Bioinformatics Algorithms

(Fundamental Algorithms, module 2)

Zsuzsanna Lipták

Masters in Medical Bioinformatics
academic year 2018/19, II. semester

$\boxed{\text{Pairwise Alignment in Practice}}$

Visualization with dotplots

Dot plots

The simplest way of visualizing similarities between two sequences is a dot plot (or dot matrix):

- matrix of size $|s| \times |t|$;
- put a dot in position (i, j) iff $s_i = t_j$.
- can also be used to show self-similarity (repeats)

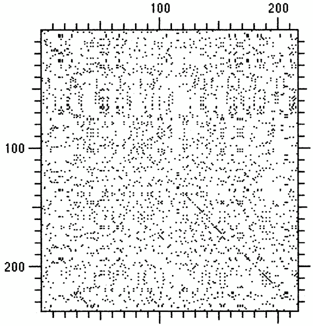


Figure 3.5. Dot matrix analysis of the amino acid sequences of the phage λ *cI* (*horizontal sequence*) and phage P22 *c2* (*vertical sequence*) repressors performed as described in Fig. 3.4. The window size and stringency were both 1.

source: D. Mount: Bioinformatics

Dot plots

The simplest way of visualizing similarities between two sequences is a dot plot (or dot matrix):

- matrix of size $|s| \times |t|$;
- put a dot in position (i, j) iff $s_i = t_j$.
- can also be used to show self-similarity (repeats)
- Advantage: easy to compute and easy to understand.
- Drawback: not always easy to interpret, esp. with small alphabets (too many dots!)

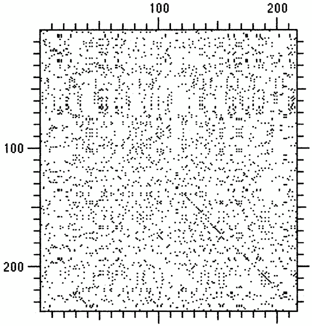


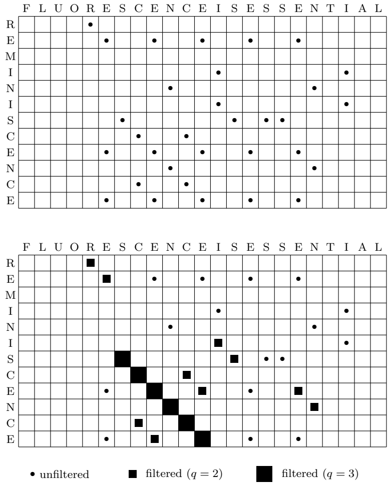
Figure 3.5. Dot matrix analysis of the amino acid sequences of the phage λ *cI* (*horizontal sequence*) and phage P22 *c2* (*vertical sequence*) repressors performed as described in Fig. 3.4. The window size and stringency were both 1.

source: D. Mount: Bioinformatics

Dot plots

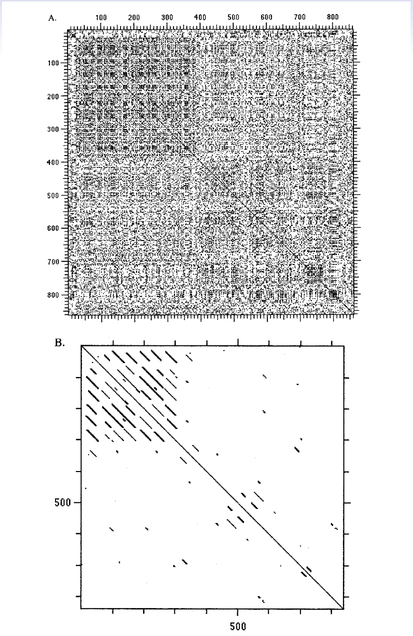
One solution is to restrict dots to positions which are part of a longer stretch of exact matches:

- choose parameter q
- if $s_i \cdots s_{i+q-1} = t_j \cdots t_{j+q-1}$, then put a dot in positions $(i,j), (i+1, j+1), \ldots, (i+q-1, j+q-1)$.
- on the right: unfiltered dot plot for two strings s, t, and with filters $q = 2, 3$.



source: Lecture Notes "Seq. Analysis", Bielefeld Univ.

- choose parameters q, r (q windowsize, r stringency)
- if there are at least r matches within a window of size q, then put a dot in each of these positions, i.e. if the Hamming distance of $s_i \cdots s_{i+q-1}$ and $t_j \cdots t_{j+q-1}$ is at least r, then put a dot in positions $(i, j), (i+1, j+1), \ldots, (i+q-1, j+q-1)$.
- on the right: Human LDL receptor against itself; A. window=1, str.=1, B. window=23, str.=7.



source: D. Mount: Bioinformatics

Database search with BLAST

Database search

- Until now: compare two sequences
  - how similar/different are they? (score/value)
  - where are the similarities/differences? (alignment)

Database search

- Until now: compare two sequences
  - how similar/different are they? (score/value)
  - where are the similarities/differences? (alignment)
- Now: compare one sequence to a database (i.e. to many sequences)

Database search

Goal:

Identifying sequences in the DB which have high local similarity with the query.

