On counting position weight matrix matches in a sequence, with application to discriminative motif finding

Article *in* Bioinformatics · August 2006

DOI: 10.1093/bioinformatics/btl227 · Source: PubMed

Saurabh Sinha
University of Illinois, Urbana-Champaign

BIOINFORMATICS
Vol. 22 no. 14 2006, pages e454–e463
doi:10.1093/bioinformatics/btl227

On counting position weight matrix matches in a sequence, with application to discriminative motif finding

Saurabh Sinha
Department of Computer Science, University of Illinois, Urbana-Champaign, 201 N. Goodwin Ave, Urbana, IL 61801, USA

ABSTRACT

Motivation and Results: The position weight matrix (PWM) is a popular method to model transcription factor binding sites. A fundamental problem in cis-regulatory analysis is to "count" the occurrences of a PWM in a DNA sequence. We propose a novel probabilistic score to solve this problem of counting PWM occurrences. The proposed score has two important properties: (1) It gives appropriate weights to both strong and weak occurrences of the PWM, without using thresholds. (2) For any given PWM, this score can be computed while allowing for occurrences of other, *a priori* known PWMs, in a statistically sound framework. Additionally, the score is efficiently differentiable with respect to the PWM parameters, which has important consequences for designing search algorithms.

The second problem we address is to find, *ab initio*, PWMs that have high counts in one set of sequences, and low counts in another. We develop a novel algorithm to solve this "discriminative motif-finding problem", using the proposed score for counting a PWM in the sequences. The algorithm is a local search technique that exploits derivative information on an objective function to enhance speed and performance. It is extensively tested on synthetic data, and shown to perform better than other discriminative as well as non-discriminative PWM finding algorithms. It is then applied to cis-regulatory modules involved in development of the fruitfly embryo, to elicit known and novel motifs. We finally use the algorithm on genes predictive of social behavior in the honey bee, and find interesting motifs.

Availability: The program is available upon request from the author.

Contact: sinhas@cs.uiuc.edu

1 INTRODUCTION

The study of transcriptional regulation is a pervasive topic in bioinformatics, due to growing realization of the role it plays in all cellular processes, and in the evolution of organismal novelty. A crucial step in such studies is to identify transcription factor binding sites that mediate the regulation of genes. This has proved to be a challenging computational task, largely due to the high variability and short length of binding sites of any given transcription factor (Tompa *et al*., 2005). To model this variability, the position weight matrix (PWM) has emerged as a probabilistic construct of popular choice. A PWM specifies the frequency distribution of nucleotides at each position of the binding sites, and is considered to be related to the energy of binding of the transcription factor to the DNA (Stormo and Fields, 1998). The motif finding problem is to find a PWM representing binding sites of an unknown transcription factor, *ab initio* from sequence data.

A particular variant of this problem, the discriminative motifs problem, is *to find PWMs that are present in one set of sequences (the positive set) and absent in another set of sequences (the negative set)*. There is an abundance of data sets that pose this problem. For example, the segmentation pathway in fruitfly comprises genes that are expressed in certain spatial domains in the embryo, and genes that are not expressed in those domains. Such information is easily available (Tomancak *et al*., 2002), and this spatial partitioning of genes is known to be under transcriptional control (Schroeder *et al*., 2004). In general, any gene expression data set may be mined to obtain positive and negative sets of genes (and their promoters/enhancers), thereby presenting a typical instance of the discriminative motifs problem. Compared to the traditional motif-finding approach, where the positive set of sequences is contrasted with a large ''background'' sequence set, discriminative motif-finding presents much cleaner and more informative negative sequence data to contrast with—the promoters that *do not contain* the desired motif. However, in contrast to the vast body of literature on the motif finding problem, the amount of research done on this useful variant of the problem, especially for the PWM model, is very little.

To solve this problem, the first question to address is: How do we ''count the occurrences'' of a PWM in a sequence? No satisfactory answer has emerged for this problem, despite the wide acceptance of PWMs as a motif model. Let us see some of the implications of the question, and pitfalls of some possible answers.

(1) Counting is different from simply asking if a PWM occurs in the sequence (a ''yes''/''no'' question), and it is inadequate to simply report the quality of the best match to the PWM in the sequence. This is especially relevant for cis-regulatory sequences where the number (and strengths) of binding sites determines function (Schroeder *et al*., 2004).

(2) An ''occurrence'' of a PWM has been traditionally defined by the ''sampling probability'', which is the probability of sampling a given k-mer from the probability distribution induced by a k-length PWM. The naïve approach then is to count all k-length substrings that have sampling probability above a certain threshold, as occurrences of the PWM. The problem with this approach is that one does not differentiate

strong (high sampling probability) occurrences from weaker occurrences, as long as they are above the threshold.

(3) In many motif-finding applications today, there is prior knowledge of one or more motifs that are relevant to the data set being analyzed. It is thus extremely useful to be able to count motif occurrences while factoring in the presence of these known motifs. The standard heuristic used for this is to mask out (partly or completely) the strong matches to the known motif(s) in the sequences, as a pre-processing step. This heuristic may be problematic if a match to a PWM either *overlaps with* or *is* a match to another (known) PWM.

We propose a novel probabilistic score, called the ''w-score'', to quantify the total number of occurrences of a PWM in a sequence, while handling strong and weak occurrences appropriately. (This addresses points 1 and 2 above.) Intuitively, the w-score may be understood as the average number of times the PWM is ''planted'' by a probabilistic model generating the sequence, the average being over all possible ways to generate the sequence as a concatenation of PWM and background sites. Since the score is based on a probabilistic model for sequence generation, any known motif(s) may be incorporated into the model as prior knowledge, and the score of a PWM computed in the context of the known PWMs. (This addresses point 3 above.) The known motifs may be those that were computationally discovered in previous executions of the program.

Having addressed the problem of *counting* occurrences, the next question is: How to find PWMs (motifs) with high counts (w-scores) in the positive set of sequences and low counts in the negative set? One has to define a ''discrimination score'', which captures how different a PWM's counts are in the two sets, and devise an algorithm that maximizes such a score over the space of PWMs. We implement two different discrimination scores—one that directly compares the average counts (w-scores) in the two sets of sequences, and one that models the problem as a classification task based on the counts. We propose a novel hill-climbing algorithm that exploits derivative information on the discrimination score to guide the search. Certain algorithmic choices, explained in Section 3.3, make the algorithm less susceptible to local optima as opposed to a conjugate gradients search. (See Section 4.) It is worth noting that the proposed PWM-finding framework can be trivially modified to optimize other objective functions (not necessarily a discrimination score), e.g., correlation of motif counts with gene expression data.

The main novel contributions of this work are:

(1) A probabilistic model-based score to count PWMs while accounting for number and strengths of occurrences and incorporating any known motif(s), within a statistically sound framework.

(2) An algorithm for the discriminative motif-finding problem, that is based on the new PWM-counting score, and which is empirically shown to be resilient to local optima.

