

Pairwise Sequence Alignment

15.6.08

Biological Meaning of Alignment.

Alignment (pairwise and multiple) is extremely central in biological sequence analysis. Some of the purposes in aligning sequences are:

- i. Reconstructing Molecular Evolution.
- ii. Matching of Functionally Equivalent Regions.
- iii. Definition of patterns the sequences must contain.

Only alignment of homologous sequences seems meaningful as it is unclear what an alignment of non-homologous sequences means. So as viewed here, an alignment is a partial reconstruction of the path of molecular evolution.

Solving ii. above is highly motivating for a molecular biologist. The moment a new molecule A is found to be homologous to an old well studied molecule B, much of the knowledge of B probably is true for A as well. This is because function and structure evolves much slower than sequences do. Sequence alignments can also be used to transfer knowledge on a finer scale - now regions are mapped to regions and a similar tentative transfer of knowledge is possible. Sometimes functional and structural is absent for both sequences aligned, but certain areas are conserved and are then focus of extra experimental attention.

Pairwise alignments are in general inferior to multiple alignment as multiple alignments make use of a larger amount of data. However, multiple alignment is a much harder problem, both computationally and conceptually. Many multiple alignment methods are built up from pairwise alignment methods.

Basic Pairwise Algorithm & Problem.

The underlying assumption of the "pairwise sequence alignment problem" as presented here is:

- i. The two sequences are homologous, i.e. they have evolved from a common ancestor.
- ii. Differences between them are due to only two kinds of events, namely insertion-deletions (*indels*) and substitutions (change of single elements of the sequence - amino acids if the sequence is a protein and nucleic acid, if the sequence is DNA).

The most widely used approach to finding an alignment is to apply the principle of Parsimony: Assume the alignment that postulates the fewest events is "good". Knowledge that different events occur with very different frequencies has led to weighted parsimony - different kinds of events can have different weights.

The first algorithm to align sequences was given by Levenstein in 1966 and is presented here.

The following three points should be illustrated by the figure:

	40	32	22	14	9	17
T						
	30	22	12	4	12	22
G						
	20	12	2	12	22	32
T						
	10	2	10	20	30	40
T						
	0	10	20	30	40	50
		C	T	A	G	G

Alignment: CTAGG
 i v Cost 17
 TT-GT

Two sequences, CTAGG (x-axis 0-5) and TGGT (y-axis 0-4), are given and are to be aligned. A transition (C-T or G-A) cost 2, transversions (the remaining events) cost 5 and an insertion-deletion 10. The entry (i,j) in the matrix will contain the cost of transforming s1[1:i] into s2[1:j]. The entry (0,0) corresponds to the cost of transforming the empty sequence into the empty sequence and has cost 0 per definition. The entry (5,4) is the cost of transforming s1 into s2. To explain the algorithm, look at entry (4,3) that has value 12, but pretend it isn't known, but the entries (3,2), (4,4) and (3,5) are. What could be the value of (4,3) be? It must be the best possibility of 4+10, 12+0 or 22+10, since the shortest path from (0,0) to (3,4) must go through (3,2), (4,4) or (3,5) and then take one step (edge) to reach (3,4). Clearly the matrix can be filled out row-by-row, column-by-column or by a moving diagonal $i+j=k$ and letting k increase.

Applying this procedure allows the calculation of the entry (5,4) and the distance between the two sequences has been calculated. Which events actually transform s1 into s2? This (called backtracking) can be obtained by using the same reasoning as above. How was 17 obtained in the right top corner? It must have come from (4,4), (4,3) or (5,3), but only $17=12$ (value at (4,3))+ 5 (cost of changing T→G) is true. Thus the last nucleotide has a mutation. Place G on top of T on two lines. The same line of reasoning can now be applied to (4,3) revealing that before TG there must be a column GG. Etc etc. Scanning the obtained alignment will reveal events with a cumulative cost of 17.

