THE VALUE OF PRIOR KNOWLEDGE IN DISCOVERING MOTIFS WITH MEME

TIMOTHY L. BAILEY TBAILEY@CS.UCSD.EDU
Department of Computer Science and Engineering, University of California, San Diego, La Jolla, California 92093-0114

CHARLES ELKAN ELKAN@CS.UCSD.EDU
Department of Computer Science and Engineering, University of California, San Diego, La Jolla, California 92093-0114

Abstract

MEME is a tool for discovering motifs in sets of protein or DNA sequences. This paper describes several extensions to MEME which increase its ability to find motifs in a totally unsupervised fashion, but which also allow it to benefit when prior knowledge is available. When no background knowledge is asserted, MEME obtains increased robustness from a method for determining motif widths automatically, and from probabilistic models that allow motifs to be absent in some input sequences. On the other hand, MEME can exploit prior knowledge about a motif being present in all input sequences, about the length of a motif and whether it is a palindrome, and (using Dirichlet mixtures) about expected patterns in individual motif positions. Extensive experiments are reported which support the claim that MEME benefits from, but does not require, background knowledge. The experiments use seven previously studied DNA and protein sequence families and 75 of the protein families documented in the Prosite database of sites and patterns, Release 11.1.

Keywords: motif discovery, sequence databases, mixture models, expectation maximization, Dirichlet priors

Acknowledgements: Timothy Bailey is supported by NIH Genome Analysis Pre-Doctoral Training Grant No. HG00005. The authors are grateful to Michael Gribskov for many useful conversations during the course of the work reported here, and to other colleagues for advice and encouragement.

1 Introduction

MEME is an unsupervised learning algorithm for discovering motifs in sets of protein or DNA sequences. This paper describes the third version of MEME. Earlier versions were described previously [Bailey and Elkan, 1994], [Bailey and Elkan, 1995a]. The MEME extensions on which this paper focuses are methods of incorporating background knowledge, or coping with its lack. For incorporating background knowledge, these innovations include automatic detection of inverse-complement palindromes in DNA sequence datasets, and using Dirichlet mixture priors with protein sequence datasets. Dirichlet mixture priors bring information about which amino acids share common properties and thus are likely to be interchangeable in a given position in a protein motif. This paper also describes a new type of sequence model and a new heuristic for automatically determining the width of a motif which remove the need for the user to provide two types of information. The new sequence model type allows each each sequence in the training set to have exactly zero or one occurrences of each motif. This type of model is ideally suited to discovering multiple motifs in the majority of cases encountered in practice. The motif-width heuristic allows MEME to automatically discover several motifs of differing, unknown widths in a single DNA or protein dataset. We also describe an improved method of finding multiple, different motifs in a single dataset.

2 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 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 a model are to the observed data (the sequences), and outputs the parameters motif component of the 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 a series of 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 alleviate 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 several 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 a series of 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.

3 Models

3.1 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. We 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]. This paper describes for the first time a generalization of OOPS, called ZOOPS, which 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, as described by Bailey and Elkan [1994].

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.

3.2 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,

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

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.

4 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 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 $Pr(X|\phi)$.

[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, $\lambda = \gamma/m$.

4.1 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. 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

$$\begin{aligned}
&\log Pr(X, Z|\theta) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} Z_{i,j} \log Pr(X_i|Z_{i,j}=1,\theta) + n \log \frac{1}{m}.
\end{aligned}$$

For a ZOOPS model, the joint log likelihood is

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

For a TCM model, the joint log likelihood is

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

The variable Q_i used in the ZOOPS likelihood equation 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. The conditional sequence probabilities for sequences containing a motif used by OOPS and ZOOPS models are defined as

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

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).

4.2 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 Section???. 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,

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

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

For a TCM model,

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

4.3 The M-step

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

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

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

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

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

5 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. It incorporates information about the motifs already discovered into the current model to avoid rediscovering the same motif. The process of discovering one motif is called a pass of MEME.

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.

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

To help compute the new prior on $Z_{i,j}$ we 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 $V_{i,j} = 0$.

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

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

To compute $V_{i,j}$ we 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 $j = 1, \ldots, m$.

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

$$U_{i,j}^{(p)} = U_{i,j}^{(p-1)}(1 - \max_{k=j-W+1,\ldots,j} Z_{i,k}^{(t)}) \quad (1)$$

where $Z_{i,j}^{(t)}$ 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 (2) 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

$$Pr(V_{i,j} = 1) = \min_{k=j,\ldots,j+W-1} Pr(U_{i,k} = 1). \quad (2)$$

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. An approximate formula for reestimating $Z_{i,j}$ in the E-step of EM which takes motifs found on previous passes into account and thus approximates a multiple-motif model can be shown to be

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

MEME uses $\hat{Z}_{i,j}^{(t)}$ in place of $Z_{i,j}^{(t)}$ in the M-step of EM and in equation (1) above.

5.1 Multiple motif details

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 one 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

$$U_{i,j}^{(p)} = U_{i,j}^{(p-1)} Pr(\overline{C_p}). \tag{3}$$

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 a motif provided by $Pr(\overline{C_p})$ will tend to reduce our belief that $X_{i,j}$ is *not* contained in a 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 theorem, 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 a union 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

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

The value of $Z_{i,k}^{(t)}$ gives an estimate of $Pr(M_k)$, so we can write

$$\begin{aligned} Pr(\overline{C_p}) &\leq \min_{k=j-W+1,j} Pr(\overline{M_k}) \\ &= 1 - \max_{k=j-W+1,j} Pr(M_k) \\ &= 1 - \max_{k=j-W+1,j} Z_{i,k}^{(t)}. \end{aligned} \tag{4}$$

The updating of the values of U 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 (4), 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 (erroneosly) 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.

