### Profile HMM Problem

*Construct a profile HMM from a multiple alignment.*

**Input:** A multiple alignment *Alignment* and a threshold *θ*.

**Output:** HMM(*Alignment*, *θ*), in the form of transition and emission matrices.

**Input Format.** The first line of the input contains the threshold *θ*. The second line of the input is “--------” (a delimiter). The third line of the input is the list of symbols in the alphabet *Σ* (space-separated). The fourth line of the input is “--------” (a delimiter). The remaining lines are the multiple alignment *Alignment*, with one sequence (containing gaps) on each line.

**Output Format.** Let *States* denote the set of all states in the resulting profile HMM, where we have one Match state for each column of *Alignment* with fewer than *θ* gaps (M1, M2, …), one Insertion state for every Match state as well as before the first Match state (I0, I1, I2, …), one Deletion state for every Match state (D1, D2, …), a Start state (S), and an End state (E). The first |*States*|+1 lines should contain the transition matrix of HMM(*Alignment*, *θ*), the next line should be “--------” (a delimiter), and the remaining |*States*|+1 lines should contain the emission matrix of HMM(*Alignment*, *θ*), with both matrices drawn in the format as shown in the sample output.

**Constraints.** 0.2 ≤ *θ* ≤ 0.4; |*Σ*| = 5; 5 ≤ |*Alignment*| ≤ 10; 80 ≤ |*Alignment*[*i*]| ≤ 100

**SAMPLE DATASET:**

Input:

0.289

--------

A B C D E

--------

EBA

E-D

EB-

EED

EBD

EBE

E-D

E-D

Output:

S I0 M1 D1 I1 M2 D2 I2 E

S 0 0 1.0 0 0 0 0 0 0

I0 0 0 0 0 0 0 0 0 0

M1 0 0 0 0 0.625 0.375 0 0 0

D1 0 0 0 0 0 0 0 0 0

I1 0 0 0 0 0 0.8 0.2 0 0

M2 0 0 0 0 0 0 0 0 1.0

D2 0 0 0 0 0 0 0 0 1.0

I2 0 0 0 0 0 0 0 0 0

E 0 0 0 0 0 0 0 0 0

--------

A B C D E

S 0 0 0 0 0

I0 0 0 0 0 0

M1 0 0 0 0 1.0

D1 0 0 0 0 0

I1 0 0.8 0 0 0.2

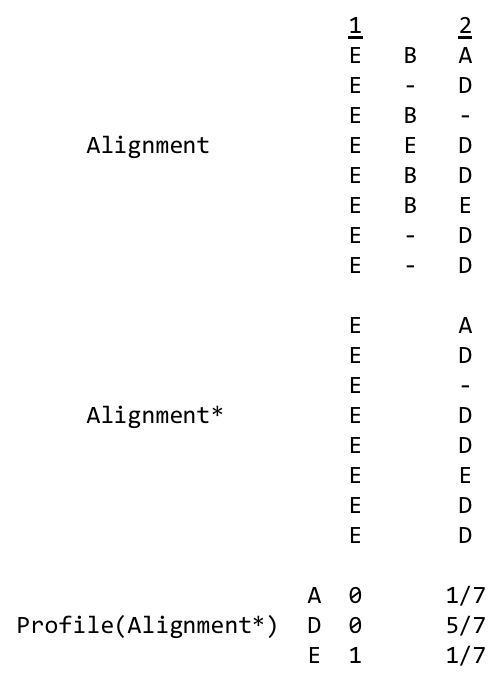
M2 0.143 0 0 0.714 0.143

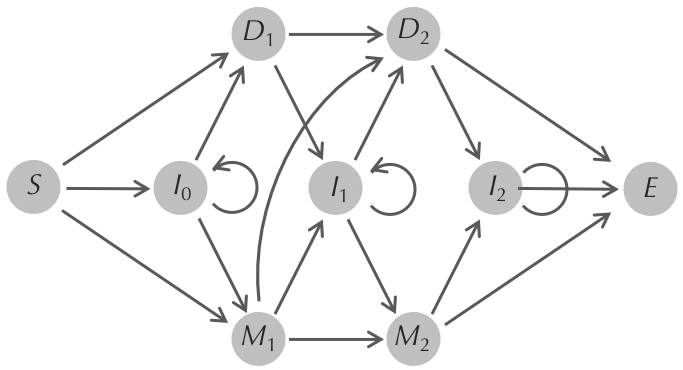
D2 0 0 0 0 0

I2 0 0 0 0 0

E 0 0 0 0 0

The above transition and emission matrices are the result of constructing a profile-HMM from the given multiple alignment *Alignment* using a threshold of *θ* = 0.289.





