Algorithms in Computational Biology, 2018
Exercise 2 - Programming - Motif finding

Due date: 31/12/18

Submission should include a tar file named $ex2.tar$, containing the following files:

- A PDF report with answers to questions about the programming part.
- $motif1.txt$ and $History.tab$ as described below.
- Motif_viterbi.py
- TCM_EM.py
- Other python files are optional
- README file describing the project files and design.

TCM - Two Component Mixture

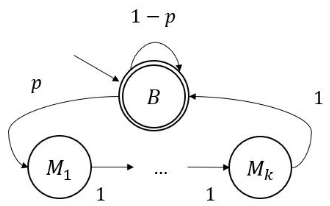
In class we discussed using EM with HMMs for finding motifs in biological sequences. We presented three different models, OOPS (One Occurance Per Sequence), ZOOPS (Zero Or One Occurance Per Sequence) and TCM (Two Component Mixture).

In this exercise you will implement the TCM variant for finding motifs where a sequence contains zero, one or multiple occurrences of a motif of size k. In each sequence motifs can begin at a different positions, if at all. There are no overlaps between motifs.

Recall a HMM with k states $S = \{s_i\}_{i=1}^{k}$ that emits data from an alphabet $\Sigma = \{\sigma_j\}_{j=1}^{m}$ can be described with parameters $\theta = (\tau, e)$ where:

- τ - transition probabilities matrix ($k \times k$), where $\tau_{i,j}$ is the probability of moving from state s_i to state s_j
- e - emission probabilities matrix ($k \times m$), where $e_{i,j}$ is $e_{s_i}(\sigma_j)$: the probability of state s_i to emit letter σ_j.

Consider the following TCM model:



States:

- B state is the background (non-motif) state. The model is forced to begin and end with this state.
- $m_1, ..., m_k$ are states denoting each letter of the motif.

Transitions:

- Let p be the probability of transition from the background state to m_1. This is the probability of entering a motif.

Emissions:

- Set uniform emission probabilities for the background state (0.25).
- Emission probabilities for the motif states will be given/learned.

1 Viterbi Decoding

Given the model parameters (i.e. transition probability p and emission probabilities for motif states) and a sequence of observations $X = (x_i)_{i=1}^n$, we want to find the most likely assinment for the hidden states that emitted x_i. That is, which positions in the sequence are part of a motif.

Write code that reads the model probabilities, initializes a new HMM model, and retrieves the most probable path for the hidden states given the emitted sequence.

Input:

- The code receives an initial emission matrix: initial_emission.tsv - defines the emission probabilities of the states for each nucleotide. Columns correspond to A, C, G, T in order, left to right. The rows describe the motif states in order $1, ..., k$.
- Transition probability is passed as command line argument p.
- The sequence, as a string (seq).

It should be possible to invoke the decoder from the command line using the following syntax:

```
python3 Motif_viterbi.py seq initial_emission.tsv p
```

The program should print the most probable path of hidden states that could emit the sequence *seq*, using Viterbi algorithm. The output format should be as follows:

```
TCGAATCCGTACGGTATTAAGTACGGCGCCTCGAATTCGAATCCGTACGGCGCCCC
BBBBBBBBBBBBBBMMMMMBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

CCGTACGGTATTAAGAAT
BBBBBBBBMMMMMBBBBB
```

Blocks of 2 lines of length up to 50 characters followed by an empty line. Top row should be the given sequence and bottom row the corresponding hidden state ('B' - Background, 'M' - Motif).

Report: Read the first sequence in *yeastGenes.fasta* (contains sequences in fasta format) and determine how many occurences of the motif described in 'initial_emission.tsv' it contains, if any, when running with $p = 0.03$. Paste your output in the pdf.

2 Probability of Observations

Using the HMM model it is possible to evaluate the likelihood that some sequence of observations came from the model.

Implement the Forward algorithm. The function should receive a sequence, transition and emission probabilities and emit the Forward table.

Report: Compute the log likelihood of the first sequence in *yeastGenes.fasta* and write it in the pdf.

3 Posterior Decoding

Implement the Backward algorithm.

Report: Compute the most likely state for each index in the first sequence in *yeastGenes.fasta*, given the sequence $\vec{x}$, $P(S_i = s|\vec{x})$. Plot the Viterbi result overlayed with the Posterior Decoding results (0 for indexes that are background and 1 for indexes considered as motifs). Write in the pdf, are there any differences? if so, why?