2 PREVIOUS WORK

The work of Jensen and Liu, 2004 proposed a Bayesian score to be used by a PWM search algorithm, though not in a discriminative setting. This score *evaluates a specific set of sites* assumed to be occurrences of an unknown PWM, and their algorithm BioOptimizer searches for the highest scoring set of sites. In contrast, the w-score evaluates a PWM by considering *all possible* sets of occurrences, weighing each such set by its likelihood. In other words, the w-score ''sums away'' the hidden variables representing motif occurrence positions. (This is done at the expense of added but manageable time complexity.) Segal *et al*., 2003 counted occurrences (of a *k*-length PWM) by summing the sampling probability (defined above) of all *k*-mers in the sequence, and deployed a conjugate gradients search algorithm to find discriminative motifs. However, their underlying probabilistic model allows exactly one motif occurrence in a sequence, and their motif-counting score is therefore different from the w-score. Moreover, the principled incorporation of known PWMs into the score is an important advantage of our approach over that of (Segal *et al*., 2003), and this is demonstrated experimentally in Section 4. The work of Xing *et al*., 2003 (LOGOS) provides a general framework for sequence analysis with multiple motifs. It takes a Bayesian approach to motif detection, allows for prior distributions on PWMs, and in fact allows for a more general motif model than PWMs. LOGOS uses a Hidden Markov model (HMM) for distribution of motif occurrences in a sequence, which is identical to the model used in defining the w-score. As such, both LOGOS and our approach can learn motifs while allowing for multiple motifs and handling the statistical dependency of overlapping motif occurrences. However, in contrast to the Bayesian approach of LOGOS, we use an expectation (derived from the same model) as the motif score, and this choice is crucial in how the discriminative motif-finding problem is solved. (The LOGOS framework does not allow finding discriminative motifs.) The DME program of Smith *et al*., 2005 finds PWMs that are most overrepresented in one set of sequences relative to another set, and uses an enumerative search of a discrete PWM space with a specific lower bound on information content of the PWM. The enumerative search affords nice guarantees on the optimization procedure, but this program, unlike our approach, does not allow incorporating *a priori* known motifs during the search. (It discovers additional motifs by erasing the predicted occurrences of the currently predicted motifs.) Finally, note that the commonly used ''relative entropy'' score (Lawrence *et al*., 1993; Stormo and Fields, 1998) of a PWM measures the specificity of the PWM itself, and does not ''count'' its occurrences in sequences.

The discriminative motif-finding problem was also addressed earlier in Sinha, 2003, Sumazin *et al*., 2005 and Takusagawa and Gifford, 2004, among others. The motif model assumed was the ''consensus string'' model, for which the motif counting problem is trivially solved. However, this is a less sensitive motif model than the PWM, and arguably less realistic for complex transcriptional systems.

3 METHODS

3.1 The w-score

This is a score to represent the number and strength of occurrences of a PWM in a given sequence. We go through an informal description first, and a formal definition will follow. Consider the sequence illustrated in Figure 1a that has substrings (''sites'') A, B, and C matching one of two motifs M1 and M2 as shown in Fig. 1b—overlapping sites A and B are matches to M1 and M2 respectively; site C matches M1. Fig. 1c shows six different ''configurations'' for the sequence. A configuration labels non-overlapping sites as

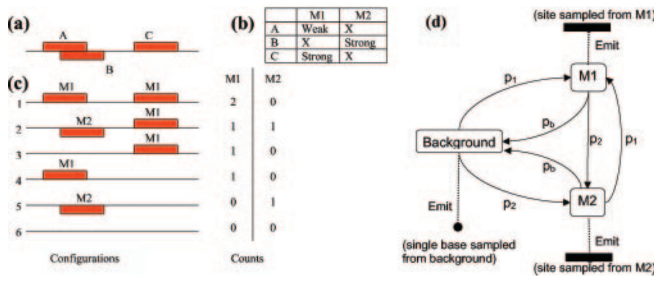


Fig. 1. The w-score. (a) Sites in a sequence (b) Matches between sites and motifs. X means ‘‘no match’’. (c) All possible configurations of the sequence with sites assigned to motifs, and count of each motif in each configuration. (d) Generative model (HMM) used to define w-score.

matches to particular motifs. The probability of a configuration depends on the strengths of these (site, motif) matches. For instance, configuration (1) has lower probability than (2) because the (A, M1) match is weak and the (B, M2) match is strong. We can count the occurrences of a motif, say M1, in each configuration, and compute an average of this count, over all configurations weighted by their probability. Roughly speaking, this average count is the ‘‘w-score’’ of M1. Note that configurations with weak sites will also contribute (albeit less) to the count. In general, every substring (and not just the three shown) is considered as a potential site for a transcription factor (motif), and will be labeled as a match to that motif, in some configurations. Also note that the set of configurations includes all combinations of (site, motif) matches for *both motifs* M1 and M2. Therefore, taking an average of M1’s count over all configurations accounts also for matches to M2 in the sequence, in a natural way.

More formally, the w-score is defined as follows. Suppose we are given a set of PWMs $W = \{w^1, w^2, \ldots w^\kappa\}$, in addition to a ‘‘background’’ PWM w^b of length 1. We assume, as in Sinha *et al.*, 2003, a stochastic process that generates the sequence from left to right by successively ‘‘planting’’ occurrences of PWMs. (Fig. 1d.) At any position, the process chooses to plant a PWM w^i ($i \in \{1, 2, \ldots \kappa, b\}$) with probability p_i. A substring is sampled from the probability distribution defined by w^i (see Appendix), and appended to the sequence generated so far. The process stops when the total length of generated sequence reaches a fixed value L. The p_i, called ‘‘transition probabilities’’ are parameters of the model, with $\sum_{i=1}^{\kappa} p_i + p_b = 1$. The sequence of PWMs chosen in successive steps of the process is called a ‘‘parse’’ of the sequence, which is exactly the same as a ‘‘configuration’’ in the informal description above. Note that the motif location model described here is a zeroth order HMM. (Fig. 1d.)

The model parameters $\Phi = \{p_i\}$ and the PWMs $W \cup \{w^b\}$ associate a well-defined probability $Pr(S, T \mid \Phi)$ with each parse T of a sequence S of length L. This allows us to compute, via Bayes rule, the conditional probability of parse T given the sequence, i.e., $Pr(T \mid S, \Phi)$. We define the w-score of any PWM w^m in the sequence S, with model parameters Φ, as

$$\sigma(w^m, S, \Phi) = \sum_T \chi_m(T) Pr(T \mid S, \Phi) \tag{1}$$

where $\chi_m(T)$ is the number of times w^m occurs in parse T. As we can see, the w-score is the expected number of occurrences of w^m planted while generating S with model parameters Φ. The integral count of w^m in T is weighted by the probability of T given the sequence S. In this sense, the w-score is a natural probabilistic extension of the notion of motif ‘‘count’’. The conditional probability $Pr(T \mid S, \Phi)$ is lower for a parse T with weak (low sampling probability) sites than for a parse with equal number of strong sites, implying a lesser contribution made to the w-score by weak sites. Notice that the w-score is defined in terms of a probability distribution induced jointly by all PWMs $W \cup \{w^b\}$, though this dependence is not made explicit in the notation. The w-score can be computed using the Forward-Backward algorithm (Durbin *et al.*, 1998) in $O(L \times |W| \times l)$ time, where l is the maximum length of a PWM.

