Sequence alignment (part 2)

Local vs. global alignment
Linear gap penalties
Statistics of alignments

Last time: Two extensions of the basic algorithm

• Bounded-space computation
  – Space: \(O(km)\)
  – Time: \(O(km)\), where \(k = \) radius explored
  – Heuristic
    • Not guaranteed optimal answer
    • Works very well in practice
  – Practical interest

• Linear-space computation
  – Save only one col / row / diag at a time
  – Computes optimal score easily
  – Recursive call modification allows traceback
  – Theoretical interest
    • Effective running time slower
    • Optimal answer guaranteed

Today: Variations on the theme

• Changing the DP update rules
  – Global vs. local alignments
  – Overlap detection and semi-global alignment

• Affine gap penalties
  – Augmenting the state-space
  – Linear, affine, piecewise linear, general gap penalty

• Statistical significance of alignment
  – Where does \(s(x_i, y_j)\) come from?
  – Are two aligned sequences actually related

Intro to Local Alignments

• Statement of the problem
  – A local alignment of strings \(s\) and \(t\)
    is an alignment of a substring of \(s\)
    with a substring of \(t\)

• Why local alignments?
  – Small domains of a gene may be only conserved portions
  – Looking for a small gene in a large chromosome (search)
  – Large segments often undergo rearrangements
The local alignment problem

Given two strings $x = x_1 \ldots x_M$, $y = y_1 \ldots y_N$

Find substrings $x', y'$ whose similarity (optimal global alignment value) is maximum

e.g. $x = \text{aaaaccgggg}$
$y = \text{ttgcctgggaaccaacc}$

How can we use dynamic programming for local alignment?

Global Alignment

Initialization:
• Top left: 0
Update Rule:
$$A(i,j) = \max\{$$
• $A(i-1, j) - 2$
• $A(i, j-1) - 2$
• $A(i-1, j-1) \pm 1$
$$\}$$
Termination:
• Bottom right

Local Alignment

Initialization:
• Top left: 0
Update Rule:
$$A(i,j) = \max\{$$
• $A(i-1, j) - 2$
• $A(i, j-1) - 2$
• $A(i-1, j-1) \pm 1$
• 0
$$\}$$
Termination:
• Anywhere

Local Alignment issues

• Resolving ambiguities
  – When following arrows back, one can stop at any of the zero entries. Only stop when no arrow leaves. Longest.
• Correctness sketch by induction
  – Assume we’ve correctly aligned up to $(i,j)$
  – Consider the four cases of our max computation
  – By inductive hypothesis recurse on $(i-1,j-1), (i-1,j), (i,j-1)$
  – Base case: empty strings are suffixes aligned optimally
• Time analysis
  – $O(mn)$ time
  – $O(mn)$ space, can be brought to $O(m+n)$

Best local alignment vs. all local alignments

Termination:

1. If we want the best local alignment…
   $$F_{OPT} = \max_{i,j} F(i, j)$$
2. If we want all local alignments scoring $> t$
   For all $i, j$ find $F(i, j) > t$, and trace back

Global Alignment vs. Local alignment

Needleman-Wunsch algorithm

Initialization: $F(0,0) = 0$
Iteration: 
$$F(i, j) = \max\{F(i-1, j) - d, F(i, j-1) - d, F(i-1, j-1) + s(x_i, y_j)\}$$
Termination: Bottom right

Smith-Waterman algorithm

Initialization: $F(0, j) = F(i, 0) = 0$
Iteration: 
$$F(i, j) = \max\{F(i-1, j) - d, F(i, j-1) - d, F(i-1, j-1) + s(x_i, y_j)\}$$
Termination: Anywhere
Semi-global alignment and overlap detection

Different types of overlaps
• Application: genome assembly
  – Search for consecutive sequence segments
  – Boundaries of compared sequences not fully known

A variant of the basic algorithm:
• Allow unlimited # of gaps in the beginning and end
• No penalty for end gaps

The Overlap Detection variant

Changes:
1. Initialization
   For all i, j,
   \( F(i, 0) = 0 \)
   \( F(0, j) = 0 \)

2. Termination
   \( F_{OPT} = \max_{i} F(i, N) \)
   \( \max_{j} F(M, j) \)

Scoring the gaps more accurately

Current model: Linear gap penalty function

- Gap of length 1: incurs penalty: \( g \)
- Gap of length \( k \): incurs penalty: \( k \cdot g \)

However, in evolution, larger segments are gained and lost ⇒ gaps usually occur in bunches

More accurate model: Convex gap penalty function:

\[ \gamma(n): \]
for all \( n, \gamma(n + 1) - \gamma(n) \leq \gamma(n) - \gamma(n - 1) \]
General gap dynamic programming

Initialization: same

Iteration:
\[ F(i, j) = \max_{k=0}^{i-1} F(k, j) - \gamma(i-k) \]
\[ \max_{k=0}^{j-1} F(i, k) - \gamma(j-k) \]

Termination: same

Running Time: \( O(N^2M) \) (cubic)
Space: \( O(NM) \)

Can we do better?