6 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,

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

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

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

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

$$d_x \quad = \quad \sum_{j=1}^{R} Pr(\beta^{(j)}|\mathbf{c})\beta_x^{(j)}, \;\; x \in \mathcal{L}.$$

for $i = 1$ to A. 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.

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 paper 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.

7 Determining the number of model free 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). 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.[3] Likewise, let ϕ_0 be the maximum likelihood estimate for the parameters

[3]We overlook the possibility that EM converged to a local maximum of the likelihood function. We 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.

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

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

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 $LRT(\phi)\nu$.

8 The MEME algorithm

The complete MEME algorithm is sketched below. The number of passes and maximum and minimum values of motif widths to try are input by the user. 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 $\lambda_{max} = 1/(W+1)$.[4] The dynamic programming implementation of the inner loop, the EM-based heuristic for choosing a good value of $\theta^{(0)}$ as a starting point for EM, and the algorithms for shortening motifs and applying the DNA palindrome constraints and the time complexity of the algorithm are described below.

procedure MEME (X: dataset of sequences)
  for $pass = 1$ **to** $pass_{max}$ **do**
    for $W = W_{min}$ **to** W_{max} **by** $\times\sqrt{2}$ **do**
      for $\lambda^{(0)} = \lambda_{min}$ **to** λ_{max} **by** $\times 2$ **do**
        Choose good $\theta^{(0)}$ given W and $\lambda^{(0)}$.
        Run EM to convergence from chosen
        value of $\phi^{(0)} = (\theta^{(0)}, \lambda^{(0)}, W)$.
        Remove outer columns of motif
        and/or apply palindrome constraints
        to maximize $G(\phi)$.
      end
    end
    Report model which maximizes $G(\phi)$.
    Update prior probabilities $U_{i,j}$ to
    approximate multiple-motif model.
  end
end

9 Avoiding local optima

The model to which EM converges *locally* maximizes the likelihood function. We would like to find the sequence model of the given type and motif width which *globally* maximizes the likelihood. The inner loop of MEME attempts to do this by

[4] 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.

running EM once from a succession of different initial values of the mixing parameter λ. 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, $\log(m(W+1)\sqrt{n})$ starting points are found and EM is run to convergence from each of them. MEME uses a dynamic programming algorithm based on a single EM iteration to simultaneously choose a good initial value for θ for each initial value of λ.[5] EM is run to convergence from each of the (θ, λ) pairs selected by the starting point finding algorithm. Emprical results show that this approach works well at avoiding local optima.

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) pairs. 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) pairs for a given value of θ. 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 $X_{i,j+1}$.

9.1 Mining the dataset for EM starting points

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 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

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

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

$$\theta_1 = [\mathbf{p}_{\mathbf{t}}^{(0)}\mathbf{p}_{\mathbf{g}}^{(0)}\mathbf{p}_{\mathbf{t}}^{(0)}\mathbf{p}_{\mathbf{c}}^{(0)}\mathbf{p}_{\mathbf{a}}^{(0)}\mathbf{p}_{\mathbf{t}}^{(0)}] \ = \ \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 2/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 2/5 & 1/5 & 1/5 \\ 1/5 & 2/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 2/5 & 1/5 & 2/5 & 1/5 & 1/5 & 2/5 \end{bmatrix}.$$

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 paper 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 9.1. Sequence-to-theta mapping tables can easily 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.

9.2 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 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 algorithm is sketched below.

	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 1: 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.

```
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, given top nmλ subsequences in (sorted) 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
```

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 **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 programing. 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

$$Pr(X_{i,j+1}|\theta_1^{(k,l+1,W)}) = \frac{Pr(X_{i,j}|\theta_1^{(k,l,W)})Pr(X_{i,j+W}|\theta_1^{(k,l+W,1)})}{Pr(X_{i,j}|\theta_1^{(k,l,1)})} \tag{5}$$

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 (5). 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 yeilds 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.

Time complexity of TEST**.** The maximum time complexity of computing the probabilities without using dynamic programming would be proportional to

$$T_{bf} = O(n^2 L^3).$$

However, the total time with the dynamic programming approach used by algorithm TEST is proportional to

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

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 $O(n^2 L^2)$ when $W = L/2$. We 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 $O(2nL^2)$.

10 Measuring performance

We measured the performance of the motifs discovered by MEME by using the final sequence model output after each pass of 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 $t = \log((1-\lambda)/\lambda)$. The scoring matrix and 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 were 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.

We 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

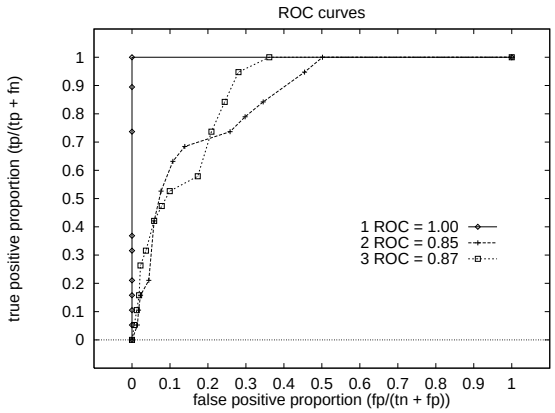

Figure 1: Example ROC curves.

by score, and then numerically integrating tpp over fpp using the trapezoid rule. Sample ROC curves are shown in Figure 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.

