

Different alignment algorithms of nucleotide sequences

Alexandre Tchourbanov *

PKI 357, UNOmaha, 1110 South 67th St.

Omaha, NE 68182-0694, USA

Phone: (402)554-6004

E-mail: tchourba@cse.unl.edu

June 14, 2002

Abstract

In this paper we try to compare different types of alignments for the purposes of exon/intron finding. Local and global alignments differences are addressed as well as HMM models recently developed for these purposes. A small discussion describes differences between scoring matrices.

1 Introduction

Suppose we have two strings $x = \text{acgtggattagcgta}$ and $y = \text{acgtcgt}$. String x has the size n and y has the size m .

There are several alignment problems

- Global alignment(Needleman–Wunch): of all x must be aligned with all of y
 - With linear gap penalty
acgtggattagcgta
ac---g--t--cgta
 - The same alignment with affine gap penalty
acgtggattagcgta
acgt-----cgta
- Local alignment(Smith–Waterman): a subsequence of x must be aligned with subsequence of y
acgtgg
acgtcg
- Repeated matches: all of x must be aligned with some (possibly repeated) subsequences of y
acgtggattagcgta
acgtcg.....cgta

*I would like to thank professors Hesham Ali and Jitender Deogun for the opportunity to work on this project

- Overlap matches: a prefix or suffix of x must be aligned with a prefix or suffix of y

```
acgt-gga
acgtcgta
```

We introduced several gap penalty functions. Here we use an idea that a gap of length k is more probable than k gaps of length 1 since

- A gap may be due to 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 function involving two terms

- A penalty d associated with opening a gap
- A smaller penalty e for extending the gap

Accordingly, there gap function is either linear

$$w(k) = ek \tag{1}$$

or affine

$$w(k) = \begin{cases} d + e(k - 1) & : k \geq 1 \\ 0 & : k = 0 \end{cases} \tag{2}$$

2 Needleman–Wunsch algorithm (global alignment)

The Needleman–Wunsch algorithm is similar to the Smith-Waterman algorithm, but sequence comparisons are global, not local. Global comparisons force an alignment of the entire query sequence against the entire database sequence.

While local alignments always begin and end with a match, global alignments may begin or end with an insertion or deletion (**indel**). For a given query sequence and database sequence, a global score will be less than or equal to a local score due to **indel**'s on the ends.

We build the table F in which

$$F(i, j) = \text{the maximum score for an alignment between } x_{1..i} \text{ and } y_{1..j}$$

We will fill the table F from left to right and top to bottom. This filling in the table is called *dynamic programming*. By definition $F(m, n)$ is the maximal score for a global alignment between x and y .

The following observations take place

- Any prefix of the optimal alignment between x and y is an optimal alignment between a prefix $x_{1..i}$ and a prefix $y_{1..j}$ of y
- The value in the score matrix $F(i, j)$ depends only on the values $F(i - 1, j - 1)$, $F(i - 1, j)$, and $F(i, j - 1)$. This is because an optimal alignment between $x_{1..i}$ and $y_{1..j}$ consists of either

- An optimal alignment between $x_{1...(i-1)}$ and $y_{1...(j-1)}$ extended with a match between x_i and y_j
- An optimal alignment between $x_{1...(i-1)}$ and $y_{1...j}$ extended with a match between x_i and a gap
- An optimal alignment between $x_{1...i}$ and $y_{1...(j-1)}$ extended with a match between a gap and y_j

So an alignment can be computed by scanning x and y from left to right, recording only the optimal alignments between prefixes of x and y , and forgetting all non-optimal ones.

The table F gives us the maximal score. In order to find the traceback, when filling in $F(i, j)$, we record the traceback from (i, j) . The traceback points to the cell that led to the maximal score: $(i - 1, j - 1)$ or $(i - 1, j)$ or $(i, j - 1)$. when we finished, we find an optimal alignment just by following the traceback from (n, m) to $(0, 0)$.

The following recurrence is used for the matrix F of size $n \times m$

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

where

d - penalty for the gap introduction

$F(i, j)$ - score of the best alignment between the initial segments $x_{1...i}$ and $y_{1...j}$

		A	C	G	T	G	G	A	T	T	A	G	C	G	T	A	
		0	← -8	← -16	← -24	← -32	← -40	← -48	← -56	← -64	← -72	← -80	← -88	← -96	← -104	← -112	← -120
A		↑ -8	↖ 5	← -3	← -11	← -19	← -27	← -35	↖ -43	← -51	← -59	↖ -67	← -75	← -83	← -91	← -99	↖ -107
C		↑ -16	↑ -3	↖ 13	← -5	← -3	← -11	← -19	← -27	← -35	← -43	← -51	← -59	↖ -67	← -75	← -83	← -91
G		↑ -24	↑ -11	↑ 5	↖ 20	← 12	4	↖ -4	← -12	← -20	← -28	← -36	↖ -44	← -52	← -60	← -68	← -76
T		↑ -32	↑ -19	↑ -3	↑ 12	↖ 27	← 19	← 11	← 3	↖ -5	← -13	← -21	← -29	← -37	← -45	↖ -53	← -61
C		↑ -40	↑ -27	↑ -11	↑ 4	↑ 19	25	↖ 17	9	5	↖ -3	← -11	← -19	↖ -21	← -29	← -37	← -45
G		↑ -48	↑ -35	↑ -19	↑ -4	↑ 11	26	↖ 32	24	← 16	← 8	← 0	↖ -4	← -12	↖ -14	← -22	← -30
T		↑ -56	↑ -43	↑ -27	↑ -12	↑ 3	18	↖ 25	31	↖ 31	23	← 15	← 7	← -1	← -9	↖ -7	← -15
A		↑ -64	↑ -51	↑ -35	↑ -20	↑ -5	10	↑ 17	30	↖ 30	30	28	← 20	← 12	← 4	← -4	↖ -2

