Algorithms in Genome Analysis, Spring 2023

Veli Mäkinen

Week 2

Alignments

Global alignment

- *Input*: Two sequences A and B
- *Output*: Two aligned sequences A' and B' with max alignment score S(A,B)
- Alignment means adding gaps "-" to the input sequences to make the output sequences of equal length
- Aligning "-" with "-" is not allowed
- $S(A,B) = \max_{\{alignments\ A',B'\}} \sum_{i=1}^{|A'|} s(A'[i], B[i'])$
- See the prerecorded video on rigorous ways to define
  - substitution scores s(a,b) and
  - gap penalties s(-,b) and s(-,b) equalling constant -d

Global alignment example

- A= *AGCTGAT*
- B =*GCAGACT*
- A'= *AGCTGA-T*
- B'= *-GCAGACT*
- Assume (A',B') is an optimal alignment of A and B, s(a,a)=1,s(a,b)=0 for a≠b, and s(-,a)=s(-,b)=-1.
- S(A,B)=-1+1+1+0+1+1-1+1=3.

Global alignment through dynamic programming

- S[i,j]=S(A[1..i],B[1..j])
- An optimal alignment of prefixes A[1..i] and B[1..j] can end in three ways:
  - A[i] is aligned with B[j]
  - A[i] is aligned with a gap ”-”
  - Gap ”-” is aligned with B[j]
- S[i,j]=max(
  - S[i-1,j-1]+s(A[i],B[j]),
  - S[i-1,j]+s(A[i],-),
  - S[i,j-1]+s(-,B[j]))
- S[0,j]=-jd, S[i,0]=-id as initialization, S(A,B)=S[|A|,|B|] as finalization
- See course book for an induction proof.

Global alignment traceback

- Once S[i,j] are computed for all i and j, we can reverse the decisions starting from S[|A|,|B|]:
  - If S[|A|-1,|B|-1]+s(A[i],B[j])=S[|A|,|B|], we know there is an optimal alignment ending with aligning A[i] with B[j].
  - If S[|A|-1,|B|]+s(A[i],-)=S[|A|,|B|], we know there is an optimal alignment ending with aligning A[i] with a gap.
  - If S[|A|,|B|-1]+s(-,B[j])=S[|A|,|B|], we know there is an optimal alignment ending with aligning a gap with B[j].
  - Continuing this way with one of the options above gives us an optimal aligment backwards.

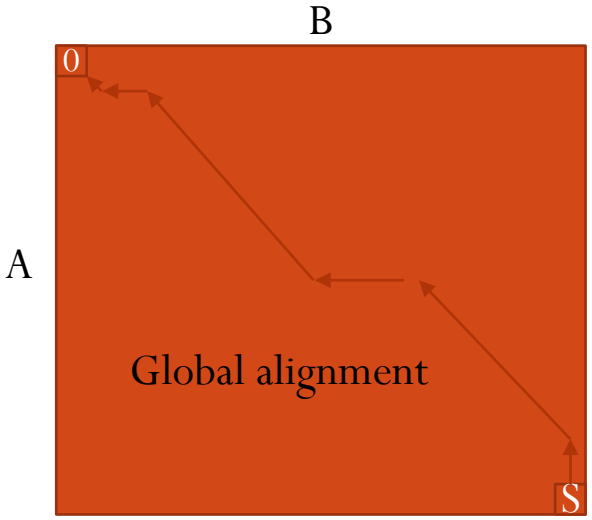
Local alignment

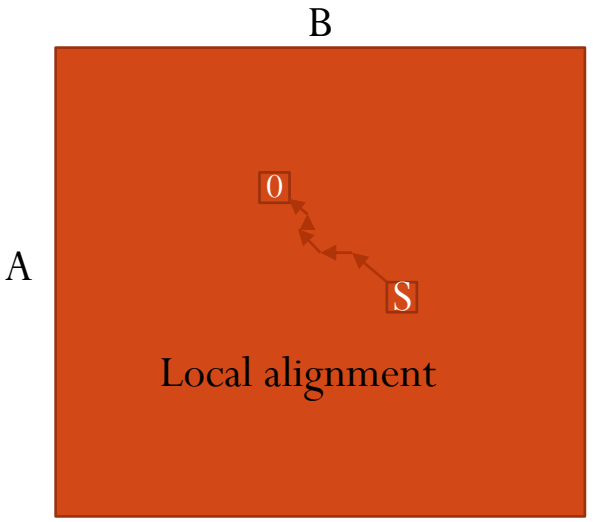
- *Input*: Two sequences A and B
- *Output*: Substrings C and D of A and B, respectively, with maximum global alignment score S(C,D)
- Naive computation in time $O(|A|^3|B|^3)$ by extracting all substring pairs and computing global alignment through dynamic programming.
- Easy speed-up to $O(|A|^2|B|^2)$ by doing global alignment computation on all suffix pairs and looking for maximum S[i,j] value.

