CSE8803/CX4803 - Homework 1 - Spring 2022

1 Global and local alignments [10 pts]

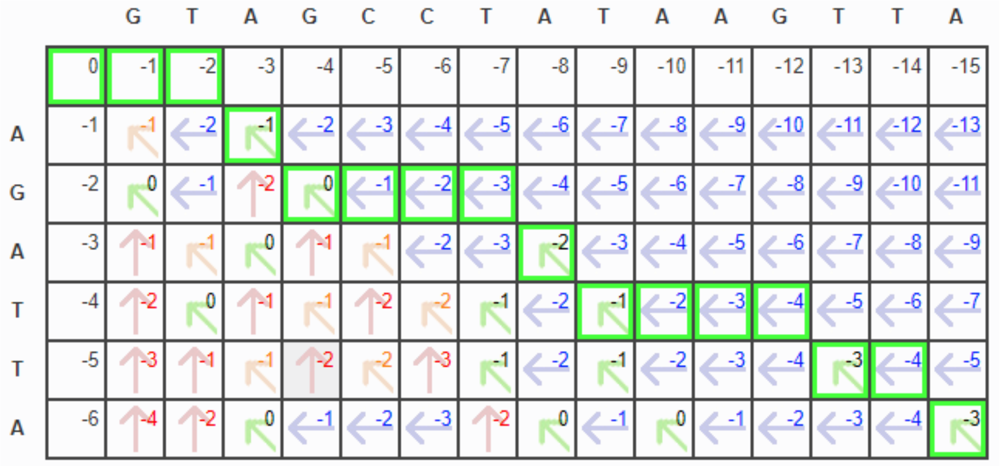
Consider two DNA sequences $\boldsymbol{x}$ = AGATTA and $\boldsymbol{y}$ = GTAGCCTATAAGTTA. In this question, we will align the two sequences using a score of +1 for a match, -1 for a mismatch, and -1 for insertion/deletion (i.e., gap). Note that in this problem, we will align sequences by *maximizing* the alignment score (instead of *minimizing* the alignment cost).

a. [5 pts] Align the two sequences using the *global* alignment algorithm introduced in the lecture. You need to i) compute the final alignment score, ii) fill out the following dynamic programming table (i.e., fill in *all* cells with its alignment scores), and iii) highlight the path of the optimal alignment using backtrace.

		G	T	A	G	C	C	T	A	T	A	A	G	T	T	A
	0															
A																
G																
A																
T																
T																
A																

Solution: Score: -3

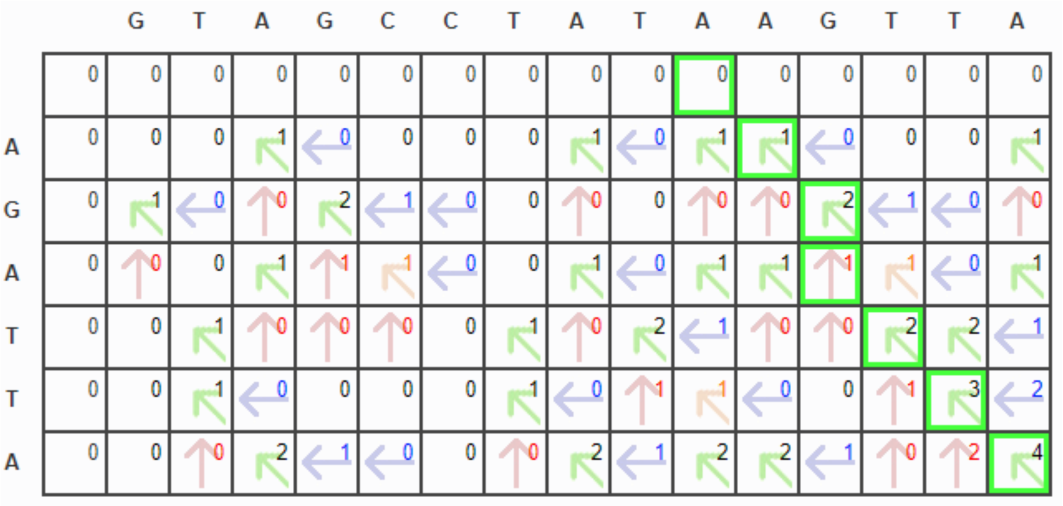
There are multiple paths of optimal alignment. One example is given below.