Figure 1: Global alignment F matrix for nucleotides sequences (traceback is shown bold)

3 Smith–Waterman algorithm (local alignment)

This algorithm, developed by T.F. Smith and M.S. Waterman [7], was based on the original homology algorithm of Needleman – Wunsch.

A much more common situation is where we are looking for the best alignment between the subsequences of x and y . This arises for example when it is suspected that two protein

sequences may share a common domain, or when comparing extended sections of genomic DNA sequence.

The alignment is obtained by determining what transformations the query sequence would need to undergo to match the database sequence. Transformations include substituting one character for another and inserting or deleting a string of characters. A score is assigned for each character-to-character comparison—positive scores for exact matches and some substitutions (according to substitution matrix), negative scores for other substitutions and insertions/deletions. The following recurrence is used for the matrix F of size $n \times m$.

$$F(i, j) = \max \begin{cases} 0, \\ F(i-1, j-1) + s(x_i, y_j), \\ F(i-1, j) - d, \\ F(i, j-1) - d \end{cases}$$

Here score $F(i, j)$ never goes below 0, allowing us to start a local alignment from any place. Analogously, if we want to trace the local alignment in the F matrix, we start with the maximum in the matrix and trace it back until we reach 0 – the starting point of our alignment.

	A	C	G	T	G	G	A	T	T	A	G	C	G	T	A
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	5	0	0	0	0	0	5	0	0	5	0	0	0	0
G	0	0	13	← 5	↖ 2	0	0	0	↖ 7	↖ 2	0	↖ 3	↖ 8	0	↖ 2
T	0	0	↑ 5	20	← 12	↖ 9	↖ 7	0	0	↖ 6	0	↖ 7	↖ 1	↖ 15	← 7
C	0	0	↖ 2	↑ 12	27	← 19	← 11	↖ 6	↖ 7	↖ 7	↖ 5	0	↖ 9	↑ 7	↖ 22
G	0	0	↖ 8	↑ 4	↑ 19	25	↖ 17	↖ 9	↖ 8	↖ 9	↖ 5	↖ 3	↖ 8	↖ 7	↑ 14
T	0	0	0	↖ 15	↑ 11	↖ 26	32	← 24	← 16	← 8	↖ 7	↖ 12	← 4	↖ 15	← 7
A	0	0	↖ 2	↑ 7	↖ 22	↑ 18	↖ 25	↖ 31	↖ 31	↖ 23	← 15	← 7	↖ 14	↑ 7	↖ 22
A	0	5	0	0	↑ 14	↖ 20	↑ 17	↖ 30	↖ 30	↖ 30	↖ 28	← 20	← 12	↖ 12	↑ 14

Figure 2: Local alignment for nucleotides sequences

4 Repeated matches

If one or both of the sequences are long, it is quite possible that there are many different local alignments with a significant score, and in most cases we would be interested in all of these. An example would be where there are many copies of a repeated domain or motif in a protein.

We are only interested in matches scoring higher than some threshold T , because there are always small positive scores even between entirely unrelated sequences.

This method is asymmetric: it finds one or more non-overlapping copies of sections of one sequence (e.g. the domain or motif) in the other. Let y be the sequence containing the domain or motif, and x be the sequence in which we are looking for multiple matrices. In the final

