CS466 Mini Final Project

Jeremy Tang
jtang38

Abstract

This paper describes an implementation for Motif Finding, an algorithm in which we find frequent motifs with length L in a dataset of multiple sequences. This implementation uses Gibbs sampling in order to find motifs. This paper will first discuss the problem, introduce a high level overview of the dataset and implementation, show results from this implementation, and discuss possible future work to be done.

Problem

An important challenge in the field of Bioinformatics is the problem of detecting specific gene sequences used during transcription. These gene sequences regulate gene expression, and are able to activate and deactivate certain protein generation functions at a certain time. These patterns, called “motifs”, occur more frequently in DNA sequences than other patterns. Motifs are most likely to be binding sites for transcription factors, which directly influence gene regulation.

Motifs are not necessarily the same in different sequences, as different transcription factors can bind to different variants of a pattern. For example, we can have the pattern **ACTAGG** in one sequence, and a similar pattern, **TCTAGG** in another sequence. Though these two patterns are different, they share many similar nucleotides.

It is important to discover the sites where these motifs occur, because they can help Biologists understand if a certain gene is being regulated by a transcription factor.

Dataset

The datasets used for testing are generated through a script, generate_sequences.py. This script generates sequences in the alphabet [‘A’, ‘C’, ‘G’, ‘T’] and writes them to a Fasta file ‘sequences.fa’. Inside the script itself, we can set parameters such as the number of sequences we want to generate inside the file, the length of each sequence, and the length of the motif we want generated inside each sequence.

The script works by first finding a random motif location for each sequence. Then, the sequence is created by selecting nucleotides using priors for the locations in the motif sequence, while all the other nucleotides are uniformly randomly selected from [‘A’, ‘C’, ‘G’, ‘T’]. We use priors for the nucleotides in the motif to ensure that our motifs are very similar to each other. The script then writes each sequence to a file in Fasta format in the form of:

```
>seq0
ACTCGCGGCTTAGCT
>seq1
CGTTACAAGGCGTTAA
```

The nucleotides highlighted in red and underlined represent the chosen motif location for each sequence.

Implementation

For this implementation, I used Gibbs Sampling in order to find the motifs in the sequences. First, we choose at random a possible start location for the motif in each sequence. We then iterate between two different steps. The first step leaves one sequence out, and the position weight matrix and background probabilities are then calculated using this set of sequences with one left out. In the second step, we calculate a new possible position of the motif based on the previously calculated position weight matrix. We iterate between these two steps until convergence.

Results

From testing on different datasets, I got pretty reasonable results. I created datasets ranging from 10-40 sequences, with each sequence ranging in length from 150-500 nucleotides. In general, the code did not take very long to run. At the end, we include a plot for time taken to run the code vs. motif length in order to compare the efficiency of our algorithm.

Test 1

Number of Sequences: 20
Sequence Length: 150
Motif Length : 4
Time taken: 0.385299921036

```
# 1  GGCG : 177
# 2  CACC : 300
# 3  GAAG : 488
# 4  CGCG : 299
# 5  GGCC : 302
# 6  CACG : 495
# 7  CAAG : 14
# 8  GACG : 54
# 9  CACC : 79
#10  GACG : 480
#11  GACG : 228
#12  GACC : 446
#13  GACC : 298
#14  CGCG : 140
#15  CACG : 122
#16  TACG : 322
#17  CACG : 50
#18  GGCC : 376
#19  GGCC : 164
#20  CGCG : 388
```

