Finding and Modeling a Transmembrane Region of Homologous Proteins. (100 points)

Due Date: Thu, Feb 5th at 11:45PM
Accepted up to 3 days late (with 10% point loss per day)

For this assignment you will be given a set of known transmembrane genes. You will need to convert these DNA sequences into protein sequences and find the most likely candidate portion of each protein sequence (that is the same width across all sequences) that could be the transmembrane region of the protein. You will then build a probabilistic consensus sequence for this region based upon your set of sequences and use this model to predict the transmembrane regions in an unknown protein that may have multiple transmembrane regions.

The assignment is divided up into parts with a point score attached to each part.

When working with a set of sequences of the same length it is common to build a consensus sequence, which is simply the most common base at each position in the sequence. So from the following DNA sequences:

ACGGTCGT
CCGTGCGC
ACGGACGT
GGCCGCAA

The consensus DNA sequence is

ACGGGCGT

You will be doing something similar but with amino acid sequences. Also, the model that you build will be more probabilistic (described below). In addition to building a consensus sequence it is nice to know some statistics about how frequent each amino acid is at each position. Entropy, Information, and the contribution of the amino acid to the Information are common metrics used for this purpose. Your Task is to write a python file, **TransmembraneFinder.py**, which holds the following functions:

Part 1 Reading the file (10 points)

readInput(fileName) – The readInput() function should accept a single input, a string of the name of a file. The function should read the contents of the file and output a list containing the nucleic acid sequences found in the file. The file format is one nucleic acid sequence per line.

Example:
```
>>>readInput('sequences1.txt')
['ACGGTCGT', 'CCGTGCGC', 'ACGGACGT', 'GGCCGCAA']
```

Part 2 Converting to protein (10 points)

translate(dna) – the translate() function should accept a single string input which is a DNA sequence and return a single string output which is the amino acid sequence beginning at the first start codon and ending at the first stop codon in the same reading frame.

Example:
```
>>>translate('ATGTTGATTTAATAGTTT')
'MLI'
```

Part 3 Finding Highest Hydrophobic area in protein (15 points)

mostHydrophobicRegion(aaSeq,winSize) – The function takes as input two things, a string representation of an amino acid sequence and an integer window size. The function should return the start location of a window of the given size which is the most hydrophobic region of that size. To calculate the hydrophobic value of a window just sum the hydrophobicity of the amino acids in that window. There are several scales for measuring hydrophobicity. Take a look at the wikipedia page for scales on measuring hydrophobicity at http://en.wikipedia.org/wiki/Hydrophobicity_scales. Each team may use a different method for measuring the hydrophobicity of each amino acid.

Example:
>>>mostHydrophobicRegion('MNQSTKRKHVLIPLPPVSTKRHD',8)
9

Part 4 Finding the best window size (20 points)

bestWindowSize(listOfAASeqs,lowWinSize,highWinSize) – the function should take in a list of amino acid sequences and two numbers representing low and high values for possible window sizes. The function should return the window size which has the highest total hydrophibcity across all of the amino acid sequences, where the total hydrophobicity is the sum of the most hydrophobic region of each amino acid sequence.

Example:
>>>bestWindowSize(['MNQSTKRKHVLIPLPPVSTKRHD','MNQSTKRKHVLIPLPPVSTKRHD','MNQSTKRKHVLIPLPPVSTKRHD']
,6,10)
8

Part 5 Calculating some statistics (25 points)

gatherContributions(listOfAASeqs) -- The function should accept a list of amino acid sub-sequences (the best window size and most hydrophobic region of each amino acid) and produce a list of dictionaries where every entry in the dictionary is the base and the contribution of that base to the information for that position. See the course slides for a description of Entropy, Information, and the contribution of the base towards the Information. These terms are also discussed in the bioinformatics text on page 76 and 77.