**TEST DATASET 1:**

Input:

0.1

--------

A B

--------

AA

AA

Output:

S I0 M1 D1 I1 M2 D2 I2 E

S 0 0 1.0 0 0 0 0 0 0

I0 0 0 0 0 0 0 0 0 0

M1 0 0 0 0 0 1.0 0 0 0

D1 0 0 0 0 0 0 0 0 0

I1 0 0 0 0 0 0 0 0 0

M2 0 0 0 0 0 0 0 0 1.0

D2 0 0 0 0 0 0 0 0 0

I2 0 0 0 0 0 0 0 0 0

E 0 0 0 0 0 0 0 0 0

--------

A B

S 0 0

I0 0 0

M1 1.0 0

D1 0 0

I1 0 0

M2 1.0 0

D2 0 0

I2 0 0

E 0 0

This dataset makes sure that your code assigns the correct states to a basic multiple alignment. Since there are no gap characters in the multiple alignment in this dataset no columns are removed and *Alignment\** is the same as *Alignment*. The profile HMM then must have 2 match states, 2 deletion states, and 3 insertion states along with start and end states. Since both of our strings are the same, we only see match states in our profile HMM and our match states will always emit an A character. Check that your output has the correct states. The I0, S, and E states are easy to forget. If your output ever goes to a deletion or insertion state make sure that you are using the correct rules in creating the profile matrix for your HMM. Deletion states can only be reached if there is a gap in a symbol in the seed alignment and insertion states can only be reached if the seed alignment is not the same as the original multiple alignment.

**TEST DATASET 2:**

Input:

0.4

--------

A B

--------

AB

A-

Output:

S I0 M1 D1 I1 E

S 0 0 1.0 0 0 0

I0 0 0 0 0 0 0

M1 0 0 0 0 0.5 0.5

D1 0 0 0 0 0 0

I1 0 0 0 0 0 1.0

E 0 0 0 0 0 0

--------

A B

S 0 0

I0 0 0

M1 1.0 0

D1 0 0

I1 0 1.0

E 0 0

This dataset makes sure that your code is correctly applying the threshold when determining the seed alignment. Since half of the second column is gap characters the second column will not be a part of the seed alignment (0.5 > threshold of 0.4). If your output has the wrong states there is likely an error in application of the threshold value. Be sure that all columns with a fraction of gap characters greater than the threshold are not in the seed alignment. If your outputted transition matrix is incorrect make sure that your code follows the rules for profile construction from the seed alignment and original multiple alignment. Similarly to Test Dataset 1, the only option from the starting state is to go into a match state and emit an A. Since the second column is not a part of the seed alignment the B character from the first string corresponds to an insertion state. The gap in the second string is ignored since it isn’t a part of the seed alignment. This means that from state M1 the HMM can either transition to I1 and emit a B character or transition to E and terminate. Check that your code correctly implements this logic if your output doesn’t match the correct output.

**TEST DATASET 3:**

Input:

0.4

--------

A B C

--------

AB

A-

Output:

S I0 M1 D1 I1 E

S 0 0 1.0 0 0 0

I0 0 0 0 0 0 0

M1 0 0 0 0 0.5 0.5

D1 0 0 0 0 0 0

I1 0 0 0 0 0 1.0

E 0 0 0 0 0 0

--------

A B C

S 0 0 0

I0 0 0 0

M1 1.0 0 0

D1 0 0 0

I1 0 1.0 0

E 0 0 0

This dataset makes sure that your code outputs emission entries for characters that were never seen in the multiple alignment. The output for this dataset is the same as Test Dataset 2, except that there is an extra column in the emission matrix corresponding to the probabilities of emitting the character C from each state. Even though all the probabilities are zero, the C column must be included in the final output.

