BLAST 検索の中身について

清水顕史

BLAST 検索は非常に強力なツールであるため、生物学研究の様々な場面で利用されています。通常はデフォルトのオプション設定で使用しますが、ここでは類似配列探索の仕組みとして以下の 1)および 2)をみてみましょう。

1) スコア行列とギャップペナルティ
2) 動的計画法(ダイナミックプログラミング)

1) スコア行列とギャップペナルティ

BLAST 検索の元になる、配列対アライメント(整列化)の類似性評価法をみてみましょう。まず例 1 として、SHIGAKEN と CHIBAKEN というアミノ酸配列対の類似性をみてみましょう。アミノ酸配列を比較して類似性を比較する際、同位置のアミノ酸は進化の過程で他のアミノ酸に置換することがある、と考えます。例 1 の場合、同じ 8 残基なのでそのままアライメントします(下図)。

```
1 2 3 4 5 6 7 8
S H I G A K E N
C H I B A K E N
```

75%のアミノ酸が一致しています。この二つのアミノ酸配列が同一祖先配列に由来する時、位置 2,3 と位置 5,6,7,8 のアミノ酸は同祖配列から置換(変異)せず保存されてきたと考えることができます。また位置 1,4 にはアミノ酸置換が起こったと考えます。

アミノ酸配列の類似度(BLAST 検索のスコア)は、アミノ酸の保存や置換の割合を起こりやすさに応じて点数化した合計です。各スコアの計算にはスコア行列が用いられます。スコア行列とは、多くの類縁タンパク質からアミノ酸毎の置換頻度を見積もりまとめたもので、アミノ酸対が A と A であるスコア、A と B であるスコアのように総当りペアの点数表です。置換頻度の見積もり方に応じて様々なスコア行列が存在しますが、ここではその中の一つ BLOSUM62 をみてみましょう。このスコア行列はデフォルトの BLAST 検索で用いられるもので、多様なアミノ酸配列を比較する際に一番よいスコア行列であることが経験的に知られています。

BLOSUM62

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

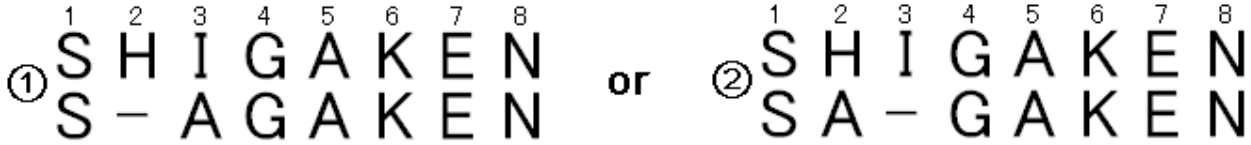
この表は、NCBIの提供するstand-alone BLAST(2002Nov19)に付属のBLOSUM62を転載したものです。
BはDまたはN、ZはEまたはQ、Xは不定のアミノ酸を示す。

対称行列になった表の、左端列と上端行のアミノ酸の組合せからアミノ酸ペアのスコアを読みます。例えば、アミノ酸対 N と C のスコアは、左端列が N の行と上端行が C の列から-3 と読みます。アライメントのスコアは各アミノ酸対のスコアの和を計算します。よって SHIGAKEN と CHIBAKEN のアライメントのスコアは 30 となります。

1 2 3 4 5 6 7 8
S H I G A K E N
C H I B A K E N
-1 8 4 -1 4 5 5 6 計 30

スコアはアミノ酸が置換する確率を表す値ですが、比の対数をとっているため、加算すると各ペアの起こりうる確率を表すことになります。

例 2 として、SHIGAKEN と SAGAKEN というアミノ酸配列の類似性をみてみましょう。今度は 8 残基と 7 残基と配列間の長さが異なりますのでアライメントには工夫が必要です。同一祖先から由来した配列に 1 残基の長さの違いがある場合、1 つのアミノ酸の挿入または欠失が起こったと考えます。これをギャップといい、”-“で表します。例 2 の場合は SAGAKEN のどこかに一つのギャップを入れてアライメントをとることになります。一致するアミノ酸対は同一祖先から保存されたものと考えると、ギャップの入りうる箇所は二箇所で、下の①か②のいずれかになるでしょう。



