Harvard-MIT Division of Health Sciences and Technology
HST.951J: Medical Decision Support, Fall 2005
Instructors: Professor Lucila Ohno-Machado and Professor Staal Vinterbo

Three Bioinformatics Problems

Staal A. Vinterbo

Harvard-MIT Division of Health Science and Technology

Decision Systems Group, BWH

Harvard Medical School

Nov 2005: HST 951/MIT 6.873 Class

Motivation

Introduce bioinformatics through discussinb

- sequence alignment
- maximum parsimony phylogenetic trees
- haplotype tagging

Bioinformatics

- Advances in molecular biology and technology in this area
- lots of data $\Rightarrow$
- application of information technology $\Rightarrow$
- bioinformatics

Sequence Alignment

DNA and RNA

DNA

- polymer made up of nucleotides
- adenine (A), cytosine (C), guanine (G), and thymine (T), and
- abstracted as a string over $\Sigma = \{A, C, G, T\}$.

RNA

- another molecule abstracted as a string over $\Sigma = \{A, C, G, U\}$, U = uracil.

DNA and RNA are parts of the reproductive machinery, and insight is warranted.

Definition (Sequencing)

process of determining the DNA and RNA sequences from organisms

Sequence Alignment
Problem

Problem
evolutionary process does not necessarily preserve the exact location of the (somewhat changed) original sequence part.

I.e. do not know exactly how different sequences fit together.

Need:
alignment

Sequence Alignment
Alignment of Two Sequences

Let

- $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$ strings over Σ,
- Special symbol $- \notin \Sigma$.

An *alignment* of x and y:

- pair of strings x' and y', $|x| = |y| = k$, $\max(n, m) \leq k \leq m + n$ over $\Sigma \cup \{-\}$ such that
    1. at no position can the strings x' and y' both contain the special symbol $-$, and
    2. by removal of all occurrences of $-$ in both x' and y' we get x and y, respectively,

  both hold.

Sequence Alignment
Alignment of Two Sequences

Example
Consider sequences `CACT` and `ACGCTT`. Three possibilities of aligning them are

```
C — A — C T
A C G C T T
```

and

```
— C A C T —
A C G C T T
```

```
— C A — C T
A C G C T T
```

Sequence Alignment
Alignment of Two Sequences

Let

- $d : \Sigma \cup \{-\} \to \mathbb{R}$ be a function that assigns a "distance" between the elements of $\Sigma \cup \{-\}$ such that for $a \in \Sigma$

$$d(a, -) = d(-, a) = g(a)$$

  for some suitable function g.

Then

- $d(a, b)$ = cost of mutating (changing) a to b
- $g(a)$ = cost of inserting or deleting the letter a

Sequence Alignment

Alignment of Two Sequences

If we let $A(x, y)$ denote the set of all alignments (x', y') of x and y, we can define

$$D(x, y) = \min_{(x', y') \in A(x,y)} \sum_{i=1}^{k} d(x'_i, y'_i)$$

as the cost of an optimal alignment of x and y.

Sequence Alignment

Alignment of Two Sequences

Crucial Observation

The optimal alignment of x and y can end in one of three possible ways:

$$\begin{matrix} \ldots & x_n \\ \ldots & - \end{matrix}, \quad \begin{matrix} \ldots & - \\ \ldots & y_m \end{matrix}, \quad \begin{matrix} \ldots & x_n \\ \ldots & y_m \end{matrix}.$$

$D(x, y)$ is the cost of the optimal alignment up to the $(k-1)$st character plus the smallest cost of the three possibilities.

Sequence Alignment

Alignment of Two Sequences

Define

$$D_{i,j} = D(x_1 x_2 \cdots x_i, y_1 y_2 \cdots y_j),$$

and

$$\begin{aligned} D_{0,0} &= 0 \\ D_{0,j} &= \sum_{i=1}^{j} d(-, y_i) \\ D_{i,0} &= \sum_{j=1}^{i} d(x_j, -). \end{aligned}$$

Then we have that

$$D_{i,j} = \min\{D_{i-1,j} + d(x_i, -), D_{i-1,j-1} + d(x_i, y_j), D_{i,j-1} + d(-, y_j)\}.$$

Sequence Alignment

Alignment of Two Sequences

Have that

- $D(x, y) = D_{n,m}$
- At position i, j, a step
  - left = appending $-$ to x'
  - down = appending a $-$ to y'
  - down and to the left = x_i and y_j to x' and y' respectively.