3.2 The discrimination-score

We now define a score to discriminate positive and negative sets of sequences based on w-scores of a PWM w^m in the sequences. Suppose we are given two sets of sequences S^+ and S^-, and the w-score σ_s of w^m for each sequence s, i.e., $\sigma_s = \sigma(w^m, s, \Phi)$. The ‘‘t-score’’ is a discrimination score defined as

$$\tau(S^+, S^-, w^m) = \frac{\sum_{s \in S^+} \sigma_s / L_s}{|S^+|} - \frac{\sum_{s \in S^-} \sigma_s / L_s}{|S^-|} \tag{2}$$

($|S|$ for a set S represents its cardinality; L_s for a sequence s represents its length.) The t-score is the difference in mean w-score of the PWM in the two sets of sequences, after normalization of each w-score σ_s by the respective sequence length L_s.

We also defined and implemented an alternative discrimination score, called the ‘‘logistic-score’’, described in the Appendix. It is similar in spirit to the treatment in Segal *et al.*, 2003 and Hong *et al.*, 2005, where a motif score is transformed to a soft class prediction using the logistic function.

3.3 Algorithm

We first provide details of the algorithm, followed by a clearly marked explanation. The algorithm separates the search space from the objective function. Any set of substrings of positive sequences is a candidate motif in the search space, while the objective function is the discrimination score of the unique *PWM constructed from the candidate motif.*

Input : Two sets of sequences S^+ and S^-, integer l_m (desired motif length), background motif w^b and any known motifs.

Parameters : The model parameters $\Phi = \{p_i\}$, integer n (called ‘‘motif cardinality’’).

Search space : Any set of n substrings (of length l_m each) of sequences in S^+ is a candidate motif, also called a ‘‘site-set’’.

Notation : $\delta(w)$ denotes the discrimination-score to be maximized, e.g., $\delta(w) = \tau(S^+, S^-, w)$ if the t-score is being used. $w_{k\alpha}$ is the weight matrix entry for base α in column k. For any site-set C, we use $W(C)$ to denote the PWM constructed from C in the obvious manner.

Desired Output : Site-set C such that $\delta(W(C))$ is maximized over all C.

Algorithm : Initialize the site-set C to one chosen randomly from the search space. In successive iterations, update C (as described next) to improve $\delta(W(C))$ until no improvement is obtained. Repeat the entire process a fixed number of times, starting from new initial site-sets, and report the site-set with the highest score over all such random restarts. Construct a PWM from this site-set, and report the PWM and its occurrences sorted by their quality of match (sampling probability).

Update : The goal of the update step is to go from the current site-set C to any new site-set C' such that $\delta(W(C')) > \delta(W(C))$. This is achieved in two steps, as follows. (Also see ‘‘Explanation’’.)

(1) **DELETE STEP**: For each site $c \in C$, compute the score $\delta(W(C - \{c\}))$. Choose the c that gives the highest such score, and delete it from C to obtain a site-set C^{del}, of cardinality $n - 1$.

(2) **ADD STEP**: For each l_m-length substring s of each sequence in S^+ (on both strands), let C_s^{add} represent the result of adding s to C^{del}. Estimate the score of C_s^{add} as

$$\delta_{est}(W(C_s^{add})) = \delta(W(C^{del})) + \sum_{k,\alpha} \left(\left. \frac{\partial \delta(w)}{\partial w_{k\alpha}} \right|_{w = W(C^{del})} \times [(W(C_s^{add}))_{k\alpha} - (W(C^{del}))_{k\alpha}] \right) \tag{3}$$

Sort all s in descending order of δ_{est} to obtain a list Λ. Traverse this list, and for each encountered s, compute the exact value of $\delta(W(C_s^{add}))$. If this computed score is better than the score $\delta(W(C))$ before the delete-step, update the current site-set to C_s^{add}, and stop the list traversal. In the implementation, the list traversal stops after 500 single-site additions have been examined without improvement in score.

Explanation: The update step deletes one site from the current site-set C, and replaces it with a site (substring of some sequence in S^+) that improves the discrimination-score δ. The deletion step follows steepest ascent hill climbing—try all possible single-site deletions, and choose the one that gives the highest δ score, obtaining C^{del}. The addition step is a simple hill climbing heuristic, which attempts to find *any* single-site addition that causes a net increase in the δ score *over that before the deletion step*. However, unlike the deletion step, all possibilities C_s^{add} are not evaluated exactly before deciding, since exact computation is an expensive operation. (See below.) Instead, all possibilities are quickly *estimated* using derivative information—for each possible single-site addition to C^{del}, the change in $W(C^{del})$, the $4l_m$-dimensional vector (PWM) corresponding to C^{del}, is computed, and the score of the new PWM is estimated by using the partial derivative information in each dimension (k, α). (This estimation ignores quadratic and higher order terms in the Taylor series expansion of δ about $W(C^{del})$.) By then sorting the possibilities C_s^{add} on their estimated score, more promising candidate motifs are brought to the front of the list Λ. This drastically reduces the number of expensive exact evaluations done before finding an improvement, and also leads to more optimal moves. Note that if the estimated new scores for each possible addition were accurate, then the estimation step would automatically produce the best (greedy) choice—the head of the sorted list Λ. However, the estimations are not accurate due to the linear approximation made; hence the list Λ may have to be traversed beyond the head, with an exact evaluation of $\delta(W(C_s^{add}))$ for each traversed site s, before finding a score-improving choice.

Let L_{total} be the total length of all input sequences. Each exact evaluation of δ is an $O(L_{total}l_m)$ operation using the Forward-Backward algorithm for w-score calculations. The calculation of δ_{est} is an inexpensive operation, requiring only $O(L_{total}l_m^2)$ time for all possible single-site addition moves. (See Appendix.) By using these δ_{est} values, we are able to induce an order on the exact δ evaluations that reduces the number of evaluations made before a score-improving choice is found. Our experiments with the algorithm (Section 4) revealed that over 90% of the ADD steps require only one exact evaluation. An implementation that did not sort Λ by δ_{est} typically required 22 times as many evaluations per successful ADD step, and took 3 times as many updates (moves) to find a solution whose score was typically about 0.6 times that reported by the default implementation. The reason for splitting the update into two separate steps (delete and add) is that this causes the change in $W(C)$ to be smaller for each possible move, giving better estimates δ_{est}. A special move, described in the Appendix (‘‘big move’’), is executed when the ADD step fails to find an improvement.

An A^* algorithm that uses backtracking was implemented and found to give better solutions in some runs. Use of this algorithm is available as an option in the code. We also implemented alternative algorithms such as Gibbs sampling and Conjugate Gradients optimization, which fail due to reasons explained in Section 5.

Initialization of Parameters : The motif cardinality n is set to 20. The background PWM w^b is trained from a user-specified background sequence. (Higher order Markov models may also be specified for the background.) An optional user input e represents the *a priori* expected number of sites of the PWM in all of S^+. (If not specified, we set $e = |S^+|$.) We then set the model parameter $p_m = e/(\sum_{s \in S^+} L_s)$. There are two implemented ways to specify the model parameters p_i for the known PWMs. One option is to set all p_i except p_b to be equal to p_m. The other option is based on maximum likelihood, and is described in the Appendix. The assigned values of the model parameters are kept fixed during the algorithm's execution.

