
PS00144	1.0000	0.7273	8	3	0
PS00158	0.8500	1.0000	17	0	3
PS00180	0.9455	0.8125	52	12	3
PS00185	0.9000	0.9000	9	1	1
PS00188	1.0000	0.7500	15	5	0
PS00190	0.9821	0.5601	219	172	4
PS00194	1.0000	0.7869	48	13	0
PS00198	0.9541	0.8814	104	14	5
PS00209	0.7857	1.0000	11	0	3
PS00211	0.8908	0.8760	106	15	13
PS00215	0.9744	0.1929	38	159	1
PS00217	0.9130	0.6000	42	28	4
PS00225	1.0000	0.3013	47	109	0
PS00281	1.0000	0.5946	22	15	0
PS00283	0.9667	0.7632	29	9	1
PS00287	0.8750	0.7568	28	9	4
PS00301	0.9459	0.8468	105	19	6
PS00338	0.9651	0.8830	83	11	3
PS00339	0.8158	0.3163	31	67	7
PS00340	0.6216	0.5000	23	23	14
PS00343	0.9600	0.1875	24	104	1

Table B.5: **Performance of Prosite signatures at characterizing protein families.**

Prosite accession number	*recall*	*precision*	*true positives (tp)*	*false positives (fp)*	*false negatives (fn)*
PS00372	1.0000	0.3684	7	12	0
PS00399	1.0000	0.8000	4	1	0
PS00401	0.8333	1.0000	5	0	1
PS00402	0.8205	0.7273	32	12	7
PS00422	0.9167	0.7333	11	4	1
PS00435	0.9268	0.8261	38	8	3
PS00436	0.7750	0.9118	31	3	9
PS00490	0.7778	1.0000	7	0	2
PS00548	0.8333	0.8824	15	2	3
PS00589	1.0000	0.8333	10	2	0
PS00599	0.9524	0.7692	20	6	1
PS00606	1.0000	0.8095	17	4	0
PS00624	0.7778	1.0000	7	0	2
PS00626	1.0000	0.2000	6	24	0
PS00637	0.8889	0.8000	8	2	1
PS00639	0.8548	0.7681	53	16	9
PS00640	0.8387	1.0000	52	0	10
PS00643	0.8000	1.0000	4	0	1
PS00656	1.0000	0.6250	5	3	0
PS00659	0.9500	0.8636	38	6	2
PS00675	0.8333	1.0000	30	0	6
PS00676	0.8056	1.0000	29	0	7
PS00678	1.0000	0.3377	26	51	0
PS00687	0.9091	0.8333	30	6	3
PS00697	1.0000	0.7333	11	4	0
PS00700	0.9231	0.9231	12	1	1
PS00716	0.8378	0.9688	31	1	6
PS00741	0.8333	1.0000	5	0	1
PS00760	0.6250	1.0000	5	0	3
PS00761	1.0000	0.7273	8	3	0
PS00831	0.8333	1.0000	5	0	1
PS00850	1.0000	0.8000	4	1	0
PS00867	1.0000	0.7407	20	7	0
PS00869	1.0000	0.8333	5	1	0
PS00881	1.0000	0.7500	3	1	0
PS00904	1.0000	0.3333	4	8	0
PS00933	1.0000	0.7857	11	3	0
mean	0.92	0.75	31.09	15.97	2.48
sd	0.09	0.21	34.31	31.65	3.12

Table B.6: **Performance of Prosite signatures at characterizing protein families (continued).**

Bibliography

[Abramowitz and Stegun, 1972] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*. Dover Publications, Inc., 1972.

[Altschul and Lipman, 1990] S. F. Altschul and D. J. Lipman. Protein datasase searches for multiple alignments. *Proc. Natl. Acad. Sci. USA*, 87(14):5509–5513, 1990.

[Apostolico *et al.*, 1992] A. Apostolico, S. Browne, and C. Guerra. Fast linear-space computations of longest common subsequences. *Theoretical Computer Science*, 92(1):3–17, 1992.

[Bairoch, 1992] Amos Bairoch. PROSITE: a dictionary of sites and patterns in proteins. *Nucleic Acids Research*, 20:2013–2018, 1992.

[Bairoch, 1993] Amos Bairoch. The PROSITE dictionary of sites and patterns in proteins, its current status. *Nucleic Acids Research*, 21(13):3097–3103, July 1993.

[Bairoch, 1994] Amos Bairoch. The SWISS-PROT protein sequence data bank: current status. *Nucleic Acids Research*, 22(17):3578–3580, September 1994.