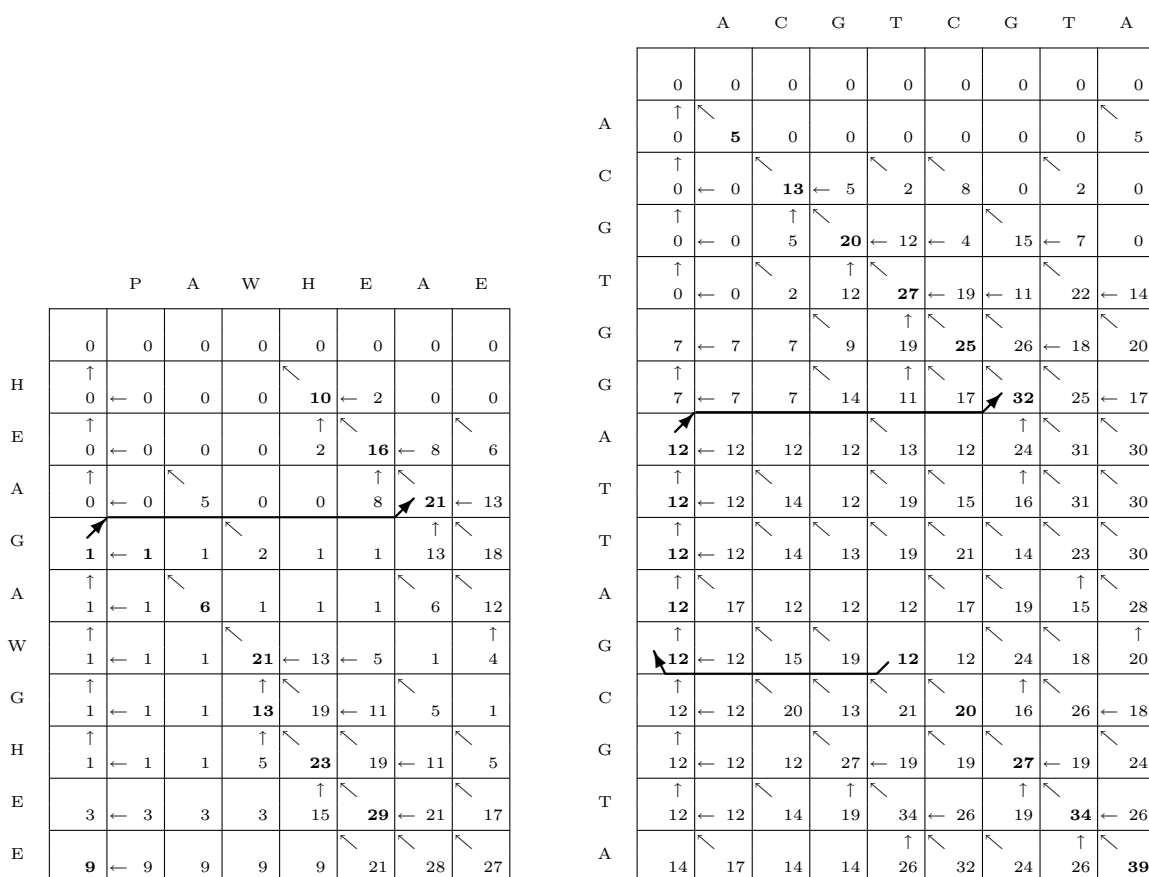
alignment, x will be partitioned into regions that match parts of y in gapped alignments, and regions that are unmatched.

We fill matrix using the following recurrence relations:

$$F(i, j) = \max \begin{cases} F(0, j - 1), \\ F(i, j - 1) - T, i = 1, \dots, n \end{cases} \quad (3)$$

$$F(i, j) = \max \begin{cases} F(i, 0), \\ F(i - 1, j - 1) + s(x_i, y_j), \\ F(i - 1, j) - d, \\ F(i, j - 1) - d \end{cases} \quad (4)$$

Equation (3) handles unmatched regions and ends of matches, only allowing matches to end when they have score at least T . Equation (4) handles starts of matches and extensions.



(a) Protein sequences aligned using BLOSUM50

(b) Nucleotides sequences aligned using PAM1

Figure 3: Repeat dynamic programming alignment

5 Overlap matches

Another type of search is appropriate when we expect that one sequence contains the other, or that they overlap. This often occurs when comparing fragments of genomic DNA sequence to

each other, or to target chromosomal sequences.

What we want is really a type of global alignment, but one that does not penalize overhanging ends.

We fill matrix using the following recurrence relations:

$$F(i, 0) = \max \begin{cases} F(i - 1, 0), \\ F(i - 1, m) - T \end{cases} \quad (5)$$

$$F(i, j) = \max \begin{cases} F(i - 1, j - 1) + s(x_i, y_j), \\ F(i - 1, j) - d \\ F(i, j - 1) - d \end{cases} \quad (6)$$

Note, that the recursion (5) is now just looking at complete matches to $y_{1\dots m}$, rather than all possible subsequences of y in the previous section.

		A	C	G	T	G	G	A	T	T	A	G	C	G	T	A	
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A		0	↖ 5	↖ -2	↖ -2	↖ -1	↖ -2	↖ -2	↖ 5	↖ -1	↖ -1	↖ 5	↖ -2	↖ -2	↖ -2	↖ -1	↖ 5
C		0	↖ -2	↖ 13	← 5	↖ 0	↖ -3	↖ -4	↖ -3	↖ 7	↖ 1	↖ -3	↖ 3	↖ 6	← -2	↖ 0	↖ -3
G		0	↖ -2	↖ 5	↖ 20	← 12	↖ 7	↖ 4	← -4	↖ -1	↖ 6	↖ -1	↖ 4	↖ 1	↖ 13	← 5	↖ -2
T		0	↖ -1	↖ 0	↖ 12	↖ 27	← 19	← 11	↖ 3	↖ 3	↖ 6	↖ 5	↖ -2	↖ 6	↖ 5	↖ 20	← 12
C		0	↖ -2	↖ 7	↖ 4	↖ 19	↖ 25	↖ 17	↖ 9	↖ 5	↖ 5	↖ 4	↖ 3	↖ 6	↖ 4	↖ 12	↖ 18
G		0	↖ -2	↖ -1	↖ 14	↖ 11	↖ 26	↖ 32	← 24	← 16	← 8	↖ 3	↖ 11	← 3	↖ 13	← 5	↖ 10
T		0	↖ -1	↖ 0	↖ 6	↖ 21	↖ 18	↖ 25	↖ 31	↖ 31	↖ 23	← 15	← 7	↖ 13	↖ 5	↖ 20	← 12
A		0	↖ 5	↖ -3	↖ -2	↖ 13	↖ 19	↖ 17	↖ 30	↖ 30	↖ 30	↖ 28	← 20	← 12	↖ 11	↖ 12	↖ 25