4 Expectation-Maximization

Write a program named TCM_EM.py that receives a list of DNA sequences and outputs probable motifs and their locations in the sequences.

Repeat for L_1 motifs The program will first recognize the L_1 most-occuring seeds (words) of length k in all sequences (L_1, k are paremeters). These are the motifs. Then it will perform the EM algorithem for each of them. You may use the python script *countWords.py* (provided) to create the dictionary. The script receives a name of a fasta file and k, and outputs for each k-mer that appears in at least one sequence the number of its occurrences.

Initialization the EM algorithm requires an initial guess of the parameters: emission and transition probabilities. For each of the seeds, an initial emission probabilities should be constructed as follows:

- Softening parameter α: Set the initial emission probabilities for the motif states to have $1 - 3 \cdot \alpha$ for the motif letter, and α for others letters (α is given as input).

  For example suppose $k = 4$ and ATTA is the motif, and $\alpha = 0.1$. Then set the initial guess for emission of the motif state to be:

$\Sigma \backslash pos$	1	2	3	4
A	0.7	0.1	0.1	0.7
C	0.1	0.1	0.1	0.1
G	0.1	0.1	0.1	0.1
T	0.1	0.7	0.7	0.1

- Uniform background state probabilities.

The initial estimate for the transition probability, p, should be the seed's abundance in the sequences. That is the total number of the seed's occurrences in all sequences (overlaps allowed), divided by the total number of times a seed of that length could have occur in all of the sequences (for each sequence, $len(seq) - len(seed) + 1$).

Training: the program should run the EM algorithm for each of the seeds. Note that the only transition probability to be learned is p and the only emission probabilities to be learned are those of the states representing the motif.

Spesifications

It should be possible to invoke the program from the command line using the following format:

```
python3 TCM_EM.py trainSeq k convergenceThr L1 alpha
```

Input

- *trainSeq*- File name of the list of sequences in which to search for the motifs. $yeastGenes.fasta$ is a file example.
- k - The length of the motifs.
- *convergenceThr* - The stopping condition for the EM is a threshold on the improvement of the log likelihood. Once the improvement of the lod likelihood is below this threshold you should stop. Meaning, when $convergenceThr > ll_t - ll_{t-1}$, where ll_t is the log-likelihood at the end of the t'th iteration.
- $L1$- Number of seeds used for initialization.
- *alpha* - Softening parameter (α) used to create the initial profile (as described above).

Output The program should output the following files:

- *History.tab*- A file containing the log likelihood of the sequences at each iteration. Each row is a different iteration, and each column is a different initiation (k-mer). There should be a header row which contains the initiation word of each column.
- *motifJ.txt* for $J = 1, ...L_1$- Files containing the profiles and the positions. There should be a different file for every initiation word according to the order of the number of occurrences of this word. The word with the largest number of occurances should be in *motif1.txt*, the second *motif2.txt* and so on. The file has two parts:
  - Emmisions: The first four lines of the file are the profile (A,C,G and T respectively), each column is a position in the motif (separated by a single space), and the content of this matrix is the probability of observing this letter at this position (written with a precision of 2 digits).
  - Positions: from line 5 and on, each line should contain two or more columns (tab separated): the sequence name and the 1-based locations of this motif within this sequence. The locations should be found using Viterbi. If the motif does not exist, the location should be 0. The sequence name is extracted from the sequence File.

  Example files 'logLikelihoodHistory.tab' and 'motif1.txt' are given.

Report

1. Choose a set of reasonable positive parameters ($k, \alpha, convergenceThr$), with $L1 = 1$.

2. Plot the mean log likelihood of the model for each iteration.
3. Submit the output files $motif1.txt$ and $History.tab$ for these parameters. Write in the pdf, does the model successfully "learn" a motif? Does it converge?

Tips:

- To force state B to be the only accepting state, you can choose from the tables (Forward, Backward, Viterbi) only paths that end with B.
- You may use logaddexp, or logsumexp, to avoid underflow.
- Optional: You might find it useful to print colored text on your terminal. Try this, for example:

```
>>> print('\033[91m' + 'Red text!' + '\033[00m')
```