全局比对与动态规划

前言

学过生信的肯定知道 **Needleman—Wunsch** 算法和 **Smith—Waterman** 算法，一个用来进行全局比对，一个用来进行局部比对。单纯看算法抽象后的算法公式的话也不复杂，用两个短序列来套公式计算的话，画过箭头的同学都知道不难。本文简单讲述 **Needleman—Wunsch** 算法是如何利用动态规划来寻找两序列最大相似度。

注：有些数学公式没渲染出来可点击链接查看：https://www.jianshu.com/p/002bbebcaaef

注：代码实现见 https://www.cnblogs.com/huanping/p/11273391.html

动态规划

动态规划是数学上的优化方法，也是计算机编程的一种方法。对于一名生物狗，从数学层面去看动态规划，就太过硬核了。我们就从计算机编程角度理解动态规划就好了。

动态规划是一种解决问题的方法，将一个复杂大问题拆分为几个小的问题，从几个小问题的最优解推出大问题的最优解。当然这不是说将一个大问题拆分了就能用动态规划了。同时问题也需要满足两个性质，这两个性质后面再讲。先回到我的主要探讨问题 **全局比对** 上：对于两条蛋白质或者核酸序列来说，寻找它们的最大相似度，如何拆分成几个小问题呢？（提示：矩阵、算法公式）

拆分问题

就这样凭空想，如何拆分，我觉得我是想破脑袋也想不了T_T. 所以我们按照，学全局比对的套路手动来一遍序列比对操作，从中看看如何拆分问题。

为方便计算，得分矩阵就弄个简单点的：

N	A	T	G	C	–

N	A	T	G	C	–
A	10	5	5	5	–5
T	5	10	5	5	–5
G	5	5	10	5	–5
C	5	5	5	10	–5
–	–5	–5	–5	–5	0

也就说完全匹配得10分，不全得5分，匹配到空位–5分，空位对空位一开始用来初始化时，才有意义。在之后比对中空位对空位没有意义。

先给出公式（d为gap罚分，即碱基比对空格的得分）：

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \end{cases}$$

现假设我们有两条序列Seq1: CATCCCCCAAA 和 Seq2: CAAAA。长度分别为11，5。根据它们的长度画一个合适的二维矩阵，填充好初始数据：

	-	C	A	T	C	C	C	C	C	A	A	A
-	0	-5	-10	-20	-25	-30	-35	-40	-45	-50	-55	-60
C	-5											
A	-10											
A	-15											
A	-20											
A	-25											

huangsh

加粗的分别是 Seq1,Seq2，其中 – 代表空格。一开始空格对空格，我们初始化，记$F(0,0)=0$，同时记空格分别对应的行和列是第0行，第0列。然后第1行，第1列分别是两条首字母对应的行列。用(i,j)代表（行，列），i,j能有多大，取决于序列对的长度。$s(x_i,y_i)$的分数是一条序列中第i个碱基和另一条序列中第j个碱基比对的分数。放在矩阵中就是了第i行的碱基和第j列的碱基比对结果。

按照$F(i,j)$函数公式，$F(i,j)$的值由左上的值($F(i-1,j-1)$),上方值($F(i-1,j)$)和左方值($F(i,j-1)$)确定。
（注:这里的左上，上，左是一个相对位置，在当前设定矩阵坐标系中才有意义，这里是将 $(0,0)$ 位置放在了左上方，向右和向下延伸。但其实，你也可以将出放在其它位置，比如右下方）

