Clarification on logistic regression and tree-based multiple alignment

2. Progressive alignment for M.A
   Tree approach (using multiple pairwise comparisons)
   1. Compute pairwise distances between all seqs
   2. Create a tree using hierarchical clustering, and computing weights based on distance from root.
   3. Align closest pair, using weights to reflect number of sequences contributing to each member of pair
   4. When at top of tree, should have all aligned.

See Figure 4.8 in Mount

Tree method for MA
1. Create an ancestral tree using pairwise distances
2. Merge sequences to find ancestor seqs BY FINDING SEQUENCE WITH MINIMUM EDIT DISTANCE TO THE TWO CHILDREN SEQUENCES.
3. Assign weights to each branch of tree, based on distance between sequences (see next slide)
4. Align sequences (starting from the closest) using weights in the score function (see next next slide)

Think about match matrices

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>R</th>
<th>N</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>-2</td>
<td>6</td>
<td>0</td>
<td>-1</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-4</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>-1</td>
<td>2</td>
<td>4</td>
<td>-5</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>-2</td>
<td>-4</td>
<td>-4</td>
<td>-5</td>
<td>12</td>
<td>-5</td>
</tr>
<tr>
<td>Q</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>-5</td>
<td>4</td>
</tr>
</tbody>
</table>

Tree-based weighting

\[ F(S) = \frac{1}{D} \]

\[ D = 0.2 \]
\[ D = 0.3/2 = 0.35 \]

\[ D = F(S) = 0.5 \]
\[ D = 0.1 \]
\[ D = 0.3/2 = 0.25 \]

\[ D = 0.5 \]

Linear logistic regression is the simplest two-class version of logistic regression and fits the feature vectors into a log odds (logit) function:

\[ \log \left( \frac{p}{1-p} \right) = \beta x \]

with some manipulation:

\[ p = \frac{e^{\beta x}}{1 + e^{\beta x}} \]

where \( p \) is the probability of seeing an abbreviation, \( \beta \) is the feature vector, and \( x \) is the vector of weights. Thus, training this model consists of finding the \( \beta \) vector that maximizes the difference between the two classes.

Training this model is finding the optimal \( \beta \) by maximizing the likelihood over all the training examples using:

\[ L(\beta) = \sum_{i=1}^{n} [y_i \log p_i + (1 - y_i) \log (1 - p_i)] \]

where \( y_i \) is 1 if training example \( i \) is in one class and 0 otherwise.

There is no known closed form solution to this equation, so Newton’s method is often used to optimize this equation to a global maximum. Conventionally, \( \beta \) is initialized to zero, and although this is not...
Consensus with minimum edit distance

1. If exact match, accept
2. If inexact, place AA where sum of Match-matrix distances to the two characters is minimized.

A V K D C A
I V H – C V
-----------
L V N – C V  (example not based on PAM250)

Weighting an alignment

Sequence A (weight a) ..K..
Sequence B (weight b) ..I..
Align to
Sequence C (weight c) ..L..
Sequence D (weight d) ..V..
Score for matching these two columns =

\[
\frac{a \times c \times \text{SCORE}(K,L) + a \times d \times \text{SCORE}(K,V) + b \times c \times \text{SCORE}(I,L) + b \times d \times \text{SCORE}(I,V)}{4}
\]

Computing with sequence motifs

Russ B. Altman
BMI 214
CS 274

What is a sequence motif?

A subsequence (substring) that occurs in multiple sequences with a biological importance.

Motifs can be totally constant or have variable elements.

Motifs often result from structural features (e.g., binding heme group in globins). Note: not all amino acids are oriented to affect the binding to the heme group.
Simple conserved motif

Two globin motifs:
FLASDFTGAAMTWGKALVALH
FFSDAWAGPTMVIGRGILMPH

After looking at a number of globin motifs:
FFSDAWAGPTMVIGRGILMPH

Globin sequence signature (PROSITE PDOC00933):
F-[LF]-x(5)-G-[PA]-x(4)-G-[KRA]-x-[LIVM]-x(3)-H

A regular expression

[ ] = choice
x(N) = wildcard of length N

Note:
H = conserved histidine at heme binding location.
LIVM = all hydrophobic amino acids
G = conserved glycine

PROSITE Database

A manually created collection of regular expressions associated with different protein families/functions.

USE: have a description of sequence motifs associated with function, for elucidating function of new sequences.

Sequence motifs rely on multiple alignments for definition

• Use local alignment method (eg. Smith-Waterman) to find local areas in protein sequences that are high scoring.

• Create a multiple alignment of all pairs that share same local areas (multiple pairwise comparisons).

