

Discriminative motifs *

Saurabh Sinha

Department of Computer Science and Engineering
Box 352350
University of Washington
Seattle, WA 98195, U.S.A.
saurabh@cs.washington.edu

October 15, 2001

Abstract

This paper takes a new view of motif discovery, addressing a common problem in existing motif finders. A motif is treated as a feature of the input promoter regions that leads to a good classifier between these promoters and a set of background promoters. This perspective allows us to adapt existing methods of feature selection, a well studied topic in machine learning, to motif discovery. We develop a general algorithmic framework that can be specialized to work with a wide variety of motif models, including consensus models with degenerate symbols or mismatches, and composite motifs. A key feature of our algorithm is that it measures over-representation while maintaining information about the distribution of motif instances in individual promoters. The assessment of a motif's discriminative power is normalized against chance behaviour by a probabilistic analysis. We apply our framework to two popular motif models, and are able to detect several known binding sites in sets of co-regulated genes in yeast.

*This material is based upon work supported in part by the National Science Foundation under grant DBI-9974498 and in part by Microsoft under a Microsoft Research Fellowship

1 Introduction

Suppose we are given a set of DNA sequences that are hypothesized to contain several instances of a short pattern. How do we find that unknown pattern? This is the generic motif finding problem and it has been studied extensively, in different forms. One of its incarnations is the problem of finding transcription factor binding sites in the *promoter* regions of a set of co-regulated genes. Solving this problem provides a hypothesis about the interaction of genes with specific transcription factors, and such a hypothesis, when verified, leads to a better understanding of the regulation of gene expression. Binding site sequences are often approximately conserved across promoter regions, and the term “motif” means the common pattern in different binding sites of a transcription factor. Some previous approaches to *de novo* motif finding include those by Lawrence *et al.* [8], Roth *et al.* [13], Bailey and Elkan [2], Hertz and Stormo [6], van Helden *et al.* [16], Tompa [15] and Sinha and Tompa [14]. Most approaches evaluate a motif by measuring how the frequency of its occurrences (allowing some variation) in the given promoter regions compares with the frequency in typical promoter regions. Our work belongs to this genre of over-representation based motif finding. The salient features of this work are:

- It takes a new view of motif discovery, treating it as a feature selection problem.
- It describes a general algorithmic framework that can be specialized to work with a large class of motif models.
- It utilizes information about the distribution of motif instances among the given promoters when assessing the motif’s over-representation, rather than looking only at the total count of occurrences.

The latter three of the motif finders mentioned above explicitly count the occurrences of a motif and compare this count against what is expected if the sequences had been generated by some suitable random process. A major strength of this approach is that evaluation of the over-representation of a motif has a sound statistical basis. A shortcoming of the technique is that it does not differentiate between the case when occurrences are distributed across the different promoters and the case when they are more concentrated in one promoter than in others, as long as the total count is the same. Given a set of genes believed to be co-regulated, it is typical to find binding sites in most of the promoters. In such situations, where we are interested only in motifs that are distributed over all or most of the promoters, the count based approach can lead to false positives. This is a common scenario and may arise for instance when there are several copies of a short repeat in one of the promoters.¹ Our work addresses this problem by proposing a new method to evaluate motif over-representation that utilizes information about the distribution of motif counts in individual promoters.

An over-represented motif is, in a sense, one whose occurrences yield high *discrimination* between promoters that are believed to contain the binding sites (called “positive” promoters) and those that are not (called “negative” promoters). This suggests that a motif is a *feature* of the positive sequences that leads to a good classification of positive and negative promoters. We call such motifs “discriminative motifs” or “d-motifs” for brevity. Finding good d-motifs thus reduces to doing feature selection, a well studied topic in machine learning. Given a motif, we count its occurrences in each promoter, and measure the error made by the best linear classifier that classifies

¹For instance, the motif AGTANNNNACAS occurs 10 times in 7 genes that are believed to be co-regulated by Daf-19p, in *C.elegans*. A count based approach considers this as over-represented, even though all its 10 occurrences are in one promoter, and it is a short repeat rather than a binding site.

the positive and negative promoters based on these counts. The best d-motif is the one with the *most surprisingly low* classification error. The surprise is measured based on the *a priori* probability distribution of the count of the motif. Since it is difficult in practice to know negative promoters, we use randomly generated promoters as negatives. This heuristic is shown to work well in practice.

The notion of a d-motif is a characterization of an over-represented motif. It is not restricted to any particular type of motif (e.g., consensus models with mismatches or degenerate symbols, or composite motif models), as long as there is a way to count occurrences of any motif of that type in a given promoter. The algorithm to find d-motifs is developed with minimal assumptions about the motif model, and can be applied to look for different types of motifs. Section 2 describes how a d-motif is evaluated, and Section 3 gives a general algorithm to find the best d-motifs. We demonstrate two applications of d-motif finding, each with a different motif model, in Sections 4 and 5. When run on real sets of co-regulated genes, the first application finds *simple* motifs that are known to be binding sites. (A simple motif models the binding site of a single transcription factor.) In the second application, the same technique is able to find a *composite* motif, which is a pair of simple motifs separated by a variable but restricted distance. (A composite motif corresponds to a pair of transcription factors, and has more variability than simple motifs.) Section 6 reports the results of experiments with synthetic data that evaluate the different aspects of the algorithm.

