Local Sequence Alignment & Heuristic Local Aligners

Lectures 18 – Nov 28, 2011  
CSE 527 Computational Biology, Fall 2011  
Instructor: Su-In Lee  
TA: Christopher Miles  
Monday & Wednesday  12:00-1:20  
Johnson Hall (JHN) 022

Review: Probabilistic Interpretation

\[ \text{x: TCCAGGTG-GAT} \]
\[ \text{Y: TGCAAGTGCG-T} \]

Chance or true homology?

Sharing a common ancestor
Review: Likelihood Ratio

\[ \begin{align*}
\mathbf{X}: & \quad \text{TCCAGGTG–GAT} \\
\mathbf{Y}: & \quad \text{TGCAAGTGCG–T} \\
\end{align*} \]

\[
Pr(Data|\text{Homology}) \quad Pr(Data|\text{Chance})
\]

Review: Log Likelihood Ratio Score

- The most commonly used alignment score of aligning two sequences is the log likelihood ratio of the alignment under two models:
  - Common ancestry
  - By chance

\[
Score = \log \left( \prod_i \frac{Pr(x_i, y_i)}{Pr(x_i)Pr(y_i)} \right) = \sum_i \log \left( \frac{Pr(x_i, y_i)}{Pr(x_i)Pr(y_i)} \right) = \sum_i s(x_i, y_i)
\]
Outline: Scoring Alignments

- Scoring alignments
  - Probabilistic meaning
  - Scoring matrices
    - PAM: scoring based on evolutionary statistics
    - BLOSUM: tuning to evolutionary conservation
    - Gaps revisited

- Local vs global alignment

- Database search
  - FASTA
  - BLAST

Gap Initiation and Extension

TCCACCGTG–GA

|   |   |   |   |   |   |

TGCA––GTGCGA
Gap Initiation and Extension

TCCACCGTG–GA
CSCCDDCCCIICC
TGCA--GTGCAGA

Insertion / deletion (indel)

Scoring Indels: Naive Approach

- A fixed penalty $d$ is given to every indel:
  - $-d$ for 1 indel,
  - $-2d$ for 2 consecutive indels
  - $-3d$ for 3 consecutive indels, etc.

Can be too severe penalty for a series of 100 consecutive indels!
Affine Gap Penalties

- In nature, a series of $k$ indels often come as a single event rather than a series of $k$ single nucleotide events:

\[
\begin{align*}
AT\underline{A__GC} & \quad ATAG\_GC \\
AT\underline{ATTGC} & \quad AT\_GTGC
\end{align*}
\]

| This is more likely. | Normal scoring would give the same score for both alignments | This is less likely. |

---

Gap Initiation and Extension

\[
\begin{align*}
TCC ACC GTG - GA \\
C S C C D D C C C C I C C \\
T G C A - - G T G C G A
\end{align*}
\]

Insertion / deletion (indel)
Scoring the Gaps More Accurately

- Current model:
  Gap of length \( n \) incurs penalty \( n \times d \)

- However, gaps usually occur in bunches

- 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:**
  \[ V(i, j) = \max \begin{cases} V(i-1, j-1) + s(x_i, y_j) \\ \max_{k=0}^{i-1} V(k, j) - \gamma(i-k) \\ \max_{k=0}^{j-1} V(i, k) - \gamma(j-k) \end{cases} \] Previously...
- **Termination:** same
- **Running Time:** \( O(N^2M) \) (assume \( N > M \))
- **Space:** \( O(NM) \)
Accounting for Gaps

- **Gaps**: contiguous sequence of spaces in one of the rows

- Score for a gap of length $x$ is:
  
  $$-(d + ex)$$

  where $d > 0$ is the penalty for introducing a gap:
  - gap opening penalty
  
  $d$ will be large relative to $e$:
  - gap extension penalty

  because you do not want to add too much of a penalty for extending the gap.

Affine Gap Penalties

- Gap penalties:
  - $-d - e$ when there is 2 indel
  - $-d - 2e$ when there are 3 indels
  - $-d - 3e$ when there are 4 indels, etc.
  - $-d - (n-1)e$ when there are $n$ indels

- Somehow reduced penalties (as compared to naïve scoring) are given to runs of horizontal and vertical arrows in the V matrix
Needleman-Wunsch With Affine Gaps

- $\gamma(n) = d + (n-1) \times e$
  - gap open
  - gap extend

- To compute optimal alignment,
  - At position $i, j$, need to “remember” best score if gap is open
  - best score if gap is not open

- $F(i, j)$: score of alignment $x_1...x_i$ to $y_1...y_j$ if $x_i$ aligns to $y_j$
- $G(i, j)$: score if $x_i$ or $y_j$ aligns to a gap

**Initialization:**
- $F(i, 0) = d + (i-1) \times e$
- $F(0, j) = d + (j-1) \times e$

**Iteration:**
- $F(i, j) = \max \left\{ F(i - 1, j - 1) + s(x_i, y_j), G(i - 1, j - 1) + s(x_i, y_j) \right\}$
- $G(i, j) = \max \left\{ F(i, j - 1) - d, F(i - 1, j) - d, G(i, j - 1) - e, G(i - 1, j) - e \right\}$

**Termination:**
- same
Outline: Scoring Alignments

- Scoring alignments
  - Probabilistic meaning
  - Scoring matrices
    - PAM: scoring based on evolutionary statistics
    - BLOSUM: tuning to evolutionary conservation
    - Gaps revisited

- Local vs global alignment

- Database search
  - FASTA
  - BLAST

Local vs. Global Alignment

- The Global Alignment Problem tries to find the highest scoring alignment between input sequences S (of length n) and T (of length m) – S[1-n] and T[1-m].

