CS 284A: Algorithms for Computational Biology Notes on Lecture:
BLAST. The statistics of alignment scores.

prepared by Oleksii Kuchaiev,
based on presentation by Xiaohui Xie on February 20th.

1 Introduction

BLAST (Basic Local Alignment Search Tool) is a fast pair-wise alignment and database searching tool. This is a heuristic algorithm - it does not guarantee an optimal solution and emphasizes speed over sensitivity. This emphasis on speed is vital to making the algorithm practical on the huge genome databases currently available.

Conceptually BLAST algorithm can be divided into 3 stages:

- In the first stage, BLAST searches for exact matches of a small fixed length W between the query and sequences in the database. This is done using Dot Plots and Hashtables.
- In the second stage, BLAST tries to extend the match in both directions, starting at the seed. The ungapped alignment process extends the initial seed match of length W in each direction in an attempt to boost the alignment score. Insertions and deletions are not considered during this stage.
- In the third stage, BLAST performs a gapped alignment between the query sequence and the database sequence using a variation of the Smith-Waterman algorithm. Statistically significant alignments are then displayed to the user.

The original BLAST algorithm makes only local gapless alignment, while Gapped BLAST (BLAST 2.0) allows gaps in the alignment and is about 3 times faster than original BLAST.

2 Dot Plots

To be able quickly find exact matches BLAST uses strategy of dot plots. Dot Plot is a table whose rows and columns are marked using sequences letters.For example:

Identify internal repeats and inversions

	C	A	T	D	O	G	C	A	T	G	O	D
C	*						*					
A		*						*				
T			*						*			
D				*								*
O					*						*	
G						*				*		
C	*						*					
A		*						*				
T			*						*			
G						*				*		
O					*						*	
D				*								*

This dot plot allows us quickly identify internal repeats and inversions. Repeats are dot lines which go from left to right, inversions - dot lines which go from right to left.

Compare two sequences

	T	H	E	F	A	T	C	A	T
T	*					*			*
H		*							
E			*						
F				*					
A					*			*	
S									
T	*					*			*
C							*		
A					*			*	
T	*					*			*

In general, dot plots:

- Allow quick detection of high similarity.
- Allow to identify internal repeats and inversions of a new sequence.
- Is a global alignment strategy that is also useful for visualizing local matches

To filter out noise from the random matches window sliders are used. A dot is recorded at window positions where the number of matches is greater than or equal to the stringency.

Sliding Window; Window length=3, stringency=2

	T	H	E	F	A	T	C	A	T
T	*								
H		*							
E			*						
F				*					
A									
S					*				
T						*			
C				*			*		
A								*	
T									*

In this example we plot a dot iff 2 out of 3 next positions (including current position) are identical. For example we didn't put dot in position [A][A], because we have following 2 3-mers: AST and ATC, which have only 1 identical character (order matters). And we put dot in [S][A], because we have following 3-mers: STC and ATC, which have 2 identical positions.

3 Words and Hash scores.

During it's first step BLAST algorithm divides all sequences in the database into overlapping constituent words (of some length w), and stores starting positions of these words in sequences in special hash tables. These hashtables allow finding positions of the words in the sequences in O(1) time. In practice, these hashtables already precomputed and stored together with sequences in database which is used by BLAST.

BLAST divides all database sequences into overlapping constituent words of size w like in the following example:

Database sequence

1	2	3	4	5	6	
N	L	N	Y	T	P	W
N	L					
	L	N				
		N	Y			
			Y	T		
				T	P	
					P	W

Then it converts each such word into Hash Score (HS) between 0 and $|S|^w - 1$ where S - size of the alphabet, which is S=20 for protein sequences and S=4 for DNA sequences. This is very similar to using scale of notation with the base S (S=20 in case of proteins). By analogy, letters are "digits" and words are "numbers".

A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

The Hash Score is then computed using following formula: $HS(NL) = 20^1 * (Scoreof N) + 20^0 * (Scoreof L) = 20^1 * 11 + 20^0 * 9 = 229$. For w=2, hash scores are between 0 and 399; HS(AA) = 0 and HS(YY) = 399.