Figure 4: Overlap alignment F matrix

6 Alignment with affine gap scores

The standard alternative to using linear gap function (1) is to use affine gap model (2). This model penalizes the alignment break up and may obtain more meaningful results.

Let $M(i, j)$ be the best score up to (i, j) given that x_i is aligned to y_j (left case on fig. 5). $I_x(i, j)$ be the best score given x_i is aligned with to a gap. Finally $I_y(i, j)$ be the best score given that y_j is in and insertion with respect to x .

I	G	A	x_i	A	I	G	A	x_i	G	A	x_i		
L	G	V	y_i	G	V	y_i	-	-	S	L	G	V	y_i

Figure 5: Three possible situations

The recurrences describing the affine global alignments are the following

	A	C	G	T	G	G	A	T	T	A	G	C	G	T	A	
0	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	
A	-∞	5	-14	-16	-17	-20	-22	-17	-25	-27	-23	-32	-34	-36	-37	-33
C	-∞	-14	13	-9	-7	-13	-15	-17	-15	-17	-23	-25	-17	-29	-27	-33
G	-∞	-16	-9	20	0	6	4	-7	-8	-10	-13	-6	-17	-10	-20	-23
T	-∞	-17	-7	0	27	7	5	3	9	7	-3	-5	-4	-9	-3	-13
C	-∞	-20	-3	-3	10	25	13	11	13	11	5	3	11	-1	1	-5
G	-∞	-22	-15	4	5	22	32	11	10	12	9	12	1	18	-2	-1
T	-∞	-23	-13	-6	11	12	21	31	27	25	15	13	14	9	25	5
A	-∞	-19	-19	-9	1	9	10	26	30	26	30	13	11	12	8	30

(a) Matrix M

	A	C	G	T	G	G	A	T	T	A	G	C	G	T	A	
0	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	
A	↑	-12	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞
C	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
G	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
T	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
C	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
G	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
T	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
A	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
	-26	-19	-9	0	9	9	18	19	15	13	3	1	2	4	13	-7

(b) Matrix I_x

	A	C	G	T	G	G	A	T	T	A	G	C	G	T	A	
0	← -12	← -14	← -16	← -18	← -20	← -22	← -24	← -26	← -28	← -30	← -32	← -34	← -36	← -38	← -40	
A	-∞	-∞	← -7	← -9	← -11	← -13	← -15	← -17	← -19	← -21	← -23	← -25	← -27	← -29	← -31	← -33
C	-∞	-∞	← -26	← 1	← -1	← -3	← -5	← -7	← -9	← -11	← -13	← -15	← -17	← -19	← -21	← -23
G	-∞	-∞	← -28	← -21	← 8	← 6	← 4	← 2	← 0	← -2	← -4	← -6	← -8	← -10	← -12	← -14
T	-∞	-∞	← -29	← -19	← -12	← 15	← 13	← 11	← 9	← 7	← 5	← 3	← 1	← -1	← -3	← -5
C	-∞	-∞	← -32	← -15	← -15	← -2	← 13	← 11	← 9	← 7	← 5	← 3	← 1	← -1	← -3	← -5
G	-∞	-∞	← -34	← -27	← -8	← -7	← 10	← 20	← 18	← 16	← 14	← 12	← 10	← 8	← 6	← 4
T	-∞	-∞	← -35	← -25	← -18	← -1	← 0	← 9	← 19	← 17	← 15	← 13	← 11	← 9	← 7	← 13
A	-∞	-∞	← -31	← -31	← -21	← -11	← -3	← -2	← 14	← 18	← 16	← 18	← 16	← 14	← 12	← 10

(c) Matrix I_y Figure 6: Affine gap penalty global alignment with $d = 12$ and $e = 2$

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

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

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

7 Alignment scoring matrices

7.1 Substitution matrix motivation

Consider simplest alignment: ungapped global alignment of two sequences, x and y , of length n . In scoring of these alignments we would like to assess fraction of probabilities

$$\frac{P(x, y|M)}{P(x, y|R)} \Leftarrow \text{sequences have common ancestor}$$

$$\frac{P(x, y|M)}{P(x, y|R)} \Leftarrow \text{sequences are aligned by chance}$$

Scoring matrices are based on this assessment.