Compromise: affine gaps

\[ \gamma(n) = p + (n - 1)q \]

• Gap open
  - To introduce the first gap, cost of introducing a DNA break
  - Fixed cost for opening a gap: \( p+q \)
• Gap extend
  - Larger segments more costly than smaller ones
  - Linear cost increment for increasing number of gaps: \( q \)
• How can we compute this using dynamic programming?
  - Achieve \( O(mn) \) running time & space

Additional Matrices

• The amount of state needed increases
  - In scoring a single entry in our matrix, we need remember an extra piece of information
    • Are we continuing a gap in s? (if not, start is more expensive)
    • Are we continuing a gap in t? (if not, start is more expensive)
• Dynamic programming framework
  - We encode this information in three different matrices
    - For each element \((i,j)\) we use three variables
      • \(a(i,j)\): best alignment of \(s[1..i]\) & \(t[1..j]\) that aligns \(s[i]\) with \(t[j]\)
      • \(b(i,j)\): best alignment of \(s[1..i]\) & \(t[1..j]\) that aligns gap with \(t[j]\)
      • \(c(i,j)\): best alignment of \(s[1..i]\) & \(t[1..j]\) that aligns \(s[i]\) with gap

Simplified update rules

• Transitions from \(b\) to \(c\) are not necessary...
  ...if the worst mismatch costs less than \( p+q \)

Using a single gap matrix

Initialization:
\[ F(0, 0) = p \]
\[ F(0, j) = p \]
\[ F(i, 0) = p \]

Iteration:
\[ F(i, j) = \max \left\{ \begin{array}{l}
F(i-1, j-1) + s(x_i, y_j) \\
F(i-1, j) - p \\
F(i, j-1) - p
\end{array} \right\} \]

Termination: same
To generalize a little…
…think of how you would compute optimal alignment with this gap function

\[ \gamma(n) \]

...in time \( O(m*n) \)

**General Gap Penalty**

- Gap penalties are limited by the amount of state
  - Linear gap penalty: \( w(k) = k*p \)
    - State: Current index tells if in a gap or not
  - Affine gap penalty: \( w(k) = p + q*k \), where \( q < p \)
    - State: add binary value for each sequence: starting a gap or not
  - Quadratic: \( w(k) = p + q*k + r*k^2 \)
    - State: needs to encode the length of the gap, which can be \( O(n) \)
    - To encode it we need \( O(\log n) \) bits of information. Not feasible
  - Length \( (\text{mod } 3) \) gap penalty for protein-coding regions
    - Gaps of length divisible by 3 are penalized less: conserve frame
    - This is feasible, but requires more possible states
    - Possible states are: starting, \( \text{mod } 3 = 1 \), \( \text{mod } 3 = 2 \), \( \text{mod } 3 = 0 \)

**What have we learned so far?**

- Sequence alignment and dynamic programming
  - Global
  - Local
  - Semi-global
- Different gap penalty functions
  - Linear
  - Last lecture
  - 2nd matrix
  - General \( O(m*n^2) \)

**The statistics of alignments**

Where does \( s(x, y) \) come from?
Are two aligned sequences actually related?

**Probabilistic Model of Alignments**

- We’ll focus on protein alignments without gaps
  - Given an alignment, we can consider two possibilities
    - \( R \): the sequences are related by evolution
    - \( U \): the sequences are unrelated
- How can we distinguish these possibilities?
- How is this view related to amino-acid substitution matrices?

**Model for Unrelated Sequences**

- We’ll assume that each position in the alignment is sampled randomly from some distribution of amino acids
- Let \( q_a \) be the probability of amino acid \( a \)
- The probability of an \( n \)-character alignment of \( x \) and \( y \) is given by

\[
\Pr(x, y | U) = \prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} q_{y_i}
\]
Model for Related Sequences

- we'll assume that each pair of aligned amino acids evolved from a common ancestor
- let $p_{a,b}$ be the probability that evolution gave rise to amino acid $a$ in one sequence and $b$ in another sequence
- the probability of an alignment of $x$ and $y$ is given by

$$\Pr(x, y \mid R) = \prod_{i=1}^{n} p_{x_i,y_i}$$

Probabilistic Model of Alignments

- How can we decide which possibility ($U$ or $R$) is more likely?
- one principled way is to consider the relative likelihood of the two possibilities (the odds ratio)

$$\frac{\Pr(x, y \mid R)}{\Pr(x, y \mid U)} = \frac{\prod_i p_{x_i,y_i}}{\prod_i q_{x_i} q_{y_i}}$$

