第3章

序列比对

比较是科学研究中最常见的研究方法之一，通过将研究对象进行相互比较，以寻找研究对象可能具备的某些特征和特性。在生物信息学研究中，序列比对就是对生物分子序列进行比较，它通过对两个或多个核苷酸或氨基酸序列按照一定的规律排列起来，逐列比较其字符的异同，判断它们之间的相似程度和同源性，从而推测它们的结构、功能以及进化上的联系。

序列比对是生物信息学中最基本、最重要的操作之一，它的理论基础是进化学说，即如果两个序列之间具有足够高的相似性，那么二者可能是由共同的进化祖先经过序列内残基的替换、残基或序列片段的缺失或插入以及序列重组等遗传变异过程分别演化而来。因此，通过序列比对可以发现生物序列中的功能、结构和进化的信息。序列比对的任务就是通过比较生物分子序列，发现它们的相似性，找出序列之间的共同区域，同时辨别序列之间的差异。

在分子生物学中，不同核酸分子（或蛋白质分子）的相似性包括多方面的含义，可能是分子序列之间的相似，可能是分子空间结构的相似，也可能是分子功能的相似。对于生物大分子，尤其是蛋白质分子，一个普遍的规律是分子的序列决定其空间结构，而这种折叠结构决定它的功能。研究序列相似性的目的之一就是通过比较未知序列和已知序列的相似性来预测未知序列的结构和功能；研究序列相似性的另一个目的是通过序列的相似性，推断序列之间的同源性，推测序列之间的进化关系。

3.1 序列比对基础

3.1.1 序列比对的分类

根据同时进行比对的序列数目的不同，序列比对分为双序列比对（pairwise alignment）和多序列比对（multiple sequence alignment）。两条序列的比对称为双序列比对；三条或以上序列的比对称为多序列比对。

序列比对如果从比对范围考虑也可分为全局比对(global alignment)和局部比对(local alignment)。全局比对是从全长序列出发，考察序列之间的整体相似性；而局部比对则着眼于序列中的某些特殊片断，比较这些片断之间的相似性。局部相似性比对的生物学基础是蛋白质功能位点往往是由较短的序列片段组成的，尽管在序列的其他部位可能有插入、删除或突变，但这些功能位点的序列具有相当大的保守性，而应用局部比对的方法可以发现不同序列中的这些保守序列，其结果更具有生物学意义。

3.1.2 序列的相似性

序列相似(similarity)和序列同源(homology)是两个完全不同的概念。序列之间的相似可以用一个数值来表示，即序列比对结果中序列之间相同核苷酸或氨基酸所占比例的大小；而同源序列是指从某一共同祖先经过趋异进化而形成的不同序列。当两条序列同源时，它们的氨基酸或核苷酸序列通常有显著的一致性(identity)。如果两条序列有一个共同的进化祖先，那么它们是同源的。两条序列要么是同源的，要么是不同源的，不存在同源性(homology)的程度问题。在实际应用中，可以根据序列的相似程度来推断比对序列是否具有同源性。

序列比对是对序列相似性的描述，它反映了比对序列在什么部位相似，或在什么部位存在差异。序列比对的结果根据比对序列的条数和每条序列的长度会有许多种，其中有一种或几种结果能够揭示序列的最大相似程度，指出序列之间的根本差异，这个比对结果被称为最优比对。序列比对的目的就是应用不同的算法，从众多的比对结果中找出最优比对，在此基础上对比对序列进行相应的生物信息学分析。

1. 字母表和序列

在生物分子信息处理过程中，将生物分子序列抽象为字符串，其中的字符取自特定的字母表。字母表是一组符号或字符，字母表中的元素组成序列。

(1) 一些重要的字母表

4 字符 DNA 字母表{A，C，G，T}；

扩展的遗传学字母表或 IUPAC 编码，见表 3.1；

表 3.1 扩展的遗传学字母表(IUPAC 编码)

含义		说明
G	G	Guanine
A	A	Adenine
T	T	Thymine
C	C	Cytosine
R	G or A	Purine
Y	T or C	Pyrimidine
M	A or C	Amino
K	G or T	Keto
S	G or C	Strong interaction (3 H bonds)
W	A or T	Weak interaction (2 H bonds)
H	A or C or T	Not-G

续表

含　义		说　明
B	G or T or C	Not-A
V	G or C or A	Not-T(Not-U)
D	G or A or T	Not-C
N	G or A or T or C	Any

单字母氨基酸编码，见表 3.2。

表 3.2　20 种标准氨基酸的英文简写

氨基酸名称	英文缩写	简　写	氨基酸名称	英文缩写	简　写
甘氨酸	Gly	G	丝氨酸	Ser	S
丙氨酸	Ala	A	苏氨酸	Thr	T
缬氨酸	Val	V	天冬酰胺	Asn	N
异亮氨酸	Ile	I	谷酰胺	Gln	Q
亮氨酸	Leu	L	酪氨酸	Tyr	Y
苯丙氨酸	Phe	F	组氨酸	His	H
脯氨酸	Pro	P	天冬氨酸	Asp	D
甲硫氨酸	Met	M	谷氨酸	Glu	E
色氨酸	Trp	W	赖氨酸	Lys	K
半胱氨酸	Cys	C	精氨酸	Arg	R

(2) 序列的表示方法

为了说明序列 s 的子序列和 s 中单个字符，我们在序列 s 中各字符之间用数字标明分割边界。

例如，设 s=ACCACGTA，则 s 可表示为 $_0A_1C_2C_3A_4C_5G_6T_7A_8$。其中，$_i:s:_j$ 表示序列 s 中的第 i 位和第 j 位之间的子序列，$0\leqslant i\leqslant j\leqslant|s|$；子序列 $_0:s:_i$ 称为前缀，即 prefix(s,i)；而子序列 $_i:s:_{|s|}$ 称为后缀，即 suffix(s, $|s|-i+1$)；当 $i=j$ 时，$_i:s:_j$ 表示空序列；当 $i=j-1$ 时，$_{(j-1)}:s:_j$ 表示序列 s 中的第 j 个字符，记为 s_j。

两条序列 s 和 t 的连接用 s ++ t 来表示，如

$$\text{ACC++CTA} = \text{ACCCTA}$$

字符串操作除连接操作之外，还有一个删除操作(k 操作)，即删除一个字符串中的某些字符。有三种情况：

删除后缀得到前缀，prefix(s,l) = $sk^{|s|-l}$，其中 l 为保留下的字符个数；

删除前缀得到后缀，suffix(s,l) = $k^{|s|-l}s$ ，其中 l 为保留下的字符个数；

删除前缀和后缀得到中间序列，$_i:s:_j=k^{i-1}sk^{|s|-j}$。

2. 编辑距离(edit distance)

量化两条序列的相似程度有两种方法，一种为相似度，它是两条序列的函数，其值越大，表示两条序列越相似；另一种是两条序列之间的距离，距离越大，则两条序列的相似度就越小。在大多数情况下，相似度和距离可以交互使用，并且距离越大，相似度越小，反之亦然。

怎样计算两条序列间的距离呢？

观察这样两条核酸序列：s=GCATGACGAATCAG，t=TATGACAAACAGCA。一眼看上去，这两条序列并没有什么相似之处，见图 3.1(a)；如果将序列 t 右移一位并对比排列起来以后就可以发现它们的相似性，见图 3.1(b)；如果在序列 t 中的第 6 个字符 C 和第 7 个字符 A 之间加上一空位，就会发现原来这两条序列有很多相似之处，见图 3.1(c)。

```
s: GCATGACGAATCAG      s: GCATGACGAATCAG-      s: GCATGACGAATCAG--
                                                  |||||  ||  |||
t: TATGACAAACAGCA      t: -TATGACAAACAGCA      t: -TATGAC-AAACAGCA
       (a)                    (b)                     (c)
```

图 3.1　序列 s 和序列 t 的 3 种比对结果

图 3.1 只是两条序列相似性的一种定性表示方法，为了说明两条序列的相似程度，还需要定量计算。

最简单的定量计算就是海明(Hamming)距离。对于两条长度相等的序列，海明距离等于对应位字符不同的个数。例如图 3.1(a)比对的海明距离为 13。

使用海明距离来计算序列相似程度不够灵活，这是因为序列可能具有不同的长度，两条序列中各位置上的字符并不一定是真正的对应关系。例如，在 DNA 复制的过程中，可能会发生像删除或插入一个碱基这样的错误，虽然两条序列的其他部分相同，但由于位置的移动导致海明距离的失真。就上述例子来看，图 3.1(a)比对的海明距离为 13，看不出两条序列有什么相似之处，但是，如果在两条序列中做相应的处理，就可以看出这两条序列还是有很多的相似之处，见图 3.1(b)和图 3.1(c)。实际上，在许多情况下，直接运用海明距离来衡量两条序列的相似程度是不合理的。

为了解决字符插入和删除问题，引入字符“编辑操作”(edit operation)的概念，通过编辑操作将一个序列转化为一个新序列。用一个新的字符“－”代表空位(space)，并定义下述字符编辑操作：

(1) match(a,a)— 字符匹配；

(2) delete(a,-)— 从第一条序列删除一个字符，或在第二条序列相应的位置插入空白字符；

(3) replace(a,b)—以第二条序列中的字符 b 替换第一条序列中的字符 a，a≠b；

(4) insert(-,b)—在第一条序列插入空位字符，或删除第二条序列中的对应字符 b。

很显然，在比较两条序列 s 和 t 时，在 s 中的一个删除操作等价于在 t 中对应位置上的一个插入操作，反之亦然。需要注意的是，两个空位字符不能匹配，因为这样的操作没有意义。引入上述编辑操作后，重新计算两条序列的距离，就称为编辑距离。

以上的编辑操作仅仅是关于序列的常用操作，在实际应用中还可以引入复杂的序列操作。图 3.2(a)是两条序列的一种比对，如果将第二条序列头尾倒置，可以发现两条序列惊人地相似，见图 3.2(b)；再比如，图 3.2(c)中两条序列有什么关系呢？如果将其中一条序列中的碱基替换为其互补碱基，就会发现其中的关系。

```
ACCGACAATATGCATA      ACCGACAATATGCATA      CTAGTCGAGGCAATCT
|   |   |   |   |     ||  |||||||||  |||
ATAGGTATAACAGTCA      ACTGACAATATGGATA      GAACAGCTTCGTTAGT
       (a)                   (b)                   (c)
```

图 3.2　复杂的序列操作举例

综上所述，序列比对就是对序列进行编辑操作，通过字符匹配和替换，或者插入和删除字符，使比对序列中相同的字符尽可能地一一对应，然后计算序列之间的编辑距离，判断序列之间的相似程度。

计算两条序列之间的编辑距离，实际上就是根据某一规则计算比对序列各个位置上字母的比对得分，然后将这些位置上的得分累加得到整条序列的比对得分，并用这个分值表示两条序列的相似性。计算得分的规则可以是得分函数，如式(3.1)，也可以是代价函数，如式(3.2)。