[Baker, 1994] M. E. Baker. Sequence analysis of steroid- and prostaglandin-metabolizing enzymes: Application to understanding catalysis. *Steroids*, 59:248–258, 1994.

[Baldi *et al.*, 1994] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. Mcclure. Hidden Markov models of biological primary sequence information. *Proc. Natl. Acad. Sci. USA*, 91(3):1059–1063, 1994.

[Berg and von Hippel, 1988] Otto G. Berg and Peter H. von Hippel. Selection of DNA binding sites by regulatory proteins. *Journal of Molecular Biology*, 200:709–723, 1988.

[Boguski *et al.*, 1992a] M. S. Boguski, R. C. Hardison, S. Schwartz, and W. Miller. Analysis of conserved domains and sequence motifs in cellular regulatory proteins and locus control regions using new software tools for multiple alignment and visualization. *New Biologist*, 4(3):247–260, 1992.

[Boguski *et al.*, 1992b] M. S. Boguski, A. W. Murray, and S. Powers. Novel repetitive sequence motifs in the alpha and beta subunits of prenyl-protein transferases and homology of the alpha subunit to the MAD2 gene product of yeast. *New Biologist*, 4(4):408–411, 1992.

[Brown *et al.*, 1993] Michael Brown, Richard Hughey, Anders Krogh, I. Saira Mian, Kimmen Sjolander, and David Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In *Intelligent Systems for Molecular Biology*, pages 47–55. AAAI Press, 1993.

[Cardon and Stormo, 1992] Lon R. Cardon and Gary D. Stormo. Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments. *Journal of Molecular Biology*, 223:159–170, 1992.

[Carillo and Lipman, 1988] H. Carillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Math*, 48:1072–1082, 1988.

[Chang and Lawler, 1994] W. I. Chang and E. L. Lawler. Sublinear approximate string matching and biological applications. *Algorithmica*, 12(4–5):327–344, 1994.

[Chin and Poon, 1994] F. Chin and C. K. Poon. Performance analysis of some simple heuristics for computing longest common subsequences. *Algorithmica*, 12(4–5):293–311, 1994.

[Cohen *et al.*, 1986] F. E. Cohen, R. A. Abarbanel, I. D. Kuntz, and R. J. Fletterick. Turn prediciton in proteins using a pattern matching approach. *Biochemistry*, 25:266–275, 1986.

[Dayhoff *et al.*, 1983] Margaret O. Dayhoff, Winona A. Barker, and Lois T. Hunt. Establishing homologies in protein sequences. *Methods in Enzymology*, 91:524–545, 1983.

[de Crombrugghe *et al.*, 1984] Benoit de Crombrugghe, Stephen Busby, and Henri Buc. Cyclic AMP receptor protein: Role in transcription activation. *Science*, 224:831–838, May 1984.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[Duda and Hart, 1973] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, Inc., 1973.

[Efron, 1983] Bradley Efron. Estimating the error rate of a prediction rule: Improvement on cross-validation. *Journal of the American Statistical Association*, 78(382):316–331, June 1983.

[Feng and Doolittle, 1987] D. Feng and R. F. Doolittle. Progressive multiple sequence alignment. *Journal of Molecular Evolution*, 25:351–360, 1987.

[George *et al.*, 1986] D. George, W. Barker, and L. Hunt. The Protein Identification Resource. *Nucleic Acids Research*, 14:11–15, 1986.

[Gribskov and Devereux, 1991] Michael Gribskov and John Devereux. *Sequence Analysis Primer.* Stockton Press, 1991.

[Gribskov *et al.*, 1990] Michael Gribskov, Roland Luthy, and David Eisenberg. Profile analysis. *Methods in Enzymology*, 183:146–159, 1990.

[Harley and Reynolds, 1987] C. B. Harley and R. P. Reynolds. Analysis of *E. coli* promoter sequences. *Nucleic Acids Research*, 15:2343–2361, 1987.

[Henikoff and Henikoff, 1991] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Research*, 19(23):6565–6572, 1991.

[Hertz *et al.*, 1990] Gerald Z. Hertz, George W. Hartzell, III, and Gary D. Stormo. Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Computer Applications in Biosciences*, 6(2):81–92, 1990.

[Hunter, 1983] Lawrence Hunter, editor. *Artificial intelligence and molecular biology.* MIT Press, 1983.

[Karlin and Altschul, 1990] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. Natl. Acad. Sci. USA*, 87:2264–2268, 1990.