• Use this alignment to extract a summary of the key features of the motif.

Methods for representing motifs

Consensus sequence: a single string with the most likely sequence (+/- wildcards)

Regular expression: a string with wildcards, constrained selection

Profile: a list of the amino acid frequencies at each position

Sample Profile

<table>
<thead>
<tr>
<th></th>
<th>C*</th>
<th>A</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>W</th>
<th>Gap</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>6</td>
<td>-17</td>
<td>12</td>
<td>9</td>
<td>-30</td>
<td>8</td>
<td>-2</td>
<td>-9</td>
<td>-23</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>-48</td>
<td>32</td>
<td>35</td>
<td>-52</td>
<td>28</td>
<td>-6</td>
<td>-28</td>
<td>-44</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>10</td>
<td>-32</td>
<td>7</td>
<td>1</td>
<td>-38</td>
<td>43</td>
<td>-20</td>
<td>-24</td>
<td>-40</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>-6</td>
<td>-52</td>
<td>49</td>
<td>53</td>
<td>-60</td>
<td>5</td>
<td>5</td>
<td>21</td>
<td>-23</td>
<td>-44</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>L</td>
<td>-9</td>
<td>-59</td>
<td>18</td>
<td>15</td>
<td>-18</td>
<td>11</td>
<td>-21</td>
<td>-12</td>
<td>-17</td>
<td>5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>V</td>
<td>-5</td>
<td>-8</td>
<td>-18</td>
<td>15</td>
<td>-18</td>
<td>-13</td>
<td>-20</td>
<td>14</td>
<td>100</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>63</td>
<td>-53</td>
<td>-15</td>
<td>-10</td>
<td>-68</td>
<td>1</td>
<td>-31</td>
<td>-52</td>
<td>-63</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>-45</td>
<td>-96</td>
<td>-83</td>
<td>-65</td>
<td>-7</td>
<td>-37</td>
<td>-61</td>
<td>-65</td>
<td>-42</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

C* = consensus sequence down first column

Columns = scores for using amino acid listed at top of column
BLOCKS database
Collection of protein sequences with high level of similarity and which occur in PROSITE database, aligned in “blocks”.
USE: understand the sequence variability of a particular motif. Also, can use to create substitution matrices, e.g. BLOSUM matrices.

Sequence Weighting
One problem with creating motifs is that the set of sequences used is usually biased.
* because close sequences are easier to align.
* because databases do not have a “random” selection of sequences--clusters of organisms, and clusters of protein-types
We can “weight” the sequences used to create motifs to get a less biased motif.

Sequence Weighting
Human: HAVDL
Chimp: HAIEV
Possible motif:
H-A-Hydrophic-Negative-Hydrophobic

Need for weighting
Human: HAVDL
Chimp: HAIEV
Yeast: HGVFI
Fern: HSLRM
Possible motif (more general now):
H-Small-Hydrophobic-Polar-Hydrophobic
Possible profile for second column:
0.5 (A), 0.25 (G), 0.25 (S)
Better profile?
0.33 (A), 0.33 (G), 0.33 (S)

Methods for Weighting
1. Based on trees (construct tree from pairwise sequence distances). Weight close neighbors lower, since more similar.
2. Based on measure of entropy (Krogh et al, ISMB ‘95). Weight based
3. Based on variability within individual columns (overweight rare occurrences).
The key task is to apply a theory for how weights should be applied to make motif have sensitive detection of correct examples.

Simple scheme for weighting based on variability in individual columns
Replace each amino acid by 1/N * 1/n, Where N = total number of different Aas in column
n = total number of type found at position I
Weight of sequence = Average weight of elements

D F = 1/6 + 1/2 => 0.333
D W = 1/6 + 1/8 => 0.146
E W = 1/3 + 1/8 => 0.229
N W = 1/6 + 1/8 => 0.146
N W = 1/6 + 1/8 => 0.146
### Machine learning approaches to discovering motifs

**Supervised Learning:**
- Given a set of positive and negative examples, and then find rules for classifying new examples as positive or negative.

**Unsupervised Learning:**
- Given a set of examples (not labelled) find patterns that allow clustering of examples.

### Supervised Learning

1. Neural networks
2. Decision trees
3. Statistical regression
4. Bayesian classification
5. Genetic algorithms/programming

### Unsupervised Learning

- and others.

### Pitfalls of creating motifs

- Depend on quality of multiple alignment.
- Multiple alignments are easier for related sequences. Much harder for distantly related sequences.
- The database of sequences does not have a random sample of sequences, it is a biased selection.

Therefore, motifs will tend to be too specific and not sensitive enough.