1.1 Comparison with Previous Work

The work of Hu *et al.* [7] is similar to this work in some of the goals and methods. Their algorithm finds simple motifs that are well conserved, over-represented and also distributed across promoters. This is done by computing a separate score for each of these three motif characteristics, and combining normalized versions of these three scores by taking their harmonic mean. This is a somewhat *ad hoc* method of combining the different axes of information, and we propose a more natural method with statistical justification. Their paper also attempts to classify two sets of promoters using a decision tree classifier that looks at occurrences of composite motifs. There is a feature selection step in this classification procedure that is equivalent to finding discriminative motifs. However, the feature selection is done based on classification score, which is not a powerful criterion for motif finding. As we see later, some motifs are expected to have a better classification score than others just by chance. Our algorithm normalizes raw classification scores by computing their p-values, which are shown in Section 6 to have more discriminative power.

One of the important applications of d-motifs is in finding composite motifs, described in Section 5. Marsan and Sagot [9] present an efficient suffix-tree based algorithm that finds pairs of motifs (with distance constraints) that occur frequently in given promoters. However, raw frequencies should not be the sole criterion for evaluating motifs, since these frequencies have different probability distributions for different motifs. More recently, GuhaThakurta and Stormo [5] have developed an algorithm that evaluates composite motifs by an objective function in which both positive and background sequences are considered, and which captures the joint likelihood of occurrence of two simple motifs. However, the motif model used is that of “position weight matrices” or PWMs, whereas our framework is for a different class of models - those where motifs have integral counts in a sequence. Their motif model allows more variability in binding sites, but the motif evaluation assumes that there is exactly one binding site in each input promoter. The d-motifs framework imposes no such restriction.

The most interesting comparison of our work is with that of count based statistical methods such as those in [16], [15] and [14]. Both approaches count motif occurrences in each promoter, and therefore reward a motif that occurs multiple times in the same promoter. This is an important

property of the algorithms, since binding sites often occur more than once in a promoter. However, there is an important difference between the approaches. If we want to find motifs whose total count in a set of promoters is surprisingly high, the count based approach is better. On the other hand, if we are looking for over-represented motifs that are also well distributed over the promoters, d-motifs is the preferable approach. In fact, when restricted to 0 or 1 occurrence per promoter, the count based approach becomes a special case of the d-motif approach. The latter is compared with one of the count based algorithms in terms of performance on synthetic data with planted motifs, and is found to do significantly better when the motif occurrences are well distributed.

2 Evaluating motifs

This section describes in detail how a given motif s is evaluated for significance. We shall assume that we are given p “positive” promoters, and n “negative” promoters, each of length L , and the goal is to objectively determine how well the occurrences of s discriminate between the positive and negative promoters. While no assumption is made about what s may look like, we do assume that there is an algorithm to count the occurrences of s in any given promoter, and that the count is an integer. We also assume that the *a priori* distribution of the count of s in a randomly generated sequence of length L is known. A parameter to the evaluation is k_{max} , which indicates that k_{max} or more occurrences in a promoter are treated as if there were exactly k_{max} occurrences. While it is desirable to consider motifs with more occurrences in a promoter as more significant, it is reasonable from biological considerations to assume that occurrences beyond a certain large number do not add to the significance of the motif. Moreover, a large motif count in a promoter is sufficiently unlikely to not affect our calculations considerably. Additionally, assuming a small value of k_{max} leads to more efficient computations. Our implementation uses $k_{max} = 4$ for motifs of length 6.

We borrow ideas from the work of Ben-Dor *et al.* [3], where feature selection was used to score genes for relevance to tissue classification. Some of the terminology used here is hence borrowed from that work. The first step to evaluating a motif is computing its “TNoM score”, which is defined next. Each positive promoter is plotted as a ‘+’ on the integer line, at the point x if x is the count of s in that promoter. Similarly each negative promoter is plotted as a ‘-’. A “decision stump” is an integer l in this range, and the classification score of a decision stump is the sum of E_n and E_p , where E_n is the number of ‘-’s plotted at $x \geq l$ and E_p is the number of ‘+’s plotted at $x < l$. It is thus the number of errors made by the linear classifier that calls each $x \geq l$ positive. The TNoM score of s in such a configuration of ‘+’s and ‘-’s is the minimum classification score of any decision stump for that configuration. The significance of s is given by the probability that the TNoM score of s in a configuration derived from random sequences is less than or equal to the TNoM score observed for the configuration derived from the input promoters. This is called the “p-value score” of s , or simply $pval_s$. We need to use p-values because the TNoM score has a different probability distribution for different motifs. Motifs with higher variance are expected to have lower TNoM score.

A prerequisite to computing $pval_s$ is the knowledge of $Pr[\text{count of } s \text{ in a random text of length } L \text{ is } k]$, denoted by $p_{s,k}$, for $k = 0..k_{max}$. The computation of this distribution depends upon the type of motif s , and is seen in Sections 4 and 5 for two particular types. As mentioned earlier, the TNoM score has been previously used by Ben-Dor *et al.* [3] to score genes for relevance in tissue classification, and the authors present recurrence relations to compute its p-value under slightly different assumptions.

