Motif Discovery

Pål Sætrom

Topics
- Introduction – What controls co-expressed genes?
- Motifs
- Motif-discovery – exact methods
- Motif-discovery – randomized methods

Microarrays measure global gene expression
1. Isolate RNA
2. Reverse-transcribe to cDNA
3. Label cDNA
4. Hybridize
5. Scan
6. Analyze

Microarrays measure global gene expression (2)
- Array of probes measure gene RNA levels
- "All" genes in one sample
- Microarrays identify co-expressed genes

Which genes are affected by our inner clock?
1. Measure gene expression every 4 hours for 2 days
2. Identify cycling genes (cosine pattern, $T = 20$-28h)
3. What controls (evening) cycling genes’ expression?

Transcription regulation
- Proteins (transcription factors - TFs) bind short DNA segments
  - Initiate transcription (core promoters)
  - Activate transcription (enhancers)
  - Repress transcription (silencers)
- Location of sequence elements varies
  - Core promoter close to transcription start site
  - Enhancers and silencers many kb away
- Problem: Identify DNA elements causing co-expression
**Motif finding problem**

- Find motif common to genes with common expression patterns

**Input:** Set of genes, G  
**Output:** Motif common to all genes in G (and significant)

**Easy?**

**Related: common domain in set of proteins**

“Zinc finger” domains bind DNA and RNA

**Prosite patterns define protein motifs**

- Regular expressions
  - Concatenations: -
  - Alternatives: [ ]
  - Wildcards: x(l, h)
  - Fixed repeats: x(l, h)

- Model the dominant positions in the domain

**Patterns in DNA**

- Myc is a transcription factor  
  - “leucine zipper”  
  - “helix-loop-helix”  
  - Dimer with Max  

**Identifying motifs by counting l-mers**

1. Create index of all l-mers  
2. Motifs are l-mers that occur in all (most) sequences  
3. Compare actual occurrences with reference (negative control) set

**Binding sites for Myc are ambiguous**

- Consensus string too strict (3 of 7)  
- Pattern too loose  
  - C[AG][CG][TAC]G  
  - CGGGAG, CGGGCG

**Solution?**
Profiles are weighted sequence patterns

- Position weight matrix (PWM)
- Position-specific scoring matrix (PSSM)
- Position-specific weight matrix (PSWM)

Different alignments give different PWMs

Length and set of positions define PWM

- Set of $t$ DNA sequences with $n$ nucleotides each
- PWM length $l$
- PWM given by $s = (s_1, s_2, ..., s_t)$, $1 \leq s_i \leq n - l + 1$

PWM scores define “best” PWM

- Consensus score:
  \[ \text{Score}(s) = \sum \max_j M_{p,j}(s) \]
  - $P(s)$ is profile matrix (PWM)
  - $M_{p,j}(s)$ is max in column $j$ in PWM
- Entropy:
  \[ \text{Entropy}(s) = \sum \sum \frac{P_{i,j}}{\max_j M_{p,j}} \log \frac{P_{i,j}}{\max_j M_{p,j}} \]
  - $P_{i,j}$ is count at element $(i,j)$ in PWM

Using consensus score to find best PWM

Motif finding problem (feasible definition)

- Given set of DNA sequences, find $l$-length PWM that maximizes consensus score

Input: A $t \times n$ matrix of DNA, and the length $l$
Output: An array of $t$ starting positions $s = (s_1, s_2, ..., s_t)$ maximizing $\text{Score}(s)$

Easy?
Consensus score = counting mismatches with consensus string

- Max consensus score: \( lt \)
- Min consensus score: \( lt/4 \)

\[
\text{Score}(s) = \sum_{i=1}^{t} M_{n,i}(i) - lt - d_H(w,s) - lt - \sum_{i=1}^{t} d_H(w,s_i)
\]

- \( w \) is consensus string of PWM
- \( d_H(w,s_i) \) is Hamming distance between \( w \) and \( s_i \)

Solving motif finding = finding string with minimum total Hamming distance

Median string problem

- Given set of DNA sequences, find \( l \)-mer median string

Input: A \( t \times n \) matrix of DNA, and the length \( l \)
Output: A string \( v \) of \( l \) nucleotides (\( l \)-mer) with minimal total Hamming distance \( d_H(v,s) \) of all possible \( l \)-mers

Easier?

