Gibbs Sampling in Motif Finding

Gibbs Sampling (1)

- **Given:**
  - \( x_1, \ldots, x^n \)
  - motif length \( K \)
  - background \( B \)
- **Find:**
  - Model \( M \)
  - Locations \( a_1, \ldots, a_n \) in \( x_1, \ldots, x^n \)

Maximizing log-odds likelihood ratio:

\[
\sum_{i=1}^{N} \sum_{k=1}^{K} \log \frac{M(k, x'_{a_{i+k}})}{B(x'_{a_{i+k}})}
\]

Gibbs Sampling (2)

- AlignACE: first statistical motif finder
- BioProspector: improved version of AlignACE

Algorithm (sketch):

1. **Initialization**
   a. Select random locations in sequences \( x_1, \ldots, x^n \)
   b. Compute an initial model \( M \) from these locations
2. **Sampling Iterations**
   a. Remove one sequence \( x_i \)
   b. Recalculate model
   c. Pick a new location of motif in \( x_i \) according to probability the location is a motif occurrence

Gibbs Sampling (3)

**Initialization:**
- Select random locations \( a_1, \ldots, a_n \) in \( x_1, \ldots, x^n \)
- For these locations, compute \( M \):

\[
M_{ij} = \frac{1}{N} \sum_{i=1}^{N} (x_{i+k} = j)
\]

- That is, \( M_{ij} \) is the number of occurrences of letter \( j \) in motif position \( k \) over the total

Gibbs Sampling (4)

**Predictive Update**

- Select a sequence \( x = x^1 \)
- Remove \( x_i \), recompute model:

\[
M_{ij} = \frac{1}{(N-1)+B} (\beta_j + \sum_{a_{i+k}} (x_{a_{i+k}} = j))
\]

where \( \beta_j \) are pseudocounts to avoid \( 0 \), and \( B = \sum \beta_j \)

Gibbs Sampling (5)

**Sampling:**

For every \( K \)-long word \( x_j \ldots x_{j+k-1} \in X \):

\[
Q_j = \text{Prob}[\text{word} | \text{motif}] = M(1, x_j \ldots x_{j+k})
\]
\[
P_j = \text{Prob}[\text{word} | \text{background}] = B(x_j \ldots x_{j+k})
\]

Let

\[
A_j = \frac{Q_j}{P_j} \quad \text{Prob} \quad \sum_{j=1}^{N} \frac{Q_j}{P_j}
\]

Sample a random new position \( a \), according to the probabilities \( A_j \).
Gibbs Sampling (6)

Running Gibbs Sampling:
1. Initialize
2. Run until convergence
3. Repeat 1, 2 several times, report common motifs

Advantages / Disadvantages

- Very similar to EM

Advantages:
- Easier to implement
- Less dependent on initial parameters
- More versatile, easier to enhance with heuristics

Disadvantages:
- More dependent on all sequences to exhibit the motif
- Less systematic search of initial parameter space

Repeats, and a Better Background Model

- Repeat DNA can be confused as motif
  - Especially low-complexity CACACA... AAAAA, etc.

Solution:
more elaborate background model
0th order: \( B = \{ p_A, p_C, p_G, p_T \} \)
1st order: \( B = \{ P(A|A), P(A|C), \ldots, P(T|T) \} \)
... Kth order: \( B = \{ P(X | b_1 \ldots b_K); X, b_i \in \{A,C,G,T\} \} \)

Has been applied to EM and Gibbs (up to 3rd order)

Example Application: Motifs in Yeast

Group:
Tavazoie et al. 1999, G. Church’s lab, Harvard

Data:
- Microarrays on 6,220 mRNAs from yeast Affymetrix chips (Cho et al.)
- 15 time points across two cell cycles

Processing of Data

1. Selection of 3,000 genes
   Genes with most variable expression were selected
2. Clustering according to common expression
   - K-means clustering
   - 30 clusters, 50-190 genes/cluster
   - Clusters correlate well with known function
3. AlignACE motif finding
   - 600 long upstream regions
   - 50 regions/trial

