Motif-finding by Gibbs Sampling

2017/6/28

Minstein

Problem:

Given p strings and a length k, find the most "mutually similar" length-k substring from each string.

Assumption:

1. Each motif appears exactly once in one sequence
2. The motif has fixed length k

Algorithm:

1. Start with random motif locations and calculate a motif model
2. Randomly select a sequence, remove its motif and recalculate temporary model
3. With temporary model, calculate probability of motif at each position on sequence
4. Select new position based on this distribution
5. Update model and iterate

Implementation:

To solve this problem, we need to find out the motif start indexes in the input sequences. We denote the motif start indexes as $s = (s_1, s_2, \dots, s_p)$. Once s is obtained, we can generate the corresponding PWM (position weight matrix), motif information as well as the final representative motif. Here, we use Kullback-Leibler distance to measure the motif information. The alphabet here is $\{A, T, C, G\}$, which can be easily extended according to the specific situations. We set the background frequency of every nucleotide equally as 0.25. This parameter can also be adjusted to the occurrence of nucleotides in the input sequences.

In addition, there are other parameters adjustable for users. They are motif length, average information content per site and iteration times, and the default value are 6, 1.6

and 100000, respectively.

The score function applied for scoring a sequence by a given PWM is:

$$Score(x) = \frac{\Pr\,(x|\text{PWM})}{\Pr\,(x|\text{background})} = \prod_{i=1}^{k} \frac{e_i(x_i)}{b_i(x_i)}$$

where $e_i(x)$ represents the probability that ith position has the given character and $b(x_i)$ represents the probability of observing character x_i at random.

For any given PWM, its information content is calculated by the following formula:

$$\sum_i \sum_{\beta \in \{A,T,C,G\}} W_{\beta i} \log \frac{W_{\beta i}}{q_\beta}$$

where $W_{\beta i}$ equals to the frequency of base β at position i and q_β equals to frequency of base β by chance.

Test:

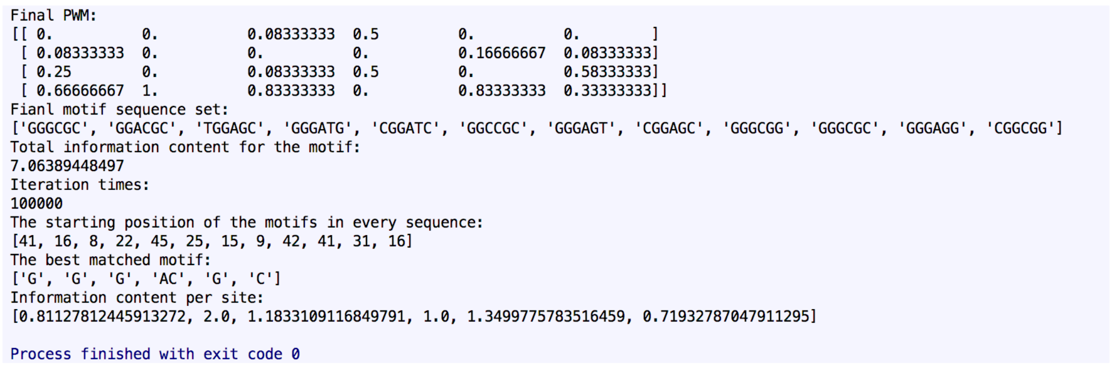
We use 12 randomly generated sequences to test our program and the detailed sequences are listed in Table 1. Notably, the input sequences are not of the same length. All adjustable parameters are set to the default value.

Table 1. Input sequences for testing the program

Item	Sequence
Seq1	CCCGGGCCCAGCGTCTGGCCTCGCGATGGAAGCTGCTGTGCGGGCGCTGAG
Seq2	ACCGCCTGCTGGCCTGGGACGCAGCATGCCTCCCGCCGCCGCCCGCCGCCT
Seq3	AGCGAGGATGGAGCCCGCCATCAGCTGGTGCTG
Seq4	GGGCTCGACGCCTCCCCGCTTCGGGATGAATTAGCGGCGGGTTCTTCTCCC
Seq5	CACTGTGAACGGAAGACAAAAAACAATGTATAGTAAAACAGAAGGCGGATC
Seq6	GGGCCCTTCACCTCTGACCCAAAATGGCCGCGCCCAGAGCGTAGTTCTT