极端比对

我们看下第0行，它们只能由左方的值得到。我们算几个值：

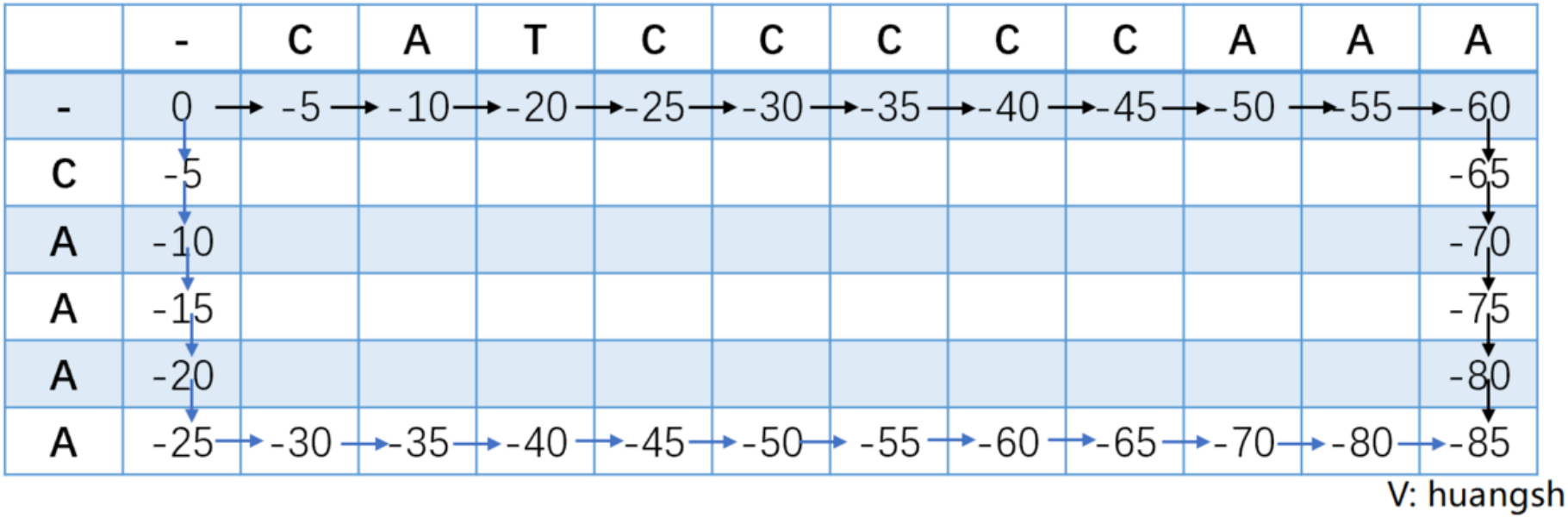
$$F(0,1)=F(0,1-1)-5=0-5$$
$$F(0,2)=F(0,2-1)-5=-5-5=-10$$
$$F(0,3)=F(0,3-1)-5=-10-5=-15$$

第0列，只能由上方的值得到，计算同理，不再演示了。
因为第0行和0列，很容易就得出来了，所以一般画矩阵时，就顺便一起画出来了。

你应该知道，往右或往下延申，则比对到了空格。那这是为啥?
就拿第0行的比对情况举个例子，第0行中，$i=0$已经固定了，它代表一条序列（后面称为$S1$序列）的第0元素。$j=0$, $1, 2, 3...$,分别代表另一条序列（后面称为$S2$序列）第0，1，2，3...元素。序列第一个元素是序列的第一个碱基。第0个是啥玩意，第0个啥也不是，那我们就把它当做空格吧。

1. 序列比对从两条序列的第 $(0, 0)$ 元素开始，两个空格比对，得分为0。
2. $i=0$固定，一直表示$S1$序列第0个元素，但是j是一直增加的，所以$j=1$时，就是用$S2$序列的的第j个元素和$S1$序列的第$i=0$个元素比对，但是$S1$序列第$i=0$元素已经和$S2$序列的第$j=0$元素比对过了，你不能重复比对啊，你也不能私自和$S1$序列的$i=1$元素比对，所以没办法，就只能和一个空格比对，产生空位罚分，在之前得分基础上上加上空位罚分。
3. $S2$序列的第$2, 3, 4...$元素都是如此，只能比对空格，产生空位罚分。

比对往右延申的过程就是行数i固定了，没有增加，但是列数j增加，又两条序列的中的元素不能重复比对，也不能比对序列其它的元素，所以就只能比对空格了。（向下延申同理。）
我们看两个极端的例子：