Can use this to reconstruct an optimal sequence.

Sequence Alignment

Alignment of Two Sequences

$x = \texttt{ACCGTAC}$ and $y = \texttt{GCCTAA}$. If we let $g = 1$, $d(a, a) = 0$ for all a, and $d(a, b) = 1$ for all distinct a and b, the matrix D then becomes

$$\begin{array}{c|ccccccc}
 & & \texttt{G} & \texttt{C} & \texttt{C} & \texttt{T} & \texttt{A} & \texttt{A} \\
\hline
 & \boxed{0} & 1 & 2 & 3 & 4 & 5 & 6 \\
\texttt{A} & 1 & \boxed{1} & 2 & 3 & 4 & 4 & 5 \\
\texttt{C} & 2 & 2 & \boxed{1} & 2 & 3 & 4 & 5 \\
\texttt{C} & 3 & 3 & 2 & \boxed{1} & 2 & 3 & 4 \\
\texttt{G} & 4 & 3 & 3 & \boxed{2} & 2 & 3 & 4 \\
\texttt{T} & 5 & 4 & 4 & 3 & \boxed{2} & 3 & 4 \\
\texttt{A} & 6 & 5 & 5 & 4 & 3 & \boxed{2} & 3 \\
\texttt{C} & 7 & 6 & 5 & 5 & 4 & 3 & \boxed{3}
\end{array}$$

Sequence Alignment

Alignment of Two Sequences

Starting in the lower right hand corner, we notice that $2 + d(\texttt{C}, \texttt{A}) = 3$, which is the number in our current position. Hence, we step to $D_{5,6}$ containing a 2. We now note that $2 + d(\texttt{A}, \texttt{A}) = 2$, and we step to $D_{4,3}$. The choices we make along this traceback are indicated by the boxed entries in D above. Note that these choices are not unique, and that other optimal alignments are possible. Having made our choices, we can now start in the upper left corner and construct the alignment according to the step rules given above. The resulting alignment is

```
A C C G T A C
G C C − T A A
```

Sequence Alignment

BLAST

The above: Global alignment.

Alternative: Local Alignment

Basic Local Alignment Search Tools (BLAST) is a set of tools to perform *local* alignment: the search for similar segments of two sequences.

Sequence Alignment

Alignment of Multiple Sequences

An alignment of k strings $x_1, x_2, \ldots, x_k$ over Σ is a k-tuple of strings $x'_1, x'_2, \ldots, x'_k$ over alphabet $\Sigma \cup \{-\}$, where $- \notin \Sigma$, of equal length l such that

1. for each position $i \in \{1, 2, \ldots, l\}$ there is a j such that the character in position i in x'_j is different from $-$, namely $x'_{j,i} \neq -$, and
2. deletion of $-$ from all $x'_1, x'_2, \ldots, x'_k$ yields $x_1, x_2, \ldots, x_k$, respectively,

both hold.

Sequence Alignment

Alignment of Multiple Sequences

Given an alignment x' and y' of length k of strings x and y, we define

$$\mathrm{SP}(x', y') = \sum_{i=1}^{k} d(x'_i, y'_i).$$

The measure SP is the sum of pairwise alignment costs. Let $M = (x'_1, x'_2, \dots, x'_k)$ be an alignment of strings $x_1, x_2, \dots, x_k$. Abusing notation slightly, we define

$$\mathrm{SP}(M) = \sum_{i<j} \mathrm{SP}(x'_i, x'_j)$$

as the sum of pairwise alignment cost of M.

Sequence Alignment

Alignment of Multiple Sequences

Given strings $x_1, x_2, \dots, x_k$, the problem of finding M that minimizes $\mathrm{SP}(M)$ is NP-hard.
Can use the *Center Star Method*, a 2-approximation algorithm.

