Pairwise Sequence Alignment using Dynamic Programming

Russ B. Altman, MD, PhD

BMI 214
CS 274

What is sequence alignment?

Given two sequences of letters, and a scoring scheme for evaluating matching letters, find the optimal pairing of letters from one sequence to letters of the other sequence.

Align:
THIS IS A RATHER LONGER SENTENCE THAN THE NEXT.
THIS IS A SHORT SENTENCE.

THIS IS A RATHER LONGER SENTENCE THAN THE NEXT.
THIS IS A ######SHORT#### SENTENCE####################
OR
THIS IS A SHORT#########SENTENCE###############.

Aligning biological sequences

• DNA (4 letter alphabet + gap)
  TTGACAC
  TTTACAC

• Proteins (20 letter alphabet + gap)
  RKVA---GMAKPNM
  RKIAVAAAASKPAV

Statement of Problem

Given
• 2 sequences
• scoring system for evaluating match(or mismatch) of two characters
• penalty function for gaps in sequences

Produce

Optimal pairing of sequences that retains the order of characters in each sequence, perhaps introducing gaps, such that the total score is optimal.

Why align sequences?

Lots of sequences with unknown structure and function. A few sequences with known structure and function.

• If they align, they are similar, maybe due to common descent.

• If they are similar, then they might have same structure or function.

• If one of them has known structure/function, then alignment to the other yields insight about how the structure or function works.

Multiple alignment

Pairwise alignment (two at a time) is much easier than multiple alignment (N at a time).

This is a rather longer sentence than the next.
This is a short sentence.
This is the next sentence.
Rather long is the next concept.
Rather longer than what is the next concept.
Drawing alignments

Exact Matches OK, Inexact Costly, Gaps cheap.
This is a rather longer sentence than the next.
This is a sentence.

Exact Matches OK, Inexact Costly, Gaps cheap.
This is a rather longer sentence than the next.
This is a sentence.

Exact Matches OK, Inexact Moderate, Gaps cheap.
This is a rather longer sentence than the next.
This is a sentence.

Exact Matches cheap, Inexact cheap, Gaps expensive.
This is a rather longer sentence than the next.
This is a short sentence.

Multiple Alignment (NP-hard)

This is a rather longer sentence than the next.
This is a very long sentence.

Problems with dot matrices

- Rely on visual analysis
- Difficult to find optimal alignments
- Need scoring schemes more sophisticated than “identical match”
- Difficult to estimate significance of alignments

There used to be dot matrices.

- Put one sequence along the top row of a matrix.
- Put the other sequence along the left column of the matrix.
- Plot a dot everytime there is a match between an element of row sequence and an element of the column sequence.
- Diagonal lines indicate areas of match.

Gaps

The thing that makes alignment hard is the possibility that gaps are introduced in one sequence (corresponding to a shortening of the protein chain, for example).
Gaps

The computational complexity of alignment when gaps are allowed anywhere is exponential in the length of the sequences being aligned.

Informally, we must generate sequences with gaps in every position and of every length, and compare these sequences with a dot-matrix like computation that is $O(N^2)$.

Dynamic Programming

- An age old computer science concept...
  “Frequently, however, there are only a polynomial number of subproblems... if we keep track of the solution to each subproblem solved, and simply look up the answer when needed, we obtain a polynomial-time algorithm.”
  --Aho, Hopcroft, Ullman

- Reported to biologists for sequences by Needleman & Wunsch, 1969 (course reader).

- Computational complexity can be $O(N^2)$ since there are only $N^2$ subproblems. As we present, will be $O(N^3)$.

DP Ideas (from Gonnet)

- Compute at each new state the maxima (or minima) of the values for all possible transitions from previous states.

- Once the final state is reached, backtrack to find the path(s), which lead to the optimal final state.

- It is clear that the set of all transitions must not have cycles. It is also assumed that the total number of states is polynomial in the size of the problem. Hence the computation of each node will require polynomial time.

- Each transition is (possibly) associated with a cost. The computation of each state requires the use of optimal values of previous states and the cost of the transitions.

Biology of Gaps

AGKLAVRSTMIESTRVILTWRKW
AGKLAVRS###IE###RVILTWRKW
vs.
AGKLAVRSTMIEST###RVILTWRKW
AGKLAVRS#####IERVILTWRKW

Note on Finite-state Automata

Automata take strings and accept/reject them based on criteria. They search for patterns. If you allow them to produce output, you can create an audit of the search.

