12-2016

Finding DNA Motifs: A Probabilistic Suffix Tree Approach

Abhishek Majumdar
University of Nebraska-Lincoln, majumdar@cse.unl.edu

FINDING DNA MOTIFS: A PROBABILISTIC SUFFIX TREE APPROACH

by

Abhishek Majumdar

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professors Stephen Scott and Jitender Deogun

Lincoln, Nebraska

December, 2016

FINDING DNA MOTIFS: A PROBABILISTIC SUFFIX TREE APPROACH

Abhishek Majumdar, Ph.D.

University of Nebraska, 2016

Advisers: Stephen Scott and Jitender Deogun

We address the problem of *de novo* motif identification. That is, given a set of DNA sequences we try to identify motifs in the dataset without having any prior knowledge about existence of any motifs in the dataset. We propose a method based on Probabilistic Suffix Trees (PSTs) to identify fixed-length motifs from a given set of DNA sequences. Our experiments reveal that our approach successfully discovers true motifs. We compared our method with the popular MEME algorithm, and observed that it detects a larger number of correct and statistically significant motifs than MEME. Our method is highly efficient as compared to MEME in finding the motifs when processing datasets of 1000 or more sequences. We applied our method to sequences of mutant strains of *Exophiala dermatitidis* and successfully identified motifs which revealed several transcription factor binding sites. This information is important to biologists for performing experiments to understand their role in different regulatory pathways affected by cdc42. We also show that our PST approach to *de novo* motif discovery can be used successfully to identify motifs in ChIP-Seq datasets. These motifs in turn identify binding sites for proteins in the sequences.

ACKNOWLEDGMENTS

My heart felt gratitude goes to my advisors, Prof. Stephen D. Scott and Prof. Jitender S. Deogun, for their continuous support, guidance and encouragement during the dissertation. Without their patience, motivation, guidance, immense knowledge and encouragement it would not have been possible for me to complete the dissertation.

I would also like to thank Prof. Steven Harris, for his continued support, insight and help. His expertise has been a guiding star in the development of this dissertation.

Table of Contents

Chapter 1

Introduction

A motif is a subsequence that occurs frequently in a given set of biological sequences. Motifs have some biological significance associated with them, e.g., denoting transcription factor binding sites [12]. The motif-finding problem is to identify all motifs in a given set of sequences. This problem has been studied for more than two decades and is very interesting both from application and theoretical points of view. Numerous approaches (expectation maximization, graphical, biclustering, etc.) such as MEME [7], WINNOWER [26] and MUSA [22] have been investigated. However, even after much effort no good approach that gives a complete and correct solution to the problem has been reported.

In this dissertation we develop a motif-finding approach based on Probabilistic Suffix Trees (PSTs). A string which occurs randomly in a data set of sequences cannot be a motif. Conversely, for a string to be a motif, the probability of its individual positions must be dependent on each other. Therefore a model like a Markov chain can be used to identify motifs. A Probabilistic Suffix Tree (PST) is a similar concept but with more efficient memory usage. PSTs are capable of identifying frequently occurring subsequences in datasets. PSTs also provide an approximately accurate probability distribution of symbols by observing at most L symbols in the preceding

subsequence [9]. This makes PSTs memory efficient tools. A PST is a simple and powerful tool which can be used easily. PSTs have been previously used for classifying protein sequences [9]. They can also be used for correcting corrupted texts and DNA base predicting [27]. In this dissertation we use PST to build a tool for *de novo* motif identification in DNA sequences. Our method is an unsupervised method and does not require any of alignment between sequences or learning of labeled data. It uses PST to generate seeds which act as regular expressions for searching candidate motifs in the dataset. It then evaluates statistical significance of candidate motifs to label them as motifs or non-motifs. We also show that our method is robust enough to identify motifs in ChIP-Seq data.

Our validation experiments reveal that the predicted motifs are *correct* meaning they either match exactly to some existing motifs or are a substring of some bigger known motif. Our results can be summarized as follows:

1. Accurately finds motifs of different lengths (accuracy near 1.0 for smaller motifs and decreases slightly as motif size increases).

2. Finds motifs with high accuracy on datasets both small and large ($\approx 33,000$ sequences).

3. Finds a larger number of correct and statistically significant motifs than popular motif finding tool MEME.

4. Finds motifs more efficiently than MEME when handling real datasets of more than 1000 sequences.

5. Successfully identifies important transcription factors which are known to play important role in regulatory pathway affected by cdc42.

6. Provides a tentative list of transcription factors which will used as a starting point for wet lab experiments in determining their role in regulatory pathways.

7. Finds motifs in ChIP-Seq data, to identify protein binding sites. Shows how this process can be applied to ChIP-Seq data in general.

This dissertation is structured as follows: In Chapter 2, we state and describe the motif identification problem. Chapters 3 and 4 discuss the related work and background needed for understanding Probabilistic Suffix Trees and their applications. In Chapter 5, we describe our method, and associated metrics for its performance eval-

uation. We perform several experiments by applying our method to various datasets (synthetic and real). We also compare our method to the popular motif finding tool MEME and discuss the results of these experiments on the real datasets. We observe that our approach finds more correct and statistically significant (with e-value ≤ 0.05) motifs than MEME. It also takes much lesser time than MEME in doing so (when applied to datasets of > 1000 sequences). Chapter 6 describes the application of our method to a generalized form of our problem. That is how to find significant motifs of a range of sizes instead of a fixed one. We rerun our experiments on the data used in Chapter 5, for validation purposes. We then apply our method to find motifs of length range 7 to 9 bases among sequences of three mutant strains of black yeast *E. dermatitidis*. We successfully identify different transcription factor binding sites in these mutant using the predicted motifs. This information will be central to performing further experiments in understanding their roles. In Chapter 7, we show how to apply our approach to data obtained from ChIP-sequencing and identify motifs. Finally, in Chapter 8, we talk about the future projects that can be undertaken with respect to our PST based approach.

Chapter 2

Problem Description

DNA or deoxyribonucleic acid is a molecule that carries the genetic information of all living organisms. Most DNAs consist of two strands which are coiled around each other to form a double helix structure. Each strand is composed of nucleotides, deoxyribose sugar and phosphate group. A nucleotide can be any one of the four nitrogen containing bases: adenine (A), guanine (G), cytosine (C) or thymine (T). A DNA sequence is a long string of nucleotide symbols. The information contained in a DNA is based on the order of these base symbols.

DNA motifs are defined as recurring patterns in the DNA. Motifs are typically 5–20 symbols long. A sequence might have a single or multiple copies of a motif. Motifs are associated with important biological functions like binding sites for proteins (nucleases and transcription factors), involvement in ribosome binding, mRNA processing and transcription termination [12]. Thus successful identification of motifs plays an important role in understanding the inner mechanisms of biological functions.

We investigate the problem of *de novo* motif identification. Our objective is to identify motifs which are most enriched from an unbiased set of DNA sequences, about which we do not have any prior information. A motif is said to occur frequently in a dataset if it occurs in multiple (> 1) sequences in the dataset S with e-value ≤ 0.05.

E-value of a motif is the expected number of strings of same length as the motif, with equal or greater information content than the motif, that occur simply by chance in the dataset S [17].

Now we formally define the problem as: Given a set S of DNA sequences and a parameter m, find all motifs of length m that occur in S. In Chapter 5 we address this problem and propose our solution.

In Chapter 6 we deal with the generalized version of the above problem. We define it as follows: Given a set S of DNA sequences and parameters m_1 and m_n where $m_1 \leq m_n$, find all motifs of lengths m_1 to m_n.

Numerous approaches (MEME [7], WINNOWER [26], MUSA [22]) have been developed to address the motif finding problem. However so far no method gives a complete and efficient solution. As pointed out by Simche et al. [29], unbiased validation of *de novo* motifs is difficult. A statistical method for validating such motifs is to treat the motif discovery problem as discrete classification problem. That entails clustering sequences into clusters, identifying motifs specific to each cluster and then using those motifs to distinguish its cluster from the other clusters. Simcha *et al.* [29] observe that most algorithms perform poorly in this regard. Some other common problems associated with *de novo* motif discovery include a high rate of false positives, inability to identify all possible motifs actually present, and time taken to find the motifs.

In this dissertation we propose a motif finding approach based on Probabilistic Suffix Trees. Our approach aims to develop a method that can discover accurate motifs from data sets in lesser time than existing tools. PSTs do not require any kind of aligning of sequences for identifying any motifs and have very efficient memory utilization. Since there is no alignment process involved, the actual process of searching for motifs, is pretty fast for datasets of size greater than thousand sequences.

Applying on synthetic dataset we observe that our PST-based method identifies motifs specific to clusters of sequences, such that the motifs are successfully capable of distinguishing its cluster from the other clusters (AUC ≈ 0.983).

In our comparison with the popular motif finding tool MEME, we find that our PST based approach takes much less time and finds larger number of statistically significant motifs when used on the same dataset. We also observe from our validation experiments, that our accuracy is very good (near 1.0 for smaller motifs and decreases slightly with increasing motif size).

Chapter 3

Related Work

In this chapter we review some of the algorithms proposed for the motif discovery problem.

Das and Dai [11] present a comprehensive survey of DNA motif finding algorithms. A comparison of different motif finding algorithms to help a user better understand which algorithms to use is given by Tompa et al. [31].

Pevzner et al. [26] investigate a graph-theoretic approach for motif discovery. Their approach is called WINNOWER. First, given a set of sequences $S = \{s_1, \ldots, s_t\}$, motif length ℓ and maximum number of mismatches d (allowed in the motifs), a graph $G(S, \ell, d)$ is constructed in the following manner. Each vertex represents a ℓ-length substring of each sequences s_i starting at all positions j in s_i such that $1 \leq j \leq |s_i| - \ell + 1$. Vertices s_{ij} and s_{ab} are connected if they are not part of the same sequence i.e. $i \neq a$ and the Hamming distance between the sunstrings is $\leq d$. For instances, let $s_1 = ATGAAATG$ and $s_2 = GTGAAACA$ be two sequences $\in S$. Let $\ell = 4$ and $d = 2$. Then from s_1 we have vertices representing $\{ATGA, TGAA, GAAA, AAAT, AATG\}$ and from s_2 we have vertices representing $\{GTGA, TGAA, GAAA, AAAC, AACA\}$. There can be edge between vertices $ATGA$ and $GTGA$ because they are from different sequences and the Hamming distance be-

tween them is 1. A motif corresponds to a clique in the graph $G(S, \ell, 2d)$. The problem of finding motifs is reduced to finding large cliques in the graph. The algorithm basically converges to a collection of cliques (motifs) by eliminating inconsistent edges iteratively.