After computing hash scores, algorithm finds positions of each word in the sequences, sort all hash scores and adds them into hashtables. For example, for database sequence **NLNYTPW** we have:

Before sorting

word	**HS**	**Location**
NL	229	1
LN	191	2
NY	239	3
YT	396	4
TP	332	5
PW	258	6

After sorting

HS	**Location**
0	-1
1	-1
2	-1
...	
191	2
...	
229	1
...	
239	3
...	
258	6
...	
258	6
...	
332	5
...	
396	4
...	

In practice, this step is precomputed - i.e these hash tables are already computed and stored in the database for each database sequence on the servers which provide BLAST.

Next, BLAST algorithm divides the query sequence into overlapping constituent words and convert them into hash scores. In fact, this is the first what BLAST tools does, because hashtables for sequences in the database are already precomputed.

Query	**HS**
QLNFSAGW	
QL	269
LN	191
NF	224
FS	95
SA	300
AG	5
GW	118

After it has hastables for each sequence in the database and for query sequence it identifies all synonyms of the words in query sequence. This identification is based on the substitution matrix (for

example PAM 120) and similarity threshold.

Let our query sequence is **QLNFSAGW** and database sequence is **NLNYTPW**, then if we use substitution matrix PAM 120 we will have the following table of synonyms for words in query sequence.

2-letter words that score at least T=8 when aligned with the query 2-letter word using the PAM 120 matrix:

1 QL	QL=11, QM=9, HL=8
2 **LN**	**LN**=9
3 **NF**	NF=12, AF=8, **NY**=8, DF=10, QF=8, EF=9, GF=8, HF=10, KF=9, SF=9, TF=8, BF=11, ZF=8
4 FS	FS=12, FA=9, FN=9, FD=8, FG=9, FP=9, FT=10, YS=8
5 SA	none
6 AG	AG=8
7 **GW**	AW=13, RW=8, NW=12, DW=12, QW=9, EW=11, GW=17, HW=8, IW=8, KW=9, MW=8, **PW**=10, SW=13, TW=11, VW=10

(Possible matches between words in query sequence and database sequence are shown in bold)

After that BLAST locates the hits by looking up the synonyms of the query words in the sorted database hash table.

HS	**words**	**Location in query**	**Syn**	**HS**	**Location in database**	**Hits Diagonal**
269	QL	1				
191	LN	2	LN	191	2	0
224	NF	3	NY	239	3	0
95	FS	4				
300	SA	5				
5	AG	6				
118	GW	7	PW	258	6	-1

And at the end of it's first step BLAST uses dot plots to identify matches between query secuence and database sequence.

		1	2	3	4	5	6	7	
		Q	L	N	F	S	A	G	W
1	N								
2	L		*						
3	N			*					
4	Y								
5	T								
6	P							*	
	W								

4 Extension step.

This is the second step of BLAST algorithm. During this step it extends hits (found on the first step) in both directions. There are several BLAST versions and they may differ in the way they do extensions of hits.

Original BLAST. Original BLAST constructs gapless alignment. Each hit is extended in both directions until the running alignments score has dropped more than X below the maximum score yet attained.

BLAST 2.0. This version of algorithm allows constructing alignments with gaps. If two non-overlapping hits are found within distance A of one another on the same diagonal, then merge the hits into an alignment and extend the alignment in both directions until the running alignments score has dropped more than X below the maximum score yet attained. If an extended alignment has a score above S then it is a high-scoring segment pair or HSP.

BLAST 2.0 evokes a gapped alignment for any HSP exceeding some score S_g.

- Dynamic Programming is used to find the optimal gapped alignment. (Approach similar to Needleman-Wunsch and Smith-Waterman algorithms).
- Only alignments that drop in score no more than X_g below the best score yet seen are considered.

- Because a Dynamic Programming approach is used a gapped extension takes much longer to execute than an ungapped extension, but S_g is chosen so that no more than about one extension is invoked per 50 database sequences.
- The resulting gapped alignment is reported only if it has an E-value low enough to be of interest.

Default parameters for Gapped BLAST (BLAST 2.0)

- word size: w = 3.
- threshold parameter: T = 11.
- window length for extending two hits: A=40.
- amino acid substitution matrix: BLOSUM62.