Naïve motif finding and median string solutions

- Motif finding
  - Find optimal starting positions \( s = (s_1, s_2, \ldots, s_t) \)
  - Consider all \( (n - l + 1)^t \) starting positions
- Median string
  - Find optimal \( l \)-mer
  - Consider all \( 4^t \) \( l \)-mers (DNA)

Brute force motif finding

\text{BruteForceMotifSearch}(t, n, l):
1. bestScore = 0
2. for each \( (s_1, s_2, \ldots, s_t) \) from \( (1, \ldots, 1) \) to \( (n - l + 1, \ldots, n - l + 1) \):
   1. if Score\((s)\) > bestScore:
      1. bestScore = Score\((s)\)
      2. bestMotif = \((s_1, s_2, \ldots, s_t)\)
3. return bestMotif

\( O((n - l + 1)^t) \times O(l) = O(nt^4) \)

Brute force median string

\text{BruteForceMedianString}(t, n, l):
1. bestWord = AA...AA
2. bestDistance = \( \infty \)
3. for each \( l \)-mer word from A...A to T...T:
   1. if \( d_H(word,s) < bestDistance \):
      1. bestScore = Score\((s)\)
      2. bestWord = word
4. return bestWord

\( O(4^t) \times O(nt) = O(nt4^t) \)

Using tree to implement for loop

- Store starting positions in leaves
  - \( (n - l + 1)^t \) starting positions
  - \( 'l\)-mer” in alphabet of \( (n - l + 1) \) symbols
- Traverse tree (ignoring internal nodes)

\text{NextLeaf}(a, L, k):
1. for \( i \) in range\((L, 1, -1)\):
   1. if \( a_i < k \):
      1. \( a_i = a_i + 1 \)
   2. return \( a \)
2. \( a = k \)
2. return \( a \)
Using search tree to skip unproductive branches

- Total score is bounded by score for \( i \)th first starting positions

\[
\text{Score}(s) \leq \text{Score}(s, i) + (t - i) \cdot l
\]

\( \text{NextVertex}(a, i, L, k): \)
1. if \( i < L \):
   1. \( a_{i+1} = 1 \)
   2. return \((a, i+1)\)
2. else:
   1. for \( j \) in range\((L, 1, -1)\):
      1. if \( a_j < k \):
         1. \( a_j = a_j + 1 \)
         2. return \((a, j)\)
   3. return \((a, 0)\)

Skip unproductive branches

\( \text{Bypass}(a, i, L, k): \)
1. for \( j \) in range\((i, 1, -1)\):
   1. if \( a_j < k \):
      1. \( a_j = a_j + 1 \)
      2. return \((a, j)\)
2. return \((a, 0)\)

Branch and bound motif finding

\( \text{BranchAndBoundMotifSearch}(t, n, l): \)
1. \( s = (1, \ldots, 1) \)
2. bestMotif = \( s \)
3. bestScore = 0
4. \( i = 1 \)
5. while \( i > 0 \): 
   1. if \( i < t \):
      1. optimisticScore = Score\((s, i)\) + \((t - i) \cdot l\)
      2. if optimisticScore < bestScore:
         1. \((s, i) = \text{Bypass}(s, i, t, n - l + 1)\)
      3. else:
         1. \((s, i) = \text{NextVertex}(s, i, t, n - l + 1)\)
   2. else:
      1. if \( \text{Score}(s) > \text{bestScore} \):
         1. bestScore = \( \text{Score}(s) \)
         2. bestMotif = \((s_1, s_2, \ldots, s_t)\)
         2. \((s, i) = \text{NextVertex}(s, i, t, n - l + 1)\)
6. return bestMotif

Conservative bound on median string

- Internal nodes represent word prefixes
- Skip subtrees where prefix has higher distance than current best
- Might be some extension with 0 distance

5.1.1: \( \text{optimisticDistance} = d_p(\text{word}, s) \)

