Project Proposal - Algorithms for Molecular Biology

Members

Han Ngo
Christine Samson
Monte Anderson
Sushma Colanukudhuru
Rasmi Lamichhane
Byron Bearden

Introduction

Finding motifs in sequences is an interesting problem to tackle. A motif consists of a nucleotide or amino-acid sequence pattern that can occur in a multitude of different sequences and may have biological significance that can then be examined and analyzed. Special binding sites that produces a specific motif is an important component in motif finding. The goal is to improve upon the current motif finding sequence tools that currently exist.

While there are many motif finding sequence tools that currently exist, the current algorithm they use can be improved upon. A more specific way to state the motif finding problem would be: given a set of DNA sequences, the goal is to find a set of k-mers, one from each sequence that maximizes the consensus score. Any current or earlier motif finding algorithms that were developed are dependent upon word-based manipulative methods; these methods usually rely on exhaustively counting and comparing. Another method for motif finding includes a string manipulative methods - the advantage to this one could be fast when implemented with suffix trees. However, for a long and less-constrained dataset, using suffix trees could result in a problematic and complicated post-process. In order to compensate for this issue, one could provide model parameters and using probabilistic methods to estimate the

maximum likelihood of the motifs. Finding globally optimal solutions using these algorithms are not guaranteed because of the local search that is employed (ex. Gibbs sampling), but such greedy algorithms can lead to a locally optimal solution.

The standard algorithms from HMMs and local alignment will be utilized in order to create an ensemble of profile HMMs. In particular, an initial ensemble of profile Hidden Markov Models (HMMs) can be constructed from the local alignment of two sequences. This ensemble is continuously updated and help in generating a list of best motif candidates. This approach can be tested against the algorithm MEME on synthesized data and real genome data.

Dataset Generation

The yeast genome will be where the dataset is taken from. The yeast genome is the least complicated and the most well mapped genome that currently exists. The human genome is too complicated and is also not completely mapped; therefore, it is beyond the scope of this project, and would be difficult to try to implement the human genome for analysis by the algorithm generated.

The algorithm will be tested on small datasets. It would be difficult to ascertain whether or not the results generated from the coded algorithm would be correct if a large dataset was used. Using a smaller dataset can also ensure that what the algorithm generated is correct. Using a smaller dataset would also ensure that the coded algorithm/application doesn't time out. This accounts for the length and type of dataset that the coded algorithm can handle, thus, we are taking measures to ensure that the algorithm we coded can run a small dataset first, before moving on to a large (and possibly) neverending dataset.

The algorithm and application will also be tested on random and known datasets; this would be examples of a test case that we would test for. If we provide small test cases of things that we should be aware of, we can model it so we can get exactly what we would expect to get. For random datasets, we can check if motifs overlap, and if we catch them or miss certain ones (error and bug testing). Using a random dataset would cover the edge cases for our algorithm. Overlapping motifs would be an edge case where a motif (such as a 3-mer) could overlap with another 3-mer. For example, if a 3-mer was found at position 502, and another 3-mer can be found at position 504, a random dataset might be able to generate more overlaps (than what can be generated manually), thus, we can see whether or not we caught all the overlaps, missed some, or caught none of them. Since we are generating the datasets, we know how many we will create of each kmer and how many, so our algorithm should catch them all, unless we decide that some of the edge cases shouldn't count.

Known datasets will be generated from the existing application (in this case, MEME) itself, or used from Saccharomyces Genome Database. A broad overview of how known datasets will be generated will be according to the following: first, we will count for yeast genomes. The yeast genomes will then be broken down into smaller samples to account for a smaller dataset. Next, the smaller dataset that is generated would be run through MEME to generate the results that the existing application would give. This would allow us to have a baseline for what results we need to hit with our coded application/algorithm. By putting the same dataset through the coded algorithm, and comparing those results with the existing algorithm's results, we can discover whether or not we are getting the same results, and if our code is correct.

Randomly generated motifs can help us ascertain whether or not we are able to catch edge cases (for example, being able to detect overlapping motifs). This part will be more explained in the benchmarking section, as it pertains more to that than generating the actual datasets. Different lengths of k-mers and motifs will be another test case in addition to random and known test cases, mostly just to test whether different lengths of k-mers and motifs can be found.