We use an alternate set of recurrences to compute $pval_s$, that were preferred over those in [3] for reasons mentioned shortly. The recurrences are described in the Appendix (Section 9.1). For

a TNoM score of e , these recursive computations can be implemented by a dynamic programming algorithm that runs in time $\theta(e^2(n^3 + p^3))$, as compared to the $\theta(en^2p^2)$ time that a dynamic programming implementation of the recurrences adapted from [3] would take. Since it is often the case that the algorithm needs the probabilities only for $e = o(\max(n, p))$, and since in our algorithm p is equal to n , our set of recurrences is more efficient. Another advantage of these recurrences is that they may be extended to work when the configurations are on the 2-dimensional integer grid, rather than the integer line. It is an open problem to compute the extended recurrences efficiently by using suitable approximations. The recurrences in [3] have no natural extension to the 2-dimensional case, which arises when considering some natural models of composite motifs.

3 Algorithm

The algorithm described here is given a set P of p sequences of length L each, and the goal to find the best d-motifs. We assume a well-defined random process R that generates sequences of A,C,G,T in a way that mimics typical background sequences. Other parameters include an integer, *numbags*, which is explained later, and an integer k_{max} described in Section 2. The algorithm is described without assuming any particular motif model, but it is required that some finite set S enumerates all motifs satisfying the model. The algorithm outputs the motifs in S sorted by their significance.

Algorithm DMotifs

1. For $i = 1$ to *numbags* do
 2. Create a set N of p random strings (length L each), using the random process R .
 3. For each motif s in S do
 4. Compute TNoM score of s using P and N as the positive and negative sets of promoters.
 5. Compute $pval_s$ using the algorithm described in the Appendix.
 6. End
 7. Sort the motifs in S in non-decreasing order of $pval_s$ and call this sorted list Λ_i .
 8. End
 9. For all motifs s in S do
 10. Compute the combined rank of s as the sum of its ranks from each Λ_i .
 11. End
 12. Sort all motifs in S by their combined rank in non-decreasing order and report this sorted list.
- End Algorithm DMotifs**

Steps 2 to 7 implement one iteration of d-motif finding, using the p input promoters as positives and a randomly generated set of sequences as negatives. The loop 1..8 repeats this iteration *numbags* times, each time using a different set of negatives, and thus implements a simple form of “bagging”, a technique popular in machine learning. An important feature of the algorithm is that it makes no commitment to the exact search space S used, and can therefore be used with any motif model and any heuristic search that traverses only part of the search space.

4 Simple motifs

This section describes a simple application of the DMotifs algorithm. The motif model used is that of consensus sequences with IUPAC degenerate symbols. In particular, each motif in the space

is a string over the alphabet $\{A,C,G,T,R,Y,S,W,K,M,N\}$, with each of R,Y,S,W,K,M representing a disjunction of two bases, and N representing a spacer (any base). This model, henceforth called the *consensus model*, has been successful in describing transcription factor binding sites in *S. cerevisiae* [19] and has been used in other algorithmic approaches, especially enumerative ones such as [15] and [14], to detect binding sites in that organism. The typical length of the string, excluding the spacers, is 6-8.

To use this model in the d-motif framework, we need to compute $p_{s,k}$, the probability distribution of the count k of a motif s in a randomly generated sequence of length L . We adopt a simulation-based approach to estimate this probability distribution, since existing methods are either too inefficient to be run on all motifs in the model [10] or are incompatible with the model [18]. Each simulation generates a random text of length L and counts the occurrences of s as specified by the model. Suppose that N such simulations are done, and let X be the number of simulated sequences that yielded a count of k . Then we estimate $p_{s,k}$ as being X/N . Suppose $p_{s,k}$ is said to be correctly estimated if $|p_{s,k} - X/N| < c_1$. Also suppose we wish to compute $p_{s,k}$ for all s in some set S , and wish to bound by c_2 the probability of estimating *any* $p_{s,k}$ incorrectly. We prove by a simple probabilistic analysis (given in the Appendix, Section 9.2) that the above bounds are achieved if

$$N > \frac{\ln(2|S|/c_2)}{2c_1^2}$$

A critical feature of the relation is that N grows only logarithmically with the size of S . Our implementation uses $c_1 = 0.001$ and $c_2 = 0.01$. Since we shall be interested in computing $p_{s,k}$ only for small values of k (typically less than 4), it is reasonable to use such relatively large values of c_1 .

In this application, the consensus model is further constrained to have spacers only in the middle of the motif, and only a small number (1 or 2) of degenerate symbols. These restrictions are motivated by efficiency issues and justified by an empirical study of the binding sites in the SCPD database [19]. The count of a motif in a promoter includes overlapping occurrences, and looks at both strands. This can be easily computed in $O(L)$ time. The random text is generated according to a 3^rd order Markov model that is trained on all promoters of the organism under study (yeast, in our case). Thus, since we have a well defined motif search space, an algorithm to count motif occurrences, and accurate estimates of the probability distributions of motif counts, we can invoke the d-motif framework of Section 3 for finding simple motifs. For efficiency of search, the p-value scores are computed (Step 5 of algorithm DMotifs) only if the TNoM score is below a threshold.