### Sensitivity/Specificity

- Sensitivity = true positive rate (probability that a positive result is correct)
  \[ \text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negs}} \]
- Specificity = true negative rate (probability that a negative result is correct)
  \[ \text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Pos}} \]

These quantities trade-off. Can always have 100% specificity and 0% sensitivity or vice versa. (HINT: call everything “negative”)

### Finding exact patterns between two sequences

1. Naïve algorithm
2. Boyer-Moore method
3. Finite automata implementation

### Naïve method

Given template, T and probe, P find occurrence of P in T:

\[
\begin{align*}
\text{T: } & \text{xabxyabxyabxz} \\
\text{P: } & \text{abxyabxz} \\
& \text{abxyabxz} \\
& \text{abxyabxz} \\
& \text{abxyabxz} \\
& \text{abxyabxz}
\end{align*}
\]
Smarter method

Use knowledge of P (cached in a data structure, like a suffix tree) to notice that need another “a” to start the next promising match:

T: xabxyabxyabxz
P: abxyabxz

abxyabxz ; jump to a
abxyabxz ; may not need
; to compare
; first “abx”

Suffix Tree

Figure 1. A suffix tree for sequence AACTT

Compact Suffix Tree

Figure 1. Suffix tree for sequence AACTT. Some edges have multi-symbol labels resulting from collapsing nodes with single children.

Problems with Naïve Method

1. Worst case running O(nm), where n = length of P, m = length of T.
2. Can achieve O(n+m) if you use preprocessing to remember things about P.
   (Preprocessing can be performed in linear time O(m)).

Boyer-Moore Algorithm for exact string matching


Three rules used:
1. Right to Left scan
2. Bad character rule
3. Good suffix rule

Boyer-Moore: R->L Scan

T: xpbctbxabpqxctbpq
P: tpabxab
* = mismatch
(Still like naïve, but benefits from the next two rules…)
**Boyer-Moore: Bad character**

When a mismatch occurs at position \( i \) of \( P \) and the mismatched character in \( T \) is \( x \), shift \( P \) to the right so that the closest \( x \) to the left of position \( i \) in \( P \) is below the mismatched \( x \) in \( T \).

\[
\begin{align*}
T & : \text{xpbctbxabpqxcxbpq} \\
P & : \text{tpabxab} \\
* & : \text{use to skip 2} \\
P & : \text{tpabxab}
\end{align*}
\]

**Boyer-Moore: Good suffix**

After a mismatch, find a string in \( P \) to the left of the mismatch that aligns with the match to the right of the mismatch.

\[
\begin{align*}
T & : \text{prstabstubabvqrst} \\
P & : \text{qcabdabqcbdabqdab}
\end{align*}
\]

Note: bad character would shift only 1.
Note: repeated dab would be a weaker implementation of this rule.

---

**Boyer-Moore Exact String Match:**

Requires a linear time preprocessing to find all the suffixes.

Worse case running time of \( O(m) \), with \( 5m \) comparisons.

Other proofs have shown \( 4m \) and \( 3m \) comparisons. One has shown \( 2m \).

Bad character alone = \( O(nm) \) worst case, but sublinear on random strings.

---

**Implementing a simple linear string matcher with finite state automata.**

Need to preprocess the string into a table with all the transitions from prefix to prefix.

\( P = \text{TGG} \); the codon for the amino acid Tryptophan

\[
\begin{align*}
\text{Start} & \rightarrow \text{T} \\
\text{T} & \rightarrow \text{T} \\
\text{TG} & \rightarrow \text{TGG}
\end{align*}
\]

---

**Table Format**

<table>
<thead>
<tr>
<th>Start</th>
<th>A</th>
<th>T</th>
<th>G</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>START</td>
<td>T</td>
<td>START</td>
<td>START</td>
<td></td>
</tr>
<tr>
<td>TG</td>
<td>START</td>
<td>T</td>
<td>TG</td>
<td>START</td>
<td></td>
</tr>
</tbody>
</table>

**Find LEU codon = UUA, UUG, CUU, CUG, CUA, CUC**

\[
\begin{align*}
\text{Start} & \rightarrow \text{U} \\
\text{U} & \rightarrow \text{UU} \\
\text{C} & \rightarrow \text{CU} \\
\text{G,A} & \rightarrow \text{UUA, UUG}
\end{align*}
\]
Multiple Sequence Alignment using Gibbs Sampling

Goals of Multiple Sequence Alignment

- Determine Consensus Sequences
  - Consensus sub-sequences aligned together
- Building Gene/Protein Families
  - Block signature of protein/DNA family
- Develop Phylogenies
  - Small changes in the motif region indicate the evolutionary distance between organisms
- Model Protein Structure for Threading
  - Predict a protein structure by its motifs

Gibbs Sampling

The heart of multiple sequence alignment is to capture the characteristics of the motif or consensus sequence.

