Local Alignment

Local sequence alignment
BLAST
References:
Gusfield, §11.7

Local vs Global Alignment

- Recall from previous slides:
  - An alignment is a mapping between characters of two strings
    \[ S_i \rightarrow T_j \mid '-' \]
  - A **global** alignment accounts for all characters in each string
  - A **local** alignment identifies smaller regions of high similarity, omits surrounding areas

Book

- **BLAST**, by Korf, Yandell, and Bedell
  - Published by O'Reilly, 2003
- You might be able to read the book on-line
  - go to www.ora.com
  - click on “Safari Books Online”
  - search titles for “BLAST”
- It looks like UO has an account, and I was able to read the book when connecting from my office
  - Complete URL:
    http://proquest.safaribooksonline.com:80/0596002998

History

- Needleman-Wunsch (1970)
  - first to discuss global alignment
  - \[ O(n^3) \] method
- Smith-Waterman (1981)
  - defined local alignment
  - \[ O(n^2) \] implementation
- Literature:
  - N-W synonymous with global alignment, S-W synonymous with local alignment (even if implementation details differ)
- Today: Smith-Waterman local alignment
Applications of Local Alignment

- Situations where global alignments are useful:
  - Comparing sequences of the same size, e.g. tRNA
  - Comparing members from the same gene family, e.g. when reconstructing phylogeny

- Situations where local alignment is useful:
  - Comparing long pieces of DNA (find local similarities)
  - Searching protein databases (find proteins that share motifs)

Suffix Alignment Problem

- We will be trying to find regions of similarity in two strings $S_{1..n}$ and $T_{1..m}$

- For a pair of indices $i$, $j$:
  - $\alpha$ = a suffix of $S_{1..i}$
  - $\beta$ = a suffix of $T_{1..j}$

- Define
  - $v(\alpha, \beta)$ = alignment score of $\alpha$, $\beta$
  - $v(i, j)$ = the maximum value of $v(\alpha, \beta)$ over all suffix strings of $S_{1..i}$ and $T_{1..j}$

Suffix Alignment Problem

- Visualization using the dynamic programming matrix
  - $v(i, j)$ = the maximum value of $v(\alpha, \beta)$ over all suffix strings of $S_{1..i}$ and $T_{1..j}$

Local Alignment

- Given this definition of $v(i, j)$, finding an optimum local alignment is a matter of calculating
  \[ v^* = \max \{ v(i, j) : 1 \leq i \leq n \text{ and } 1 \leq j \leq m \} \]

- And it would be nice to know the strings $\alpha$ and $\beta$ and their alignment..
Complexity of Local Alignment

- How many possible substrings are there?
  - For $S_{1..n}$, there are $i$ different suffix strings at each location $i$
    - $i = 1$: $S_{1..1}$
    - $i = 2$: $S_{1..2}$, $S_{2..2}$
    - $i = 3$: $S_{1..3}$, $S_{2..3}$, $S_{3..3}$
    - ...
  - There are $\Theta(n^2)$ substrings of $S$, $\Theta(m^2)$ substrings of $T$
- Straightforward alignment of each pair of substrings would be $O(n^3m^3)$

Complexity of Local Alignment (cont’d)

- The next few slides describe a recurrence that allows us to use dynamic programming
- Construct table of $v(i,j)$ values
  - As in global alignment, a constant number of steps in each cell, so total time is $O(nm)$

Weights for Local Alignment

- A key difference between local and global alignment is the method for assigning weights
  - use positive values for matching positions
  - use negative values for mismatch or pairing with a space
- Example: weight for a position $i, j$ of an alignment might be
  - $w = 2$ if $S_i = T_j$
  - $-1$ if $S_i \neq T_j$
  - $-1$ if $S_i$ aligns with $-$ or $T_j$ aligns with $-$

Alignment Scores

- The sum of weights in a local alignment is a similarity score, not a cost as it is in global alignment
  - In global alignment, we minimized the sum of a set of positive costs
  - In local alignment, we try to maximize similarity (using both positive and negative scores)
- Define the similarity score for matching a substring with the empty string to be 0
Recurrence for Local Alignment

- The base cases for local alignment are to set the edges of the dynamic programming table to 0
  - \( V[0,0] = 0 \)
  - \( V[i,0] = 0 \) for \( 1 \leq i \leq n \)
  - \( V[0,j] = 0 \) for \( 1 \leq j \leq m \)
- These correspond to aligning characters from \( S \) or \( T \) with no characters from the other string.