Let q_a be frequency of aminoacid a . Consider the case where alignment of x and y is random

$$P(x, y|R) = \prod_i q_{x_i} \prod_i q_{y_i}$$

Let p_{ab} be the probability that a and b derived from a common ancestor. Then the case where the alignment is due to common ancestry is:

$$P(x, y|M) = \prod_i p_{x_i y_i}$$

The odds ratio of these two alternatives is given by

$$\frac{P(x, y|M)}{P(x, y|R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} q_{y_i}}$$

Taking the log we get

$$\log \frac{P(x, y|M)}{P(x, y|R)} = \sum_i \log \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

The score for the alignment is thus given by

$$S = \sum_i s(x_i, y_i) = \log \frac{P(x, y|M)}{P(x, y|R)}$$

Substitution matrix score for the pair a, b is then given by

$$s(a, b) = \log \left(\frac{p_{ab}}{q_a q_b} \right)$$

7.2 Substitution matrix PAM1 for nucleotides

The problem arises is how do we get values for p_{ab} (probability that a and b arose from the common ancestor)?

It depends on how long ago two sequences diverged

- diverged recently $p_{ab} \approx 0$ for $a \neq b$
- diverged long ago $p_{ab} \approx q_a q_b$

The idea how to resolve this issue in PAM substitution matrices was presented in [3]. We need to estimate the probability that b was substituted for a given measure of evolutionary distance. Trusted alignments of closely related sequences provide information about biologically permissible mutations.

There were several steps involved

Step 1 For 71 protein families

- Construct hypothetical phylogenetic trees
- From the trees, fill 20×20 substitution matrix A (20 is the number of aminoacids) with number of observed substitutions $a \rightarrow b$

Step 2 From substitution matrix A , calculate matrix containing

$$P(b|a) = \frac{A_{ab}}{\sum_c A_{ac}}$$

Step 3 Normalize this matrix, so that the expected number of substitutions is 1% out of the whole sequence we get PAM1 matrix

An element of PAM matrix is

$$s(a, b|t) = \log \frac{P(b|a, t)}{q_b} \begin{array}{l} \Leftarrow \text{conditional probability that } a \text{ is substituted by } b \text{ in time } t \\ \Leftarrow \text{frequency of aminoacid } b \end{array}$$

There are several PAM1 substitution matrices available both for the purposes of local and global alignments. Since our work is mostly based on nucleotide database comparisons, we use small matrices for nucleotides. Usually there are more possibilities for *transversions* rather than for *transitions*, observed number of transitions is usually higher in natural processes [8].

- A model of uniform mutations rates among nucleotides used by BLAST

	A	G	T	C
A	+5	-4	-4	-4
G	-4	+5	-4	-4
T	-4	-4	+5	-4
C	-4	-4	-4	+5

- A model of 3-fold higher transition than transversions

	A	G	T	C
A	2	-5	-7	-7
G	-5	2	-7	-7
T	-7	-7	2	-5
C	-7	-7	-5	2

- A PAM1 model taking into account methylation process (according to CpG islands theory)

	A	G	T	C
A	5	-2	-1	-2
G	-2	7	-1	-2
T	-1	-1	7	2
C	-2	-2	2	8

8 Ungapped BLAST

This discussion on BLAST algorithm was mostly borrowed from [9]. Part of the discussion on statistics is coming from [5].

8.1 Overview

The BLAST package provides programs for finding high scoring local alignments between a query sequence and a target database, both of which can be either DNA or protein.

BLAST originated from three convergent efforts.

1. General effort by David Lipman, Warren Gish, and others at NCBI to increase the speed of FASTA by introducing more stringent rules to locate (fewer and better) alignment hot-spots.
2. The sublinear searching work of Myers which introduced the idea of substring neighborhoods and finite-state machines to locate initial alignment hot-spots.
3. The work of Karlin, Altschul, and Dembo, who derived the probability results used in BLAST to evaluate the statistical significance of the reported matches.

BLAST is actually a collection of programs, each tuned to a different problem domain. For example, BLASTP is the version used to search protein databases, and BLAST applies to DNA.

The performance goals of BLAST and all other similar heuristic methods include

1. Maximize speed
2. Minimize storage requirement
3. Maximize sensitivity (percentage of true homologues in database being hit)
4. Maximize specificity (percentage of non-homologues in database not being hit)

BLAST was first published in [1].

8.2 What is HSP (high scoring segment pair)?

Definition 1 Given two strings x and y , a segment pair is a pair of equal-length substrings of x and y , aligned without spaces.

Definition 2 A locally optimal segment pair is a segment pair whose alignment score (without spaces) would fall either by expanding or shortening the segments on either side. All such locally optimal segment pairs are called high scoring segment pairs (HSPs).

8.3 Algorithmic steps

1. Compile a list of high scoring words

- BLAST makes a list of words of a fixed length w (by default 3 for protein sequences, and 11 for nucleic acids), that would match the query sequence somewhere with score higher than threshold T .
- For proteins, the list consists of all w -mers that score at least T when compared to some word in the query sequence.
- For DNA, a simpler word list is used; i.e., the list of all contiguous w -mers in the query sequence.

