## Week 3: Sequence alignment heuristics*

- What is a heuristic method?
- Sequence comparison and database search
- Why can't we use dynamic Programming for large databases of sequences?
- The heuristic methods: FASTA, BLAST
- PAM units and PAM matrices
- The BLOSUM score matrix
- Sensitivity and selectivity
- Z-value. E-value.

* Material for week 3 is in http://www.math.tau.ac.il/~rshamir/algmb/01/algmb01.html and in http://www.cs.tau.ac.il/~rshamir/algmb/98/algmb98.html (thanks to Ron Shamir)
Additional reading: the textbook mentioned in the course syllabus



## Heuristics

•A heuristic method is an algorithm that gives only approximate solution to a given problem.

•Sometimes we are not able to formally prove that this solution actually solves the problem, but heuristic methods are commonly used because they are much faster than exact algorithms.

•This week we will talk on some of the most commonly used heuristics for sequence comparison.



## Motivation

The problem we tackle this week is searching for a query string in a large database of strings. We want to find (i) a similar string, (ii) similar domains in the database.

Databases for proetin sequences and gemone (base) sequences are pretty long. The data they contains is of magnitude $10^{6-9}$. If our query string is of length around 300 then we cannot use the DP. Recall that the DP ran in time O(m*n) for sequnce S and T of lengths m and n respectively.

This size (~$10^{10}$) runtime for one query protein is too large.

By the way, is the DP formally solving the sequence alignment problem?



## What are our options?

• Implement DP in hardware, as is done for graphics applications, speeding their computations by factor of million and more. – Costly and not availabale to most researchers.

•Use parallel hardware. It is easy to distribute the DP runs on thousands of processors and integrates the good results later. – As before, this is pretty costly.

•Use different heuristics that allow very fast rejection of non-similar sequences in the database.



## The general idea

Preprocess the database once and for all, and prepare it for fast alignment queries.

The methods we show are based on the following observations:

1. Numerous queries are compared with a database which is not updated often. Preprocessing can take a very long time, but can be done offline and on many machines, and while it is updated the previous version is available.
2. Substitutions are much more likely than indels.
3. We expect homologous sequences to contain many small segments with matches (or substitutions) and without indels. These segments will be used as starting points for the computations.



## FASTA(1985,1987)

FASTA compares a query sequence against each sequence in the DB.

Illustration of the method

1. Given two sequences, we draw a table like in DP, and put * where two letters are identical in both sequences (*hot spots*).
2. Run along the diagonals, giving each hot spot a positive score and the substitutions between them (interspots) a negative score. Connect-the-dots for finding identical regions. Find 10 best *diagonal runs.*
3. A diagonal run specifies an alignment with matches (hotspots) and substitutions (interspots). Re-evaluate the runs using the substitution matrix. Discard all low scoring runs.
4. Concatenate some such regions, not necessarily all on the same diagonal, allowing spaces.

Step 1: Sequence Seq1 in the DB (index by j)

| | j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| i | | A | A | G | T | C | C | C | G | T | G |
| 1 | A | * | * | | | | | | | | |
| 2 | G | | | * | | | | | * | | * |
| 3 | G | | | * | | | | | * | | * |
| 4 | T | | | | * | | | | | * | |
| 5 | C | | | | | * | * | * | | | |
| 6 | C | | | | | * | * | * | | | |
| 7 | G | | | * | | | | | * | | * |
| 8 | T | | | | * | | | | | * | |
| 9 | T | | | | * | | | | | * | |
| 10 | C | | | | | * | * | * | | | |

'GT' starts at j=3,9 in P1

The query sequence Q (index i)

'GT' starts at i=3,7 in Q



Steps 2-3: A diagonal run -matches and substitutions on the same diagonal

| | j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| i | | A | A | G | T | C | C | C | G | T | G |
| 1 | A | * | * | | | | | | | | |
| 2 | G | | | * | | | | | * | | * |
| 3 | G | | | * | | | | | * | | * |
| 4 | T | | | | * | | | | | * | |
| 5 | C | | | | | * | * | * | | | |
| 6 | C | | | | | * | * | * | | | |
| 7 | G | | | * | | | | | * | | * |
| 8 | T | | | | * | | | | | * | |
| 9 | T | | | | * | | | | | * | |
| 10 | C | | | | | * | * | * | | | |



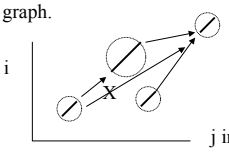
Step 4 – the alignment graph

Construct a weighted graph whose nodes are the diagonal runs, and there a directed edge from node v to node u if the row and column indices of u and v do not intersect.

The weights of the nodes are their respective scores from the previous step. An edge gets a negative score depending on number of spaces which would be created by the alignment u,spaces,v.

We can easily discard long edges.

This is a *directed a-cyclic* graph. Find a *maximum weight path* on this graph.




