Motif Finding

Regulation of Genes

Transcription Factor (Protein) → RNA polymerase (Protein)

DNA → Regulatory Element → Gene

Regulatory Element

Gene

Microarrays

- A 2D array of DNA sequences from thousands of genes
- Each spot has many copies of same gene
- Allow mRNAs from a sample to hybridize
- Measure number of hybridizations per spot

Finding Regulatory Motifs

Tiny Multiple Local Alignments of Many Sequences
Finding Regulatory Motifs

Given a collection of genes with common expression, Find the TF-binding motif in common

Characteristics of Regulatory Motifs

- Tiny
- Highly Variable
- ~Constant Size
- Often repeated
- Low-complexity-ish

Problem Definition

Given a collection of promoter sequences $s_1, ..., s_N$ of genes with common expression

<table>
<thead>
<tr>
<th>Probabilistic</th>
<th>Combinatorial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motif: $M_i$: $1 \leq i \leq W$ $1 \leq j \leq 4$ $M_i = \text{Prob[ letter j, pos i ]}$</td>
<td>Motif $M$: $m_1 ... m_W$ Some of the $m_i$'s blank</td>
</tr>
<tr>
<td>Find best $M$, and positions $p_1, ..., p_N$ in sequences</td>
<td>Find $M$ that occurs in all $s_i$ with $\leq k$ differences</td>
</tr>
</tbody>
</table>

Essentially a Multiple Local Alignment

- Find "best" multiple local alignment

Alignment score defined differently in probabilistic/combinatorial cases

Algorithms

- Probabilistic
  1. Expectation Maximization: MEME
  2. Gibbs Sampling: AlignACE, BioProspector
- Combinatorial
  CONSENSUS, TEIRESIAS, SP-STAR, others

Discrete Approaches to Motif Finding
Discrete Formulations

Given sequences $S = \{x_1, \ldots, x^n\}$

- A motif $W$ is a consensus string $w_1 \ldots w_K$
- Find motif $W^*$ with “best” match to $x_1, \ldots, x^n$

Definition of “best”:
\[
d(W, x) = \min \text{hamming dist. between } W \text{ and a word in } x
\]
\[
d(W, S) = \sum_i d(W, x_i)
\]

Approaches

- Exhaustive Searches
- CONSENSUS
- MULTIPROFILER, TEIRESIAS, SP-STAR, WINLOWER

Exhaustive Searches

1. Pattern-driven algorithm
   - For $W = AA \ldots A$ to $TT \ldots T$ ($4^K$ possibilities)
   - Find $d(W, S)$
   - Report $W^* = \arg \min d(W, S)$
   - Running time: $O(KN4^K)$
     (where $N = \sum_i |x_i|$)
   - Advantage: Finds provably best motif $W$
   - Disadvantage: Time

Exhaustive Searches

2. Sample-driven algorithm
   - For $W = a$ a K-long word in some $x_i$
   - Find $d(W, S)$
   - Report $W^* = \arg \min d(W, S)$
     OR Report a local improvement of $W^*$
   - Running time: $O(KN^2)$
   - Advantage: Time
   - Disadvantage: If true motif does not occur in data, and true motif is “weak”
     Then, random motif may score better than any instance of true motif

CONSENSUS (1)

Algorithm:

Cycle 1:
For each word $W$ in $S$
    For each word $W'$ in $S$
        Create alignment (gap free) of $W, W'$
        Keep the $C_1$ best alignments, $A_1, \ldots, A_{C_1}$
        $ACGGTTG$, $CAGACTT$, $GGGCTCT$ ...
        $ACGCTTG$, $AGAACTA$, $GGGCGT$ ...

CONSENSUS (2)

Algorithm (cont'd):

Cycle t:
For each word $W$ in $S$
    For each alignment $A_i$ from cycle t - 1
        Create alignment (gap free) of $W, A_i$
        Keep the $C_t$ best alignments $A_t, \ldots, A_{C_t}$
        $ACGGTTG$, $CAGACTT$, $GGGCTCT$ ...
        $ACGCTTG$, $AGAACTA$, $GGGCGT$ ...
        $ACGGCTC$, $AGATCTT$, $GGGGTCT$ ...

CONSENSUS (3)

\( C_1, \ldots, C_n \) are user-defined heuristic constants

**Running time:**

\[
O(N^2) + O(N C_1) + O(N C_2) + \ldots + O(N C_n)
\]

\[
= O(N^2 + NC_{\text{total}})
\]

Where \( C_{\text{total}} = \sum C_i \), typically \( O(nc) \), where \( C \) is a big constant

MULTIPROFILER

- Extended sample-driven approach
  
  Given a \( K \)-long word \( W \), define:
  
  \( N_a(W) = \) words \( W' \) in \( S \) s.t. \( d(W, W') \leq a \)

**Idea:**

Assume \( W \) is occurrence of true motif \( W^* \)
Will use \( N_a(W) \) to correct "errors" in \( W \)

MULTIPROFILER (2)

**Assume** \( W \) differs from true motif \( W^* \) in at most \( L \) positions

**Define:**

A wordlet \( G \) of \( W \) is a \( L \)-long pattern with blanks, differing from \( W \)

**Example:**

\( K = 7; \; L = 3 \)

\( W = \text{ACGTGGA} \)

\( G = \text{--AG--CG} \)

MULTIPROFILER (2)

**Algorithm:**

For each \( W \) in \( S \):

For \( L = 1 \) to \( L_{\text{max}} \)

1. Find all "strong" \( L \)-long wordlets \( G \) in \( N_a(W) \)
2. Modify \( W \) by the wordlet \( G \) \( \rightarrow W' \)
3. Compute \( d(W', S) \)

