Pairwise Sequence Alignment

BMI 877
Mark Craven
craven@biostat.wisc.edu
February 20, 2014

Global Ocean Sampling Expedition

• the GOSE sequenced metagenomic samples from throughout the oceans
• discovered millions of new genes, thousands of new protein families
• How do you compare a sequence to previously known sequences?

• How do you know if you have found something new?

Pairwise alignment: task definition

Given
– a pair of sequences (DNA or protein)
– a method for scoring a candidate alignment

Do
– determine the correspondences between substrings in the sequences such that the similarity score is maximized
The role of homology in alignment

- *homology*: similarity due to descent from a common ancestor

- often we can infer homology from sequence similarity

- thus we can sometimes infer structure/function from sequence similarity

Mismatches and gaps

- substitutions in *homologous* sequences result in mismatches in an alignment

- insertions/deletions in *homologous* sequences result in gaps in an alignment

```
CA--GATTCGAAT
CGCCGATT---AT
```

mismatch

gap
DNA alignment example

Genomic sequence for part of the 6T6Gal gene in mouse and rat

Protein alignment example

Alignment of the PhaK protein from Pseudomonas putida and OprD protein from Pseudomonas aeruginos
Types of alignment

- **global**: find best match of both sequences in their entirety
- **local**: find best subsequence match
- **semi-global**: find best match without penalizing gaps on the ends of the alignment

Scoring an alignment: what is needed?

- substitution matrix
  - $s(a, b)$ indicates score of aligning character $a$ with character $b$
- gap penalty function
  - $w(g)$ indicates cost of a gap of length $g$
Blosum 62 substitution matrix

Linear gap penalty function

- different gap penalty functions require somewhat different dynamic programming algorithms

- the simplest case is when a linear gap function is used

\[ w(g) = -g \times d \]

where \( d \) is a constant

- we’ll start by considering this case
Scoring an alignment

- the score of an alignment is the sum of the scores for pairs of aligned characters plus the scores for gaps

- example: given the following alignment

\[
\begin{align*}
\text{VAHV-} & \text{--D--DMPNALSALS} \\
\text{D} & \text{DMPNALSALS} \\
\text{LH} & \text{AHKL} \\
\text{AIQLQVTGVVVTDATLKNLGS} & \text{HVHSKG}
\end{align*}
\]

- we would score it by

\[
s(V, A) + s(A, I) + s(H, Q) + s(V, L) - 3d + s(D, G) - 2d \ldots
\]

The space of global alignments

- some possible global alignments for ELV and VIS

<table>
<thead>
<tr>
<th>ELV</th>
<th>-ELV</th>
<th>--ELV</th>
<th>ELV-</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIS</td>
<td>VIS-</td>
<td>VIS--</td>
<td>-VIS</td>
</tr>
<tr>
<td>E-LV</td>
<td>ELV--</td>
<td>EL-V</td>
<td></td>
</tr>
<tr>
<td>VIS-</td>
<td>--VIS</td>
<td>-VIS</td>
<td></td>
</tr>
</tbody>
</table>

- Can we find the highest scoring alignment by enumerating all possible alignments and picking the best?
Number of possible alignments

- given sequences of length \( m \) and \( n \)
- assume we don’t count as distinct \( c^- \) and \( g^- \)
- we can have as few as 0 and as many as \( \min\{m, n\} \) aligned pairs
- therefore the number of possible alignments is given by

\[
\sum_{k=0}^{\min\{m,n\}} \binom{n}{k} \binom{m}{k} = \binom{n+m}{n}
\]

- e.g. two sequences of length 100 have \( \approx 10^{77} \) possible alignments

Pairwise alignment via dynamic programming


- *dynamic programming*: solve an instance of a problem by taking advantage of computed solutions for smaller subparts of the problem

- determine best alignment of two sequences by determining best alignment of all prefixes of the sequences
Dynamic programming idea

