BMEG3105 (24/25 fall) - Data analytics for personalized genomics and precision medicine | Week 2 Short Lecture: Sequence data & dynamic programming

Lecturer: Yu LI (李煜) from CSE | Liyu95.com, liyu@cse.cuhk.edu.hk | Friday, Sept 13th 2024 Scriber: Yunwen ZHANG (1155173743)

From last lecture

Comments on survey inputs

- On discrepancy in perceived difficulty, pace and detail
  - advised that students pay effort to understand the **concepts**, as staying scatterbrained throughout the time leads to bad performance at exams;
  - advised that students show up at the **revision lectures** where he will talk about what the exams will cover.

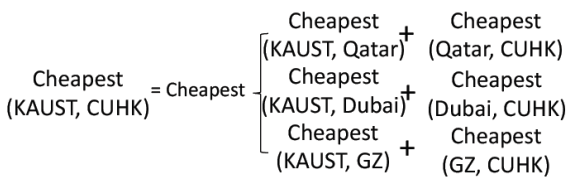
Last lecture recap

- Data types - the 6 basic types + **multi-modality data** + **unknown data** (i.e. data type unknown before we see the data)
- Python programming - way of communication with computer, essentially = a messaging app + a translating app
- Sequence data (DNA, RNA, protein), what we can do to them (sequencing & sequence alignment), & what we can obtain from them (similarity → possible relations between species)
  - Pairwise alignment: when facing long sequences, possible alignments too much → enumeration fails. Therefore here we are in front of dynamic programming.

Today's topic: dynamic programming

Analogy: Cheapest flight problem

- Problem nature: optimization (comparing)
- Constraint: no direct flight, or breaking is cheaper
- → problem reduced to: finding cheapest total transportation cost between routes corresponding to each transfer place - i.e.,



  - such breaking down of a problem into sub-problems is called **dynamic programming** (DP).

Resolving a simple example of sequence alignment using DP

- Problem:

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10

Input sequences: ACCG, ACG

Questions:
Optimal alignment score: F(ACCG, ACG)??
Optimal alignment??

gut feeling: should be ACCG - AC_G or ACCG - A_CG.

▼ Part 1: finding the optimal score

- Breaking it down into sub-problems and solving them recursively:
  1. the **last pair** of alignment: 3 options only (G-G, G-gap, gap-G)
     - thus, the final alignment score is partially specified as:

$$F(ACCG, ACG) = \text{Best} \begin{cases} F(ACC, ACG) + F(G, \_) = F(ACC, ACG) - 10 \\ F(ACCG, AC) + F(\_, G) = F(ACCG, AC) - 10 \\ F(ACC, AC) + S(G, G) = (ACCG, AC) + 2 \end{cases}$$

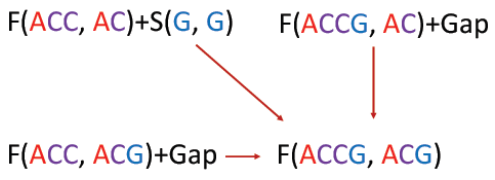
     - effect: reducing base count roughly 1 base at a time
  2. **repeat that for each** of F(ACC, ACG), F(ACCG, AC) and (ACC, AC)
     - taking F(ACC, AC),

$$F(ACC, AC) = \text{best} \begin{cases} F(AC, AC) + F(C, \text{gap}) \\ F(ACC, A) + F(\text{gap}, C) \\ F(AC, A) + F(C, C) \end{cases}$$

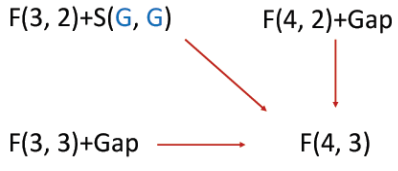
$$= \text{best} \begin{cases} F(AC, AC) - 10 \\ F(ACC, A) - 10 \\ F(AC, A) + 2 \end{cases}$$

     - and so forth.
  3. continue all those until you resolve the whole problem.

- a more systematic way to visualize this process:
  - observe that each 3-possibility large bracket correspond to 3 fixed directions. e.g.,



  abstracting into number of bases, essentially,



  - the **dynamic programming table**:

		A	C	C	G
	0 →	-10 →	-20 →	-30 →	-40
A	-10	2 →	-8 →	-18 →	-28
C	-20	-8	4 →	-6 (-6)+2 =-4	-16
G	-30	-18	-6	-3	??

  giving the optimal alignment score F(4,3) = -4.

- How to fill in the DP table?
  1. Start with the first empty row and the first empty column, which are alignment of ‘gap’s to every possibility:

		A	C	C	G
	0	-10	-20	-30	-40
A	-10				
C	-20				
G	-30				

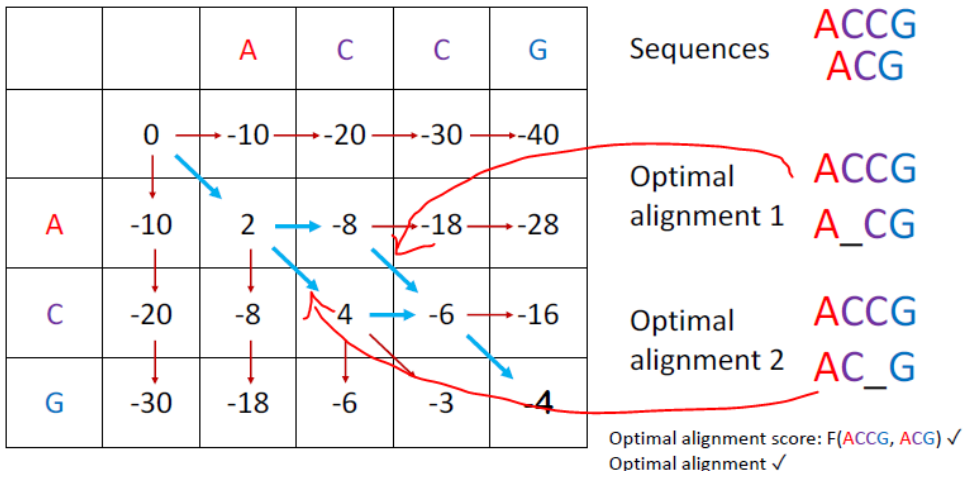
  2. Fill the rest layer by layer. fill in one cell only the largest of the three addition results.

▼ Part 2: finding the optimal alignment (the path)

1. Trace up (change into another color) the arrows that lead to the final cell. There may be more than one such path. Each path corresponds to an optimal alignment. Preserve all such paths.

		A	C	C	G
	0 →	-10 →	-20 →	-30 →	-40
A	-10	2 →	-8 →	-18 →	-28
C	-20	-8	4 →	-6 →	-16
G	-30	-18	-6	-3	**-4**

2. Translate the arrows into alignments, starting from the last cell. Horizontal arrow translates to a gap in the sequence you put as the vertical axis while the horizontal sequence proceeding by one letter. Vertical arrow the reverse. Diagonal arrow means both sequences proceeding by one letter.



- Sanity check: consistent with our gut feeling.
- Reflection on computation time, for two sequences with length n
  - enumerations requires $\binom{2n}{n} = \frac{(2n)!}{(n!)^2}$ times of computation
  - DP (n by n table) requires $(n+1)^2 \cdot 3 = 3(n+1)^2$ times of computation