4.1 Results on known regulons

We ran DMotifs on known regulons, which are sets of genes known to be co-regulated, and where the binding site has been biologically characterized. SCPD catalogues such regulons in yeast, along with the binding site consensus sequence for each. DMotifs was run on *all* regulons in SCPD that have at least 3 genes, and have a consensus sequence for the binding site. The motif length (excluding spacers) is a parameter that was varied according to the length of the known motif. The other parameter is the maximum number of spacers allowed by the motif model. In all runs, a maximum of 1 degenerate symbol was allowed. Table 1 describes the results. We see that in 18 out of the 22 regulons, the known consensus closely matches one of the top 10 motifs reported by the algorithm, and in 15 of these 18, the match is in the top 2.

The regulon roX1 is an example of a count based approach leading to a false positive. As we see in Table 1, DMotifs finds the motif CCTATTG, which matches the binding site closely, at rank 7. However, if we run the algorithm YMF of Sinha and Tompa [14], which measures over-representation

REGULON	BINDING SITE	MOTIF FOUND	RANK	PARAMETERS
ABF1	TCRNNNNNACG	TCANNNNNNAMG	2	6,6
CPF1	TCACGTG	CACGTG	1	6,6
CSRE	YCGGAYRRRAWGG	CGGATGRA	8	8,0
SCB	CNCGAAA	TCGCGAA	2	7,0
GAL4	CGGNNNNNNNNNNCCG	CGGNNNNNNNNNNCCG	1	6,11
GCR1	CWTCC	CTTCC	13	5,0
HAP1	CGGNNNTANCGG	GGANNNNNCGG	1	6,6
HSE	TTCNNGAA	TTMTAGAA	6	8,0
	TTCNNGAA			
	GAANNNTCC			
	GAANNNTCC			
MCB	WCGCGW	ACGCGT	1	6,6
MCM1*	CCNNNWWRGG	TTTCCATA	1	8,0
MATa2	CRTGTWWWW	CATGTMA	2	7,0
MIG1	CCCCRNNWWWWW	MCCCCAG	1	7,0
PHO4	CACGTK	CACGTG	1	6,6
PDR3	TCCGYGGA	TCCGYGGA	2	8,0
REB1	YYACCCG	YTACCCG	1	7,0
ROX1	YYNATTGTTY	CCTATTG	7	7,0
RAP1	RMACCCA	ACCCAGW	1	7,0
CAR1	AGCCGCSA	TAGCCGCS	2	8,0
SFF	GTMAACAA	NOT FOUND	-	-
STE12	ATGAAA	ATGNAAC	1	6,6
TBP	TATAAW	NOT FOUND	-	-
UASPHR	CTTCCT	NOT FOUND	-	-
	GTSAAAGTAWG			

Table 1: Results on yeast regulons: Columns 1 and 2 are the regulon name and the known binding site consensus. The third column is the first close match to this consensus in the list of top 10 motifs reported by DMotifs. Column 4 is its rank in this list, and Column 5 gives the parameter values used: motif length, maximum number of spacers. * In the case of MCM1, the reported motif does not match the known consensus, but is found to occur overlapping with the MCM1 binding site in most of the promoters. Hence it is counted as a match.

based on total counts, the top motif is TTYTTTT, and all of the best 20 motifs are variations of this motif. TTYTTTT has 58 occurrences in the three genes, whereas the expectation is 8.9 per gene, so the total count is significantly high. A closer examination reveals that the 58 occurrences are distributed as 10,11 and 37 in the 3 genes. Thus in two of the genes, the count is close to expectation, and the total count is boosted by the third gene. DMotifs is able to recognize this phenomenon and does not report the motif, while a count based approach like YMF incorrectly reports it as significant.

5 Composite motifs

For some time now, there have been attempts to model and detect *combinations* of motifs that occur in promoter regions [7], [9], [5] and [4]. Such a motif is the footprint of an interaction of multiple transcription factors and other regulatory proteins, found on the promoter, and is often called a “higher order” motif or a “composite” motif. For instance, in the yeast *S. cerevisiae*, the transcription factors MCM1 and STE12 are known to jointly regulate the basal expression of genes FAR1 and STE2, with the corresponding promoters having clustered binding sites for both factors [17]. Similarly, the promoters of the eight core histone genes have a conserved pattern $m_1 - d_1 - m_2 - d_2 - m_2$, where m_1 and m_2 are well characterized simple motifs, d_1 is a tight interval of spacers, and d_2 is a less tight interval [12]. Composite motifs in other eukaryotes are known to

be more complex [1].

Here we show the application of the DMotifs algorithm to the detection of composite motifs. A composite motif s is modelled as a triplet (s_1, s_2, d) , where s_1 and s_2 are simple motifs following the consensus model, and s is said to occur when s_1 and s_2 occur, on either strand and in either order, within d bases of each other. This is a simple “paired motif” model that has support from the biology literature [11]. Additional constraints may be imposed to obtain a different model to which d-motif finding can then be applied, thereby exploiting the generality of the framework. To use DMotifs, we need to specify the search space of motifs, and the *a priori* probability distribution of motif counts.