• consider last step in computing alignment of AAAC with AGC

• three possible options; in each we'll choose a different pairing for end of alignment, and add this to best alignment of previous characters

$$\begin{array}{ccc}
\text{AAA} & \text{C} & \text{AAAC} \\
\text{AG} & \text{C} & \text{AG} \\
\text{AAA} & \text{C} & \text{AGC} \\
\text{AGC} & - & -
\end{array}$$

Consider best alignment of these prefixes + score of aligning this pair

Dynamic programming idea

• given an $n$-character sequence $x$, and an $m$-character sequence $y$

• construct an $(n+1) \times (m+1)$ matrix $F$

• $F(i,j) = \text{score of the best alignment of } x[1...i] \text{ with } y[1...j]$
DP algorithm for global alignment with linear gap penalty

• one way to specify the DP is in terms of its recurrence relation:

\[
F(i, j) = \max \begin{cases} 
F(i-1, j-1) + s(x_i, y_j) \\
F(i-1, j) - d \\
F(i, j-1) - d 
\end{cases}
\]

DP algorithm sketch: global alignment

• initialize first row and column of matrix

• fill in rest of matrix from top to bottom, left to right

• for each \( F(i, j) \), save pointer(s) to cell(s) that resulted in best score

• \( F(m, n) \) holds the optimal alignment score; trace pointers back from \( F(m, n) \) to \( F(0, 0) \) to recover alignment
Global alignment example

• Suppose we choose the following scoring scheme:

\[ s(x_i, y_i) = \]

\[ +1 \text{ when } x_i = y_i \]

\[ -1 \text{ when } x_i \neq y_i \]

\[ d \text{ (penalty for aligning with a gap)} = 2 \]
DP comments

• works for either DNA or protein sequences, although the substitution matrices used differ

• finds an optimal alignment

• the exact algorithm (and computational complexity) depends on gap penalty function (we’ll come back to this issue)

Computational complexity

• initialization: $O(m), O(n)$ where sequence lengths are $m, n$
• filling in rest of matrix: $O(mn)$
• traceback: $O(m + n)$
• hence computational complexity is $O(mn)$
Local alignment

- so far we have discussed *global alignment*, where we are looking for best match between sequences from one end to the other

- often we want a *local alignment*, the best match between subsequences of $x$ and $y$

Example local alignment

- aligning my name against the sequence for dTDP-4-dehydrorhamnose reductase from the bacterium *opitutus terrae*

MARKCRAVEN
...LSGAYHLAASGHTSWHGFASAIIDLMPDARKCRAVEAIT...
Local alignment DP algorithm


- interpretation of array values is somewhat different: 
  \[ F(i, j) = \text{score of the best alignment of a suffix of } 
  x[1\ldots i] \text{ and a suffix of } y[1\ldots j] \]

Local alignment DP algorithm

- the recurrence relation is slightly different than for global algorithm

\[
F(i, j) = \max \begin{cases} 
F(i-1, j-1) + s(x_i, y_j) \\
F(i-1, j) - d \\
F(i, j-1) - d \\
0 
\end{cases}
\]
Local alignment DP algorithm

- initialization: first row and first column initialized with 0’s

- traceback:
  - find maximum value of $F(i, j)$; can be anywhere in matrix
  - stop when we get to a cell with value 0

Local alignment example

```
          A   A   G   A
T   0   0   0   0   0   0
T   0   0   0   0   0   0
A   0   1   1   0   1
A   0   1   2   0   1
G   0   0   0   3   1
```

x:   A   A   G
y:   A   A   G
More on gap penalty functions

- a gap of length $k$ is more probable than $k$ gaps of length 1
  - a gap may be due to a single mutational event that inserted/deleted a stretch of characters
  - separated gaps are probably due to distinct mutational events

- a linear gap penalty function treats these cases the same

- it is more common to use gap penalty functions involving two terms
  - a penalty $d$ associated with opening a gap
  - a smaller penalty $e$ for extending the gap

Gap penalty functions

linear

$$w(g) = -g \times d$$

affine

$$w(g) = \begin{cases} -d - (g - 1)e, & g \geq 1 \\ 0, & g = 0 \end{cases}$$
Dynamic programming for the affine gap penalty case