Dynamic programming algorithms can, in general, be formulated as finite state automata.

Transducers are automata that take an input string and creates output string. Sequence alignment can be formulated as a transducer.

Key Ideas behind D-P algorithm

The score of the best possible alignment that ends at a given pair of positions $(i,j)$ in the two sequences is the score of the best alignment previous to those two positions PLUS the score for aligning those two positions.

Best Previous Alignment

$$ i \quad j $$

$\Rightarrow$
New Best Alignment = Previous Best + Local Best

Score of Best Previous Alignment

Mathematical recurrence relation

\[ S_{i-1, j-1} \text{ or } \max_{2 \leq x \leq i} S_{i-x, j-1} + w_{x-1} \]
\[ \max_{2 \leq y \leq l} S_{i-1, j-y} + w_{y-1} \]

(\text{big } S) \ S_{ij} \text{ is score for alignment ending at } i \text{ in seq1 and } j \text{ in seq2}

(\text{small } s) \ s_{ij} \text{ is score for aligning character } i \text{ and } j

w_x \text{ is the score for a gap of length } x \text{ in seq1}

w_y \text{ is the score for a gap of length } y \text{ in seq2}

Always search the previous row and column for best previous alignment

How is it implemented? Three Steps

1.  Produce a sequence vs. sequence matrix, and fill in from [0,0] to [N, M] the best possible scores for alignments including the residues at [i,j]. (Also keep track of dependencies of scores, usually in a “pointer matrix.”) \ O(NM) 
2.  Find the best score in the entire matrix
3.  Trace back through pointer matrix to get position by position alignment of elements of sequence, including gaps.

1. Create and fill matrix using recurrence relation.

\[ \text{BestScore}[i,j] = \text{BestScore}[<i, <j] + \text{Match}[i,j] + \text{Gap Cost} \]

(Also maintain pointer to previous best)

A A T G C
A 1 1 0 0 0
G
G
C

From Gribskov example...
Assume no gap penalty…

BestScore[i,j] = BestScore[<i, <j] + Match[i,j] + Gap

A A T G C
A 1 1 0 0 0
G 0 1 1 2 1
G
C
2. Find best score in matrix
- For optimal GLOBAL alignment, we want best score in the final row or final column.

GLOBAL—best alignment of entirety of both sequences.
- For optimal LOCAL alignment, we want best score anywhere in matrix (we will discuss further in a moment).

LOCAL—best alignment of segments, without regard to rest of two sequences.

3. Reconstruct alignment working from scoring matrix.
- Create alignment in reverse order from scoring (i.e., start at highest score)
- Follow pointer back to (one of) previous best scores.
  ° if SeqB[i-1] and SeqA[j-1] then report
  ° if SeqB[i-1] and SeqA[j-N] then report matches to gaps for A[j-1] ... A[j-(N-1)]
  ° if SeqB[i-N] and SeqA[j-1] then report matches to gaps for B[i-1] ... B[i-(N-1)]

A note about complexity
- The complexity of filling in the matrix is apparently \( O(N^3) \) because we fill in a matrix of size \( N \times N \), and each point must search the previous row/col \( O(N) \) for best score.

- The actual complexity can be \( O(N^2) \) as long as we CACHE the best score in each row and each column for constant time lookup! Requires a linear gap penalty.

(Gotoh paper in “classics” for the interested)
Following the lowest pointer

```
C → GC  → ATGC  → AATGC
C → G_C  → A_GGC  → _A_GGC
```

Following the highest pointer

```
C → G_GC  → AG_GC
C → GGC  → A_GGC
```

In this case, gap penalty of \( g=1, l=1 \) favors the lower path over higher path (1 gap vs. 2 gaps).

---

**A note on computational complexity**

The traceback routine can be quite expensive if ALL possible optimal paths are required, since there may be lots of branches.

However, if choices are made arbitrarily about which pointers to follow, then it can be reconstructed in \( O(N) \)—by simply traversing \( O(N) \) pointers from lower right to upper left...