Let S be the set of simple motifs used in Section 4. For the motif space, we use the set $S' = \{(s_1, s_2, d) | s_1 \in S, s_2 \in S, d = \text{a fixed constant}\}$. Given a triplet $s = (s_1, s_2, d)$, we compute $p_{s,k} = Pr[\text{count of } s \text{ in a random text of length } L \text{ is } k]$ by making some simplifying assumptions. We assume that the counts of s_1 and s_2 are independent. Then

$$p_{s,k} = \sum_{k_1=k}^{k_{max}} \sum_{k_2=k}^{k_{max}} p_{s_1,k_1} p_{s_2,k_2} p_{k_1,k_2,k}$$

where $p_{k_1,k_2,k}$ is the probability that there are k occurrences of the paired motif s given that there are k_1 occurrences of s_1 and k_2 occurrences of s_2 . Under certain assumptions, this probability distribution can be computed irrespective of the patterns s_1 and s_2 . This is done using a dynamic programming algorithm that is described in the Appendix, Section 9.3. The analysis assumes that when counting occurrences of a paired motif, only non-overlapping occurrences are considered.

For efficiency, the implementation searches a subset of the motif space S' . It first computes the set of the best t simple motifs, where t is a parameter (set to 2000), and then searches the space of all pairs of motifs from this set, with a fixed value of d .

5.1 Application to biological data

The composite motif application of DMotifs was validated on a biological data set. Transcription factors MCM1 and STE12 are known to jointly regulate the expression of gene FAR1. SCPD catalogues three other genes MFA1, MFA2 and STE2 in which the binding sites of both MCM1 and STE12 are known to occur. We took these four genes (FAR1, MFA1, MFA2 and STE2) and ran DMotifs on them, to look for composite motifs. The search space included all pairs of 6-mers, separated by up to 40 bases. The 2^{nd} motif reported was the pair (TGAAAC, GGAAAT). We noted that TGAAAC is a close match to the STE12 consensus ATGAAA, and GGAAAT occurs overlapping with MCM1 binding sites in these four genes, and is better conserved than the binding sites themselves. While ATGAAA was ranked 2^{nd} among the simple motifs in this regulon, GGAAAT was ranked 56^{th} and would not have been deemed significant had it not been for the composite motif it is a part of. The top ranking composite motif was (GGAAAT, TACATG), which represents a long simple motif GGAAATTTACATG that occurs in three of the four promoters.

6 Experiments with synthetic data

We now describe experiments where the DMotifs algorithm was run on synthetic sequences with planted motifs. Ten sequences were generated according to a random process that mimics upstream sequences in yeast. A pattern s of length 6 was chosen at random. We computed its expected count E_s in 10 random sequences, and planted it at $r \times E_s$ random places in the 10 synthetic sequences. (r is the “over-representation” factor, a parameter to the experiments. The results given below

are for $r = 2.5$.) The planting was done in a way that distributed the instances evenly across the sequences. This set of sequences was then fed as input to the motif finding algorithm under study, and the algorithm was made to report motifs in decreasing order of significance. One algorithm is said to *win* against another if it reports s at a higher (better) rank. The difference in the ranks of s in the two reported lists is called the *margin*.

In the first experiment, algorithm DMotifs was compared with the algorithm YMF [14], a count based algorithm. The above experiment was run 500 times, each time with a different randomly chosen planted motif. Both algorithms were run with the same parameters - 0 spacers and 0 degenerate symbols. DMotifs won against YMF on 285 of the 500 trials, YMF winning 49 times, and the remaining 166 were ties. We repeated the same experiment, but now with motif occurrences being planted independently at random. YMF outperformed DMotifs comprehensively, winning 343 and losing 69. These experiments demonstrate the strength of the d-motif approach when motif instances are well distributed over the promoters, as one would expect in real data.

In another experiment, two variants of DMotifs were compared - one that uses the p-value score and one that uses the TNoM score for motif significance. Out of 500 trials (as above), the p-value variant won in 136 and the TNoM score variant won in 51, the rest being ties. The average margin for the former's wins was 14, while that for the latter was 2. The scale is tipped further in favor of the p-value variant when the over-representation factor r is reduced. These results provide clear experimental evidence that the p-value score has more discriminative power. In a similar experiment, we studied the effect of bagging on the performance. The two variants of DMotifs that were compared used $numbags = 10$ and $numbags = 1$ respectively. The former won 260 times out of 500, and lost 48 times, thereby making a strong case for using the bagging heuristic. In yet another heuristic-evaluation experiment, we implemented DMotifs to use as negative promoters a randomly chosen subset of all yeast promoters, instead of the the randomly generated sequences that the standard implementation uses. Both variants performed equally well, and out of 500 trials, 260 were ties, with the two variants winning 122 and 118 times respectively.