Workman et al. [32] propose the algorithm ANN-SPEC for motif discovery. It is basically a neural network approach. It is a sparsely encoded perceptron with single processing unit. Given a set of sequences and the genome background frequencies, the neural network tries to find parameter values which maximize the probability that the motif occurs in each sequence of the set. Their training process is based on the Gibbs sampling process.

Liu et al. [19] present BioProspector, a Gibbs sampling-based algorithm for motif discovery. The algorithm implements a process called a threshold sampler. This algorithm differs from the normal Gibbs sampling approach at several points. For instance, it computes the score of the segments differently (using a 3rd-order Markov model) and it accommodates for the fact that a sequence can have multiple motifs while some might not have any. This is done by controlling the sampling process with two thresholds. It is also capable of capturing two-block motifs and palindromic motifs. The algorithm uses a different motif scoring scheme to quantify the statistical significance of a motif.

Fogel et al. [15] use a genetic algorithm [23] approach for discovery of transcription factor binding sites. Genetic algorithms (GA) are heuristic search techniques based on evolutionary and genetic concepts. GAs simulate survival of fittest process to solve an optimization problem. Solutions to the problem represent chromosomes/individuals. Collection of chromosomes form a generation. Quality of an individual (solution) is evaluated using a fitness function. Only individuals with high fitness score are eligible for mating. Parents are selected from the eligible group and mated to form offsprings

(new solutions). To keep the size of the population constant, individuals with low fitness scores are eliminated. In this way new generations of individuals (solutions) are obtained by exhaustively mating parent solutions in the previous generation. This process is continued till an optimal solution is obtained. In their approach, the width of each motif is assumed to be fixed at 8. The initial parent solution set is created by randomly placing a window of size 8 on each sequence. Offspring solutions are created from the parent by using up to three different operations. The number of operations to be used is again determined randomly. One operation is to randomly select a window on a parent solution and slide it to the left or right (direction selected randomly) by random number of positions. Another operation is to randomly select points of recombination between two parents, following which the parts of the parents are interchanged at each recombination point. The last operation is to select a parent randomly and move the window to a location such that $G+C$ percentage of that area is greater than equal to the average $G+C$ percentage (percentage of nucleotides in the sequence that is Guanine or Cytosine) of the rest of the sequences. Once offspring solutions are added to the population, the fitness of a solution is computed using a fitness function. The fitness function is a weighted sum of the overall similarity score and overall complexity score. Based on the fitness score, the "fit" solutions are retained while the rest are discarded. The whole process is continued until a local optimum is reached.

Mendes et al. [22] present MUSA, a parameter-free algorithm for identifying motifs based on biclustering. Biclustering is the process of simultaneous clustering of rows and columns of the matrix representing the entire dataset. It allows to identify significant sub-matrices within the data matrix [10]. The paper discusses the concept of complex motifs as a composition of smaller simpler motifs required to be present in a certain percentage (quorum) of the given set of sequences. They define the matrix

of co-occurrences as a symmetrical matrix indexed by subsequences of fixed length denoting the abundance of the occurrence of the two indices in the dataset. The matrix is used to identify the presence of complex patterns. A biclustering approach on the matrix of co-occurrence for that purpose is used. Starting with a single element of the matrix of co-occurrence, the matrix is expanded till a specified score is reached.

Xia [33] investigates the use of a position weight matrix (PWM) and Gibbs sampler methods for motif characterization and prediction. A PWM is a probabilistic method of representing motifs. A PWM for a motif of length m may be defined as a $4 \times m$ matrix, where each row represents a nucleotide(row-ordering: A, C, G, T) and each column represents a position in the motif. In this representation $PWM_{ij} = log_2(p_{ij}/b_i)$ where p_{ij} = probability of symbol i in position j and b_i = background probability of symbol i. Consider the string 'ACTG'. Let the background probability of each symbol be 0.25. Then $b_A = b_C = b_G = b_C = 0.25$. Also in ACTG, $p_{A1} = 1$, $p_{C2} = 1$, $p_{G4} = 1$, $p_{T3} = 1$. So the PWM representation of ACTG is:

$$\begin{vmatrix} log_2(1/.25) & log_2(0) & log_2(0) & log_2(0) \\ log_2(0) & log_2(1/.25) & log_2(0) & log_2(0) \\ log_2(0) & log_2(0) & log_2(0) & log_2(1/.25) \\ log_2(0) & log_2(0) & log_2(1/.25) & log_2(0) \end{vmatrix}$$

Typically pseudo-counts are added while computing PWM so that there are no undefined ($log_2(0)$) values. Position Frequency Matrix (PFM) is same as PWM, with only one difference. Instead of logarithmic value, it contains the number of occurrences of

symbols in different positions. So the PFM representation of ACTG is:

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

Assume there are N sequences with exactly one motif occurrence in each sequence. Let n_i represent the starting position of the motif in the ith sequence. The algorithm has two phases. The first phase is the initialization phase, where random n_i values are assumed and different nucleotide frequencies are computed. The second phase is the predictive update phase. In this phase, each of the N sequences is processed in a random order to improve its corresponding n_i value by the use of position weight matrix. Once a local solution of n_is is obtained, the whole process is repeated until a local optimum is reached. This local optimum, n_i values, together with the nucleotide frequencies describes the motif and its position in each sequence.

Bailey et al. [7] introduce the MEME algorithm. It is based on the Expectation Maximization concept [24]. Expectation Maximization is an iterative method of calculating maximum likelihood estimates of statistical models with hidden/latent variables. It alternates between two steps, expectation step (E step) and maximization step (M step). During E step, it creates a function (g) which calculates the log likelihood using the current estimate of the model parameters. In the M step, it calculates new parameters which maximizes the function g. This is continued till convergence is reached [13]. In this paper the authors test different starting points for estimating a probabilistic model of a motif, and chooses the model with the highest log likelihood. Once the model of a motif has been output, it removes all traces of the

motif from the input data and looks for new motifs. MEME has a time complexity of $O(n^2)$,where n is the total number of characters in the provided dataset.

Simcha et al. [29] discuss the potential limitations of different algorithms for *de novo* motif discovery. They introduce a logistic regression based method, the LR algorithm as well as a generalized version of LR called ALR. The authors design certain tests using both synthetic and real data to evaluate the performance of motif discovery algorithms as classifiers. They show that all the algorithms tested gave poor results. It leads to the conclusion that improvements are needed in the existing computational techniques for motif discovery. Integration of biological knowledge coupled with high-throughput data would increase the accuracy of the existing algorithms.

Chapter 4

Background

In this chapter, we briefly present the idea of string representation of motifs and introduce Probabilistic Suffix Trees. Ron et al. [27] showed that PSTs are a suitable statistical approach for modeling biological sequences. It is based on the idea that for a sequence, the probability distribution of the next symbol can be accurately approximated by just the last L symbols of that subsequence.

4.1 Motif representation

In our method we use string representation of motifs. This is letter representations of a motif. In its simplest form it can be an exact sequence of letters (nucleotides) describing the motif. For example ATGCAAA describes a length-7 motif whose letters are A, T, G, C, followed by 3 A's in that order. Or it can be used to describe a general pattern. For instance, AAAnnTGC represents a length-8 motif which always starts with AAA and ends with TGC. The middle two positions can be anything.

We use an 11-letter alphabet Σ = {A, T, G, C, r, y, s, w, m, k, n} in the syntax of the string representation of motifs. The last seven symbols are degenerate symbols that represent multiple nucleotides. Table 4.1 shows the multiple representations of the degenerate symbols. The degenerate symbols are useful in the event we represent

Table 4.1: Degenerate symbols and their meaning

Symbol	Nucleotides
r	A or G
y	C or T
s	C or G
w	A or T
m	A or C
k	G or T
n	A or C or G or T

a motif whose instances vary in some positions. For instance AArTG represents that both instances of this motif, AAGTG and AAATG.

4.2 Probabilistic Suffix Trees

In this section we review the concept of the Probabilistic Suffix Tree. Probabilistic Suffix Trees (PST) were introduced by Ron et al. [27]. A PST defined over an alphabet Σ is a non-empty tree where each node can have a maximum out-degree of $|\Sigma|$. The edges from a node are labeled by symbols from Σ such that no two edges have the same label. Each node is labeled by a string x and a probability distribution Γ, where x is the string obtained by traversing from the current node to the root. Γ denotes the probability distribution of the different symbols of Σ after observing string x.

Terminology related to PSTs

Before going into the details of the construction of a PST we present relevant definitions. Let $P(x)$ be the probability of observing a string x in a given dataset of strings S. It is defined as

$$P(x) = N_x/N \ ,$$

where N_x = Number of occurrences of x in S (overlaps included) and N = Total number of (overlapping) occurrences of any string of length $|x|$ in S. Let $P(\alpha \mid x)$ be the conditional probability of observing a symbol α after string x in S. It is defined as

$$P(\alpha \mid x) = N_{x\alpha}/N_{x*} \ ,$$

where $N_{x\alpha}$ denotes the number of occurrences of string "$x\alpha$" in S (overlaps included) and N_{x*} denotes the number of occurrences of string x followed by any symbol in S.

For instance consider the string AAATGCGTAAA. In this string, AAA occurs 2 times while the total number of length-3 strings that can be found is 9. Following the above definition $P(AAA) = 2/9$. To illustrate $P(T \mid AA)$: $N_{AAT} = 1$ as T occurs after AA only once while $N_{AA*} = 3$ as the number of length-3 strings starting with AA is 3. So $P(T \mid AA) = 1/3$.

In a PST, if the label of node is $x_1x_2\cdots x_{|x|}$, then the label of its parent would be $x_2\cdots x_{|x|}$.

For a string $x = x_1x_2\cdots x_{|x|}$, we define the function Υ as $\Upsilon(x) = x_2\cdots x_{|x|}$. So in a PST, $\Upsilon(node)$ gives the label of its parent.