得分(score)函数

$$\begin{cases} p(a,a) = 1 \\ p(a,b) = 0 \quad (a \neq b) \\ p(a,-) = p(-,b) = -1 \end{cases} \tag{3.1}$$

代价(cost)函数

$$\begin{cases} w(a,a) = 0 \\ w(a,b) = 1 \quad (a \neq b) \\ w(a,-) = w(-,b) = 1 \end{cases} \tag{3.2}$$

在实际应用中可以根据实际情况定义得分函数和代价函数中的分值。

在进行序列比对分析时，可根据实际情况选用得分函数或代价函数。如果用得分函数评价比对序列，得分的分值越大，两条序列的相似性就越大；如果用代价函数评价比对序列，得分的分值越大，两条序列的相似性就越小。

图 3.3 是序列 s 和 t 的两种比对结果，分别用得分函数和代价函数评价这两种比对结果，计算结果见表 3.3。从计算结果可以看出，比对(a)要优于比对(b)。

```
s: AGCACAC-A        AG-CACACA
   |  |||||  |      |   |||  |
t: A-CACACTA        ACACACT-A
     (a)                (b)
```

图 3.3　序列 s 和序列 t 的两种比对结果

表 3.3　图 3.3 所示两种比对结果分别使用得分函数和代价函数计算的得分

	序列 s 和 t 比对(a)	序列 s 和 t 比对(b)
使用得分函数 p	5	3
使用代价函数 w	2	4

从上述的例子中可以看到：两条序列 s 和 t 的比对的得分(或代价)等于将 s 转化为 t 所用的所有编辑操作的得分(或代价)总和；序列 s 和 t 的最优比对是所有可能的比对中得分最高(或代价最小)的一个比对；序列 s 和 t 的真实距离应该是最优比对时的距离。

3.1.3　序列比对的打分矩阵

式(3.1)和式(3.2)都是简单相似性评价模型，在计算比对的代价或得分时，对字符替换操作只进行统一的处理，没有考虑“同类字符”替换与“非同类字符”替换的差别。实际上，不同类型的字符替换，其代价或得分是不一样的。对于核酸序列，嘌呤和嘧啶间替换的代价要大于嘌呤间或嘧啶间替换的代价；而对于蛋白质序列，某些氨基酸可以很容易地相互取代而不用改变它们的理化性质。例如，两条蛋白质序列比对，其中一条在某一位置上是丙氨

酸，如果该位点被替换成另一个较小且疏水的氨基酸，比如缬氨酸，那么对蛋白质功能的影响可能较小；如果被替换成较大且带电的残基，比如赖氨酸，那么对蛋白功能的影响可能就要比前者大。直观地讲，比较保守的替换比起较随机替换更可能维持蛋白质的功能，且更不容易被淘汰。因此，在为比对打分时，人们可能更倾向对丙氨酸与缬氨酸的比对位点多些“奖励”，而对于丙氨酸与那些大而带电氨基酸的比对位点则相反。也就是说，理化性质相近的氨基酸残基之间替换的代价显然应该比理化性质相差甚远的氨基酸残基替换得分高，或者代价小，同样，保守的氨基酸替换得分应该高于非保守的氨基酸替换。

基于以上原因提出了打分矩阵的概念。在打分矩阵中，详细地列出各种字符替换的得分，从而使得计算序列之间的相似度更为合理。在比对蛋白质序列时，可以用打分矩阵来增强序列比对的敏感性。打分矩阵是序列比对的基础，选择不同的打分矩阵将得到不同的比对结果，而了解打分矩阵的理论依据将有助于在实际应用中选择合适的打分矩阵。

1. 核酸序列比对的打分矩阵

核酸序列所用的字母表为：{ A,C,G,T }。

(1) 等价矩阵

等价矩阵是最简单的一种打分矩阵，见表 3.4。其中，相同核苷酸匹配的得分为 1，而不同核苷酸的替换得分为 0。

表 3.4 等价矩阵

	A	T	C	G
A	1	0	0	0
T	0	1	0	0
C	0	0	1	0
G	0	0	0	1

(2) BLAST 矩阵

BLAST 是一种通用的核酸序列比对程序，表 3.5 是其打分矩阵。如果被比对的两个核苷酸相同，则得分为+5，反之被比对的两个核苷酸不同，得分为−4。

表 3.5 BLAST 矩阵表

	A	T	C	G
A	5	−4	−4	−4
T	−4	5	−4	−4
C	−4	−4	5	−4
G	−4	−4	−4	5

(3) 转换-颠换矩阵

核酸的碱基按照环结构分为两类，一类是嘌呤(腺嘌呤 A，鸟嘌呤 G)，其分子结构有两个环；另一类是嘧啶(胞嘧啶 C，胸腺嘧啶 T)，其分子结构只有一个环。核酸序列发生碱基突变时，如果碱基替换保持环数不变，则称为转换(transition)，如 A→G，C→T 等；如果碱基替换环数发生了变化，则称为颠换(transversion)，如 A→C，A→T 等。在进化过程中，转换发生的频率远比颠换发生的频率高，而表 3.6 所示的序列比对打分矩阵正好反映了这种情况，其中转换的得分为−1，而颠换的得分为−5，匹配的得分为 1。

表 3.6　转换-颠换矩阵

	A	T	C	G
A	1	−5	−5	−1
T	−5	1	−1	−5
C	−5	−1	1	−5
G	−1	−5	−5	1

2. 蛋白质序列比对的打分矩阵

蛋白质的字母表为表 3.2 中氨基酸的简写字符。

(1) 等价矩阵

$$R_{ij}=\begin{cases}1 & i=j\\ 0 & i\neq j\end{cases} \tag{3.3}$$

式(3.3)中，R_{ij} 代表打分矩阵元素，i、j 分别代表字母表第 i 个字符和第 j 个字符。从式(3.3)可以看出，相同氨基酸匹配得分为 1，而不同氨基酸的替换得分为 0。

(2) 遗传密码矩阵 GCM

GCM(genetic code matrix)矩阵是通过计算一个氨基酸残基转变到另一个氨基酸残基所需的密码子中碱基变化数目而得到的，矩阵元素的值对应于代价。如果变化一个碱基，就可以使一个氨基酸的密码子改变为另一个氨基酸的密码子，则这两个氨基酸的替换代价为 1；如果需要 2 个碱基的改变，则替换代价为 2；以此类推，见表 3.7。

表 3.7　遗传密码矩阵

	A	S	G	L	K	V	T	P	E	D	N	I	Q	R	F	Y	C	H	M	W	Z	B	X
A	0	1	1	2	2	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2
S	1	0	1	1	2	2	1	1	2	2	1	1	2	1	1	1	1	2	2	1	2	2	2
G	1	1	0	2	2	1	2	2	1	1	2	2	2	1	2	2	1	2	2	1	2	2	2
L	2	1	2	0	2	1	2	1	2	2	2	1	1	1	1	2	2	1	1	1	2	2	2
K	2	2	2	2	0	2	1	2	1	2	1	1	1	1	2	2	2	2	1	2	1	2	2
V	1	2	1	1	2	0	2	2	1	1	2	1	2	2	1	2	2	2	1	2	2	2	2
T	1	1	2	2	1	2	0	1	2	2	1	1	2	1	2	2	2	2	1	2	2	2	2
P	1	1	2	1	2	2	1	0	2	2	2	2	1	1	2	2	2	1	2	2	2	2	2
E	1	2	1	2	1	1	2	2	0	1	2	2	1	2	2	2	2	2	2	2	1	2	2
D	1	2	1	2	2	1	2	2	1	0	1	2	2	2	2	1	2	1	2	2	2	1	2
N	2	1	2	2	1	2	1	2	2	1	0	1	2	2	2	1	2	1	2	2	2	1	2
I	2	1	2	1	1	1	1	2	2	2	1	0	2	1	1	2	2	2	1	2	2	2	2
Q	2	2	2	1	1	2	2	1	1	2	2	2	0	1	2	2	2	1	2	2	1	2	2
R	2	1	1	1	1	2	1	1	2	2	2	1	1	0	2	2	1	1	1	1	2	2	2
F	2	1	2	1	2	1	2	2	2	2	2	1	2	2	0	1	1	2	2	2	2	2	2
Y	2	1	2	2	2	2	2	2	2	1	1	2	2	2	1	0	1	1	3	2	2	1	2
C	2	1	1	2	2	2	2	2	2	2	2	2	2	1	1	1	0	2	2	1	2	2	2
H	2	2	2	1	2	2	2	1	2	1	1	2	1	1	2	1	2	0	2	2	2	1	2
M	2	2	2	1	1	1	1	2	2	2	2	1	2	1	2	3	2	2	0	2	2	2	2
W	2	1	1	1	2	2	2	2	2	2	2	2	2	1	2	2	1	2	2	0	2	2	2
Z	2	2	2	2	1	2	2	2	1	2	2	2	1	2	2	2	2	2	2	2	1	2	2
B	2	2	2	2	2	2	2	2	2	1	1	2	2	2	2	1	2	1	2	2	2	1	2
X	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

例 3.1 甲硫氨酸(Met)与酪氨酸(Tyr)之间替换的代价。

解 查遗传密码表可知，Met 的密码子为 AUG，Tyr 的密码子为 UAU，UAC。可以看出，甲硫氨酸与酪氨酸之间的替换是需要密码子的三个碱基都发生改变，因此它们之间的替换代价为 3。

例 3.2 脯氨酸(Pro)与甘氨酸(Gly)之间替换的代价。

解 查遗传密码表可知，Pro 的密码子为 CCU，CCC，CCA，CCG，Gly 的密码子为 GGU，GGC，GGA，GGG。

可以看出，脯氨酸与甘氨酸之间的替换是需要密码子的两个碱基发生改变，因此它们之间的替换代价为 2。

GCM 矩阵常用于进化距离的计算，其优点是计算结果可以直接用于绘制进化树，但是它在蛋白质序列比对尤其是相似程度很低的序列比对中很少被使用。

(3) 疏水矩阵

疏水矩阵是根据氨基酸残基替换前后疏水性的变化而得到的得分矩阵。若某一次氨基酸替换前后疏水特性不发生太大的变化，则这种替换得分高，否则替换得分低。疏水矩阵物理意义明确，有一定的理化性质依据，适用于偏重蛋白质功能方面的序列比对。疏水矩阵见表 3.8。

表 3.8 蛋白质疏水矩阵

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

(4) PAM 矩阵

PAM(Point Accepted Matrix)矩阵是目前蛋白质序列比对中广泛使用的打分方法之一,它是基于氨基酸进化的点接受突变模型(point accepted mutation),统计自然界中各种氨基酸残基的相互替换率而得到的。如果两种特定的氨基酸之间替换发生频繁,说明自然界容易接受这种替换,那么这一对氨基酸在打分矩阵中的互换得分就比较高。