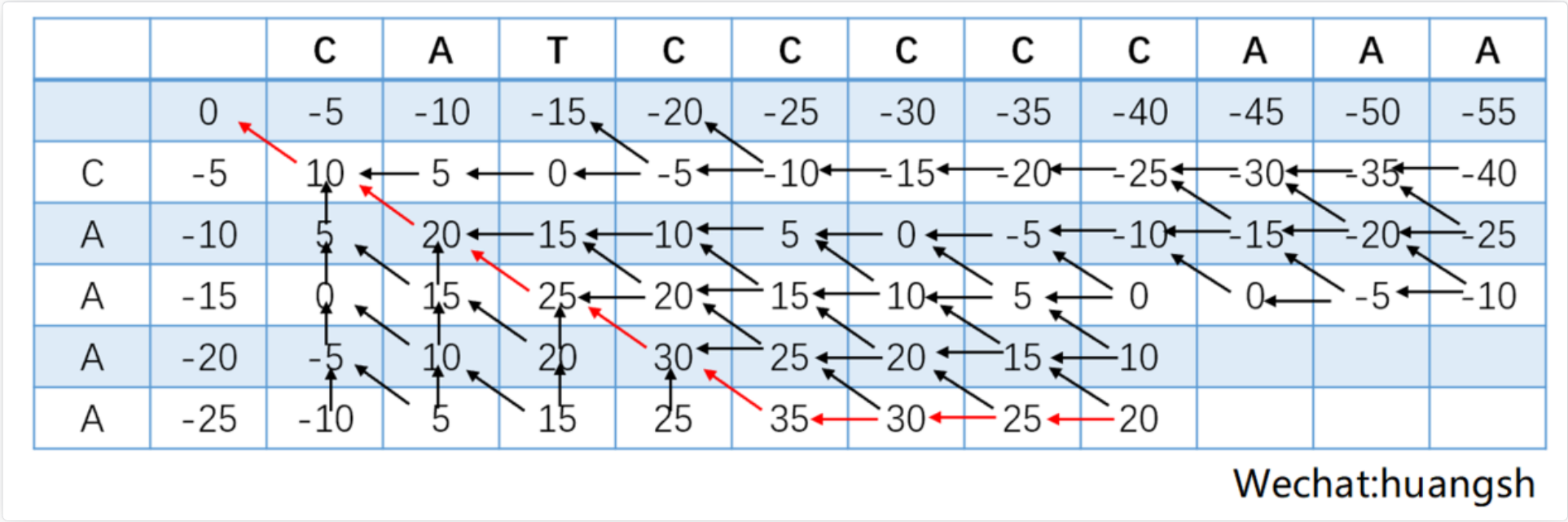

```
##黑色路线
CATCCCCCAAA-----
----------CAAAA
##红色路线
-----CATCCCCCAAA
CAAAA----------
```