- to do in $O(nm)$ time, need 3 matrices instead of 1

$M(i, j)$ best score given that $x[i]$ is aligned to $y[j]

$I_x(i, j)$ best score given that $x[i]$ is aligned to a gap

$I_y(i, j)$ best score given that $y[j]$ is aligned to a gap

Global alignment DP for the affine gap penalty case

$M(i, j) = \max\begin{cases} M(i - 1, j - 1) + s(x_i, y_j) \\ I_x(i - 1, j - 1) + s(x_i, y_j) \\ I_y(i - 1, j - 1) + s(x_i, y_j) \end{cases}$

$I_x(i, j) = \max\begin{cases} M(i - 1, j) - d \\ I_x(i - 1, j) - e \end{cases}$

$I_y(i, j) = \max\begin{cases} M(i, j - 1) - d \\ I_y(i, j - 1) - e \end{cases}$
Global alignment DP for the affine gap penalty case

- initialization
\[ M(0,0) = 0 \]
\[ I_x(i, 0) = -d - (i - 1)e \quad \text{for } i > 0 \]
\[ I_y(0, j) = -d - (j - 1)e \quad \text{for } j > 0 \]
other cells in top row and leftmost column = $-\infty$

- traceback
  - start at largest of \( M(m,n), I_x(m,n), I_y(m,n) \)
  - stop at \( M(0,0) \)
  - note that pointers may traverse all three matrices

Global alignment example (affine gap penalty)

d = 4, e = 1
Global alignment example (continued)

three optimal alignments: ACACT, ACACT, ACACT

Pairwise alignment summary

- the number of possible alignments is exponential in the length of sequences being aligned
- dynamic programming can find optimal-scoring alignments in polynomial time
- the specifics of the DP depend on
  - local vs. global alignment
  - gap penalty function
- affine penalty functions are most commonly used
Heuristic alignment motivation

- $O(mn)$ too slow for large databases with high query traffic
- heuristic methods do fast approximation to dynamic programming
  - FASTA [Pearson & Lipman, 1988]
  - BLAST [Altschul et al., 1990; Altschul et al., Nucleic Acids Research 1997]

Heuristic alignment motivation

- consider the task of searching UnitProtKB/Swiss-Prot against a query sequence:
  - say our query sequence is 362 amino-acids long
  - most recent release of DB contains 192,888,369 amino acids
  - finding local alignments via dynamic programming would entail $O(10^{11})$ matrix operations
- many servers handle thousands of such queries a day (NCBI > 500,000)
Heuristic alignment

• heuristic algorithm: a problem-solving method which isn’t guaranteed to find the optimal solution, but which is efficient and finds good solutions

• key heuristics in BLAST
  – look for seeds of high scoring alignments
  – use dynamic programming selectively

• key tradeoff made: sensitivity vs. speed
  
sensitivity = \frac{\text{# significant matches detected}}{\text{# significant matches in DB}}

Overview of BLAST
(Basic Alignment Search Tool)

• given: query sequence \( q \), word length \( w \), word score threshold \( T \), segment score threshold \( S \)
  – compile a list of “words” (of length \( w \)) that score at least \( T \) when compared to words from \( q \)
  – scan database for matches to words in list
  – extend all matches to seek high-scoring alignments

• return: alignments scoring at least \( S \)
Determining query words

Given:
query sequence: QLNFSAGW
word length \( w = 2 \) (default for protein usually \( w = 3 \))
word score threshold \( T = 9 \)

Step 1: determine all words of length \( w \) in query sequence

QL LN NF FS SA AG GW

Step 2: determine all words that score at least \( T \) when compared to a word in the query sequence

<table>
<thead>
<tr>
<th>words from sequence</th>
<th>query words w/ ( T \geq 9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>QL</td>
<td>QL=9</td>
</tr>
<tr>
<td>LN</td>
<td>LN=10</td>
</tr>
<tr>
<td>NF</td>
<td>NF=12, NY=9</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>none</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Scanning the database