1978 年 Margaret Dayhoff 提出了一个衡量蛋白质进化变化的模型。她和同事们研究了 71 个紧密相关的蛋白质家族的 1572 个突变,结果见表 3.9,表中描述了任一氨基酸对 (i,j) 比对在一起的频率。研究发现蛋白质家族中氨基酸的替换并不是随机的,一些氨基酸的替换比其他替换更容易发生,其主要原因是这些替换不会对蛋白质的结构和功能产生太大的影响。如果氨基酸的替换是随机的,那么,每一种可能的替换频率仅仅取决于不同氨基酸出现的背景频率。然而,在相关蛋白质中,替换频率大大地倾向于那些不影响蛋白质功能的替换,换句话说,这些点突变已经被进化所接受。这意味着,在进化过程中,相关的蛋白质在某些位置上可以出现不同的氨基酸。

表 3.9　可接受点突变数目

		原氨基酸																			
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
替代氨基酸	A																				
	R	30																			
	N	109	17																		
	D	154	0	532																	
	C	33	10	0	0																
	Q	93	120	50	76	0															
	E	266	0	94	831	0	422														
	G	579	10	156	162	10	30	112													
	H	21	103	226	43	10	243	23	10												
	I	66	30	36	13	17	8	35	0	3											
	L	95	17	37	0	0	75	15	17	40	253										
	K	57	477	322	85	0	147	104	60	23	43	39									
	M	29	17	0	0	0	20	7	7	0	57	207	90								
	F	20	7	7	0	0	0	0	17	20	90	167	0	17							
	P	345	67	27	10	10	93	40	49	50	7	43	43	4	7						
	S	772	137	432	98	117	47	86	450	26	20	32	168	20	40	269					
	T	590	20	169	57	10	37	31	50	14	129	52	200	28	10	73	696				
	W	0	27	3	0	0	0	0	0	3	0	13	0	0	10	0	17	0			
	Y	20	3	36	0	30	0	10	0	40	13	23	10	0	260	0	22	23	6		
	V	365	20	13	17	33	27	37	97	30	661	303	17	77	10	50	43	186	0	17	
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V

注:表中数据来自《生物信息学与功能基因组学》。

Margaret Dayhoff 和同事们用这些可接受突变的数据和每个氨基酸的发现频率来产生突变概率矩阵 M(mutation probability matrix),结果见表 3.10。矩阵中元素 m_{ij} 表示在一给定进化时期内氨基酸 j(列)替换成氨基酸 i(行)的概率。在表 3.10 的例子里,进化时期

为一个 PAM。一个 PAM 就是一个进化的变异单位，即 1%的氨基酸改变。但是，这并不意味着经过 100 次 PAM 后，每个氨基酸都发生变化，因为其中一些位置可能会经过多次改变，甚至可能变回到原先的氨基酸。因此，另外一些氨基酸可能不发生改变。

表 3.10　PAM-1 突变概率矩阵

		原氨基酸																			
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
替代氨基酸	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	9856	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	21	7	0	1	7	0	1	23
	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
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V

注：表中数据来自《生物信息学与功能基因组学》。

观察表 3.10 会发现，最高分位于从左上至右下的对角线上，每列值的和为 10000(对应 100%)。表中第一列第 1 个数据 9867 意味着在一个 PAM 的进化期内原序列中的丙氨酸将有 98.67%的概率保持不变，第一列第 16 个数据 28 表示在一个 PAM 的进化期内原序列中的丙氨酸将有 0.28%的概率替换成丝氨酸；最容易发生突变的天冬酰胺仅有 98.22%的概率保持不变，最不容易发生突变的色氨酸有 99.76%的概率保持不变。

PAM-1 矩阵基于紧密相关蛋白质序列的比对，这些蛋白质家族内的序列相似程度大于 85% 。但通常情况下，人们会对那些相似程度小于 85%的序列比对感兴趣，这时，就要用到能够反映相关性较远的蛋白质序列比对中氨基酸替换矩阵 PAM-N。

将 PAM-1 自乘 N 次，可以得到矩阵 PAM-N。可以根据待比较序列的长度以及序列间的先验相似程度来选用特定的 PAM 矩阵，以发现最适合的序列比对。一般地，在比较差异大的序列时，采用 N 值较高的 PAM 矩阵可以得到较好的结果，比如选择 PAM-200 到 PAM-250 之间的矩阵；而 N 值较低的 PAM 矩阵一般用于高度相似的序列。实践中用得较多且比较折中的矩阵是 PAM-250，表 3.11 是 PAM-250 突变概率矩阵。

表 3.11　PAM-250 突变概率矩阵

		原氨基酸																			
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
替代氨基酸	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	5	4	15	10	4	10	5	5	5	7	2	4	17

注：表中数据来自《生物信息学与功能基因组学》。

可以把 PAM 突变概率矩阵转换成用于序列比对的打分矩阵，即对数比值矩阵(log-odds matrix)或相关比值矩阵(relatedness odds matrix)。

对数矩阵的每个元素基于两个概率的“比值比(odds ratio)”，这两个概率描述了某一 PAM 间隔内，氨基酸 a 替换氨基酸 b 的概率。比对 a 和 b 的分值 S 由式(3.4)给出

$$S(\mathrm{a},\mathrm{b}) = 10\lg(M_{\mathrm{ab}}/p_{\mathrm{b}}) \tag{3.4}$$

式(3.4)中，M_{ab}为真实比对下氨基酸 a 和 b 比对的概率，p_{b} 为归一化频率，代表随机情况下氨基酸 b 出现的频率。

PAM-250 的对数比值矩阵见表 3.12，所有的值近似到最近的整数。不同于表 3.11 的突变概率矩阵，这个打分矩阵是对称的，因为比对两条序列时，哪条序列为原序列或替代序列都不影响比对结果。

PAM-250 的对数比值矩阵中：①主对角线上的分值是两个相同残基之间的相似性分数，该分值较高，说明对应的氨基酸比较保守，不易突变，而分值较低时，说明这些氨基酸比较容易突变；②不同氨基酸之间的分数值越高，它们之间的相似性越高，进化过程中越容易发生互相替换；③相似性分数值为负数的氨基酸之间的相似性较低，它们在进化过程中不易发生互相替换。

PAM 矩阵有不同的类型，如 PAM-250，PAM-120，PAM-80，PAM-60 等，名称中短线后的数字代表的是序列的进化距离。PAM-250 表示两序列的进化距离大，则相应的相似性

分数就小，约为 14%～27%；PAM-60 表示两序列的进化距离小，则相应的相似性分数就大，约为 60%。

表 3.12　PAM-250 对数比值矩阵

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

注：表中数据来自《生物信息学与功能基因组学》P53，图 3-14。

(5) BLOSUM 矩阵

BLOSUM 打分矩阵是另一种常用的氨基酸替换矩阵，它也是通过统计相似蛋白质序列的替换率而得到的。PAM 矩阵是从蛋白质序列的全局比对结果推导出来的，而 BLOSUM 矩阵则基于蛋白质序列块比对。基本数据来源于 BLOCKS 数据库，其中包括局部多重比对，包含较远的相关序列。虽然在这种情况下没有使用进化模型，但它的优点在于可以通过直接观察而不是通过外推获得数据。

同 PAM 打分矩阵一样，BLOSUM 矩阵也有一系列的矩阵 BLOSUM-N，可以根据亲缘关系的不同来选择不同的 BLOSUM-N 矩阵进行序列比对。BLOSUM-N 矩阵中 N 值的意义与 PAM-N 矩阵中 N 值的意义正好相反。N 值较低的 PAM 矩阵适合用来比较亲缘较近的序列，而 N 值较低的 BLOSUM 矩阵更多是用来比较亲缘较远的序列。一般来说，BLOSUM-62 矩阵适于用来比较大约具有 62%相似度的序列，而 BLOSUM-80 矩阵更适合于相似度为 80%左右的序列。表 3.13 是 BLOSUM-62 氨基酸置换矩阵。

PAM 矩阵和 BLOSUM 矩阵建立的理论基础不同，PAM 矩阵是对相关序列中所有氨基酸位置进行记分，而 BLOSUM 矩阵则是基于相关序列中最相似的共同区域中氨基酸的替换和匹配，所以在实际应用中，PAM 矩阵可用于寻找蛋白质的进化起源，而 BLOSUM 矩阵则是用于发现蛋白质的保守区域。

表 3.13　BLOSUM62 打分矩阵

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

注：表中数据来自《生物信息学与功能基因组学》P56，图 3-17。

3.1.4　序列比对的空位罚分

为了获得两个序列的最优比对，在序列比对中需要引入插入 Insert(-，b)和删除 Delete(a，-)这两种编辑操作，即在两条序列的某个位置上插入空位。比对的计分是由得分与罚分两部分组成，罚分又包括对失配(替换)进行罚分和对空位进行罚分。对得分和失配罚分是根据上述的打分矩阵进行计算的，那么空位罚分又是怎样计算的呢？空位罚分的选取必须与打分矩阵相匹配，如果用一个太高的空位罚分，空位就不会出现在比对中，相反，如果空位罚分过低，空位将出现在比对的任何地方。

空位罚分涉及几个问题：①空位罚分是否大于失配罚分；②空位的引入与延伸是否予以不同的罚分，这些问题对序列比对的结果有显著影响。

对于空位罚分是否大于失配罚分，其确定类似于替换打分矩阵的选择，需要根据序列特征而定。如果已知比对的序列中包含较多的在进化中引入的插入和删除突变，则引入空位可合理地代表这些突变，这时可令空位罚分小于失配罚分，使得同源的未突变的字符得以匹配；如果比对的序列少有插入和删除突变，但有许多替换突变，则不应轻易引入空格去破坏已变异但仍同源的字符序列，这时可令空位罚分大于失配罚分。

对于空位的引入与延伸是否予以不同的罚分，可如下处理：定义“k 阶空位”是一个具有连续“空位”字符的区域，其空位字符的数目 $k>1$。对于序列的突变，k 阶空位出现的可能性要大于 k 个不连续的空位出现的可能性，这是因为一个 k 阶空位对应于一串字符的插入

或者删除，对应于一个遗传突变事件，而不连续的空位可能对应于多个不同的突变事件，从遗传变异的角度来分析，一个突变的发生要比多个突变同时发生的可能性大。从这个意义上说，我们希望将 k 阶空位与 k 个不连续的空位在打分方面区别开来，k 阶空位的罚分应该比 k 个不连续空位的罚分低。所以，空位罚分应分两种：第一个空位(gap_open)罚分和延伸空位(gap_extend)罚分，且第一个空位罚分要大于延伸空位罚分。

3.2 双序列比对

3.2.1 概述

双序列比对是指通过一定算法对两个核酸或蛋白质序列按照一定的规律排列起来，逐对比较其字符的异同，判断它们之间的相似程度和同源性，从而推测它们的结构、功能以及进化上的联系。

双序列比对是常用的序列分析方法之一，序列的测定和拼接、RNA 和蛋白质的结构功能预测、系统发育树的构建等都需要对生物分子进行序列相似性的比较。序列比对在发现生物序列有关功能、结构和进化信息等方面具有非常重要的意义。

双序列比对还是数据库搜索的基础，将查询序列与整个数据库的所有序列进行比对，从数据库中获得与其最相似序列的数据，能快速地获得有关查询序列的参考信息，对于进一步分析其结构和功能都会有很大的帮助。