Example (this example is with nucleic acid sequences, but your code should work with amino acid sequences):
>>>gatherContributions(['ACGGTCGT', 'CCGTGCGC', 'ACGGACGT', 'GGCCGCAA'])
[{'A':XXXXX, 'C':XXXXX, 'G':XXXXX, 'T':XXXXX}, {'A':XXXXX, 'C':XXXXX, 'G':XXXXXX, 'T':XXXXXX}, {'A':XXXXX, 'C':XXXXX, 'G':XXXXX, 'T':XXXXX}, {'A':XXXXX, 'C':XXXXX, 'G':XXXXXX, 'T':XXXXXX}, {'A':XXXXX, 'C':XXXXX, 'G':XXXXX, 'T':XXXXX}, {'A':XXXXX, 'C':XXXXX, 'G':XXXXXX, 'T':XXXXXX}, {'A':XXXXX, 'C':XXXXX, 'G':XXXXX, 'T':XXXXX}, {'A':XXXXX, 'C':XXXXX, 'G':XXXXXX, 'T':XXXXXX}]

where the XXXXX are replaced by the number corresponding to there contribution to the information.

Part 6 Finding the best hydrophobic regions of an unknown amino acid sequence (20 points)

findHydrophobicRegions(listOfDicts,aaSeq) – This function should take in a list of dictionaries (the model) containing the contributions of the base to the information (output from the gatherContributions() function) and an amino acid sequence. The function should return a list of strings where every string is a region of the amino acid that had a high match to the model of contributions. You can determine what a high match would mean by seeing how well your model matches the sequences given in your training data set.

Extra Credit (Graphing it) (30 points)

constructGraph(listOfDicts,aaSeq) -- This function should take in a list of dictionaries (the model) containing the contributions of the base to the information (output from the gatherContributions() function) and an amino acid sequence. The function should create a graph where the X axis is the position in the amino acid sequence and the Y axis is how well the model matches at that location, graphically representing the same information as part 6. Matplotlib is one of the default libararies that is installed with Spyder. It is up to you to learn this library. There is some excellent on-line documentation at http://matplotlib.org/users/pyplot_tutorial.html#pyplot-tutorial with a tutorial on creating pyplots.

Handin

The TransmembraneFinder.py file should contain all the above functions. Any additional functions that your code uses that you create should be imported from other files (see below about importing). Zip up the entire directory containing all of the files needed to use the defined functions found in TransmembraneFinder.py. Attach the .zip file to this assignment in Moodle.

ADDITONAL POINTS

Point 1

You will probably want to write several helper functions which carry out some portion of the task needed for the above functions. You can place these helper functions in a separate file (e.g. HelperFunctions.py) and then import those functions into the ConsensusSequence.py file. These functions are then available for use inside any functions of ConsensusSequence.py.

I recommend first sitting down with your partner and designing the full set of functions that you will need for this assignment. Define exactly what the input and output from these functions will be. For example you may want to design some of the following functions:

gatherCounts() -- Accepts a list of sequences and returns a list of the frequence counts of each amino acids at each position
>>>gatherCounts(['ACG','CGG','CGT','CGA'] (this is with DNA, but yours should work with amino acids)
[[1,3,0,0],[0,1,3,0],[1,0,2,1]]

entropy() -- Accepts a list of probabilities and returns a number representing the entropy
>>>entropy([0.33, 0.14, 0.41, 0.12])
1.81939

information() -- Accepts two numbers representing the entropy and the number of bases and returns the information
>>>information(1.81939,4)
1.80615

calcProbs() -- Accepts a list of frequence counts and outputs a list of probabilities
>>>calcProbs([1,3,0,0])
[0.25, 0.75, 0.0, 0.0]

etc.

Point 2

You can import from one file to another. If you have a file called HelperFunctions.py which defines the entropy() function then you can use it in another file, like ConsensusSequence.py by placing the following line at the top of your ConsensusSequence.py file:

from HelperFunctions import *

Then you have access to the entropy() function in your ConsensusSequence.py file.

Point 3

You will want to become familiar with all of the methods that are available to you for handling strings. For example strip(), join(), and split() make it easy to write the readInput() function. It becomes just a few lines of code. Read over the documentation on Strings in python at https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str.