> 你们说这第0行，0列的有啥用处，放这么多负分而产生极端比对。要排除这些不合适比对结果也要花费许多资源。

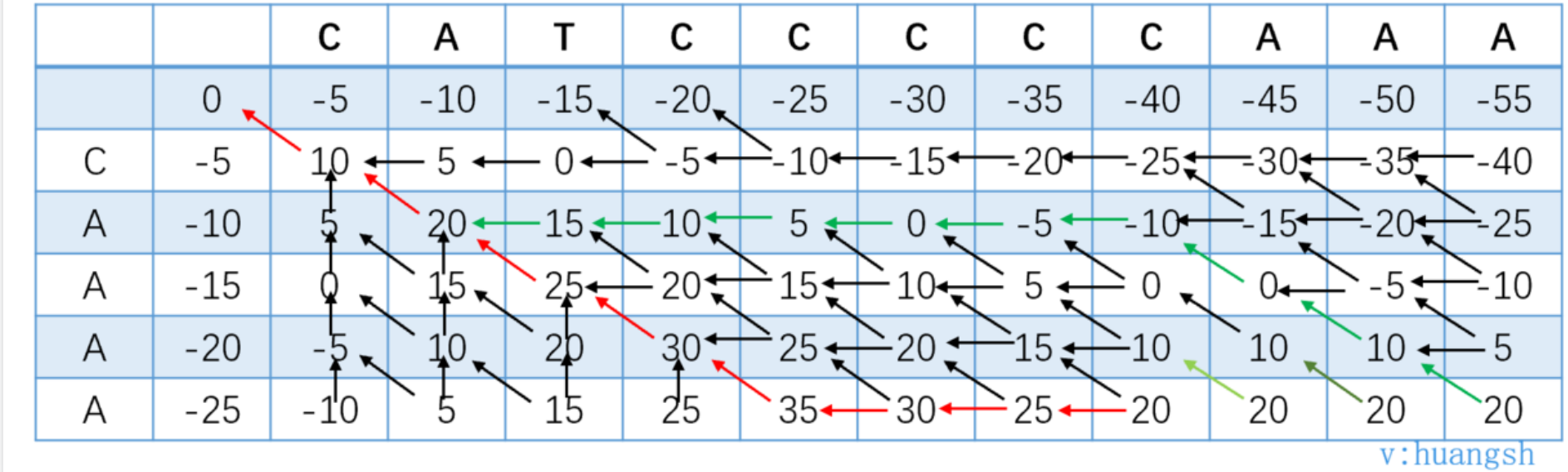
比对开始

上面进行了一次小探讨。比对直接向右或向下会产生gap。因为行列有一方被固定，没法提供可比对元素。如果要正确比对的话，要使得i和j同时增加，即向右下方延申。除了第0行和0列的元素，只有单一来源，矩阵其它的元素都有三个来源，即上方，左方，左上。如此确定$F(i, j)$的值，按照公式，选择其中最大的值。$F(m, n)$(m, n为两条序列的长度)为最终序列比对的相似得分。

既然我们想使$F(m, n)$最大，肯定不能过多的产生空位罚分吧。那么我们从一开始就一直向右下方延申，直至不能继续，能到得到最大的相似得分呢？我们试一下吧：



我们用箭头表示从哪个方向得到的最大值，两个箭头则可以从两个方向得到。从图中红色箭头的路线可以看出来，一直选择往右下延申的比对，似乎可以达到最大值。而且再比对三对元素就可以到达终点了。现在看红色箭头，似乎是全场全场最靓的仔。然而最终结果如何？我们继续把剩余几个各自补全看看。



		C	A	T	C	C	C	C	C	A	A	A
	0	-5	-10	-15	-20	-25	-30	-35	-40	-45	-50	-55
C	-5	10	5	0	-5	-10	-15	-20	-25	-30	-35	-40
A	-10	5	20	15	10	5	0	-5	-10	-15	-20	-25
A	-15	0	15	25	20	15	10	5	0	0	-5	-10
A	-20	-5	10	20	30	25	20	15	10	10	10	5
A	-25	-10	5	15	25	35	30	25	20	20	20	20

现在再看全场最靓的仔似乎被绿了？？而且是一个绿一个的。简直就是螳螂捕蝉，黄雀在后，然后渔翁得利。。。

> 贪心算法：红色路线选择的策略，从一开始，三个方向的选择，每一步，它都选择三个方向中得分最大的那个，但最后结果却不是最好的。这种在每一步都做出当前看来是最好选择的方法，而不考虑整体最优的方法，叫做贪心算法。这种方法在这里得不到最优结果。
> 穷举搜索：这是一种很直观的方法，就是把所有可能的情况都列举出来，从中找最优的。但是用于序列比对是是非常差的一种方法。从矩阵左上方到达矩阵右下方，有多少种路线？矩阵每一个小格有三种可能路线，矩阵大小$m * n$,即有3^{mn}种路线。其中m, n稍微取大一点的值，数值就爆炸了。完全不可取。

emm，写到这里，还没讲到全局比对是如何拆分问题的，现在我们就开始讨论这个问题吧。。
首先，在矩阵中当比对到第（m, n）格时，代表着比对终止。因为两条序列的每个元素都参加了比对过程，无论序列中的元素是比对了空格，还是另一条序列的某个元素。因为序列中的元素都比对完了，在比对下去就只有空格对空格了，没什么意义。所以（m, n）是