双序列比对可以用以下方法来实现，即点阵法(dot matrix method)、动态规划法(dynamic programming method)、K-元法(K-tuple method)/字串法(word method)。

点阵法是双序列比对的基本方法，通常用图示法表示，这种方法能以矩阵对角线的形式显示尽可能多的比对。点阵法可以很容易地发现插入/ 删除序列和重复序列，这对于其他方法要困难一些。该方法的主要局限性在于，对绝大多数点阵，计算机程序不能显示出精确的序列。

1970 年 Needleman 和 Wunsch 首先将动态规划法用于全局比对。到 1981 年，Smith 和 Waterman 又将该算法用于局部比对。动态规划的基本思路是将复杂问题分解成与之相似的子问题，再通过求解各个子问题来得到整个问题的解决方案。

K-元法/字串法，用于 FastA 和 BLAST 搜索程序。该方法从寻找完全匹配的短片段(称为 K-元或字)出发，并以此为基础运用动态规划方法将这一片段向两端延伸，得到较长的相似性匹配。其采用启发式算法提高运行速度，可以在普通计算机上运行。

3.2.2 点阵法

1. 点阵法原理

点阵法是由 Gibbs & McIntyre 于 1970 年首先提出。点阵法是双序列比对中最基本也是最直观的方法，它是利用点阵图来显示出相似的区域。在构造一个简单点阵图时，第一条比对的序列自左向右排列在点阵空间的横轴，第二条比对的序列自下而上排列在纵轴。点阵空间中两条序列中的核苷酸或氨基酸相同时，在对应的位点上做出标记。两条序列间连续相同的区域在图中会形成由标记组成的斜线，如图 3.4 所示，点阵图的名称由此而来。

2. 点阵法的应用

(1) 寻找两条比对序列间所有可能的相似之处

将两条待比较的序列分别放在矩阵的两个轴上，一条从左到右排列在 X 轴上，一条从下往上排列在 Y 轴上，如图 3.4 所示。当对应的行与列的序列字符匹配时，则在矩阵对应的位置作出“点”标记。逐个比较所有的字符对，最终形成点矩阵。

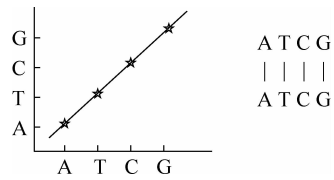

图 3.4　序列比对点阵图

显然，如果两条序列完全相同，则在点矩阵主对角线的位置都有标记；如果两条序列存在相同的子串，则对于每一个相同的子串对，有一条与对角线平行的由标记点所组成的斜线，如图 3.5 中的斜线代表相同的子串 TAT 和 GCCT，而对于两条互为反向的序列，则在反对角线方向上有标记点组成的斜线，如图 3.6 所示。

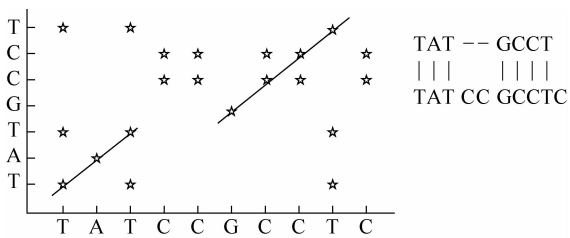

图 3.5　多个连续子序列点阵图

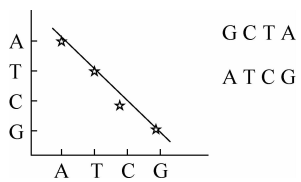

图 3.6　反向序列点阵图

对于矩阵标记图中非重叠的与对角线平行的斜线，可以组合起来，形成两条序列的一种比对，在两条子序列的中间可以插入符号“-”，表示插入空位字符，如图 3.5 所示。找两条序列的最佳比对(对应位置等同字符最多)，实际上就是在点阵图中找非重叠平行斜线最长的组合。

除非已经知道待比较的序列非常相似，一般先用点矩阵方法比较，因为这种方法可以通过观察点阵的对角线迅速发现可能的相同片段。

(2) 发现序列正向或反向的重复

重复序列(repeated sequence)是指一个序列中出现不止一次的子序列，重复序列可以彼此方向相同，也可以相反。如序列 ctgactgactga 就为重复序列，点阵分析如图 3.7 所示。从图中可以看出，点阵分析可以用来发现序列中的正向(或反向)重复序列。当把序列同自身进行比对时，从点阵中的斜线可以揭示出其中的重复片断。自身序列比对的点阵图是沿着主对角线对称的。

重复序列常常给比对造成困难，因为它会人为地造成比对的分值增高，而这种重复所在的区域仅能提供较少的生物信息，称为低复杂区域(low-complexity regions)。分析程序通常自动不考虑这些区域，通过对序列自身的比对，就可以找到其中的重复子序列。

(3) 分析长且相似的序列

当对长且相似的序列进行比对时，点阵图会变得非常复杂和拥挤，以至于看不清相似序列在什么位置。以 HEXA 基因为例，用点阵法比对人的 HEXA 基因和小鼠的 HEXA 基因，其结果见图 3.8。人的 HEXA 基因在 GenBank 中的编号为 NM_000520，小鼠的 HEXA 基因在 GenBank 中的编号为 AK_080777。从图 3.8 看不到两条序列有什么相似性。

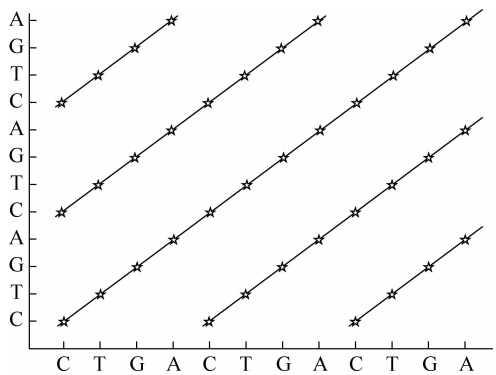

图 3.7 重复序列点阵图

如果使用滑动窗口技术代替一次一个位点的比较，会得到什么结果呢？图 3.9 就是采用滑动窗口技术对人的 HEXA 基因和小鼠的 HEXA 基因进行点阵比对的结果，使用的滑动窗口的大小为 10，相似性阈值为 8。首先，X 轴序列第 1-10 的核苷酸与 Y 轴序列第 1-10 的核苷酸进行比较。如果在第一次比较中，这 10 个核苷酸中有 8 个或者 8 个以上相同，那么就在点阵空间(1,1)的位置画上圆点。然后窗口沿 X 轴向前移过一个核苷酸的位置，将 X 轴序列第 2-11 的核苷酸与 Y 轴序列第 1-10 的核苷酸比较。不断重复这个过程，直到 X 轴中每个长度为 10 的子序列都与 Y 轴第 1-10 的核苷酸比较过为止。接着，Y 轴的窗口向前移过一个核苷酸的位置，重复以上过程，直到两条序列中所有长度为 10 的子序列都被两两比较过为止。图 3.9 显示出了人的 HEXA 基因和小鼠的 HEXA 基因的相似序列。

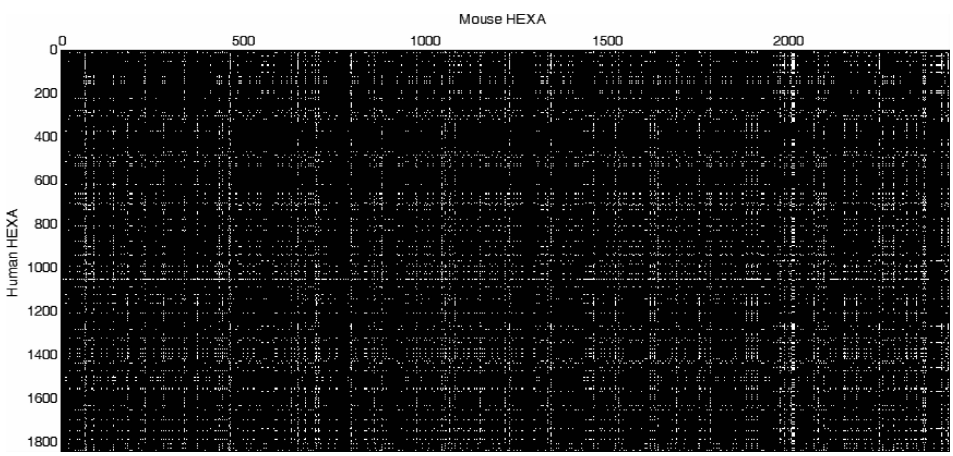

图 3.8 人类的 HEXA 基因和小鼠的 HEXA 基因比对的点阵图

3.2.3 动态规划法

进行序列的两两比对最直接的方法就是生成两条序列所有可能的比对，分别计算得分或代价，然后挑选一个得分最高(或代价最小)的比对作为最优比对。但是，两条序列可能的比对结果非常多，而且随着序列长度的增长，计算量以更快的速度增长。因此，对于较长的

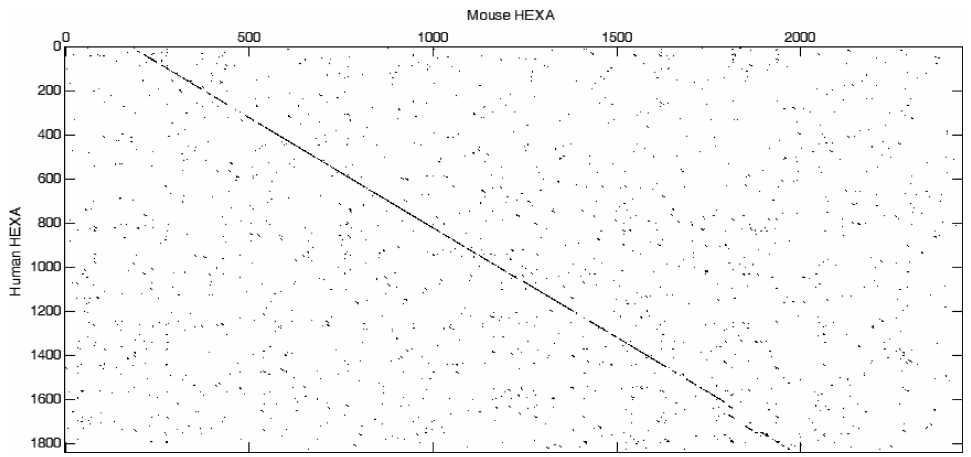

图 3.9　滑动窗口大小为 10，相似度阈值为 8 的人类和小鼠的 HEXA 基因比对的点阵图

序列，这种比对方法显然不合适。用前面所介绍的点矩阵分析方法，在寻找斜线及斜线组合时，仍然需要较大的运算量。因此，必须提高算法的效率以找出最优比对。动态规划算法就是一个很好的选择。

1. 动态规划法原理

动态规划把一个复杂问题分解成计算量较小的子问题，并用这些子问题的结果来计算整个问题的答案，即动态规划算法以递归形式来解决决策过程中的最优化问题。

什么样的复杂问题能够用动态规划法来解决呢？动态规划法要求被解答的问题应具有以下 4 个特点：①被解答的问题能够划分成一系列相继的阶段；②起始阶段包含基本子问题的解；③在后续阶段中，能够按递归方式根据前面阶段的部分结果计算每个部分解；④最后阶段包含全局解。