Motifs in Periodic Clusters
Motifs in Non-periodic Clusters

Phylogenetic Footprinting

(Slides by Martin Tompa)

Functional sequences evolve slower than nonfunctional ones

- Consider a set of orthologous sequences from different species
- Identify unusually well conserved regions

Substring Parsimony Problem

Given:
- phylogenetic tree \( T \),
- set of orthologous sequences at leaves of \( T \),
- length \( k \) of motif
- threshold \( d \)

Problem:
- Find each set \( S \) of \( k \)-mers, one \( k \)-mer from each leaf, such that the "parsimony" score of \( S \) in \( T \) is at most \( d \).

This problem is NP-hard.

Small Example

Solution

Size of motif sought: \( k = 4 \)
An Exact Algorithm

$W_u[s] = \min_{v: \text{child of } u} (W_v[t] + d(s, t))$

Recurrence

$W_u[s] = \sum_{v: \text{child of } u} \min_{t} (W_v[t] + d(s, t))$

Running Time

$W_u[s] = \sum_{v: \text{child of } u} \min_{t} (W_v[t] + d(s, t))$

$O(k \cdot 4^k)$

time per node

Limits of Motif Finders

- Given upstream regions of co-regulated genes:
  - Increasing length makes motif finding harder - random motifs clutter the true ones
  - Decreasing length makes motif finding harder - true motif missing in some sequences
Limits of Motif Finders

A (k,d) motif is a k-long motif with d random differences per copy.

Motif Challenge problem:
Find a (15,4) motif in N sequences of length L.

CONSENSUS, MEME, AlignACE, & most other programs fail for N = 20, L = 1000.

RNA Secondary Structure

RNA and Translation

RNA and Splicing

Hairpin Loops
Stems
Bulge loop
Multi-branched loop
Modeling RNA Secondary Structure: Context-Free Grammars

A Context Free Grammar

S → AB
A → aAc | a
B → bBd | b

Derivation:
S → AB → aAcB → ... → aaaacccccB → ... → aaaacccbbbbbbddd

Produces all strings $a^i c^j b^k d^l$, for $i, j, k, l > 0$

Example: modeling a stem loop

S → a W₁ u
W₁ → c W₂ g
W₂ → g W₃ c
W₃ → g L c
L → agucg

ACGG AG UGCC CG

What if the stem loop can have other letters in place of the ones shown?

Example: modeling a stem loop

S → a W₁ u
W₁ → g W₂ c
W₂ → g W₃ c
W₃ → g L c
W₄ → a Lu
L → agucg | agccg | cuguc

ACGG AG UGCC CG

More general: Any 4-long stem, 3-5-long loop:

S → a W₁ u | g W₂ u
W₁ → c W₂ g
W₂ → g W₃ c
W₃ → g L c | a L u
L → agucg | agccg | cuguc

ACGG AG UGCC CG

Example: modeling a stem loop

S → a W₁ u | g W₂ u
W₁ → c W₂ g
W₂ → g W₃ c
W₃ → g L c
W₄ → a Lu
L → agucg | agccg | cuguc

ACGG AG UGCC CG

More general: Any 4-long stem, 3-5-long loop:

S → a W₁ u | g W₂ u
W₁ → c W₂ g
W₂ → g W₃ c
W₃ → g L c | a L u
L → agucg | agccg | cuguc

ACGG AG UGCC CG

Example: modeling a stem loop

S → a W₁ u | g W₂ u
W₁ → c W₂ g
W₂ → g W₃ c
W₃ → g L c
W₄ → a Lu
L → agucg | agccg | cuguc

ACGG AG UGCC CG

More general: Any 4-long stem, 3-5-long loop:

S → a W₁ u | g W₂ u
W₁ → c W₂ g
W₂ → g W₃ c
W₃ → g L c | a L u
L → agucg | agccg | cuguc

ACGG AG UGCC CG
Alignment scores for parses!

We can define each rule \( X \rightarrow s \), where \( s \) is a string, to have a score.

