Algorithmic Bioinformatics DD2450, spring 2010, Lecture 5

Lecturer Jens Lagergren
Several current and previous students
will be acknowledged in a separate document.

20 april 2010

Multi alignment

In a multi alignment more than two sequences are compared. This alignment contains more information than pairwise alignments. From the multi alignment it is possible to construct a consensus sequence which can be used to recreate what the ancestor to the original sequences looked like.

Exemple of pairwise alignment:

S_1: ...AATGCG...
S_2: ...ACCGCT...

Exemple of multi alignment:

S_1:	AATGCG
S_2:	ACCGCT
S_3:	AATCCT
Consensus:	AATGCT

Applications:

- Database searches, create HMM
- First step in phylogeny
- Signal identification

Definition: *A multialignment of sequences $S_1, \ldots, S_r$ is an $r \times k$-matrix A such that:*

- The support of row i is S_i.
- No column contains only blank symbols.

Let $|A|$ be the number of columns in A and let $s : \{A, C, G, T, -\}^2 \to R$ be a nucleotide similarity function.

Definition: *Sum of Pairs (SP) score, σ, of A is defined as:*

$$\sigma(A) = \sum_{i=1}^{|A|} \sum_{i \leq j < k \leq r} s(A_{ji}, A_{ki})$$

where $|A| =$ the number of columns in A.

$$\gamma(S_1, \ldots, S_r) = \max_{\text{align. } A \text{ of } S_1, \ldots, S_r} \sigma(A)$$

Notation: *For a string $X = x_1, \ldots, x_m$*

1. $X^{\leq i} = X_1, \ldots, X_i$
2. $X^{>i} = X_{i+1}, \ldots, X_m$
3. $X^{>i, \leq i+1} = X_i$
4. $X^{>i, \leq i} = -$

Let

$$g(i_1, \ldots, i_r) = \gamma(S_1^{\leq i_1}, \ldots, S_r^{\leq i_r})$$

Recursion for g:

$$g(\vec{0}) = 0$$

$$g(i_1, \ldots, i_r) = -\infty, \ \forall i_1, \ldots, i_r \in \{-1, 0\} \text{ such that } \sum_{j=1}^{r} i_j < 0$$

$$g(i_1, \ldots, i_r) = \min_{\delta_1 \delta_r \in \{0,1\}^r \} \setminus \vec{0}} g(i_1 - \delta_1, \ldots, i_r - \delta_r) + \sum_{i \leq j < k \leq r} s(S_j^{>i_j - \delta_j \leq i_j}, S_k^{>i_k - \delta_k \leq i_k})$$

Complexity:

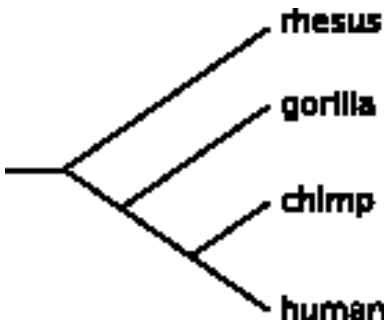
Time: $O(2^r n^r)$ since filling an element in the matrix takes time 2^r and the matrix has n^r elements.
Space: $O(n^r)$ for score and alignment.

Progressive Alignments

Introduction

Some well known programs use progressive alignments, like ClustalW, muscle and T-coffee. Why use this type of alignment ?
The time complexity of dynamic programming for multialignments is $\Omega(n^k)$ for k sequences of length n. With more than 6 sequences to align, this becomes much too big.

For homologous sequences, a good starting point is to use the tree relating the sequence and align the most similar sequences first.



In this figure, you would first align human sequence with chimp sequence, then align the resulting alignment with gorilla, and again with rhesus monkey. Then we need to define how to align alignments.