双序列比对问题是否具有动态规划法所要求的上述 4 个特点呢？

假设要比对这样两条序列：ACAGA 和 AGA，那么第一个位点的比对就存在 3 种可能：①两条序列的第一个字母比对；②第一条序列加一空位与第二条序列的第一个字母比对；③第二条序列加一空位与第一条序列的第一个字母比对，如表 3.14 的“第一位点”所示。根据式(3.1)得分函数的定义，对于第①种情况，因为是两个 A 比对，所以比对会得到匹配奖励，得 +1 分，对于第②、③种情况，都是空位与字母比对，要进行罚分，得 −1，见表 3.14 的“得分”所示。如果能够得到剩余序列的最优比对得分，就可以计算 3 种情况下的比对得分，然后选择得分最高的作为最优比对得分，对应的比对就是最优比对。

求解过程中是这样的，在第一位点比对的 3 种情况中，选择最优的比对结果，即第①种情况作为第一位点的比对结果。然后再计算这种情况下剩余序列 CAGA 和 GA 中第一个字母的比对，同样有 3 种情况，见表 3.15。根据式(3.1)得分函数的定义计算 3 种情况下的得分，然后将表 3.14 中第一位点的最优比对结果“+1”带到表 3.15 中的 3 种情况的比对结果中进行加和，分别得到 3 种情况下的得分：+1；0；0。在这 3 个结果中选择得分最高的 +1，接着将剩余序列依次比对，直到最后位点。

从上述这个求解过程中可以看到，双序列比对问题具有动态规划法所要求的 4 个特点，并且在每个位点的比对得分计算中，我们只是选择得分最优的结果，并据此比对剩余的序

列，并将上一步计算结果带到本次计算中，这样能够保证每一步的计算结果都是最优的；同时每一步都略去了其他的非最优结果及剩余序列的比对，大大地降低了计算量。

表 3.14　序列 ACAGA 和 AGA 第一位点比对的 3 种可能

	第一位点	得　分	待比对的剩余序列
①	A	+1	CAGA
	A		GA
②	—	−1	ACAGA
	A		GA
③	A	−1	CAGA
	—		AGA

表 3.15　剩余序列 CAGA 和 GA 第一位点比对的 3 种可能

	第一位点	得　分	待比对的剩余序列
①	C	0	AGA
	G		A
②	—	−1	CAGA
	G		A
③	C	−1	AGA
	—		GA

2. 全局比对的动态规划法

1970 年 Saul Needleman 和 Christian Wunsch 将动态规划思想引入到两条序列的全局比对中，后称为 Needleman-Wunsch 算法。这种算法能够获得核酸序列和蛋白质序列的最优比对，也允许在序列中引入空位。

设序列 s、t 的长度分别为 m 和 n，其前缀分别为 $_0:s_{:i}$ $(1\leqslant i\leqslant m)$ 和 $_0:t_{:j}$ $(1\leqslant j\leqslant n)$。假如已知序列 $_0:s_{:i}$ 和 $_0:t_{:j}$ 所有较短的连续子序列的最优比对，即已知：①$_0:s_{:(i-1)}$ 和 $_0:t_{:(j-1)}$ 的最优比对；②$_0:s_{:(i-1)}$ 和 $_0:t_{:j}$ 的最优比对；③$_0:s_{:i}$ 和 $_0:t_{:(j-1)}$ 的最优比对。则 $_0:s_{:i}$ 和 $_0:t_{:j}$ 的最优比对一定是上述三种情况之一的扩展，即：①的最优比对得分 + 替换 (s_i,t_j) 或匹配 (s_i,t_j) 的得分；②的最优比对得分 + 删除 $(s_i,-)$ 的得分；③的最优比对得分 + 插入 $(-,t_j)$ 的得分。

令 $S(_0:s_{:i},_0:t_{:j})$ 为序列 $_0:s_{:i}$ 与序列 $_0:t_{:j}$ 比对的得分，可根据下面的递推公式计算最大值

$$S(_0:s_{:i},_0:t_{:j})=\max\begin{cases}S(_0:s_{:(i-1)},_0:t_{:(j-1)})+p(s_i,t_j)\\S(_0:s_{:(i-1)},_0:t_{:j})+p(s_i,-)\\S(_0:s_{:i},_0:t_{:(j-1)})+p(-,t_j)\end{cases}\tag{3.5}$$

其初值为

$$\begin{aligned}&S(_0:s_{:0},_0:t_{:0})=0\\&S(_0:s_{:i},_0:t_{:0})=S(_0:s_{:(i-1)},_0:t_{:0})+p(s_i,-),\quad(i=1,2,\cdots,m)\\&S(_0:s_{:0},_0:t_{:j})=S(_0:s_{:0},_0:t_{:(j-1)})+p(-,t_j),\quad(j=1,2,\cdots,n)\end{aligned}\tag{3.6}$$

按照这种方法，根据给定的打分函数 $p(s_i,t_j)$，两条序列所有前缀的最优比对得分值可

以定义一个得分矩阵

$$\boldsymbol{D}=(d_{ij})$$

其中，$d_{ij}=S(_0:s_{:i},_0:t_{:j})$，即两条序列任一前缀对的最优比对得分。对于一个长度为 n 的序列，有$(n+1)$个前缀(包括一个空序列)，所以，得分矩阵的大小为$(m+1)\times(n+1)$。其中，矩阵的纵轴方向自上而下对应于第一条序列 s，横轴方向从左到右对应于第二条序列 t。矩阵横向移动表示在纵轴序列相应的位置加入一个空位，纵向的移动表示在横轴序列相应的位置加入一个空位，而斜对角方向的移动表示两序列各自相应的字符进行比对。注意，两轴第一个元素的索引下标为 0。

得分矩阵中任一元素 d_{ij} 的计算公式如下

$$d_{ij}=\max\begin{cases}d_{i-1,j-1}+p(s_i,t_j)\\ d_{i-1,j}+p(s_i,-)\\ d_{i,j-1}+p(-,t_j)\end{cases}\tag{3.7}$$

其中，d_{ij} 最大值的三种选择决定了各矩阵元素之间的关系，用图 3.10 表示。

矩阵右下角元素即为期望的结果：$d_{mn}=S(_0:s_{:m},_0:t_{:n})=S(s,t)$。

应用动态规划算法求两条序列的最优比对的步骤分以下四步：

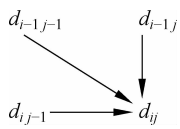

图 3.10　得分矩阵中元素 d_{ij} 的计算

(1) 初始化得分矩阵 $\boldsymbol{D}$。根据得分函数对得分矩阵的第一行和第一列进行初始化，以式(3.1)的得分函数为例，矩阵第一行的初始化为 $d_{0j}=-j$，$(0\leqslant j\leqslant n)$，矩阵第一列的初始化为 $d_{i0}=-i$，$(1\leqslant i\leqslant m)$。

(2) 计算得分矩阵中的每个元素。根据式(3.7)从 d_{00} 开始，可以是按行计算，每行从左到右，也可以是按列计算，每列从上到下，任何计算过程，只要满足在计算 d_{ij} 时 $d_{i-1,j}$、$d_{i-1,j-1}$ 和 $d_{i,j-1}$ 都已经被计算即可。在计算 d_{ij} 后，需要保存 d_{ij} 是从 $d_{i-1,j}$、$d_{i-1,j-1}$ 或 $d_{i,j-1}$ 中的哪一个推进的，即保存计算的路径，以便于后续处理。上述计算过程直到 d_{mn} 结束。

(3) 求最优路径。从 d_{mn} 开始反向前推，假设在反推时到达 d_{ij}，现在要根据保存的计算路径判断 d_{ij} 究竟是根据 $d_{i-1,j}$、$d_{i-1,j-1}$ 和 $d_{i,j-1}$ 中的哪一个计算而得到的，找到这个点以后，再从此点出发，反推下一个元素，一直到 d_{00} 为止。走过的这条路径就是最优路径(即得分最大路径)，对应于两条序列的最优比对。

(4) 根据最优路径求出两条序列的最优比对。根据第(3)步反推出的最优路径，从 d_{00} 开始，在得分矩阵 $\boldsymbol{D}$ 中，最优路径斜对角线方向的移动表示两序列各自相应的字符进行比对，最优路径横向移动表示在纵轴序列相应的位置加入一个空位，纵向移动表示在横轴序列相应的位置加入一个空位，直到 d_{mn}。

例 3.3　用动态规划法对序列 $s=$ AGCACACA 和 $t=$ ACACACTA 进行全局比对，采用式(3.1)的得分函数。

解　① 初始化得分矩阵 $\boldsymbol{D}$。

根据条件，矩阵第一行的初始化为 $d_{0j}=-j(0\leqslant j\leqslant 8)$，矩阵第一列的初始化为 $d_{i0}=$

$-i(1\leqslant i\leqslant 8)$，结果见图 3.11。

		A	C	A	C	A	C	T	A
	0	−1	−2	−3	−4	−5	−6	−7	−8
A	−1								
G	−2								
C	−3								
A	−4								
C	−5								
A	−6								
C	−7								
A	−8								

图 3.11　序列 s = AGCACACA 和 t = ACACACTA 比对的初始化得分矩阵

② 计算得分矩阵 $\boldsymbol{D}$ 中的每个元素。

根据式(3.7)从 d_{11} 开始顺序计算得分矩阵 $\boldsymbol{D}$ 中的每个元素，直到 d_{88}。结果见图 3.12。从图 3.12 矩阵中可以看到，两条序列的最优比对得分为 d_{88} 的值，即最优比对得分为 5。

以得分矩阵中 d_{11} 元素的计算为例，d_{11} 可以通过三种途径到达该位置：

- 从 d_{00} 出发，经过两条序列第一个字符的比对(A，A)；
- 从 d_{01} 出发，在第二条序列加上的空位(A，−)；
- 从 d_{10} 出发，在第一条序列加上的空位，(− ，A)。

因此，矩阵中 d_{11} 值来自于下列三个值中最大的一个：

- d_{00} 的值加上匹配(A，A)的得分 1，和为 1；
- d_{01} 的值加上空位罚分−1，和为−2；
- d_{10} 的值加上空位罚分−1，和为−2。

最终，矩阵中 d_{11} 的值等于 1。

		A	C	A	C	A	C	T	A
	0	−1	−2	−3	−4	−5	−6	−7	−8
A	−1	1	0	−1	−2	−3	−4	−5	−6
G	−2	0	1	0	−1	−2	−3	−4	−5
C	−3	−1	1	1	1	0	−1	−2	−3
A	−4	−2	0	2	1	2	1	0	−1
C	−5	−3	−1	1	3	2	3	2	1
A	−6	−4	−2	0	2	4	3	3	3
C	−7	−5	−3	−1	1	3	5	4	3
A	−8	−6	−4	−2	0	2	4	5	5