**Example:**
- \( W \rightarrow a W' u : 3 \) (forms 3 hydrogen bonds)
- \( W \rightarrow g W' c : 2 \) (forms 2 hydrogen bonds)
- \( W \rightarrow g W' u : 1 \) (forms 1 hydrogen bond)
- \( W \rightarrow x W' z : -1 \), when \( (x, z) \) is not an a/u, g/c, g/u pair

**Questions:**
- How do we best align a CFG to a sequence? (DP)
- How do we set the parameters? (Stochastic CFGs)

The Nussinov Algorithm

Given sequence \( X = x_1 \ldots x_N \),

Define DP matrix:

\[
F(i, j) = \text{maximum number of bonds if } x_i \ldots x_j \text{ folds optimally}
\]

**Two cases, if } i < j:**

1. \( x_i \) is paired with \( x_j \)
   \[
   F(i, j) = s(x_i, x_j) + F(i+1, j-1)
   \]
2. \( x_i \) is not paired with \( x_j \)
   \[
   F(i, j) = \max\{ k : i \leq k < j \} F(i, k) + F(k+1, j)
   \]

**Initialization:**

\[
F(1, i-1) = 0; \quad \text{for } i = 2 \text{ to } N
\]
\[
F(i, 1) = 0; \quad \text{for } i = 1 \text{ to } N
\]

**Iteration:**

For \( i = 2 \) to \( N \):

For \( k = 1 \) to \( N - 1 \):

\[
F(k+1, j) + s(x_k, x_j)
\]

\[
F(i, j) = \max\{ k : i \leq k < j \} F(i, k) + F(k+1, j)
\]

**Termination:**

Best structure is given by \( F(1, N) \)

(Need to trace back; refer to the Durbin book)

The Nussinov Algorithm and CFGs

Define the following grammar, with scores:

\[
S \rightarrow a S u : 3 \quad | \quad u S a : 3
\]
\[
g S c : 2 \quad | \quad c S g : 2
\]
\[
g S u : 1 \quad | \quad u S g : 1
\]
\[
S S : 0 \quad | \quad a S : 0 \quad | \quad c S : 0 \quad | \quad g S : 0 \quad | \quad u S : 0 \quad | \quad \epsilon : 0
\]

**Note:** \( \epsilon \) is the "string"

Then, the Nussinov algorithm finds the optimal parse of a string with this grammar.
**The Nussinov Algorithm**

- **Initialization:**
  \[ F(i, i-1) = 0; \text{ for } i = 2 \text{ to } N \]
  \[ F(i, i) = 0; \text{ for } i = 1 \text{ to } N \]
  \[ S \rightarrow a | c | g | u \]

- **Iteration:**
  For \( l = 2 \) to \( N \):
    - For \( i = 1 \) to \( N - l \)
      - \( j = i + l - 1 \)
      - \[ F(i+1, j-1) + s(x_i, x_j) \]
      - \[ F(i, j) = \max \left\{ \max \{ i \leq k < j \} F(i, k) + F(k+1, j) \right\} \]

- **Termination:**
  Best structure is given by \( F(1, N) \)

---

**Stochastic Context Free Grammars**

In an analogy to HMMs, we can assign probabilities to transitions:

- Given grammar
  \[ X_i \rightarrow s_{i1} | ... | s_{in} \]
  \[ ... \]
  \[ X_m \rightarrow s_{m1} | ... | s_{mn} \]

- Can assign probability to each rule, s.t.
  \[ P(X_i \rightarrow s_{i}) + ... + P(X_i \rightarrow s_{in}) = 1 \]

---

**Example**

- \( S \rightarrow a S b : \frac{1}{6} \)
  - \( a : \frac{1}{4} \)
  - \( b : \frac{1}{4} \)

- Probability distribution over all strings \( x \):
  - \( x = a^i b^{n-i} \),
    
  \[ P(x) = 2^{-n} \times \frac{1}{4} = 2^{-(n+2)} \]
  - \( x = a^n b^n \)
    
  same
  
  Otherwise: \( P(x) = 0 \)