11 Results

We studied the performance of MEME on a number of datasets with different characteristics. Seven datasets which were used in the development of MEME are summarized in Table 2. Another 75 datasets each consisting of all the members of a Prosite family are summarized in Table 3.

11.1 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 hth proteins contain DNA-binding

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 2: Overview of the datasets used in developing MEME showing 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 3: 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 sequence(s).

features involved in gene regulation. The farn dataset contains isoprenyl-protein transferases, each with multiple appearances of three motifs.

The *E. coli* DNA datasets, crp, lex and crplex, are described in detail in [Bailey and Elkan, 1995a]. The crp sequences contain binding sites for 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 *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.

11.2 Prosite datasets.

The 75 Prosite families described in general terms in Table 3 correspond approximately to the 10% of fixed-width Prosite patterns with worst combined (summed) recall and precision. Fixed-width patterns such as

$$\texttt{D} - [\texttt{SGN}] - \texttt{D} - \texttt{P} - [\texttt{LIVM}] - \texttt{D} - [\texttt{LIVMC}]$$

are a proper subset of the patterns expressible by MEME motifs, and they form a majority in Prosite. Recall and precision for Prosite patterns and for corresponding MEME motifs were calculated using information in the Prosite database about matches found when searching the large (36000 sequence) SWISS-PROT database of protein sequences [Bairoch, 1994]. The actual Prosite signatures for the 75 families are given in Table 11. Detailed statistics on the families are shown in the Appendix in Tables 9, 10, 12 and 13.

11.3 Performance of different model types

Table 4 shows the ROC motifs found by MEME in the development datasets when MEME was run with the motif width set at $W \leq 100$ for 5 passes. The first lines for each of the three model types shows the performance of MEME without background information—DNA palindromes were not searched for and the one-component Dirichlet prior was used. As expected, the ZOOPS model type outperforms both the OOPS and TCM model types on those datasets which conform to the ZOOPS assumptions, as seen from the higher values of ROC for the ZOOPS model type (line 4) compared with the OOPS model type (line 1) for datasets hrp and crplex in Table 4. Accuracy is not sacrificed when *all* of the sequences contain a motif occurrence: the performances of the OOPS and ZOOPS model types are virtually identical on the first four datasets. The TCM model type outperforms the other two model types on the farn dataset whose sequences contain multiple occurrences of multiple motifs.

model type	*dataset*						
	OOPS-*like*				ZOOPS-*like*		TCM-*like*
	crp	lex	hth	lip	hrp	crplex	farn
OOPS	**0.9798**	0.9998	0.9979	**1.0000**	0.9123	0.9615	0.9446
OOPS_PAL	0.9792	**1.0000**			0.9123	0.9565	
OOPS_DMIX			**1.0000**	**1.0000**			0.9336
ZOOPS	**0.9798**	0.9999	0.9992	**1.0000**	**0.9244**	**0.9881**	0.9112
ZOOPS_PAL	0.9792	**1.0000**			**0.9244**	0.9867	
ZOOPS_DMIX			**1.0000**	**1.0000**			0.9324
TCM	0.9240	0.9895	0.9888	0.9842	0.8772	0.9764	0.9707
TCM_PAL	0.9786	0.9811			0.8772	0.9792	
TCM_DMIX			0.9841	0.9952			**0.9880**
OOPS_GIBBS	0.9709	1.0000	1.0000	0.9999	0.8881	0.9672	0.9291

Table 4: Average ROC of the best motif discovered by MEME for all known motifs contained in dataset. Highest ROC for each dataset is printed in boldface type. Blank fields indicate that the model type is not applicable to the dataset.

model type	*dataset*													
	OOPS-*like*								ZOOPS-*like*				TCM-*like*	
	crp		lex		hth		lip		hrp		crplex		farn	
	R	P	R	P	R	P	R	P	R	P	R	P	R	P
OOPS	79	**90**	84	100	97	97	100	83	42	45	50	30	24	90
OOPS_PAL	75	90	100	100					42	45	54	31		
OOPS_DMIX					100	97	100	92					20	84
ZOOPS	71	89	84	100	97	97	100	83	45	49	75	62	23	79
ZOOPS_PAL	75	90	**100**	**100**					**45**	49	**85**	**79**		
ZOOPS_DMIX					**100**	**97**	**100**	**92**					22	**90**
TCM	21	38	84	42	80	21	60	8	29	59	52	31	53	28
TCM_PAL	**79**	79	84	43					29	**59**	82	54		
TCM_DMIX					80	19	90	12					**79**	44

Table 5: Average percentage precision (P) and recall (R) of the best motif discovered by MEME++ for all known motifs contained in a given dataset. The best model for each dataset is printed in boldface. Blank fields indicate that the model type is not applicable to the dataset.

	dataset										
	OOPS-*like*					ZOOPS-*like*			TCM-*like*		
	crp	lex	hth	lip		hrp	crplex		farn		
known width	20	20	18	16	16	29	20	20	12	12	12
OOPS	**15**	18	15	**5**	**6**	46	29	18	7	9	10
OOPS_PAL	16	**16**				46	24	24			
OOPS_DMIX			**18**	**7**	**6**				8	16	11
ZOOPS	**15**	18	21	**5**	**6**	**46**	**21**	**18**	12	12	9
ZOOPS_PAL	16	**16**				**46**	22	20			
ZOOPS_DMIX			**18**	**7**	**6**				7	8	12
TCM	11	11	10	8	8	29	21	12	10	7	10
TCM_PAL	16	9				29	20	11			
TCM_DMIX			11	7	7				**11**	**7**	**8**

Table 6: Width of the best motif discovered by MEME for all known motifs contained in dataset. Blank fields indicate that the model type is not applicable to the dataset. A width in boldface indicates that this model type has the best average ROC for this dataset.

