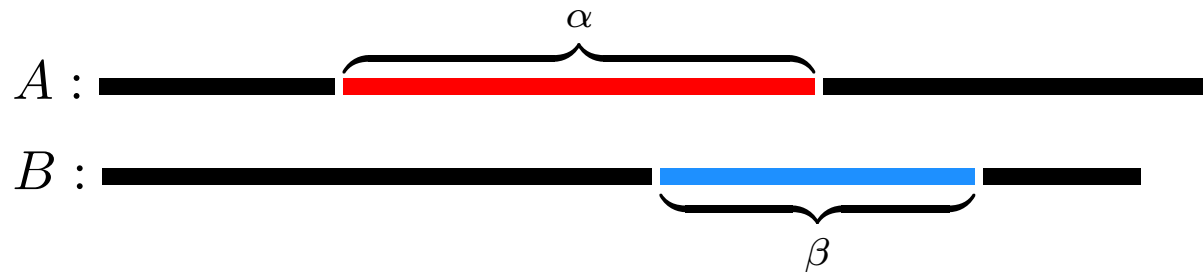

Local Alignment

Local Alignment

Doolittle: "...one must be alert to regions of similarity even when they occur embedded in an overall background of dissimilarity."



Local alignment:

Given two strings $A[1..n]$ og $B[1..m]$ and a similarity measure $Sim(A, B)$. Find a pair of substrings α of A and β of B , such that $Sim(\alpha, \beta)$ is maximal over all possible choiches α of β , i.e.:

$$LocalSim(A, B) = \max_{\alpha, \beta} Sim(\alpha, \beta)$$

— Local alignment, a few thoughts —

If $Sim(A, B)$ is defined by:

- score $s(a, b)$ of a “match-column” $\begin{pmatrix} a \\ b \end{pmatrix}$
- penalty α of a “gap-column” $\begin{pmatrix} a \\ - \end{pmatrix}$ or $\begin{pmatrix} - \\ b \end{pmatrix}$

Then computing $Sim(A, B)$ takes time $O(nm)$...

How can we compute $LocalSim(A, B)$?

There are $\Theta(n^2m^2)$ possible pairs of substrings α and β . The naive solution of computing $Sim(\alpha, \beta)$ for all possible pairs takes time $\Theta(n^3m^3)$!!

Can we do better? [Smith and Waterman, 1981]

Local alignment, algorithm, part 1

We have ...

$$\alpha = A[h + 1 .. i], \text{ for } 0 \leq h \leq i \leq n$$

$$\beta = B[k + 1 .. j] \text{ for } 0 \leq k \leq j \leq m$$

We can thus write:

$$LocalSim(A, B) = \max_{i,j} \underbrace{\max_{h \leq i, k \leq j} Sim(A[h + 1 .. i], B[k + 1 .. j])}_{v(i,j)} = \max_{i,j} v(i, j)$$

If $v(i, j)$ is known for $0 \leq i \leq n$ og $0 \leq j \leq m$, then we can compute $LocalSim(A, B)$ in time $O(nm)$...

Special case:

$$v(0, 0) = Sim(\underbrace{A[1 .. 0]}_{\epsilon}, \underbrace{B[1 .. 0]}_{\epsilon}) = S(0, 0) = 0$$

What about the rest ...

Local alignment, algorithm, part 2

We have that $v(i, j)$ is the score of an optimal alignment ...

$$v(i, j) = \text{OPT} \left(\begin{bmatrix} \cdots A[h + 1 .. i] \cdots \\ \cdots B[k + 1 .. j] \cdots \end{bmatrix} \right)$$

We can describe this optimal score by considering the following cases:

- Hvis $h = i$ og $k = j$ så $v(i, j) = \text{Sim}(\epsilon, \epsilon) = 0$.

- Ellers ...

1. $\begin{bmatrix} \cdots A[h + 1 .. i - 1] \cdots \mathbf{A[i]} \\ \cdots B[k + 1 .. j - 1] \cdots \mathbf{B[j]} \end{bmatrix}$, then $v(i - 1, j - 1) + s(A[i], B[j])$.

2. $\begin{bmatrix} \cdots A[h + 1 .. i] \cdots - \\ \cdots B[k + 1 .. j - 1] \cdots \mathbf{B[j]} \end{bmatrix}$, then $v(i, j - 1) - \alpha$.

3. $\begin{bmatrix} \cdots A[h + 1 .. i - 1] \cdots \mathbf{A[i]} \\ \cdots B[k + 1 .. j] \cdots - \end{bmatrix}$, then $v(i - 1, j) - \alpha$.

Local alignment, algorithm, part 3

$$LocalSim(A, B) = \max_{\alpha, \beta} Sim(\alpha, \beta) = \max_{i, j} v(i, j), \text{ where } \dots$$

$$v(i, j) = \max \begin{cases} v(i-1, j-1) + s(A[i], B[j]) & i > 0 \text{ og } j > 0 \\ v(i-1, j) - \alpha & i > 0 \text{ og } j \geq 0 \\ v(i, j-1) - \alpha & i \geq 0 \text{ og } j > 0 \\ 0 & i \geq 0 \text{ og } j \geq 0 \end{cases}$$

Algorithm ...

- Fill out the v -table row by row, **time $O(nm)$**
- Find entry (i', j') , where $v(i', j') = \max v(i, j)$, **time $O(nm)$**
- Back-track from (i', j') to (h, k) , where $v(h, k) = 0$, **time $O(n + m)$**

$$Sim(\underbrace{A[h+1 .. i']}_{\alpha}, \underbrace{B[k+1 .. j']}_{\beta}) = v(i', j') = \max_{i, j} v(i, j) = LocalSim(A, B)$$