1. There is a 1-1 correspondence between alignments and a special kind of paths from node (0,0) to node (l_1, l_2) in a graph with $(l_1+1)*(l_2+1)$ nodes, where $i=0,1,2,\dots,l_1$ and $j=0,1,2,\dots,l_2$. A node (i,j) is connected to three predecessors $(i-1,j-1)$, $(i,j-1)$ and $(i-1,j)$ (negative indices are not allowed). I.e. from (i,j) , where $i>1$ & $j>1$ there are three incoming edges, one vertical, one horizontal and one diagonal edges corresponding to matches of $s1[i]$ (the i 'th base of sequence 1) and $s2[j]$ (the j 'th base of sequence 2). Each of these edges are weighted: The horizontal and vertical edges have the weight of an indel. The edge $(i-1,j-1)-(i,j)$ has weight $d(s1[i],s2[j])$ - the distance between the two elements (zero if they are identical).

A bit of notation: $s1$ and $s2$ are sequences of length l_1 and l_2 , respectively. $s1[i]$ is the i th element of $s1$, $s1_i$ is the sequence consisting of the first i elements of $s1$. Correspondingly for $s2$. $D_{i,j}$ is the distance between $s1_i$ and $s2_j$, i.e. the smallest possible weight of the events transforming $s1_i$ into $s2_j$.

2. The cost of an alignment is the cost of each column in the alignment. So the cost of an alignment of $s1_i$ and $s2_j$ ($D_{i,j}$) must be one of following three quantities

$$D_{i-1,j-1} + d(s1[i],s2[j]), D_{i-1,j} + d(s1[i],"-") \text{ or } D_{i,j-1} + d("-",s2[j])$$

because an alignment of $s1_i$ and $s2_j$ can only have one of three possible last columns: corresponding to a matching, $s1[i]$ versus "-" or "-" versus $s2[j]$. Could $D_{i,j}$ be any of these quantities? No, it can only be the smallest one. Assume it wasn't, then the partial alignment of $s1_i$ and $s2_j$ would be sub optimal.

3. An optimal alignment can be reconstructed by backtracking: Assume, that an optimal path up through the matrix passed the node (i,j) . Then the edge (column in alignment) leading to (i,j) would come from one of the three predecessors, that had given rise to the smallest among the three possibilities above. This observation allows a "crawling backwards" down from (l_1, l_2) towards $(0,0)$, reconstructing an optimal alignment column by column.

Because newly collected sequences are becoming longer and longer, the computational properties of such an algorithm is important. As formulated above, finding the distance between $s1$ and $s2$ would need the filling out of a matrix with $(l_1+1)*(l_2+1)$ entries. Each entry would be obtained by taking minimum over three quantities, i.e. the computation time would be proportional to l_1*l_2 (a quadratic function of sequence length). The backtracking would take time proportional to the length of the alignment, i.e. much faster ($< (l_1 + l_2)$ - a linear function of sequence lengths).

If only the distance between $s1$ and $s2$ is needed only two layers in the matrix (linear in sequence length) is necessary, but that would preclude backtracking beyond the last element in the total alignment. Thus, if the alignment is needed, then the whole matrix is needed as formulated above, which demands memory growing quadratic in sequence length.

Plan:

- Read Jotun Hein's lecture notes on optimization alignment. Read Python programming in PYTHON
- Make pairwise alignment program. Simulate pairs of pairs of sequences of length L nucleotides ($L=10, 100, 1000$).
- Finish program and report

Comment. i. This project is designed for two high school students in 6 weeks. Ii. Skills obtained through this project: a. increased programming proficiency. b. understanding of basic probabilistic models of DNA evolution. c. Introduction to an important field within modern biosciences: comparative genomics. Iii. Students are always encourage to write a report continuously throughout the project.

