Motif Discovery

Pål Sætrom
Eivind Coward

Topics

- Introduction: Motifs in DNA sequences
- Position Weight Matrices
- Motif-discovery – exact methods
- Motif-discovery – randomized methods

Microarrays measure global gene expression

- Array of probes measure gene RNA levels
- "All" genes in one sample
- Microarrays identify co-expressed genes

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)

Related: common domain in set of proteins

"Zinc finger" domains bind DNA and RNA

Formal description of domain?
Prosite patterns define protein motifs

- Regular expressions
  - Concatenations: 
  - Alternatives: []
  - Wildcards: x
  - Fixed repeats: x(l, h)
- Model the dominant positions in the domain
- Limitations?

Patterns in DNA

- Myc is a transcription factor
  - "leucine zipper"
  - "helix-loop-helix"
  - Dimer with Max
- Binds DNA
  - CACGTG
- Find the binding sites?

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

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

Profiles are weighted sequence patterns

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

Binding sites for Myc are ambiguous

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

Different alignments give different PWMs

- Last left by 1
  - Best PWM?
- Last right by 1
  - Best PWM?
Length and set of positions define PWM
- Set of $t$ DNA sequences with $n$ nucleotides each
- PWM length $l$
- PWM given by $s = (s_1, s_2, \ldots, s_t), 1 \leq s_i \leq n - l + 1$

PWM scores define “best” PWM
- Consensus score:
  \[ \text{Score}(s) = \sum_{j=1}^{l} \max(P_{ij}) \]
  - $P(s)$ is profile matrix (PWM)
  - $M_j(i)$ is max in column $j$ in PWM
- Entropy:
  \[ \text{Entropy}(s) = -\sum_{j=1}^{l} \sum_{i=1}^{n} \frac{p_{ij}}{t} \log \frac{p_{ij}}{t} \]
  - $p_{ij}$ is count at element $(i,j)$ in PWM

Entropy measures degree of disorder
- Plot of entropy for a distribution of two symbols with frequency $f$ and $1-f$:
  \[ H = -\sum_{j=1}^{2} f_j \log f_j - f \log f - (1-f) \log (1-f) \]
- We can also define information content as $R = 1 - H$.

Sequence logos
- A sequence logo (Schneider & Stephens 1990) is a popular visualisation of a PWM motif.
- The relative height of each letter is proportional to its frequency in the PWM.
- The total height is proportional to the information content of each position:
  \[ R_j = 2 - H_j = 2 \sum \frac{P_{ij}}{t} \log \frac{P_{ij}}{t} \]

PWM represented as a profile HMM
- Consensus score:
  \[ \text{Score}(s) = \sum_{j=1}^{l} \max(P_{ij}) \]
  - $P(s)$ is profile matrix (PWM)
  - $M_j(i)$ is max in column $j$ in PWM
- Entropy:
  \[ \text{Entropy}(s) = -\sum_{j=1}^{l} \sum_{i=1}^{n} \frac{p_{ij}}{t} \log \frac{p_{ij}}{t} \]
  - $p_{ij}$ is count at element $(i,j)$ in PWM

Using consensus score to find best PWM
- Last left by $1$
- Last right by $1$
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 Score(s)

_Easy?_

Consensus score = counting mismatches with consensus string
- Max consensus score: \( lt \)
- Min consensus score: \( lt/4 \)

**Score(s) =**
\[
\sum_{i=1}^{n} M_{x,j}(i) - t \cdot d_H(w, s_i) - t \cdot \sum_{i=1}^{t} d_H(w, s_i)
\]
- \( w \) 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 \( s = (s_1, s_2, ..., s_t) \)
  - Consider all \((n-l+1)^t\) starting positions
- Median string
  - Find optimal \( l \)-mer
  - Consider all \( 4^l \)-mers (DNA)

Brute force motif finding

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

**O((n-l+1)^t) \cdot O(l) = O(lnt^t)**

Brute force median string

**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^l) \cdot O(ntl) = O(ntl4^l)**
Using tree to implement for loop

- Store starting positions in leaves
  - \((n - l + 1)!\) starting positions
  - "t-mer" in alphabet of \((n - l + 1)\) symbols
- Traverse tree (ignoring internal nodes)

\[
\text{NextLeaf}(a, L, k):
1. \text{for } i \in \text{range}(L, 1, -1):
   1. \text{if } a_i < k:
      1. \ a_i = a_i + 1
      2. \text{return } a
   2. \ a_i = 1
   2. \text{return } a
\]

Using search tree to skip unproductive branches

- Total score is bounded by score for \(i\)th first starting positions
  
  \[
  \text{Score}(s) \leq \text{Score}(s, i) + \{(t - i) \cdot l\}
  \]
  
  All remaining positions = current consensus

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

Skip unproductive branches

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

Branch and bound motif finding

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

Conservative bound on median string

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

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

A greedy approach to motif discovery

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

Characteristics? \(n\) vs. \(t\)?
Greedy (gradient descent) optimization

- Optimization in energy landscapes
- Gradient descent will find local minima
  - 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

“Downhill” GD/KH

“Uphill” LA/SA

“Simulated annealing” feature

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

$P(\alpha, P) = \prod_{l} p_{\alpha l}$

$P(\text{ACACGTGG}, P) = 1/7 \times 1/6 \times 7/8 \times 1/5 \times 7/1 \times 6/7$
$P(\text{ACGGGAGT}, P) = 1/7 \times 1/6 \times 7/8 \times 1/5 \times 7/1 \times 6/7$
$P(\text{TCACGTGG}, P) = 0$
$P(\text{TCCGGAGT}, P) = 0$ (!)

Alternative target functions

- Entropy
  $$\text{Entropy}(\alpha) = -\sum_{l} p_{\alpha l} \log \frac{p_{\alpha l}}{p_{l}}$$
- Relative entropy
  $$\text{RelativeEntropy}(\alpha) = \sum_{l} p_{\alpha l} \log \frac{p_{\alpha l}}{p_{l}}$$
  $$e^{-\Delta E / (kT)}$$
Use multiple starting/solution points

- “Random restart gradient descent”
  - Run GreedyProfileMotifSearch $x$ times