I519 Introduction to Bioinformatics, Fall 2011

Pairwise alignment of DNA/protein sequences

Yuzhen Ye (yye@indiana.edu)
School of Informatics & Computing, IUB

We compare biological molecules, not any two strings!

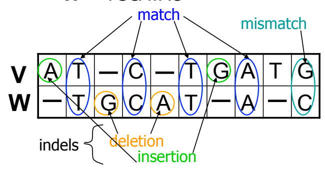
- Sequence alignment reveals function, structure, and evolutionary information!
- From edit distance to distance between two biological molecules with biological meaning—the scoring matrix
- Local alignment versus global alignment
- But still the dynamic programming algorithm is the algorithm behind pairwise alignment of biological sequences

Aligning DNA Sequences: scoring matrix

V = ATCTGATG   n = 8

W = TGCATAC   m = 7

4	matches
1	mismatches
2	insertions
2	deletions



Simple scoring

- When mismatches are penalized by $-\mu$, indels are penalized by $-\sigma$,

and matches are rewarded with $+1$,

the resulting score is:

#matches $- \mu$*(#mismatches)* $- \sigma$ *(#indels)*

Scoring matrices

To generalize scoring, consider a (4+1) x(4+1) **scoring matrix** δ.

In the case of an amino acid sequence alignment, the scoring matrix would be a (20+1)x(20+1) size. The addition of 1 is to include the score for comparison of a gap character "-".

This will simplify the algorithm as follows:

$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

Scoring matrices for DNA sequence alignment

- A simple positive score for matches and a negative for mismatches and gaps are most often used.
- Transversions penalized more than transitions
  - transitions: replacement of a purine base with another purine or replacement of a pyrimidine with another pyrimidine (A <-> G, C <-> T)
  - transversions: replacement of a purine with a pyrimidine or vice versa.
  - Transition mutations are more common than transversions

Making a scoring matrix for protein sequence alignment

- Scoring matrices are created based on biological evidence.
- Alignments can be thought of as two sequences that differ due to mutations.
- Some of these mutations have little effect on the protein's function, therefore some penalties, $\delta(v_i, w_j)$, will be less harsh than others.
- We need to know how often one amino acid is substituted for another in related proteins

Scoring matrix: example

	A	R	N	K
A	5	-2	-1	-1
R	-	7	-1	3
N	-	-	7	0
K	-	-	-	6

AKRANR

KAAANK

-1 + (-1) + (-2) + 5 + 7 + 3 = 11

- Notice that although R and K are different amino acids, they have a positive score.
- Why? They are both positively charged amino acids→ will not greatly change function of protein.

Conservation

- Amino acid changes that tend to preserve the physico-chemical properties of the original residue
  - Polar to polar
    - aspartate → glutamate
  - Nonpolar to nonpolar
    - alanine → valine
  - Similarly behaving residues
    - leucine to isoleucine

Common scoring matrices for protein sequence alignment

- Amino acid substitution matrices
  - PAM
  - BLOSUM
- Try to compare protein coding regions at amino acid level
  - DNA is less conserved than protein sequences (codon degeneracy; synonymous mutations)
  - Less effective to compare coding regions at nucleotide level

Reading: Chapter 4, 4.3

PAM

- Point Accepted Mutation (Dayhoff et al.)
- 1 PAM = PAM1 = 1% average change of all amino acid positions
  - After 100 PAMs of evolution, not every residue will have changed
    - some residues may have mutated several times
    - some residues may have returned to their original state
    - some residues may not changed at all

PAM1 & PAM250

Substitution	PAM1	PAM250
Phe to Ala	0.0002	0.04
Phe to Arg	0.0001	0.01
Phe to Asp		
..		
Phe to Phe	0.9946	0.32
…		
Sum	1	1

Normalized probability scores for changing Phe to other amino acids at PAM1 and PAM250 evolutionary distances

Chapter 3, table 3.2

Log-odds substitution matrices

- Using amino acid changes that were observed in closely related proteins; they represented amino acid substitutions that don't significantly change the structure and function of the protein.
  - "accepted mutations" or "accepted" by natural selection.
- Log-odds of the probability of matching a pair of amino acids in this database relative to a random one
  - Ref: Amino acid substitution matrices from protein blocks (PNAS. 1992, 89(22): 10915–10919)