We also tested the composite motif finder on synthetic data. 10 sequences were created as above. Two patterns of length 6 each were chosen at random, each with up to 6 spacers and 1 degenerate symbol. This pair was planted at random positions in 7 of the 10 sequences. When planting the pair, the distance between the two patterns was chosen at random from a Poisson distribution with mean 15. DMotifs was run on these 10 sequences, and configured to look for pairs of simple motifs of length 6 (each with up to 6 spacers and 1 degenerate symbol), separated by at most 40 bases. The entire experiment was repeated 10 times, each time with a different pair of motifs. In each case, the planted pair or a close variant was recovered as the top ranking motif. However, this in itself does not measure the performance of the algorithm in any quantitative sense, since planting the motif in fewer or more sequences will change the performance substantially. On the other hand, the scores of the motifs do tell us something interesting about the evaluation criteria. In one of the 10 experiments, we planted the pair (GTGTTC,AGTGCT) and examined, for one of the bagging iterations, the scores of all motifs in the search space. The planted motif was recovered as the top ranking motif based on its p-value score. It had a TNoM score of 3. At the same time, over 50% of the top 1000 motifs reported were found to have a TNoM score of 3 or less. Many of these were very different from the planted motif, and had a better TNoM score merely by chance. This result strongly argues for preferring p-value scores over TNoM scores. When exploring a large motif space, it is very likely that some spurious motifs will score equally well with TNoM. The same argument holds against using raw counts of motifs as the evaluation criterion.

7 Future work and conclusion

DMotifs invokes an enumerative search of the motif space, and leaves open the issue of an efficient traversal of the space. This is foremost on our agenda for future work. An algorithm such as that of Marsan and Sagot [9], which can efficiently output all composite motifs that occur a certain number of times in the input regions, may be used for traversal, in adjunction with a heuristic that rejects motifs with low counts without computing their p-value scores. Another issue worth investigating is how to decide upon the significance of p-value scores. Currently, we are only able to *rank* motifs by their scores. One of the tasks cut out for the immediate future is the application of the framework to other motif models, such as the mismatch model. A broad and important direction of research is to extend the d-motif technique to work with non-integral counts. This would allow more flexibility in the notion of motif occurrence.

This work presents a new characterization of the over-representation of a motif, and develops a motif discovery algorithm under this characterization. The general algorithm is adapted for two specific motif models, and shown to work well on real as well as synthetic data.

8 Acknowledgements

The author wishes to thank his advisor, Martin Tompa, for his active support through regular discussions. The contribution of Mathieu Blanchette, who had insightful comments throughout the project, cannot be overstated. The author also thanks Rimli Sengupta for the suggestions she made on the subject.

9 Appendix

9.1 Computing the p-value score of a motif

We assume the knowledge of $p_{s,k} = Pr[\text{count of motif } s \text{ in a random text of length } L \text{ is } k]$, for $k = 0 \dots k_{max}$. Let $P(n, p, k_{max}, e)$ denote the probability that the best decision stump for motif s with p positive and n negative promoters makes e errors. Let the observed TNoM score of s for the configuration derived from the given promoters be TM_s . Clearly,

$$pval_s = \sum_{e=0}^{TM_s} P(n, p, k_{max}, e)$$

Also, let $P(n, p, k_{max}, l, e)$ be the probability that of the potentially many decision stumps with the least classification score, l is the rightmost (largest in value) and has a score of e . Then we have

$$P(n, p, k_{max}, e) = \sum_{l=0}^{k_{max}} P(n, p, k_{max}, l, e)$$

Let N_i^+ and N_i^- denote the number of '+'s and '-'s respectively at integer i in a configuration, and let $\delta_i = N_i^+ - N_i^-$. We need to compute the probability that l is the rightmost best decision stump and has score e . It has score e iff i '-'s are placed at $x \geq l$ and $e - i$ '+'s are placed at $x < l$, where $0 \leq i \leq e$. It is the rightmost best decision stump iff two conditions are satisfied:

- Each decision stump to its left has score greater than or equal to e . This is equivalent to saying that $\delta_{l-1} \leq 0$, $\delta_{l-2} + \delta_{l-1} \leq 0$, ..., $\delta_0 + \delta_1 + \dots + \delta_{l-1} \leq 0$. Let $S(x, y, low, high, t)$ denote

the probability of a (partial) configuration that places x '-'s and y '+'s on the integer line anywhere between integers low and $high$, such that $\sum_{i=min}^{high} \delta_i \leq t$ for all $low \leq min \leq high$.

- Each decision stump to its right has score greater than e . This is equivalent to saying that $\delta_l > 0, \delta_l + \delta_{l+1} > 0, \dots, \delta_l + \delta_{l+1} + \dots + \delta_{k_{max}} > 0$. Let $T(x, y, low, high, t)$ denote the probability of a (partial) configuration that places x '-'s and y '+'s on the integer line anywhere between integers low and $high$, such that $\sum_{i=low}^{max} \delta_i > t$ for all $low \leq max \leq high$.