Gibbs Sampling can be used for both multiple sequence alignment and motif finding (= local multiple alignment).


Local Multiple Alignment

k sequences that share a motif of width w, with distant sequence homology

Need to find the alignment which brings the motifs into register...

How does Gibbs capture a motif?

Probability profile of a motif with length w

<table>
<thead>
<tr>
<th>Base</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.7</td>
<td>0.05</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>0.1</td>
<td>0.05</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0.1</td>
<td>0.1</td>
<td>0.59</td>
<td></td>
<td></td>
<td>0.8</td>
</tr>
<tr>
<td>C</td>
<td>0.1</td>
<td>0.8</td>
<td>0.01</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Background: A 0.3, T 0.3, G 0.2, C 0.2

The goal of Gibbs Sampling is to maximize the difference between motif base composition and background base distribution.
Basic Steps of Gibbs Sampling

Step 1: Pick random start position for all sequences, compute current motif matrix and background frequency

Step 2: Iterative update

1. Take one sequence out and update motif
2. Calculate the fitness score of each position of the out sequence as the starting position of the motif
3. Pick starting position sampling from P(a)
4. Take out another sequence…until convergence

Step 3: Reset starting position

Step 2: Iterative update

1) Take one sequence out and update motif

\[ \theta_j = \frac{n_i(s[p+j]) + \alpha_i}{n^s + \alpha} \]

s[p+j] -- letter i at position p of sequence s
n_i(j) -- number of letter i in position j of current motif matrix
n^s -- all letters observed at position j of motif = aligned segments
\( \alpha_i \) -- pseudo count for letter i

Fitness score of motif starting at p in sequence

\[ \text{Fit}(p) = \prod \theta_i / \theta_{\text{background}} \]
**Step 2: Iterative update**

3) Pick starting position sampling from \( \text{Fit}(a) \)--NOT picking the highest score (helps sampler “look around”)

![Sample from Fitness Score](image)

**Step 2: Iterative update**

3) Pick starting position sampling from \( \text{Fit}(a) \)

![Diagram](image)

**Step 2: Iterative update**

4) Take out another sequence… until converge

![Diagram](image)

**Step 2: Iterative update**

Fitness score at beginning iterations

<table>
<thead>
<tr>
<th>Motif</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>bg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.4</td>
<td>0.1</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>T</td>
<td>0.2</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>G</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Sequence: TTCCATATTACAGATTCGG...

Fitness score

- TAATC...
- AATCA 0.493 x 0.103 x 0.103 x 0.102 x 0.203 = 0.048383
- ATCAG 0.493 x 0.503 x 0.403 x 0.403 x 0.403 = 11.85185
- TCAG 0.203 x 0.203 x 0.303 x 0.302 x 0.203 = 0.444444
- CAGAT...

**Step 2: Iterative update**

Fitness score near time of convergence

<table>
<thead>
<tr>
<th>motif</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>bg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.8</td>
<td>0.04</td>
<td>0.03</td>
<td>0.75</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>T</td>
<td>0.1</td>
<td>0.7</td>
<td>0.03</td>
<td>0.06</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>G</td>
<td>0.03</td>
<td>0.15</td>
<td>0.04</td>
<td>0.07</td>
<td>0.7</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>0.07</td>
<td>0.11</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Sequence: TTCCATATTACAGATTCGG...

Fitness score

- TAATC...
- AATCA 0.8/0.3 x 0.04/0.3 x 0.03/0.3 x 0.1/0.2 x 0.1/0.3 = 0.005926
- ATCAG 0.8/0.3 x 0.7/0.3 x 0.9/0.2 x 0.75/0.3 x 0.7/0.2 = 245
- TCAG 0.1/0.3 x 0.1/0.2 x 0.03/0.3 x 0.07/0.2 x 0.1/0.3 = 0.002139
- CAGAT...

**Step 3: Reset starting position**

Reinitialize everything (starting position, motif starting sequence distribution) after \( n \) iterations to jump out of local maximum.

At the end of the program report the most frequent visited starting position of motif in each sequence.
Gibbs Sampling

Successfully uses greedy search to find motifs.

Needs to be extended to handle:

- Multiple conserved motifs separated by gaps
- Not knowing the length of the motif
- Having sequences that don’t have the motif

Nevertheless, a powerful tool for motif finding.