Name: _XIONG ZHIPENG___Student ID:_44231536___

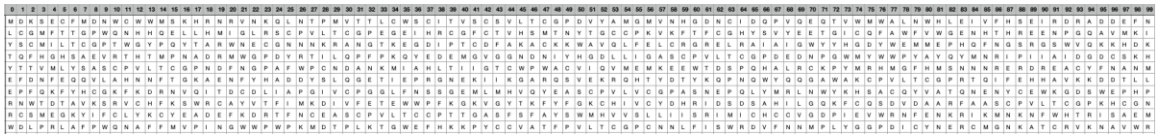
11300A Bioinformatics

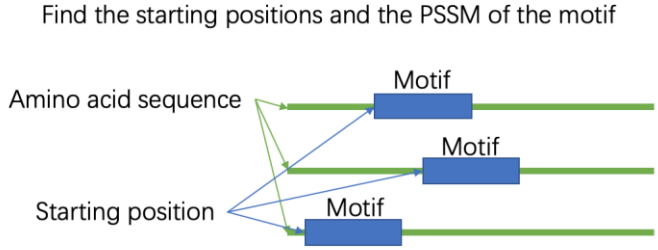
Homework 4

Dec. 19, 2023

1. **Identifying protein motifs using Expectation Maximization (EM) algorithm.**

Suppose we have 200 protein sequences of length 100, in the dataset. Each character represents an amino acid. The length of the motif is 10. The figure below shows the first 10 samples in the dataset.





Write a program to identify the starting position of the motif in each protein sequence. Write a report which includes: a) your understanding on EM algorithm, b) the core algorithm, c) the final result (starting positions and PSSM of the motif), d) the difficulties you met.

Hint:

1. In short, we need to find the common (or similar) part shared by this group of proteins.
2. How to represent a motif?
   a) We first assume the motifs have a fixed length, and each protein has one motif.
   b) Note that those similar parts are not necessarily to be exactly the same. We need a statistical model, such as the PSSM to describe them. (see PPT Multiple Sequence Alignment Page29-36 and PPT Motifs and domains Page34-36)
3. How to apply EM?
   a) Randomly initialize the starting positions for all the sequences.
   b) Given the positions, you can extract the motif from each protein.

c) Estimate the probability (or score) of each position to be the starting point given the motifs you extracted. (see PPT Motifs and domains Page36)
d) Choose the positions that has the highest probability as the updated starting position.
e) Go to b).

a) Expectation Maximization (EM) algorithm

- **Step 1: Initialization**

First, I need to initialize your algorithm with a set of starting positions for the motif in each sequence. This can be done randomly or based on some heuristic.

- **Step 2: Expectation Step**

Given the current estimate of the motif's starting positions, extract the motif instances from each sequence. A motif instance is the substring starting at the estimated position and having the specified length of the motif (in this case, 10 amino acids).

Using these instances, calculate a Position Specific Scoring Matrix (PSSM), which reflects the frequency of each amino acid at each position in the motif. This matrix represents the 'expected' motif based on the current starting positions.

- **Step 3: Maximization Step (M-step)**

With the newly constructed PSSM, I can now evaluate every possible starting position in each sequence to determine how well it matches the expected motif. Calculate a score for each position that represents the likelihood of the motif starting there, given the PSSM.

Then update the starting position for each sequence to the one with the highest score. That is, for each protein sequence, slide the window of the motif length across the sequence, calculate the score based on the PSSM, and select the position with the highest score.

- **Step 4: Iteration**

Repeat the E-step and M-step until the algorithm converges. Convergence can be determined when the changes between iterations are below a threshold, or the likelihood of the data given the model is no longer increasing significantly.

b) The Core Algorithm

```
#Expextion step
def expectation(pssm_matrix):
    expectation_matrix = []
    for i in range(sequences_coloumn_length):
        probability = 0
        for j in range(sequences_row_length - motif_length):
            for k in range(motif_length):
```

```
                probability +=
pssm_matrix[k][amino_acids.index(sequences[i][j + k])]
            probability = probability / motif_length
            expectation_matrix.append(probability)
    return expectation_matrix
```

```
#d) Implement the PSSM (Position-Specific Scoring Matrix) to describe the
statistical model.
def pssm(motifs):
    pssm_matrix = []
    overall_frequency = [0] * amino_acids_length
    for i in range(motif_length):
        column = []
        for j in range(amino_acids_length):
            count = 0
            for k in range(sequences_coloumn_length):
                if amino_acids[j] == motifs[k][i]:
                    count += 1
            overall_frequency[j] += count
            column.append(count)

        column = [x / sequences_coloumn_length for x in column]
        pssm_matrix.append(column)

    overall_frequency = [x / (sequences_coloumn_length*motif_length) for x
in overall_frequency]
    pssm_matrix = [[math.log(x) if x > 0 else -10 for x in y] for y in
pssm_matrix]
    return pssm_matrix
```

```
#c) Implement the Expectation-Maximization algorithm to update the
starting positions.
def maximization(expectation_matrix):
    updated_index = []
    for i in range(sequences_coloumn_length):
        x =
expectation_matrix.index(max(expectation_matrix[i*90:(i+1)*90])) % 90
        updated_index.append(x)
    return updated_index
```

