Finding Subtle Motifs by Branching from Sample Strings[1]

Akshita Sawhney

28 October 2017

1 Introduction

The main reason for finding motifs in a DNA sequence is to find a motif with major occurrences at unknown positions.Finding a sequence motif for gene analysis has become very much important. These motifs are repetitive and have their own prominent biological function .They are also used to help identify the binding sites for transcription factors or other proteins. Various approaches are used find motifs to become motif finding algorithms. One of a high computational cost naive algorithm is to search the space identified for motif patterns. Other better approaches are: Sample driven approach in this there are selected sample strings that are used for searching the appropriate motif. Another is the extended-sample driven approach which searches all the neighbours as well of the sample strings by using some exhaustive search technique. These approaches again have a high computational cost. One of the most trivial algorithm that find the motifs more efficiently is a technique name branching from sample strings which finds a best motif using greedy approach. In this there is branching in small subsets of motif space. There are two versions of this algorithm. One which models motif as a pattern of l consensus nucleotides, finding out the best nucleotide for the motif. The other way would be modeling it with profile, which will be a 4 l matrix of nucleotide probabilities for each position of the motif.

2 Summary of Algorithms

2.1 The pattern branching algorithm

This algorithm starts by taking an arbitrary motif sequence as the initial motif. Iteratively, the algorithm fine tunes the motif on the basis of the distance with S. Hamming distance is used as the metric. Best neighbour was also found by mutating the lmer and then finding its distance with S. The lmer with the minimum hamming distance from the set S is chosen as the final motif.

```
PatternBranching(S, l, k)
Motif ← arbitrary motif pattern
For each l-mer A_0 in S
    For j ← 0 to k
        If d(A_j, S) < d(Motif, S)
            Motif ← A_j
        A_{j+1} ← BestNeighbor(A_j)
        Output Motif
```

2.2 The profile branching algorithm

Profile branching algorithm works on the principle of sequence profiling. It takes into account the probabilities of nucleotides and build a matrix using those. Entropy is used as the metric in this algorithm. It then employs nucleotide amplification strategy to get the neighbours and then chooses the neighbour with the maximum entropy. It also functions iteratively and gives the seed which is then fed to the EM algorithm.

```
ProfileBranching(S, l, k)
Motif ← arbitrary motif profile
For each l-mer A_0 in S
    X_0 ← X(A_0)
    For j ← 0 to k
        If e(X_j, S) > e(Motif, S)
            Motif ← X_j
        X_{j+1} ← BestNeighbor(X_j)
Run EM algorithm with Motif as seed
```

3 Implementation

3.1 Pattern Branching Algortithm

The implementation of this algorithm is done in python and has the following modules:

arbitararymotif: This module takes as input the size of the motif as a parameter. It calculates a random motif with the help of RANDOM function.

lmer: This module takes one sequence at a time and lists all the lmers of each sequence according to the size of the motif.

finddistance: The role of this module is to calculate the distance. It takes the list of sequence from the above module in one parameter. In the other parameter we pass the arbitrary motif once and one lmer at a time . To calculate the distance we use the concept of hamming distance between the two sequences that are passed. Hamming distance is the number of different characters in the two sequences. We calculate this hamming distance for each and every sequence's lmer with all the lmers ,find the minimum for 1 sequence and then find the sum of all the minimums. Similarly when we pass the arbitrary motif we again take its distance with all the lmers of each sequence, find the minimums and then sum all the minimums.

getbestneighbour: In this module we modify each lmer 1 hamming distance at a time and find the distance of each changed lmer with the updated lmer and return the best neighbour.

3.2 Profile Branching Algorithm

In this algorithm following are the different modules from the above:

getprofile: Gives the 4xl matrix with the grids with same column and row value to be 1/2 and the others to be 1/6. entropy: In this we find the entropy of set entropy1 In this we find the entropy of each sequence in the set entropy2 In this module we find the final entropy of the pattern

References

[1] Alkes Price, Sriram Ramabhadran, and Pavel A Pevzner. Finding subtle motifs by branching from sample strings. 19 Suppl 2:ii149–55, 11 2003.