References

- D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, 60:351-360, 1987.
- Fitch, W.(1971) "Towards defining the course of evolution: minimum change for a specific tree topology" *Systematic Zoology* 20:406-416.
- Gotoh, O. (1982). "An improved algorithm for matching biological sequences." *J. Mol. Biol.* 162: 705-708.
- Dan Gusfield. Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, New York, NY, USA, 1997
- Richard W. Hamming. Error Detecting and Error Correcting Codes, *Bell System Technical Journal* 26(2):147-160, 1950.
- Hartigan,JA (1973) "Minimum mutation fit to a given tree" *Biometrics* 29:53-69.
- D. S. Hirschberg, A linear space algorithm for computing maximal common subsequences, *Communications of the ACM*, v.18 n.6, p.341-343, June 1975
- V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Phys. Dokl.*, 10:707-710, 1966.E. Myers, "An O(ND) Difference Algorithm and Its Variations," *Algorithmica* 1, 2 (1986), 251-266.
- Needleman, S. B. and C. D. Wunsch (1970). "A general method applicable to the search for similarities in the amino acid sequences of two proteins." *J. Mol. Biol.* 48: 443-453.
- Sankoff, D. (1975) "Minimal mutation trees for sequences" *SIAM journal on Applied Mathematics* 78:35-42.
- Sankoff, D. and Kruskal, J. (1983) "Time Warps, String Edits & Macromolecules" Addison-Wesley
- P. Sellers. An algorithm for the distance between two sequences. *J. Comb. Theory*, 16:253-258, 1974.
- Smith, T. F., M. S. Waterman, and W Fitch (1981). "Comparative Biosequence Metrics." *J. Mol. Evol.* 18: 38-46.
- E. Ukkonen: Algorithms for approximate string matching. *Information and Control* 64 (1985), 100-118.
- Vinthsuik
- Robert A. Wagner , Michael J. Fischer, The String-to-String Correction Problem, *Journal of the ACM (JACM)*, v.21 n.1, p.168-173, Jan. 1974

HISTORY OF PROBLEM:

1953: Richard Bellman invents Dynamical Programming
1966: Levenstein formulates distance measure between sequences and introduces dynamica programming algorithm finding the distance.
1970: Needleman and Wunch compares proteins maximising a similarity score.
1972: Sankoff & Sellers reinvents the basic algorithm.
1972: Sankoff can align subject to the constraint that there must be exactly k indels.
1973: Sankoff makes multiple alignment and phylogeny - both exact & heuristic.
1975: Hirschberg gives linear memory algorithm.
1976: Waterman gives cubic algorithm allowing for indels of arbitrary length without reference to phylogeny.
1981: Waterman, Smith and Fitch shows duality of similarity and distance.
1981: Smith and Waterman invents similarity based local alignment.
1982: Gotoh gives quadratic algorithm if gap penalty functionen is $g_k = a + b*k$ (for indel of length k). Uses 3 matrices in stead of 1.
1983: Waterman and Byers introduces close-to-optimal alignments.
1984-5: Ukkonen, Myers, Fickett accelerates the algorithm considerably.
1984: Hogeveg and Hespers introduces heuristic multiple phylogenetic alignment.
1984: Fredman introduces triple alignment generalisation of Needleman-Wunch algorithms.
1985: Lipman & Wilbur uses hashing.
1987: Feng-Doolittle introducesphylogenetisk alignment: "Once a gap always a gap".
1989: Kececioglu makes strong acceleration of Sankoff's exact algorithm.
1989: Myers introduces alignment with concave gap penalty function.
1991: Thorne, Kishino & Felsenstein makes good model for statistical alignment, partially introduced in 1986 by Thomson & Bishop.
1991: States & Botstein compares a DNA string with a protein in search of frameshift mutations.
1993-4: Gusfield, Lander, Waterman and others introduces parametric alignment.
1994: Krogh et al & Baldi et al. introduces Hidden Markov Models for multiple alignment.
1995: Mitchison & Durbin introduces Tree-HMMs
1999 - Resurgence of interest in statistical alignment

Extensions

Longer Indels.

A simple biological observation is that indels can have lengths much longer than one. Should the weight of an indel of length 1100 really be regarded as consisting of 1100 independent indels of length 1, as is assumed in the Levenstein approach? It would be more realistic to let it have a smaller cost. Let g_k be the cost of deleting or inserting k consecutive nucleotides. How could the optimal alignment now be found? Very simply - node (i,j) now has more predecessors. The single predecessor $(i,j-1)$ is substituted by the set of $(i,j-1), (i,j-2), \dots, (i,j-k), \dots, (i,0)$. Similarly for $(i-1,j)$. The single quantity $D_{i,j-1} + d(-,s2[j])$ is now substituted by the set of quantities $D_{i,j-k} + g_k$ ($k=1, \dots, j$). This leads to an increase in computational time from quadratic to cubic, which is a serious slowdown for longer sequences.

Gotoh's algorithm.

Gotoh proposed a very significant speedup for the "longer indel" algorithm if g_k was of the form $g_k = a + b*k$, where a and b are positive parameters. g_k now has a cost of opening an indel (a) and a term for elongating it by one nucleotide (b).

