Global alignment with general and affine gapcost

Global alignment

Match / sub columns

Score matrix:

	a	c	g	t
a	1	0	0	0
c	0	1	0	0
g	0	0	1	0
t	0	0	0	1

```
a c g t - g t c a a c g - t
a c g t c g t a - - g c t a
```

gap / indel columns

`gapcost: 0`

Note about gap cost

In general: cost of "gap block" = g(k), where k is the gap length

Our examples: g(k) = c "constant gap cost"

g(k) = b·k "linear gap cost"

Many programs: g(k) = a + b·k "affine gap cost"

Modeling gapcost

Biological observation: longer insertions and deletions (indels) are preferred to shorter indels, i.e. a “good” alignment tends to few long indels rather than many short indels ...

Can the simple algorithm for pairwise alignment be adapted to reflect this additional biological insight, i.e. a better model of biology?

Yes, we introduce the concept of a gapcost-function g(*k*) which gives the cost/penalty for a block of *k* consecutive insertions or deletions ...

Example

```
A T A C A - - - C G C A
A T - C T C C A C - C T
```

s(A,A) + s(T,T) + g(1) + s(C,C) + s(A,T) + g(3) + s(C,C) + g(1) + s(C,C) + s(A,T)

Modeling gapcost

Biological observation: longer insertions and deletions (indels) are preferred to shorter indels, i.e. a “good” alignment tends to few long indels rather than many short indels ...

Computational challenge

Can we compute an optimal global alignment with general (affine) gapcost efficiently?

Can the sir ... ed to reflect this additio ... ogy?

Yes, we introduce the concept of a gapcost-function g(k) which gives the cost/penalty for a block of k consecutive insertions or deletions ...

Example

```
A T A C A - - - C G C A
A T - C T C C A C - C T
```

s(A,A) + s(T,T) + g(1) + s(C,C) + s(A,T) + g(3) + s(C,C) + g(1) + s(C,C) + s(A,T)

Global alignment of two strings

gap/insertion   substitution   gap/deletion

$$\mathcal{A} = \begin{bmatrix} \boxed{\begin{matrix} A \\ A \end{matrix}} & \boxed{\begin{matrix} C \\ G \end{matrix}} & \boxed{\begin{matrix} - & - \\ T & A \end{matrix}} & \boxed{\begin{matrix} T \\ T \end{matrix}} & \boxed{\begin{matrix} C \\ C \end{matrix}} & \boxed{\begin{matrix} A & G \\ - & - \end{matrix}} \end{bmatrix}$$

General cost:

$$\text{cost}(\mathcal{A}) = \sum_{\text{blocks}} \text{"cost of block"}$$

Substitution cost: $s : \Sigma \times \Sigma \to \mathbb{R}$

gap cost: $g : \mathbb{N}_+ \to \mathbb{R}$

Global alignment of two strings

gap/insertion

substitution

gap/deletion

$$\mathcal{A} = \left[\begin{array}{cccccccc} A & C & - & - & T & C & A & G \\ A & G & T & A & T & C & - & - \end{array} \right]$$

The book [BA] defines a gap penalty function in definition 8.1.25 as a function g: N→R, where g(0)=0 and g(k)≥0, that is **subadditive**, i.e. where **g(k+l) ≤ g(k) + g(l)**. Why do you think that subadditivity is required?

$$\mathrm{cost}(\mathcal{A}) = \sum_{\text{blocks}} \text{"cost of block"}$$

Substitution cost: $s : \Sigma \times \Sigma \to \mathbb{R}$

gap cost: $g : \mathbb{N}_+ \to \mathbb{R}$

Computing optimal cost with general gap cost

Intuition

$$S(i,j) = \max_{\text{last block}} \left\{ \text{"cost of last block"} + \text{"opt. cost of rest"} \right\}$$

Formalization

$$S(i,j) = \max \begin{cases} 0 \\ \overbrace{s(A[i],B[j]) + S(i-1,j-1)}^{M(i,j)} & \left[\begin{matrix} \sim\sim\sim & \boxed{\begin{matrix} A[i] \\ B[j] \end{matrix}} \\ \sim\sim\sim & \end{matrix} \right] \\ \overbrace{\max_{0<k\leq i} \left[S(i-k,j) - g(k) \right]}^{D(i,j)} & \left[\begin{matrix} \sim\sim\sim & \boxed{\begin{matrix} A[i-k+1] \cdots A[i] \\ - \; - \; - \; - \end{matrix}} \\ \sim\sim\sim & \end{matrix} \right] \\ \overbrace{\max_{0<k\leq j} \left[S(i,j-k) - g(k) \right]}^{I(i,j)} & \left[\begin{matrix} \sim\sim\sim & \boxed{\begin{matrix} - \; - \; - \; - \\ B[j-k+1] \cdots B[j] \end{matrix}} \\ \sim\sim\sim & \end{matrix} \right] \end{cases}$$