Log odd scores

Define f_{ij} as the frequency of observing amino acid pair *i, j*. Then the observed probability of occurrence for each *i, j* pair is

$$q_{ij} = f_{ij} / \sum_{i=1}^{20} \sum_{j=1}^{i} f_{ij}$$

The probability of occurrence of the ith amino acid in an i, j pair is,

$$p_i = q_{ii} + \sum_{j \neq i} q_{ij}/2,$$

The *expected* probability of occurrence e_{ij} for each i, j pair is computed as,

$$\begin{aligned} e_{ij} &= p_i p_j \quad if \quad i = j \\ &= 2 p_i p_j \quad else \end{aligned}$$

A lod (logarithm of odd) is then calculated in bit units as, $s_{ij} = log_2(q_{ij}/e_{ij})$

The amino acid substitution is considered as a Markov model

- A Markov model is characterized by a series of changes of state in a system such that a change from one state to another does not depend on the previous history of the state
- Use of the Markov model makes it possible to extrapolate amino acid substitutions observed over a relatively short period of evolutionary time to longer periods of evolutionary time
  - PAMx = PAM1^x
  - The multiplication of two PAM1 matrices -> PAM2

PAM250

- PAM250 is a widely used scoring matrix:
- PAM250 = PAM1^{250}

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	...
	A	R	N	D	C	Q	E	G	H	I	L	K	...
Ala A	13	6	9	9	5	8	9	12	6	8	6	7	...
Arg R	3	17	4	3	2	5	3	2	6	3	2	9	
Asn N	4	4	6	7	2	5	6	4	6	3	2	5	
Asp D	5	4	8	11	1	7	10	5	6	3	2	5	
Cys C	2	1	1	1	52	1	1	2	2	2	1	1	
Gln Q	3	5	5	6	1	10	7	3	7	2	3	5	
...													
Trp W	0	2	0	0	0	0	0	0	1	0	1	0	
Tyr Y	1	1	2	1	3	1	1	1	3	2	2	1	
Val V	7	4	4	4	4	4	4	4	5	4	15	10	

BLOSUM

- Blocks Substitution Matrix
- Scores derived from observations of the frequencies of substitutions in **blocks of local alignments in related proteins**
- Matrix name indicates evolutionary distance
  - BLOSUM62 was created using sequences sharing no more than 62% identity

BLOSUM versus PAM

- The PAM family
  - PAM matrices are based on ***global*** alignments of closely related proteins.
  - The PAM1 is the matrix calculated from comparisons of sequences with no more than 1% divergence; Other PAM matrices are ***extrapolated*** from PAM1.
- The BLOSUM family
  - BLOSUM matrices are based on ***local*** alignments (blocks)
  - All BLOSUM matrices are based on ***observed*** alignments (BLOSUM 62 is a matrix calculated from comparisons of sequences with no less than 62% divergence)
- Higher numbers in the PAM matrix naming scheme denote larger evolutionary distance; BLOSUM is the opposite.
  - For alignment of distant proteins, you use PAM150 instead of PAM100, or BLOSUM50 instead of BLOSUM62.

Local vs. global alignment

- Global Alignment

```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

- Local Alignment—better alignment to find conserved segment

```
                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Local vs. global alignment

- The <u>Global Alignment Problem</u> tries to find the longest path between vertices *(0,0)* and (*n,m*) in the edit/alignment graph.
  - The Needleman–Wunsch algorithm
- The <u>Local Alignment Problem</u> tries to find the longest path among paths between **arbitrary vertices** (*i,j*) and (*i'*, *j'*) in the edit graph.
  - The Smith–Waterman algorithm

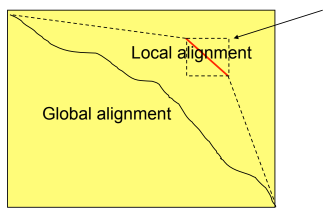
Local alignments: why?