2 つのアライメントのどちらがよりよい(確率的に起こり易い)かは、それぞれのスコアを比較すれば分かります。今、ギャップを無視して BLOSUM62 による各スコアの合計を計算すると①は 29 に②は 28 になります。つまり①がよりよいアライメントになります。

アミノ酸配列 SHIGAKEN は CHIBAKEN と SAGAKEN のどちらにより近縁であるといえるでしょうか？これは各対間のスコアを比較すればよいのですが、例 2 のようにギャップがある場合はギャップペナルティも考慮する必要があります。ギャップはアミノ酸の挿入や欠失などの変異によるものなので、ギャップの程度に合わせて(変異が起こって類似性が減った分を勘案して)スコアを減らす必要があります。
BLAST 検索で用いられるギャップペナルティ d は”G + E×(ギャップ長-1)”です。G はギャップ開始コスト、E はギャップ伸長コストで、デフォルトではそれぞれ 11 と 1 に設定されています。例 2 のアライメントのコストはつまり、18 になります。よって SHIGAKEN により近縁な配列は CHIBAKEN になります。

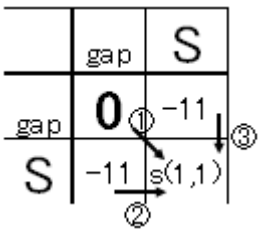
2) 動的計画法(ダイナミックプログラミング)

これまで、アミノ酸配列対のアライメントは(比較的良く似た配列なので)直感的に決められました。本来は全てのアライメント可能な配列から最高スコアのアライメントを探索する手続きを用いるのが理想的です。動的計画法(ダイナミックプログラミング)という計算アルゴリズムによって最適アライメントが解けることが分かっています。動的計画法の応用の中でも、大域的アライメントと呼ばれる方法をみてみましょう。まず、アライメントをとる 2 つの配列を上端行方向と左端列方向に配置した格子を考えます。

	gap	S	H	I	G	A
gap	**0**	1gap	2gaps	3gaps	4gaps	5gaps
S	1gap					
A	2gaps					
G	3gaps					
A	4gaps					

それぞれの行と列の先頭にギャップ用の列と行が挿入されていることに注意してください。gap 列の要素は上から 0,1gap, 2gaps～7gaps の 8 つで、gap 行の要素は左から 0,1gap, 2gaps～8gaps からなる 9 つです。gap(s)の要素にはギャップペナルティで計算した負の値が入ることになります。

0 を始点とした移動経路における各格子のスコアを考えるのですが、或る格子への経路は 3 種類の隣の格子(①左斜め上、②左、③上)からのみ(右図の矢印)であり、その中で最大のものを選ぶことにします。



最大スコアの計算方法を式で表すと、

$$s(i,j)=\max\begin{cases} s(i-1,j-1)+Score(X_i,Y_j) & \cdots① \\ s(i-1,j)-d & \cdots② \\ s(i,j-1)-d & \cdots③ \end{cases}$$

となります(手計算できるようにするため、ここでのギャップペナルティ d は-11×ギャップ長とします)。s(1,1)の場合は①s(0,0)+ Score(S,S)= 4、②s(0,1)-d= -11-d、③s(1,0)-d=-11-d なので経路①のスコア 4 が選ばれます。同様に s(1,2)は①s(0,1)+Score(S,A)=-10、②s(0,2)-d=-33、③ s (1,1)-d=-7 なので経路③のスコア-7 が選ばれます。s(2,1)は①s(1,0)+Score(S,H)=-12、②s(1,1)-d=-7、③s(2,0)-d=-33 なので経