3.4 Derivative computation

Here we provide a rough outline of how derivatives of the w-score are computed. (See Appendix for details.) The w-score defined in Equation 1 is the expectation of the random variable $\chi_m(T)$ over the conditional distribution on parses, $Pr(T \mid S, \Phi)$. By linearity of expectation, we may express this expectation as $E(\chi_m) = \sum_{i=1}^{|S|} E(\chi_{mi})$, where χ_{mi} is an indicator (0/1) variable for the presence of w^m at position i in a particular parse. Obviously, $E(\chi_{mi}) = Pr(\chi_{mi} = 1)$, and this probability can be computed by using the Forward-Backward algorithm (Durbin *et al.*, 1998). A derivative of this probability with respect to any PWM entry $w_{k\alpha}^m$ can also be computed by a similar Forward-Backward algorithm, with the same time complexity, and the derivative of the w-score follows. Thus, computation of $\frac{\partial \sigma(w^m, S, \Phi)}{\partial w_{k\alpha}^m}$ has $O(Ll_m)$ time complexity, implying that all the $4 \times l_m$ partial derivatives of the discrimination score δ may be computed in $O(L_{total}l_m^2)$ time. Once all partial derivatives have been computed, the δ_{est} values from Eqn. 3 can be computed for all possible single-site additions to C^{del} in $O(L_{total}l_m)$ time, by pre-computing the change in δ due to addition of any given base α in any column k of the PWM. (See Appendix.)

4 RESULTS

4.1 Synthetic data

We first performed experiments on randomly generated sequence data, with artificially planted motif instances, to get an insight into the algorithm's idealized performance under controlled conditions. In any experiment, 20 ‘‘positive’’ and 20 ‘‘negative’’ sequences, of length 400 bp each, were generated randomly. A target motif (PWM) of length $l = 8$ was randomly chosen with a fixed ‘‘relative entropy’’ (with respect to background) of R bits per column on average. (R is an experiment parameter. Relative entropy is measure of the column's specificity.) l-mers were sampled from the target PWM according to the sampling probability, and planted at random locations, *only in the positive sequences*, so that the sum of the w-scores of the target PWM was $20n$. (n is an experiment parameter, representing the average w-score per sequence.) Finally, the top $20n$ occurrences of the target PWM, based on their match quality, were noted as target motif occurrences. Each tested algorithm was made to report the top $20n$ occurrences of its optimal motif. The performance score of an algorithm is the number of its reported sites that overlap (at least 6 out of 8 bp) target motif occurrences, as a fraction of $20n$. Since the numbers of target and reported motif occurrences are the same, this represents both the sensitivity and the specificity. (In some experiments with the DME program of Smith *et al.*, 2005 below, sensitivity and specificity were different since the program predicted less than $20n$ sites, and a harmonic mean of sensitivity and specificity is reported.)

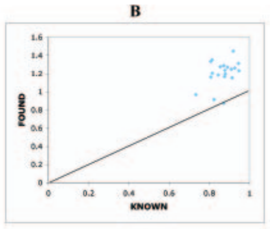
Effect of algorithm settings: In the first set of experiments, we compared two different versions of our algorithm on the same data set. We refer to the algorithm as described in Section 3.3 as **‘‘DIPS’’** (**D**iscriminative **P**WM **S**earch). Five random restarts were used by the algorithms in each run. We set $n = 1$ and $R = 1.5$, and performed 20 replicates of each of the following comparisons. These values (of n and R) are typical of PWMs and their w-scores seen in known enhancers involved in the segmentation of the early fruitfly embryo. (Data not shown.)

- DIPS was compared to a conjugate gradients (CG) search algorithm, using the same objective function, and the same number of random restarts. DIPS significantly outperformed CG on 17 replicates; CG was better on one replicate, and both algorithms failed in two cases. Conjugate gradients search was found to get stuck on low-scoring local optima in 15 of 20 replicates.
- Two versions of DIPS, one using the t-score as the discriminative score, and the other using the logistic-score (defined

A

Experiment	e=10	e=15	e=20	e=25	e=30
1	0.75	0.85	0.75	0.75	0.75
2	0.9	0.85	0.85	0.9	0.7
3	0.85	0.85	0.8	0.8	0.85
4	0.85	0.85	0.85	0.75	0.8
5	0.8	0.8	0.8	0.7	0.7
6	0.25	0.3	0.3	0.3	0.3
7	0.75	0.75	0.75	0.75	0.7
8	0.85	0.75	0.75	0.75	0.7
9	0.7	0.6	0.6	0.7	0.6
10	0.7	0.7	0.7	0.75	0.55
Average	**0.74**	**0.73**	**0.72**	**0.72**	**0.67**

B



C

	DIPS	MEME	LP	DME
R=1.25; n=1.0	0.44	0.24	0.36	0.44
R=1.25; n=1.5	0.66	0.47	0.52	0.57
R=1.25; n=2.0	0.79	0.64	0.75	0.71
R=1.5; n=1.0	0.57	0.45	0.62	0.90
R=1.5; n=1.5	0.85	0.55	0.78	0.88
R=1.5; n=2.0	0.9	0.59	0.82	0.92

D

	DIPS	MEME	LP	DME
R=1.25; n=1.0	0.41	0.03	0.34	0.20
R=1.25; n=1.5	0.72	0.09	0.5	0.16
R=1.25; n=2.0	0.81	0.11	0.65	0.28

Fig. 2. A: Performance of algorithm DIPS as a function of the input parameter *e*, the expected number of sites. Each row represents a data set. B: Comparison of the δ score (re-scaled) of planted (''KNOWN'') and reported (''FOUND'') motifs for each of 20 data sets. The bold line represents FOUND=KNOWN. C: Comparison of performance among DIPS, MEME, LearnPSSM (''LP'') and DME, in the presence of a distractor motif. Each row represents a particular combination of experiment parameters *n* and *R*. D: Comparison of performance between DIPS, MEME, LearnPSSM (''LP'') and DME, where the distractor motif is known to DIPS. (Each figure in C & D is an average over 20 experiments. Performance score of DME is the harmonic mean of its average sensitivity and specificity; see text.)

in Appendix), were compared. The t-score implementation outperformed the logistic-score on 9 of the 20 replicates; the logistic-score performed better on 3, and 8 replicates were inconclusive. This is expected since the motif instances were planted in randomly chosen positive sequences, and were not necessarily uniformly distributed among the sequences. Thus, the planting procedure was more akin to the model assumed by the t-score.

- Two versions of DIPS were compared—one in which a candidate motif was allowed to include overlapping substrings from positive sequences (default), and one in which this was disallowed. Both versions performed equally well.

Insensitivity to model parameter p_m: We ran the algorithm DIPS with different input values of the parameter *e* that represents the expected number of sites, and that is used to determine the parameter p_m. (See ''Initialization of Parameters'', Section 3.3.) While the true value of *e* was 20, we ran DIPS with $e = 10, 15, 20, 25, 30$, separately on the same data. Ten replicates of the comparison were done, and ten random restarts were allowed in each program run. Figure 2A shows the performance score as a function of *e*, for each experimental replicate. For most replicates, the performance is comparable across the different values of *e*. This is an important observation—it shows that the algorithm performance is not very sensitive to the prior expectation of number of sites, i.e., to the model parameter p_m.

Reported score versus planted motif score: The optimal value of the discrimination score, as reported by DIPS, was compared to the discrimination score of the target PWM. In all 20 experimental replicates, the reported score was better than the target PWM's score, typically by a factor of 1.3–1.5. (Fig. 2B.) This shows that the search algorithm actually always finds a motif that is as good or better than the planted motif, in terms of the algorithm's objective function.