- Two genes in different species may be similar over short conserved regions and dissimilar over remaining regions.
- Example:
  - Homeobox genes have a short region called the homeodomain that is highly conserved between species.
  - A global alignment would not find the homeodomain because it would try to align the ENTIRE sequence

The local alignment problem

- <u>Goal</u>: Find the best local alignment between two strings
- <u>Input</u> : Strings **v, w** and scoring matrix δ
- <u>Output</u> : Alignment of substrings of **v** and **w** whose alignment score is maximum among all possible alignment of all possible substrings

Local alignment: example



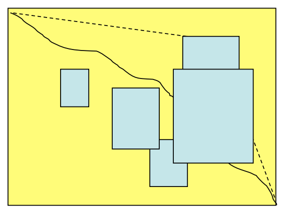
Compute a "mini" Global Alignment to get Local

Local alignment: running time

- Long run time $O(n^6)$:

  - In the grid of size *n x n* there are ~n^2 vertices *(i,j)* that may serve as a source and ~n^2 vertices *(i',j')* that may serve as a sink.

  - For each such vertices computing alignments from *(i,j)* to *(i',j')* takes $O(n^2)$ time.



We do NOT go with this algorithm!

The local alignment recurrence

- The largest value of $s_{i,j}$ over the whole edit graph is the score of the best local alignment.
- The recurrence:

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

Notice there is only this change from the original recurrence of a Global Alignment

The local alignment recurrence

- The largest value of $s_{i,j}$ over the whole edit graph is the score of the best local alignment.
- The recurrence:

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j-1} + \delta(v_i, w_j) \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \end{cases}$$

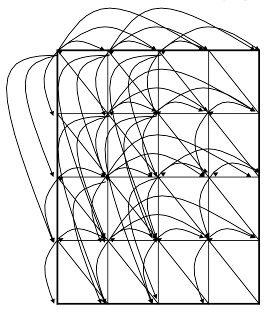
Power of ZERO: there is only this change from the original recurrence of a Global Alignment - since there is only one "free ride" edge entering into every vertex

- Complexity: *$O(N^2)$*, or *$O(MN)$*
- *Initialization will be different*

Scoring indels: naive approach

- A fixed penalty σ is given to every indel:
  - -σ for 1 indel,
  - -2σ for 2 consecutive indels
  - -3σ for 3 consecutive indels, etc.
- Can be too severe penalty for a series of 100 consecutive indels

Arbitrary gap penalty?



There are many such edges!

Adding them to the graph increases the running time of the alignment algorithm by a factor of ***n*** (where ***n*** is the number of vertices)

So the complexity increases from O(n^2) to O(n^3)

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:

ATA__GC
ATATTGC

This is more likely.

ATAG_GC
AT_GTGC

This is less likely.

Normal scoring would give the same score for both alignments

Affine gap penalties

- Score for a gap of length *x* is:
  $-(\rho + \sigma x)$
  where $\rho > 0$ is the penalty for introducing a gap:
  gap opening penalty
  ρ will be large relative to σ:
  gap extension penalty
  because you do not want to add too much of a penalty for extending the gap.
- Reduced penalties (as compared to naïve scoring) are given to runs of horizontal and vertical edges

Affine gap penalty recurrences

$$gap(L) = \sigma + \rho * L$$

$$D(i,j) = max \begin{cases} D(i-1,j) + \sigma \\ S(i-1,j) + (\sigma + \rho) \end{cases}$$

Continue gap in *y* (deletion)
Start gap in *y* (deletion)

$$I(i,j) = max \begin{cases} I(i,j-1) + \sigma \\ S(i,j-1) + (\sigma + \rho) \end{cases}$$

Continue gap in *x* (insertion)
Start gap in *x* (insertion)

$$S(i,j) = max \begin{cases} S(i-1,j-1) + \delta(x_i, y_j) \\ I(i,j) \\ D(i,j) \end{cases}$$

Match or mismatch
End with deletion
End with insertion

Readings

- Chapter 4, 4.1
  - "Alignment can reveal homology between sequences" (similarity vs homology)
  - "It is easier to detect homology when comparing protein sequences than when comparing nucleic acid sequences"
- Primer article: What is dynamic programming by Eddy

We will continue on

- Significance of an alignment (score)
  - Homologous or not?
- Faster alignment tools for database search