		G	T	A	G	C	C	T	A	T	A	A	G	T	T	A
	0	**-1**	**-2**	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15
A	-1	↖-1	←-2	**↖-1**	←-2	←-3	←-4	←-5	←-6	←-7	←-8	←-9	←-10	←-11	←-12	←-13
G	-2	↖0	←-1	↑-2	**↖0**	**←-1**	**←-2**	**←-3**	←-4	←-5	←-6	←-7	←-8	←-9	←-10	←-11
A	-3	↑-1	↖-1	↖0	↑-1	↖-1	←-2	←-3	**↖-2**	←-3	←-4	←-5	←-6	←-7	←-8	←-9
T	-4	↑-2	↖0	↑-1	↖-1	↑-2	↖-2	↖-1	←-2	**↖-1**	**←-2**	**←-3**	**←-4**	←-5	←-6	←-7
T	-5	↑-3	↑-1	↖-1	↑-2	↖-2	↑-3	↖-1	←-2	↖-1	←-2	←-3	←-4	**↖-3**	**←-4**	←-5
A	-6	↑-4	↑-2	↖0	←-1	←-2	←-3	↑-2	↖0	←-1	↖0	←-1	←-2	←-3	←-4	**↖-3**

b. [5 pts] Align the two sequences using the *local* alignment algorithm introduced in the lecture, then compute the final alignment score, fill out the dynamic programming table, and highlight the backtrace path as in (a).

Solution: Score: 4



		G	T	A	G	C	C	T	A	T	A	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	↖1	←0	0	0	0	↖1	←0	↖1	↖1	←0	0	0	↖1
G	0	↖1	←0	↑0	↖2	←1	←0	0	↑0	0	↑0	↑0	↖2	←1	←0	↑0
A	0	↑0	0	↖1	↑1	↖1	←0	0	↖1	←0	↖1	↖1	↑1	↖1	←0	↖1
T	0	0	↖1	↑0	↑0	↑0	0	↖1	↑0	↖2	←1	↑0	↑0	↖2	↖2	←1
T	0	0	↖1	←0	0	0	0	↖1	←0	↑1	↖1	←0	0	↑1	↖3	←2
A	0	0	↑0	↖2	←1	←0	0	↑0	↖2	←1	↖2	↖2	←1	↑0	↑2	↖4

Rubric. For both (a) and (b): -1 if wrong final alignment score; -1 for each missing highlighted cell or incorrect cell alignment scores in traceback, up to a maximum of -4.

2 Number of Alignments [10 pts]

Given two protein sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ of the same length n, show that the total number of different *non-boring* and *no-crossing* alignments is at least exponential with respect to n. *Non-boring* means a gap is never aligned to a gap. If there exist gaps at the same position in the two sequences in the alignment, it is considered the same as the alignment after removing the gaps. *No-crossing* is explained in the lecture.

Hint: you may find the following equation helpful: $(1+z)^n = \sum_{d\geq 0} \binom{n}{d} z^d$.

Solution:

This problem is essentially about how do we place the gaps in the alignment, because of the "no-crossing" requirement we do not change the relative order of the amino acids. The length of the alignment is at most $2n$, as we can add at most n gaps. Considering non-boring alignments with n gaps: there are $\binom{2n}{n}$ ways to insert or append gaps to $\boldsymbol{x}$. For every combination of the $\boldsymbol{x}$ sequence with n gaps, there is only one way to place n gaps to y to make the alignment non-boring, that is, the n gaps placed to y can only the at the positions of non-gap characters in $\boldsymbol{x}$.

Now if the length of the alignment is $2n-1$, which means, we add $n-1$ gaps, we have $\binom{2n-1}{n-1}\binom{n}{n-1}$ number of different alignments.

Summing up all alignment lengths from $2n$ to n, the total number of different alignments is:

$$\begin{aligned}
N &= \binom{2n}{n}\binom{n}{n} + \binom{2n-1}{n-1}\binom{n}{n-1} + \binom{2n-2}{n-2}\binom{n}{n-2} + ... + \binom{n+1}{1}\binom{n}{1} + 1 \\
&> \binom{n}{n} + \binom{n}{n-1} + \binom{n}{n-2} + ... + \binom{n}{1} + \binom{n}{0} \\
&= \sum_{d\geq 0} \binom{n}{d} \\
&= 2^n
\end{aligned}$$

The last step uses the equation in the hint by setting $z = 1$.

There are alternative solutions. For example, instead of the combinatorial method above, one can also use a recursive function to denote the number of possible alignments, and use it to show the conclusion.

3 Number of Optimal Alignments [10 pts]

Consider the optimal global alignment introduced in the lecture, we see that there can be multiple alignments that have the best score. How do you use a dynamic programming algorithm to calculate the total number of optimal global alignments?