For comparison, the last line in Table 4 shows the performance of the motifs discovered using the Gibbs sampler [Lawrence *et al.*, 1993]. The conditions of the tests were made as close as possible to those for the MEME tests using the OOPS model type, except that *the Gibbs sampler was told the correct width of the motifs* since it requires the user to specificy the width of all motifs. With each Prosite dataset, the Gibbs sampler was told to search for 5 motifs, each of the width of the Prosite signature for the family, and that each sequence contained one occurrence of each motif. It was run with 100 independent starts (10 times the default) to maximize its chances of finding good motifs. Note that we did *not* tell either the Gibbs sampler or MEME how many occurrences of a particular motif a particular sequence has as was done in [Lawrence *et al.*, 1993].

The ROC of the MEME motifs found using the ZOOPS model type without background information is as good or better than that of the sampler motifs for five of seven datasets. The MEME motifs found using the OOPS model type perform as well or better than those found by the Gibbs sampler with four of the seven datasets. Note once again that the Gibbs sampler was told the correct motif widths to use, whereas MEME was not. MEME using the ZOOPS model type does significantly better than the Gibbs sampler on the two ZOOPS-like datasets.

model type	*ROC*		*recall*		*precision*		*relative width*		*shift*	
OOPS	0.991	(0.025)	0.805	(0.356)	0.751	(0.328)	1.297	(0.753)	-0.978	(5.608)
OOPS_DMIX	0.992	(0.031)	0.815	(0.349)	0.758	(0.325)	1.210	(0.677)	-0.637	(5.337)
ZOOPS	0.992	(0.024)	0.823	(0.335)	0.775	(0.307)	1.307	(0.774)	-0.696	(5.575)
ZOOPS_DMIX	0.993	(0.026)	0.821	(0.340)	0.768	(0.314)	1.220	(0.715)	-0.585	(4.890)

Table 7: Average (standard deviation) performance and width of best motifs found by MEME in the 75 Prosite datasets. All 135 known motifs contained in the datasets are considered.

model type	*ROC*		*recall*		*precision*		*relative width*	
OOPS_DMIX $W \leq 100$	0.971	(0.065)	0.738	(0.288)	0.725	(0.310)	1.170	(0.840)
ZOOPS_DMIX, $W \leq 100$	0.960	(0.090)	0.728	(0.305)	0.699	(0.327)	1.141	(0.815)
OOPS_DMIX, $W = 20$	0.987	(0.029)	0.820	(0.211)	0.840	(0.228)	1.896	(0.785)
OOPS_GIBBS, $W = 20$	0.980	(0.053)	0.781	(0.242)	0.884	(0.169)	1.896	(0.785)

Table 8: Average (standard deviation) two-fold cross-validated performance of MEME and the Gibbs sampler on the 75 Prosite families. The training set consisted of half of the sequences in a given family. The test set consisted of the other half plus half of the 36000 sequences in SWISS-PROT.

The recall and precision values of the motifs found by MEME on the development dataset are shown in Table 5.

11.4 The benefit of background knowledge

The efficacy of using the DNA palindrome bias and the Dirichlet mixture prior can be seen in Table 4. ROC improves in 9 out of 21 cases and stays the same with another 5. The improvements are substantial in the case of the least constrained model type, TCM. For five of seven datasets, using the background information results in the model with the best or equal-best overall ROC.

The LRT-based heuristic does a good job at selecting the "right" width for the motifs in the seven non-Prosite datasets, especially when the DNA palindrome or Dirichlet mixture prior background information is used. The widths of the best motifs found by MEME are shown in Table 6. With background information and the model type appropriate to the dataset, the motif widths chosen by MEME are close to the correct widths with the exception of the lip dataset. That dataset is extremely small and the motifs are faint, which explains why MEME underestimates their widths.

11.5 Performance on the Prosite datasets

MEME does an excellent job of discovering the Prosite motifs in training sets con-

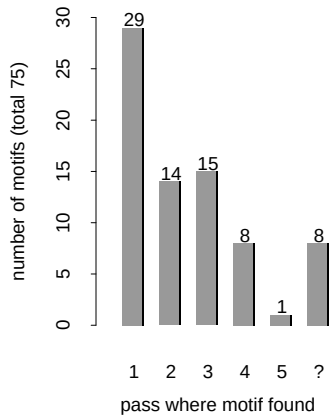

Figure 2: The pass where MEME finds the known Prosite motif is shown. MEME was run for five passes using the OOPS model without any background information. '?' means the known motif(s) were not found by MEME within five passes.

sisting of entire families. This is true with both the OOPS and ZOOPS model types and with or without the background information provided by the Dirichlet mixture prior. For 91% of the 75 Prosite families, one of the motifs found by MEME run for five passes using the OOPS model type and the simple prior corresponds to the known Prosite signature (i.e., identifies the same sites in the dataset). MEME finds multiple known motifs in many of the Prosite families. The criterion we use for saying that a MEME motif identifies a known Prosite pattern is that it have ROC of at least 0.99. MEME usually discovers the known motifs on early passes, as shown in Figure 1.