Algorithms & Methods

1> Initialization

The method starts with two arbitrary sequences in a sequence set and employs Smith-Waterman algorithms for local alignment of the two sequences. To construct an ensemble of k HMMs, a dynamic programming table needs to be maintained and the alignments with the k largest alignment scores will be selected. Each local alignment with a significant alignment score will be used to construct a HMM. Specifically, based on a tracing back procedure, we are able to identify the subsequences corresponding to the k largest alignment scores. An initial ensemble of k profile HMMs will then be constructed from the k alignments.

Steps in HMM modelling:

Cyclic protein

2> Updating Parameters and the Ensemble

We progressively use the HMMs to scan through each remaining sequence in the set. For each HMM and each position in the sequence, we align the subsequence to the HMM. For each position in the sequence we can compute an alignment score. The method selects k positions with the largest alignment scores. In particular, each HMM in the ensemble is aligned to other sequence and the alignments with larger scores are selected to update the parameters of the HMM. This process may create more HMMs, but only the alignments that have the most significant alignment scores are selected to form a new ensemble of HMMs.

3> Repeating the Pattern

We repeat this procedure until all sequences in the set have been processed and the HMMs in the ensemble provide candidate motifs.

Benchmarking Tool

The tool that will be examined in detail is the Multiple EM for Motif Elicitation (MEME) Suite, which is a collection of tools for the discovery and analysis of sequence motifs in a group of related DNA or protein sequences. MEME is hosted at http://meme-suite.org/ . The user puts into MEME some DNA or protein sequence; MEME then outputs as many motifs that were requested up to a user-specified statistical confidence threshold. MEME utilizes statistical modeling techniques to automatically choose the best width, number of occurrences, and description for each motif. However, there are some drawbacks that the individual MEME motifs are involved with:

1> Allowance for gaps/substitutions/insertions not included.

2> Erased input data each time a new motif is discovered (the algorithm assumes the new motif is correct)

Our approach of customizing the ensemble of HMMs will touch on those two restrictions of MEME suite. To accurately benchmark the results for our designated HMMs method, we will utilize the same generated/synthetic dataset.

Visualization

We hope to be able to visualize the process of finding motifs and the results that come from the application/algorithm generated. Either generating the results in a 2D format or a 3D format would be sufficient, but if time allows, the motif finding process and its results would hopefully be generated in a virtual reality format.

Responsibilities and Milestones

Week	***Responsibility***	***Members***	***Milestone***
4/9 - 4/15	Generate the Dataset Code the Algorithm MEME Research/Benchmark Criteria	Dataset - Monte, Christine Algorithm - Hannie, Sushma Research - Byron, Rasmi	Complete dataset generation. Get as far with coding the algorithm as one can. Finish MEME research - what meme does, how it works, etc. Refine/finalize the benchmark criteria.
4/16 - 4/22 ***Note:*** *Benchmark is due* ***4/19***	Code Algorithm Run Benchmark	Algorithm - Hannie, Sushma Baseline - Rasmi, Byron Monte and Christine will help wherever needs help	Finish coding the algorithm early in the week. Run the dataset on the coded algorithm and on MEME to get the baseline.
4/23 - 4/29	Refine Algorithm if needed Do Visualization if possible	Algorithm - Any Visualization - Monte, Christine Presentation - All	Algorithm is completely coded and results are completely generated. Try to start/finish visualization

	Presentation		part if there's time. Create and practice the presentation.
4/30 - 5/6 ***Note:*** *Presentation is on **5/1** and **5/3***	Finish Visualization if possible. Create and practice presentation. Present	Visualization - Monte, Christine Presentation - All	Visualization is finished. Presentation goes successfully, and we can deliver!

For our personal reference may or may not be included in actual proposal:

Steps in HMM modelling:

(a) choosing the genomic region (e.g., gene upstream regions) to be analyzed,

(b) using a suitable counting method (e.g., overlapping k-mers may or may not be counted),

(c) selecting an appropriate statistical background or null model for predicting expected k-mer frequencies

(d) using appropriate statistics to score the observed k-mer frequency against the expected background frequency (e.g. binomial probabilities, fold enrichment scores and Z-scores).

Dataset download website:

http://downloads.yeastgenome.org/sequence/S288C_reference/genome_releases/

Choice of learning methods:

Supervised or unsupervised :

Supervised is preferrable

Yeast species:

Saccharomyces cerevisiae