Building a PST

Five parameters, L, ρ, δ, κ and ζ, are associated with the construction of a PST T_S (constructed from dataset S), where L is the depth of the tree, ρ the minimum probability for any string to be considered as a node in the tree, δ, the difference in the prediction abilities of a child and its parent, κ, a factor used to smooth the probabilities of the symbols at any node, and finally, ζ, which together with κ determines the threshold of the conditional appearance of a symbol. Let D denote the strings that will be used to construct the PST. The pseudocode for constructing T_S is given

as Algorithm 1. It is adapted from Bejerano [9] and Ron [27].

Given a dataset S, a PST T_S is constructed by Algorithm 1 as follows: Algorithm 1 starts with the development of the first level of nodes in the PST. The symbols in the alphabet become the first level of nodes. Each string x associated with a node is examined to check if it has potential for extension. The extented strings are obtained by appending the current node label (x) to each symbol (σ) from the alphabet (Σ). If the probability of this new string σx exceeds ρ, σx is added to the list of strings to be checked. For a string x to be added as a node to the tree, it needs to satisfy two conditions.

- The probability that some symbol σ from Σ can appear after x, exceeds some threshold.
- The predictive ability of the node is greater than its parent.

If a parent and a child have same predictive powers, then there is no point in adding the child to the tree. Once a node is added to the tree, the probability distribution of each symbol of the alphabet (Σ) at the node is calculated and stored. This whole process is repeated until the tree reaches the specified length or runs out of iterations.

Example

Consider the set of strings $S = \{$AAATGCTT, ACTATGCA, ATGCATGC, TTGCATGC$\}$. Dataset S is given to the Build-PST-Tree algorithm with the following parameters: $L = 3$, $\rho = 0.0001$, $\delta = 1.05$, $\kappa = 0.001$, and $\zeta = 0$. Figure 4.1 show the different stages of the development of the tree.

In the graphical representation, each node is labeled with the string associated with it and all outgoing edges from a node are labeled with the probability of the

Algorithm 1 Build-PST-Tree (L, ρ, δ, κ, ζ)

Initialize T_S to a single root node and $D \leftarrow \{\sigma \in \Sigma : P(\sigma) \leq \rho\}$.
/* D initially starts off with single length strings (which are the symbols in alphabet Σ), whose probability should be less than ρ */

while $D \neq \emptyset$ **do**
Select any $x \in D$

Remove x from D

If there exists a symbol $\sigma \in \Sigma$ such that

$$P(\sigma \mid x) \geq (1 + \zeta)\kappa \tag{4.1}$$

/* constraint on the probability that symbol σ
appears after string x */
and

$$\frac{P(\sigma \mid x)}{P(\sigma \mid \Upsilon(x))} \geq \delta \text{ or } \frac{P(\sigma \mid x)}{P(\sigma \mid \Upsilon(x))} \leq (1/\delta) \tag{4.2}$$

/* comparing predictive powers of a node and its parent */

then add a node labeled x to T_S and all nodes on the path from the deepest node in T_S which is a suffix of x

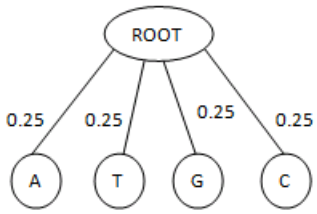
If $|x| < L$ and $\forall \varsigma \in \Sigma$ if $P(\varsigma x) \geq \rho$ add ςx to D /* calculating the probability distribution of the symbols of Σ at the newly added node */
end while

For each node labeled x in T_S, the probability distribution of the symbols is set as follows:

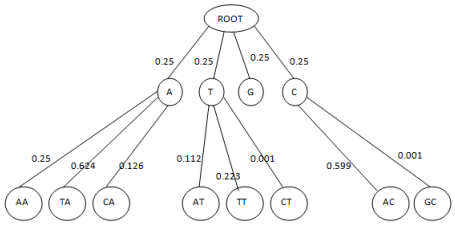
$$\Gamma(\sigma) = (1 - |\Sigma|\kappa) \times P(\sigma \mid x) + \kappa \tag{4.3}$$

next symbol. For instance, in Figure 4.1(b), the probability of symbol T appearing after node labeled A is 0.624.

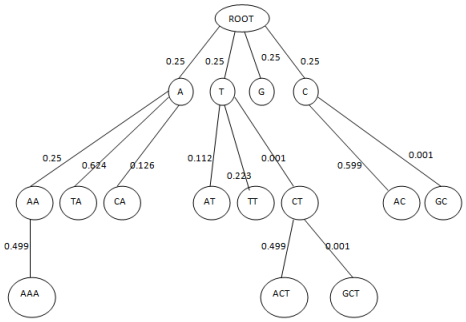
The initial stages of the development of the PST is described below. D is a set of strings that may potentially become nodes in the tree, if the requisite conditions are satisfied. Initially there is only the root node and D is initialized to {A, T, G, C}. String A is removed from D and tested for the two conditions described earlier. Since both conditions $P(A \mid A) = 0.25 > 0.001 = (1 + \zeta)\kappa$ and $P(A \mid A)/P(A) = 0.888 < 0.952 = (1/\delta)$ are satisfied, a node labeled A is added to the root. Also, since $P(AA) = 0.035 > \rho = 0.0001$, string AA is added to D. Based on $P(TA) = 0.035$ and $P(CA) = 0.107$ values, TA and CA are also added to D to be checked later by the algorithm. It should be noticed that since $P(GA) = 0$, GA is not added to D. Following this, the probability distribution of the symbols at node A are calculated using equation 4.3 in Algorithm 1 and stored. Then, T is removed from D and similar procedure is followed. Let us look at what happens when AA is removed from D. $P(A \mid AA) = 0.5 > (1 + \zeta)\kappa$ and $P(A \mid AA)/P(A \mid A) = 2.0 > \delta$. So the node AA is added to the tree as a child of node A. The probability distribution at node AA is calculated and stored. In this way D is checked and updated in each iteration until D becomes empty. Figures 4.1(a)-4.1(c) show the stage-by-stage development of the PST. Figure 4.1(a) shows the PST with only root node and its four child nodes. Figure 4.1(b) shows further development of the tree. Node A has three children labeled AA, TA and CA. Node T also has three children, node G has no children and node C has two, AC and GC. Finally, Figure 4.1(c) shows the final stage of the PST when the specified length $L = 3$ is reached. There are only three nodes labeled with length 3 strings; AAA, ACT and GCT.



(a) PST when len=1



(b) PST when len=2



(c) PST when len=3

Figure 4.1: Graphical representation of PST at different stages of construction

Scoring using PST

A PST can be used to compute the probability of occurrence of a string x. Let $P_{T_S}(x)$ denote the probability of x to be generated by the PST T_S. This probability is calculated as the product of individual probability of each letter in x. Let x_i be the ith letter in x and $x_{1..i-1}$ be the prefix of x that ends in position $i-1$. The probability of x_i is obtained by searching for the longest suffix of $x_{1..i-1}$ that appears in the PST, and using the probability distribution value of letter x_i at that node.

For instance, given the PST T_S from the previous example,

$$P_{T_S}(ATGC) = P_{T_S}(A) \times P_{T_S}(T \mid A) \times P_{T_S}(G \mid AT) \times P_{T_S}(C \mid ATG).$$

$P_{T_S}(A)$ is the probability of symbol A at the root node which is 0.25. For $P_{T_S}(T \mid A)$, the deepest suffix of A which appears in the tree is A. So $P_{T_S}(T|A)$ is the probability of T at node A which is 0.6235. Similarly $P_{T_S}(G \mid AT)$ is the probability of symbol G at node AT (since node AT appears in the tree) and is equal to 0.997. Finally for $P_{T_S}(C \mid ATG)$, since the deepest suffix of ATG is G, it is equal to the probability of symbol C at the node G which is 0.997. So

$$P_{T_S}(ATGC) = 0.25 \times 0.6235 \times 0.997 \times 0.997 = 0.1549.$$

In the context of the PST approach, we define the score of a string x (as measured by a PST), $\Psi(x) = P_{PST}(x)$.

Now we consider a string which is not a motif. Let a string y of length m be not a motif. Then each symbol in y is independent of others in the string. That is a symbol in any position in y has the same absolute probability as the symbol in the dataset. Thus the probability of any symbol y_i in $y = P_{T_S}(y_i)$. And the score of y as measured by PST T_S is

$$\Psi(y) = \prod_i P_{T_S}(y_i).$$

For instance, if the string y = TGAC is not a motif, then the score of y is $\Psi(y) = P_{PST}(y) = P_{T_S}(T) \times P_{T_S}(G) \times P_{T_S}(A) \times P_{T_S}(C)$. The probability of each symbol is obtained from the dataset. This is our null model.

Chapter 5

Finding motifs of a fixed length

In this chapter we propose an approach using Probabilistic Suffix Trees to find length-m motifs in a given set of DNA sequences.

5.1 Proposed motif finding method

Parameters

Our motif finding process involves a total of seven parameters. Five parameters (L, ρ, δ, κ, and ζ) are used for building the tree. These parameters can be set by the user.

We use two more parameters (Θ and e_{cut}), which are used for filtering false positives. Θ is the score of a string defined by our null model. It is used as a threshold to distinguish between non-motif strings and potential motifs. The parameter e_{cut} is a cut-off criterion used to eliminate potential motifs with high e-value. It is set as per the user's discretion. Table 5.1 lists the parameters and their values. These parameters can be empirically optimized for a dataset under consideration.

Table 5.1: List of parameters used in PST tree building and the search process

Purpose	Parameter	Value
PST-building	L	20
	ρ	0.00005
	δ	0.5
	κ	0.001
	ζ	0
Searching and Filtering	e_{cut}	0.05

Approach

We use a PST T_S to identify the motifs in S. We start with building T_S via Algorithm 1 for a given data set S. We consider any string x to be a candidate (i.e., potential) motif if its score as computed against T_S exceeds threshold Θ. The process is described as follows. Let L be the set of all leaves in the PST. Let L_μ be the set of leaves in L of size $\geq \mu$. We will use the leaves in L_μ in conjunction with a sliding window W to search for potential motifs. The size of window W is set to m, same as that of the motifs. If a leaf (string) ℓ is greater in length than m, then let $S_{m,\ell}$ be the set of m sized substrings of ℓ. We populate $S_{m,\ell}$ by simply dragging the window from the left end of ℓ to right, one letter at a time, to get all m-sized substrings of ℓ. Figure 5.1 shows how the sliding window is used with a bigger sized leaf. In the figure, the leaf is denoted in red color while the window is blue. The leaf is of size four and the window is of size three.