Solutions:
To calculate the number of optimal global alignment, we can maintain another matrix G and fill it along with the matrix OPT (which is defined as the scoring matrix in the lecture) in the process of the dynamic programming algorithm. $G(i,j)$ is defined as the number of optimal global alignment when $x[1...i]$ and $y[1...j]$ is considered. The initial condition for G is:
$G(0,j) = 1,\ j = 0,1,...n$
$G(i,0) = 1,\ i = 1,2,...,m.$

For the recursion, we first calculate $OPT(i,j)$ and maintain all backtracing pointers (maybe more than one because of ties); then we sum over the corresponding number of optimal alignment following each pointer. That is,

$$G(i,j) = \beta_d(i,j)G(i-1,j-1) + \beta_h(i,j)G(i-1,j) + \beta_v(i,j)G(i,j-1), \tag{1}$$

where
$\beta_d(i,j) = 1$(or 0) indicates whether (or not) there exists a backtracing pointer from (i,j) to $(i-1,j-1)$;
$\beta_h(i,j) = 1$(or 0) indicates whether (or not) there exists a backtracing pointer from (i,j) to $(i-1,j)$;
$\beta_v(i,j) = 1$(or 0) indicates whether (or not) there exists a backtracing pointer from (i,j) to $(i,j-1)$;

Finally $G(m,n)$ stores the total number of optimal alignments.

4 Hidden Markov Model [20 pts]

A hidden Markov model is a graphical model of the form:

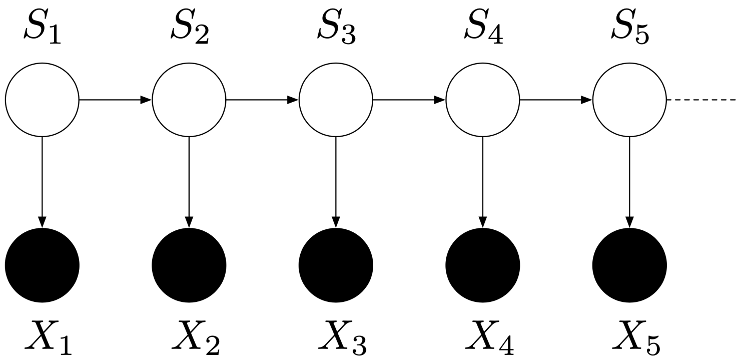


Figure 1: HMM

where $X_1, X_2, \cdots$ are the observations, and $S_1, S_2, \cdots$ are the latent states. The transition probability between hidden states can be modeled with matrix A, where element

$$A_{ij} = P(S_t = j | S_{t-1} = i) \tag{2}$$

And the emission probability can be modeled with matrix E, where element

$$E_{ik} = P(X_t = k | S_t = i) \tag{3}$$

The probability of initial hidden states can be modeled with vector π, where element

$$\pi_i = P(S_1 = i) \tag{4}$$

Now consider a DNA sequences, which can be described by a 2-states hidden Markov model with two hidden states:

- H: higher C and G content
- L: lower C and G content

The initial probabilities are

$$P(S_1 = H) = P(S_1 = L) = 0.5 \tag{5}$$

And transition probabilities are

$$\begin{cases} A_{HH} = 0.4 \\ A_{HL} = 0.6 \\ A_{LL} = 0.6 \\ A_{LH} = 0.4 \end{cases} \tag{6}$$

Nucleotide T,C,A,G are emitted from states H and L with probabilities 0.2, 0.3, 0.1, 0.4, and 0.3, 0.1, 0.4, 0.2, respectively.

a. [7 pts] Given an observed sequence x = GGCA, calculate the joint probability $P(\mathrm{x} = \mathrm{GGCA})$ using forward algorithm.

b. [6 pts] Given the same observed sequence, calculate the probability of $S_3 = L$, i.e. $P(S_3 = L|x)$. (Hint: using forward and backward algorithm)

c. [7 pts] After calculating the posterior distribution $P(S_i|x)$, we can decode the latent state S_i with maximum a posteriori (MAP) estimation. We can also decode the latent states of the whole sequence with MAP using Viterbi algorithm. Given the sequence x = GGCA, please find the hidden states (S_1, S_2, S_3, S_4) using Viterbi algorithm.

Solutions:
a.

$$\begin{cases} P(x_1 = G, S_1 = H) = P(x_1 = G|S_1 = H)P(S_1 = H) = 0.4 * 0.5 = 0.2 \\ P(x_1 = G, S_1 = L) = P(x_1 = G|S_1 = L)P(S_1 = L) = 0.2 * 0.5 = 0.1 \end{cases} \tag{7}$$

$$\begin{cases} P(x_1x_2 = GG, S_2 = H) = \sum_{S_1} P(x_1 = G, S_1)P(S_2 = H|S_1)P(x_2 = G|S_2 = H) \\ \qquad = 0.2 * 0.4 * 0.4 + 0.1 * 0.4 * 0.4 = 0.048 \\ P(x_1x_2 = GG, S_2 = L) = \sum_{S_1} P(x_1 = G, S_1)P(S_2 = L|S_1)P(x_2 = G|S_2 = L) \\ \qquad = 0.2 * 0.6 * 0.2 + 0.1 * 0.6 * 0.2 = 0.036 \end{cases} \tag{8}$$

