Algorithm Description

CS466 Miniproject
Edward W Huang, Hanchen Huang, Xinzhou Zhao

In our project, we utilized Gibbs sampling in order to implement our motif finder. Our algorithm's input is a set of p strings and a motif length k. We wish to find, across the p strings, the most mutually similar substring of length k.

First, we randomly initialize a set of p integers within the size of our input strings. Each integer represents a random starting position for each input string.

Then, we update each of the starting positions. To do this, for each starting position i, we build a profile matrix P using the k-length sequences at all other starting positions. Then, we score each possible sub-sequence in the current sequence using the following equation.

$$\text{Score}(x) = \sum_{i=1}^{k} \log\left(\frac{e_i(x_i)}{0.25}\right) \tag{1}$$

where $e_i(x_i)$ is the probability of observing the ith character in substring x according to the profile matrix P. 0.25 is simply the probability of observing any character at random, and is used to correct for expectation by chance (i.e., the background probability).

After scoring every possible k-length sub-sequence in the current sequence, we can arrive at a probability distribution after passing the scores through a softmax function. We can then select a new update index by sampling from this distribution. We update the starting position with this new index. We continue updating for every starting index for 1,000 iterations.

We re-initialize our starting indices every 1,000 iterations in case we get stuck in a local optimum. We do this 100 times.

The motif output by the algorithm is the profile matrix that has the highest information content score across all iterations, and all re-initializations.