路②のスコア-7 が選ばれます。

	gap	S	H
gap	0	-11	-22
S	-11	4	-7
A	-22	-7	2

各格子のスコアはその①,②,③の隣の格子のスコアから計算できるので、全格子のスコアを計算します。その際、その格子を得た経路も書き記しておきます。つまり、SHIGA と SAGA のアライメントは以下の様に探索できます。

	gap	S	H	I	G	A
gap	0	-11	-22	-33	-44	-55
S	-11	4	-7	-18	-29	-40
A	-22	-7	2	-8	-18	-25
G	-33	-18	-9	-2	-2	-13
A	-44	-29	-20	-10	-2	2

s(1,4) ①-33+1=-32、②-44-11、③-18-11

s(4,1) ①-33+0、②-18-11、③-44-11

s(2,4) ①-18-2、②-29-11、③-9-11

s(3,4) ①-9-1、②-20-11、③-2-11

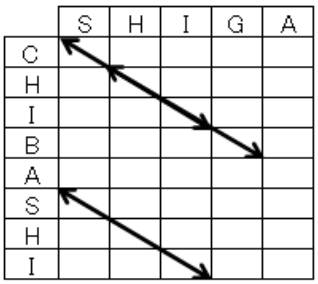
s(4,2) ①-18+0、②-8-11、③－29-11
s(5,1) ①-44-1、②-29-11、③-55-11
s(5,2) ①-29+4、②-18-11、③-40-11
s(4,3) ①-8+6、②-2-11、③-25-11
s(4,4) ①-2+0、②-10-11、③-2-11
s(5,3) ①-18+0、②-2-11、③-25-11
s(5,4) ①-2+4、②-2-11、③-13-11

全ての格子スコアを埋めることができたら、右下端から 0 までの経路の”さかのぼり”によって最良アライメントが得られます。上へさかのぼりは上端行に配置したアミノ酸配列にギャップを挿入すること、左へのさかのぼりは左端列に配置したアミノ酸配列にギャップを挿入することに相当します。つまり、SHIGA に対する SAGA のアライメントでは、2 個目のアミノ酸にギャップが入る S-AGA が採用されることになります。

以上が、大域的アライメントといって長さの似た配列対の最適スコア計算に適した方法です。

実際の BLAST 検索では、部分的なアミノ酸類似性を検出するための局所的アライメントが用いられています。局所的アライメントも基本的操作手順は大域的アライメントと同様です。また、更に高速な検索を実現するため、BLAST ではクエリーの配列を指定したワード長に分割します(Word size オプションで設定、デフォルトは 3)。例えば SHIGA というクエリーの場合は、SHI, HIG, IGA に分割しそれぞれについてデータベース配列中のワードとのスコアを計算します。例えばクエリー側の SHI に対して、デ

ータベース側のワードが SHI 時のスコアは 16、SHA の時は 11、SGI の時は 6 のように各ワードに対するスコアを計算することができます。このうち閾値(デフォルトで 11)以上のワードを一致可能とみなしアライメントを配列の上下に沿って拡張します(下図例、一致したワードを←→で示す)。



ワード対	スコア
SHI と CHI	11
SHI と HIB	-7
SHI と IBA	-3
SHI と BAS	-4
SHI と ASH	-3
SHI と SHI	16

ワード対	スコア
HIG と CHI	-10
HIG と HIB	11
HIG と IBA	-6
HIG と BAS	-1
HIG と ASH	-6
HIG と SHI	-8

ワード対	スコア
IGA と CHI	-4
IGA と HIB	-9
IGA と IBA	7
IGA と BAS	-2
IGA と ASH	-3
IGA と SHI	-7

この方法はダイナミックプログラミングによる方法と異なり、最適なアライメントが得られる保証がなく、準最適なアライメントが得られます。しかし、膨大なデータベースから高速検索が可能になります。

BLAST 検索の仕組みについて解説してあるより詳しい文献のうち、滋賀県立大学の図書館で手に入る本としては、

『バイオインフォマティクス・ゲノム配列から機能解析へ』(MEDSi)

があります。