Comment to previous session

It was said that SAM was a MET precursor, but in fact each can be derivated from the other. It remains true, however, that SAM is biochemically important (the main donor of methyl groups for various methylation reactions), and the MET repressor senses SAM level as part of a feedback system for control and expression of related genes.

Review of Motif Discovery approaches.

After a general discussion on Maximum Likelihood Estimators and the Expectation Maximization Algorithm, we discussed approaches to motif finding, motivated by the working innards of the gene expression and regulation mechanisms.
These tools will allow us to find regulatory motifs in biological sequence data. We will ultimately search for a maximum entropy model for the distribution under study compared to a random background. The approaches we'll review are:

- Brute Force algorithms – slow, exponential complexity growth
- Greedy Search, as described in Hertz & Stormo
- Expectation maximization
- Gibbs sampler

Brute Force

From previous experience, we may know that the motif we are searching has length l. Besides this major assumption, we may also add other constraints to our search:

1. Each sequence can contribute at most one word to the alignment
2. After each sequence has contributed exactly once to the alignment, sequences can contribute additional words to the alignment.
3. Each sequence can contribute zero or more words to the alignment.

Hertz and Stormo, Bioinformatics 15 (7): 563. (1999)

A brute force algorithm would exhaustively build a set of all possible subsequences of length l and then recursively add all the other sequences to each one. Finally, we compute the relative entropy of each and pick the best. Guaranteed to be optimal, but exponential time.

Greedy Best-First Approach

The Greedy Best-First Approach performs one comparison at each iteration – before adding any subsequence – to choose only that or those with the best relative entropy. This decreases the computational cost of the algorithm (but may miss the optimal solutions).

Expectation Maximization / MEME

As in the previous approach, EM (Bailey & Elkan, 1995) shares some of the assumptions and the problem is similarly defined:

- Inputs: The sequences $s_1, s_2, \ldots, s_k$; the motif, with length l; and the random background model
- Assumptions and constraints as in the Greedy Search, but the model can be varied to allow some flexibility, esp. regarding constraint 1.
- The EM algorithm
    - Visible data – the sequences themselves
    - Hidden data – the location of the motif in the sequences, which we will denote as

$$Y_{i,j} = \begin{cases} 1 & \text{if motif instance in sequence i begins in position j} \\ 0 & \text{otherwise} \end{cases}$$

- Model Parameters (θ): The Weight Matrix Model

The algorithm proceeds from the original WMM, refining it at every iteration, so each time the parameters maximize the likelihood of the observed data. The underlying assumption is that, given a few good matches to the best motif, we'll be able to pick out more with each successive refinement of the WMM.

Expectation Step

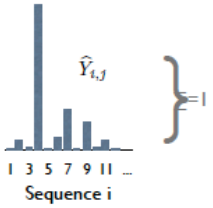
Expectation Step

(where are the motif instances?)

$$\begin{aligned} \hat{Y}_{i,j} &= E(Y_{i,j} \mid s_i, \theta^t) \\ &= P(Y_{i,j} = 1 \mid s_i, \theta^t) \\ &= P(s_i \mid Y_{i,j} = 1, \theta^t) \frac{P(Y_{i,j}=1|\theta^t)}{P(s_i|\theta^t)} \\ &= cP(s_i \mid Y_{i,j} = 1, \theta^t) \\ &= c' \prod_{k=1}^{l} P(s_{i,j+k-1} \mid \theta^t) \end{aligned}$$

$E = 0 \cdot P(0) + 1 \cdot P(1)$

Bayes



where c' is chosen so that $\sum_j \hat{Y}_{i,j} = 1$.

Because we assume that the motif can be equally present at any place in the sequence, $\frac{P(Y_{i,j} = 1 \mid \theta^t)}{P(s_i \mid \theta^t)}$ can be considered constant.

Maximization Step

At this step we try to find a WMM (θ^t) that maximizes the expected value.

Maximization Step

(what is the motif?)

Find θ maximizing expected value:

$$\begin{aligned} Q(\theta \mid \theta^t) &= E_{Y \sim \theta^t}[\log P(s, Y \mid \theta)] \\ &= E_{Y \sim \theta^t}[\log \textstyle\prod_{i=1}^{k} P(s_i, Y_i \mid \theta)] \\ &= E_{Y \sim \theta^t}[\textstyle\sum_{i=1}^{k} \log P(s_i, Y_i \mid \theta)] \\ &= E_{Y \sim \theta^t}[\textstyle\sum_{i=1}^{k} \sum_{j=1}^{|s_i|-l+1} Y_{i,j} \log P(s_i, Y_{i,j} = 1 \mid \theta)] \\ &= E_{Y \sim \theta^t}[\textstyle\sum_{i=1}^{k} \sum_{j=1}^{|s_i|-l+1} Y_{i,j} \log(P(s_i \mid Y_{i,j} = 1, \theta) P(Y_{i,j} = 1 \mid \theta))] \\ &= \textstyle\sum_{i=1}^{k} \sum_{j=1}^{|s_i|-l+1} E_{Y \sim \theta^t}[Y_{i,j}] \log P(s_i \mid Y_{i,j} = 1, \theta) + C \\ &= \textstyle\sum_{i=1}^{k} \sum_{j=1}^{|s_i|-l+1} \hat{Y}_{i,j} \log P(s_i \mid Y_{i,j} = 1, \theta) + C \end{aligned}$$