图 3.12　序列 s = AGCACACA 和 t = ACACACTA 比对的得分矩阵及最优路径

③ 求最优路径。

从 d_{88} 开始反向前推，根据保存的计算判断 d_{88} 究竟是根据 d_{78}、d_{77} 和 d_{87} 中的哪一个计算而得到的，找到这个点以后，再从此点出发，反推下一个元素，一直到 d_{00} 为止。从 d_{88} 到 d_{00} 的最优路径为：$d_{88}\rightarrow d_{77}\rightarrow d_{76}\rightarrow d_{65}\rightarrow d_{54}\rightarrow d_{43}\rightarrow d_{32}\rightarrow d_{21}\rightarrow d_{11}\rightarrow d_{00}$。结果见图 3.12。

④ 根据最优路径求出两条序列的最优比对。

从 d_{00} 开始，最优路径斜对角线方向的移动表示两序列各自相应的字符进行比对(匹配或替换)，最优路径横向移动表示在纵轴序列相应的位置加入一个空位，纵向的移动表示在横轴序列相应的位置加入一个空位，直到 d_{88}。结果见图 3.13。

```
s  A G C A C A C - A
   |   | | | | |   |
t  A - C A C A C T A
```

图 3.13　序列 s = AGCACACA 和 t = ACACACTA 的最优比对

值得注意的一点是，在有些情况下，得分矩阵中某些元素的值来自两个以上的路径，亦即存在几条最优路径，因此最优比对并非唯一。

以上计算是在得分函数的基础上进行的，得分值表示序列之间的相似程度。在实际应用中，也可以利用“代价”函数进行计算。两条序列比对的代价越低，序列就越相似；比对的代价越高，序列的差异就越大。因为计算方式刚好与得分函数相反，所以，具体计算时应求出最小代价所对应的路径。一般来讲，由于比对的得分可正可负，使用起来就更加灵活，所以大量的序列比对实用程序在计算时都采用得分的概念。

在计算序列相似程度时还应该考虑序列长度的影响。令 $S(s,t)$ 表示两条长度分别为 m 和 n 的序列的相似性得分，假设 $S(s,t)=99$，两条序列有 99 个字符一致。如果 $m=n=100$，则可以说这两条序列非常相似，几乎一样。但是，如果 $m=n=1000$，则仅有 10% 的字符相同。所以，在实际序列比较时，使用相对长度的得分就更有意义：

$$sim(s,t)=\frac{2S(s,t)}{m+n} \tag{3.8}$$

以 $sim(s,t)$ 作为衡量序列相似性的指标。

3. 局部比对的动态规划法

有些同源序列虽然序列的整体相似性很小，但是存在高度相似的局部区域，如果在进行序列比对时注重序列的局部相似性，则可能会发现重要的序列相似信息。1981 年，Temple Smith 和 Michael Waterman 对双序列的局部比对进行了研究，在 Needleman-Wunsch 算法的基础上进行改进，提出序列局部比对算法，后称 Smith-Waterman 算法。

设序列 s、t 的长度分别为 m 和 n，在这里使用的数据结构依然是一个 $(m+1)\times(n+1)$ 的得分矩阵 $\boldsymbol{D}$，但是，对矩阵中每个元素含义的解释与 Needleman-Wunsch 算法有所不同。因为在进行两条序列局部比对时，不计某序列前缀与空位的罚分，也不计某序列后缀与空位的罚分，所以，矩阵中每个元素的值代表序列 $_0:s_{:i}$ $(0\leqslant i\leqslant m)$ 某个后缀和序列 $_0:t_{:j}$ $(0\leqslant j\leqslant n)$ 某个后缀的最佳比对。

因为在双序列局部比对中不计前缀的得分，所以新的得分矩阵初始化见下式

$$\begin{aligned} d_{0,j}&=0 \quad (0\leqslant j\leqslant n)\\ d_{i,0}&=0 \quad (1\leqslant i\leqslant m)\end{aligned} \tag{3.9}$$

另外，由于$_0:s_{:i}$和$_0:t_{:j}$总有一个得分为 0 的空后缀比对，所以矩阵 $\boldsymbol{D}$ 中的所有元素大于或等于 0。于是，得分矩阵中任一元素的计算见下式

$$d_{i,j}=\max\begin{cases}d_{i-1,j-1}+p(s_i,t_j)\\ d_{i-1,j}+p(s_i,-)\\ d_{i,j-1}+p(-,t_j)\\ 0\end{cases}\tag{3.10}$$

阈值 0 意味着矩阵中的 0 元素分布区域对应于不相似的子序列，而正数区域则是局部相似的区域。最后，在矩阵中找出最大值，该值就是最优的局部比对得分，它所对应的点为序列局部比对的末点；然后，反向推演前面的最优路径，直到局部比对的起点。

例 3.4 用动态规划法对序列 s = GCGATATA 和 t = AACCTATAGCT 进行局部比对，采用式(3.1)的得分函数。

解 ① 初始化得分矩阵 $\boldsymbol{D}$。

根据式(3.9)对得分矩阵 $\boldsymbol{D}$ 进行初始化，矩阵第一行的初始化为 $d_{0j}=0$，$(0\leqslant j\leqslant 11)$，矩阵第一列的初始化为 $d_{i0}=0(0\leqslant i\leqslant 8)$，结果见图 3.14。

		A	A	C	C	T	A	T	A	G	C	T
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	1	0	0
C	0	0	0	1	1	0	0	0	0	0	2	0
G	0	0	0	0	0	0	0	0	0	1	1	2
A	0	1	1	0	0	0	1	0	1	0	0	0
T	0	0	0	0	0	1	0	2	1	0	0	1
A	0	1	1	0	0	0	2	1	3	2	1	0
T	0	0	0	0	0	1	1	3	2	3	2	2
A	0	1	1	0	0	0	2	2	4	3	3	2

图 3.14 序列 s = GCGATATA 和 t = AACCTATAGCT 局部比对的得分矩阵

② 计算得分矩阵 $\boldsymbol{D}$ 中的每个元素。

根据递推公式式(3.10)和得分函数公式式(3.1)，从 d_{11} 开始顺序计算得分矩阵 $\boldsymbol{D}$ 中的每个元素，直到 $d_{8,11}$。结果见图 3.14。

③ 求最优路径。

从图 3.14 的得分矩阵中找到最大得分元素 $d_{88}=4$，反向前推，其最优路径为：$d_{88}=4\rightarrow d_{77}=3\rightarrow d_{66}=2\rightarrow d_{55}=1\rightarrow d_{44}=0$。结果见图 3.14。

④ 根据最优路径求出两条序列的最优局部比对。

根据最优路径，求出最优局部比对，见图 3.15。

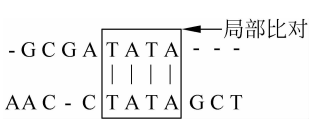

图 3.15 序列 s = GCGATATA 和 t = AACCTATAGCT 的最优局部比对

3.2.4　BLAST 算法

1. BLAST 算法原理

BLAST 第 2 章已定义，即基本局部比对搜索工具，其基本原理是先找出某些“种子”，即两条比对序列中的一对子序列，它们的长度相等且可以形成无空位的完全匹配。BLAST 首先找出两条比对序列间所有匹配度超过一定阈值的“种子”，然后以每个种子为基准，根据给定的相似性阈值向两端扩展延伸，得到一定长度的相似性片段，比较每个“种子”得到的相似性片段，最后给出高分值片段对(high-scoring pairs，HSPs)，即最优比对结果。不带空位的片段比对是标准 BLAST 的一个特征，改进后允许在延伸的过程中插入空位。

2. BLAST 软件

BLAST 是目前最常用的数据库搜索程序。在 NCBI 的工具箱中有专门用于双序列比对的 BLAST 在线软件，软件的界面见图 3.16，网址为：

http://blast.ncbi.nlm.nih.gov/Blast.cgi? PROGRAM=blastp&BLAST_PROGRAMS=blastp&PAGE_TYPE=BlastSearch&SHOW_DEFAULTS=on&BLAST_SPEC=blast2seq&LINK_LOC=blasttab。

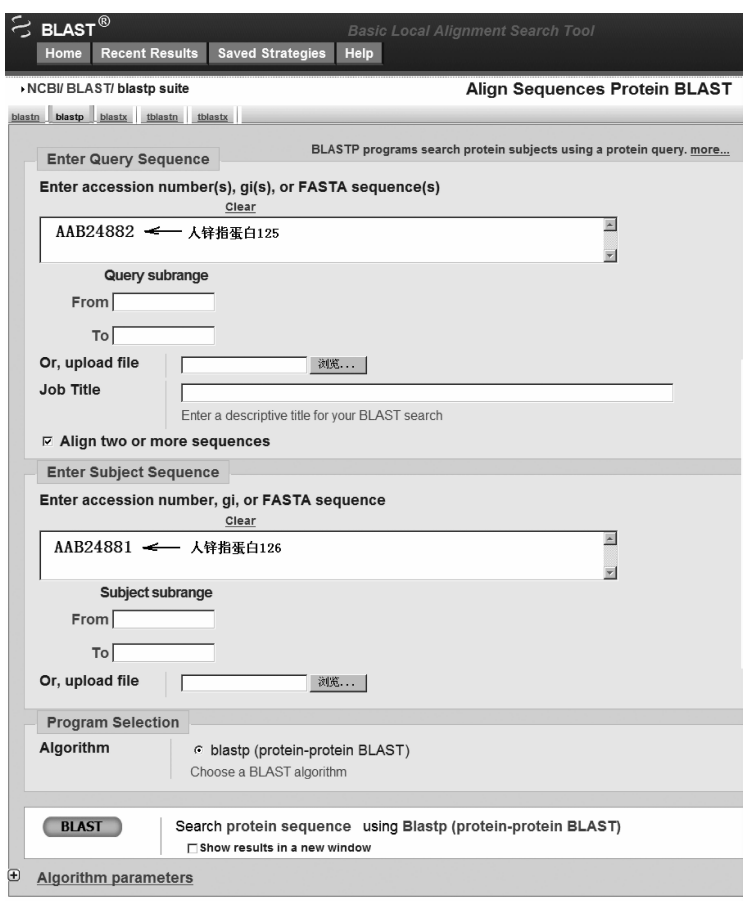

图 3.16　用于双序列比对的 BLAST 在线软件界面

例 3.5 以人锌指蛋白 125 和人锌指蛋白 126 为例，用 BLAST 在线软件进行双序列比对分析，寻找这两种人类锌指蛋白中相同或相似的氨基酸组成区域。

解 人锌指蛋白 125 在 GenBank 的检索号为 AAB24882，人锌指蛋白 126 在 GenBank 的检索号为 AAB24881，将这两个检索号分别写在图 3.16 所示界面的输入框中，选择不同的参数 Algorithm parameters，然后单击 BLAST 按钮，即可得到两条序列的比对结果。图 3.17 是在默认参数下得到的人锌指蛋白 125 和 126 的比对结果。