c) the final result (starting positions and PSSM of the motif)

```
The final starting position is:  [40, 23, 0, 57, 8, 73, 49, 85, 29, 46, 82, 4, 49, 79, 71, 23, 78, 41, 0, 43, 69, 82,
76, 3, 25, 5, 36, 82, 34, 74, 30, 89, 15, 54, 42, 49, 84, 66, 71, 22, 23, 11, 51, 31, 77, 45, 30, 83, 48, 7, 80, 45, 6
0, 33, 84, 82, 86, 23, 79, 22, 21, 12, 68, 69, 81, 38, 68, 64, 89, 65, 47, 58, 53, 12, 56, 30, 12, 72, 16, 85, 57, 42,
 86, 81, 30, 8, 6, 65, 32, 82, 24, 53, 11, 26, 15, 75, 44, 41, 56, 7, 27, 47, 34, 63, 46, 9, 32, 35, 13, 27, 88, 78, 8
6, 76, 26, 26, 74, 44, 51, 7, 29, 87, 46, 64, 64, 72, 13, 5, 57, 19, 80, 10, 68, 1, 28, 64, 33, 82, 56, 88, 63, 41, 57
, 31, 30, 29, 8, 51, 26, 29, 31, 58, 69, 84, 47, 1, 55, 88, 34, 31, 65, 66, 7, 7, 89, 76, 60, 58, 37, 73, 84, 87, 18,
6, 18, 14, 83, 1, 60, 16, 68, 42, 31, 10, 47, 8, 57, 32, 52, 51, 32, 59, 64, 2, 72, 26, 66, 58, 7, 5]
```

[[-0.12216763397420753, -4.605170185988091, -4.199705077879927, -5.298317366548036, -5.298317366548036, -4.199705077879927, -5.298317366548036, -10, -10, -10, -4.605170185988091, -10, -10, -5.298317366548036, -10, -4.605170185988091, -10, -4.605170185988091, -4.199705077879927, -4.605170185988091], [-5.298317366548036, -5.298317366548036, -5.298317366548036, -5.298317366548036, -10, -5.298317366548036, -4.199705077879927, -4.605170185988091, -10, -4.605170185988091, -4.605170185988091, -5.298317366548036, -4.605170185988091, -10, -0.11093156070728166, -5.298317366548036, -10, -10, -4.605170185988091], [-4.605170185988091, -10, -4.605170185988091, -5.298317366548036, -0.11653381625595151, -5.298317366548036, -10, -10, -4.605170185988091, -4.199705077879927, -4.605170185988091, -10, -10, -10, -5.298317366548036, -4.605170185988091, -10, -5.298317366548036, -5.298317366548036, -10, -3.912023005428146], [-5.298317366548036, -4.605170185988091, -5.298317366548036, -10, -5.298317366548036, -10, -10, -10, -4.605170185988091, -10, -10, -10, -3.912023005428146, -10, -0.07257069283483537, -5.298317366548036, -10, -10, -5.298317366548036, -5.298317366548036], [-10, -5.298317366548036, -10, -5.298317366548036, -5.298317366548036, -5.298317366548036, -10, -10, -10, -4.199705077879927, -10, -4.605170185988091, -5.298317366548036, -10, -10, -5.298317366548036, -4.605170185988091, -10, -10, -0.06720874969344999], [-5.298317366548036, -5.298317366548036, -5.298317366548036, -5.298317366548036, -10, -4.199705077879927, -5.298317366548036, -5.298317366548036, -10, -10, -0.0779615414697118, -10, -10, -5.298317366548036, -5.298317366548036, -4.199705077879927, -10, -10, -5.298317366548036, -10], [-4.199705077879927, -5.298317366548036, -10, -4.605170185988091, -5.298317366548036, -5.298317366548036, -10, -10, -5.298317366548036, -5.298317366548036, -4.199705077879927, -5.298317366548036, -4.605170185988091, -5.298317366548036, -5.298317366548036, -4.605170185988091, -0.11653381625595151, -10, -10, -4.605170185988091], [-4.605170185988091, -4.605170185988091, -5.298317366548036, -4.605170185988091, -0.11093156070728166, -5.298317366548036, -10, -4.605170185988091, -4.199705077879927, -4.605170185988091, -5.298317366548036, -5.298317366548036, -10, -10, -4.199705077879927, -4.605170185988091, -4.605170185988091, -10, -0.11093156070728166, -5.298317366548036, -10, -10, -5.298317366548036, -5.298317366548036, -4.605170185988091, -5.298317366548036, -4.199705077879927, -5.298317366548036, -4.605170185988091, -10, -5.298317366548036], [-10, -5.298317366548036, -10, -4.199705077879927, -5.298317366548036, -10, -5.298317366548036, -5.298317366548036, -10, -5.298317366548036, -10, -10, -0.09431067947124129, -10, -3.912023005428146, -5.298317366548036, -5.298317366548036, -10]]

d) the difficulties you met.

(ア) Although the dataset is small, it still is difficult for me to debug my program, especially involving several multi-dimensions matrices.

(イ) I find it difficult to perform computations, using programming, involving multidimensional matrices, which often leads to errors.

(ウ) Some problems occur during my programming, such as oversight of the boundaries of the matrices, normalizing the index and so on