Then for each $\ell_i \in S_{m,\ell}$ we search for ℓ_i in each sequence $s \in S$. So if ℓ_i is found in s and its score $\Psi(\ell_i) > \Theta$, then we consider ℓ_i to be a candidate motif in S. If, however, $|\ell| < W$, we first find the location of ℓ in the sequence s. Then we place the window on that location and get all possible m-sized substrings of s, in which ℓ is a part. This is done by sliding the window left and right, all the while covering ℓ. Figure 5.2 shows the process of sliding the window. In the figure leaf is colored red

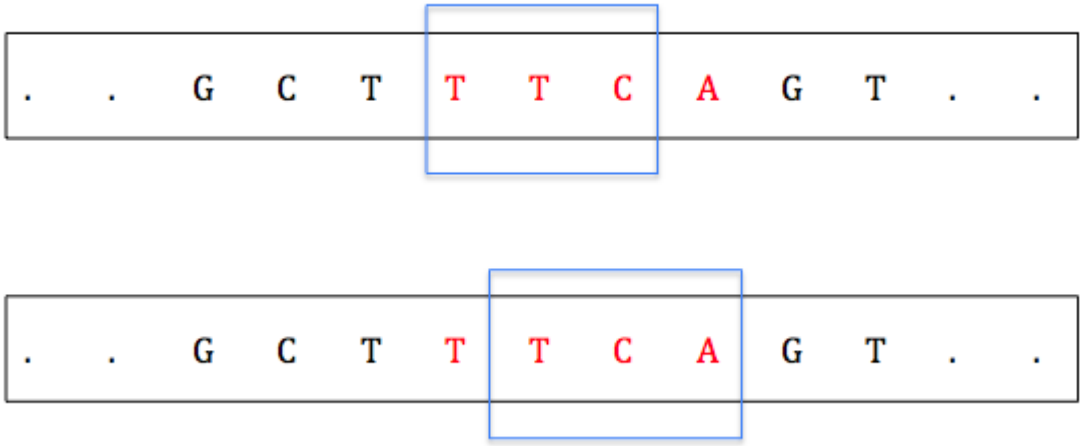


Figure 5.1: Smaller sliding window on a bigger leaf

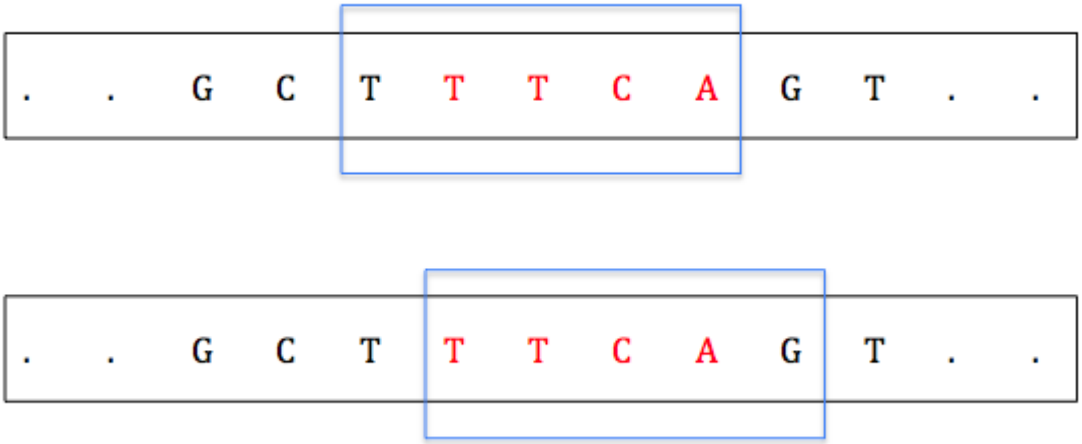


Figure 5.2: Bigger sliding window on a smaller leaf

while the window is blue. The leaf is of size four and the window is of size five. We then score each string thus obtained against T_S and compare their scores with the threshold Θ.

Once a set of candidate motifs CM is obtained, we calculate the statistical significance of each candidate. We do this by computing the e-value of its information

content as defined by Stormo and Hertz [17]. They define e-value of a candidate motif as the expected number of random strings in the entire dataset which have the same or more information content as the candidate. In their paper, Stormo and Hertz show how to calculate the p-value of the information content of any motif represented in alignment matrix format, and then scale it, based on the size of the dataset to get the e-value. Alignment matrix is simply a matrix that contains the number of occurrences of each nucleotide at each position in an alignment. They define p-value of the information content (ic) of a given alignment matrix as the probability of finding an information content $\geq ic$ given the alignment matrix details (width of the motif, and the matrix contents). Information content ic of an alignment matrix is defined as follows:

$$ic = \sum_{j=1}^{W} \sum_{i=1}^{4} f_{ij} ln(f_{ij}/p_i)$$

where W is the width of the alignment, p_i is the background probability of symbol i and f_{ij} is the frequency of symbol i at position j. They show that they can accurately estimate the p-value using large-deviation statistics. We represent each candidate motif in alignment matrix form and use their method to calculate the p-value and subsequently the e-value of that candidate motif. While calculation the e-value, we assume each sequence can contribute zero or more times to a motif alignment matrix and use the appropriate formula for calculating the dataset size. Once the e-values of all the candidates have been computed, we use a cut-off value e_{cut} to filter the candidates in CM. Those with e-value $\leq e_{cut}$ are retained while the rest are discarded. Figure 5.3 describes our proposed approach for motif finding using PST.

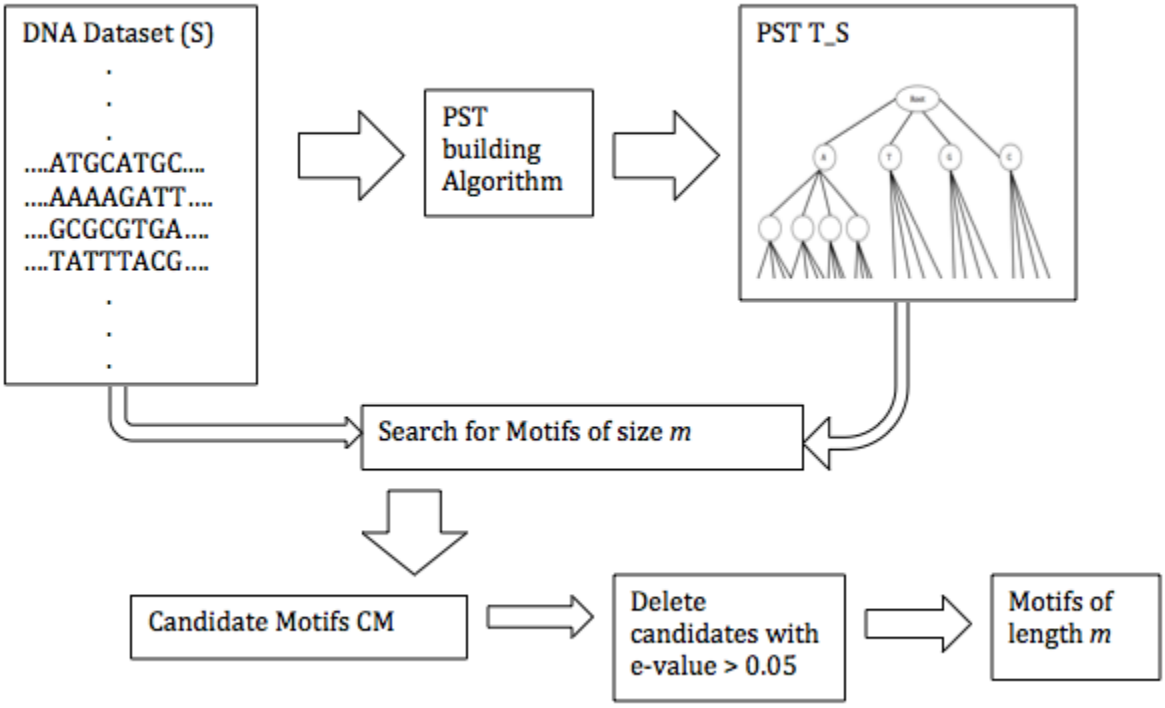


Figure 5.3: PST based approach for finding motifs of length m from dataset S

5.2 Experiments

5.2.1 Experiment 1 on Synthetic Data

Objective

The objective of the experiment is to see how our PST based approach performs, in terms of the unbiased validation method suggested by Simche *et al.* [29].

Dataset

Four synthetic datasets are constructed: C_1, C_2, C_3 and C_4. Each dataset is built around a single type of motif. That is, all sequences of dataset C_i have the motif m_i in them. Thus the presence of a particular motif should identify whether a sequence belongs to the corresponding dataset or not. In each sequence a single instance of the motif is placed at random. The sequences are generated in such a manner such that the probability distribution of the nucleotides in the dataset C_i is the same as that in the genomic sequences from the Saccharomyces Genome Database (SGD) [3]. The size of the planted motifs is 8 nucleotides. Each sequence is 1000 nucleotides long. Each dataset has 1000 sequences. The motifs are completely distinct from each other, that is, each has a Hamming distance of 8 from each other. For simplicity's sake, the four motifs used are 'AAAAAAAA', 'TTTTTTTT', 'GGGGGGGG' and 'CCCCCCCC'. So C_1 contains the motif 'AAAAAAAA', C_2 contains the motif 'TTTTTTTT', C_3 contains the motif 'GGGGGGGG', and C_4 contains the motif 'CCCCCCCC'. This is done so as to facilitate easy distinction of the clusters among themselves. That is, it should be easy to distinguish the four clusters solely based on the presence or absence of the respective motifs. This experimental setup is adapted from the planted motif simulation scenario of Simche [29]. Four test datasets DC_1, DC_2, DC_3 and DC_4 are also constructed. Each dataset DC_i contains 200 sequences; 100 sequences from C_i and 100 from the other three datasets combined.