-

∅ - a node

X - Too many spaces



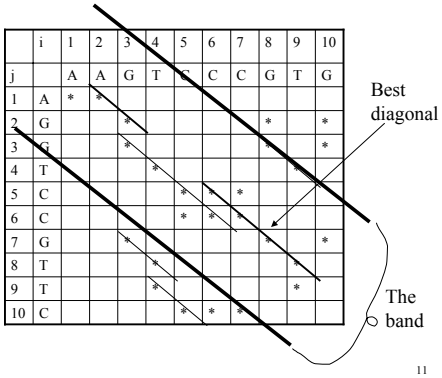
An alternative alignment by FASTA (step 4)

Take the best diagonal run and take a narrow *band* about it in the table. Fill the DP matrix just in this band, finding the best local alignment *within* this band.

The above refinements of the matching are based on the assumption that the good diagonal run with which we started is part of the local optimum and other pieces cannot be too far from it.



The band about a best scoring diagonal (alternative step 4)






FASTA- how is it done in a time saving manner?

Preprocess the database

Take k-tuples of letters-usually 2 for proteins and 6 for DNA bases

Build a lookup table for k-tuples (here k=6)

For each k-tuple keep a list of all indices in the DB where it starts

E.g,

| 6-tuple | Seq. 1 | Seq. 2 | Seq. 3 |
|---|---|---|---|
| AAAACT | J=3,21,67 | J=1,7,23,38 | |
| AAACTG | J=4,22,68,103 | J=2,8,24,39,80 | J=55 |
| GGGTGT | J=49 | | J=1,12,33 |
| ….. | | | |

We have $4^6 = 4096$ 6-tuples

The lookup table for Seq1 in the database

| | Seq. 1 |
|---|---|
| AAAACT | j=3,21,67 |
| AAACTG | j=4,22,68,103 |
| GGGTGT | j=10 |

Our query sequence Q: GGGAAAAACTGGGGTGTCC

AAAACT appears in position i=5 in Q

AAACTG appears in position i=6 in Q

GGGTGT appears in position i=12 in Q

j-i determines the number of diagonal where this match resides



Each diagonal is pointed to by a unique value of j-i

| i \ j | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | j-i=0 | 1 | 2 | 3 |
| 2 | -1 | j-i=0 | 1 | 2 |
| 3 | -2 | -1 | j-i=0 | 1 |
| 4 | -3 | -2 | -1 | j-i=0 |

To find how many k-tuples hit the same diagonal create a vector with indices j-i whose entries count the number of times we got j-i.



AAAACT, i=5, j= 3,21,67

AAACTG, i=6, j= 4,22,68,103

GGGTGT, i=12, j=10

| | C | C | A | A | A | A | C | T | G | G | G | T | G | T | A | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G | | | | | | | | | | | | | | | | |
| G | | | | | | | | | | | | | | | | |
| G | | | | | | | | | | | | | | | | |
| A | | | | | | | | | | | | | | | | |
| A | | | * | | | | | | | | | | | | | |
| A | | | | * | | | | | | | | | | | | |
| A | | | | | * | | | | | | | | | | | |
| A | | | | | | * | | | | | | | | | | |
| C | | | | | | | * | | | | | | | | | |
| T | | | | | | | | * | | | | | | | | |
| G | | | | | | | | | * | | | | | | | |
| G | | | | | | | | | | * | | | | | | |
| G | | | | | | | | | | | * | | | | | |
| T | | | | | | | | | | | | * | | | | |
| G | | | | | | | | | | | | | * | | | |
| T | | | | | | | | | | | | | | * | | |

| j-i diag # | * in diagonal |
|---|---|
| -50 | |
| -49 | |
| ….. | |
| -2 | +3 |
| …. | |
| 16 | +2 |
| …. | |
| 62 | +2 |
| …. | |
| 97 | +1 |
| | |



In summary,

What we saw today seems to consume much computer time.

Actually, the steps we have seen perform a series of fast rejections of candidate solutions, might eliminate a whole sequence from being considered altogether, and prune hopeless matches.

The next method, BLAST, is faster.



## How to evaluate results of this software?

The outputs we get depend on cutoff parameters, and other parameters like k in the k-tuple, which are controlled by the user. The measurement tool is to run a known sequence with a known set of answers and pick the parameters that yield best results.

Sensitivity and selectivity –We need to check, does the set of solutions contain all the true solutions, or does it contain many false positives, and tweak the parameters accordingly.

• Sensitivity: the ability to detect 'true positive' matches. The most sensitive search finds all true matches but might have many false positives.

• Selectivity: the ability to reject false positive matches. The most specific search will return only true matches, but might have many 'false negatives'.



## How to evaluate results of this software? (cont.)

Z-score. A score by itself is not too meaningful. What does a score=40 mean? Is it good or bad?

The way to determine this is to take the two sequences that are compared, shuffle one of them and run the comparison. Then compute many random shuffles and compute the *mean* and the *standard deviation* of all these scores.

$$\text{Z-score} = \frac{\text{my\_score} - \textit{mean}}{\textit{standard deviation}}$$

How standard deviations is my score from the mean of matching two random sequences. The higher Z-score is the more significant is the result.

BLAST – Basic Local Alignment Search Tool

• Motivation: Increase the speed of FASTA by finding better and fewer hot-spots

• The idea: Screen DB better by integrating the substitution matrix in the first stage

• BLAST was developed for protein sequence comparison in 1990 while FASTA was developed for DNA bases (and used for proteins)



To be continued.