- The Local Alignment Problem tries to find the highest scoring alignment between the substrings S[i-i’] and T[j-j’], where i,j>0, i’<n+1 and j’<m+1.

  - In the “V matrix” (alignment scores of substrings) with negatively-scored arrows, Local Alignment may score higher than Global Alignment
Local vs. Global Alignment (cont’d)

- Global alignment
  
  ```
  T--CC-C--AGT--TATGT--CAGGGACACG--A--GCATGCAGA--GAC
  AATTGCCGCC--GTCGT--T--TTCAG--CA--GTATG--T--CAGAT--C
  ```

- Local alignment: better alignment to find conserved segment
  
  ```
  tccCAGTTATGTCAgggacagcatcagacagac
  aattgcggccgtcgttttcagCAGTTATGTCAgtc
  ```

Local Alignments: Why?

- Genes are shuffled between genomes
  - Two genes in different species may be similar over short conserved regions and dissimilar over remaining regions.

- Portions of proteins (domains) are often conserved
Local Alignment: Example

- Local run time $O(n^4)$:
  - In the grid of size $n \times n$, there are $\sim n^2$ vertices $(i,j)$ that may serve as a source.
  - For each such vertex computing alignments from $(i,j)$ to $(i',j')$ takes $O(n^2)$ time.
- This can be remedied by giving free rides.
Local Alignment: Free Rides

![Vertex (0,0)](image)

The dashed arrows represent the free rides from (0,0) to every other entry in the V matrix.

Yeah, a free ride!

The Local Alignment Problem

- **Goal**: Find the best local alignment between two sequences
- **Input**: Sequences S, T and scoring matrix $\sigma$
- **Output**: Alignment of sequences S and T whose alignment score is maximum among all possible alignment of all possible substrings
The Smith-Waterman Algorithm

**Idea:** Ignore badly aligning regions

Modifications to Needleman-Wunsch:

**Initialization:** \( V(0, j) = V(i, 0) = 0 \)

**Iteration:** \( V(i, j) = \max \begin{cases} 0 \\ V(i - 1, j) - d \\ V(i, j - 1) - d \\ V(i - 1, j - 1) + s(x_i, y_j) \end{cases} \)

**Power of ZERO:** There is only this change from the original recurrence of a Global Alignment - since there is only one "free ride" arrow entering into every vertex.

**Termination:**

1. If we want the best local alignment...

   \[ V_{\text{OPT}} = \max_{i,j} V(i, j) \]

2. If we want all local alignments scoring > t

   For all \( i, j \) find \( V(i, j) > t \), and trace back
Outline: Scoring Alignments

- Scoring alignments
  - Probabilistic meaning
  - Scoring matrices
    - PAM: scoring based on evolutionary statistics
    - BLOSUM: tuning to evolutionary conservation
  - Gaps revisited

- Local vs global alignment

- Database search
  - FASTA
  - BLAST

Database Search

The problems:
- Dynamic programming: prohibitively complex
- Exact matching: prohibitively mismatch-sensitive

q=TACGAAT.. → AATATAACGAATCAAGAT..
TCGATAGTTAGCAATATΔTAG..
CGAAATAGGTAGCAATACTAC..
ACGACATCGAAGAATAAATAT..

acACGAATTaTACGAATccACGA-T..
tACGAAT-TACGAAT-tACGAaT__
State of Biological Databases

Sequenced Genomes:

<table>
<thead>
<tr>
<th>Organism</th>
<th>Number of Genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>$3 \times 10^9$</td>
</tr>
<tr>
<td>Mouse</td>
<td>$2.7 \times 10^9$</td>
</tr>
<tr>
<td>Rat</td>
<td>$2.6 \times 10^9$</td>
</tr>
<tr>
<td>Fugu fish</td>
<td>$3.3 \times 10^8$</td>
</tr>
<tr>
<td>Tetraodon</td>
<td>$3 \times 10^8$</td>
</tr>
<tr>
<td>Mosquito</td>
<td>$2.8 \times 10^8$</td>
</tr>
<tr>
<td>Drosophila</td>
<td>$1.2 \times 10^8$</td>
</tr>
<tr>
<td>Worm</td>
<td>$1.0 \times 10^8$</td>
</tr>
<tr>
<td>2 sea squirts</td>
<td>$1.6 \times 10^8$</td>
</tr>
<tr>
<td>Rice</td>
<td>$1.0 \times 10^9$</td>
</tr>
<tr>
<td>Arabidopsis</td>
<td>$1.2 \times 10^9$</td>
</tr>
<tr>
<td>Yeast</td>
<td>$1.2 \times 10^7$</td>
</tr>
<tr>
<td>Neurospora</td>
<td>$4 \times 10^7$</td>
</tr>
</tbody>
</table>