Aligning Alignments

Notation:

- $A = r \times c$ alignment
- $A' = r' \times c'$ alignment
- $i - prefix =$ the i first columns
- $A(:,j) =$ the $j - th$ column in A.
- For $v = v_1, v_2, ..., v_r$ and $u = u_1, u_2, ..., u_r$
  $SP(u, v)$ is the sum of pairs score for v and u.
- $e =$ r blanks $= \{-\}^r$
- $e' =$ r' blanks $= \{-\}^{r'}$

Definition:

An alignment of A and A' is A'' with $r + r'$ rows and:

- r first rows with e^T removed is A
- r' last rows with e'^T removed is A'
- no columns contain only blanks

The score of A'' is the sum $\sum_{C column} SP(C)$

To find the best alignment, we need to compute $g(i, j) =$ max score of alignment of the $i - prefix$ of A and the $j - prefix$ of A'.

Recursion for g:

$$g(i,j) = \begin{cases} g(i-1, j-1) + SP(A(.,i)^T, A'(.,j)^T) \\ g(i-1, j) + SP(A(.,i)^T, e'^T) \\ g(i, j-1) + SP(A'(.,j)^T, e^T) \end{cases}$$

We also need to add the initialization case to complete the algorithm.

The time complexity of this alignment is $O(c * c' * r * r')$

Progressive Alignments

Notation for rooted tree T:

- $r(T)$ = the root
- $V(T)$ = the vertices
- $E(T)$ = the edges
- $h(T)$ = the leaves

Input: rooted tree and one DNA sequence for each leaf of the tree.
Output: One multialignment

Algorithm: visit vertices of T in postorder. When u with children v and w is visited, $s(u) \leftarrow align(s(u), s(w))$.
Time complexity: The worst case is really bad, but this algorithm is usually fast.

Hidden Markov models and Bayesian statistics

Markov models

A descrete Markov model is a trippel $M = \{Q, A, q_s\}$ where

- Q is a set of states $\{q_1, q_2, ...\}$
- $A = \{a_{qq'} : q, q' \in Q\}$
- q_s is a start state

M generates a sequence of states $\{\Pi_0, \Pi_1, ...\}$ where

- $\Pi_0 = q_s$
- $P(\Pi_i|\Pi_{i-1}, ..., \Pi_1) = P(\Pi_i|\Pi_{i-1}) = a_{\Pi_{i-1},\Pi_i}$

A computer scientist often prefers to view a discrete Markov model as a directed graph with edge costs that represent transition probabilities.

Scenario

A casino uses a fair (F) dice with probability 0.99 and a weighted (W) with probability 0.01.

- $P(6|W) = \frac{1}{2}$
- $P(i|W) = \frac{1}{2}$ for $1 \leq i \leq 5$

Let A be the event that three 6 are observed consecutively. Which probability is larger of these two dices? $P(W|A)$ or $P(F|A)$?

Bayesian approach

- $P(W|A) = \frac{P(A|W)P(W)}{P(A)}$
- $P(F|A) = \frac{P(A|F)P(F)}{P(A)}$

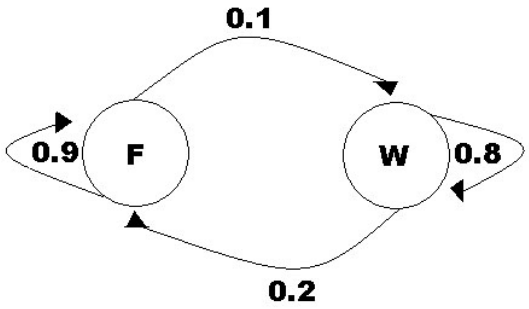
Since comparing $P(W|A)$ with $P(F|A)$ we can remove the denominator from the both exprecions.

- $P(A|W)P(W) = \frac{1}{3}^3 * 0.01$
- $P(A|F)P(F) = \frac{1}{6}^3 * 0.99$ (LARGEST)

Since $P(A|F)P(F) > P(A|W)P(W)$ we can asume that the fair dice was used.

HMM approach

We can model it like this, where we have 2 states $Q = fair, weighted$. The outcomes from the dist. dependencys are then observed on the state.



HMM are used for

- Gene finding
- Predict secondary structure of proteins
- Predict protein localization
- Identify/characterize domain/genefamilies

Algorithmic problems for HMM

- What is the probability that a sequence X is generated from an HMM M?
- Which sequence of states has the highest probability to generate X?
- Which HMM has the highest probability to generate a set of sequences S? (Expectation maximisation is used to compute the parameters of this HMM)