---

**Gap Penalties**

- The score at a position can also factor in a penalty for introducing gaps (i.e., not going from \( i,j \) to \( i-1,j-1 \)). Gap penalties are often of linear form:

\[
\text{Gap} = g + l \cdot n
\]

- Gap is the gap penalty
- \( g \) is the cost of opening a gap
- \( l \) is the cost of extending the gap by one
- \( n \) is the length of the gap

---

**Mathematical recurrence relation**

\[
S_{i,j} = s_{ij} + \max \begin{cases} 
S_{i-1,j-1} & \text{or} \\
\max S_{i-1,j-y} + w_{y-1} & (2 \leq y \leq I) \\
\max S_{i-x,j-1} + w_{x-1} & (2 \leq x \leq i) 
\end{cases}
\]

- \( s_{ij} \) is the score for aligning character \( i \) and \( j \)
- \( S_{i,j} \) is score for alignment ending at \( i \) in seq1 and \( j \) in seq2
- \( w_x \) is the score for a gap of length \( x \)

---

**What about “end gaps”**

Should we penalize alignments for the gaps that are left at the end?

\[
\begin{align*}
\text{ATCCG} & \text{CATA} \text{CG} \text{GA} \quad ? \quad \text{CCGC} \text{ATA} \text{C} \\
\text{--CCGC} & \text{ATA} \text{C} \text{--} \quad = \quad \text{CCGC} \text{ATA} \text{C}
\end{align*}
\]

If entire sequences are supposed to be similar, then reasonable (responsible for entire length).

If sequences are of very different lengths, then usually not (looking for best global match within a segment).
**Match Matrices**

- The degree of match between two letters can be represented in a matrix.
- Area of active research
- Changing matrix changes alignments
  - context-specific matching
  - information theoretic interpretation of scores
  - modeling evolution with different matrices
- Matrix is compact summary of how to map one alphabet to another.

**A Sample Match Matrix for the amino acids (PAM-250).**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>R</th>
<th>N</th>
<th>D</th>
<th>C</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>-2</td>
<td>6</td>
<td>0</td>
<td>-1</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>-4</td>
<td>-1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>-1</td>
<td>2</td>
<td>4</td>
<td>-5</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>-2</td>
<td>-4</td>
<td>-4</td>
<td>-5</td>
<td>12</td>
<td>-5</td>
</tr>
<tr>
<td>Q</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>-5</td>
<td>4</td>
</tr>
</tbody>
</table>

**Where do matrices come from?**

1. Manually align protein structures (or, more risky, sequences)
2. Look for frequency of amino acid substitutions at structurally nearly constant sites.
3. Entry $\sim \log\left(\frac{\text{freq( observed )}}{\text{freq( expected )}}\right)$
   - $\rightarrow$ More likely than random
   - $\rightarrow$ At random base rate
   - $\rightarrow$ Less likely than random

**Statistical Significance of Scores**

Match matrix elements, $s(i,j)$, provide score for substituting amino acid $i$ for amino acid $j$:

If:

- $q(i,j) =$ desired frequency of aligning $i$ and $j$
- $p(i) =$ probability of amino acid $i$ (average 5%)

then

$$S(i,j) = \frac{\ln \left( \frac{q(i,j)}{p(i)p(j)} \right)}{\lambda}$$  

($\lambda$ is a fudge factor to give scores good range)

or

$$q(ij) = p(i)p(j) \exp(\lambda * S(i,j))$$

Alphabets need not be the same.

The match matrix need not be symmetric, or even have the same set of letters along top row and left column.

If two different alphabets, then can be used to relate sequences in one alphabet to sequences in another.
Can relate amino acids to environments...

| E1  | E2  | E3  | E4  | E5  | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-0.77</td>
<td>-1.05</td>
<td>-0.54</td>
<td>-0.65</td>
<td>-1.52</td>
</tr>
<tr>
<td>R</td>
<td>-1.80</td>
<td>-1.52</td>
<td>-2.35</td>
<td>-0.11</td>
<td>-0.41</td>
</tr>
<tr>
<td>N</td>
<td>-1.76</td>
<td>-2.18</td>
<td>-2.61</td>
<td>-0.48</td>
<td>-0.26</td>
</tr>
<tr>
<td>D</td>
<td>-2.48</td>
<td>-1.80</td>
<td>-2.63</td>
<td>-0.80</td>
<td>-2.08</td>
</tr>
<tr>
<td>C</td>
<td>-0.43</td>
<td>-0.45</td>
<td>-0.59</td>
<td>0.15</td>
<td>-0.72</td>
</tr>
<tr>
<td>Q</td>
<td>-1.38</td>
<td>-2.03</td>
<td>-0.84</td>
<td>0.16</td>
<td>-0.79</td>
</tr>
</tbody>
</table>

Altschul SF.
Amino acid substitution matrices from an information theoretic perspective.


Jones DT; Taylor WR; Thornton JM.
The rapid generation of mutation data matrices from protein sequences.