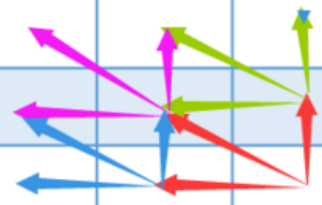
终点。

我们就从终点开始探讨如何分割大问题为小问题吧。

$$F(m,n) = max\begin{cases} F(m,n) + s(x_i, y_j) \\ F(m-1,n) + d \\ F(m,n-1) + d \end{cases}$$

其中$s(x_i, y_j)$和d是知道的,那问题$F(m,n)$则可以通过求出$F(m-1,n-1)$,$F(m,n-1)$和$F(m-1,n)$的分数得到。即一个大问题，分成了3个子问题了。三个子问题也不知道分数，就继续分子问题呗。如图：

	-	C	A	T	C	C	C	C	C	A	A	A
-	0	-5	-10	-20	-25	-30	-35	-40	-45	-50	-55	-60
C	-5											
A	-10											
A	-15											
A	-20											
A	-25											



huangsh

> 分治算法：分治的思想也是将一个问题分为几个问题来解决。这与动态规划的分离子问题有啥区别？从上面的图我们可以看出来由$F(m,n)$分割的子问题们是存在重复问题的。分治分离的子问题一般独立的不会相互重叠，而动态规划的子问题则一般发生重叠现象。

两个性质

之前空着没讲的两个性质，现在来讲一下。一个是 **最优子结构** ，一个是 **无后效性**

最优子结构：如果一个问题的最优解，可以将其分为几个子问题，然后能从子问题的最优解推出大问题的最优解。就称其有最优子结构性质。这一性质很容易从函数公式中看出来。

无后效性：给定某一阶段的状态，这一阶段以后的发展过程不受这阶段以前各段状态的影响（来自阮行止的回答）这话是什么意思呢，以我们序列比对为例。比如：当你确定了前面某个阶段的比对，比对情况如下面的比对方案的前面一部分，这时比对分数已经确定了。无后效性就是前面的比对情况对后面的比对结果的分数不会有任何影响。前面部分的比对方案的任意变动也不会影响后面部分的结果。

```
##1
CAT  CCCCC-----AAA
CA-   ----------AAA
##2
CAT  CCCCCAAA-----
CA-  AA----------A
##3
CAT  CCCCCAAA-----
CA-   ----------AAA
```

无后效性，可以让我们放心大胆的在矩阵每个格子存放最优的结果，舍弃许多不是最优的分数。因为不存在前面阶段的最差比对方案，还能与后面阶段的最差方案组合成为全局最优比对方案。

复杂度

我们来看一下 Needleman—Wunsch 算法的复杂度。从二维矩阵中，可以看到我们需要计算每一格的所占最优分值，同时也为了之后的计算方便也得储存每一个的分值。故时间复杂度和空间复杂度都是$O(mn)$。

其它

为了理解动态规划，我在知乎上这个问题什么是动态规划（Dynamic Programming）？动态规划的意义是什么？的答案下看了许多回答。许多答主都给了自己的理解。他们的回答有相似之处，也有不同。不能说谁对谁错（小声bb:也不敢说谁对谁错）

就全局比对我也谈谈我的看法：

动态规划和其它方法有许多相似的地方。要是将其它方法的定义定广一点，也能把动态规划说成其它方法。比如：

穷举搜索： 之前上文曾提到过，不过那时我们说要穷举的对象是每一个比对方案。现在我们换一个观点想一想：穷举每一个矩阵中的格子，即穷举每一个最优的$F(i, j)$。这里我们的对象就从比对路径而换成了$F(i, j)$（序列比对到第(i, j)的最大相似度）

贪心算法: 之前提到的贪心算法，我们狭隘的将最优，认为是向下、向右或向右下而能得到的分数。现在我们将最优看成相邻$F(i, j)$中最大的那个。即按照这个贪心选取，它会可能不断的对比相邻的$F(i, j)$而选择最优。直至，到最后剩下三条路径到达$F(m, n)$,选中最优的那个。

总之，不管黑猫白猫，能帮助解决问题就是好猫！选择你认为最好的一种理解。在以后多次遇到动态规划后，在慢慢完善你的认知。

标签: 算法 , 动态规划

0 0