- taking the log, we get

$$\log \frac{\Pr(x, y \mid R)}{\Pr(x, y \mid U)} = \sum \log \left( \frac{p_{x_i,y_i}}{q_{x_i} q_{y_i}} \right)$$

Probabilistic Model of Alignments

- the score for an alignment is thus given by:

$$S = \sum_i s(x_i, y_i) = \log \frac{\Pr(x, y \mid R)}{\Pr(x, y \mid U)}$$

- the substitution matrix score for the pair $a, b$ should thus be given by:

$$s(a,b) = \log \left( \frac{p_{a,b}}{q_a q_b} \right)$$

Substitution Matrices

- two popular sets of matrices for protein sequences
  - PAM matrices [Dayhoff et al., 1978]
  - BLOSUM matrices [Henikoff & Henikoff, 1992]
- both try to capture the relative substitutability of amino acid pairs in the context of evolution

Substitution Matrices

- the substitution matrix score for the pair $a, b$ is given by:

$$s(a, b) = \log \left( \frac{p_{a,b}}{q_a q_b} \right)$$

- but how do we get values for $p_{a,b}$ (probability that $a$ and $b$ arose from a common ancestor)?
- it depends on how long ago sequences diverged
  - diverged recently: $p_{a,b} \approx 0$ for $a \neq b$
  - diverged long ago: $p_{a,b} \approx q_a q_b$
Substitution Matrices

- key idea: trusted alignments of related sequences provide information about biologically permissible mutations

BLOSUM Matrices

- [Henikoff & Henikoff, PNAS 1992]
- probabilities estimated from “blocks” of sequence fragments that represent structurally conserved regions in proteins
- transition frequencies observed directly by identifying blocks that are at least
  - 45% identical (BLOSUM-45)
  - 50% identical (BLOSUM-50)
  - 62% identical (BLOSUM-62)
  - etc.

BLOSUM Matrices

- given: a set of sequences in a block
- fill in matrix $A$ with number of observed substitutions (we won’t worry about details of some normalization that happens here)

$P_{ab} = \frac{A_{ab}}{\sum_j A_{aj}} \quad q_a = \frac{\sum_i A_{ai}}{\sum_i A_{ij}}$

Statistics of Alignment Scores

- earlier we considered how do decide if a single alignment was more likely due to relatedness or chance
- but what if we’re considered many alignments?
  - e.g. what if we’re doing a BLAST search against a large protein database?
- we’d like to know how many high-scoring alignments we’re likely to get by chance

Statistics of Alignment Scores

Q: How do we assess whether an alignment provides good evidence for homology?
A: determine how likely it is that such an alignment score would result from chance.

3 ways to calculate chance: look at alignment scores for
- real but non-homologous sequences
- real sequences shuffled to preserve compositional properties
- sequences generated randomly based upon a DNA/protein sequence model

Distribution of Scores

- Karlin & Altschul, PNAS, 1990
- consider a random model in which
  - we’re looking for ungapped local alignments
  - the lengths of the sequences in each pair are $m$ and $n$
- the expected number of alignments, $E$, with score at least $S$ is given by:

$$E(S) = Knm^e^{-2S}$$
**Summary**

- Dynamic Programming variations
  - Local alignment vs. global alignment
  - Semi-global alignment
  - Linear gap penalties
- Sequence alignment statistics
  - Computing the substitution scores
  - Protein alignment
  - BLOSUM62 matrix
  - Extreme value distribution
- Next lectures:
  - String matching in linear time!
  - Database search and hashing

**What's next**

- Find the best match to a single gene in a genome
  - (Lecture 5: Database search)
- Understand the correspondence of all genes
  - (Lecture 24: Genome-scale comparative genomics)
  - **Align two genes**
    - (Lectures 2 + 3: Sequence alignment)
- Evolutionary relationships of many genes
  - (Lecture 12: Phylogeny)
- Align many genes simultaneously
  - (Lecture 13: Profile alignment)

**Project ideas**

- Coding-centric alignment
- Alignment with inversions
- Consensus-word multiple alignment
- Alignment with multiple seeds
- Multiple progressive alignment
- Statistics of genome alignment

**Distribution of Scores**

\[ E(S) = Kmne^{-2S} \]

- \( S \) is a given score threshold
- \( m \) and \( n \) are the lengths of the sequences under consideration
- \( K \) and \( \lambda \) are constants that can be calculated from
  - the substitution matrix
  - the frequencies of the individual amino acids

**Statistics of Alignment Scores**

- To generalize this to searching a database, have \( n \) represent the summed length of the sequences in the DB
- the NCBI BLAST server does just this
- with this analysis, can also calculate \( p \)-values (the probability of a random alignment scoring at least \( S \))
- theory for gapped alignments not as well developed
- computational experiments suggest this analysis holds for gapped alignments (but \( K \) and \( \lambda \) must be estimated from data)