[Kendall *et al.*, 1983] Sir Maurice Kendall, Alan Stuart, and J. Keith Ord. *The Advanced Theory of Statistics.* Charles Griffin & Company Limited, 1983.

[Krogh *et al.*, 1994] A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, 1994.

[Lawrence and Reilly, 1990] Charles E. Lawrence and Andrew A. Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *PROTEINS: Structure Function and Genetics*, 7:41–51, 1990.

[Lawrence *et al.*, 1993] Charles E. Lawrence, Stephen F. Altschul, Mark S. Boguski, Jun S. Liu, Andrew F. Neuwald, and John C. Wootton. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, 1993.

[Li *et al.*, 1985] W. H. Li, C. I. Wu, and C. C. Luo. A new method for estimating synonymous and nonsynonymous rates of nucleotide substitution considering the relative likelihood of nucleotide and and codon changes. *Molecular Biology and Evolution*, 22:140–174, 1985.

[Moore *et al.*, 1990] J. Moore, D. Benton, and C. Burks. The GenBank nucleic acid databank. *Focus*, 11(4):69–72, 1990.

[Needleman and Wunsch, 1970] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[Persson *et al.*, 1991] B. Persson, M. Krook, and H. Jornvall. Characteristics of short-chain alcohol dehydrogenases and related enzymes. *European Journal of Biochemistry*, 200:537–543, 1991.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Redner and Walker, 1984] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

[Santner and Duffy, 1989] T. J. Santner and D. E. Duffy. *The Statistical Analysis of Discrete Data*. Springer Verlag, 1989.

[Saqi and Sternberg, 1994] M. A. Saqi and M. J. Sternberg. Identification of sequence motifs from a set of proteins with related function. *Protein Engineering*, 7(2):165–171, 1994.

[Schiffer *et al.*, 1990] C. A. Schiffer, J. W Caldwell, P. A. Kollman, and R. M. Stroud. Prediction of homologous protein structures based on conformational searches and energetics. *PROTEINS: Structure Function and Genetics*, 8(1):30–43, 1990.

[Schneider *et al.*, 1986] Thomas D. Schneider, Gary D. Stormo, Larry Gold, and Andrzej Ehrenfeucht. Information content of binding sites on nucleotide sequences. *Journal of Molecular Biology*, 188:415–431, 1986.

[Seber, 1984] G. A. F. Seber. *Multivariate observations*. John Wiley & Sons, Inc., 1984.

[Smith and Waterman, 1981] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[Smith *et al.*, 1990] H. O. Smith, T. M. Annau, and S. Chandrasegaran. Finding sequence motifs in groups of functionally related proteins. *Proc. Natl. Acad. Sci. USA*, 87:826–830, 1990.

[Stormo and Hartzell, III, 1989] Gary D. Stormo and George W. Hartzell, III. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA*, 86:1183–1187, 1989.

[Sweet and Eisenberg, 1983] R. M. Sweet and D. Eisenberg. Correlation of sequence hydrophobicities measures similarity in three-dimensional protein structures. *Journal of Molecular Biology*, 171:479–488, 1983.

[Swets, 1988] John A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 270:1285–1293, June 1988.

[Tatusov *et al.*, 1994] R. L. Tatusov, S. F. Altschul, and E. V. Koonin. Detection of conserved segments in proteins: iterative scanning of sequence databases with alignment blocks. *Proc. Natl. Acad. Sci. USA*, 91(25):12091–12095, 1994.

[Towell *et al.*, 1990] G. G. Towell, J. W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the National Conference on Artificial Intelligence*, pages 861–866, 1990.

[Ukkonen and Wood, 1993] E. Ukkonen and D. Wood. Approximate string matching with suffix automata. *Algorithmica*, 10(5):353–364, 1993.

[Wang *et al.*, 1994] J. T. Wang, T. G. Marr, D. Shasha, B. A. Shapiro, and G. W. Chirn. Discovering active motifs in sets of related protein sequences and using them for classification. *Nucleic Acids Research*, 22(14):2769–2775, 1994.

[Weber and Steitz, 1984] I. T. Weber and T. A. Steitz. Model of specific complex between catabolite gene activator protein and β-dna suggested by electrostatic complementarity. *Proc. Natl. Acad. Sci. USA*, 81, 1984.

[Weber, 1990] I. T. Weber. Evaluation of homology modeling of HIV protease. *PROTEINS: Structure Function and Genetics*, 7(2):172–184, 1990.