$$\begin{cases} P(x_1x_2x_3 = GGC, S_3 = H) = \sum_{S_2} P(x_1x_2 = GG, S_2)P(S_3 = H|S_2)P(x_3 = C|S_3 = H) \\ \qquad = 0.048 * 0.4 * 0.3 + 0.036 * 0.4 * 0.3 = 0.01008 \\ P(x_1x_2x_3 = GGC, S_3 = L) = \sum_{S_2} P(x_1x_2 = GG, S_2)P(S_3 = L|S_2)P(x_3 = C|S_3 = L) \\ \qquad = 0.048 * 0.6 * 0.1 + 0.036 * 0.6 * 0.1 = 0.00504 \end{cases} \tag{9}$$

$$\begin{cases} P(x_1x_2x_3x_4 = GGCA, S_4 = H) = \sum_{S_3} P(x_1x_2x_3 = GGC, S_3)P(S_4 = H|S_3)P(x_4 = A|S_4 = H) \\ \qquad = 0.01008 * 0.4 * 0.1 + 0.00504 * 0.4 * 0.1 = 0.0006048 \\ P(x_1x_2x_3x_4 = GGCA, S_4 = L) = \sum_{S_3} P(x_1x_2x_3 = GGC, S_3)P(S_4 = L|S_3)P(x_4 = A|S_4 = L) \\ \qquad = 0.01008 * 0.6 * 0.4 + 0.00504 * 0.6 * 0.4 = 0.0036288 \end{cases} \tag{10}$$

$$P(x_1x_2x_3x_4 = GGCA) = 0.0036288 + 0.0006048 = 0.0042336 \tag{11}$$

b.

Forward algorithm:

$$P(x_1x_2x_3 = GGC, S_3 = L) = 0.00504 \tag{12}$$

Backward algorithm:

$$P(x_4 = A|S_3 = L) = \sum_{S_4} P(x_4 = A|S_4)P(S_4|S_3 = L) = 0.1 * 0.4 + 0.4 * 0.6 = 0.28 \tag{13}$$

$$P(x_1x_2x_3x_4 = GGCA, S_3 = L) = 0.00504 * 0.28 = 0.0014112 \tag{14}$$

Then

$$P(S_3 = L|x_1x_2x_3x_4 = GGCA) = 0.0014112/0.0042336 = 0.33 \tag{15}$$

c.

From forward algorithm

$$\begin{cases} V_{1H} = P(x_1 = G|S_1 = H)P(S_1 = H) = 0.4 * 0.5 = 0.2 \\ V_{1L} = P(x_1 = G|S_1 = L)P(S_1 = L) = 0.2 * 0.5 = 0.1 \end{cases} \tag{16}$$

$$\begin{cases} V_{2H} = \max\{V_{1H} * A_{HH} * P(G|H), V_{1L} * A_{LH} * P(G|H)\} = \max\{0.2 * 0.4 * 0.4, 0.1 * 0.4 * 0.4\} = 0.032 \\ V_{2L} = \max\{V_{1H} * A_{HL} * P(G|L), V_{1L} * A_{LL} * P(G|L)\} = \max\{0.2 * 0.6 * 0.2, 0.1 * 0.6 * 0.2\} = 0.024 \end{cases} \tag{17}$$

$$\begin{cases} V_{3H} = \max\{V_{2H} * A_{HH} * P(C|H), V_{2L} * A_{LH} * P(C|H)\} = \max\{0.032 * 0.4 * 0.3, 0.024 * 0.4 * 0.3\} = 0.00384 \\ V_{3L} = \max\{V_{2H} * A_{HL} * P(C|L), V_{2L} * A_{LL} * P(C|L)\} = \max\{0.032 * 0.6 * 0.1, 0.024 * 0.6 * 0.1\} = 0.00192 \end{cases} \tag{18}$$

$$\begin{cases} V_{4H} = \max\{V_{3H} * A_{HH} * P(A|H), V_{3L} * A_{LH} * P(A|H)\} = \max\{0.00384 * 0.4 * 0.1, 0.00192 * 0.4 * 0.1\} = 0.0001536 \\ V_{4L} = \max\{V_{3H} * A_{HL} * P(A|L), V_{3L} * A_{LL} * P(A|L)\} = \max\{0.00384 * 0.6 * 0.4, 0.00192 * 0.6 * 0.4\} = 0.0009216 \end{cases} \tag{19}$$

With back-tracking,

$$\begin{aligned} S_4 &= L \\ S_3 &= H \\ S_2 &= H \\ S_1 &= H \end{aligned} \tag{20}$$