Sequence Alignment

Center Star

Assume that d is such that for all $a, b, c \in \Sigma$ we have that $d(a, b) = d(b, a)$ and $d(a, c) \leq d(a, b) + d(b, c)$, i.e., d is symmetric and observes the triangle inequality. The center star method is as follows:

1. Find $c \in \{1, 2, \dots, k\}$ that minimizes

   $$\sum_{i \neq c} D(x_c, x_i).$$

   and set $M = (x_c)$
2. let $y = x_c$
3. For all $j \in \{1, 2, \dots, k\} - \{c\}$ in order,
   - 3.1 add the gaps in y to x_j forming z
   - 3.2 compute the optimal alignment (y', z') of y and z
   - 3.3 add the newly added gaps in y to all sequences in M,
   - 3.4 add z' to M as x'_j, and
   - 3.5 set $y = y'$

Sequence Alignment

ClustalW

A popular alternative in practice is ClustalW:

1. Determine all pairwise alignments between sequences and the degree of similarity between them
2. Construct a dendrogram
3. Combine the alignments from 1 in the order specified in 2

Maximum Parsimony Phylogenetic Trees

Introduction

- Phylogeny = the evolutionary history
- At a molecular level, evolution can proceed by insertions, deletions, substitutions, inversions and transpositions
- A *phylogenetic tree* is a representation of evolution based on genetic sequences. Starting at a root, a branch represents a time where there is a divergence in the sequence.

The phylogenetic tree is often inferred from contemporary sequences, with internal nodes labelled according to the most plausible ancestral sequence.

Maximum Parsimony Phylogenetic Trees

Phylogenetic tree

Definition (Phylogenetic tree)

Let Σ be an alphabet, and let Σ^m denote the set of all strings of length m. Let $T \subseteq \Sigma^m$ be a finite set.

A *phylogenetic tree* over T is a tuple (G, l) consisting of a graph $G = (V, E)$, and a vertex labeling function $l : V \rightarrow \Sigma^m$ such that

- G is a spanning tree of V,
- $T \subseteq l(V)$,
- the leaves of G are uniquely labeled by T.

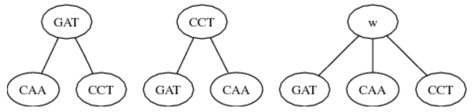
Maximum Parsimony Phylogenetic Trees

Phylogenetic tree

Example

Consider the sequences $x = \texttt{GAT}$, $y = \texttt{CAA}$ and $z = \texttt{CCT}$. There are a number of ancestral relationships we can think of, among them are

- x is an ancestor of z and y,
- z is an ancestor of x and y,
- an unknown w is an ancestor of all three.



Maximum Parsimony Phylogenetic Trees

Maximum Parsimony Principle

In order to choose between alternatives, we need to evaluate their plausibility. One principle is to choose the tree that incorporates the least evolutionary changes, i.e., the least number of mutations in the sequences along the branches of the tree. This principle is called the *maximum parsimony* principle.

Maximum Parsimony Phylogenetic Trees
Definition

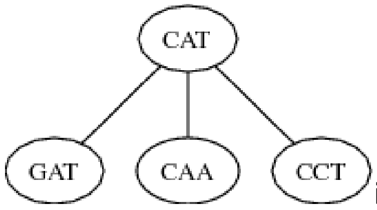
Define the score $\sigma(G, l)$ of a phylogenetic tree (G, l) over T as

$$\sigma(G, l) = \sum_{(a,b)\in E} h(l(a), l(b)),$$

where $h(l(a), l(b))$ is the hamming distance between the labels $l(a)$ and $l(b)$.

Definition (Maximum Parsimony Phylogenetic Tree (MPPT))
A phylogenetic tree (G, l) over T that minimizes σ.

Maximum Parsimony Phylogenetic Trees
Example

Example
From the preceding example $x = \texttt{GAT}$, $y = \texttt{CAA}$ and $z = \texttt{CCT}$.


is a maximum parsimony phylogenetic tree.

Maximum Parsimony Phylogenetic Trees
Steiner Trees

Definition (Steiner Tree)
Let $G = (V, E, w)$ be a graph with vertices V, edges E, and a non-negative edge cost function w. Any tree in G spanning a given set of *terminals* $T \subseteq V$ is called a *steiner tree* (over T).

Problem (Steiner Tree Problem (STP))
Let $G = (V, E, w)$ be a graph with vertices V, edges E, and a non-negative edge cost function w. For terminals $T \subseteq V$, find a minimum cost steiner tree in G.

Maximum Parsimony Phylogenetic Trees
Steiner Trees

We recognize the MPPT problem as a an instance of the STP for the some unknown graph G' with (again unknown) terminals S such that $T \subseteq l(S)$.

- MST(T) is a 2-approximation for STP
- STP is ≈1.55-approximable

This means that MST(T) is a 2-approximation for the MPPT problem.

Problem:
The application of steiner tree problem algorithms is sometimes problematic as the graph G' can be taken to have a labelling function with co-domain that includes Σ^m, and consequently can be *huge*.

