Bioinformatics, YYYY, 0–0
doi: 10.1093/bioinformatics/xxxxx
Advance Access Publication Date: DD Month YYYY
Manuscript Category

Discovering motifs in DNA sequences using Expectation Maximization (EM) Algorithm

Huimin Lu[1*]

[1]Department of Biology, Box 118, 221 00, Lund University, Sweden

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Sequence motif is a highly conserved sequence pattern that has biological significant. A typical example is transcription factor binding site, which is highly related to gene expression regulation. Discovering and researching motif could probably help find potential target in disease. And expectation maximization is an iterate algorithm that is wildly used in motif searching such as MEME. And the goal of this project is to construct a program to find motif in DNA sequences using EM.

Results: This program could find motif quicker than MEME, if the DNA sequences are short and the length of motif is short. But the complexity of this program is not enough to find motif that longer than 7 base among DNA sequences which are longer than 18.

Availability: This program can be used to find motif in other DNA sequences as well.

Contact: hu6451lu-s@student.lu.se

Supplementary information: This program and code can be browsed and downloaded in GitHub: https://github.com/luhuim/find_motif.git.

1 Introduction

Motif is short DNA sequences that have biological significant. And one of typical example for motif is that transcription factor binding sites which will combine with transcription factor during the transcription [1]. 73% of protein expressions are regulated by gene transcription in which DNA motifs play central roles [2]. So, finding and researching motifs among DNA sequences could help find potential target that could influence gene expression and get disease.

And expectation maximization (EM) algorithm is widely used in motif discovery [3]. And MEME tends to calculate multiple starting point in DNA sequences, which could get the motif model that matches, especially in finding motif among long sequences [3]. But, the computing method in MEME would take too much time to finding motif in short sequences.

The computation firstly assumes a position weight matrix (P-matrix) as initial matrix, and then it is iterated calculation that include estimation step (E-step) and maximization step (M-step) [4]. In E-step, it calculates Z matrix on the top of previous P-matrix; and in M-step it calculate a new P-matrix according to Z matrix. And this iteration will stop when the difference between new-generated P-matrix and last round P-matrix is lower that a specific value. In this program, the value is 0.0001.

And this program is to compute the position weight matrix according to basic EM algorithm only choose one subsequence from dataset as initial P matrix. That means this program has a light computing load to process short DNA input sequences and generate a motif logo. There are three DNA *fasta* files tested to find motif by both this program and MEME. And this program would generate motif logo quicker than MEME and the motif logo might be similar to the searching result in MEME, if the input sequences are short. More specifically, the length of motif should be 4-6 bases, and the length of DNA sequences should be 2-3 times longer than motif.

This program can be used to do rough motif-scanning in short DNA sequences. In this case, this program would be more efficient to generate the motif logo compared with MEME.

2 Methods

Feature of input file

The requirement of input *fasta* file is that it must be DNA sequences and the length of sequence should be the same, and the letters are all upper case. And the length of motif should be 4-6 bases, and the length of one sequence should sequences should be 2-3 times longer than motif.

There are three input *fasta* files prepared for this program. One file of them is referred from lecture slide in University of Otago and the rest two files are modified from two *sites* file from *JASPER* database whose ID are MA0004.1 and MA0006.1. In sites file from JASPER, the upper cases are motif that was found and lower case is non-motif base. Because the sequences lengths in *sites* files are not same, so some modification was made, selecting the sequences that has same length or removing non-motif bases, and transforming all letter into upper case.

Feature of output file

For every input file, there is a motif logo generated by MEME or JASPOR database. These motif logos can be regarded as reference result, and can be used to comparing with the result generated from this program for same input file.

Program environment and package

The program runs in conda environment and here is the dependencies and versions:

Conda: 22.9.0
Weblogo: 3.7
Python: 3.9
Pandas: 1.4
Numpy: 1.21

This program can be separated into two parts. The first part is a python program that calculate P-matrix for input DNA sequences and store this P-matrix in a text file. And the second step is to generate the motif logo from the position weight matrix in first program, using a software called "weblogo" [5].

EM computing

This part is in supplement.

3 Discussion

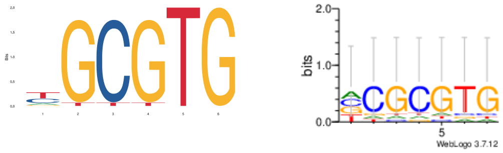
This program implements three input file that prepared, and contrast the searching time with MEME, and compare the motif logo with reference result.