Of the 75 Prosite families we studied, 45 significantly overlap other families. We define significant overlap to mean two families share five or more sequences in common. If we include the motifs contained in these overlapping families, there are 135 known motifs present in the 75 Prosite family datasets. The overlapping Prosite families we studied are described in more detail in Table 14. When run for 5 passes using the OOPS model type with the simple Dirichlet prior, MEME discovers 112 of these known motifs. The ZOOPS model type does better, discovering 117 of the 135 motifs. With the Dirichlet mixture prior, MEME does even better, discovering 119 out of 135 known motifs using either the OOPS or ZOOPS model types.

Small improvements are seen in the performance of MEME motifs discovered in

the Prosite datasets when the Dirichlet mixture prior is used. This is especially true for the datasets containing few (under 20) sequences. For the 36 Prosite datasets we used which meet this criterion and would thus be most likely to benefit from the background information contained in the Dirichlet mixture prior, the improvement in ROC is statistically significant at the 5% level for the OOPS model type according to a paired t-test. The motifs discovered using the ZOOPS model type are slightly superior to those found with the OOPS model type. Table 7 shows the average performance results on the Prosite datasets when MEME is run for five passes with various model types, with or without Dirichlet priors, 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, as described above. The difference in ROC between the OOPS and ZOOPS model types when the simple Dirichlet prior is used is significant at the 5% level. When the Dirichlet mixture prior is used, the difference in ROC between the two model types is not statistically significant. For both model types, whether or not the Dirichlet mixture prior is used does not make a statistically significant difference in the ROC of the discovered motifs.

The MEME motifs are extremely similar to the Prosite signatures. In general, they identify almost exactly the same positions in the sequences in the families. This fact can be seen in Table 7 from the high ROC, relative width close to 1, and small shift of the MEME motifs.

11.6 Generalization

Cross-validation experiments show that the motifs discovered by MEME on the Prosite datasets can be expected to correctly identify new members of the protein families. Table 8 shows the results of 2-fold cross-validation experiments on the 75 Prosite families using MEME and the Gibbs sampler. The first two lines of the table show the results when MEME is forced to choose the motif width. The performance of the OOPS model type is slightly better than that of the ZOOPS model type (ROC better at 5% significance level). Performance is better if MEME is given background information in the form of being told a good width ($W = 20$), as seen in the third line in Table 8.[6] Then the generalization performance (cross-validated ROC) of the MEME motifs is better than that of sampler motifs at the 5% significance level. In these experiments, both MEME and the Gibbs sampler were allowed to generate only one motif per training set. The Gibbs sampler was instructed to use motif width $W = 20$ and 250 (25 times the default) independent starts to ensure

[6] As can be seen in Table 3, the average width of the known motifs is about 12 with a standard deviation of about 5, so a motif width of 20 is a good compromise between the need to capture all the information in the motif and avoiding including too many uninformative columns.

that the two algorithms got approximately the same number of CPU cycles. The performance figures in Table 8 are based on the number of hits scored on sequences in SWISS-PROT known to be in the family, and do *not* require the hit to be at any particular position within the sequence. We used a threshold of 18 bits for determining if scores were hits.[7]

A direct comparison of the predicted generalization performance of motifs discovered by learning algorithms such as MEME and the Gibbs sampler with that of the Prosite signatures is not possible. The Prosite signatures were created by hand and cannot easily be cross-validated, so their generalization performance is not known. However, the average performance of the Prosite signatures *on their own training sets*, ROC $= 0.99(0.02)$, is the same as the *cross-validated performance* of the MEME OOPS-model motifs found when the algorithm is given a hint about the width of the motifs. This is impressive since the MEME motifs were learned from only half of the members of the families so the cross-validated ROC is likely to be an underestimate of the actual ROC of the motifs. The non-cross-validated estimate of the Prosite signature performance is likely to overestimate their actual performance on new sequences.

[7]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 $347 \times 36000 \approx 10^7$ 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.

References

[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.

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

[Bailey and Elkan, 1995a] Timothy L. Bailey and Charles Elkan. Unsupervised learning of multiple motifs in biopolymers using EM. *Machine Learning*, 1995. In press.

[Bailey and Elkan, 1995b] Timothy L. Bailey and Charles Elkan. The value of prior knowledge in discovering motifs with MEME. Technical Report CS95-413, Department of Computer Science, University of California, San Diego, February 1995.

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

[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.

[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.

[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.

[Harley and Reynolds, 1987] C. B. Harley and R. P. Reynolds. Analysis of *E. coli* promoter sequences. *Nucleic Acids Research*, 15:2343–2361, 1987.

[Kendall *et al.*, 1983] Sir Maurice Kendall, Alan Stuart, and J. Keith Ord. *The Advanced Theory of Statistics*. Charles Griffin & Company Limited, 1983.

Prosite accession number	*width of signature*	*number of sites*	*number of sequences*	*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 9: Characteristics of the individual Prosite datasets (continued in next table). Prosite families are identified by their accession number in the Prosite databse. The width of the Prosite signature, the number of known sites, the size of the family (number of sequences), length of shortest and longest sequence, average length of sequences and total size of sequences are shown. The sequences in the family include members missed by the Prosite signature, but the known sites are just those sites identified by the Prosite signature. Since sequences may have multiple sites, the number of sites sometimes exceeds the number of sequences.

Prosite accession number	*width of signature*	*number of sites*	*number of sequences*	*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 10: Characteristics of the individual Prosite datasets (continued from previous table). The sample mean and standard deviations of each of the quantities are shown at the bottom.