Maximum Parsimony Phylogenetic Trees

Transformations

We can define transformations on phylogenetic trees that do not increase the score of the tree, but have the potential to decrease it.

Maximum Parsimony Phylogenetic Trees

Transformations

Definition

Define the *consensus* of a collection A of strings from Σ^m as the set of strings

$$c_1 c_2 \cdots c_m$$

where c_i is a character that occurs with the highest frequency in position i in the elements from the collection A.

Let $\mathrm{cons}(A)$ be an element from the consensus of A.

Further define the neighbors $n(v)$ of $v \in V$ in G to be the set of vertices w such that $\{(v, w), (w, v)\} \cap E \neq \emptyset$.

Maximum Parsimony Phylogenetic Trees

Transformations

Example

The consensus of {“abc”, “acb”, “aca”, “bad”} is {“aca”,“acb”, “acc”, “acd”}.

Maximum Parsimony Phylogenetic Trees

Transformations

Proposition

Let $(G = (V, E), l)$ be a phylogenetic tree over T. Then, for any $v \in V$ we have that if we let $A = n(v) \cup \{v\}$ then

$$\sum_{w \in n(v)} h(l(v), l(w)) \geq \sum_{w \in A} h(\mathrm{cons}(l(A)), l(w))$$

holds.

Proposition

Let $a, b, c \in \Sigma^m$ for some positive integer m. Then, a maximum parsimony phylogenetic tree over a, b and c is $\{(1, 4), (2, 4), (3, 4)\}$, and the mapping function l is such that $l(\{1, 2, 3\}) = \{a, b, c\}$ and $l(4) = \mathrm{cons}(a, b, c)$.

Maximum Parsimony Phylogenetic Trees

Mutable and Immutable vertices

Given a phylogenetic tree (G, l) of T we can partition the vertices V into two classes, mutable $\mathcal{M}$ and immutable $\mathcal{I}$. This partition must meet the following requirement: $T \subseteq l(\mathcal{I})$.

Maximum Parsimony Phylogenetic Trees

Transformations

We define the following transformations on a phylogenetic tree (G, l) of T. They are:

- s for all internal vertices v in turn, compute $c = \mathrm{cons}(n(v) \cup \{v\})$, connect a new leaf w to v, change l such that $l(w) = l(v)$ and $l(v) = c$,
- m for all immutable vertices v in turn, find $w \in V \cap l^{-1}(l(v))$ such that $h(cons(l(n(w))), l(w))$ is minimal, make $(l^{-1}(l(v)) \cap \mathcal{I}) - \{v\}$ mutable,
- k recursively delete all mutable leaves,
- d delete mutable vertices that have degree 2, connecting the two edge endpoints,
- c for all mutable vertices v in turn, set $l(v) = \mathrm{cons}(l(n(v)))$.

Maximum Parsimony Phylogenetic Trees

Transformations

Proposition

Let τ be one of the transformations s,m,d,k,c. The application of τ to phylogenetic tree (G, l), results in a phylogenetic tree with score at most that of (G, l).

Maximum Parsimony Phylogenetic Trees

Heuristic

Let APPROX be an algorithm that takes T as input and returns the phylogenetic tree (G, l) that corresponds to the minimum spanning tree over the terminals. We then define the following algorithm RETREE. Set $T' = T$.

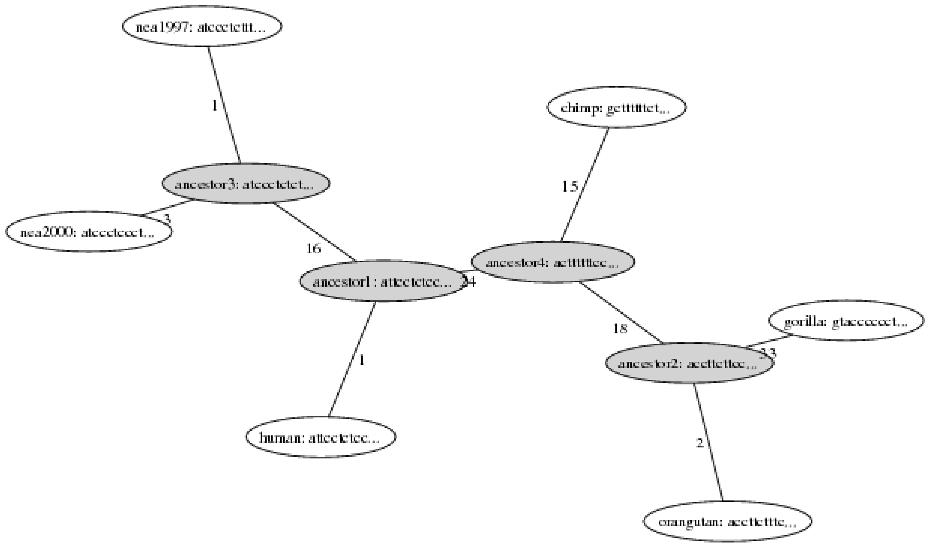
1. $(G, l) \leftarrow$APPROX(T')
2. $s \leftarrow \sigma(G, l)$
3. $(G, l) \leftarrow c(d(k(m(s(G, l)))))$
4. $T' \leftarrow l(G[V])$
5. if $\sigma((G, l) < s$ goto step 1
6. return (G, l)