The length $w = 3$ for protein sequences means that in hastables constructed by BLAST there are $20^3 = 8000$ possible words. Longer sequences ($w > 3$) are not usually used because then it is possible that the algorithm won't find hits in the query and database sequences. For nucleotide sequences search, the w by default equals 11, which means $4^{11} = 4194304$ possible words. Shorter word lengths may increase sensitivity, at the expense of increased run time.

Execution time

- The extension step in the original version of BLAST usually accounts for $> 90\%$ of the execution time.
- Since BLAST 2.0 requires two hits rather than one to invoke an extension, the threshold parameter T must be lowered to retain comparable sensitivity.
  - Many more single hits are found but only a small fraction have an associated second hit on the same diagonal that triggers an extension.
  - The computation saved by requiring fewer extensions more than offsets the extra computation required to process the larger number of hits.

5 Reported Quantities and Implementations.

As a result BLAST report the following quantities:

- **Nominal Score:** The raw score which is the sum of the similarity scores and gap penalties. This is dependent on the query, the database and the scoring scheme.
- **Bit Score S:** Normalized score of the final gapped alignment. This is still dependent on the lengths of the query and the database, but presumably is independent of the scoring scheme.
- **E-value:** Expected number of times of finding such a score (or better) by chance.
- **P-value:** Probability of finding such a score (or better) by chance.

5.1 BLAST implementations.

For protein sequences.

blastp: compares a protein sequence with a protein database.

- Allows to learn something about the structure and function of a protein.
- This search is similar to the standard protein-protein BLAST with the parameters set to optimize for searching with short sequences.
- $http://www.incogen.com/public_documents/vibe/details/NcbiBlastp.html$

tblastn: compares a protein sequence with a nucleotide database

- discover genes that encode a protein
- Searches translate either query sequences or databases from nucleotides to proteins so that protein - nucleotide sequences can be performed. Tblastn takes a protein query sequence and compares it against an NCBI nucleotide database which has been translated in all six reading frames.
- $http://www.incogen.com/public_documents/vibe/details/NcbiTblastn.html$

For DNA sequences.
blastn: compares a DNA sequence with a DNA database

- Compares very similar DNA sequences.
- $http://www.incogen.com/public_documents/vibe/details/NcbiBlastn.html$

tblastx: compares a translated DNA sequence with a translated DNA database

- Can be used to discover new proteins.
- Searches translate either query sequences or databases from nucleotides to proteins so that protein - nucleotide sequences can be performed. Tblastx converts a nucleotide query sequence into protein sequences in all 6 reading frames and then compares this to an NCBI nucleotide database which has been translated on all six reading frames.
- $http://www.incogen.com/public_documents/vibe/details/NcbiTblastx.html$

blastx: compares translated DNA with a protein database

- Used to analyze the query DNA sequence.
- Use the BLAST algorithm to compare the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database.
- $http://www.incogen.com/public_documents/vibe/details/blastx.html$

A lot of different implementations of BLAST algorithm are available online at:

BLAST web server at NCBI:

$http://www.ncbi.nlm.nih.gov/blast/Blast.cgi$

European Bioinformatics Institute

$http://www.ebi.ac.uk/$

6 Statistics of BLAST scores

We need to be able to estimate statistical significance of the scores obtained as a result from BLAST algorithm. Basically, we need to be able to answer the following question. I obtained a score S from my pair-wise sequence alignment using BLAST. How significant is this score S? Or, the equivalent question - What is the probability of obtaining a score S or better from chance alone? This could be asked under certain conditions: When I align a pair of biological but non-homologous sequences, When I align a pair of shuffled sequences that preserve compositional properties of biological sequences and When I align a pair of sequences that have been computationally generated, based upon a statistical model of DNA or protein sequences.

One way to answer these questions is to do empirical simulations.

- Generate many random sequence pairs of the appropriate length and composition.
- Calculate the optimal alignment score for each pair using a specific scoring scheme.
- If 100 random alignments have score inferior to the alignment of interest, the P-value in question is likely less than 0.01.

- However one must take into account multiple testing in database searching. When many alignments are generated between the query sequence and the database sequences, the significance of the best must be discounted accordingly. An alignment with P-value 0.0001 in the context of a single trial may be assigned a P-value of only 0.1 if it was selected as the best among 1000 independent trials.

However, doing empirical simulations is not practical because of computational expenses. The more precise results we need the more trails we need to do.