**TEST DATASET 4:**

Input:

0.4

--------

A B

--------

A-

-A

-B

Output:

S I0 M1 D1 I1 E

S 0 0.333 0.667 0 0 0

I0 0 0 0 1.0 0 0

M1 0 0 0 0 0 1.0

D1 0 0 0 0 0 1.0

I1 0 0 0 0 0 0

E 0 0 0 0 0 0

--------

A B

S 0 0

I0 1.0 0

M1 0.5 0.5

D1 0 0

I1 0 0

E 0 0

This dataset makes sure that your code correctly handles deletion states. The first column of the multiple alignment will not be a part of the seed alignment since it contains two-thirds gap characters. This dataset has a chance to go from the starting state to an insertion state due to the A character in the first string that isn’t a part of the seed alignment. Since the gap character in the first string is part of the seed alignment it corresponds to a deletion state. If your output does not have a guaranteed transition from state I0 to state D1, be sure that your code considers the possibility of deletion states. Also be sure that your code does not assign any non-zero emission probabilities to the deletion state. Deletion states are always silent and should never emit characters.

The second and third strings have their non-seed gaps ignored and move from the start state to a match state. Since this match state is reached by two strings with different characters the emission probabilities for state M1 are evenly split between A and B. If your output transition matrix has different values in state S row be sure that your code correctly counts that there were 2 instances (second and third strings) of transitions from the starting state to state M1 and 1 instance (first string) of a transition from the starting state to state I0.

**TEST DATASET 5:**

Input:

0.5

--------

A B

--------

AA-

--A

--B

Output:

S I0 M1 D1 I1 E

S 0 0.333 0.667 0 0 0

I0 0 0.5 0 0.5 0 0

M1 0 0 0 0 0 1.0

D1 0 0 0 0 0 1.0

I1 0 0 0 0 0 0

E 0 0 0 0 0 0

--------

A B

S 0 0

I0 1.0 0

M1 0.5 0.5

D1 0 0

I1 0 0

E 0 0

This dataset makes sure that your code allows for insertion states to transition to themselves. This dataset is identical to Test Dataset 4, except that the first column is repeated twice in the multiple alignment. If your output doesn’t match the correct output it’s likely that there is a mistake in the I0 row of the transition matrix. The first time that state I0 is reached both the first and second strings will transition to I0 again because the second column is not in the seed alignment. The second time that I0 is reached both the first string will transition to a deletion state because of its gap character in a column that is part of the seed alignment. This means that state I0 can transition to itself or state D1 with equal probability. Be sure that your code correctly applies this logic if your output doesn’t match the correct output.

**TEST DATASET 6:**

Input:

0.9

--------

A B

--------

A-

-A

Output:

S I0 M1 D1 I1 M2 D2 I2 E

S 0 0 0.5 0.5 0 0 0 0 0

I0 0 0 0 0 0 0 0 0 0

M1 0 0 0 0 0 0 1.0 0 0

D1 0 0 0 0 0 1.0 0 0 0

I1 0 0 0 0 0 0 0 0 0

M2 0 0 0 0 0 0 0 0 1.0

D2 0 0 0 0 0 0 0 0 1.0

I2 0 0 0 0 0 0 0 0 0

E 0 0 0 0 0 0 0 0 0

--------

A B

S 0 0

I0 0 0

M1 1.0 0

D1 0 0

I1 0 0

M2 1.0 0

D2 0 0

I2 0 0

E 0 0

This dataset makes sure that your code includes all relevant columns in the seed alignment. Since the threshold value in this dataset is so high, all columns in the original multiple alignment should be included in the seed alignment. If your output doesn’t match the correct output check that the fraction of gap characters in each column is correctly compared to the threshold value. If your output has the correct states but an incorrect transition matrix then it is likely that your code is failing to transition from the start state to the first deletion state. Remember that gap characters in seed alignment columns correspond to a transition to a deletion state. Double check that your code implements this rule.