2. Scan the database for hits

- Search those database entries that have hits with any w -mer of the above compiled word list.
- One approach is to use hash table.
- Map each word to an integer between 1 and 20^w (in case of amino acid residues), so a word can be used as an index into an array of size 20^w .
- Make the i^{th} entry of such an array point to the list of all occurrences in the query sequence of the i^{th} word.
- Typically only small fraction of the 20^w possible words will be in the table, and it is easy to make the table more compact.
- Another approach is to use a deterministic finite automaton.
- Build a finite automaton from the list of compiled words as the search engine.

3. Extend hits

- Extend the hit in both direction to find an HSP with $score \geq S$.

8.4 Statistical background

8.4.1 The Extreme value distribution

Suppose we take n samples of random variable X with probability density $g(x)$. Probability that the largest amongst them is less than x is the n^{th} order statistics

$$P(X \leq x) = F(x) = G(x)^n = \left(\int_{-\infty}^x g(u) du \right)^n$$

To get probability density of the new distribution, we differentiate distribution function

$$\frac{dF(x)}{dx} = ng(x)G(x)^{n-1}$$

Definition 3 The limit $\lim_{n \rightarrow \infty} ng(x)G(x)^{n-1}$ is called extreme value density (EVD) for $g(x)$

It has a wide variety of practical uses, from modelling the breaking point of a chain (which is determined by the weakest link), to assessing the significance of the maximum score from a set of alignments

Example 1 Let us consider the EVD when $g(x) = \alpha e^{-\alpha x}$ is exponential density.

Integration gives us distribution function $G(x) = \int_{-\infty}^x g(x)dx = \int_{-\infty}^x \alpha e^{-\alpha x} dx = 1 - e^{-\alpha x}$

Choosing auxiliary variable y so that $e^{-\alpha y} = \frac{1}{n}$, and introducing another variable $z = x - y$, we rewrite EVD for $g(x)$

$$ng(x)G(x)^{n-1} = n\alpha e^{-\alpha x} (1 - e^{-\alpha x})^{n-1} = \alpha e^{-\alpha z} \left(1 - \frac{e^{-\alpha z}}{n}\right)^{n-1}$$

according to definition 3 we find EVD

$$\lim_{n \rightarrow \infty} \alpha e^{-\alpha z} \left(1 - \frac{e^{-\alpha z}}{n}\right)^{n-1} = \alpha e^{-\alpha z} \exp(-e^{-\alpha z})$$

where we used well-known limit

$$\lim_{n \rightarrow \infty} \left(1 - \frac{x}{n}\right)^n = e^{-x}$$

The extreme value cumulative probability $P(Z \leq z) = \exp(-e^{-\alpha z})$ is called Gumbel distribution

The Gumbel distribution is the EVD for a variety of underlying densities $g(x)$; it holds when $g(x)$ is Gaussian, for instance.

More generally, an EVD must have the form

$$\exp(-f(a_n x + b_n))$$

where

a_n and b_n are constants depending on n

$f(x)$ is either exponential e^{-x} or $|x|^{-\lambda}$ for some positive constant λ .

8.5 Statistical significance of HSP scores

8.5.1 Pairwise P value and E value

The probability distribution of HSP scores can be approximated by the following extreme value distribution [6].

$$P(\text{HSP score for } x \text{ and } y > S) = 1 - \exp(-K m n e^{-\lambda S})$$

where

m and n are length of two random sequences x and y ,
 S is predefined threshold score,
 K and λ are constant values derived from the background probabilities of individual residues and the scores for aligning pairs

Complementary probability is

$$P(\text{no segment pair with score at least } S \text{ was found}) = \exp(-Kmn e^{-\lambda S})$$

Here we model the number of segment pairs with score at least S as a Poisson random variable. Define E to be the expected number of segment pairs with score at least S . Then

$$P(\text{exactly } k \text{ segment pairs with score at least } S \text{ were found}) = \frac{e^{-E} E^k}{k!}$$

$$P(\text{no segment pair with score at least } S \text{ was found}) = e^{-E} = \exp(-Kmn e^{-\lambda S}) \quad (7)$$

From (7) it follows that $E = Kmn e^{-\lambda S}$.

P value and E value are thus derived from extreme-value and Poisson distributions $P = 1 - \exp(-Kmn e^{-\lambda S})$, the probability of HSP score exceeding S by chance.

The expected number of segment pairs with score at least S is

$$E = Kmn e^{-\lambda S},$$

If raw score S is normalized to a bit score with the formula $S' = \frac{\lambda S - \ln K}{\ln 2}$, then E value can be expressed in terms of S' as $E = mn 2^{-S'}$. When $E < 0.01$, E and P values are nearly identical. The larger the value of S , the smaller the P value.

8.5.2 "Database search" P value and E value

Treat the database as a single long sequence. The n in formulas is replaced with total number of residues in database.

8.6 Parameter settings and default values

For amino acid sequence comparisons, PAM-120 scoring matrix is used by default. DNA comparisons score identities +5, and mismatches -4. A scoring matrix is selected such that the expected score for a pair of residues must be negative; that is,

$$\sum P_i P_j s_{ij} < 0$$

where

s_{ij} is the score for aligning residues i and j ,
 P_i is the background probability of occurrence of residue i ,
 K and λ are constant values derived from the background probabilities of individual residues and the scores for aligning pairs.