Then we can write the recurrences

$$\begin{aligned}
P(n, p, k, l, e) &= \sum_{i=0}^e \binom{n}{i} \binom{p}{e-i} S(n-i, e-i, 0, l-1, 0) T(i, p-e+i, l, k, 0) \\
S(x, y, low, high, t) &= \sum_{i=0}^x \sum_{j=0}^{t+i} \binom{x}{i} \binom{y}{j} p_{s,high}^{i+j} S(x-i, y-j, low, high-1, t-(j-i)) \\
T(x, y, low, high, t) &= \sum_{i=0}^x \sum_{j=t+i+1}^y \binom{x}{i} \binom{y}{j} p_{s,low}^{i+j} T(x-i, y-j, low+1, high, t-(j-i))
\end{aligned}$$

The base conditions are:

$$\begin{aligned}
S(x, y, low, low, t) &= p_{s,low}^{x+y} \quad \text{if } y-x \leq t \\
&= 0 \quad \text{otherwise} \\
T(x, y, high, high, t) &= p_{s,high}^{x+y} \quad \text{if } y-x > t \\
&= 0 \quad \text{otherwise}
\end{aligned}$$

9.2 Estimating probability distributions of motif counts by simulation

We want to estimate $p_{s,k} = Pr[\text{motif } s \text{ occurs } k \text{ times in a random text of length } L]$, for $k = 0..k_{max}$. Each simulation generates a random text of length L and counts the occurrences of s as specified by the model. Suppose that N such simulations are done, and X of these yield promoters with k occurrences. To bound the absolute error in computing $p_{s,k}$ by c_1 , we need $|p_{s,k} - X/N| < c_1$, i.e., $|X - Np_{s,k}| < Nc_1$, and we shall say that $p_{s,k}$ is estimated incorrectly if this condition is violated. We know that X is a binomial variable with parameters N and $p_{s,k}$, and hence its mean and variance are $Np_{s,k}$ and $Np_{s,k}(1-p_{s,k})$ respectively. Hence, using Chernoff's bound, we have $Pr[|X - Np_{s,k}| > Nc_1] \leq 2e^{-2Nc_1^2}$. Thus, $Pr[p_{s,k} \text{ is estimated incorrectly}] \leq 2e^{-2Nc_1^2}$. If we wish to compute $p_{s,k}$ for all s in some set S , we can claim that $Pr[p_{s,k} \text{ is incorrect for some } s] \leq 2|S|e^{-2Nc_1^2}$. Also suppose we want to bound the probability of making *any* incorrect estimate at all by c_2 . This is achieved if we have $2|S|e^{-2Nc_1^2} < c_2$, which is the same as

$$N > \frac{\ln(2|S|/c_2)}{2c_1^2}$$

This relation is used in deriving the number of simulations that are done. For instance, if $c_1 = 10^{-3}$, $c_2 = 10^{-2}$ and S contains all motifs of length 6 in the consensus model, with at most 1 degenerate symbol and no spacers ($|S| = \binom{6}{0}6^04^6 + \binom{6}{1}6^14^5$), we have $N > 7.96 \times 10^6$.

9.3 Computing probabilities of paired motif occurrences

The goal is to compute $p_{k_1, k_2, k}$, which is the probability that there are k occurrences of the paired motif $s = (s_1, s_2, d)$ given that there are k_1 occurrences of s_1 and k_2 occurrences of s_2 . If we assume that all choices of positions of the k_1 occurrences of s_1 and k_2 occurrences of s_2 are equally likely, then computing $p_{k_1, k_2, k}$ amounts to solving a particular balls-and-urns problem, and can be done irrespective of the patterns s_1 and s_2 .

The BALLS_AND_URNS problem is: *Given that k_1 black balls and k_2 white balls are placed in a sequence of L urns ordered from left to right, at most one ball in an urn, how many distinct configurations or placements have at least k pairs, each of which includes a black and a white ball, such the number of urns between the two balls of each pair is less than or equal to d ? The k pairs are non-overlapping - no two pairs share a ball and no ball in a pair lies between the balls of another pair.*

It can be shown that the following process P' can be used to systematically generate all distinct configurations described in the above problem. On inputs $low, high, k_1, k_2, k$, it places k_1 black balls and k_2 white balls between positions low and $high$ such that at least k non-overlapping pairs are formed.

Process P' :

Inputs: $low, high, k_1, k_2, k$

1. If $k = 0$, place the $k_1 + k_2$ balls between low and $high$. Return.
2. Choose a pair of positions π_1, π_2 with $low \leq \pi_1 < \pi_2 \leq high$ and $\pi_2 - \pi_1 \leq d + 1$. Place a black ball at π_1 and a white ball at π_2 , or vice versa.
3. Choose two numbers $x \leq k_1 - 1$ and $y \leq k_2 - 1$. Place, if possible, x black balls and y white balls in the range $low \dots (\pi_1 - 1)$, making sure that no valid pair (of black and white balls, with less than $d + 1$ urns between them) whose right ball is to the left of π_2 , is formed by this placement.
4. Repeat process P' on inputs $(\pi_2 + 1, high, k_1 - 1 - x, k_2 - 1 - y, k - 1)$.

End Process P'

By making different choices in each step, this process, when called with inputs $1, L, k_1, k_2, k$, can generate exactly the set of configurations mentioned in the BALLS_AND_URNS problem. The process is recursive, and the number of configurations it generates can be counted by a dynamic programming algorithm.