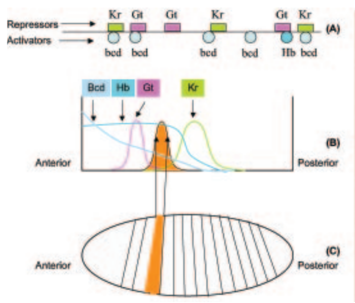
Comparison to alternative motif finding programs: We compared DIPS to a popular non-discriminative motif-finder, MEME (Bailey and Elkan, 1995), that was run on the positive sequences, and made to search for a motif with $20n$ sites. A ''distractor'' motif was also planted, mimicking the planting of the target PWM, except that the distractor was planted in both the positive and negative sets of sequences, in the same amount as the target PWM ($20n$). We expected that MEME, running on the positive sequences alone, will be confounded by the distractor motif, while DIPS will correctly identify the target motif as the true ''discriminative'' motif. We performed 20 experimental replicates for each combination of the experiment parameters $R = 1.25$, 1.5 and $n = 1$, 1.5, 2. The results are shown in Fig. 2C. The average performance of DIPS is significantly higher than that of MEME for all experimental settings.

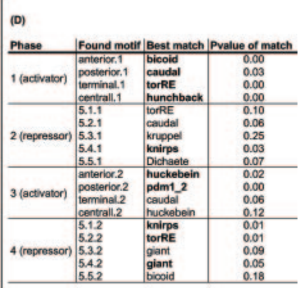
We also compared DIPS to two discriminative PWM searchers, LearnPSSM (Segal *et al*., 2003) and DME (Smith *et al*., 2005). (The LOGOS program (Xing *et al*., 2003), while similar in the underlying model, is not a discriminative motif finder.) LearnPSSM takes into account both positive and negative sequences, and typically performs comparably to DIPS (and better than MEME), except for the weak-motif experiments ($R = 1.25$, $n = 1$, 1.5) where DIPS did significantly better. This is presumably because with weak motifs there is a greater advantage to a model (DIPS) that rewards multiple occurrences in the same sequence over a model that allows exactly one occurrence (LearnPSSM) per sequence. DME (Smith *et al*., 2005) performs better than all other methods when the motif is highly specific ($R = 1.5$), but its performance takes a hit when the motif is weak ($R = 1.25$, $n = 1.5$, 2.0), where DIPS performs significantly better. We noticed that this drop in performance actually reflected a drop in sensitivity, not specificity, and this was because the motif space searched was limited to PWMs with high information content. (See Appendix for details of how DME was run.)

We performed a second kind of experiment, where a distractor motif was planted only in positive sequences, and was made available to DIPS as a known motif. LearnPSSM was not informed of this distractor motif. (We tried masking out occurrences of the distractor motif before input to LearnPSSM, but the program crashed on such input.) Instead, LearnPSSM was made to report two motifs, and the performance score of the better of the two was taken. We also included the program DME in these comparisons. Since DME cannot take a known motif as prior knowledge, we treated it similarly to LearnPSSM, i.e., it was made to report two motifs, and the better performance score taken. (DIPS was made to report only one motif.) DIPS was found to perform significantly better than both LearnPSSM and DME in these experiments. (Fig. 2D). Due to our limited experience with LearnPSSM and DME, it is possible that the optimal choice of parameters was not made for these programs, and a rigorous comparison is beyond the scope of this paper (Tompa *et al*., 2005). For instance, the performance of LearnPSSM improves when using seed words identified by the SeedSearcher program (Yoseph Barash, personal communication); the same seeds may be used for DIPS also.

4.2 Segmentation network in fruitfly embryo

We next applied our algorithm to cis-regulatory modules (CRM's) involved in segmentation of the early fruitfly embryo. Each CRM is known to drive gene expression in a particular domain




(D)

Phase	Found motif	Best match	Pvalue of match
1 (activator)	anterior.1	**bicoid**	0.00
	posterior.1	**caudal**	0.03
	terminal.1	**torRE**	0.00
	centrall.1	**hunchback**	0.00
2 (repressor)	5.1.1	torRE	0.10
	5.2.1	caudal	0.06
	5.3.1	kruppel	0.25
	5.4.1	**knirps**	0.03
	5.5.1	Dichaete	0.07
3 (activator)	anterior.2	**huckebein**	0.02
	posterior.2	**pdm1_2**	0.00
	terminal.2	caudal	0.06
	centrall.2	huckebein	0.12
4 (repressor)	5.1.2	**knirps**	0.01
	5.2.2	**torRE**	0.01
	5.3.2	giant	0.09
	5.4.2	**giant**	0.05
	5.5.2	bicoid	0.18

Fig. 3. A, B, C: Example of CRM action in fruitfly embryo. A: CRM with binding sites for transcription factors bcd, Hb (activators) and Kr, Gt (repressors). B: Concentration profiles of each transcription factor along the A-P axis. The shaded region is where the CRM drives expression. Bcd and Hb activate, Gt represses from anterior and Kr represses from posterior. C: The domain of expression driven by the CRM, in the embryo. D: Motifs discovered in each phase, and for each expression domain; the best matching known motif, and the p-value of this match.

along the embryo's anterior-posterior axis. This is achieved by the combined action of multiple activators and repressors, whose binding sites are harbored by the CRM. (See Fig. 3A-C.) The goal was to discover PWMs for these binding sites using knowledge of the CRMs' expression patterns. We started with a comprehensive set of 51 CRM's, of median length 1383 bp (Schroeder *et al*., 2004). While many of the involved activators/repressors and their expression domains are well-known, we overlooked this information in this illustrative exercise, to simulate the typical application where the target genes/CRM's and their expression patterns are available and the transcription factors are unknown.

We defined ''activator domains'', i.e., our guesses for domains where the activators are present, as broad regions such as ''anterior'' (50−100% egg length along the anterior-posterior (AP) axis, measured from the tail), ''posterior'' (0−50% egg length), ''terminal'' (0−20% e.l.,80−100% e.l.), and ''central'' (30−70% e.l.). This is in tune with the belief that the early stage activators are maternally deposited proteins with broad expression domains. Repressors in this pathway are believed to have ''gapped'' patterns (one or two bands of expression, about 20% e.l. long, at various positions along the AP axis). Therefore, we defined ''repressor domains'', i.e., our guesses for domains where repressors are present, as successive fifths of the axis: ''5.1'' (80−100% egg length), ''5.2'' (60−80%), ''5.3'' (40−60%), ''5.4'' (20−40%), and ''5.5'' (0−20%). For each defined domain, a ''positive'' set of CRM's and a ''negative'' set was obtained, as follows. For an activator domain, CRM's driving expression in the domain were labeled positive and the rest were labeled negative. For a repressor domain, CRM's driving expression in the domain were labeled negative, and CRM's driving expression in the flanking domains were labeled positive. On average, each data set (of positive and negative sequences) included 22 CRM's. Our algorithm was made to report motifs for each such data set, in phases. In the first phase, each activator domain was analyzed, and in the second phase, each repressor domain was analyzed. The third and fourth phases were repeats of the first two, in that order. The top reported motif from each domain in each phase was input as prior knowledge in the following phases. Thus, a total of 4 + 5 + 4 + 5 = 18 motifs were obtained, each of length 9. These are tabulated in Figure 3D. Each reported motif was compared to a small compendium of 14 experimentally determined PWMs (Schroeder *et al*., 2004), using the relative entropy (per column) as the similarity score. A p-value was assigned to each similarity score, using 1000 random permutations of the entries of the reported motif. In Fig. 3D, reported motifs that match some known motif with a p-value of 0.05 or less are shown in bold. All such matches to known motifs are consistent with the literature on this pathway (Schroeder *et al*., 2004), as discussed next.