从图 3.17 中可以看到人锌指蛋白 125 和 126 在默认参数下比对结果共有 4 个：①Score＝113bits，②Score＝96.7bits，③Score＝77.4bits，④Score＝32.3bits。在每种结果中都给出了两条序列的比对，标出了两条序列相同或相似的氨基酸组成区域。

```
>gb|AAB24881.1|  zinc finger [Homo sapiens]
Length=98
                                  Sort alignments for this subject sequence by:
                                    E value  Score  Percent identity
                                    Query start position  Subject start position

 Score =  113 bits (282),  Expect = 7e-38, Method: Compositional matrix adjust.
 Identities = 60/85 (71%), Positives = 68/85 (80%), Gaps = 0/85 (0%)

Query  21   YECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPTPSHLQYHERTHTGEKPYECH  80
            YECN+  KAF+  S L+CH R  IGEK +E NQCGKAF   SHLQ H+RTHTGEKPYEC+
Sbjct  1    YECNQCGKAFAQHSSLKCHYRTHIGEKPYECNQCGKAFSKHSHLQCHKRTHTGEKPYECN  60

Query  81   QCGQAFKKCSLLQRHKRTHTGEKPY  105
            QCG+AF +  LLQRHKRTHTGEKPY
Sbjct  61   QCGKAFSQHGLLQRHKRTHTGEKPY  85

 Score = 96.7 bits (239),  Expect = 2e-31, Method: Compositional matrix adjust.
 Identities = 50/65 (77%), Positives = 54/65 (83%), Gaps = 0/65 (0%)

Query  52   NQCGKAFPTPSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHTGEKPYECNQCG  111
            NQCGKAF   S L+ H RTH GEKPYEC+QCG+AF K S LQ HKRTHTGEKPYECNQCG
Sbjct  4    NQCGKAFAQHSSLKCHYRTHIGEKPYECNQCGKAFSKHSHLQCHKRTHTGEKPYECNQCG  63

Query  112  KAFAQ  116
            KAF+Q
Sbjct  64   KAFSQ  68

 Score = 77.4 bits (189),  Expect = 4e-24, Method: Compositional matrix adjust.
 Identities = 43/63 (68%), Positives = 46/63 (73%), Gaps = 0/63 (0%)

Query  15   HSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPTPSHLQYHERTHTGE  74
            H GEK YECN+  KAFS  SHLQCHKR   GEK +E NQCGKAF     LQ H+RTHTGE
Sbjct  23   HIGEKPYECNQCGKAFSKHSHLQCHKRTHTGEKPYECNQCGKAFSQHGLLQRHKRTHTGE  82

Query  75   KPY  77
            KPY
Sbjct  83   KPY  85

 Score = 32.3 bits (72),  Expect = 2e-07, Method: Compositional matrix adjust.
 Identities = 23/42 (55%), Positives = 25/42 (60%), Gaps = 3/42 (7%)

Query  6    QFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEK  47
            Q H R    H+GEK YECN+  KAFS    LQ HKR   GEK
Sbjct  45   QCHKRT---HTGEKPYECNQCGKAFSQHGLLQRHKRTHTGEK  83
```

图 3.17 应用 BLAST 在线软件对人锌指蛋白 125 和 126 比对结果

3.3 多序列比对

3.3.1 概述

多序列比对是指通过一定算法，对一组核酸或蛋白质序列按照一定的规律排列起来，比较这组序列的异同。

在实际研究中，生物学家并不是仅仅分析单个蛋白质，而是更着重于研究蛋白质之间的关系，研究一个家族中的相关蛋白质，研究相关蛋白质序列中的保守区域，进而分析蛋白质的结构和功能。双序列比对往往不能满足这样的需要，难以发现多个序列的共性，必须同时比对多条同源序列。多序列比对还可用于一组同源蛋白质的比对分析，研究隐含在蛋白质序列中的系统发育的关系，以便更好地理解这些蛋白质的进化。

通过序列的多重比对，可以得到一个序列家族的序列特征。当给定一个新序列时，根据序列特征，可以判断这个序列是否属于该家族。对于多序列比对，现有的大多数算法都基于渐进比对的思想，在序列两两比对的基础上逐步优化多序列比对的结果。进行多序列比对后，可以对比对结果进行进一步处理，如构建序列的特征模式，将序列聚类构建分子进化树等。

3.3.2　SP 模型

SP 模型是一种多重序列比对的评价模型，用来评价所得到的多重序列比对，以确定其优劣。

SP(sum-of-pairs)，逐对加和函数，具有如下特点：①函数形式简单，具有统一的形式，不随序列的个数而发生形式变化。②根据得分函数的意义，函数值应独立于各参数的顺序，即与待比较的序列先后次序无关。③对相同的或相似字符的比对，奖励的得分值高，而对于不相关的字符比对或空白，则进行惩罚(得分为负值)。

SP 函数用于多重序列比对的评价有两种方法。

方法一　先计算多重比对结果的每一列字符的得分，然后将各列的和加起来求整体多重比对得分，即

$$\mathrm{SP-score}(c_1,c_2,\cdots,c_k)=\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}p(c_i,c_j) \tag{3.11}$$

其中 $c_1,c_2,\cdots,c_k$ 是多重序列比对中某一列中的 k 个字符，p 是关于一对字符相似性的得分函数。SP-score($c_1,c_2,\cdots,c_k$)是多重序列比对中某一列的 SP 得分。

方法二　先计算多重序列比对结果的两两比对得分，然后将其相加求整体多重比对得分。

$$\mathrm{SP-score}(\alpha)=\sum_{i<j}\alpha_{i,j} \tag{3.12}$$

其中，α 是一个多重比对，α_{ij} 是由 α 推演出来的序列 s_i 和 s_j 的两两比对。

在 $p(-,-)=0$ 条件下，方法一和方法二的计算才等价。

例 3.6　已知得分函数为 $p(a,a)=1$；$p(a,b)=-1$；$p(a,-)=p(-,b)=-1$；$p(-,-)=0$。计算图 3.18 所示多重序列比对的 SP 得分。

```
N H V K W Y Q Q L P G
I  T  V N W Y Q Q L P G
Y A MY W V R Q A P G
Y Y S  T W V R Q P P G
```

图 3.18　多重序列比对结果

解 根据式(3.11)有

$$\begin{aligned}\mathrm{SP}(c_1) &= p(N,I)+p(N,Y)+p(N.Y)+p(I,Y)+p(I,Y)+p(Y,Y)\\ &=(-1)+(-1)+(-1)+(-1)+(-1)+1\\ &=-4\\ \mathrm{SP}(c_2) &= p(H,T)+p(H,A)+p(H,Y)+p(T,A)+p(T,Y)+p(A,Y)\\ &=(-1)+(-1)+(-1)+(-1)+(-1)+(-1)\\ &=-6\end{aligned}$$

同理，$\mathrm{SP}(c_3)=-4$，$\mathrm{SP}(c_4)=-6$，$\mathrm{SP}(c_5)=6$，$\mathrm{SP}(c_6)=-2$，$\mathrm{SP}(c_7)=-2$，$\mathrm{SP}(c_8)=6$，$\mathrm{SP}(c_9)=-4$，$\mathrm{SP}(c_{10})=6$，$\mathrm{SP}(c_{11})=6$。则

$$\begin{aligned}\mathrm{SP}(c_1,c_2,\cdots,c_{11}) &= (-4)+(-6)+(-4)+(-6)+6+(-2)+(-2)+6+(-4)+6+6\\ &=-4\end{aligned}$$

3.3.3 动态规划法

和双序列比对一样，动态规划法仍然能够用于多重序列比对。

在例 3.3 中，图 3.12 所示的得分矩阵相当于二维平面；而对于三条序列的比对得分会形成三维立体空间，每一种可能的比对可用三维晶格中的一条路径表示，而每一维对应于一条序列；如果是多条序列的比对，得分形成的空间则是超晶格(hyperlattice)。

如图 3.10 所示，在计算双序列比对的得分矩阵中的任一元素 d_{ij} 时，要依赖 $d_{i-1,j}$、$d_{i-1,j-1}$ 和 $d_{i,j-1}$ 的数值，此时 d_{ij} 称为当前节点，$d_{i-1,j}$、$d_{i-1,j-1}$ 和 $d_{i,j-1}$ 称为前趋节点。

如图 3.19 所示，在计算三条序列比对的当前节点的得分时，要依赖于与它相邻的 7 条边，分别对应于匹配、替换或引入空位等三种编辑操作，计算各操作的得分，并与相应的前趋节点的得分相加，选择一个得分最大的操作，并将得分存放于该节点。在三维晶格中，计算当前节点的得分要考虑 7 个前趋节点，在 k 维情况下要考虑 2^k-1 个前趋节点。

随着待比对的序列数目增加，计算量和所要求的计算空间猛增。对于 k 个序列的比对，动态规划算法需要处理 k 维空间里的每一个节点，计算量自然与晶格中的节点数成正比，而节点数等于各序列长度的乘积；另外，计算每个节点依赖于其前趋节点的个数为 2^k-1。因此，用动态规划方法计算多序列比对的最优得分的时间与空间复杂性太高，所以人们发展了该算法的多种变体使得它们能够在合理的时间内找到优化比对。

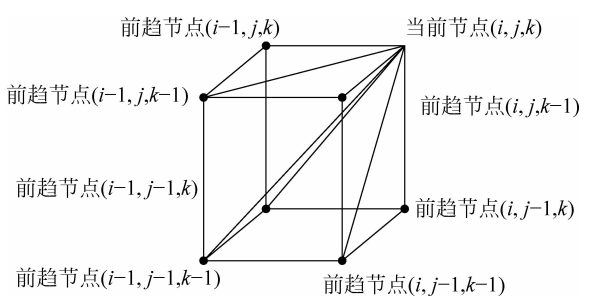

图 3.19 三维晶格中当前节点计算的依赖关系

3.3.4　星形比对算法

采用标准的动态规划算法计算最优的多重序列比对时，其时间与空间复杂性太高以至于难以完成，所以必须考虑其他的方法。首选的就是启发式方法。启发式方法不一定保证最终能得到最优解，但在大多数情况下，其计算结果接近于最优结果；重要的一点是，这类方法能够大大减少所需的计算时间，加快计算速度。目前所用的算法大部分将序列多重比对转化为序列两两比对，逐渐将两两比对组合起来，最终形成完整的多序列比对。这种方式又称为渐进法。

星形比对是一种启发式方法，是由 Gusfield 首先提出的。星形比对的基本思想是：在给定的若干条序列中，选择一个核心序列，通过该序列与其他序列的两两比对，形成所有序列的多重比对 α，从而使得 α 在核心序列和任何一个其他序列方向的投影是最优的两两比对。

设 $s_1, s_2, \cdots, s_k$ 是 k 条待比对的序列，假设已知核心序列是 s_c（$1 \leqslant c \leqslant k$），则可以利用标准的动态规划算法求出所有 s_c 和 s_i（$1 \leqslant i \leqslant k$）的最优比对。将这些序列的两两比对聚集起来，并采用“只要是空位，则永远是空位”的原则。聚集过程从某一个两两比对开始，如 s_c 和 s_1，然后逐步加上其他的两两比对。在这个过程中，逐步增加 s_c 中的空位字符，以适应其他的比对，但决不删除 s_c 中已存在的空位字符。假设在上述过程中的某一时刻，有一个由 s_c 指导的、已经建立好的部分序列的多重比对，接下来就是加入一个新的、与 s_c 两两比对的序列，如果需要，则插入新的空位字符。这个过程一直进行到所有的两两比对都加入之后结束。