The trick he proposed was very simple. Split the distance $D_{i,j}$ into three types $D0_{i,j}, D1_{i,j}$ and $D2_{i,j}$

$D0_{i,j}$ is the distance between $s1i$ and $s2j$, but subject to the constraint, that the last elements in the sequences were matched. $D1_{i,j}$ is the distance between $s1i$ and $s2j$, but subject to the constraint, that the last element in $s1i$ was inserted relative to $s2j$. $D2_{i,j}$ is the distance between $s1i$ and $s2j$, but subject to the constraint, that the last element in $s2j$ was inserted relative to $s1i$. This allows a new recursion in these distances. The conditioning on the type of the last position in the alignment allows the penalisation of an initiation of an indel relative to a continuation. Is also assumed that weights satisfy (c and d are elements in the sequences):

c- is always more expensive than c
-d d

which will be the case, if $\max(d, c) < 2b$. Didn't the satisfy the constraint, the recursive definition below of $D1_{i,j}$ would also refer to $D2_{i-1,j} + a + b$ and thus be the minimum over three quantities, not. Similarly for $D2_{i,j}$.

The algorithm is now the following:

Initialisation $D0_{0,0} = 0, D1_{0,0} = D2_{0,0} = \text{infinity}$.

Recursion

$$D0_{i,j} = d(s1[i],s2[j]) + \min\{D0_{i-1,j-1}, D1_{i-1,j-1}, D2_{i-1,j-1}\}$$

$$D1_{i,j} = \min\{D0_{i-1,j} + a + b, D1_{i-1,j} + b\}$$

$$D2_{i,j} = \min\{D0_{i-1,j} + a + b, D2_{i,j-1} + b\}$$

This algorithm needed 3 matrices (one for each type of distance), but has a quadratic runtime.

*

Similarity Alignments.

It is sometimes advantageous to formulate sequence alignment not as a distance minimisation problem but as a similarity maximisation problem. A few reasons are:

- The parameter space for similarity is larger as similarity can be both negative and positive, but there is little meaning in discussing negative distance.
- Intuitively, a biologist search will for regions with high similarity and not avoid regions of low similarity, so it conforms well with what a biologist feels he/she is doing.
- Historically, the first alignment method in molecular biology formulated by Needleman and Wunch (1970) was a similarity method.

The algorithm is extremely similar: $w(a,b)$ is the similarity of matching a with b. g is the gap penalty.

Initial condition: $S_{0,0} = 0. (S_{i,j} = S(s1_i, s2_j))$

$$S_{i,j} = \max\{S_{i-1,j-1} + w(s1[i],s2[j]), S_{i,j-1} - g, S_{i-1,j} - g\}$$

Similarity-Distance.

Given the resemblance of the distance and similarity approach it is natural to ask if one doesn't convert one into the other by multiplying the parameters with minus and switching a "min" for a "max". The answer is almost. Let $d(,)$ be the distance function on the elements, $w(,)$ the similarity function, g_k the cost of deleting k elements and h_k the gap penalty.

Then $w(,)=c-d(,)$ and $h_k = g_k - kc/2$. In the example for distances above would give similarity parameters

$$c = 10$$

$$\begin{aligned} w(\text{identity}) &= c && 10 \\ w(\text{transition}) &= c - 2 && 8 \\ w(\text{transversion}) &= c - 5 && 5 \\ h_k &= 10*k - kc/2. (g = 10 - c/2) && 5 \end{aligned}$$

This 1-1 correspondence is not surprising. If we looked at the parameter switch in going from similarity to distance, it looked as if S_{ij} was reflected in $S_{ij} = 0$ and then sunk/lifted with c . Obviously, the gap penalty is only sunk/lifted with $c/2$.

Weighting of Events.

The extension from shorter to longer indels was motivated by knowledge of frequency of evolutionary events. Which weights to use is a major problem. In the distance approach, it is an intuitive rule that the more frequent an event the lower should the cost be of postulating it in the alignment, but more precisely what should the relationship be is subject to much debate. M.O.Dayhoff tabulated the frequency of point mutations in closely related proteins and suggested scoring schemes. Different scoring schemes will be discussed later in connection with database searching.