Arthi Haripriyan

Anna Ritz

Biology 131

14 May 2020

Randomized Motif Search and Gibb's Sampling

Biological Question

Finding patterns in DNA is an important and useful part of analyzing DNA sequences which offers many applications for computational methods. One example of this is the circadian clock, an organism's internal timekeeper. Since the circadian clock must have some molecular basis, some questions that can be asked are: How do individual cells know when they should slow down or increase the production of certain proteins? Is there a "clock gene?" In plant cells, three genes LHY, CCA1, and TOC1 are the "clock genes." These genes are able to control the transcription of other genes because the regulatory proteins that they encode are transcription factors that turn other genes on and off. A transcription factor regulates a gene by binding to a transcription factor binding site (motif) in the gene's upstream region. Finding these binding sites would be relatively easy if the motifs were completely conserved, but regulatory motifs can vary at some positions, making this search more complicated. The fundamental question here would be: how can we locate these regulatory motifs without knowing what they look like in advance? This then becomes a computational problem: to develop algorithms for motif finding.

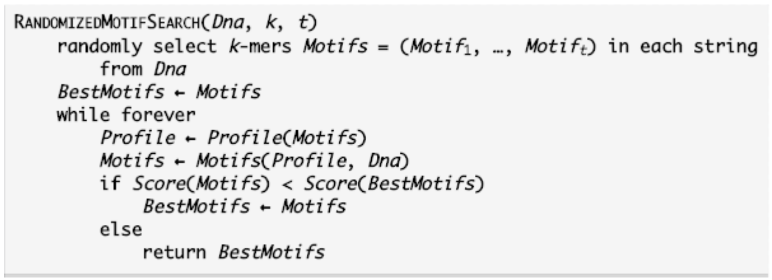
Greedy Motif Search

Three similar methods for motif finding are the Greedy motif search, Randomized motif search, and Gibb's Sampling. The basic idea of these algorithms is to find the set of motifs across a number of DNA sequences that match each other most closely. The steps for the

Greedy Motif Search algorithm are: 1) Run through each possible *k*-mer in our first dna string, 2) Identify the best matches for this initial *k*-mer within each of the following dna strings (using a profile-most probable function) thus creating a set of motifs at each step, and 3) Score each set of motifs to find and return the best scoring set. The other two algorithms simply build on this.

Randomized Motif Search

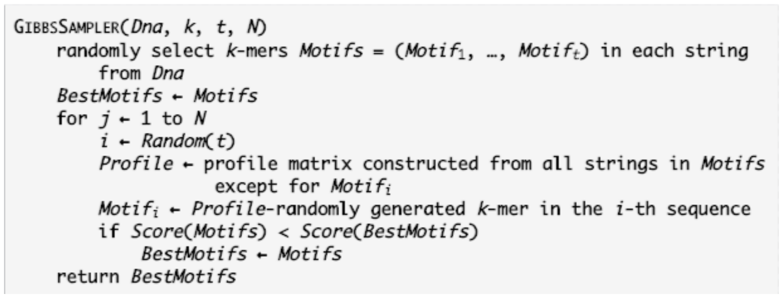
Randomized algorithms are those that flip coins and roll dice in order to search for motifs. The Las Vegas algorithms find solutions that are guaranteed to be exact. The Monte Carlo algorithms are not guaranteed to return exact solutions, but they do quickly find approximate solutions. The best approximation can be chosen from thousands of runs.



```
RANDOMIZEDMOTIFSEARCH(Dna, k, t)
    randomly select k-mers Motifs = (Motif_1, …, Motif_t) in each string
        from Dna
    BestMotifs ← Motifs
    while forever
        Profile ← Profile(Motifs)
        Motifs ← Motifs(Profile, Dna)
        if Score(Motifs) < Score(BestMotifs)
            BestMotifs ← Motifs
        else
            return BestMotifs
```

The randomized motif search is a modification of the Greedy motif search. The Greedy algorithm selects the first *k*-mers in each string from Dna as the initial BestMotifs. However, the Randomized search randomly selects *k*-mers from each string of Dna as the initial BestMotifs, using a random number generator. Otherwise, the rest of the code is basically the same. The profile is still calculated, a new set of motifs is created based on the profile and the Dna, and the new set of motifs are scored. The code stops when an iteration through the loop does not improve the new score.

Gibb's Sampler



```
GibbsSampler(Dna, k, t, N)
    randomly select k-mers Motifs = ($Motif_1$, …, $Motif_t$) in each string
        from Dna
    BestMotifs ← Motifs
    for j ← 1 to N
        i ← Random(t)
        Profile ← profile matrix constructed from all strings in Motifs
                   except for $Motif_i$
        $Motif_i$ ← Profile-randomly generated k-mer in the i-th sequence
        if Score(Motifs) < Score(BestMotifs)
            BestMotifs ← Motifs
    return BestMotifs
```

Like the Randomized Search, GibbsSampler starts with randomly chosen *k*-mers in each DNA sequence as the BestMotifs. However, in the GibbsSampler, motifs are changed one at a time. From the current set of motifs, one is randomly selected to be removed. A profile is created using the remaining motifs (after the one is removed). A new motif is computed based on the profile and reinserted into the set. The motifs are then scored as in the previous two algorithms. The main difference between the last two methods is that the Randomized search might change all the *k*-mers in one step, and the GibbsSampler changes one *k*-mer in one step. This is an improvement because there is a better chance of preserving better matched motis.

Results

Both programs were run on sample inputs from Rosalind. It seemed impractical to verify if each output was correct since they were going to different every single time, simply given the nature of the randomization. However, I had the old and new scores printed each time and the new score was always smaller than the old score. These lines got printed 2–3 times (for the Randomized) or more (for the Gibbs) in an effort to find the smallest score possible. For the Randomized, the smallest score found seemed to usually be in the range 16–18; however, at least a couple tests had a new score of 9. For the Gibbs, the smallest score for the inputs run from Rosalind seemed to consistently give the smallest score of 9.

The code is in the Fake Assignment Project created in repl.it.

I would be happy to have my report on the course webpage!