14 more fungi within next year

~250 bacteria/viruses

Current rate of sequencing:

4 big labs × $3 \times 10^9$ bp/year/lab
10s small labs

State of Biological Databases

- Number of genes
  - Vertebrate: ~30,000
  - Insects: ~14,000
  - Worm: ~17,000
  - Fungi: ~6,000-10,000
  - Small organisms: 100s-1,000s

- Each known or predicted gene has an associated protein sequence

- >1,000,000 known / predicted protein sequences
Some Useful Applications of Alignments

- Given a newly discovered gene,
  - Does it occur in other species?
  - How fast does it evolve?

- Assume we try Smith-Waterman:

![Diagram of a new gene aligned with a large genomic database]

Some Useful Applications of Alignments

- Given a newly sequenced organism,
  - Which subregions align with other organisms?
    - Potential genes
    - Other biological characteristics

- Assume we try Smith-Waterman:

![Diagram of a newly sequenced mammal aligned with a large genomic database]
Reconsider DP Geometry

- **Diagonal matching segments**: Basis for alignment
- **Alignment**: Connecting matching diagonals
  - With mismatched diagonals or horizontal/vertical gaps

- **Score**: Additive contributions of diagonals and connectors
  - Connectors may reduce the score
  - Focus: high score diagonals, positive score connectors

Dot Matrix Heuristics

**Rule 1**: Find high-scoring diagonals
- Search small diagonal segments
- Extend to max diagonal matches
- Connect diagonals to max score

**Rule 2**: Focus on meaningful alignments
- Filter out low-scoring diagonals
FASTA

- Key idea (Pearson & Lipman 88):
  - Find short diagonals by indexing the DB
  - Extend these to high scoring diagonals
  - Use DP to connect them

- A 4 steps process

Step (a):
Find diagonal matches by indexing

- Key idea: k-mers index of of the DB

- Preprocess:
  - Scan database to index words of size k (k-mers) [k=1..5]

- Query:
  - Scan query to index k-mers
  - Compare hashes to find all diagonal matches of length k
  - Merge short diagonals into maximal diagonal matches
Find Diagonal Matches by Indexing

**Example:**

<table>
<thead>
<tr>
<th>Database d: TATCGATCGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position: 1 2 3 4 5 6 7 8 9 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query q: GATCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position: 1 2 3 4 5</td>
</tr>
</tbody>
</table>

1. **Extract index**

<table>
<thead>
<tr>
<th>d</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAT</td>
<td>1</td>
</tr>
<tr>
<td>ATC</td>
<td>2</td>
</tr>
<tr>
<td>TCG</td>
<td>3</td>
</tr>
<tr>
<td>CGA</td>
<td>4</td>
</tr>
<tr>
<td>GAT</td>
<td>5</td>
</tr>
</tbody>
</table>

2. **Find matches**

<table>
<thead>
<tr>
<th>d</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAT</td>
<td>1</td>
</tr>
<tr>
<td>ATC</td>
<td>2</td>
</tr>
<tr>
<td>TCG</td>
<td>3</td>
</tr>
<tr>
<td>CGA</td>
<td>4</td>
</tr>
<tr>
<td>GAT</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset = q - d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, -4</td>
</tr>
<tr>
<td>0, -4</td>
</tr>
<tr>
<td>-4</td>
</tr>
</tbody>
</table>

3. **Merge diagonal matches**

**FASTA Steps (b-d): Optimize score**

1. **Filter low-score diagonals**

2. **Extend diagonals to max score; keep high-scoring segments**

3. **Use DP in a narrow band around the high scoring segments**
BLAST: Basic Local Alignment Search Tool

- Altschul & Karlin [1990]; a family of algorithms
  - BLAST, WU-BLAST, BlastZ, MegaBLAST, BLAT
- Idea: find matches with significant score statistics
  - Find maximal segment pairs (MSP):
    segments with significant score

BLAST Algorithm

- Step 1: index DB for words of size W (W-mers);
  index query sequence for W-mers with score >Threshold
  - W= 3 for protein, 11 for nucleotides
- Step 2: search for matches with high score (HSP=high scoring pairs)
- Step 3: extend hits to maximal score segments
- Step 4: report matches with score above S
BLAST Step 1-3:
Finding Short High-Scoring Pairs (HSP)

- Create an index of W-mers for database & query
  - For proteins W=3 \(\Rightarrow\) a dictionary of \(20^3=8000\) words

- Match W-mers that score above a threshold T
  - FASTA searches for exact matches of k-mers
  - BLAST searches for high scoring pairs (HSP)
  - Key idea: exploit fast part of the search to max the score rather than push maximization for later, slower, phases

Blast Steps 3-4: Extending Short HSPs

- The short HSPs are extended to increase the score

- Report above threshold HSPs and their scores