EE6435 Homework 3 (Programming homework on hidden Markov model)
Out: Nov. 2, 2020
Due: 11:59PM, Nov. 16 (Monday). Start early. Debug may take time.
Full mark: 80 pts
Mark your calendar. No late work will be graded.

Handin method and requirement: name your notebook file (.ipynb) as yourlastname-firstname-studentID-hw4.ipynb. For example, if your name is Amy Zhang, the file should be named as zhang-amy-5678910-hw4.ipynb. Also, attach an html file (generated by the notebook file) with your notebook using the naming rule: yourlastname-firstname-studentID-hw4.html. (-10 points if missing these files)

You are allowed to form a group of size <=2 for this homework. In that case, you two will get the same grade for this homework. If you choose to do this by yourself, +10 points (that means, you could get 90)

New rule:

If you have difficulties doing this homework because you have not mastered basic Python programming, you can let me know and I will call for a volunteer to help you. The price you need to pay is 30 pts (to the volunteer). The volunteer has the responsibility to give you hand-to-hand training of python programming for this homework. You should implement it yourself with the volunteer's help. You cannot copy the volunteer's codes. So, if you are able to finish your codes correctly with the help, you will get 50 pts and the volunteer will get 120 pts. If you need this help, please email me with the title "EE6435: helper needed for hw4" before noon Tuesday, which will allow me sufficient time to call for volunteers.

Implementation and application of the Viterbi algorithm

The base composition of most genome sequences is not homogeneous. In particular, GC composition can vary regionally. In the human genome, for instance, there are "CpG islands" which show a very strong statistical signal (even stronger at the dinucleotide composition level than the mononucleotide composition level) and which tend to mark the 5' end of many genes. Therefore the problem of objectively segmenting a genome sequence into regions of different compositions arises naturally. One way to segment a genome is with HMM algorithms.
Let's assume that a genome can be modeled as a two-state HMM, with two states A and B. Assume that the parameters of this HMM are as follows. State A has an AT-biased emission distribution {0.35, 0.15, 0.15, 0.35} for the probabilities of {A,C,G,T}. State B has a GC-biased emission distribution {0.15, 0.35, 0.35, 0.15}. State A switches to state B with probability 0.001. State B switches back to state A with probability 0.01. Assume that the initial distribution for the HMM is uniform; 0.5, 0.5 for the two states.

The format of the hidden Markov model can be found in example.hmm and example.hmm.readme.

The file example.fa contains a simulated 1 Mb genome sequence, generated by this HMM. Lower case residues were generated by state A; upper case residues by state B. Note that *.fa means a format named "fasta". In fasta files, the first line is a header starting with ">". This is the standard format for describing sequences. The line starts with ">" contains the information of the following sequence and the Viterbi parsing is applied to the sequence starting with the next line (not the line with >).

Viterbi parsing

Implement a Viterbi parser for the HMM above using Python, including the initialization, matrix fill, and traceback stages. Your program should output the alignment as a series of segments **using the following format**.:

```
1 153 state A (or state 1)
154 252 state B (or state 2)
253 1651 state A
(... etc...)
```

Each line is a continuous segment with the same state. The format of each line is: `start-position end-position state X`

Note that the start position is 1-indexed (not 0-indexed)

Run your program on the 1 Mb simulated sequence in example.fa. **How many segments of the genome are in state B? Output this information as well.** (The true state path is in the file example.positions.)

Your program should take two arguments with this order: <input HMM file> <input fasta file>

Don't hard code any input parameters because we will change them during testing. For example, my testing file may be named as "test3.txt" or "inputdata.fa". But they always have the fasta format.

When we test your program, we will run additional input files besides the one provided on Canvas. Below please find the detailed instructions about what to submit.

1. your notebook file (.ipynb) and the html file.
2. In your source codes, clearly **comment the part about reading files, table generation, table initialization, table filling, and the traceback part**.

More details about grading.

1. Can generate correct results on 3 input files (each 15 pts, total 45 pts). The three input files have the same format as the example.fa. It is a fasta format file and contain multiple lines or just two lines.
   e.g.
   >test1
   AACCGGA
   Or
   >test2
   AACGG
   AGCT
   AAACGTA

2. The format (input, output, not hardcode any inputs) follows the instructions (15 pts)
3. The output of your program on example.fa. **How many segments of the genome are in state B? Output this information as well.** (20 pts)