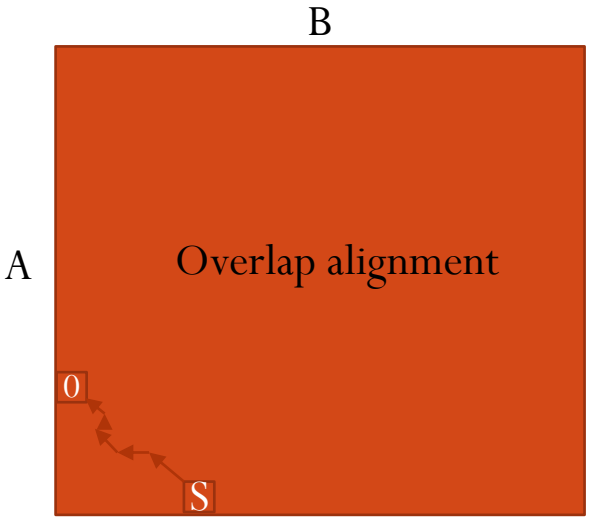
Local alignment with a twist

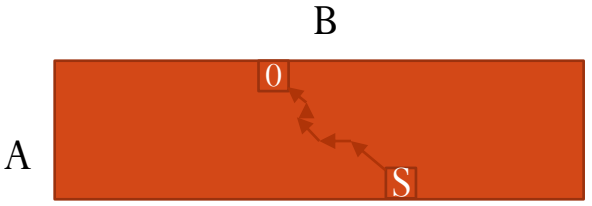
- Assume max local alignment score is non-negative.
- Let C and D be substrings yielding max score S(C,D)≥0.
- Let C' and D' be alignments of C and D with score S(C,D).
- Score s(C'[1],D'[1])≥0, as otherwise we could shorten C or D or both and get an alignment with at least as good score.
  - We can ignore alignments that start with negative score
- L[i,j]=max(0,
  - L[i-1,j-1]+s(A[i],B[j]),
  - L[i-1,j]+s(A[i],-),
  - L[i,j-1]+s(-,B[j]))
- Max L[i,j] value reveal substrings ending at A[..i] and B[..j] with highest non-negative alignment score.
- Traceback reveals the start of the substrings.
- Running time is O(|A||B|), that is, the same as for global alignment.

Variations of the theme







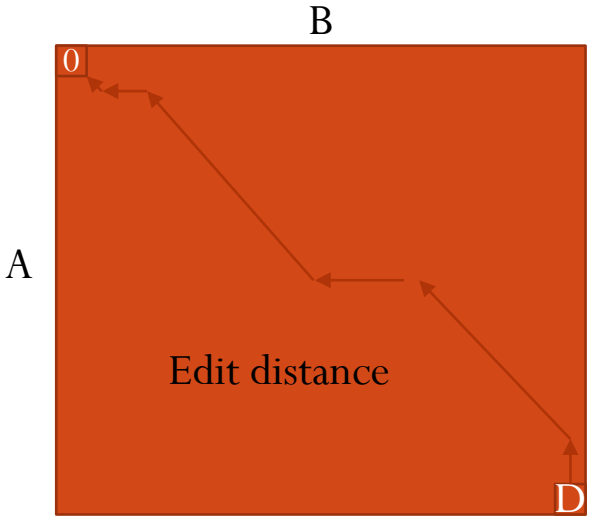


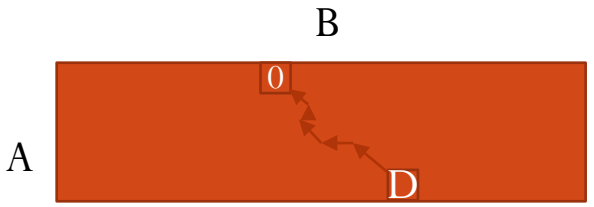
Semi-global alignment

Connection to edit distance

- An alignment can be interpreted as editing instructions to convert A into B:
  - A[i] is aligned with B[j] → Substitute A[i] with B[j]
  - A[i] is aligned with a gap ”-” → Delete A[i]
  - Gap ”-” is aligned with B[j] → Insert B[j]
- D[i,j]=min(
  - D[i-1,j-1]+(A[i]=B[j]?0:1),
  - D[i-1,j]+1,
  - D[i,j-1]+1)
- D[0,j]=j, D[i,0]=i as initialization, D(A,B)=D[|A|,|B|] as finalization
- Here D(A,B) is the unit cost edit distance.

More variations of the theme





Approximate string matching