Computing optimal cost with general gap cost

Can be implemented using dynamic programming

Running time: $\left(\frac{n}{2}\right)^2 \cdot \frac{n}{2} \leq T(n) \leq n^2 \cdot n \qquad O(n^3)$

Space consumption: $O(n^2)$

$$S(i,j) = \max \begin{cases} 0 \\ \overbrace{s(A[i], B[j]) + S(i-1, j-1)}^{M(i,j)} & \left[\begin{array}{c|c} \sim\sim\sim & A[i] \\ \sim\sim\sim & B[j] \end{array}\right] \\ \overbrace{\max\limits_{0<k\leq i} \left[S(i-k, j) - g(k) \right]}^{D(i,j)} & \left[\begin{array}{c|c} \sim\sim\sim & A[i-k+1] \cdots A[i] \\ \sim\sim\sim & - \; - \; - \; - \end{array}\right] \\ \overbrace{\max\limits_{0<k\leq j} \left[S(i, j-k) - g(k) \right]}^{I(i,j)} & \left[\begin{array}{c|c} \sim\sim\sim & - \; - \; - \; - \\ \sim\sim\sim & B[j-k+1] \cdots B[j] \end{array}\right] \end{cases}$$

Computing optimal cost with affine gap cost

$$g(k) = \alpha \cdot k + \beta \quad , \quad \alpha, \beta \geq 0$$

Intuition

... as generel gap cost but reduce time to compute $D(i,j)$ and $I(i,j)$. Let us consider $D(i,j)$ ...

Trick

Consider the best deletion-block, two possibilities:

① A new block
$$\left[\overbrace{\begin{matrix}\sim A[i-1] \\ \sim B[j]\end{matrix}}^{M(i-1,j)\,-\,(\alpha+\beta)} \boxed{\begin{matrix}A[i] \\ -\end{matrix}}\right] \qquad \left[\overbrace{\begin{matrix}\sim\sim\sim \quad - \\ \sim\sim\sim \ B[j]\end{matrix}}^{I(i-1,j)\,-\,(\alpha+\beta)} \boxed{\begin{matrix}A[i] \\ -\end{matrix}}\right]$$

② Continuation of existing block
$$\left[\overbrace{\begin{matrix}\sim \cdot A[i-k] & \cdots A[i-1] \\ \sim \cdot \ B[j] & - \ - \ -\end{matrix}}^{D(i-1,j)\,-\,\alpha} \begin{matrix}A[i] \\ -\end{matrix}\right]$$

Computing optimal cost with affine gap cost

Because
$S(i-1,j) = \max\{M(i-1,j), I(i-1,j), D(i-1,j)\}$
and $D(i-1,j)-(\alpha+\beta) \le D(i-1,j)-\alpha$

$$g(k) = \alpha \cdot k + \beta$$

Hence,

$$D(i,j) = \max\{M(i-1,j)-(\alpha+\beta),\ I(i-1,j)-(\alpha+\beta),\ D(i-1,j)-\alpha\}$$

$$= \max\{S(i-1,j)-(\alpha+\beta),\ D(i-1,j)-\alpha\}$$

Trick

Consider the best deletion-block, two possibilities:

① A new block

$$\left[\overbrace{\begin{matrix}\sim A[i-1] \\ \sim B[j]\end{matrix}}^{M(i-1,j)-(\alpha+\beta)} \boxed{\begin{matrix}A[i] \\ -\end{matrix}}\right] \qquad \left[\overbrace{\begin{matrix}\sim\sim\sim\ - \\ \sim\sim\sim\ B[j]\end{matrix}}^{I(i-1,j)\ -\ (\alpha+\beta)} \boxed{\begin{matrix}A[i] \\ -\end{matrix}}\right]$$

② Continuation of existing block

$$\left[\begin{matrix}\sim A[i-k] \\ \sim B[j]\end{matrix} \boxed{\overbrace{\begin{matrix}\cdots A[i-1] \\ - \ - \ -\end{matrix}}^{D(i-1,j)\ -\ \alpha} \begin{matrix}\vdots\ A[i] \\ \vdots\ -\end{matrix}}\right]$$