Test 2

Number of Sequences: 20
Sequence Length: 150
Motif Length : 15
Time taken: 0.449369907379

```
# 1  CTCCCCCCTCCCTCC : 70
# 2  CTCGGCCCCCACAGA : 18
# 3  TTCGCTCCCTACGCG : 123
# 4  CCTCCTCCCAACTCT : 6
# 5  CCCCCCCCTCACTCT : 10
# 6  CCTCCTCCTTCCACT : 24
# 7  CTTGCTCCCTACGTA : 66
# 8  CCCCCTCCTCACTCG : 55
# 9  GCCCCTCCTTCCTCT : 25
#10  CTCCCCCCCTTCTGC : 13
#11  GCTGCTTCCTTCTCT : 126
#12  CTTCCCCCCACCGGT : 47
#13  CCCGCCCCTCCCAGT : 129
#14  GTCCGCCCCTACTCT : 134
#15  CCTCCTCCTCACTCG : 50
#16  GTCCCTTCTCACTCT : 20
#17  GCCCCCCCTTATGGC : 42
#18  CCCCCTCCCTCCTGG : 130
#19  GATGCCCCCCACGCG : 100
#20  CCCCCCCCCTACTGT : 4
```

Test 3

Number of Sequences: 20
Sequence Length: 500
Motif Length: 15
Time taken: 2.21990084648

```
# 1  CGGGCGGAGACTGGG : 337
# 2  GGGCGGGACAAGGCC : 322
# 3  CCGCGCCCCACGGCG : 2
# 4  CCGGGCGCCCAGGGT : 447
# 5  CGCCGCTCCCCGGGC : 46
# 6  CTGCAGCCCAAGGGC : 191
# 7  CCGCGGCCGACGGCC : 321
# 8  CGCCGCCACAAGGGT : 309
# 9  GCGCAGGCCACGGCT : 225
#10  GGCGGCGACACGGGT : 191
#11  CCCGGCGCCCAGCGG : 360
#12  CCGTCCCCCAAGGGC : 48
#13  CTGGGGTCCCCGGGT : 195
#14  CGCGCCGCCACGGGC : 108
#15  CGGTCCCACAAGGCC : 7
#16  CCCTGCGAGCCGCGC : 259
#17  CTCCAGGACACGGCT : 220
#18  GGGGCGGCCCATGGC : 285
#19  GCGGGGGCCACGGGG : 159
#20  CCGGAGGCGACGGGT : 299
```

Test 4

Number of Sequences: 20
Sequence Length: 500
Motif Length: 45
Time taken: 4.99748301506

```
# 1  CACATCCCGGGGCGCGACCGGGCCGGCGCACGAGGACCACCCGAC : 57
# 2  CACAGCCACAGGGTGGGCAGCACCCCCGTCCCCGCCCCCGGCGGC : 123
# 3  GCCAAGCTACGGGGGGACACGCGCGGGCATCGCGGCTGCCGGAAT : 234
# 4  CACGAGAGGCCCCGGCACGCGGCGCCCCGCCACGGAGGAGGGGGG : 234
# 5  CCGCATCGGCTGGTACCCCCTCGGCACTGAGCCCCAGGTCGCCGT : 223
# 6  ACGCAACCGGTGCGCAGAACCGGCGCGTTTGCCCACCCCGCTGGT : 231
# 7  GCGCACCAGCGGCGGGACCGTCCCGCGGCCGACGGCCGAAGCGGT : 299
# 8  GCCGCCACAAGGGTGACCGGCACGCAGTCACGGCCCCCAGCTGGC : 310
# 9  ACCCGGAAGGGGCTGTGCAAGGGGGAGGTTGCCAGCGCAGGCCAC : 191
#10  GCCCCCAACCCGGGCACAAAGGCCCGAGTGCCCGGCGGAGGTGGG : 260
#11  GCCACACACGGGCGGGAGCAGGGGGCCGCCGCCCACGCCGGCGAT : 400
#12  CACCCCAGGCGCCGCCACCACCCCCCGACGCGACGAGCGCGCCAC : 164
#13  GCCATCCGGACGCGACGACCGCGCCGCGCAGCCCAGGGTAGTAGC : 353
#14  GCCACGGGCCCCGTCCCCCAGACGCGCGCGGGCCACCGTAGCGGC : 114
#15  GCGAAGCAGCGGCGCGGCGGCCGCGACCCGCAGCGCGGTGGGCAT : 340
#16  CAGCCACTGCTCGGCTGCGAGGCGCACGGCCCCCCGGAGCGTGCG : 87
#17  GCGGACGGGGTGCTAGGCGAAAGCGCCCCCGCGGGCCTTCGCCGT : 392
#18  GACGTCCCCCTCGGACGCCCACCGCCGGGGGGCCGGGCCGCGGAC : 341
#19  GCCCCCACGGGGCGGCGCACAGCCGAGCCCGCGCGATTGGCGGCC : 283
#20  GCCGCCACCCCCCGAGGCACAGGCCCGGCCCACGCCGGAGGCGAC : 265
```

