Homework 7

CSIS 758

Spring 2017

1 Genbank Files

The file *bacteria.tar* contains several files that end with the extension *.gb.txt*. These are Genbank files.

The file *genbank.py* can read these files. The following code shows how to read one file. Line 2 is the name of the Genbank file. Line 3 reads the entire file. Line 4 extracts the DNA string. Lines 7 and 8 are used to get the positions of the START codons. Two examples are shown. Lines 9 and 10 show the first gene. The `False` indicates that this gene is not a complement. This is the data that we want. The last two lines show a gene that is a complement and we do not want this.

```
1  >>> import genbank as gb
2  >>> fname = '/science/Genbank/AE000512.gb.txt'
3  >>> data = gb.ReadFile( fname )
4  >>> dna = gb.ParseDNA( data )
5  >>> len( dna )
6  1860725
7  >>> klocs = gb.FindKeywordLocs( data )
8  >>> glocs = gb.GeneLocs( data, klocs )
9  >>> glocs[0]
10 ([(323, 448)], False)
11 >>> glocs[10]
12 ([(10472, 10858)], True)
```

Line 10 also shows some numbers. The first number is 323 which is the location of the START codon. Almost. The Genbank files start at index 1 and Python arrays start at index 0. In other words, the 323 value is too high. The following code shows that the data that starts at 322 begins with an ATG.

```
1  >>> dna[323:343]
2  'tggtttatggaaaggaagga'
3  >>> dna[322:343]
4  'atggtttatggaaaggaagga'
```

2 Gathering Data

You will want to lists of strings (starts and nonstarts). Each string will include the 30 characters before an ATG and 20 characters after an ATG. So the string is 53 characters long.

In steps you will get **all** strings that and ATG at position 30. Use the data from `glocs` to determine if this particular string is a START or a NONSTART. Put the string in the appropriate list. Any string that does not have 30 characters in front of the ATG, 20 characters after the ATG, or has letters other than A, T, G¡ or C should be discarded.

The nonstart list should be significantly larger than the start list.

Separate the start list into two groups: training and nontraining. You should have three lists:

- Training STARTS,
- Nontraing STARTS, and
- NONSTARTS.

3 Log Odds Matrices

It is not necessary to use *hmm3.py* for this problem. The training data is a set of strings, $\mathcal{S}$. The $\mathcal{S}_k$ is the k-th string, and $\mathcal{S}_{k,i}$ is the i-th element in the k-th string.

The first matrix is $\mathbf{M}_0$ and it is a 4×4 matrix because there are four letters A, C, G, and T. Associate each letter with an index: A=0, C=1, G=2, and T=3. Thus, $\mathbf{M}_{0;m,n}$ is the (m, n) element of the matrix. It is also the number of times that a transition was seen. Thus, $\mathbf{M}_{0;2,1}$ is the number of times that we saw a G in the first position and a C in the second position of the strings.

There is a matrix for each position in the string (except the last). So there are 52 matrices. The matrix $\mathbf{M}_k$ counts the transitions from the k-th position to the $k+1$ position in the strings.

Convert the each matrix to a probability matrix. Sum each row, and divide the values in a row by that sum. So,

$$\begin{pmatrix} 1 & 2 & 3 & 1 \\ 1 & 4 & 2 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1/7 & 2/7 & 3/7 & 1/7 \\ 1/8 & 4/8 & 2/8 & 1/8 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Convert the matrix to odds. Recall that in the case of random data all of the values in the matrix would be 0.25. To compute the odds the probabilities are divided by the random case. Simple, divide the probability matrix by 0.25.

Convert the matrix to log-odds. Compute the logarithm of the matrix.

Repeat this process for all matrices.

4 Scoring and Distributions

A score can be computed for each string. The value from the first matrix relates to the transition from the first to the second letter of the string. The value from the second matrix relates to the transition from the second letter to the third letter. Since these values are log-odds they are added. The process continues to the end of the string.

If the system worked perfectly then all of the START strings would return a positive score and all of the NONSTART strings would return a negative scores. Of course, this is not a perfect world.

There are three sets of strings. For each set, compute the scores for all of the strings in that set. Compute the average and standard deviation of those scores. Repeat for the other two sets of strings.

Create a graph which has three Gaussian curves which are defined by the average and standard deviation that have just been computed. Label each plot appropriately (training, non-training, nonstart).

5 Determination of Importance

This section will require creativity. There are 52 matrices representing 52 locations of transition in the data. It is expected that some of the transitions are more important than others. This has to do with transcription of the DNA.

You will create your own method to determine which locations are important through analysis of the matrices. You **will** describe what your method is and then employ it.

Determine which locations in the 53 length strings are important in determining the difference between a START and NONSTART region.

6 New and Improved HMM

Now that you have determined that some locations are more important than others, the next step is to build an HMM using only those locations.

This is real easy. Simply change the matrices of the non-important locations to have zeros in all of the elements. In this manner you can use the code that is already written.

7 New and Improved Distributions

Repeat the previous section on creating the distribution graphs using your new HMM.

8 Analysis

You have two sets of graphs from two experiments. The goal is to separate the STARTS from the NONSTARTS. It is expected that one of the HMM's worked better than the other.

Just by looking at the graphs you should be able to determine which one performed better (or they performed the same). Write a paragraph on which HMM was best and why.