Proposition

RETREE is polynomial in $m|T|$ and guarantees a solution score at most twice the optimum.

Moreover: RETREE is usually better than APPROX alone.

```
nea1997     (gi:2286205)   ATCCCTCTTTTTTTGTGGAATAATTCAACCCTCGTCTTAGTCTCCTATTAGTAGCATTACCTA
nea2000     (gi:7769684)   ATCCCTCCCTTTTTGTGAAATAATTCACCCCTCGTCTTAGTCTCCTATTAGTAGCATTACCTA
human       (gi:975204)    ATTCCTCTCCTATTGCGGAACAATTCAAACTTTGTCCTAATCCCCTGCTAGCAATATTACTTA
gorilla     (gi:3766221)   GTACCCCCCTCACCCCAATCCCCCCCCCACTCCCCTCCCAAATAACACTCCTCGCGATCACAC
chimp       (gi:6288860)   GCTTTTTTCTTCCCATAATCACCCTTCAATATCATCCCAGTCCAATGCCCCCCATCACAACCC
orangutan   (gi:1743293)   ACCTTCTTTCCTCTACAAAACCAACTCGATACTCCTCCCCAACACCGCCAGTAACCACCCTCC
```



Haplotype Tagging

Much of the population-wide variation of the human genome can be attributed to single nucleotide polymorphisms (SNPs), which are changes in single base pairs within the genome.

For the study of population genomics, SNPs can be used to measure *linkage disequilibrium*, an indication of how much more (or less) likely, compared to chance, certain combinations of neighboring SNP alleles are.

Due to linkage disequilibrium, the distribution of possible alleles at SNPs is not uniformly random, and some combinations of neighboring alleles occur more often than others. Such a combination of SNP alleles is called a *haplotype*, and a given set of SNPs can give rise to a wide variety of haplotypes.

Haplotype Tagging

It is an important problem to identify a subset of SNPs within a haplotype that allows the unique identification of all possible allele variations within a haplotype. If such a subset can be found, a haplotype can be uniquely identified by knowing only the allele values at a few SNP positions. SNPs that satisfy this requirement are called *haplotype tagging SNPs (htSNPs)*.

Haplotype Tagging

Formal Problem Definition

We can think of the problem of haplotype tagging by a minimum cardinality set of (diallelic) SNPs as the problem of finding a minimum cardinality set of columns that lets us discern between the unique rows in a binary matrix M. This means that we want to identify as few columns as possible so that if two rows agree on these columns, then they are identical. Formally, this can be expressed as follows.

Let $H = \{h_1, h_2, \ldots, h_n\}$ be a set of haplotypes with associated SNP markers at positions $S = \{1, 2, \ldots, m\}$, and let S' be a subset of S. We define $h_i[S']$ to be the string consisting of marker values of haplotype h_i found at the positions in S'. We can view H as an $n \times m$ matrix over the set of possible SNP alleles values. Find a minimum cardinality set $S' \subseteq S$ such that $h_i[S'] = h_j[S']$ implies $i = j$.

Haplotype Tagging

Complexity

Theorem (Vinterbo 2005)

The haplotype tagging problem is NP-hard but approximable within $1 + \ln((n^2 - n)/2)$ *for n haplotypes but not approximable within* $(1 - \epsilon)\ln(n/2)$ *for any* $\epsilon > 0$ *unless* $\mathrm{NP} \subset \mathrm{DTIME}(n^{\log\log n})$.

Haplotype Tagging

Algorithm

Solve the minimum hitting set instance given by the discernibility matrix over H.