Seq7	TTCTCATTTCCTTTTGGGAGTATTCATGCTTCACCATGTTGAGAGCTAAGA
Seq8	CGCAGTATCCGGAGCTGGGGAATTCATGTGGAGGTCAGAGTGAAAGCAGGT
Seq9	AGGCCGACTCCGGGAAGAAAAGAACATGGCTCCTGCGGCAGAGGGCGGCGG
Seq10	CCCCCCCCCCGCTGGCCTCAACCAATGAGCACGCACGCCAGGGCGCGCTG
Seq11	CAGCGTCACGCGTGACGTCTCGGTGATGGCTGGGAGGAAAGCGGAGAGCGG
Seq12	CGGACACAGCCAGGCGCGGCGGGATATGTGGTACATTCCCTTCCCTCCAGA

After iterating 100000 times (finished in less than 1 minute), the result comes out and the raw output by the program is listed in Figure 1. The best matched motif found is GGGMGC, where M represents A or C. The total information content for this motif is 7.06. Notably, the second position of the motif is relatively more conservative.



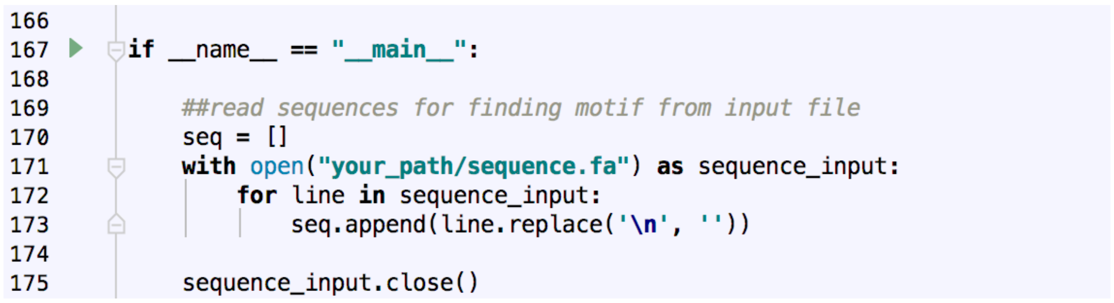
```
Final PWM:
[[ 0.          0.          0.08333333  0.5         0.          0.        ]
 [ 0.08333333  0.          0.          0.          0.16666667  0.08333333]
 [ 0.25        0.          0.08333333  0.5         0.          0.58333333]
 [ 0.66666667  1.          0.83333333  0.          0.83333333  0.33333333]]
Fianl motif sequence set:
['GGGCGC', 'GGACGC', 'TGGAGC', 'GGGATG', 'CGGATC', 'GGCCGC', 'GGGAGT', 'CGGAGC', 'GGGCGG', 'GGGCGC', 'GGGAGG', 'CGGCGG']
Total information content for the motif:
7.06389448497
Iteration times:
100000
The starting position of the motifs in every sequence:
[41, 16, 8, 22, 45, 25, 15, 9, 42, 41, 31, 16]
The best matched motif:
['G', 'G', 'G', 'AC', 'G', 'C']
Information content per site:
[0.81127812445913272, 2.0, 1.1833109116849791, 1.0, 1.3499775783516459, 0.71932787047911295]

Process finished with exit code 0
```

Figure 1. The output of our implemented program

Attention:

Before testing our program, you might need to edit the path of the file storing the input sequences (Figure 2). This program is implemented in python and the source codes are included in motif-finder.py.



```
if __name__ == "__main__":

    ##read sequences for finding motif from input file
    seq = []
    with open("your_path/sequence.fa") as sequence_input:
        for line in sequence_input:
            seq.append(line.replace('\n', ''))

    sequence_input.close()
```

Figure 2. The path of input sequences remains to be edited

Reference:

1. http://veda.cs.uiuc.edu/courses/fa08/cs466/lectures/Lecture16.pdf
2. https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/gibbs.pdf
3. https://github.com/zshamsi2/motifFinding