Analyzing output file and potential issue

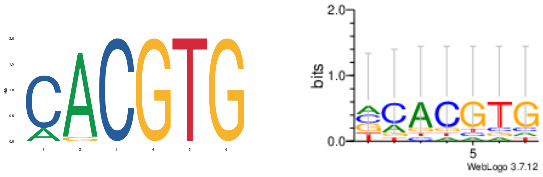
In terms of result, this program could find most of motif letter which weights much higher than other three kinds of base in that position (picture 1). The processing time of MEME doing with the same input file is much longer than this program. So, for some short sequence, this program could help quickly and roughly predict the motif inside DNA sequences.

And there is common issue for this program. Because EM is to calculate local maximization, it depends a lot on the initial P-matrix. And this program would randomly select an initial P-matrix, which means the motif logo could be totally different with reference result if the initial P-matrix was lower than most of P-matrix. Maybe running this program for couple of time could help find the local maximization P-matrix in average level.

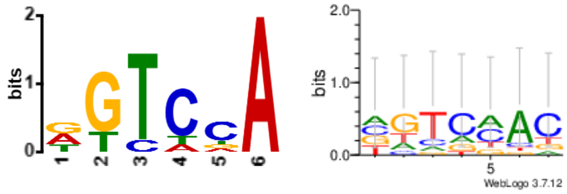
Picture 1. The reference result (left) and result from this program (right). For the picture on left, it is a six bases motif found by *JASPOR*, and the letters "GCGTG" who are in position 2-6, these five letter weight much more than other three kinds of base. And on the right, the first column is weight distribution of non-motif letter, and 2-7 columns are the distribution of six-base motif. The picture on the right shows that there are five out of six letters which get similar distribution as reference result.



Picture 2. The reference result (left) and result from this program (right). For the picture on left, it is a six bases motif found by *JASPOR*, and the letters "CACGTG", these six letters weight much more than other three kinds of base. And on the right, the first column is weight distribution of non-motif letter, and 2-7 columns are the distribution of six-base motif. The picture on the right shows that every position has a letter that has significant higher weight than other three kinds of bases in its position. And these letters are "CACGTG"



Picture 2. The reference result (left) and result from this program (right). For the picture on left, it is a six bases motif found by MEME, and the letters "G", "T", "C", "A" in position 2, 3, 4, 6, these four letters weight much more than other three kinds of base. And on the right, the first column is weight distribution of non-motif letter, and 2-7 columns are the distribution of six-base motif. The picture on the right shows that these four letters "G", "T", "C", "A" mentioned in reference result still weight more than other 3 kinds of base in its place, but their place is 1, 2, 3, 5. That means the motif that this program found has one-base move parallelly.



Limitation and future improvement

The limitation of this program is that computing complexity is not enough to implement long sequences input file. And this program only selects one initial matrix, and this would generate motif logo that doesn't represent the whole input file.

There are some aspects to improve this program. For example: select different initial P-matrixes and then give score and rank the maximization P-matrix from different initial P-matrixes. And increase the complexity of computation to implement the DNA sequences which are longer than 30 bases.

And motif finding technology could help find potential target of disease and develop drug which could aim at this target.

Acknowledgements

Thank for “Weblogo” to offer me a software to making motif logo; thank Mark Craven, Shirly Xiaole Liu for you great slide and video lectures. And finally thank Eran and Sara for your instruction.

References

[1] Trojanowski, Jorge et al. “Transcription activation is enhanced by multivalent interactions independent of phase separation.” Molecular cell vol. 82,10 (2022): 1878-1893.e10.Dormand,J.R. and Prince,P.J. (1980) A family of embedded Runge–Kutta formulae. *J. Comp. Appl. Math.*, **6**, 19–26.

[2] Li J.J., Biggin M.D. Gene expression. Statistics requantitates the central dogma. Science. 2015; 347:1066–1067.

[3] Timothy L. Bailey and Charles Elkan, "Fitting a mixture model by expectation maximization to discover motifs in biopolymers", Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, pp. 28-36, AAAI Press, Menlo Park, California, 1994.

[4] Lawrence CE, Reilly AA. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. Proteins. 1990;7(1):41-51.

[5] Crooks GE, Hon G, Chandonia JM, Brenner SE WebLogo: A sequence logo generator,Genome Research, 14:1188-1190, (2004)

Supplement

EM computing

In input file there are *I* sequences, and each sequence has *L* bases. And the length of motif is *W*.