Computing optimal cost with affine gap cost

$$S(i,j) = \max \begin{cases} 0 & i = 0 \text{ and } j = 0 \\ S(i-1,j-1) + s(A[i],B[j]) & i > 0 \text{ and } j > 0 \\ D(i,j) & i > 0 \text{ and } j >= 0 \\ I(i,j) & i >= 0 \text{ and } j > 0 \end{cases}$$

$$D(i,j) = \max \begin{cases} S(i-1,j) - (\alpha+\beta) & i > 0 \text{ and } j >= 0 \\ D(i-1,j) - \alpha & i > 1 \text{ and } j >= 0 \end{cases}$$

$$I(i,j) = \max \begin{cases} S(i,j-1) - (\alpha+\beta) & i >= 0 \text{ and } j > 0 \\ I(i,j-1) - \alpha & i >= 0 \text{ and } j > 1 \end{cases}$$

Can be implemented using dyn.prog/memorization using 3 tables of size $\Theta(n \cdot m)$

Computing optimal cost with affine gap cost

$$S(i,j) = \max \begin{cases} 0 & i = 0 \text{ and } j = 0 \\ S(i-1,j-1) + s(A[i],B[j]) & i > 0 \text{ and } j > 0 \\ D(i,j) & i > 0 \text{ and } j >= 0 \\ I(i,j) & i >= 0 \text{ and } j > 0 \end{cases}$$

$$D(i,j) = \max \begin{cases} S(i-1,j) - (\alpha+\beta) & i > 0 \text{ and } j >= 0 \\ D(i-1,j) - \alpha & i > 1 \text{ and } j >= 0 \end{cases}$$

$$I(i,j) = \max \begin{cases} S(i,j-1) - (\alpha+\beta) & i >= 0 \text{ and } j > 0 \\ I(i,j-1) - \alpha & i >= 0 \text{ and } j > 1 \end{cases}$$

Can be implemented using dyn.prog/memorization using 3 tables of size $\Theta(n \cdot m)$

Computing optimal cost with affine gap cost

Time and space is O(nm). An optimal alignment can be retrieved by backtracking in time O(n).

$$S(i,j) = \max \begin{cases} 0 & i = 0 \text{ and } j = 0 \\ S(i-1,j-1) + s(A[i],B[j]) & i > 0 \text{ and } j > 0 \\ D(i,j) & i > 0 \text{ and } j >= 0 \\ I(i,j) & i >= 0 \text{ and } j > 0 \end{cases}$$

$$D(i,j) = \max \begin{cases} S(i-1,j) - (\alpha+\beta) & i > 0 \text{ and } j >= 0 \\ D(i-1,j) - \alpha & i > 1 \text{ and } j >= 0 \end{cases}$$

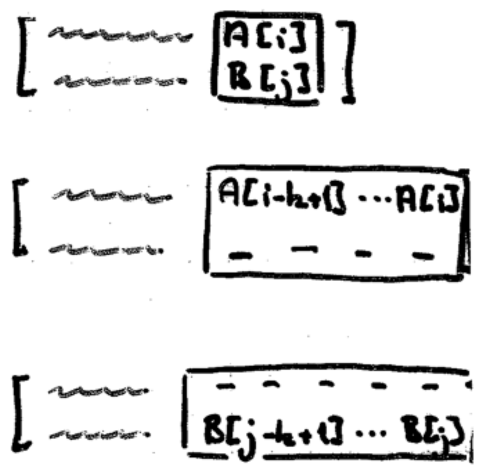
$$I(i,j) = \max \begin{cases} S(i,j-1) - (\alpha+\beta) & i >= 0 \text{ and } j > 0 \\ I(i,j-1) - \alpha & i >= 0 \text{ and } j > 1 \end{cases}$$

Can be implemented using dyn.prog/memorization using 3 tables of size $\Theta(n \cdot m)$

Global alignment with convex gap cost

g(k) = “a convex function, fx log(k) or sqrt(k)”

$$S(i,j) = \max \begin{cases} 0 \\ \overbrace{s(A[i],B[j]) + S(i-1,j-1)}^{M(i,j)} \\ \overbrace{\max_{0<k\le i} [\, S(i-k,j) - g(k) \,]}^{D(i,j)} \\ \overbrace{\max_{0<k\le j} [\, S(i,j-k) - g(k) \,]}^{I(i,j)} \end{cases}$$



In cubic time using above recursion, but it can be done in O(nmlog(n))