PAM matrix for BLAST algorithm

Churbanov Alexander

April 11,2002

Abstract

In this paper we will consider math background for PAM (*Point Accepted Mutation*). PAM matrix is used extensively in BLAST (*Basic Local Alignment Search Tool*) algorithm, which is extremely fast, robust and popular heuristic. The paper gives an intuition behind the the matrix as well the real life examples to clarify the theory. Short introduction into BLAST algorithm concludes the paper.

Introduction

With the advent of fast and reliable technology for sequencing nucleic acids and proteins, centralized databases were created to store the large quantity of sequence data produced by labs all over the world. This created a need for efficient programs to be used in queries of these databases. In a typical application, one has a *query* sequence that must be compared to all those already in the database, in search of local similarities. This means hundreds of thousands of sequence comparisons.

The quadratic complexity of exact methods, such as dynamic programming, for computing similarities and optimal alignments between two sequences makes them unsuitable for searching large databases. To speed the search, novel and faster methods have been developed. In general, these methods are based on heuristics and it is hard to establish their theoretical time and space complexity. Nevertheless, the programs based on them have become very important tools and these techniques deserve very careful study.

In this poster we introduce basics of BLAST search algorithm. BLAST heuristics is one of the most popular nowadays. In order to understand BLAST, we introduce PAM scoring matrix for amino acids, which is very important in database search and in protein sequence comparison in general.

1 PAM

Amino acids, the residues that make up protein sequence, have biochemical properties that influence their relative replaceability in an evolutionary scenario. For instance, it is more likely that amino acids of similar sizes get substituted for one another than those of different sizes.

The acronym PAM stands for *Point Accepted Mutation*. PAM is a substitution of one amino acid of a protein by another that is "accepted" by evolution, in the sense that within some given species, the mutation has not only arisen but has, over time, spread to essentially the entire species. A PAM1 transition matrix is the *Markov chain matrix* applying for a time period over which we expect 1% of the amino acids to undergo accepted point mutations within the species of interest.

Definition 1 *An* accepted mutation *is a mutation that occurred and was positively selected by the environment; that is, it did not cause the demise of the particular organism where it occurred.*

It is important for the basic 1-PAM matrix that we consider immediate mutations, $a \to b$, not mediated ones like $a \to c \to b$

The necessary ingredients to build the 1-PAM matrix M are the following:

- A list of accepted mutations
- The probability of occurrence p_a for each amino acid a

The probabilities of occurrence can be estimated simply by computing the relative frequency of occurrence of amino acids over a large, sufficiently varied protein sequence set. These numbers satisfy

$$\sum_a p_a = 1$$

From the list of accepted mutations we can compute the quantities f_{ab}, the number of times the mutation $a \leftrightarrow b$ was observed to occur. Recall that we are dealing with undirected mutations here, so $f_{ab} = f_{ba}$. We will also need the sums

$$f_a = \sum_{b \neq a} f_{ab}$$

the total number of mutations in which a was involved, and

$$f = \sum_a f_a$$

the total number of amino acid occurrences involved in mutations. The number f is also twice the total number of mutations.

PAM is 20×20 matrix with M_{ab} being the probability of amino acid a changing into amino acid b. M_{aa} is probability to be unchanged for certain amino acid a during the evolutionary interval.

Relative mutability of amino acid a defined as

$$m_a = \frac{f_a}{100\, f\, p_a}$$

Example of Calculating of mutability

Aligned	A	D	A
Sequences	A	D	B
Amino Acids	A	B	D
Observed Changes	1	1	0
Frequency of Occurence	3	1	2
(Total Composition)			
Relative Mutability	.33	1	0

Figure 1: Sample computation of relative mutability

Mutabilities are scaled to the number of replacements per occurrence of the given amino acid per 100 residues in each alignment. Mutabilities for real homologous proteins are presented in Table. 1.

Relative mutability is the probability that the given amino acid will change in the evolutionary period of interest.

Hence, the probability of a remaining unchanged is the complementary probability

$$M_{aa} = 1 - m_a$$

On the other hand, the probability of a changing into b can be computed as the product of the conditional probability that a will change into b, given that

Ala	A	0.096
Arg	R	0.034
Asn	N	0.042
Asp	D	0.053
Cys	C	0.025
Gln	Q	0.032
Glu	E	0.053
Gly	G	0.090
His	H	0.034
Ile	I	0.035
Leu	L	0.085
Lys	K	0.085
Met	M	0.012
Phe	F	0.045
Pro	P	0.041
Ser	S	0.057
Thr	T	0.062
Trp	W	0.012
Tyr	Y	0.030
Val	V	0.078

Table 1: Relative Mutabilities (Dayhoff)

a changed, times the probability of a changing

$$\begin{aligned} M_{ab} &= P(a \to b) \\ &= P(a \to b | \, a \text{ changed}) P(a \text{ changed}) \\ &= \frac{f_{ab}}{f_a} m_a \end{aligned}$$

The independence from past history in particular leads to a Markov-type model of evolution, which has good mathematical properties.

It is easy to verify that M has the following properties

$$\sum_b M_{ab} = 1 \tag{1}$$

$$\sum_a p_a M_{aa} = 0.99 \tag{2}$$

Equation (1) is merely saying that by adding up the probability of a staying the same and probabilities of it changing to every other amino acid we get 1.