"The likelihood increases at each iteration, so the procedure will always reach a local (if not global) maximum asymptotically." Note also that we do not need to maximize at each step, as long as the likelihood increases from one iteration to the next; this allows us to use numerical methods rather than analytical approaches which may be more of a challenge to implement in a computer. (Durbin, et al p. 324)

Initialization of the Algorithm

1. Try every motif-length substring, and use it as the initial WMM with a predefined arbitrary weight for that particular sequence (rest uniform distribution)
2. Run some iterations for each set
3. Choose the best few and iterate to convergence.

The Gibbs Sampler

We still have the same overarching goal: identify the WMM that maximizes likelihood of the given data. However, unlike in previous two algorithms, the probability that a motif is at a particular position will be calculated by averaging over probabilities, instead of sampling through the sequences. Therefore we are faced with the decision of how to average. Three alternatives:

1. Direct Integration – rarely solvable analytically
2. Numerical Integration – unstable, esp. in high dimensions
3. Monte Carlo Integration – To approximate the expected value of some function f(x), where x is distributed according to some probability distribution P, we can use sampling: we draw n independent samples $x_1, x_2, \ldots, x_n$, each distributes according to P and average as $E(f(x)) \approx (1/n)*\text{sum}_i\ f(x_i)$

However, independent sampling is also often hard; luckily it is not required, as we can pick subsequent samples from the current one.

Markov Chain Monte Carlo

As in the general case, the state of a Markov Chain depends on previous states. For the MCMC, we use just the previous state: $X_{t+1} \sim P(X_{t+1} \mid X_t)$

Gibbs Sampling is the most commonly used technique. We count on being able to calculate the individual distributions. Then:

1. Choose a random starting point
2. Take a random walk through the space (can't go too far)
3. At each step we move in only one dimension of the data space and hold other variables fixed. By following this procedure, we simplify the computations needed since we remove the multidimensional relationships.

It is true that the samples are not truly independent, but we also expect that more samples will be drawn from locations around high probability points.

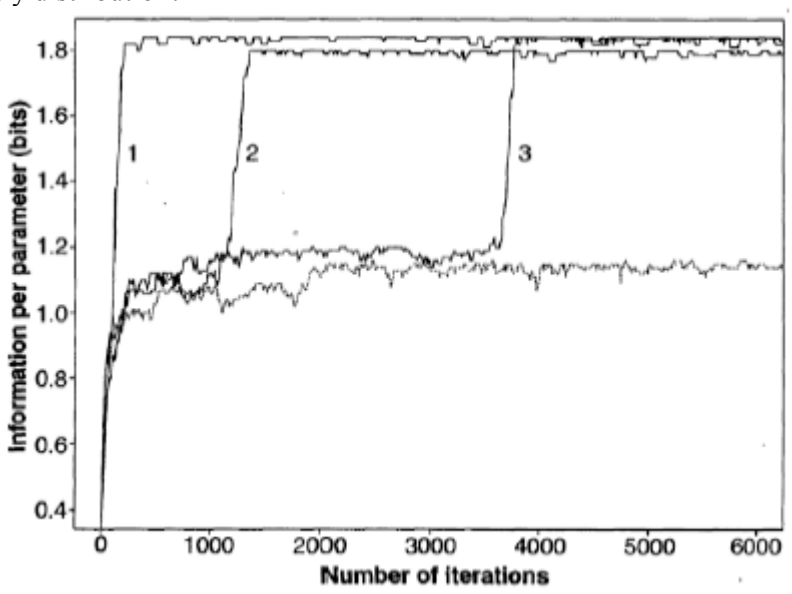
The problem is defined as:

- Inputs: The sequences $s_1, s_2, \ldots, s_k$; the motif of length l, and the model described by the WMM.
- Same general constraints as in Greedy search
- Parameters: the locations of the motifs, at $x_1, x_2, \ldots, x_k$

We proceed by building, for every sequence s_i, a WMM from all other sequences, and compute the conditional probability of the motif being present in sequence s_i at position j. At the next step, we choose a new location x_i based on the probability distribution just obtained.

Issues (from lecture slides)

- Burn-in - how long must we run the chain to reach a stationary state?
- Mixing - how long a post-burn-in sample must we take to get a good sample of the stationary distribution?



Variants and Extensions (from lecture slides)

- "Phase Shift" - may settle on suboptimal solution that overlaps part of motif. Periodically try moving all motif instances a few spaces left or right.
- Algorithmic adjustment of pattern width: Periodically add/remove flanking positions to maximize (roughly) average relative entropy per position
- Multiple patterns per string