Global vs. Local Alignment

- **Global alignment**: find alignment in which total score is highest, perhaps at expense of areas of great local similarity.

- **Local alignment**: find alignment in which the highest scoring subsequences are identified, at the expense of the overall score.

Local alignment can be done with minor modifications of the global alignment algorithm!

Three modifications for local alignment

1. The scoring system uses negative scores for mismatches.
2. The minimum score for \([i,j]\) is zero.
3. The best score is sought anywhere in matrix (not just last column or row)

*These three changes cause the algorithm to seek high scoring subsequences, which are not penalized for their global effects (modification 2), which don’t include areas of poor match (mod 1) and which can occur anywhere (mod 3).*

Local alignment asks different question than Global alignment.

**NOT**: Are these two sequences generally the same?

**BUT**: Do these two sequences contain high scoring subsequences?

*Local similarities may occur in sequences with different structure or function that share common substructure/subfunction.*

Other methods for Alignment

- \(O(N^2)\) is too slow for large databases
- Heuristic methods based on frequency of shared subsequences
- Usually look for ungapped small sequences

See, for example, FASTA, BLAST, BLAZE.
**FASTA**

- Use hash table of short words of the query sequence. Short = 2 to 6 characters.
- Go through database and look for matches in to the query hash table (linear in size of database)
- Score matching segments based on content of these matches. Extend the good matches empirically.

\[
\begin{array}{cccccccc}
\text{Seq0} & \text{Seq1} & \text{Seq2} & \text{Seq3} & \text{Seq4} & \text{Seq5} & \text{Seq6} & \ldots & \text{SeqN-1} & \text{SeqN} \\
\text{Word 0} & \text{Word 1} & \text{Word 2} & \text{Word 3} & \text{Word 4} & \text{Word 5} & \text{Word 6} & \ldots & \text{Word N} \\
\end{array}
\]

**BLAST**

- Karlin (Stanford Dept. Math) and Altschul
- Finds inexact, ungapped “seeds” using a hashing technique (like FASTA) and then extends the seed to maximum length possible.
- Based on strong statistical/significance framework “What is a significantly high score of two segments of length N and M?”
- Most commonly used for fast searches and alignments. New versions now do gapped segments.

**Maximal Segment Pair**

A maximal segment pair (MSP) is the pair of equal length segments of two sequences that, when aligned, have the greatest local alignment score.

The number of segments between two sequences with score of S (or greater) can be approximated:

\[ = K N \exp(-\lambda S) \]

where \( K \) is a constant that can be estimated from theory, and \( N \) is the product of the sequence lengths

(Karlin & Altschul, PNAS 87, 1990, p. 2264-2268)

**Significance of Scores**

For ungapped, local alignments, what is a significant score?

The score of an alignment will depend on:

- the individual match scores (\( \lambda + \) frequencies),
- the lengths of the two sequences being aligned,
- the gap penalties

Gap penalties introduce complications and will not be considered here.

**Importance of Significance**

The significance of a local, ungapped alignment can therefore be estimated by comparing the score that is obtained with the maximal MSP score expected based on the scoring system used and the length of the sequences.

*It has been shown that such estimates of significance are much more sensitive than those based on percent identity in the alignment.*

This is basis of significance numbers provided in BLAST.

**Words: Similarity vs. Homology**

Sequence similarity can be measured in many ways:

- % of identical residues in an alignment
- % of “conservative” mutations in an alignment

(but what is significant?)

Homology implies a common ancestry of the two sequences

Similarity may be used as evidence of homology, but does not necessarily imply homology.
Types of homology
Orthologs are genes that have the same function in various species, and that have arisen by speciation.
Paralogs are other members of multigene families

\[ \text{A} \quad \text{Species} \quad \text{A is the parent gene} \]
\[ \text{B} \quad \text{C} \quad \text{B and C are ORTHOLOGS} \]
\[ \text{Dduplication} \quad \text{C'} \quad \text{B and C are PARALOGS} \]

Types of Match Matrix
1. Matrices based on observed rates of substitution in sequences aligned using structural criteria (PAM, BLOSUM)
2. Matrices based on analysis of amino acid properties and similarities to predict substitutability.
3. Matrices based on analysis of environments in which substitution takes place (WAC).

Summary
1. Dynamic programming algorithm finds optimal alignment in \( O(N^2) \).
2. Critical user choices are gap penalty function, matching matrix, and local vs. global preference.
3. Multiple uses in sequence & structure comparisons.