Report \( W^* = \arg\min d(W', S) \)

**Step 1 above:** Smaller motif-finding problem;
Use exhaustive search

Expectation Maximization (1)

- The MM algorithm, part of MEME package uses Expectation Maximization

**Algorithm (sketch):**

1. Given genomic sequences find all \( K \)-long words
2. Assume each word is motif or background
3. Find likeliest
   - Motif Model
   - Background Model
   - classification of words into either Motif or Background
Expectation Maximization (2)

- Given sequences $x^1, ..., x^i$.
- Find all $k$-long words $X_1, ..., X_n$.
- Define motif model:
  $M = (M_1, ..., M_k)$
  $M_i = (M_{i1}, ..., M_{i4})$ (assume {A, C, G, T})
  $\text{where } M_{ij} = \text{Prob[ motif position } i \text{ is letter } j ]}$

Expectation Maximization (3)

- Define
  $Z_{i1} = \begin{cases} 1, & \text{if } X_i \text{ is motif;} \\ 0, & \text{otherwise } \end{cases}$
  $Z_{i2} = \begin{cases} 0, & \text{if } X_i \text{ is motif;} \\ 1, & \text{otherwise } \end{cases}$

- Given a word $X_i = x[1]...x[k]$,
  $P( X_i, Z_{i1} = 1 ) = \lambda M_{1x[1]}...M_{kx[k]}$
  $P( X_i, Z_{i2} = 1 ) = (1 - \lambda) B_{x[1]}...B_{x[k]}$
  Let $\lambda_1 = \lambda; \lambda_2 = (1 - \lambda)$

Expectation Maximization (4)

Define:
Parameter space $\theta = (M, B)$

Objective:
Maximize log likelihood of model:
$$
\log P(X_1, ..., X_n, Z \mid \theta, \lambda) = \sum_{i=1}^{n} \sum_{j=1}^{k} Z_{ij} \log(\lambda_j P(X_i \mid \theta_j))
$$
$$
\sum_{i=1}^{n} \sum_{j=1}^{k} Z_{ij} \log P(X_i \mid \theta_j) + \sum_{i=1}^{n} \sum_{j=1}^{k} Z_{ij} \log \lambda_j
$$

Expectation Maximization (5)

- Maximize expected likelihood, in iteration of two steps:
  Expectation:
  Find expected value of log likelihood:
  $$E[ \log P(X_1, ..., X_n, Z \mid \theta, \lambda) ]$$
  Maximization:
  Maximize expected value over $\theta, \lambda$

Expectation Maximization (6): E-step

Expectation:
Find expected value of log likelihood:
$$E[ \log P(X_1, ..., X_n, Z \mid \theta, \lambda) ] =$$
$$\sum_{i=1}^{n} \sum_{j=1}^{k} E[Z_{ij}] \log P(X_i \mid \theta_j) + \sum_{i=1}^{n} \sum_{j=1}^{k} E[Z_{ij}] \log \lambda_j$$

where expected values of $Z$ can be computed as follows:
$$Z_{ij} = \frac{\lambda_j P(X_i \mid \theta_j)}{\sum_{k=1}^{k} \lambda_k P(X_i \mid \theta_k)}$$

Expectation Maximization (7): M-step

Maximization:
Maximize expected value over $\theta$ and $\lambda$ independently

For $\lambda_j$, this is easy:
$$\lambda_{j_{\text{new}}} = \arg \max_{\lambda_j} \sum_{i=1}^{n} E[Z_{ij}] \log \lambda_j = \frac{\sum_{i=1}^{n} Z_{ij}}{n}$$
Expectation Maximization (8): M-step

- For \( \theta = (M, B) \), define
  
  \[ c_{jk} = E[ \text{# times letter } k \text{ appears in motif position } j] \]

  \[ c_{0k} = E[ \text{# times letter } k \text{ appears in background}] \]

  It easily follows:

  \[ \sum_{j=1}^{J_{\text{new}}} \frac{c_{jk}}{c_{0k}} = \frac{M_{\text{new}}}{\sum_{j=1}^{J_{\text{new}}} c_{0k}} \]

  to not allow any 0's, add pseudocounts

Overview of EM Algorithm

1. Initialize parameters \( \theta = (M, B), \lambda \):  
   - Try different values of \( \lambda \) from \( N^{\frac{-1}{2}} \) to \( 1/(2K) \)

2. Repeat:
   a. Expectation
   b. Maximization

3. Until change in \( \theta = (M, B), \lambda \) falls below \( \epsilon \)

4. Report results for several "good" \( \lambda \)

Initial Parameters Matter!

Consider the following "artificial" example:

- \( x_1, ..., x_N \) contain:
  - \( 2^k \) patterns A…A, A…AT, …, T…T
  - \( 2^k \) patterns C…C, C…CG, …, G…G
  - \( D \ll 2^k \) occurrences of K-mer ACTG…ACTG

Some local maxima:

- \( \lambda = \frac{1}{2}; \quad B = \frac{1}{2}C, \frac{1}{2}G; \quad M_i = \frac{1}{2}A, \frac{1}{2}T, i = 1, ..., K \)

- \( \lambda = D^{2^{-1/2}}; \quad B = \frac{1}{4}A, \frac{1}{4}C, \frac{1}{4}G, \frac{1}{4}T; \quad M_1 = 100\% A, M_2 = 100\% C, M_3 = 100\% T, etc. \)

Conclusion

- One iteration running time: \( O(NK) \)
  - Usually need \( < N \) iterations for convergence, and \( < N \) starting points
  - Overall complexity: unclear - typically \( O(N^2 K) \) - \( O(N^3 K) \)

- EM is a local optimization method
- Initial parameters matter