To "customize" a scoring matrix, the most direct way is to estimate target and background frequencies, and calculate log-odds scores from the corresponding target and background frequencies.

A class of alignment can be characterized by the frequency with which each possible pair of residues is aligned. For example, if *Valine* in the first sequence and *Leucine* in the second appear in 1% of all alignment positions, the target frequency for (*Valine*, *Leucine*) is 0.01. Denote target frequency of pair of residues i and j by q_{ij} . Target frequencies must sum to 1. Use the theorem $q_{ij} = P_i P_j \exp(\lambda s_{ij})$ to compute log-odds scores of s_{ij} 's. Upper bound on number of false hits (called *EXPECT*) is set to be 10 by default.

8.6.1 Choice of word length w

- w is set to 3 by default for protein sequences, and 12 for nucleic acids.
- If the goal is to minimize the probability of of a random hit (increasing specificity), we should increase w but at the expense of execution speed because the number of words generated would grow exponentially with w .
- If the goal is to increase sensitivity, we should decrease w .

8.6.2 Determination of threshold T

The choice of threshold T is guided by the theorems of [6] and [4] based on the scoring matrix and on characteristics of the query sequence and of the database sequences. Define q to be the probability that an HSP of two random sequences was filtered out by the first phase of the algorithm because its score is less than the threshold T . Probability q should decrease exponentially with increasing S . For given w and S , q decreases (thus sensitivity increases) as T decreases but at the expense of execution speed. There is tradeoff between sensitivity and time.

8.6.3 Determination of threshold S

Conceptually, $EXPECT = N \times P$ where $N = n \times m$, the total window size, is the product of database size and query sequence length. Since P is a function of S , once the value of $EXPECT$ is chosen, the value of S can be determined.

$$S \approx \lambda^{-1} \ln \left(\frac{KN}{q} \right) \text{ where } q = \frac{EXPECT}{N}$$

Its derivations are described below.

$$q = 1 - \exp(-KWe^{-\lambda S}), \text{ by } EXPECT = N \times P$$

$$-KNe^{-\lambda S} = \ln(1 - q), \text{ by algebraic manipulations}$$

$$KNe^{-\lambda S} \approx q, \text{ since } \ln(1 - x) \approx -x \text{ if } x \ll 1$$

$$S \approx \lambda^{-1} \ln \left(\frac{KN}{q} \right), \text{ by algebraic manipulations}$$

8.7 Performance tunings

For DNA, it is advantageous to compress the database by packing 4 nucleotides into a single byte. This allows to scan the database byte-wise. For each 8-mer hit, check for an enclosing w -mer hit. DNA sequences are highly non-random, with locally biased base composition (e.g., AT-rich regions), and repeated sequence elements (e.g., *Alu* sequences). If a given query sequence has, for example, an AT-rich subsequence, or a commonly occurring repetitive element, then a database search will produce a copious output of matches with little interest. In database, those 8-mers occurring much more frequently than expected by chance are stored and used to filter "uninformative" words from the query word list.

9 Gapped BLAST

9.1 Two-hit method [2]

9.1.1 Motivation and rationale

- The primitive BLAST algorithm spent 90% of time in hit-extension phase. The two-hit methods is to reduce the number of extensions performed.
- The two-hit method is based on the observation that an HSP of interest is much longer than a single word pair, and may therefore entail multiple hits on the same diagonal and within a relatively short distance of one another.

9.1.2 The method

- Invoke an extension only when two non-overlapping hits are found within distance A of one another on the same diagonal.
- The extension between the two hits may contain gaps.

9.1.3 Modifications to parameter setting

The threshold parameter T must be lowered to retain comparable sensitivity because two hits rather than one is needed to invoke an extension. The effect is that many more single hits are found, but the majority of them may be dismissed, and thus the total number of extensions are in fact reduced.

9.2 Autonomous gapped-extension triggering

9.2.1 Motivation and rationale

The two-hit method allows gapped extension only when the two hits are on the same diagonal and are within certain distance. If any one of the two was filtered out, a potential longer HSP would be missed.

9.2.2 The method

- A gapped extension is triggered for any HSP that exceeds score S_g . The HSP to be extended may come from the result of the two-hit method.

- Starting from a single pair of aligned residues (called seed), the dynamic programming proceeds both forward and backward through the path graph by considering only cells for which the optimal local alignment score falls no more than X_g below the best alignment score yet found. The alignment might wander arbitrarily many diagonals away from the seed, but the number of cells expanded on each row tends to remain limited.
- The resulting alignment is reported only when its E -value is sufficiently low.

9.2.3 The benefit

The threshold T can be set higher without sacrificing sensitivity. The execution speed can thus be further improved.

9.2.4 Heuristics in choosing parameters

- The "seed" is chosen from the highest-score length-11 segment along the HSP to be extended. The central residue pair of such a length-11 segment is picked up as the "seed".
- S_g is determined such that a gapped extension is invoked less than once per 50 database sequences.

9.3 Statistics

- The statistical parameters λ and K can no longer be calculated analytically on the fly, but must be pre-estimated from comparisons of random sequences.
- Once λ and K are determined, the statistics for P -value and E -value can be computed in the same manner, although there lacks of analytical theory for gapped alignments.