The statistics of global sequence comparison [3]

Unfortunately, under even the simplest random models and scoring systems, very little is known about the random distribution of optimal global alignment scores. Monte Carlo experiments can provide rough distributional results for some specific scoring systems and sequence compositions, but these can not be generalized easily. Therefore, one of the few methods available for assessing the statistical significance of a particular global alignment is to generate many random sequence pairs of the appropriate length and composition, and calculate the optimal alignment score for each. While it is then possible to express the score of interest in terms of standard deviations from the mean, it is a mistake to assume that the relevant distribution is normal and convert this Z-value into a P-value; the tail behavior of global alignment scores is unknown.

The statistics of local sequence comparison [3] Fortunately, statistics for the scores of local alignments, unlike those of global alignments, are well understood. This is particularly true for local alignments lacking gaps, which we will consider first. Such alignments were precisely those sought by the original BLAST database search programs. A local alignment without gaps consists simply of a pair of equal length segments, one from each of the two sequences being compared. A modification of the Smith-Waterman or Sellers algorithms will find all segment pairs whose scores can not be improved by extension or trimming. These are called high-scoring segment pairs or HSPs.

To analyze how high a score is likely to arise by chance, a model of random sequences is needed.

A model for random sequences.

To construct the model we need the following information:

- $\{a_1, a_2, ..., a_r\}$ - Amino acid or nucleotide alphabet(r=20 or 4 respectively).
- $\{p_1, p_2, ..., p_r\}$ and $\{p_1, p_2, ..., p_r\}$. The abundances of the two input sequences.
- Substitution matrix $\{s_{ij}\}_{i,j=1}^{r}$. sij is the similarity score between amino acid types a_i and a_j. It can be the elements of a log-likelihood-ratio scoring matrix (such as PAM and BLOSUM): $s_{ij} = \log(\frac{q_{ij}}{p_i p_j})$

E-Value of score S.

In the limit of sufficiently large sequence lengths m and n, the statistics of HSP(Highest-scoring Segment Pairs) scores are characterized by two parameters, K and λ. Most simply, the expected number of HSPs with score at least S is given by the formula:

$$E = Kmne^{-\lambda S}$$

.

This formula makes eminently intuitive sense. Doubling the length of either sequence should double the number of HSPs attaining a given score. Also, for an HSP to attain the score 2x it must attain the score x twice in a row, so one expects E to decrease exponentially with score. λ and K can be thought of simply as natural scales for the search space size and the scoring system respectively. They can be computed from $\{p_1, p_2, ..., p_r\}, \{p'_1, p'_2, ..., p'_r\}$ and s_{ij}. λ is a solution to the equation

$$1 = \sum_{i=1}^{r} \sum_{j=1}^{r} p_i p_j e^{\lambda s_{ij}}$$

The way to compute can be found in Karlin Altschul (1990), PNAS, 87:2264.

There are natural constraint on scoring system.

Constraints on $\{s_{ij}\}$.

- At least one of the sij is positive. Otherwise you will end up with (on average) 0-length HSP.

- The expected score for aligning a random pair of amino acid is required to be negative. Were this not the case, long alignments would tend to have high score independently of whether the segments aligned were related, and the statistical theory would break down.

$$\sum_{i=1}^{r} p_i p_j s_{ij} < 0$$

- Log-likelihood-ratio scores naturally satisfy the above two constraints. (PAM and BLOSUM matrices)

Bit score

Raw scores have little meaning without detailed knowledge of the scoring system used, or more simply its statistical parameters K and lambda. Unless the scoring system is understood, citing a raw score alone is like citing a distance without specifying feet, meters, or light years. By normalizing a raw score using the formula

$$S' = \frac{\lambda S - \ln K}{\ln 2}.$$

one attains a "bit score" S', which has a standard set of units. The E-value corresponding to a given bit score is simply

$$E = mn2^{-S'}.$$

Bit scores subsume the statistical essence of the scoring system employed, so that to calculate significance one needs to know in addition only the size of the search space.

P-values.