- We know how to do this: Smith-Waterman DP-algorithm.

Database search

Goal:
Identifying sequences in the DB which have high local similarity with the query.

- We know how to do this: Smith-Waterman DP-algorithm.
- But: too slow!

Say all sequences have length n (query t and all DB seq's), and there are r sequences in the DB.

- time of exact solution (Smith-Waterman):

Say all sequences have length n (query t and all DB seq's), and there are r sequences in the DB.

- time of exact solution (Smith-Waterman): $O(r \cdot n^2)$

Say all sequences have length n (query t and all DB seq's), and there are r sequences in the DB.

- time of exact solution (Smith-Waterman): $O(r \cdot n^2)$

Example

- UniProt/SwissProt (protein database): 548 454 sequences, 195 409 447 aa's (avg. length 350 aa's) version 29/04/15
- NCBI Genbank (nucleotide database): 182 188 746 sequences, 189 739 230 107 nucleotides (avg. length 1041 nucl.) April 2015, no WGS

Say all sequences have length n (query t and all DB seq's), and there are r sequences in the DB.

- time of exact solution (Smith-Waterman): $O(r \cdot n^2)$

Example

- UniProt/SwissProt (protein database): 548 454 sequences, 195 409 447 aa's (avg. length 350 aa's) version 29/04/15
- NCBI Genbank (nucleotide database): 182 188 746 sequences, 189 739 230 107 nucleotides (avg. length 1041 nucl.) April 2015, no WGS

So we would get something like $350 \cdot 350 \cdot 548454 = 67\,185\,615\,000 =$ about 67 billion ($67 \cdot 10^9$) steps, which takes 18 hours on a computer that performs 1 million operations per second (for UniProt), and 197 434 482 454 026 ($\approx 1.9 \cdot 10^{12}$), about 6 years, for Genbank. And still about 1 hour on a computer performing 1 billion operations per second.

Say all sequences have length n (query t and all DB seq's), and there are r sequences in the DB.

- time of exact solution (Smith-Waterman): $O(r \cdot n^2)$

Example

- UniProt/SwissProt (protein database): 548 454 sequences, 195 409 447 aa's (avg. length 350 aa's) version 29/04/15
- NCBI Genbank (nucleotide database): 182 188 746 sequences, 189 739 230 107 nucleotides (avg. length 1041 nucl.) April 2015, no WGS

So we would get something like $350 \cdot 350 \cdot 548454 = 67\,185\,615\,000 =$ about 67 billion ($67 \cdot 10^9$) steps, which takes 18 hours on a computer that performs 1 million operations per second (for UniProt), and 197 434 482 454 026 ($\approx 1.9 \cdot 10^{12}$), about 6 years, for Genbank. And still about 1 hour on a computer performing 1 billion operations per second.

And this is for one query only!

BLAST: Basic Local Alignment Search Tool

- Altschul *et al.* 1990, 1997 (among the most highly cited papers in bioinformatics)
- looks for sequences in a database with high local similarity to query
- heuristic algorithm
- solid mathematical foundations (Karlin-Altschul statistics)
- extremely successful, now the database search tool ("to blast a sequence against a database")
- NCBI[1] Blast at:
  `http://blast.ncbi.nlm.nih.gov/Blast.cgi`

[1] NCBI = National Center for Biotechnology Information

Basic idea

Basic idea

If there is a good local alignment between two sequences, then this local alignment is likely to contain a pair of short substrings with high score when aligned without gaps.

Basic idea

Basic idea

If there is a good local alignment between two sequences, then this local alignment is likely to contain a pair of short substrings with high score when aligned without gaps.

Basic steps of BLAST

1. create list of high-scoring words with query
2. scan DB for these words (called seeds)
3. extend seeds in both directions to form good gapless local alignment (locally maximal segment pairs = HSPs)

Parameters

The original BLAST uses the following parameters:

- w: word size (length of high-scoring words)
  default for DNA: $w = 11$, for protein: $w = 3$.
- T: threshold for high-scoring words
- d: absolute drop from highest scoring extension so far, or
  α: relative drop from highest scoring extension so far
- S: threshold for retaining HSPs

Underlying theory of MSPs (maximal segment pairs) allows to estimate the highest MSP score S at which chance similarities are probable. HSPs are an approximation of MSPs; BLAST retains only those HSPs from the last step whose score is above this threshold S.