Methodology

A PST T_{C_i} is built on C_i and used to identify motifs in C_i as shown in Figure 5.3. The set of motifs returned is converted to the corresponding Position Frequency Matrix form (PFM_i). PFM_i is used to compute the probability that a sequence s contains

the motif m_i which in turn indicating that probability that $s \in C_i$. The probability that a sequence s belongs to cluster C_i is given by:

$$\sum_{pos=1}^{(|s|-|m|+1)} \left[\prod_{j=pos}^{(pos+|m|-1)} \left(\frac{PFM_i[s_j, j-pos+1]}{NRM_i} \right) \right]$$

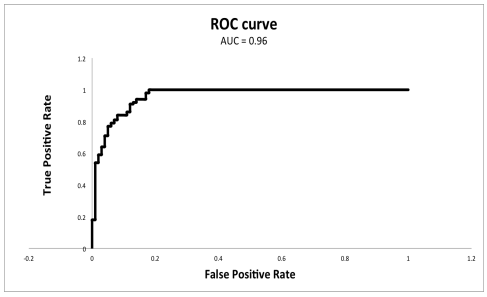
where

$$\prod_{j=pos}^{(pos+|m|-1)} (\frac{PFM_i[s_j, j-pos+1]}{NRM_i})$$

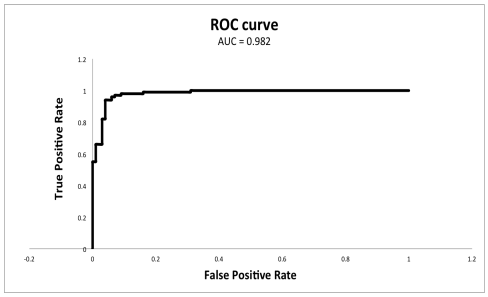
corresponds to the probability that motif represented by PFM_i is present in sequence s at position j. Since the exact location of the motif is not known, all possible positions $(|s| - |m| + 1)$ are accounted for by using the summation term. So the formula basically sums up the the probability that motif m_i is present in the position *pos* for all permissible values of *pos*. The PFM_i is thus used as a classifier for the sequences in DC_i and the performance is measured using area under ROC curve (AUC) [14]. A ROC curve is a graphical plot which evaluates the performance of a binary classifier (in this case PFM_i). The area under ROC quantifies the performance of the classifier. In the worst case the area is 0.5 indicating that the classifier is no better than a coin flip. In the best case scenario the value of the area is 1.0, which indicates a perfect classifier. This is repeated for all the four datasets C_1 through C_4.

Result

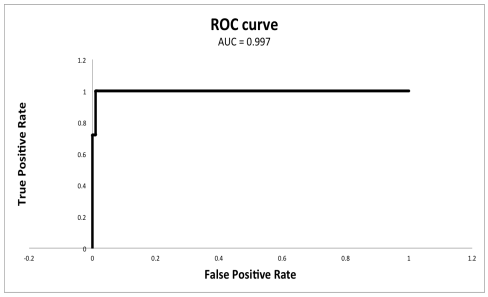
The average area under ROC curve returned by our PST method on the synthetic dataset is 0.983. This shows the quality of the motifs found by our PST based approach given a set of sequences. It is capable of finding motifs which can accurately classify the cluster they belong to from other clusters. This is a good validation of the performance of our PST-based method. Figure 5.4 shows an instance of the ROC curves for each of the four datasets.



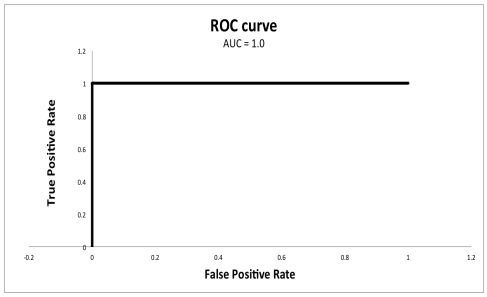
(a) ROC curve for dataset DC_1



(b) ROC curve for dataset DC_2



(c) ROC curve for dataset DC_3



(d) ROC curve for dataset DC_4

Figure 5.4: ROC curves for each of the four datasets

5.2.2 Experiment 2 on Synthetic Data

Experiment 2 is a repetition of Experiment 1 in the exact same manner, with different datasets. The motifs planted in the datasets are now real *Saccharomyces* motif strings taken from the JASPAR database (http://jaspar.genereg.net).

JASPAR is the largest open-access database of Position Frequency Matrix (PFM) profiles describing the binding sites of transcription factors from multiple species [21]. These are based on published experiments from various different sources. The profiles are manually curated. Putative binding patterns are confirmed by independent publications. We use JASPAR database for validation purposes.

Let the four constructed datasets be E_1, E_2, E_3 and E_4. The motifs used in this experiment are: 'TCCGCGGA', 'GTTACGAT', 'ACACGAAA' and 'CACTGCGA'. Let the four test datasets be DE_1, DE_2, DE_3 and DE_4. They are created in the same way DC_i was created from the C_is. The experiment is done in the exact manner as the previous one

Result

With the given set of motifs, our PST-based method performs poorly. The average area under ROC curve returned is 0.535. On closer examination we see, that although our method identifies the given motif in each cluster, it also discovers several false positives which results in the poor ROC performance.

5.2.3 Evaluation Metrics

We use the following metrics to evaluate the quality of predicted results by our approach when applied on real datasets. A predicted motif of size m bases is considered to be correct if it:

1. matches a true motif of size m.
2. is a proper substring of a true motif of size $> m$.
3. overlaps its last (first) $m-1$ bases with the first (last) $m-1$ bases of a true motif of size m.

The last case identifies string which have high overlap with a true motif. In many applications, the user might find such information useful.

We formally define correctness as follows:

$$correctness = \frac{\sum |x|/|\overline{x}|}{|X_m|} \quad \forall x \in X_m \quad ,$$

where X_m is the set of predicted motifs of length m, x is any motif from the set X_m, and $\overline{x}$ corresponds to the true motif which x identifies following any three of the cases mentioned above.

We define *coverage* as the fraction of same sized true motifs discovered by our approach:

$$coverage = \frac{|\widehat{X}_m|}{|Y_m|} \quad ,$$

where Y_m is the set of all true motifs of size m bases and $\widehat{X}_m$ is the set of all candidate motifs of size m which exactly matches some motif in Y_m.

To illustrate the definitions we give the following example. Let us assume in a dataset there are 10 true motifs of size 8 and 6 true motifs of size 12. Let our method predict 10 motifs of size 8. Upon comparison it is seen that 6 of our predicted motifs match perfectly with 6 true motifs and 2 are substrings of the larger true motifs. One motif overlaps with the the left end of a true size 8 motif by 7 base-pairs and one motif has no match. So $m = 8$, $|X_m| = 10$, $|\widehat{X}_m| = 6$ and $|Y_m| = 10$. Then by the

above definitions,

$$correctness = \frac{6 + 2 + (7/8)}{10} = 0.875$$

and

$$coverage = \frac{6}{10} = 0.6 \quad .$$

5.2.4 Experiment and Validation on Yeast dataset

Dataset

We downloaded genomic sequences from the Saccharomyces Genome Database (SGD) [3]. These sequences are upstream sequences of the orginal ATG for all Yeast genes except the ones marked dubious or pseudogene. We process the sequences to get exactly 1000 base pairs upstream for each gene. We also remove sequences that contain consecutive degenerate symbols for the sake of simplicity. After processing, we have our dataset D_{Sacc}, consisting 5917 sequences. We use PWM representations of 177 known *Saccharomyces* motifs from the JASPAR database for validation.

Experiment

We construct PST T_{Sacc} from the dataset D_{Sacc}. Once T_{Sacc} is obtained, we use it to predict motifs (M_{maybe}) for the whole dataset. We then calculate correctness and coverage of our method. We do this process for different values of motif size starting from 6 through 12 nucleotides. The whole process is repeated for different values of e_{cut} from 0.0 to 100. The objective of the experiment is to find out which value of e_{cut} gives the best combination of correctness and coverage.

Results

Table 5.2 lists the correctness and coverage values for different motif sizes and for different values of the parameter e_{cut}.

For any motif size, correctness = 1.0 implies that the motif discovered by our method is either a true motif or a substring of a larger true motif. A coverage = 1.0 implies that our method could discover some motif strings belonging to all the different PWMs of a particular size.

From this table, we see that correctness = 1.0 for all values of e_{cut} for sizes 6, 7, 8 and 9. However, the *coverage* values are very low; never exceeding 0.512. This means that it does not find all motifs of those sizes from the data set. For motif sizes of 10, 11 and 12, the correctness values decrease with increasing e_{cut}. However, it never drops below 0.9. On the other hand, coverage is much better than for the smaller sizes, remaining between 0.888 and 1.0. Inspecting the different values, we select $e_{cut} = 0.05$ to be the threshold for future use with unknown/new data.

We see for smaller sizes, we have perfect correctness values and pretty low coverage. However, for bigger sizes, both our correctness and coverage values are better than 0.96.

5.2.5 Comparison with MEME

We compare the performance of our method with respect to a popular motif finding software called MEME [5]. For objective evaluation of the proposed method, the dataset and motif length used is the same as used by MEME. We test the two methods on both synthetic and real data.