Test 5

Number of Sequences: 40
Sequence Length: 500
Motif Length: 15
Time taken: 0.437600851059

```
# 1  TGGGTTTTTGGTTGG : 4
# 2  GTTTGGAGTGGTTGG : 22
# 3  AGTGTTGTTTGTTTG : 25
# 4  TTAGAGGTGGATTTT : 82
# 5  GGTATTTGTGGTTGG : 68
# 6  TTAGTTGTTGGATTG : 6
# 7  GATGAGGGGGGGTTG : 10
# 8  ATTAGTATTAGTTAG : 11
# 9  TGATGGAGGATTGGG : 49
#10  AGTGTGGTTTGATGT : 72
#11  GGGGATGTTTTTTTG : 47
#12  GGTGGTTGTGGTGGT : 12
#13  TTAGATTTTGATTTG : 43
#14  TGAAGTTGTTTGTGG : 43
#15  TGAAGGTTTGTGTTG : 40
#16  AGGAAGGTGGAGTGT : 84
#17  TGTGGTTGTGTTTTG : 55
#18  ATTATTTTTAGTTGG : 5
```

```
#19  AGGAGGGGGGGTTGT : 56
#20  AATAGTTTTGTTTGT : 49
#21  AGTTATGGTGGGGGG : 8
#22  GGTGGTGTTGGAGTG : 11
#23  AAGGAGGGTTTTTGG : 29
#24  TGATGTTGCTGTTGG : 34
#25  GGTAGTGTTGTTTGT : 13
#26  GGGGAGGGTTTTTGG : 38
#27  TTTAGTGTTGTTTTG : 13
#28  GATGGCTGTGTTGGT : 6
#29  ATGGTTATTGTTTTG : 20
#30  GGAAGGTTTGTTTGT : 69
#31  AGTGTGTGTGAATAT : 71
#32  TGAAAGGGCAATTTT : 86
#33  ATTGGGGGTGGGTGG : 65
#34  AGTGGGGGGAGGTTG : 55
#35  TAGAGTGTTAGTTGG : 51
#36  GATAAGAGTTGGTTG : 42
#37  GGAAGTATTTGTTTG : 9
#38  ATGAGGGTTGTGTTG : 37
#39  TTTGGTGTTGGTTGG : 13
#40  AGGAGTTGTGGGTAG : 24
```

As we can see, the execution time is dependent on motif length. This makes sense, because the longer the motif we search for, the tougher it is to find in a sequence. A smaller length motif is more common in a sequence, and so the code can find it much more easily.

Possible Future Work

There is a lot of possible work to be completed surrounding Motif Finding in Bioinformatics. Exploring different algorithms and their complexities is a possibility, as well as improving this current algorithm to work better in certain situations. We can also compare different algorithms and different implementations, in order to better understand which algorithms fair better than other algorithms in different situations.

Conclusion

In this paper, we discussed the problem revolving Motif Finding and why it is needed in the field of Bioinformatics and identifying the mechanisms surrounding gene expression. By identifying motifs in nucleotide sequences, we can determine binding sites for transcription factors, and in turn discover certain gene regulatory mechanisms that might improve how scientists understand life.