Exam 3 answers

1c.

-The first line is the gapped consensus sequence includes symbols that have max symbol counts with gaps if the number of gaps at a position is greater than the max of symbol counts. For example:

A---D means A and D have the highest symbol counts at position 0 and 4. Position 1, 2 and 3 gap counts are greater than max symbol counts.

-The second line is the consensus sequence includes symbols that have max symbol counts without comparing with number of gaps at a position. So, the second line has more symbols than the first line because symbols have count less than number of gaps will present in consensus sequence. For example, ACEAD means A has the highest symbol counts at position 0 and 3, C has the highest symbols counts at position 1, etc.

-The third line is a list of information for each position in the consensus sequence. It quantifies how much a position is conserved compare with other positions. The greater the value, the more a position is conserved. The range of numbers is [log20 - ~0]. From this list, the most conserved amino acids will have the highest values.

1d.

For each position in the sequence, symbol frequency comes from the division with:

the **nominator** includes symbol counts of a symbol + gap count/len(alphabet). As the comment in the program, we add gap count because it contributes to the uncertainty. This is gap pseudo counts and should be divided by all 20 amino acids. For example, we have gap count is 20 so we have 1 more count to symbol counts because we assume probability that we meet each amino acid is equal.

The **denominator** includes total of symbol + gap count. This makes sense because it sums all possibilities we meet.

The whole line:

symbol\_freq = (symbol\_counts[symbol] +

(gap\_count / len(alphabet))) / (symbol\_sums + gap\_count)

Return the frequency of a symbol at a position.

Information for a position is calculated by the formula:

Information = Hbefore - Hafter

with maximum uncertainty at the beginning and subtract uncertainty from our observations

Hafter = uncertainty = - sum(Pi\*log2(Pi)) so

Hafter = sum(Pi\*log2(Pi)) (-- = +)

Hbefore = log2(M)= log2(20) = information(start)

information[position] += symbol\_freq \* log(symbol\_freq, 2) is used to calculate Information from the formula above.

1f. If we split the protein structure by a vertical line, one side is common. Most of residues of 2 sequences is at the same place. The second half of the sequences looks a lot more sparse than the first half because the program is only able to align a sequence with one chain. This makes us hard to visualize all the common of 2 sequences.

1g. The symmetry is more obvious when we look at the amino acids on the side of the whole structure where we can find the same amino acids on the opposite side. From the 1EI1.pdb file, I see 2 chains A and B of the same length and their order of amino acid is the same.

1h. I filter out the indices of amino acids that have information above a threshold (>4) from Findinfo.py. I got 9 amino acids. If threshold is > 3.5, I got 19 amino acids. Then I import these indices into infothreading.py and find their equivalent position in comparison summary. I create a new chain E that composes amino acids at positions I have just found.

1i. The most conserved amino acids reside on the DNA-gate, the center area of DNA gyrase structure, this area is built by all gyrase subunits. This makes sense because all DNA gyrase needs these amino acids in their structure.

Reference: [DNA gyrase](https://en.wikipedia.org/wiki/DNA_gyrase#:~:text=Gyrase%20structure,-Scheme%20of%20gyrase&text=DNA%20gyrase%20is%20a%20tetrameric,introduction%20of%202%20negative%20supercoils)

We achieved our overall goals that finding position of amino acids that are most conserved base on known protein structure.

2b. Yes, log-transformed version of my program still gives me the same value for 2 ways of calculating.

2c.

Observables for this HMM model are amino acid sequences.

2e.

>KDGL\_ECOLI

Sequence: \_ANNTTGFTRIIKAAGYSWKGLRAAWINEAAFRQEGVAVLLAVVIACWLDVDAITRVLLISSVMLVMIVEILNSAIEAVVDRIGSEYHELSGRAKDMGSAAVLIAIIVAVITWCILLWSHFG

Viterbi: SIIIIIIMOOMMOOOOOOOOOMOOOMMIIIIIIIIIIIIIIIIIIIIIIIIIIMOOMMMMOOOOMMOMMOMMOOOOOOMMOOMIIIIIIIIIIIIIIIIIIMMMOMMMOMMOOOMIIIIIII

Actual: SIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIMMMMMMMMMMMMMMMMOOOMMMMMMMMMMMMMMMMMIIIIIIIIIIIIIIIIIIIIIIIIIIIMMMMMMMMMMMMMMMMMMMMMMMOOO

Probability Viterbi -399.05032548688564

Evaluate Probability is: -399.05032548688564

At this step, our model can start to recognize membrane amino acids, but the accuracy is not high. Many places have only 1-3 membrane amino acids which are not correct. Our model misses a lot of residues which are membrane amino acids or just recognizes a part of these residues. I am not satisfied with these results.

2f. I think you want to omit entries with *E.coli* to build model and produce the output with *E.coli* to test the model.

2g. Our model does a pretty good job of recognize when we meet an amino acid belonging to membrane protein. With new states dict, we got better predictions. We can calculate the accuracy base on predict /actual with:

Predict = total number of amino acid sequences belong to membrane that our program predicts (ex. IIIII**MMM**OOOO**MMMM**IIIIII, predict = 2)

Actual = how many actual amino acid sequences belonging to membrane (ex. IIIM**MMMM**OOO**MMM**III**MMM**, actual = 3)

So, accuracy = 2/3.

2h. For the set of *E.coli* sequences, Viterbi decoded path results in higher probability. I was surprised at the difference because it is a significant difference. Viterbi returns the path that has the highest probability to get a sequence. So, probability from Viterbi should be higher than probability from other paths. We know that the sequence has protein membrane domain so the probability from Viterbi must be higher than the probability with fake state path (which is not membrane path).

2i.

Loading model file acid\_dict

>1AD2.\_ RIBOSOMAL

Sequence: \_KRYRALLEKVDPNKIYTIDEAAHLVKELATAKFDETVEVHAKLGIDPRRSDQNVRGTVSLPHGLGKQVRVLAIAKGEKIKEAEEAGADYVGGEEIIQKILDGWMDFDAVVATPDVMGAVGSKLGRILGPRGLLPNPKAGTVGFNIGEIIREIKAGRIEFRNDKTGAIHAPVGKACFPPEKLADNIRAFIRALEAHKPEGAKGTFLRSVYVTTTMGPSVRINPHS

Viterbi: SIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII

: SIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII

Actual path not given.

Probability Viterbi -655.5950583101196

Probability non-membrane: -655.5950583101196

Odds ratio: 1.0

Most of the time our model correctly predicts a sequence that is not a membrane protein. So, I think our model does a good job in this case. Odds ratio equal 1 mean probability from Viterbi equal probability from the actual path. In this case, it means a correct prediction. Odds ratio greater than 1 usually means one probability is higher than other but in this case, odds ratio greater than 1 is actually a wrong prediction.

2j. Our model is constructed simpler than the model in the paper. We only have 3 hidden states and a state dict. The quality of the state dict affects result of our model.

There are many differences between their model to our model:

-Their hidden Markov model has more states (loop cytoplasmic, cap cytoplasmic, globular, etc.) than our model.

-Their HMM training is more complicated. They have 3 states in HMM training section as they demonstrate in Materials and Methods. We just use a training set for our model which is simpler.

-Cell type is treated differently: eukaryote, gram-positive, gram-negative.

-Authors used N-best algorithm to find the most probable topology of a membrane protein, we don’t do it.

Overall, we just try a small part of their work, but the result we get is still quite good.

Q3.

I got errors, got accident.

They can’t stop me find outputs.

Because Paul is very strict.

Ps. The last sentence of the poem is a joke. Because I enjoy programming for biology questions.