- search database for all occurrences of query words
  - approach:
    - index database sequences into table of words (pre-compute this)
    - index query words into table (at query time)

Extending hits in BLAST

- BLAST extends hits into local alignments
  - key idea: do extension only when there are two hits on the same diagonal within distance $A$ of each other
  - dynamic programming used to extend pairs of hits
Extending hits in BLAST

'•' hits with $T > 10$

'+' hits with $T > 12$

extend these cases

Figure from: Altschul et al. *Nucleic Acids Research* 25, 1997

BLAST programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Query</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLASTP</td>
<td>Protein</td>
<td>Protein</td>
</tr>
<tr>
<td>BLASTN</td>
<td>DNA</td>
<td>DNA</td>
</tr>
<tr>
<td>BLASTX</td>
<td>Translated DNA</td>
<td>Protein</td>
</tr>
<tr>
<td>TBLASTN</td>
<td>Protein</td>
<td>Translated DNA</td>
</tr>
<tr>
<td>TBLASTX</td>
<td>Translated DNA</td>
<td>Translated DNA</td>
</tr>
</tbody>
</table>

BLAST results

<table>
<thead>
<tr>
<th>Source (Accession)</th>
<th>Description</th>
<th>E Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>gb</td>
<td>AAN84548.1</td>
<td>beta globin chain variant [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AAK29639.1</td>
<td>beta globin chain variant [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AAK00489.1</td>
<td>hemoglobin beta subunit variant [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AAA39957.1</td>
<td>beta-globin</td>
</tr>
<tr>
<td>gb</td>
<td>AAK37051.1</td>
<td>hemoglobin beta [synthetic construct]</td>
</tr>
<tr>
<td>gb</td>
<td>AAK66298.1</td>
<td>hemoglobin beta [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AAL69978.1</td>
<td>mutant beta-globin [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AAK29557.1</td>
<td>hemoglobin beta [synthetic construct]</td>
</tr>
<tr>
<td>ref</td>
<td>XP_00523492.1</td>
<td>beta globin [Homo sapiens]</td>
</tr>
<tr>
<td>ref</td>
<td>XP_00523492.1</td>
<td>hemoglobin subunit beta [Hemoglobin beta chain]</td>
</tr>
<tr>
<td>gb</td>
<td>AAK10106.1</td>
<td>hemoglobin beta chain [Homo sapiens]</td>
</tr>
<tr>
<td>ref</td>
<td>CA26204.1</td>
<td>beta-globin [Pan troglodytes]</td>
</tr>
<tr>
<td>gb</td>
<td>AAN16468.1</td>
<td>hemoglobin beta chain variant Hb.Sinai-Bel Air [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AB047031.1</td>
<td>hemoglobin [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AA912923.1</td>
<td>hemoglobin beta [Homo sapiens]</td>
</tr>
<tr>
<td>ref</td>
<td>CA26432.1</td>
<td>beta-globin [Gorilla gorilla]</td>
</tr>
<tr>
<td>gb</td>
<td>AAK39225.1</td>
<td>beta globin chain [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AAK00220.1</td>
<td>mutant beta globin [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AAN11321.1</td>
<td>hemoglobin beta chain variant Hb-E_toulouse [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AA966184.1</td>
<td>mutant beta-globin [Homo sapiens]</td>
</tr>
<tr>
<td>gb</td>
<td>AA9532128.1</td>
<td>hemoglobin, beta (predicted) [Papio anubis]</td>
</tr>
<tr>
<td>gb</td>
<td>AAD50656.1</td>
<td>hemoglobin [Homo sapiens]</td>
</tr>
<tr>
<td>pco</td>
<td>100114</td>
<td>Chain B, High-Resolution X-Ray Study Of Deoxynemoglobin</td>
</tr>
</tbody>
</table>
BLAST comments

• it’s heuristic: may miss some good matches

• it’s fast: empirically, 10 to 50 times faster than Smith-Waterman

• large impact:
  – NCBI’s BLAST server handles more than 500,000 queries a day
  – most used bioinformatics program in the world