A greedy approach to motif discovery

\( \text{GreedyProfileMotifSearch}(t, n, l): \)
1. Randomly select starting positions \( s = (s_1, s_2, \ldots, s_t) \)
2. Form profile \( P \) from \( s \)
3. bestScore = 0
4. while \( \text{Score}(s) > \text{bestScore} \):
   1. bestScore = \( \text{Score}(s) \)
   2. for \( i \) in range\((t)\):
      1. Find the highest scoring \( l \)-mer \( a \) from \( i \)th sequence
      2. \( s_i = \text{starting position of } a \)
5. return \( s \)

Greedy (gradient descent) optimization

- Optimization in energy landscapes

- Gradient descent will find local minima
  - \( \text{GreedyProfileMotifSearch} \):
    1. Find the highest scoring \( l \)-mer \( a \) from \( i \)th sequence
    2. \( s_i = \text{starting position of } a \)
- Need means to escape
- How?
Allow moves “uphill”

- From statistical mechanics
  - State, $S$
  - Probability of finding a system in state $S$, $P(S)$
- Simulating system of discrete states
  - Allow suboptimal state $S$ with probability $\sim P(S)$
  - Focus on neighboring states to narrow search

Gibbs sampling (in general)

1. Choose random starting state $S$
2. Choose random neighbor state $S'$
3. If $P(S') \leq P(S)$:
   1. $S = S'$
4. Else
   1. $\Delta E = P(S') - P(S)$
   2. Set $S = S'$ with probability $e^{-\Delta E / (kT)}$
5. Go to 2

Energy landscapes and Gibbs sampling

Gibbs sampler for motif discovery

GibbsProfileMotifSearch($t, n, l$):
1. Randomly select starting positions $s = (s_1, s_2, \ldots, s_t)$
2. Start from random sequence
3. Form profile $P$ from $s$
4. For each position $j$ in $s$:
   1. Calculate probability $p_j$ of $l$-mer starting at position $j$ being from $P$
5. Randomly choose new starting position in $s$ based on $(p_1, \ldots, p_{n-l+1})$
6. Go to 2

Computing probability of $l$-mer from $P$

PWM (counts):

<table>
<thead>
<tr>
<th></th>
<th>A 1 0 0 0 0 1 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C 2 7 0 0 0 1 0 0</td>
</tr>
<tr>
<td></td>
<td>G 4 0 1 1 0 0 7 6</td>
</tr>
<tr>
<td></td>
<td>T 0 0 0 0 5 0 1 0</td>
</tr>
</tbody>
</table>

$P(x, P) = \prod_{i=1}^{l} P_{ix}$

$P(\text{ACACGTGG}, P) = 1/7 \ast 1/6 \ast 1/6 \ast 1/7 \ast 1/5 \ast 1/7 \ast 1/5 \ast 6/7$
$P(\text{ACGGGAGT}, P) = 1/7 \ast 1/7 \ast 1/7 \ast 1/7 \ast 1/7 \ast 1/7 \ast 1/7 \ast 1/7$
$P(\text{TCACGTGG}, P) = 0$
$P(\text{TGGAGGT}, P) = 0$ (!)

Alternative target functions

- Entropy
  \[ \text{Entropy}(s) = -\sum_{i=1}^{l} \frac{p_i}{n} \log \frac{p_i}{n} \]
- Relative entropy
  \[ \text{RelativeEntropy}(s) = \sum_{i=1}^{l} \frac{p_i}{n} \log \frac{p_i}{p_j} \]
  \[ e^{-\Delta E / (kT)} \]
Consensus likely infrequent

...but consensus with mismatches likely frequent
- $xCxCGxGG$ (5 of 7)
- $xCAXGxGG$ (5 of 7)
- $xCACGXGxG$ (5 of 7)

- Index random ($k$ of $l$)-mers
- Remember frequent positions (in DNA)
- Use most frequent positions

<table>
<thead>
<tr>
<th></th>
<th>$l$-mer Counting revisited</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C$</td>
<td>$A$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$C$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$G$</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>$T$</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$G$</td>
<td>0</td>
</tr>
</tbody>
</table>