Table 5.2: Correctness and Coverage values for motifs of size 6 through 12 in Yeast dataset

EValue	Motif Size	Correctness	Coverage
0.001	6	1.0	0.1
0.001	7	1.0	0.1
0.001	8	1.0	0.326
0.001	9	1.0	0.434
0.001	10	0.998	0.8
0.001	11	0.979	0.888
0.001	12	0.915	1.0
0.005	6	1.0	0.1
0.005	7	1.0	0.1
0.005	8	1.0	0.326
0.005	9	1.0	0.478
0.005	10	0.998	0.9
0.005	11	0.979	0.888
0.005	12	0.913	1.0
0.01	6	1.0	0.1
0.01	7	1.0	0.1
0.01	8	1.0	0.326
0.01	9	1.0	0.478
0.01	10	0.998	0.9
0.01	11	0.979	0.888
0.01	12	0.909	1.0
0.05	6	1.0	0.1
0.05	7	1.0	0.1
0.05	8	1.0	0.346
0.05	9	1.0	0.521
0.05	10	0.998	0.9
0.05	11	0.978	0.888
0.05	12	0.905	1.0
0.1	6	1.0	0.1
0.1	7	1.0	0.1
0.1	8	1.0	0.346
0.1	9	1.0	0.521
0.1	10	0.998	0.9
0.1	11	0.977	0.888
0.1	12	0.902	1.0

Table 5.3: MEME vs our method on Synthetic data

Dataset	Method	Correctness	Coverage
C_1	MEME	1.0	1.0
C_1	Our method	0.892	1.0
E_1	MEME	1.0	1.0
E_1	Our method	0.186	1.0

Synthetic Data Experiments

We use the dataset C_1 with the single planted motif 'AAAAAAAA' in Section 5.2.1 for this experiment. We make both methods find all motifs of length 8 bases in C_1. We repeated the above experiment with dataset E_1 containing the planted motif 'TCCGCGGA' from Section 5.2.2. Table 5.3 shows the correctness and coverage values of both methods for both datasets. As we can see, both methods have perfect coverage, that is all motifs were found. For dataset C_1, MEME has a perfect correctness (1.0) while our method is less than 1. This indicates the presence of false positives. We found that the PST-based method considered strings such as 'TAAAAAAA', 'GAAAAAAA', 'CAAAAAAA' etc, as motifs which is why we got lower correctness value. In case of dataset E_1, our method has significantly low correctness value. An inspection of our predicted motifs, reveals that numerous predicted motifs overlapped the true planted motif by six out of eight bases at either end. This is the reason for the low correctness value of our method.

Small-sized motif discovery

First we compare peformance of our method with MEME for motifs of small size. The data set selected to be used consists of 1000 sequences randomly selected from the Yeast dataset used in the previous section. Both MEME and our method are operated on this dataset, to find ten motifs of size-6 bases. MEME lists 10 tentative motifs

Table 5.4: Performance of MEME and our method for finding motifs of size-6 bases

Method	Motifs found	Correctness	Coverage
MEME	0	0.0	0.0
Our Method	2	1.0	0.1

Table 5.5: Performance of MEME and our method for finding motifs of size-8 bases

Method	Motifs found	Correctness	Coverage
MEME	10	1.0	0.08
Our Method	115	1.0	0.285

but all of them have e-values of the order of 10^8 (as calculated by MEME). That indicates that these motifs are not statistically significant. Our method finds two statistically significant motifs with e-values < 0.05. Table 5.4 lists the performance of both methods.

Motif discovery accuracy

In this comparison, we try to see which approach performs better by finding more correct motifs from a given dataset. We use the same data set which is used for the small-sized motif discovery experiment. We operate our method and the MEME software to find all motifs of size 8 bases from the provided dataset. MEME took a long time and we killed the process after 2 - 3 days. In that time, it only produced 10 significant motifs.Table 5.5 shows the performance of both methods. As Table 5.5 shows, our PST based approach gives more number of correct significant motifs than MEME from the same dataset in a much shorted amount of time. So we see, that in synthetic datasets, MEME outperforms our method, whereas in real dataset, PST-based method is much better than MEME. This can attributed to the fact that in real datasets motif distribution is not as simple as we had in the synthetic ones.

Table 5.6: Motifs found by PST method on *Arabidopsis thaliana* datasets

Dataset	Motifs Found by Our method	Motifs which were matched by Tomtom
A_{big}	910	906

5.2.6 Experiment on Arabidopsis dataset

Dataset

Experiment

We construct PST, T_{Ara} from the processed dataset Ara_{big}. Once T_{Ara} is obtained, we use it to predict motifs M_{ara} of length 8 bases from Ara_{big}. For validation purpose, we use Tomtom Motif Comparison Tool [16] to see if the motifs in M_{ara} match to any entries in the JASPAR AradibopsisDAPv1 database. Tomtom is a motif comparison tool that compares given query motifs to databases of known motifs (target). It is a tool for quantifying the similarity between query motif and target motifs. It lists for each query, a list of target motifs ranked in ascending order by e-value of the match. That is, the first match is the best one. We only consider the best match (lowest e-value), whenever we use Tomtom. Default values are used for the Tomtom search. In this case, we want to see, if the motifs predicted by our PST-based method match with publicly known Arabidopsis transcription factor binding sites.

Results

Table 5.6 lists the results for the above experiment. Dataset Ara_{big} returned 910 significant size-8 motifs. We see that Tomtom did not find any match for four out of 910 (0.004%) motifs. These 4 unmatched ones, might indicate some novel, yet to be identified motifs in the *A. thaliana* dataset.

5.3 Discussion

From our experimental results we find that our method gives good correctness values for different sized motifs. For motif sizes ≥ 10 bases, the coverage is nearly perfect. To inspect the low coverage value for smaller sized motifs, we checked the different candidate motifs before the filtration by e_{cut} step. It appears that a majority of the candidates get filtered due to their high e-value that otherwise could have improved the coverage. There are two possible explanations for this. That is, our assumption that a candidate motif has to be rare, i.e., low e-value to be a true motif may be too stringent. The other is we need to use a less stringent form of e-value calculation for establishing the statistical significance of our discovered motifs. MEME uses a form of e-value calculation method similar to ours, which is probably why they also had low discovery rate for small sized motifs as shown in the previous section. We further check and observe that many of the candidate motifs which have high e-values appear as substrings of valid motifs (e-value ≤ 0.05) of higher sizes. Our experiments on the *A. thaliana* dataset reveal our PST-method can operate on datasets of size approx. 33,000 sequences, with most ($\approx 99\%$) of the motifs returned matching to known *Arabidopsis* motifs.

Chapter 6

Finding motifs of a range of lengths

In this chapter, we look at the problem of finding all motifs between sizes m_1 and m_n (where $m_1 \leq m_n$) from a given dataset S of DNA sequences.

6.1 Method

The following steps describe our approach to solving this problem.

1. Use the method described in section 5.1 to find motifs of sizes m_1 through m_n.
2. Remove motifs of size m_i if it is a substring of any motif of size m_j where $m_i \in \{m_1, \ldots, m_n\}$ and $m_j \in \{m_{i+1}, \ldots, m_n\}$.

The logic behind step 2 is as follows. If a string appears in both a small and larger motif list, then it is probably part of a bigger motif which remains conserved. So it makes sense to report it only once as a part of the bigger motif. Also, reporting it twice will falsely boost the performance measure of the approach.

Step 2 will affect the coverage value for any motif size. This is because the number of strings reported for smaller motifs are likely to decrease. For a sanity check we used our Yeast data set to find motifs form size 7 to 12. The results are detailed in

Table 6.1: Correctness and Coverage values for motifs of size 6 through 12 in Yeast dataset for e-value = 0.05

Motif Size	Correctness	Coverage
8	1.0	0.041
9	1.0	0.043
10	1.0	0.3
11	1.0	0.556
12	0.904	1.0

Table 6.1. As expected, the coverage dropped for each size less than 12. However the correctness increased for sizes 10 and 11.

6.2 Application

6.2.1 Motivation

Dr. Steven Harris's lab at UNL uses black yeast *Exophiala dermatitidis* as their model system to investigate the molecular basis of physiological and morphological traits that allow these organisms to survive extreme environments. The accumulation of pigments, melanin and carotenoids likely play a key role in the stress tolerance of *E. dermatitidis*. CDC42 and Rac1 are molecular GTPases which are conserved across eukaryotes and involved in a number of signaling mechanisms. Molecular genetics studies using these GTPases in *E. dermatitidis* revealed the involvement of carotenoid synthesis pathway. To determine whether the deletion of cdc42 reveals the involvement of other pathways besides the carotenoid synthesis pathway, they performed RNAseq analysis on *Exophiala dermatitidis* and three albino mutants. Transcriptomics analyses was done with Pvalue 0.05. Let the three mutants be called C, G and R. The C mutant has CDC42 gene deletion. We use the upstream 1000 base DNA sequences of three albino mutants (from BROAD [1]) to find over-represented

motifs in each mutant. We look for motifs of size 7 to 9 bases. Let mC, mG and mR represent the sets of motifs for Cmutant, Gmutant and Rmutant, respectively. From these three sets, we categorize the motifs into the following different groups.

1. Motifs common to all three strains. Let them be represented by $mCGR$.
2. Motifs specific to Cstrain and not present in mR and mG; represented by mC_CGR.
3. Motifs common to Gstrain and Rstrain but not present in Cstrain. Let mRG represent that set.

The differential gene expression analysis using DEseq2 revealed a set of upregulated and downregulated genes in each of the mutants. Our objective is to identify DNA Motifs which can lead to candidate transcription factors. In turn these transcription factors could be part of the regulatory mechanisms involved in different pathways which are part of the transcriptomic analysis. To this end, we categorize the motifs into the three groups mentioned above. We hypothesize that motifs in mC CGR would help us in identifying the transcription factors specific to the effect of cdc42 deletion. In both Gmutant and Rmutant, Rac1 is constitutively expressed and over expressed, respectively. So mRG will indicate transcription factors which affect pathways in conjunction with cdc42. Finally motifs in mCGR will reveal common pathways between all three mutants irrespective of the absence/presence of cdc42.

The motifs in the different groups thus obtained are then compared against public databases to see, it they match any known transcription factors. This is done using the Tomtom Motif Comparison Tool [16]. The JASPAR CORE 2016 database is used for matching the motifs. Once a tentative list of transcription factors is obtained from

Table 6.2: Motifs for the three mutant strains of *E. dermatiditis.*

	Motif Size 8	Motif Size 9
mC	13	364
mG	12	170
mR	14	238
$mCGR$	1	123
mRG	1	9
mC_CGR	10	154

the Tomtom match results (based on available literature), these will be experimentally verified in the lab.

6.2.2 Results

Table 6.2 lists the number of motifs of different sizes found in the three strains as well as the different groups of motifs we are interested in. All these motifs are significant (e-value ≤ 0.05). No separate motifs of size 7 were found, i.e., any motifs of size 7 must have shown to be substrings of size 8 or size 9. So those are not included in Table 6.2.

In Table 6.3 we list the five motifs of highest occurrence in sets $mCGR$, mRG and mC_CGR for the two sizes.