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 11: Prosite signatures of the 75 Prosite families.

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 12: Performance of the individual Prosite signatures.

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 13: Performance of the individual Prosite signatures (continued).

family X	*family* Y	*overlap* $\|I\|$	$\frac{\|I\|}{\|X\|}$	$\frac{\|I\|}{\|Y\|}$	*family* Y	*overlap* $\|I\|$	$\frac{\|I\|}{\|X\|}$	$\frac{\|I\|}{\|Y\|}$	*family* Y	*overlap* $\|I\|$	$\frac{\|I\|}{\|X\|}$	$\frac{\|I\|}{\|Y\|}$
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 14: Prosite families with at least 5 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.

accession	*site-level*					*sequence-level*				
number	*pass*	*W*	*ROC*	*recall*	*precision*	*pass*	*W*	*ROC*	*recall*	*precision*
PS00030	1	8	0.999900	0.990	0.890	1	8	0.999880	1.000	0.204
PS00037	3	20	0.999800	0.714	1.000	4	21	1.000000	1.000	0.900
PS00038	1	12	0.999900	1.000	0.922	1	12	0.999950	1.000	0.511
PS00043	1	21	1.000000	1.000	1.000	1	21	1.000000	1.000	0.357
PS00060	2	9	1.000000	1.000	0.714	1	21	1.000000	1.000	0.636
PS00061	1	16	0.999900	1.000	0.964	1	16	0.999980	1.000	0.804
PS00070	4	13	1.000000	1.000	0.941	4	13	1.000000	1.000	0.523
PS00075	3	18	1.000000	1.000	1.000	3	18	1.000000	1.000	0.971
PS00077	3	14	1.000000	1.000	1.000	2	11	0.999950	1.000	0.855
PS00079	1	10	0.997000	0.556	0.769	1	10	1.000000	1.000	0.140
PS00092	1	9	1.000000	1.000	0.946	1	9	0.999960	1.000	0.174
PS00095	4	20	0.961500	0.000	0.000	3	14	0.999940	1.000	0.579
PS00099	4	37	0.951700	0.000	0.000	1	21	1.000000	1.000	0.824
PS00118	1	11	1.000000	1.000	0.982	1	11	0.999960	1.000	0.965
PS00120	1	10	1.000000	1.000	0.861	1	10	0.999950	1.000	0.201
PS00133	4	15	1.000000	1.000	1.000	1	10	1.000000	1.000	0.188
PS00141	1	10	0.999600	0.563	0.980	2	24	0.999990	0.940	0.959
PS00144	3	8	1.000000	1.000	1.000	1	18	1.000000	1.000	0.571
PS00158	5	11	0.977200	0.000	0.000	1	28	1.000000	1.000	1.000
PS00180	2	9	1.000000	0.981	0.944	1	10	0.999960	0.982	0.574
PS00185	3	13	1.000000	1.000	0.900	1	25	1.000000	1.000	0.909
PS00188	1	20	1.000000	1.000	1.000	1	20	0.999970	1.000	0.750
PS00190	1	5	1.000000	1.000	0.971	1	5	0.999260	1.000	0.088
PS00194	1	5	1.000000	1.000	1.000	1	5	0.999860	1.000	0.072
PS00198	1	12	0.999900	0.993	0.850	1	12	0.999980	1.000	0.852
PS00209	4	23	1.000000	1.000	0.786	1	28	1.000000	1.000	1.000
PS00211	2	15	1.000000	0.992	0.858	2	15	0.999730	1.000	0.788
PS00215	1	10	0.999700	0.670	0.894	4	29	0.999980	1.000	0.929
PS00217	2	15	1.000000	1.000	0.875	2	15	1.000000	1.000	0.730
PS00225	2	13	0.999400	0.503	0.965	2	13	0.999990	1.000	0.627
PS00281	2	10	1.000000	1.000	1.000	4	10	0.999990	1.000	0.095
PS00283	1	15	0.999900	1.000	0.933	1	15	0.999990	1.000	0.423
PS00287	1	8	1.000000	1.000	0.884	2	11	0.999990	1.000	0.235
PS00301	3	8	1.000000	0.990	0.945	4	16	0.999930	1.000	0.925
PS00338	1	19	1.000000	1.000	0.976	2	16	0.999870	0.988	0.876
PS00339	1	15	1.000000	1.000	0.816	2	20	0.999980	1.000	0.422
PS00340	1	8	0.999900	1.000	0.548	1	8	0.999950	1.000	0.168
PS00343	1	13	0.999800	0.750	0.667	3	20	0.999980	1.000	0.439
PS00372	1	14	0.999800	0.857	0.750	2	22	1.000000	1.000	0.368
PS00399	2	8	1.000000	1.000	1.000	1	9	1.000000	1.000	0.004
PS00401	3	9	1.000000	1.000	1.000	1	17	1.000000	1.000	0.455
PS00402	1	20	0.999900	1.000	0.821	1	20	0.999980	1.000	0.619
PS00422	3	26	0.979800	0.727	0.667	2	22	1.000000	1.000	0.923
PS00435	3	12	1.000000	1.000	0.905	4	15	0.999990	1.000	0.291
PS00436	1	8	1.000000	1.000	0.775	5	15	0.999990	1.000	0.303
PS00490	2	25	1.000000	1.000	0.700	1	25	1.000000	1.000	0.900
PS00548	2	22	0.988700	0.867	0.722	1	48	1.000000	1.000	1.000
PS00589	2	24	1.000000	1.000	1.000	1	41	1.000000	1.000	1.000
PS00599	1	14	1.000000	1.000	0.952	1	14	1.000000	1.000	0.429
PS00606	4	36	1.000000	1.000	1.000	1	20	1.000000	1.000	1.000
PS00624	4	14	0.977200	0.000	0.000	1	26	1.000000	1.000	1.000
PS00626	4	28	0.991300	0.409	0.692	1	11	1.000000	1.000	0.051
PS00637	3	8	0.998700	0.875	0.304	5	21	1.000000	1.000	0.173
PS00639	3	11	1.000000	1.000	0.883	2	8	0.999860	0.984	0.436
PS00640	2	8	1.000000	1.000	0.852	2	8	0.999860	0.984	0.436
PS00643	3	11	1.000000	1.000	0.800	1	23	1.000000	1.000	0.714
PS00656	2	7	0.971400	0.000	0.000	1	10	1.000000	1.000	0.017
PS00659	1	10	1.000000	0.973	0.900	3	14	0.999980	1.000	0.377
PS00675	4	15	1.000000	1.000	0.833	5	26	1.000000	1.000	0.973
PS00676	5	26	0.999900	1.000	0.806	5	26	1.000000	1.000	0.973
PS00678	3	21	0.997700	0.519	0.645	1	29	0.999990	1.000	0.703
PS00687	3	20	1.000000	1.000	0.909	3	20	1.000000	1.000	0.917
PS00697	1	14	1.000000	1.000	1.000	5	14	0.999990	1.000	0.097
PS00700	2	16	1.000000	1.000	0.923	2	16	1.000000	1.000	0.153
PS00716	4	28	1.000000	1.000	0.857	2	17	0.999840	1.000	0.783
PS00741	1	15	1.000000	1.000	0.714	1	15	1.000000	1.000	0.122
PS00760	3	21	0.999800	1.000	0.625	1	8	1.000000	1.000	0.007
PS00761	2	13	1.000000	1.000	1.000	1	8	1.000000	1.000	0.007
PS00831	1	8	1.000000	1.000	0.833	2	9	1.000000	1.000	0.006
PS00850	4	7	0.983500	0.000	0.000	2	10	1.000000	1.000	0.007
PS00867	2	25	1.000000	0.893	0.962	1	21	1.000000	1.000	0.909
PS00869	3	12	0.994100	0.000	0.000	1	56	1.000000	1.000	1.000
PS00881	1	5	1.000000	1.000	0.750	4	5	0.999870	1.000	0.004
PS00904	2	8	0.999800	0.650	1.000	1	8	1.000000	1.000	0.010
PS00933	4	44	1.000000	1.000	1.000	1	13	1.000000	1.000	0.153