Let $S(l, k_1, k_2, k)$ be the number of ways to place k_1 black balls and k_2 white balls in the range of positions $0 \dots l$, such that at least k non-overlapping pairs are formed. Let $X(l, k_1, k_2)$ be the number of ways to place k_1 balls of color c_1 and k_2 balls of color c_2 in the range $0 \dots l$, such that no valid pair of black and white balls is formed in the range $0 \dots (l + 1)$, given that a ball of color c_1 is placed at position $l + 1$. The recurrence for S is:

$$S(l, k_1, k_2, k) = \sum_{\pi_1=0}^{l-2k+1} \sum_{\delta=0}^d \sum_{x=0}^{k_1-1} \sum_{y=0}^{k_2-1} ([X(\pi_1 - 1, x, y) + X(\pi_1 - 1, y, x)] \\ \times S(l - \pi_1 - \delta - 2, k_1 - x - 1, k_2 - y - 1, k - 1))$$

The important base condition is:

$$S(l, k_1, k_2, k) = \binom{l+1}{k_1+k_2} \binom{k_1+k_2}{k_1} \quad \text{if } k = 0$$

Other base conditions are trivial, and are omitted here. The recurrence for X is :

$$X(l, k_1, k_2) = \sum_{u=k_1+k_2-1}^l X(u-1, k_1-1, k_2) + \sum_{u=k_1+k_2-1}^{l-d-1} X(u-1, k_2-1, k_1)$$

The important base conditions are:

$$\begin{aligned} X(l, k_1, k_2) &= \binom{l-d}{k_2} \text{ if } k_1 = 0 \wedge (0 < k_2 \leq l-d) \\ &= \binom{l+1}{k_1} \text{ if } k_2 = 0 \wedge (0 < k_1 \leq l+1) \end{aligned}$$

The probability that there are at least k pairs formed is then given by

$$\frac{S(L-1, k_1, k_2, k)}{\binom{L}{k_1+k_2} \binom{k_1+k_2}{k_1}}$$

The probability of forming exactly k pairs is then trivially computed. The dynamic programming algorithm runs in time $O(L^2 k_1^2 k_2^2 kd)$ and has to be run only once for fixed values of L and d .

References

- [1] M. I. Arnone and E. H. Davidson. The hardwiring of development: organization and function of genomic regulatory systems. *Development*, 124:1851–1864, 1997.
- [2] T. L. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1-2):51–80, Oct. 1995.
- [3] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and Z. Yakhini. Tissue classification with gene expression profiles. *Journal of Computational Biology*, 7:559–584, 2000.
- [4] W. N. Grundy, T. L. Bailey, C. P. Elkan, and M. E. Baker. Meta-meme: Motif-based hidden markov models of protein families. *Computer Applications in the Biosciences*, 13(4):397–406, 1997.
- [5] D. GuhaThakurta and G. D. Stormo. Identifying target sites for cooperatively binding factors. In *RECOMB01: Proceedings of the Fifth Annual International Conference on Computational Molecular Biology*, Montreal, Canada, Apr. 2001.
- [6] G. Z. Hertz and G. D. Stormo. Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps. In H. A. Lim and C. R. Cantor, editors, *Proceedings of the Third International Conference on Bioinformatics and Genome Research*, pages 201–216. World Scientific Publishing Co., Ltd., Singapore, 1995.
- [7] Y.-J. Hu, S. Sandmeyer, C. McLaughlin, and D. Kibler. Combinatorial motif analysis and hypothesis generation on a genomic scale. *Bioinformatics*, 16(3):222–232, 2000.
- [8] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 8 October 1993.

- [9] L. Marsan and M.-F. Sagot. Extracting structured motifs using a suffix tree - algorithms and application to promoter consensus identification. In *RECOMB00: Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, pages 210–219, Tokyo, Japan, Apr. 2000.
- [10] P. Nicodème, B. Salvy, and P. Flajolet. Motif statistics. Technical Report RR-3606, INRIA Rocquencourt, Jan. 1999.
- [11] Y. Ohmori, R. D. Schreiber, and T. A. Hamilton. Synergy between interferon-gamma and tumor necrosis factor alpha in transcriptional activation is mediated by cooperation between signal transducer and activator of transcription 1 and nuclear factor kappa b. *The Journal of Biological Chemistry*, pages 14899–14907, 1997.
- [12] P. Pavlidis, T. Furey, M. Liberto, D. Haussler, and W. Grundy. Promoter region-based classification of genes. *Pacific Symposium on Biocomputing*, 2000.
- [13] F. P. Roth, J. D. Hughes, P. W. Estep, and G. M. Church. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnology*, 16:939–945, Oct. 1998.
- [14] S. Sinha and M. Tompa. A statistical method for finding transcription factor binding sites. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Aug. 2000.
- [15] M. Tompa. An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 262–271, Heidelberg, Germany, Aug. 1999. AAAI Press.
- [16] J. van Helden, B. André, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology*, 281(5):827–842, Sept. 4 1998.
- [17] A. Wagner. Genes regulated cooperatively by one or more transcription factors and their identification in whole eukaryotic genomes. *Bioinformatics*, 15(10):776–784, 1999.
- [18] M. S. Waterman. *Introduction to Computational Biology*. Chapman & Hall, 1995.
- [19] J. Zhu and M. Q. Zhang. SCPD: a promoter database of the yeast *Saccharomyces cerevisiae*. *Bioinformatics*, 15(7/8):563–577, July/August 1999. <http://cgsigma.cshl.org/jian/>.