Recurrence (cont’d)

- The score for an interior cell \( V[i,j] \) will reflect the value of aligning a suffix of \( S_{1..i} \) with a suffix of \( T_{1..j} \)
- Using the weights just described, the sum could be negative
- The Smith-Waterman method sets a lower limit of 0
- Equivalent to saying an alignment that strays into the negative range is the same as an alignment with an empty substring
  - i.e. the local alignment stops here….

Recurrence (cont’d)

- The formula for an interior cell \( i, j \) is the maximum of
  - \( V[i-1,j-1] + s(S_i,T_j) \)
  - \( V[i-1,j] + g \)
  - \( V[i,j-1] + g \)
  - 0
- \( s(S_i,T_j) \) is the score for aligning two characters from the input strings, e.g. 2 if they match and -1 if they don’t
  - \( g \) is the gap penalty

Traceback

- As in global alignment, record the neighboring cell that contributes the entry used to compute the score
- The best alignment has a score \( v^* \), the maximum of \( V[i,j] \) over all \( i,j \)
- Notes:
  - the single best local alignment is defined by a path starting in the cell containing \( v^* \) and ending at a cell containing 0
  - one can also keep a table of the best \( v \) scores to produce more than one alignment per pair of strings (but users won’t want to see alignments that are substrings of better alignments)
Substitution Matrices

- Clearly the success of local alignment depends on the choice of scores
  - positive values for the same character
  - negative scores for gaps
- It is possible to define a range of scores
  - slightly positive scores for similar characters
  - more negative scores for highly unlikely pairings
- Most common for protein alignments
- Potential for nucleotide alignments (e.g. transitions vs transversions)

PAM

- The Percent Accepted Mutation (PAM) matrices of Dayhoff et al. were the first similarity matrices
- Define “1 PAM” as a measure of the amount of change between two proteins:
  - \( S \) and \( T \) differ by 1 PAM if \( S \) mutated to \( T \) via 1 change per 100 amino acid positions
  - Insertions/deletions not considered
  - Changes are “accepted” mutations, i.e. not fatal

PAM (cont’d)

- Note that change is not difference
- A position could have mutated to a new amino acid and then mutated back to the original state
  - 2 changes, 0 observed differences
- An amino acid position could have changed twice
  - 2 changes, 1 observed difference

PAM (cont’d)

- Dayhoff et al used empirical data to estimate similarity
  - Collect and globally align similar sequences from protein families
  - Count the number of times each amino acid changes into other amino acids
    - Normalize for frequency of occurrence and sequence length
  - The PAM1 matrix is a 20x20 matrix based on this data
  - Define \( P[i,j] \) to be the probability amino acid \( i \) will change to amino acid \( j \) after 100 changes have occurred
PAM (cont’d)

- Multiplying the matrix by itself gives the probability of changes over two PAM units
- PAM\(_n\) is PAM1 multiplied \(n\) times
- PAM250 is the basis of a common scoring matrix
  - \(s_{ij} = \log(P_{ij}/P_x) * 10\)
  - \(P_x\) is the probability \(i\) and \(j\) are paired by chance (depends on background frequencies of letters)
  - \(s\) values range from 2 to 17 on the diagonal (pairs of identical letters)
  - some 0 entries (highly interchangeable pairs)
  - many strongly negative entries

Limitations of PAM

- PAM is based on a number of assumptions
  - Markov model: mutations are independent and reversible
  - constant rate of mutation
  - no insertions or deletions
- Another problem:
  - protein changes are based on codon changes
    - e.g. GTT (Val) \(\Rightarrow\) AGT (Ser) requires two changes
    - GTT \(\Rightarrow\) ATT \(\Rightarrow\) AGT is V \(\Rightarrow\) I \(\Rightarrow\) S
    - GTT \(\Rightarrow\) GGT \(\Rightarrow\) AGT is V \(\Rightarrow\) G \(\Rightarrow\) S
- Despite these limitations, it has been very effective and is widely used