Table 15: $W \leq 100$, OOPS model, simple Dirichlet prior, training set is entire family. The table shows detailed performance results for the 75 Prosite families. Shown is the Prosite accession number of the family, the pass of MEME++ when the best motif was found, the width chosen by MEME++ for the motif, and the ROC, recall and precision of the best motif.

accession	*site-level*					*sequence-level*				
number	*pass*	*W*	*ROC*	*recall*	*precision*	*pass*	*W*	*ROC*	*recall*	*precision*
PS00030	1	8	0.999900	0.990	0.815	1	8	0.999890	1.000	0.165
PS00037	3	17	1.000000	0.714	1.000	4	21	1.000000	1.000	0.857
PS00038	1	12	0.999900	1.000	0.922	1	12	0.999950	1.000	0.409
PS00043	1	21	1.000000	1.000	1.000	1	21	1.000000	1.000	0.179
PS00060	1	30	1.000000	1.000	0.714	1	30	0.999990	1.000	0.583
PS00061	1	16	0.999900	1.000	0.964	1	16	0.999980	1.000	0.845
PS00070	4	28	1.000000	1.000	0.941	4	28	0.999980	1.000	0.919
PS00075	3	10	1.000000	1.000	1.000	3	10	1.000000	1.000	0.189
PS00077	3	14	1.000000	1.000	1.000	4	22	0.999940	1.000	0.883
PS00079	1	10	0.999100	0.556	0.769	4	15	1.000000	1.000	0.207
PS00092	1	7	1.000000	1.000	0.946	1	7	0.999930	1.000	0.088
PS00095	2	20	0.910200	0.000	0.000	3	14	0.999940	1.000	0.440
PS00099	3	28	0.943000	0.000	0.000	4	37	1.000000	1.000	0.519
PS00118	1	11	1.000000	1.000	0.982	1	11	0.999960	1.000	0.973
PS00120	1	10	1.000000	1.000	0.861	1	10	0.999910	1.000	0.137
PS00133	3	13	1.000000	1.000	1.000	1	10	1.000000	1.000	0.170
PS00141	1	10	0.999500	0.563	0.980	1	10	0.999980	1.000	0.926
PS00144	3	7	1.000000	1.000	1.000	3	7	0.999990	1.000	0.009
PS00158	3	11	0.993400	0.000	0.000	1	28	1.000000	1.000	1.000
PS00180	2	6	1.000000	1.000	0.945	1	10	0.999940	0.982	0.783
PS00185	1	13	1.000000	1.000	0.900	1	13	1.000000	1.000	0.323
PS00188	1	13	1.000000	1.000	1.000	1	13	0.999970	1.000	0.625
PS00190	1	5	1.000000	1.000	0.960	1	5	0.999240	1.000	0.166
PS00194	1	5	1.000000	1.000	1.000	1	5	0.999840	1.000	0.062
PS00198	1	12	0.999900	1.000	0.851	1	12	0.999970	1.000	0.784
PS00209	4	21	1.000000	1.000	0.846	1	26	1.000000	1.000	1.000
PS00211	2	15	1.000000	0.992	0.829	2	15	0.999720	1.000	0.788
PS00215	1	10	0.999700	0.716	0.900	4	29	0.999980	1.000	0.929
PS00217	1	14	1.000000	1.000	0.875	1	14	0.999980	1.000	0.442
PS00225	2	13	0.999500	0.582	0.970	2	13	0.999990	1.000	0.635
PS00281	2	11	1.000000	1.000	1.000	2	11	0.999990	1.000	0.133
PS00283	1	15	0.999900	1.000	0.933	1	15	0.999990	1.000	0.261
PS00287	1	8	1.000000	1.000	0.884	1	8	0.999980	1.000	0.074
PS00301	3	8	1.000000	0.990	0.937	4	16	0.999930	1.000	0.925
PS00338	1	19	1.000000	1.000	0.976	1	19	0.999810	0.988	0.904
PS00339	1	15	1.000000	1.000	0.775	1	15	0.999850	1.000	0.113
PS00340	1	9	0.999900	1.000	0.561	1	9	0.999910	1.000	0.098
PS00343	1	19	0.997700	0.750	0.692	1	19	0.999620	1.000	0.103