Firstly, this program will extract subsequences whose length is *W* from sequences. And then randomly choose one subsequence to make initial position weight matrix, just like the matrix below. And p_{ck} represents the probability of character *c* in column *k*.

$$p = \begin{array}{cccccc} & 0 & 1 & 2 & \cdots\cdots & W \\ A & 0.25 & X & \frac{1-X}{3} & \cdots\cdots & X \\ C & 0.25 & \frac{1-X}{3} & \frac{1-X}{3} & \cdots\cdots & \frac{1-X}{3} \\ G & 0.25 & \frac{1-X}{3} & X & \cdots\cdots & \frac{1-X}{3} \\ T & 0.25 & \frac{1-X}{3} & \frac{1-X}{3} & \cdots\cdots & \frac{1-X}{3} \end{array}$$

In this matrix, X is assumed as the weight of chose subsequence character in every position, and column 0 in this matrix represents the weight each kind of non-motif character which was assumed 0.25. In the matrix above, it represents the chosen subsequence is “AG……A”, and other weight of other kinds of character get average weight.

Then, it goes to EM iteration computing step: E-step and M-step.

E-step

E-step is to make a matrix *Z* which has *j* columns and *I* rows. The element of matrix Z is Z_{ij} which means probability that motif starts in position *j* in sequence *i*.

$$Z = \begin{array}{ccccc} & 1 & 2 & \cdots & j \\ seq1 & Z_{11} & Z_{12} & \cdots & Z_{1j} \\ seq2 & Z_{21} & Z_{22} & \cdots & Z_{2j} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ seqI & Z_{I1} & Z_{I2} & \cdots & Z_{Ij} \end{array}$$

The calculation of Z_{ij} is shown below:

$$\Pr(X_i \mid Z_{ij} = 1, p) = \prod_{k=1}^{j-1} p_{c_k,0} \prod_{k=j}^{j+W-1} p_{c_k,k-j+1} \prod_{k=j+W}^{L} p_{c_k,0}$$

After filling all Z_{ij} in Z-matrix, it will go to M-step, calculating a new P-matrix.

$$p_{c,k}^{(t+1)} = \frac{n_{c,k} + d_{c,k}}{\sum_b (n_{b,k} + d_{b,k})}$$

pseudo-counts

$$n_{c,k} = \begin{cases} \sum_i \sum_{\{j \mid X_{i,j+k-1} = c\}} Z_{ij} & k > 0 \\ n_c - \sum_{j=1}^{W} n_{c,j} & k = 0 \end{cases}$$

total # of c's
in data set

After getting a new P-matrix, the program will compare the difference between previous P-matrix and new-generated P-matrix by calculating Frobenius norm. If Frobenius norm value is lower than threshold that means P-matrix converged and the iteration can stop. In this program the threshold is 0.0001.

Hers is an example of Frobenius norm. Here are two matrix:
P1 = | 0.2 0.6 | | 0.4 0.4 |
P2 = | 0.3 0.5 | | 0.3 0.5 |
Subtract the elements:
| (0.2-0.3) (0.6-0.5) | | (0.4-0.3) (0.4-0.5) |=| -0.1 0.1 | | 0.1 -0.1 |
Square the results:
| (-0.1) ^2 (0.1) ^2 | | (0.1)^2 (-0.1)^2 |=| 0.01 0.01 | | 0.01 0.01 |
Add up all the squared numbers: 0.01 + 0.01 + 0.01 + 0.01 = 0.04
Take the square root of the sum: $\sqrt{0.04}$ = 0.2
Frobenius norm=0.2

After the program get Frobenius norm, it will compare with threshold.

Finally, the program will use software “WebLogo” to make motif logo on the top of final position weight matrix.

Dataset for Picture 1

```
>seq1
CACAGTCGCGTGT
>seq2
ACTATCGCGTGTT
>seq3
ACTGTTCGCGTGC
>seq4
ATCTCATCGCGTG
>seq5
AGGAATCGCGTGC
>seq6
GGAGTGTCGCGTG
>seq7
TAGGGGTCGCGTG
>seq8
GGATCGCGTGTCC
```

Dataset for Picture 2

```
>seq1
CACGTGATGTCCTC
>seq2
CACGTGGGAGGTAC
>seq3
CACGTGCCGCGCGC
>seq4
CACGTGAAGTTGTC
>seq5
AACGTGACTTCGTA
```

Dataset for Picture 3

```
>seq1
CTATAAACGTTACA
>seq2
ATAGCGATTCGACT
>seq3
CAGCCCAGAACCCT
>seq4
CGGTGAACCTTACA
>seq5
TGCATTCAATAGCT
>seq6
TGCTCTGTCCACTC
>seq7
TGCTCTGTCCACTC
>seq8
GGTCTACCTTTATC
```