那么，如何选择核心序列 s_c 呢？一种方法就是尝试将每一个序列分别作为核心序列，按上述过程进行，取结果最好的一个。另一种方法是计算所有的两两比对，取下式值最大的一个

$$\sum_{i \neq c} sim(s_i, s_c) \tag{3.13}$$

例 3.7　应用星形比对算法对图 3.20 所示的 5 条序列进行多重序列比对。

解　根据式(3.1)所示的得分函数，计算这 5 条序列中任意两条序列的最优比对得分，建立图 3.21 所示的矩阵，矩阵中的每个元素都是所对应的两条序列的最优比对得分。

S_1 = ATTGCCATT
S_2 = ATGGCCATT
S_3 = ATCCAATTTT
S_4 = ATCTTCTT
S_5 = ACTGACC

图 3.20　五条序列 s_1, s_2, s_3, s_4, s_5

	s_1	s_2	s_3	s_4	s_5
s_1		7	−2	0	−3
s_2	7		−2	0	−4
s_3	−2	−2		0	−7
s_4	0	0	0		−3
s_5	−3	−4	−7	−3	

图 3.21　图 3.20 所示的 5 条序列的最优比对得分

根据图 3.21 所示的矩阵中的数据，计算式(3.13)的得分

$$\sum_{i \neq 1} sim(s_i, s_1) = 7 + (-2) + 0 + (-3) = 2$$

$$\sum_{i \neq 2} sim(s_i, s_2) = 7 + (-2) + 0 + (-4) = 1$$

$$\sum_{i\neq 3} sim(s_i, s_3) = (-2)+(-2)+0+(-7) = -11$$

$$\sum_{i\neq 4} sim(s_i, s_4) = 0+0+0+(-3) = -3$$

$$\sum_{i\neq 5} sim(s_i, s_5) = (-3)+(-4)+(-7)+(-3) = -17$$

通过计算可以看出，当选 $s_c = s_1$ 时，式(3.13)的取值最大。接下来，根据 s_1 与其他序列的最优比对得分矩阵，得出 s_1 与其他序列的最优两两比对，见图 3.22。

S_1 ATTGCCATT　　ATTGCCATT--　　ATTGCCATT　　ATTGCCATT

S_2 ATGGCCATT　S_3 ATC-CAATTTT　S_4 ATCTTC-TT　S_5 ACTGACC--

图 3.22　s_1 与其他序列的最优两两比对

最后多重序列比对的结果见图 3.23。

ATTGCCATT--

ATGGCCATT--

ATC-CAATTTT

ATCTTC-TT--

ACTGACC----

图 3.23　例 3.7 中 5 条序列的多重序列比对结果

星形比对是一种近似的方法。可以证明，用该方法所得到的多重序列比对的代价不会大于最优多重序列比对代价的两倍。

3.3.5　CLUSTAL W 算法

ClustalW 是一种渐进的多重序列比对方法，它包括三个主要阶段：①先将多个序列进行两两比对，基于这些比对，计算得到一个距离矩阵，该矩阵反映每对序列之间的关系；②根据距离矩阵计算产生系统发生树，对关系密切的序列进行加权；③从最紧密的两条序列开始，逐步引入邻近的序列并不断重新构建比对，直到所有序列都被加入为止。如果加入的序列较多，必须加入空位以适应序列的差异。

ClustalW 的程序可以自由使用，在任何主要的计算机平台上都可以运行。在美国国家生物技术信息中心 NCBI 的 FTP 服务器上可以找到下载的软件包，在欧洲生物信息学研究所 EBI 的主页还提供了基于 Web 的 ClustalW 服务，用户可以把序列和各种要求通过表单提交到服务器上，服务器把计算的结果用 Email 返回给用户。EBI 的 ClustalW 网址是：http://www.ebi.ac.uk/clustalw/。ClustalW 对用户输入序列的格式和输出格式的选择比较灵活，可以是 FASTA 格式，也可是其他格式。

例 3.8　表 3.16 是中国不同地区发现的 25 个 H9N2 型禽流感病毒在 GenBank 中的检索号，现在用 EBI 的在线 ClustalW 对这 25 个 H9N2 型禽流感病毒进行多序列比对分析。

解　(1) 根据表 3.16 所示的 25 个 H9N2 型禽流感病毒在 GenBank 中的检索号，到 GenBank 数据库中下载相应的蛋白质序列，以 FASTA 格式存储在某一文件中。

表 3.16　中国不同地区发现的 25 个 H9N2 型禽流感病毒在 GenBank 中的检索号

地　区　名	检　索　号	地　区　名	检　索　号	地　区　名	检　索　号
河北	ABI94782.1	广东-1	AAK62979.1	南京-1	AAY52512.1
宁夏	AAY52513.1	广东-2	AAY52500.1	南京-2	AAY52511.1
北京	ACB70206.1	广东-3	AAY52499.1	山东-1	ACG59777.1
河南-1	ABL61477.1	广东-4	AAY52498.1	山东-2	ACG58427.1
河南-2	AAY52508.1	广东-5	AAY52497.1	深圳	AAY52518.1)
黑龙江-1	AAY52505.1	广东-6	AAY52496.1	福建	ABV47278.1
黑龙江-2	AAY52504.1	吉林	AAY52509.1	石家庄	AAY52517.1
广西-1	AAY52502.1	江苏	AAL65235.1		
广西-2	AAY52501.1	上海	AAY52516.1		

(2) 根据 EBI 的 ClustalW 网址 http://www.ebi.ac.uk/clustalw/上网，进入 ClustalW 界面，如图 3.24 所示。

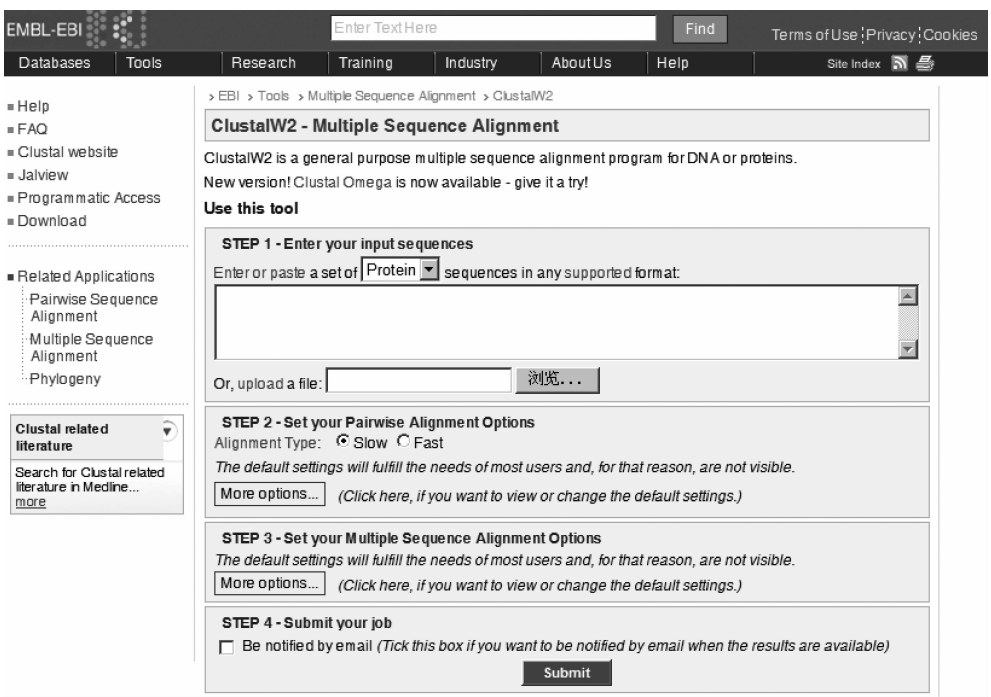

图 3.24　EBI 在线多重比对软件 ClustalW 的工作界面

(3) 将第(1)步从 GenBank 中下载的 25 条序列复制到图 3.24 中的输入框中，选择相应的参数，单击“submit”，得到如图 3.25 所示的多重序列比对结果。

(4) 图 3.25 只是比对结果的一部分，结果中氨基酸的相似程度用不同的符号来表示：“*”号表示这一组蛋白质序列中相应位置的氨基酸不发生改变，氨基酸高度保守；“:”号表示在少数蛋白质序列中氨基酸发生改变，氨基酸保守程度差些；“.”号表示有较多的序列中氨基酸发生改变，氨基酸保守程度更差些；而没有上述符号标志的氨基酸位点差别较大。

(5) 通过对表 3.16 所示的 25 条 H9N2 型禽流感病毒序列进行多重序列比对分析，可

以寻找在凝集素蛋白质中氨基酸组成相对保守和序列差别较大的区域，其中序列保守区域对应在凝集素蛋白质中相对稳定的结构，如酶的活性中心区域，而序列差别较大的区域可能是导致流感病毒产生耐药性的主要原因。

```
广东_1          MEAVSLITILLVVTASNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
河北            ----SLIAILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 56
福建            ----------LVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 50
广东_5          METVSLITILLVATVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
河南_1          --------YSLVVTASNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 52
宁夏            MEVVSLMTILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
黑龙江_1        MEVVSLITILLVATVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
吉林_1          MEVISLITILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
广东_6          MEVVSLITILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
河南_2          MEVVSLITILLVGTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
北京            MEVVSLITILLVVTVSNADKICIGHQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
山东_1          MEAVSLITILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
山东_2          MEAVSLITILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
广西_1          MEVLSLITILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
广西_2          MEVLSLITILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
广东_3          MEALSLITILLVVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
上海            MEVVSLITILLAVTVSNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
广东_2          MKAVPLITILLVVTASNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
深圳            MKAVPLITILLVVTASNADKICIGYQSTNSTETVDTLTENNVPVTHAKELLHTEHNGMLC 60
南京_1          METTSLITILLLVTTSNADKICIGYQSTNSTETVDTLAENNVPVTHAKELLHTEHNGMLC 60
南京_2          METTSLITILLLVTTSNADKICIGYQSTNSTETVDTLAENNVPVTHAKELLHTEHNGMLC 60
江苏            METISLITILLVVTASNADKICIGYQSTNSTETVDTLTESNVPVTHAKELLHTEHNGMLC 60
石家庄          METISLITILLVVTASNADKICIGYQSTNSTETVDTLTESNVPVTHAKELLHTEHNGMLC 60
黑龙江_2        METKAIIAALLMVTAANADKICIGYQSTNSTETVDTLTESNVPVTHTKELLHTEHNGILC 60
                         *  *.:********:************:*.******:**********:**
```

图 3.25　25 条 H9N2 型禽流感病毒序列多重序列比对部分结果