Phase 1,3 (activators): Motif anterior.1 matches the PWM of Bicoid, the known anterior activator. Similarly, posterior.1 and terminal.1 match known motifs of Caudal and torRE, which are known posterior and terminal activators respectively. Motif central.1 matches that of Hunchback, known to have activating role in the central domain. Motifs anterior.2 and posterior.2, discovered for activators in the third phase, match known motifs of Huckebein and Pdm_12, that are known activators in the anterior and posterior domains respectively. Note that different motifs (Huckebein and Pdm_12, versus Bicoid and Hunchback respectively) are discovered in Phases 3 and 1, on the same data sets, showcasing the iterative incorporation of known motifs into the w-score.

Phase 2,4 (repressors): Motifs 5.4.1 and 5.1.2 were discovered when searching for a repressor in domains ''5.1'' (80−100% egg length) and ''5.4'' (20−40%) respectively. They both match the PWM of Knirps, known to be a repressor expressed at 87−100% e.l and at 25−45% e.l., i.e., the same domains where the motif was found. An important repressor, Giant, is the best match for motif 5.4.2, but visual inspection revealed a few differences between the known and reported motifs. Giant is an appropriate transcription factor for the domain ''5.4'' (20−40% e.l.), being expressed at 15−33% e.l. and known to be a repressor. The known PWM of Giant was constructed from only eight binding sites, and is therefore poorly characterized. This may explain the relatively weak resemblance (p-value 0.05) of the Giant PWM to motif 5.4.2. The motif 5.2.2 that matches torRE, and that corresponds to a repressor in domain 60−80% e.l., is presumably an artifact of the torRE (activator) motif present in the terminal (80–100%) CRM's.

Thus, all 10 discovered motifs with a significant match to a known motif, correspond to transcription factors with consistent functionality. These 10 motifs correspond to 8 distinct known motifs, from a compendium of 14 known motifs. Note that this exercise did not utilize the known expression domains of the transcription factors in finding their motifs. We expect that if the positive and negative sequence sets were created based on a factor's precise expression domain, the motif recovery would improve further. The remaining 8 discovered motifs, with insignificant matches to known motifs, are candidates for being novel motifs. In particular, motifs 5.3.1, 5.5.2, corresponding to an unknown repressor in domains ''5.3'' (40−60% e.l.) and ''5.5'' (0−20% e.l.), and the motif central.2, corresponding to an activator in the central domain, are very dissimilar to the known PWMs and deserve further investigation.

4.3 Social behavior in honey bee

Whitfield *et al*., 2003 identified a small set of genes whose expression pattern in whole brain microarray experiments were most discriminative of foraging behavior versus nursing behavior

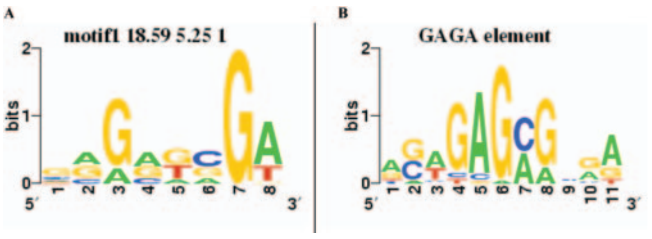


Fig. 4. (A) Motif discovered as discriminating foraging-related versus nursing-related genes in honey bees. (B) the known GAGA motif.

in honey bees. We analyzed the 2Kbp promoters of these genes (21 up-regulated in foragers, 11 up-regulated in nurses). The motif that was most discriminative of these two sets of promoters is shown in Fig 4A. This motif is very similar to the well characterized GAGA element (Fig 4B). The GAGA binding factor is known to regulate gene expression in Drosophila by modulating chromatin structure. Since foraging and nursing behavior in honey bees are controlled by social conditions, our finding represents an important progress in understanding the molecular basis of social behavior.

5 DISCUSSION

The presented algorithm searches the space of site-sets, while the objective function is defined in terms of PWMs. Using PWM w-scores in input sequences is a more sensitive measure of motif occurrence than scores based on the site-set alone (Jensen and Liu, 2004; Lawrence *et al*., 1993), and is crucial for incorporation of negative sequence information in our framework. On the other hand, the search space of site-sets is a restricted subspace of the space of all PWMs (i.e., R^{4l_m}). This imposes a minimum magnitude on any local move made by the algorithm, and is possibly the reason why the algorithm better avoids local optima than a conjugate gradients search in R^{4l_m}. (See Section 4.) A related issue here is the ‘‘motif cardinality’’, i.e., the number of substrings forming a site-set. A large value takes us closer to the R^{4l_m} space, while a very small value makes the search space too small and restrictive. We empirically found a motif cardinality of 20 to work well. Finally, we note that there are other ways to restrict the search space than using the space of site-sets; these were not explored.

The algorithm uses a hill climbing heuristic—it moves to any neighbor that gives an improvement, using derivative information to order the evaluation of possible moves. The exact score computation for any new candidate motif is an expensive operation, linear in the total length of sequences. An alternative strategy such as Gibbs sampling would require evaluating a large number of neighbors (linear in the average length of a sequence) before deciding the move, and is hence impractical in this setting. Another algorithmic choice was to run our hill climbing algorithm followed by conjugate gradients search in the R^{4l} space—we tested this option and found no improvement in performance. The presented algorithm is fairly robust to the choice of the initial random seed. In the experiments of Section 4, the optimal motif was typically obtained in at least two of the five random restarts.

The algorithm is implemented to optimize any differentiable function of the w-scores in individual sequences, and may be trivially modified to use only the positive set of sequences (as in traditional motif finding), or to optimize correlations between sequence and gene expression data. Such modifications and their performance are, however, outside the scope of this paper. The w-score may also be computed in the presence of multiple-species data, using a probabilistic model of binding site evolution (Sinha *et al*., 2004), therefore enabling a phylogenetic version of the algorithm.

The model parameters Φ (including the unknown motif's p_m) define the ‘‘global distribution’’ (Xing *et al*., 2003) $Pr(T)$ on motif occurrences. We then use Bayes rule to compute the conditional distribution $Pr(T \mid S)$ on parses, and hence compute an average count of the motif(s) *given* the sequence data. One option that was not explored is to include p_m as a trainable parameter in the search algorithm, i.e., to find the w^m *and* the p_m that maximize the discrimination score. However, this will mean that the count (w-score) of w^m in a sequence S will depend on sequences other than S.

6 FUTURE WORK AND CONCLUSIONS

The choice of the model parameter p_m is *ad hoc*. Further exploration of this issue, such as specification of a probability distribution over p_m, and integration over that distribution, will be important. Similarly, the choice of the optimal motif (site-set) cardinality is a future research direction. We will also explore a probabilistic version of the discrimination score, such as that of Segal *et al*., 2003, while using the w-score to count PWM occurrences.