The best match for each motif in Table 6.3 as given by Tomtom is listed in Table 6.4. We present the match which has the lowest e-value for each motif. As we can see, all except one motif ('CCAGTGGTG' in mRG) found matches to some transcription factor or the other. Also in some cases different motifs matched with the same transcription factor.

Table 6.5 shows whether every motif in the groups $mCGR$, mRG and mC_CGR got a match with TomTom. It also shows the total number of distinct transcription factors found by Tomtom for each group. For instance the motifs of size 9 in group

Table 6.3: Five of the highest occurring motifs

	Motif Size 8	Motif Size 9
mCGR	CATCAACA	TACCTAGGT
		ACCTAGGTA
		CAACTTGAA
		TTCAAGTTG
		GTTCAAGTT
mRG	GAGAAGGA	CCAGTGGTG
		CATTGTACA
		CCAGCCTCG
		GCACTGCAC
		GCAGTAGTA
mC_CGR	GAAGATGA	ACTTGAACC
	GAGGACAA	CTGCTGCTG
	TTTCTTTC	TTGAAGTTG
	TTGAAGTC	AGCCTCGTC
	CAGCAACA	CACTGGCAT

mCGR match with 50 different transcription factors. Table 6.6 lists the 5 most common transcription factor matches found by TomTom for each motif group. Only the best match for each motif returned by TomTom was taken into account for constructing Table 6.6. Table 6.7 contains the results of performing blastp of the transcription factors mentioned in Table 6.6. The idea is to verify if, a homologue of these TFs exist in *Exophiala* or not. These could be used a starting point for the wet lab (knock-out) experiments. Each of these TFs will be deleted systematically one by one and the resulting phenotype will be observed.

Till now, not much work has been on *Exophiala dermatitidis*. So the results obtained by analyzing these motifs will be a starting point for novel work. That fact that our PST-based approach can quickly identify motifs in the given datasets helps speeding up the entire process.

Table 6.4: Tomtom matches of the highest occurring motifs in *mCGR*, *mRG* and *mC_CGR*

Motif-Group	Motif	Matched Transcription Factor
mCGR	CATCAACA	HCM1
mCGR	TACCTAGGT	ceh-22
mCGR	ACCTAGGTA	T
mCGR	CAACTTGAA	ceh-22
mCGR	TTCAAGTTG	ceh-22
mCGR	GTTCAAGTT	Nr5a2
mRG	GAGAAGGA	Gata1
mRG	CCAGTGGTG	-
mRG	CATTGTACA	SOX10
mRG	CCAGCCTCG	NKX2-3
mRG	GCACTGCAC	SOX8
mRG	GCAGTAGTA	odd (C2H2 zinc finger factors)
mC_CGR	GAAGATGA	STAT1::STAT2
mC_CGR	GAGGACAA	ROX1
mC_CGR	TTTCTTTC	AZF1
mC_CGR	TTGAAGTC	BZIP60
mC_CGR	CAGCAACA	RAV1
mC_CGR	ACTTGAACC	ct
mC_CGR	CTGCTGCTG	odd (C2H2 zinc finger factors)
mC_CGR	TTGAAGTTG	ceh-22
mC_CGR	AGCCTCGTC	CREB3L1
mC_CGR	CACTGGCAT	SOX21

Table 6.5: TomTom match summary for all 3 motif groups

Motif-Group	Motif-Size	Motif Count	Motifs with no matches	Distinct Transcription Factor Matches
mCGR	8	1	0	1
mCGR	9	123	14	50
mRG	8	1	0	1
mRG	9	9	1	8
mC_CGR	8	10	0	7
mC_CGR	9	154	8	86

Table 6.6: Top5 Transcription factor matches by TomTom

Motif-Group	Motif-Size	Transcription Factor
mCGR	8	HCM1
mCGR	9	MYB3,NFIC,BHLH112,Nr5a2,AZF1
mRG	8	Gata1
mRG	9	Gata1,SPL4,ATHB-16,odd,SOX8
mC_CGR	8	ROX1,HCM1,AZF1,BZIP60,SPT2
mC_CGR	9	SOX8,ZNF263,ARR1,NFIC,SOX4

Table 6.7: *E. dermatiditis* homologues of Tomtom transcription factors

Motif-Group	Transcription Factor	Species	*E. dermatiditis* homologue	Identity (%)
mCGR	HCM1	*Saccharomyces cerevisiae*	XP_009157984.1	52
mCGR	MYB3	*Arabidopsis thaliana*	XP_009160990.1	35
mCGR	NFIC	*Homo sapiens*	-	0
mCGR	BHLH112	*Oryza sativa Japonica Group*	-	0
mCGR	Nr5a2	*Mus musculus*	-	0
mCGR	AZF1	*Saccharomyces cerevisiae*	XP_009157167.1	41
mRG	Gata1	*Mus musculus*	XP_009157917.1	63
mRG	SPL4	*Arabidopsis thaliana*	-	0
mRG	ATHB-16	*Arabidopsis thaliana*	XP_009160882.1	34
mRG	odd	*Drosophila melanogaster*	XP_009160737.1	36
mRG	SOX8	*Homo sapiens*	XP_009158160.1	34
mC_CGR	ROX1	*Saccharomyces cerevisiae*	XP_009158160.1	49
mC_CGR	BZIP60	*Arabidopsis thaliana*	XP_009159711.1	40
mC_CGR	SPT2	*Saccharomyces cerevisiae*	-	0
mC_CGR	ZNF263	*Homo sapiens*	XP_009160737.1	47
mC_CGR	ARR1	*Saccharomyces cerevisiae*	XP_009161509.1	40
mC_CGR	SOX4	*Homo sapiens*	XP_009158160.1	36

Chapter 7

Finding motifs in ChIP-seq data

ChIP-seq or chromatin immunoprecipitation sequencing [18, 25] is a great tool for identifying DNA-protein interaction sites across genomes. In a typical ChIP-seq experiment, the protein to be studied is first cross-linked to its DNA binding sites. The chromatin is then fragmented into smaller segments (150–500 nucleotides long). The DNA-protein complex is then immunoprecipitated and subsequently sequenced. The sequenced short fragments thus obtained are called *tags*. After this stage the number of tags available is huge (in millions). The tags are then mapped to a reference genome. Regions with high density of tag mapping are called *peaks*. The peaks indicate tentative protein binding sites in the genome. There are a large number of tools available for identifying genome regions with peaks. Once a set of tag enriched sites (peaks) are obtained, their sequences can then be fed to motif-finding algorithms to identify consensus motifs in those sites. These motifs will indicate the transcription factor binding sites for the genome and the protein under consideration.

In this chapter we show that our PST based approach can be used to identify motifs in the sequences of 'peaks' obtained from ChIP-seq experiment. These in turn will help in the identification of potential transcription factor binding sites. We validate and compare the performance of our method by comparing with the motifs

generated by MEME-ChIP tool [20].

7.1 Method

Let S denote the set of peak site sequences. We use the following steps to obtain motifs from S.

1. Each sequence in S is processed in the following way.

   - If a sequence consists of only Ns, it is removed from S.
   - If a sequence has regions of consecutive Ns, those regions are spliced out. The splice sites are remembered.

2. Use the method described in Section 6.1 to find motifs of sizes m_1 through m_n in the processed dataset S. If the position of any motif overlaps any splice site, that particular occurrence of the motif at that position is not considered while calculating its e-value.

3. Motifs obtained after Step 2 are further screened to account for the fact that motifs are expected to occur near the center of the peak sequences. We introduce two parameters f_{maj} and l_{cntr} for the screening process. l_{cntr} is used to specify the length of the region around the center of a peak sequence. $l_{cntr} = 100$ means a region of 50 nucleotides on both sides of the center of a peak sequence. We define f_{maj} as

$$f_{maj} = \frac{\text{number of occurences of the motif within the center region of all sequences}}{\text{total number of occurences of the motif in all sequences}}.$$

   So if the majority ($> f_{maj}$) of locations of any motif falls within l_{cntr} nucleotides around the center of the peak sequences, it is retained as a valid motif. Oth-

erwise it is excluded. The values of f_{maj} and l_{cntr} can be set by the user. As default values we use $f_{maj} = 0.5$ and $l_{cntr} = 100$.

7.2 Experiment

7.2.1 Dataset

We use the Klf1 dataset described in the MEME-ChIP paper [20]. We downloaded the dataset from the MEME-Suit website [2, 6]. It consists of peak regions obtained by performing KLF1 ChIP-seq in mouse [30]. The dataset consists of 904 peak sequences each 500 bases long. KLF1 is one of the few transcription factors which plays an important role in erythropoiesis, the production of erythrocytes [30]. KLF1 is the founding member of a family (KLF) of 17 transcription factors. They all play important roles in different biological processes [28].

7.2.2 Result

We used our method to identify motifs 6 to 30 nucleotides long from the processed Klf1 dataset. We then use these motifs to identify the transcription factors they match using Tomtom. Table 7.1 lists the number of significant motifs obtained by our method. In this table we summarize for each size, the total number of distinct motifs, the number of motifs with no matches to any transcription factor, and the number of distinct transcription factors matched. For the last one, we consider, only the best match (lowest e-value) returned by Tomtom. We only include those motif sizes that gave valid motifs.

Tomtom matched our motifs (from sizes 6 to 30 bases) to 26 different transcription factors. The JASPAR Core Database 2016 was used for the Tomtom matching. We

Table 7.1: Summary of Tomtom matches for motifs obtained from Klf1 dataset

Motif-Size	Motifs Found	Motifs with no matches	Distinct Transcription Factor Matches
8	1	0	1
9	14	0	12
10	16	0	6
11	33	0	9
12	15	0	4
13	5	0	4
14	4	0	3
19	1	0	1
28	1	0	1

Table 7.2: Transcription factors found by PST approach

Method	Transcription Factors
PST-approach	Klf1,Klf12,Klf4,ZNF263,Znf423,YRR1,EGR1,Tcf12,SREBF2,Gata1, KLF5,AFT2,SP8,EGR2,ESR2,NFIC::TLX1,RAP1,CTCF,SP1,KLF14, KLF16,SP3,RREB1,CREB3L1,SP2,DOF5.3

only consider the best/first match for each motif. The different transcription factors are provided in Table 7.2.