Step 1: create list of high-scoring words

Let t be the query sequence.
A word v of length w is called high-scoring if there exists a substring u of t s.t. $\mathit{score}(u, v) \geq T$, where $\mathit{score}(u, v) = \sum_{i=1}^{w} f(u_i, v_i)$, the score of a gapless alignment of u with v. In other words, high-scoring words are the elements of the set

$$\mathcal{H} = \bigcup_{i=1}^{|t|-w+1} \mathcal{N}(t_i \cdots t_{i+w-1}),$$

where $\mathcal{N}(u) = \{v \,:\, \mathit{score}(u, v) \geq T\}$ is the T-neighborhood of the word u.
Note that not every w-substring of t is necessarily element of $\mathcal{H}$ (its score with itself could be below T). Also, a word v could be high-scoring thanks to its closeness to two different w-substrings of t.

Example

- $w = 3$, $T = 22$, using the PAM250 scoring matrix.
- $t = \ldots$ `FRNFKCVDNYAWC` $\ldots$
- Step 1: Generate high-scoring words. For example, score(`FKC`,`FKC`) $= 26$, score(`FKC`,`FRC`) $= 24$, score(`FKC`,`FNC`) $= 22$, score(`FKC`,`YKC`) $= 24$, score(`FKC`,`YRC`) $= 22 \ldots$ — these are all high-scoring w.r.t. the substring `FKC` of t. Others are high-scoring w.r.t. another substring of t, e.g. `FWC` is high-scoring because score(`FWC`,`AWC`) $= 26$ (but not w.r.t. `FKC`, since score(`FWC`,`FKC`) $= 18 < 21$).
- So for each high-scoring word $v \in \mathcal{H}$, we need a list of positions i in t s.t. $\text{score}(v, t_i \cdots t_{i+w-1}) \geq T$.
- Some high-scoring words are then: `FKC,FRC,FNC,YKC,YRC`, $\ldots$ (w.r.t. `FKC`), `AWC,FWC,DWC,LWC`, $\ldots$ (wr.t. `AWC`), $\ldots$

Step 2: Find occurrences of high-scoring words in DB sequences

Step 2. For each high-scoring word v, find all occurrences of v in the DB (i.e. in some sequence s^k in the DB). These are called seeds.

Example (cont.)

Let $v =$ `FRC`, which is high-scoring w.r.t. `FKC` (substring of t). Let the following be a sequence from the DB:

$$s = \ldots \texttt{RNKDQKFRCAVDYAGM} \ldots$$

N.B.: This can be done efficiently using dedicated data structures for strings (e.g. generelized suffix array); this is beyond the scope of this course.

Step 3: Try to extend seeds

Step 3. For each of these seeds, try to extend to an HSP: Let s^k have an occurrence of a high-scoring word v (w.r.t. $u = t_i \cdots t_{i+w-1}$) in position j, then we already know that

$$\begin{pmatrix} t_i t_{i+1} \ldots t_{i+w-1} \\ s^k_j s^k_{j+1} \ldots s^k_{j+w-1} \end{pmatrix}$$

is a gapless alignment with score $\geq T$. We try to extend it in both directions to get a good HSP/MSP.

Example (cont.)

$$t = \ldots\texttt{FRNFKCVDNYAWC}\ldots$$
$$s = \ldots\texttt{RNKDQKFRCAVDYAGM}\ldots$$

We extend this alignment to both sides character by character, to get a good gapless local alignment. When do we stop? We could stop whenever we find a negative score (here at $f(\texttt{V}, \texttt{A}) = -2$); however, then we could miss a good longer local alignment. So one possibility is to set a maximum difference d to current best score: extend until score $< X - d$, where $X =$ highest-score-seen-so-far. Another is to set a relative difference α: we extend until we drop below $(1 - \alpha)X$. E.g. for $\alpha = 0.1$ we get:

$$\begin{pmatrix}\texttt{RNFKCVDNYA}\\ \texttt{QKFRCAVDYA}\end{pmatrix}$$

with score 38. This local alignment is now retained iff $38 > S$.

BLAST2

Some of the main changes in BLAST2 (Altschul *el al.* 1997)

- start with two seeds instead of one, not too far apart
- gapped alignments
- extension of statistical theory to HSPs (high-scoring segment pairs)

Note: All versions of BLAST include many complex pre- and postprocessing steps, optimizations, . . . These are explained in the cited papers, and followup publications. Here we only looked at the basic ideas of the algorithm.

The NCBI BLAST website

- Different versions of BLAST, depending on the task (nucl-nucl: blastn, megablast, . . . , prot-prot: blastp, psi-blast, nucl-prot: blastx, prot-nucl: tblastn, . . .)
- Different databases (nucl vs. prot, different organisms, different types of db, different levels of assembly, . . .)
- Very good explanations and help pages!