We have proposed a statistically sound method to ‘‘count’’ PWM (motif) occurrences in a given DNA sequence. This count is efficiently differentiable with respect to the PWM parameters, enabling search algorithms to use derivative information for a large class of objective functions. We propose a derivative-guided hill climbing algorithm to find a motif that best discriminates two different sets of sequences by its counts in those sequences. The algorithm is tested on synthetic data and is applied to the segmentation pathway in the fruitfly and on behavioral genes in honey bee to elicit several interesting motifs.

ACKNOWLEDGEMENTS

The author is grateful to Eric Siggia and Eran Segal for useful discussions, to Yoseph Barash for making the LearnPSSM code available for experimentation, and to Andrew Smith for sharing the DME code.

REFERENCES

Bailey,T.L. and Elkan,C. (1995) Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1–2): 51–80.

Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis*. Cambridge University Press.

Hong,P., Liu,S., Zhou,Q., Lu,X., Liu,J.S. and Wong,W.H. (2005) A boosting approach for motif modeling using ChIP-chip data. *Bioinformatics*, 21(11).

Jensen,S.T. and Liu,J.S. (2004) BioOptimizer: a Bayesian scoring function approach to motif discovery. *Bioinformatics*, 20(10).

Lawrence,C.E., Altschul,S.F., Boguski,M.S., Liu,J.S., Neuwald,A.F. and Wootton,J.C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262, 208–214.

Schroeder,M.D., Pearce,M., Fak,J., Fan,H., Unnerstall,U., Emberly,E., Rajewsky,N., Siggia,E.D. and Gaul,U. (2004) Transcriptional control in the segmentation gene network of Drosophila. *PLoS Biol*, 2(9).

Segal,E., Yelensky,R. and Koller,D. (2003) Genome-wide discovery of transcriptional modules from DNA sequence and gene expression. *Bioinformatics*, 19(S1).

Sinha,S. (2003) Discriminative Motifs. *J. Comput. Bio.*, 10(3–4).

Sinha,S., Blanchette,M. and Tompa,M. (2004) PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. *BMC Bioinformatics*, **5**(170).

Sinha,S., van Nimwegen,E. and Siggia,E.D. (2003) A probabilistic method to detect regulatory modules. *Bioinformatics*, 19(S1).

Smith,A.D., Sumazin,P., Das,D. and Zhang,M.Q. (2005) Mining ChIP-chip data for transcription factor and cofactor binding sites. *Bioinformatics*, 21(S1): i403–412.

Stormo,G.D. and Fields,D.S. (1998) Specificity, Free Energy and Information Content in Protein-DNA Interactions. *Trends in Biochemical Sciences*, **23**, 109–113.

Sumazin,P., Chen,G., Hata,N., Smith,A.D., Zhang,T. and Zhang,M.Q. (2005) DWE: discriminating word enumerator. *Bioinformatics*, **21**(1).

Takusagawa,K. and Gifford,D. (2004) Negative information for motif discovery. In *Pacific Symposium on Biocomputing*, Hawaii, pp. 360–371.

Tomancakal,P. *et al*. (2002) Systematic determination of patterns of gene expression during Drosophila embryogenesis. *Genome Biol*., **3**(12).

Tompa,M. *et al*. (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, **23**(1).

Whitfield,C.W., Cziko,A.M. and Robinson,G.E. (2003) Gene expression profiles in the brain predict behavior in individual honey bees. *Science*, **302**(5643), 296–9.

Xing,E.P., Wu,W., Jordan,M.I. and Karp,R.M. (2003) LOGOS: a modular Bayesian model for de novo motif detection. In *IEEE Computer Society Bioinformatics Conference*.

7 APPENDIX

7.1 Sampling probability for PWM

Let $w_{k\boldsymbol{\alpha}}$ denote the probability of base $\boldsymbol{\alpha}$ in column k of a PWM w of length l. The weight matrix induces a probability distribution on all strings of length l. The probability of sampling a string s of length l from the PWM w is defined as $Pr(s \mid w) = \prod_{k=1}^{l} w_{ks_k}$, where s_k is the k^{th} base of s.

7.2 Logistic-score

The ''logistic-score'' is an alternative discrimination score implemented, defined as $\lambda(S^+, S^-, w^m) =$

$$-\Big(\sum_{s\in S^+}(1 - logit(\sigma_s/L_s))^2 + \sum_{s\in S^-}(logit(\sigma_s/L_s))^2\Big) \tag{4}$$

where the function $logit(x) = 2/(1 + e^{-\omega x}) - 1$ is a rescaled logistic function with range 0 (at $x = 0$) to 1 (at $x = \infty$). (L_s for a string s represents its length.)

The logistic-score represents the least-squares error of a classifier that uses the $logit$ function as a soft predictor for class membership, with 0 representing S^- and 1 representing S^+. (The negative sign is to minimize the error while maximizing the function.) In the algorithm, during the initialization of parameters, the exponent factor ω in the logistic function is set to $2e/|S^+|$. (See ''Initialization of Parameters'', Section 3.3, for definition of e.)

7.3 The ''big move''

The update (Section 3.3) uses two separate hill climbing steps. As such, it may fail to find an improvement even if there exists a deletion choice that is itself non-optimal, but leads to a score improvement in combination with the appropriate addition choice. The ''big move'' is executed when the above default procedure fails. This move effects each possible deletion, computes derivatives, and estimates the score of each possible addition. It then sorts all such deletion-addition pairs by estimated score, and serially evaluates each pair, stopping when an improvement is found, or when a certain number of evaluations (500, for the current implementation) have been made.

7.4 Incorporating known PWMs into the score

The algorithm assigns the transition probabilities for the known motifs $W_{known} = \{w^1, \ldots w^\kappa\}$ separately for each sequence S, as follows. It first assumes that the only PWMs in the model are w^b and the set W_{known}, and computes the values of the corresponding p_i parameters that maximize the likelihood of the sequence S. This computation, described in Sinha *et al*. 2003, uses an Expectation-Maximization algorithm and is known to have good convergence property. The transition probabilities p_i, for $i = 1 \ldots \kappa$, are then fixed at these trained values. The transition probability p_m for the desired motif w^m is then assigned as described in Section 3.3, and p_b is obtained from the constraint $(\sum_{j=1}^{\kappa} p_j) + p_m + p_b = 1$. Note that this step causes little change in p_b, since p_m is small; hence the values of $p_1, p_2, \ldots p_\kappa$ and p_b are those determined by the maximum likelihood inference.