The Number of HSPs with score $\geq S$ is approximately Poisson distributed, with mean as E-value of S.

$$P(HSPs = a) = \frac{e^{-E}E^a}{a!}$$

where E is the E-value of S given by equation above. Specifically the chance of finding zero HSPs with score is e^{-E}, so the probability of finding at least one such HSP is

$$P = 1 - e^{-E}$$

This is the **P-value** associated with the score S. For example, if one expects to find three HSPs with score $\geq S$, the probability of finding at least one is 0.95. The BLAST programs report E-value rather than P-values because it is easier to understand the difference between, for example, E-value of 5 and 10 than P-values of 0.993 and 0.99995. However, when E$\leq$ 0.01, $P-valuesandE-valuearenearlyidentical.$

Database searches [3]

The E-value of equation $E = mn2^{-S'}$ applies to the comparison of two proteins of lengths m and n. How does one assess the significance of an alignment that arises from the comparison of a protein of length m to a database containing many different proteins, of varying lengths? One view is that all proteins in the database are a priori equally likely to be related to the query. This implies that a low E-value for an alignment involving a short database sequence should carry the same weight as a low E-value for an alignment involving a long database sequence. To calculate a "database search" E-value, one simply multiplies the pairwise-comparison E-value by the number of sequences in the database. Recent versions of the FASTA protein comparison programs take this approach. An alternative view is that a query is a priori more likely to be related to a long than to a short sequence, because long sequences are often composed of multiple distinct domains. If we assume the a priori chance of relatedness is proportional to sequence length, then the pairwise E-value involving a database sequence of length n should be multiplied by N/n, where N is the total length of the database in residues. Examining equation $E = mn2^{-S'}$, this can be accomplished simply by treating the database as a single long sequence of length N. The BLAST programs take this approach to calculating database E-value. Notice that for DNA sequence comparisons, the length of database records is largely arbitrary, and therefore this is the only really tenable method for estimating statistical significance.

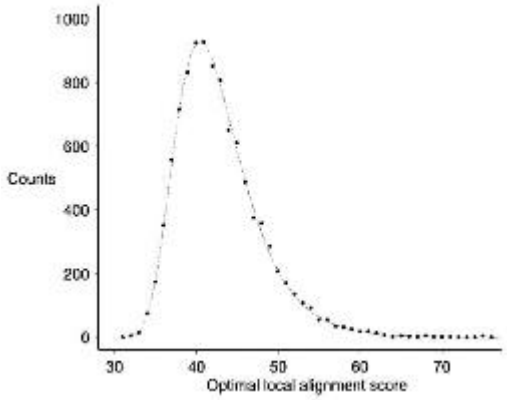
The statistics of gapped alignments [3]

The statistics developed above have a solid theoretical foundation only for local alignments that are not permitted to have gaps. However, many computational experiments and some analytic results

strongly suggest that the same theory applies as well to gapped alignments. For ungapped alignments, the statistical parameters can be calculated, using analytic formulas, from the substitution scores and the background residue frequencies of the sequences being compared. For gapped alignments, these parameters must be estimated from a large-scale comparison of "random" sequences. Some database search programs, such as FASTA or various implementation of the Smith-Waterman algorithm, produce optimal local alignment scores for the comparison of the query sequence to every sequence in the database. Most of these scores involve unrelated sequences, and therefore can be used to estimate λ and K. This approach avoids the artificiality of a random sequence model by employing real sequences, with their attendant internal structure and correlations, but it must face the problem of excluding from the estimation scores from pairs of related sequences. The BLAST programs achieve much of their speed by avoiding the calculation of optimal alignment scores for all but a handful of unrelated sequences. The must therefore rely upon a pre-estimation of the parameters lambda and K, for a selected set of substitution matrices and gap costs. This estimation could be done using real sequences, but has instead relied upon a random sequence model, which appears to yield fairly accurate results.

λ and K for Gapped Local Alignments.

Using BLOSUM-62 amino acid substitution scores and affine gap costs in which a gap of length g is assigned a score of -(10 + g), 10,000 pairs of length-1000 random protein sequences are generated and the Smith-Waterman algorithm is used to calculate 10,000 optimal local alignment scores. From these scores, l is estimated at 0.252 and K at 0.035. A plot of local alignment scores and the fitted EVD curve is shown on figure.



7 References

[1] Lecture slides

[2] *Basic Local Alignment Search Tool*, Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers and David J. Lipmanl. http://www.cs.umd.edu/ wu/paper/blast$_r e f$.*pdf*

[3]http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html