The transition probability matrix has been normalized to reflect the fact that the amount of evolution will change 1 out of 100 amino acids on average. We can see this fact from Equation 2). Sample matrix M is shown in Table 2.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	**9867**	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R	1	**9913**	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N	4	1	**9822**	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D	6	0	42	**9859**	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C	1	1	0	0	**9973**	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Q	3	9	4	5	0	**9876**	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E	10	0	7	56	0	35	**9865**	4	2	3	1	4	1	0	3	4	2	0	1	2
G	21	1	12	11	1	3	7	**9935**	1	0	1	2	1	1	3	21	3	0	0	5
H	1	8	18	3	1	20	1	0	**9912**	0	1	1	0	2	3	1	1	1	4	1
I	2	2	3	1	2	1	2	0	0	**9872**	9	2	12	7	0	1	7	0	1	33
L	3	1	3	0	0	6	1	1	4	22	**9947**	2	45	13	3	1	3	4	2	15
K	2	37	25	6	0	12	7	2	2	4	1	**9926**	20	0	3	8	11	0	1	1
M	1	1	0	0	0	2	0	0	0	5	8	4	**9874**	1	0	1	2	0	0	4
F	1	1	1	0	0	0	0	1	2	8	6	0	4	**9946**	0	2	1	3	28	0
P	13	5	2	1	1	8	3	2	5	1	2	2	1	1	**9926**	12	4	0	0	2
S	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	**9840**	38	5	2	2
T	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	**9871**	0	2	9
W	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	**9976**	1	0
Y	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	**9945**	1
V	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	**9901**

Table 2: M matrix

Once we have the basic matrix M we can derive transition probabilities for larger amounts of evolution. M^k is the transition probability matrix for period of k units of evolution. Sample matrix M^{250} is shown in Table 3.

We are now ready to define the scoring matrixs. The entries in these matrixs are related to the ratio between two probabilities, namely, the probability that a pair of mutations as opposed to being a random occurrence. This is called *likelihood* or *odds* ratio $\frac{M_{ab}}{p_b}$.

Each entry in *lod* (*logarithm of odds*) matrix S is calculated

$$S_k ab = 10 \log_{10} \frac{M_{ab}^k}{p_b}$$

Sample S^{250} matrix is shown in Table 4.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
V	7	4	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	72	4	17

Table 3: M^{250} matrix

2 BLAST

The BLAST programs are among the most frequently used to search sequence databases worldwide. BLAST returns a list of *high-scoring segment pairs* between the query sequence and sequence in the database.

BLAST finds certain "seeds", which are very short segment pairs between the query and the database sequence. These seeds are then extended in both directions, without including gaps, until the maximum possible score for extensions of this particular seed is reached. Not all extensions are looked at. The program has a criterion to stop extensions when the score falls below a carefully computed limit. There is a very small chance of the right extension not being found due to this time optimization, but in practice this tradeoff is highly acceptable.

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

Table 4: PAM 250 logarithm of odds matrix (S^{250} matrix)

BLAST undertakes the following steps

1. Compile list of high-scoring strings (or words, in BLAST jargon)
2. Search for hits - each hit gives a seed
3. Extend seeds

For protein sequence, the list of high-scoring words consists of of all words with w characters (called w-mers) that score at least T with some w-mer of the query. The recommended value for w, the seed size, is 4 for protein searches.

K	A	L	M	R		
V	A	K	N	S		
−4	3	−4	−3	−1	→	Total:−9

Figure 2: Segment pair and its score under PAM120

For DNA searches, the initial list contains only the query w-mers (usually of length 11). Because scoring of DNA sequences is easier, this is enough for

all practical purposes. The scanning strategy is radically different from protein case. Taking advantage of the fact that the alphabet size is 4, the database is first compressed so that each nucleotide is represented using 2 bits. Four nucleotides fit in a byte, so we can compare bytes in our search.

The *extension* is based on well-founded statistical theory that gives exact distribution of gapless local maximum score for random sequences, and permits a very accurate computation of the probability that the segment pair found could be possible due to chance alone. The smaller the probability, he more significant is the match.

The distribution of *Maximum segment pair* (pair with the maximum score) for random sequences s and t of lengths m and n, respectively, can be accurately approximated as described next. The approximation gets better as m and n increase.

Given matrix of replacement costs s_{ij} for the pairs of characters in the alphabet, and probability p_i of occurrence of each aminoacid in the sequence, we first compute value λ solving the equation

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

The parameter λ is the unique positive solution to the equation can be obtained by Newton's method. Once λ is known, the expected number of distinct segment pairs between s and t with score above S is

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

where K is calculatable constant. Actually, the distribution of the number of segment pairs scoring above S is a Poisson distribution with mean given by the previous formula. From this, it is easy to derive expressions for useful quantities like the average score, intervals where the score will fall 90% of the time, and so on.

References

[1] Joao Setubal and Joao Meidanis, *Introduction to computational molecular biology*, University of Campinas, Brazil, December 1997.

[2] Warren J. Ewens and Gregory R. Grant, *Statistical methods in bioinformatics: an introduction*, Springer-Verlag New York, 2001.