We also run MEME-ChIP [20] on the same Klf1 dataset. MEME-ChIP is a combination of several tools (MEME, DREME [4] and CentriMo [8]). While MEME and DREME are motif discovery tools, CentriMo searches for known motifs in the given dataset. DREME finds short, fixed length motifs and is a much faster tool than MEME. Unlike MEME, DREME uses regular expression to find motifs. Once a motif is found, it is removed from the dataset, and the next most significant motif is searched. The default settings and the JASPAR Core Database 2016 were used as parameters for running the MEME-ChIP suit. MEME-ChIP returned 44 transcription factors which it found enriched in the Klf1 data. These are listed in Table 7.3. It should be noted all 44 transcription factors identified were obtained by CentriMo, not by the motif discovery tools. The motifs discovered by MEME and DREME did not find any matches with Tomtom.

Table 7.3: Transcription factors found by MEME-ChIP

Method	Transcription Factors
MEME-ChIP	Klf1,Klf12,Klf4,GATA2,KLF5,Gata1,SP3GAT1,GATA5, GATA1::TAL1,KLF14,GZF3,DAL80,AFT2,Gata4,GLN3,KLF16,SP8, GATA3,hkb,elt-3,TBX5,Optix,TBX15,MGA,KLF13,TBX4,SP4,EGR4, SP1,btd,MET32,Hes1,TCP19,Bhlha15,TCP20,ARALYDRAFT_493022, MET31,SP2,hlh-1,ARR18,YGR067C,ZNF740,Six4

The whole idea of performing a ChIP-seq experiment is to identify the relevant binding sites on the sequences, co-regulatory transcription factors and novel motifs. Tomtom gave multiple matches for each motif, arranged in decreasing order of their importance. That is, the first match is the best match, followed by the second and so on. We would like to see how many motifs had Klf1 as its best match and how many had Klf1 as a match in general (not best). To this end we list for each motif size, the number of motifs that had Klf1 as the best match and those that had Klf1 as one of its several matches (other than first) in Table 7.4. For instance, in Table 7.4 we see that for motif of size 10, Tomtom gave matches for 16 motifs, of which 8 had Klf1 as the best match, 6 had Klf1 as a non best match and 2 didn't have Klf1 in any of their matches. It can be observed that motifs of sizes 10 to 12 have higher percentage of Klf1 hits as their best hits. So our PST method was capable of finding motifs which matched Klf1. It also found matches to other members of KLF family such as KLFs 4,5,12,14,16. This can be attributed to the fact that KLF family has a highly conserved DNA-binding domain [28]. In the Tomtom results it can be seen, if any motif had KLF 4, 5, 12, 14 or 16 as the first match, KLF1 also appeared as its match (downstream).

Any transcription factors (TFs) other than KLF1 found by ChIP-Seq data, indicate that those TFs might related to KLF1 in some way (co-regulated for example). Gata1 is one the TFs found by our method. Gata1 is also known to play a critical

Table 7.4: Klf1 transcription factor count from Tomtom results

Motif-Size	Motifs Matched	Klf1 as best match	Klf1 not as best match
8	1	0	0
9	14	0	5
10	16	8	6
11	33	20	12
12	15	11	4
13	5	0	5
14	4	1	3
19	1	1	0
28	1	1	0

role in erythropoiesis [30].

There are 12 common TFs in the results of the PST-based method and MEME-ChIP. These are Klf4, Gata1, KLF5, AFT2, SP8, Klf1, Klf12, SP1, KLF14, KLF16, SP3 and SP2. The rest 14 transcription factors should be verified to see if they are in some way related to Klf1 or not.

Thus we saw that our PST-based approach, being a *de novo* motif discovery tool was robust enough to be applied to ChIP-Seq peak sequences and find motifs in them. It could identify binding sites for Klf1 from the ChIP-Seq peak sequences. It also discovered that the dataset had enriched regions for 20 other known motifs.

The sensitivity of our method can be controlled by the user by regulating the parameters f_{maj} and l_{cntr}. If the value of l_{cntr} is increased (decreased), the center region increases (decreases), which in turn increases (decreases) the number of motifs returned by our method. If the value of f_{maj} is increased then lesser motifs are selected as valid motifs and also increases the sensitivity towards matching Klf1 transcription factor as the best choice. That is, number of motifs for other transcription factors discovered decreases.

Chapter 8

Conclusion and Future work

In this dissertation we proposed a PST-based approach for *de-novo* motif discovery from a set of DNA sequences. While this is a well studied problem, it does not have a complete and efficient solution yet. Unbiased validation of motifs obtained by computational methods is difficult as well. We showed that our method performs well in accurately finding motifs of different sizes. Compared to MEME, our method finds more correct motifs (when applied to real datasets). Our method also performed faster than MEME for datasets with more than 1000 sequences. Our experiments revealed that the PST method is capable of finding motifs in large datasets ($\approx 33,000$ sequences) with good accuracy. We applied our method to identify transcription factors from the *E. dermatiditis* genome. We further showed how to use our method to find protein binding sites in ChIP-Sequencing data. Our method was capable of finding over-expressed motifs in the ChIP-Seq data that matched with transcription factors. In the future we would like to address the following issues associated with our PST-based method.

1. Low motif discovery rates for small sized motifs. We would like to explore other options for e-value calculation.

2. Handling really big datasets ($>$ 33,000 sequences) efficiently in terms of memory

and time. The building of the PST takes a long time if the dataset is large.

3. Parallelize the PST approach. This should help with time management of dealing with big datasets.

4. In the case of ChIP-Seq data, make use of the actual fragmented reads in conjunction with the peak sequences to find motifs.

Bibliography

[1] Broad institute. `https://www.broadinstitute.org`.

[2] Meme-suite website. `http://meme-suite.org/doc/examples/memechip_example_output_files/Klf1.fna`.

[3] Saccharomyces genome database. `http://www.yeastgenome.org`.

[4] T. L. Bailey. Dreme: motif discovery in transcription factor chip-seq data. *Bioinformatics*, 27(12):1653–1659, 2011.

[5] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble. Meme suite: tools for motif discovery and searching. *Nucleic Acids Research*, 37(suppl 2):W202–W208, 2009.

[6] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble. Meme suite: tools for motif discovery and searching. *Nucleic acids research*, page gkp335, 2009.

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

[8] T. L. Bailey and P. Machanick. Inferring direct dna binding from chip-seq. *Nucleic acids research*, page gks433, 2012.

[9] G. Bejerano and G. Yona. Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17(1):23–43, 2001.

[10] Y. Cheng and G. M. Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[11] M. K. Das and H.-K. Dai. A survey of dna motif finding algorithms. *BMC bioinformatics*, 8(Suppl 7):S21, 2007.

[12] P. D'haeseleer. What are dna sequence motifs? *Nature biotechnology*, 24(4):423–425, 2006.

[13] C. B. Do and S. Batzoglou. What is the expectation maximization algorithm? *Nature biotechnology*, 26(8):897–899, 2008.

[14] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[15] G. B. Fogel, D. G. Weekes, G. Varga, E. R. Dow, H. B. Harlow, J. E. Onyia, and C. Su. Discovery of sequence motifs related to coexpression of genes using evolutionary computation. *Nucleic Acids Research*, 32(13):3826–3835, 2004.

[16] S. Gupta, J. A. Stamatoyannopoulos, T. L. Bailey, and W. S. Noble. Quantifying similarity between motifs. *Genome Biology*, 8(2):R24, 2007.

[17] G. Z. Hertz and G. D. Stormo. Identifying dna and protein patterns with statistically significant alignments of multiple sequences. 15(7):563–577, 1999.

[18] E. T. Liu, S. Pott, and M. Huss. Q&a: Chip-seq technologies and the study of gene regulation. *BMC biology*, 8(1):1, 2010.

[19] X. Liu, D. L. Brutlag, J. S. Liu, et al. Bioprospector: discovering conserved dna motifs in upstream regulatory regions of co-expressed genes. In *Pacific symposium on biocomputing*, volume 6, pages 127–138, 2001.

[20] P. Machanick and T. L. Bailey. Meme-chip: motif analysis of large dna datasets. *Bioinformatics*, 27(12):1696–1697, 2011.

[21] A. Mathelier, X. Zhao, A. W. Zhang, F. Parcy, R. Worsley-Hunt, D. J. Arenillas, S. Buchman, C.-y. Chen, A. Chou, H. Ienasescu, et al. Jaspar 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic acids research*, page gkt997, 2013.

[22] N. D. Mendes, A. C. Casimiro, P. M. Santos, I. Sá-Correia, A. L. Oliveira, and A. T. Freitas. Musa: a parameter free algorithm for the identification of biologically significant motifs. *Bioinformatics*, 22(24):2996–3002, 2006.

[23] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[24] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[25] P. J. Park. Chip–seq: advantages and challenges of a maturing technology. *Nature Reviews Genetics*, 10(10):669–680, 2009.

[26] P. A. Pevzner, S.-H. Sze, et al. Combinatorial approaches to finding subtle signals in dna sequences. In *ISMB*, volume 8, pages 269–278, 2000.

[27] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine learning*, 25(2-3):117–149, 1996.

[28] M. Siatecka and J. J. Bieker. The multifunctional role of eklf/klf1 during erythropoiesis. *Blood*, 118(8):2044–2054, 2011.

[29] D. Simcha, N. D. Price, and D. Geman. The limits of de novo dna motif discovery. *PloS one*, 7(11):e47836, 2012.

[30] M. R. Tallack, T. Whitington, W. S. Yuen, E. N. Wainwright, J. R. Keys, B. B. Gardiner, E. Nourbakhsh, N. Cloonan, S. M. Grimmond, T. L. Bailey, et al. A global role for klf1 in erythropoiesis revealed by chip-seq in primary erythroid cells. *Genome research*, 20(8):1052–1063, 2010.

[31] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, et al. Assessing computational tools for the discovery of transcription factor binding sites. *Nature biotechnology*, 23(1):137–144, 2005.

[32] C. Workman and G. Stormo. Ann-spec: a method for discovering transcription factor binding sites with improved specificity. In *Pac Symp Biocomput*, volume 5, pages 464–475, 2000.

[33] X. Xia. Position weight matrix, gibbs sampler, and the associated significance tests in motif characterization and prediction. *Scientifica*, 2012, 2012.