10 PSI-BLAST

10.1 Motivations

- Database searches using position-specific score matrices often are much better able to detect weak relationships.
- It is often desirable to determine which positions in the query sequence are conserved during evolution. Those weak but biologically relevant sequence relationships would generally be unable to reveal in regular pairwise alignments.
- PSI (Position Specific Iterated) BLAST was thus created to offer a tool for "profile alignment" for protein sequences. It constructs a position-specific score matrix automatically from the output of a BLAST run, and then operate BLAST using such a matrix [2].

10.2 Position-specific score matrix

- Position-specific score matrix is an $L \times p$ matrix, where L is the length of the query sequence, and p is the size the alphabet. In PSI-BLAST, p is 20.
- Why are position-specific score matrices more powerful?

1. They have more sensitive scoring capability because of improved estimation of the probabilities with which residues occur at various pattern positions.
2. They can lower the level of noise because of relatively precise definition of the boundaries of important "patterns".

10.3 Algorithmic steps of a PSI-BLAST iteration

1. Collect all database sequence segments that have been aligned to the query with E -value below a threshold, by default set to 0.01.
2. Construct a multiple alignment from BLAST output
 - The original query is used as a master for constructing multiple alignment M . Only one copy is retained of any rows that are greater than 98% identical to one another. Pairwise alignment columns that involve gaps inserted into the query are ignored, so that M has the same length as the master.
 - Prune raw multiple alignment M to a simpler reduced version M_c
 - First, specify the set R of sequences to be exactly those that contribute a residue to each column.
 - Next, define the columns of M_c to be just those columns of M in which all the sequences of R are represented.
 - Assign weights to rows of the resulting multiple alignment
 - Those rows being closely related receive smaller weights.
3. Construct position-specific score matrix from multiple alignment
 - The score of residue i for a specific pattern position is $\frac{\ln(\frac{Q_i}{P_i})}{\lambda_u}$, where λ_u is the scale parameter computed during the first BLAST run, Q_i is the estimated probability for residue i to be found in that column, and P_i is the background probability for residue i . The purpose of λ_u here is to make the position-specific matrix has the same scale with the substitution matrix used.
 - For a multiple alignment involving a large number of independent sequences, the estimate of Q_i could simply be observed frequency. Otherwise, Dirichlet mixture is suggested to be the best estimate. However, PSI-BLAST implemented a data-dependent pseudocount method

$$Q_i = \frac{\alpha f_i + \beta g_i}{\alpha + \beta}$$

where

f_i is observed frequency of residue i in the specific column.

g_i is pseudo count of residue i in the specific column.

α and β are the relative weights given to observed and pseudocount residue frequencies.

- Empirical data showed that $\alpha = N_c - 1$ and $\beta = 10$ is a good setting, where N_c is mean number of different residue types including gap characters, observed in the specific column of computed multiple alignment.

- For a specific column, pseudo count for residue i is obtained by $g_i = \sum_j \frac{f_j q_{ij}}{P_j}$

where

$q_{ij} = P_i P_j \exp(\lambda_u s_{ij})$ is the target frequency estimated from substitution matrix and background probabilities.

4. Use position-specific score matrix to search the database

- Local alignments are searched in the database with the built position-specific score matrix.
- In each iteration of PSI-BLAST, the same affine gap scores are used as in the first, simple BLAST run because there is no good theory for deriving position-specific gap cost from multiple alignment.
- For a score matrix constructed to the same scale as the substitution matrix, a given set of gap costs should produce the same gapped alignment scale parameter λ_g as for s_{ij} . This hypothesis has been tested valid by experimental data. λ_g can thus be precomputed and reused at each iteration of PSI-BLAST for calculating statistical significance.

References

- [1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D.J. Lipman, *Basic local alignment search tool*, Journal of Molecular Biology **215** (1990), 403–410.
- [2] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman, *Gapped blast and psi-blast: a new generation of protein database search programs*, Nucleic Acids Research **25** (1997), 3389–3402.
- [3] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt, *A model of evolutionary change in proteins*, Atlas of protein sequence and structure **5** (1978), 345–352.
- [4] A. Dembo and S. Karlin, *Strong limit theorems of empirical functionals for large exceedances of partial sums of i.i.d. variables*, Annals of Probability **19** (1991), 1737–1755.
- [5] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis*, Cambridge University press, 1998.
- [6] S. Karlin and S. F. Altschul, *Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes*, Proceedings of the National Academy of Sciences of the USA **87** (1990), 2264–2268.
- [7] T. F. Smith and M.S. Waterman, *Identification of common molecular subsequences*, Journal of Molecular Biology **147** (1981), 195–197.
- [8] A.Y. Tchourbanov, *Blosum and pam matrices for blast algorithm*, Presentation, Apr. 2002, Available at <http://www.cse.unl.edu/~tchourba/docs/slideShowGerm.pdf>.
- [9] H. Kaiser Yang, *Under the hood of blast*, Presentation, Mar. 2002, Available at <http://www.kaiseryang.com/bio/align/BLAST.doc>.