BLOSUM

- An alternative scoring system is BLOSUM, based on protein motifs in the PROSITE database
- BLOCKS: database of short, highly conserved subsequences from PROSITE
  - define \(n_{ij}\) = number of times amino acids \(i\) and \(j\) are paired in global alignments of subsequences
- Compute ratio:
  \[
  \log(n_{ij}/e_{ij})
  \]
  where \(e_{ij}\) is a term reflecting chance pairings

BLOSUM62

- Some subsequences are very common
- Pairings based on these sequences exert too much influence
- BLOSUM\(_x\) is the matrix that results after identifying pairs of sequences that are \(x\)% similar and removing one at random
- BLOSUM62 has been very successful
  - default matrix for BLAST
BLAST

- BLAST = Basic Local Alignment Search Tool
- Implements efficient search of sequence database
- User supplies an input sequence, BLAST returns a set of local alignments with sequences in its database
- It also provides a metric (the "expect value") that estimates the probability a match was found by chance
  - [see http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html]

Implementation: Overview

- BLAST uses a heuristic algorithm
  - it does not search the entire matrix
- Instead it uses strategies that identify small high scoring regions and then "grow" these into alignments
- The algorithm uses parameters to control each phase
- Tradeoff: faster search (fewer initial hits and not-too-greedy extension) means loss of sensitivity (some good matches may not be found)

Implementation: Seeding

- A key parameter is known as the word size
  - a word is a small ungapped alignment
  - for DNA alignments, the default word size is 12 (with a minimum of 7)
  - for proteins words are 2 or 3 letters
- For the input sequence S make a table of words
  - S0..2
  - S1..3
  - S2..4
  - ...
- For each word there is a "neighborhood" of similar words that can be sorted by score
- Example (Korf):
  - for the string RGD similar words according to BLOSUM62 are RGD (17), KGD (14), QGD (13), RGE (13), ...
- The algorithm uses a threshold value t
  - put all word pairs with scores above t into a table
- Now scan the database to find instances of these neighborhood words
Implementation: Seeding

- The result will be a set of “hot spots” that are potential blocks in a final alignment.

Implementation: Extension

- Early implementations of BLAST extended the initial seeds in each direction.
- Current versions use a “double hit” strategy:
  - for a given hit find a distant hit on or near the diagonal
  - see if the two can be connected into a larger alignment

BLAST Versions

- BLAST is a family of related programs
  - blastn  nucleotide  nucleotide
  - blastp  protein   protein
  - blastx  nucleotide (6X)  protein
  - blastn  protein   nucleotide (6X)
  - tblastx nucleotide (6X)  nucleotide (6X)

- 6X = all six possible translations:
  - 3 “reading frames” on main strand
  - 3 “reading frames” on complementary strand

BLAST Servers

- There are dozens of BLAST servers on the web
- Connect to a server
- Enter query sequence information
  - Specify query by ID (e.g. Genbank identifier)
  - Cut-'n-paste a query sequence
- Specify search parameters
  - Substitution matrix
  - Word size
  - Many others
BLAST Servers (cont’d)

- CGI script runs BLAST on the server
- Results are formatted and returned in HTML page
- In many cases the page links to other pages
  - IDs of matching sequences are links to pages describing that sequence
  - Local alignments are summarized; graphic is a link to the actual alignment
- To learn more, take the interactive tutorial at the NCBI web site
  - BLAST is one of the top-level links from http://ncbi.nih.gov

Running BLAST Locally

- For large projects, install BLAST locally
- Download precompiled executables from NCBI
- (1) Create a local BLAST database
  - `% formatdb -i X.fa -p T -o T` (builds a local database from a protein FASTA file)

Running BLAST Locally (cont’d)

- To do a search for every sequence in a file, use blastall
  - `% blastall -p blastp -d human -i trans.fa -o trans.br -e 1e-10`
  - Runs blastp (protein vs protein) using database human
  - Input sequences are in trans.fa
  - Put output in trans.br (br = “Blast report”)
  - Report sequences with “expect value” < 10^-10

BLAST Output Formats

- The default BLAST report format is a text file
- New versions will generate several different formats, including XML (-m 9)
- A tabular format (-m 8) prints important data one line per hit, in tab-separated fields
  - Query ID, Hit ID, ...
  - Does not print alignments