PS00372	1	11	1.000000	1.000	1.000	2	28	1.000000	1.000	0.333
PS00399	3	7	1.000000	1.000	1.000	2	9	1.000000	1.000	0.006
PS00401	3	7	1.000000	1.000	0.833	1	17	1.000000	1.000	0.072
PS00402	1	20	0.999900	1.000	0.821	1	20	0.999950	1.000	0.267
PS00422	3	22	0.987800	0.727	0.667	2	20	1.000000	1.000	0.800
PS00435	2	18	1.000000	1.000	0.927	2	18	1.000000	1.000	0.759
PS00436	1	8	1.000000	1.000	0.795	2	18	0.999990	1.000	0.625
PS00490	2	21	1.000000	1.000	0.636	1	14	1.000000	1.000	0.098
PS00548	2	22	0.996100	0.867	0.722	1	55	1.000000	1.000	1.000
PS00589	2	16	1.000000	1.000	1.000	2	16	1.000000	1.000	0.137
PS00599	1	14	1.000000	1.000	0.952	1	14	1.000000	1.000	0.447
PS00606	4	36	1.000000	1.000	1.000	1	15	1.000000	1.000	0.850
PS00624	5	20	0.992200	0.000	0.000	2	21	1.000000	1.000	0.900
PS00626	5	8	0.998200	0.500	0.688	1	12	1.000000	1.000	0.500
PS00637	3	8	0.998800	1.000	0.286	3	8	0.999990	1.000	0.018
PS00639	3	10	1.000000	1.000	0.883	2	8	0.999820	0.984	0.401
PS00640	2	8	1.000000	1.000	0.852	2	8	0.999820	0.984	0.401
PS00643	3	11	1.000000	1.000	0.800	2	15	1.000000	1.000	0.357
PS00656	2	7	0.989900	0.000	0.000	2	7	1.000000	1.000	0.003
PS00659	1	9	1.000000	1.000	0.974	4	10	0.999810	1.000	0.073
PS00675	3	15	1.000000	1.000	0.833	1	8	1.000000	1.000	0.360
PS00676	5	12	0.999900	1.000	0.806	1	8	1.000000	1.000	0.360
PS00678	3	19	0.998400	0.805	0.626	1	14	0.999980	1.000	0.342
PS00687	3	20	1.000000	1.000	0.909	3	20	1.000000	1.000	0.846
PS00697	1	14	1.000000	1.000	1.000	1	14	0.999980	1.000	0.143
PS00700	2	24	0.999800	1.000	0.923	2	24	1.000000	1.000	0.619
PS00716	4	28	1.000000	1.000	0.811	4	28	0.999880	1.000	0.750
PS00741	2	11	0.999600	0.800	0.571	2	11	1.000000	1.000	0.052
PS00760	3	11	1.000000	1.000	0.625	1	8	1.000000	1.000	0.007
PS00761	2	14	1.000000	1.000	1.000	1	8	1.000000	1.000	0.007
PS00831	1	20	1.000000	1.000	0.833	1	20	1.000000	1.000	0.286
PS00850	5	7	0.990400	0.000	0.000	2	14	1.000000	1.000	0.054
PS00867	2	25	1.000000	0.893	0.962	4	10	1.000000	1.000	0.299
PS00869	4	12	0.990000	0.000	0.000	1	40	1.000000	1.000	1.000
PS00881	1	5	1.000000	0.667	0.667	1	5	0.999860	1.000	0.001
PS00904	4	7	1.000000	0.550	1.000	1	14	1.000000	1.000	0.235
PS00933	5	14	1.000000	1.000	1.000	1	23	1.000000	1.000	0.647

Table 16: $W \leq 100$, OOPS model, Dirichlet mixture prior, training set entire family. The table shows detailed performance results for the 75 Prosite families. Shown is the Prosite accession number of the family, the pass of MEME++ when the best motif was found, the width chosen by MEME++ for the motif, and the ROC, recall and precision of the best motif.

[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.

[Seber, 1984] G. A. F. Seber. *Multivariate observations*. John Wiley & Sons, Inc., 1984.

[Swets, 1988] John A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 270:1285–1293, June 1988.