7.5 Computing derivatives of w-score

The w-score of PWM w^m in sequence S of length L with model parameter p_m is defined in Equation 1. We re-write the definition with a slight change of notation as

$$\sigma_m(S) = \sum_T \chi_m(T) Pr(T \mid S, \boldsymbol{\Theta})$$

where $\chi_m(T)$ is the number of times w^m is planted in parse T, and $\boldsymbol{\Theta}$ represents the model parameters. In the general case, this includes w^m and its transition probability p_m, the background motif w^b (with p_b), and the set of known motifs $W_{known} = \{w^1, w^2, \ldots w^\kappa\}$ with corresponding transition probabilities $p_1, p_2, \ldots p_\kappa$, with the constraint $(\sum_{j=1}^{\kappa} p_j) + p_b + p_m = 1$. Let the indicator (0/1) variable $X_{ml}(T)$ be 1 if motif w^m is planted at position l in parse T. Then $\sigma_m(S)=$

$$\sum_{l=1}^{L} \sum_{T \mid X_{ml}(T)=1} Pr(T \mid S, \boldsymbol{\Theta}) = \sum_l \sum_{T \mid X_{ml}(T)=1} \frac{Pr(S, T \mid \boldsymbol{\Theta})}{Pr(S \mid \boldsymbol{\Theta})} \tag{5}$$

Therefore, we can write the partial derivative of $\sigma_m(S)$ as $\frac{\partial \sigma_m(S)}{\partial w^m_{k\gamma}} =$

$$\frac{\frac{\partial}{\partial w^m_{k\gamma}} \sum_l \sum_{T|X_{ml}(T)=1} Pr(S,T|\boldsymbol{\Theta})}{Pr(S|\boldsymbol{\Theta})} - \frac{\sigma_m(S) \frac{\partial}{\partial w^m_{k\gamma}} Pr(S|\boldsymbol{\Theta})}{Pr(S|\boldsymbol{\Theta})} \tag{6}$$

Let B and C denote the two terms in the sum on the right hand side. Let us define the ''forward'' variable $\alpha(l)$ as the probability of generating the subsequence $S[1 \ldots l]$ by the model, such that some w^i ends at l. Similarly, let the ''backward'' variable $\beta(l)$ be the probability of generating the subsequence $S[l \ldots L]$ by the model, such that some w^i begins at l. Let $\alpha'(l) = \frac{\partial \alpha(l)}{\partial w^m_{k\gamma}}$ and $\beta'(l) = \frac{\partial \beta(l)}{\partial w^m_{k\gamma}}$. By definition, $\alpha(L) = Pr(S \mid \boldsymbol{\Theta})$, hence we have $\frac{\partial}{\partial w^m_{k\gamma}} Pr(S \mid \boldsymbol{\Theta}) = \alpha'(L)$. This gives us

$$C = \frac{\sigma_m(S)\alpha'(L)}{\alpha(L)} \tag{7}$$

$$B = \frac{1}{\alpha(L)} \sum_l \frac{\partial}{\partial w^m_{k\gamma}} \sum_{T \mid X_{ml}(T)=1} Pr(S, T \mid \boldsymbol{\Theta}) \tag{8}$$

Now, the term $\sum_{T \mid X_{ml}(T)=1} Pr(S, T \mid \boldsymbol{\Theta})$ may be expressed using the forward and backward variables as being equal to $\alpha(l - 1) p_m Pr(S[l \ldots l + l_m - 1] \mid w^m) \beta(l + l_m)$, where l_m is the length of w^m. Hence we have

$$\begin{aligned} &\tfrac{\partial}{\partial w^m_{k\gamma}} \textstyle\sum_{T \mid X_{ml}(T)=1} Pr(S, T \mid \boldsymbol{\Theta}) \\ &= \alpha'(l\text{-}1) p_m Pr(S[l \ldots l + l_m\text{-}1] \mid w^m) \beta(l + l_m) \\ &+ \alpha(l\text{-}1) p_m [\tfrac{\partial}{\partial w^m_{k\gamma}} Pr(S[l \ldots l + l_m - 1] \mid w^m)] \beta(l + l_m) \\ &+ \alpha(l - 1) p_m Pr(S[l \ldots l + l_m - 1] \mid w^m) \beta'(l + l_m) \end{aligned} \tag{9}$$

where $\frac{\partial}{\partial w^m_{k\gamma}} Pr(S[l \ldots l + l_m - 1] \mid w^m) = \prod_{j=1}^{l_m} (w^m_{j\gamma})^{(1\text{-}\delta_{jk})}$, and δ_{jk} is the Kronecker delta function. Finally, we consider the derivatives of the forward and backward variables. Since $\alpha(l) = \sum_i \alpha(l - l_i)p_i Pr(S[l - l_i + 1 \ldots l] \mid w^i)$, where l_i is the length of w^i, we can write $\alpha^{'}(l) =$

$$[\sum_i \alpha^{'}(l - l_i)p_i Pr(S[l\text{-}l_i + 1 \ldots l] \mid w^i)]$$

$$+ \alpha(l - l_m)p_m \frac{\partial Pr(S[l - l_m + 1 \ldots l] \mid w^m)}{\partial w^m_{k\gamma}} \qquad (10)$$

and $\beta'(l)$ is obtained by a similar recursion. Combining Equations 7, 8, 9, 10, and replacing into Equation 6, we obtain the partial derivative of $\sigma_m(S)$ with respect to $w^m_{k\gamma}$.

In the implementation, underflows are handled by using scaling constants, an issue not considered in the above description.

7.6 Time complexity

α' and β' can be computed using a forward and backward algorithm in time $O(L\sum_i l_i)$. Computation of B has the same time complexity, and hence computing each $\partial\sigma_m(S)/(\partial w^m_{k\gamma})$ takes $O(L\sum_i l_i)$ time, giving an $O(Ll_m\sum_i l_i)$ time complexity for the entire derivative computation of the w-score. Since this computation has to be done for each sequence in $S^+ \cup S^-$, the total time complexity is $O(L_{total} l_m \sum_i l_i)$. If the only PWMs are w^m and w^b, this reduces to $O(L_{total} l_m^2)$.

Knowing all $4l_m$ partial derivatives, we can compute the δ_{est} for all single-site additions s to the current site-set C^{del} as follows. Note that any site addition $s = s_1 s_2 \ldots s_{l_m}$ changes the PWM $W(C^{del})$ in a particular way: in column k, the frequency (integral count) of base s_k goes up by 1, and the frequency of the other bases in that column remains same. Thus, there are only four possibilities for the (vector) change in the k^{th} column, regardless of s. Hence there are only four possibilities for the term $\sum_\alpha (\frac{\partial\delta(w)}{\partial w_{k\alpha}} \mid_{w=W(C^{del})} \times [(W(C_s^{add}))_{k\alpha} - (W(C^{del}))_{k\alpha}])$, regardless of s. We can pre-compute each of these four possibilities, for each k. Then, for any single-site addition s, we only have to look up l_m of these pre-computed values (one for each k) and sum them to obtain $\delta_{est}(W(C_s^{add}))$-$\delta(W(C^{del}))$, in $O(l_m)$ time. Thus, computing the δ_{est} for all possible single-site additions takes $O(L^+ l_m)$ time, where L^+ is the total length of all positive sequences, and hence the maximum number of single-site additions s.

7.7 Experiments on synthetic data

Our program (DIPS) was run with the '-len' option set to the desired motif length, with ''-niter 5'' to try 5 random initial seeds per motif, and ''-nsites'' set to the $20n$, which is the number of sites planted.

LearnPSSM was run with seed length (-l) and PSSM length (-L) both set to the desired motif length, with the '-r' option to search both strands, and with ''-m 1000'' to try 1000 seeds for each motif. The ''Training'' file assigned a weight of 0.99 to each sequence in the positive set, and a weight of 0.01 to each sequence in the negative set.

DME was run with motif width (-w) set to the desired motif length, and the ''minimum number of bits per column'' (-i) set to 1.5. We also experimented with setting the '-i' option equal to the bits per column of the true (planted) motif, and found the results to be poorer for weak motifs (-i 1.25), and hence report the better results (from using -i 1.5).