Report for CpG Island Detector Using HMM and Viterbi Algorithm

Team Member and Job Description

Jinghan Lu: implemented the Viterbi Algorithm Tianyuan Fu: implemented algorithms for probability, finished up the implementation of Viterbi Algorithm

Functionality of CpG Island Detector

This program is supposed to identify the potential CpG Islands in a given input DNA sequence. CpG island is defined as a subsequence of a long DNA sequence, which is rich in C and G. According to biological probabilistic researches, DNA regions with high reichness in C and G are more likely to carry gene information. Therefore, this approach to detect CpG Islands can be useful in real-time biological researches.

The input format should be a file describing the composition of a DNA sequence, comprised with A, G, C, T, 4 kinds of nucleotide acids. The output will be starting and ending position pairs displayed to the console, along with a file containing the whole CpG Island subsequence and more details.

Technical Details

For our CpG island detector, we used Hidden Markov Models technique, which allows us to think about causal factors in a probabilistic model with both the observed DNA sequence, which consists of four base pairs of A, G, C, T (known as nucleic acids), and the hidden CpG islands. A HMM consists of a set of N hidden states, which in our case are eight base-sign pairs, where the bases are A, G, C, T and the signs are “+” (belongs to a CpG island) and “-” (does not belong to a CpG island), a transition probability matrix which gives the probability of moving from one state to the another, an observed sequence which is the DNA sequence, a sequence of emission probabilities which represents the probability of observe a base given a hidden state (e.g., P(A|A+) = 1, P(C|G-) = 0…), and an initial probability distribution over the eight hidden states(A+, A-, C+, C-, G+, G-, T+, T-).

In order to determine the most like state sequence given an observed DNA sequence, we implemented a Viterbi algorithm. Viterbi is a kind of dynamic programming, and the idea is to process a observed DNA sequence from left to right by calculating the probability that the HMM is in a certain state after processing all the previous observed AGCTs and output the most possible state sequence for CpG island detections. In our Viterbi, we first initialized a sequence probability matrix and a back pointer matrix to keep track of the states. For each hidden state, we recursively calculated their Viterbi path probabilities by multiplying the previous Viterbi path probability, the transition probability from the previous state to the current one, and the emission probability. Then we chose the maximum Viterbi value to be the most possible state for the that observed nucleic acid and stored the index of the previous state that led to the state with this maximum Viterbi value. At the end of the sequence after the recursion, we picked the state with the highest probability and back trace to get the best state sequence.

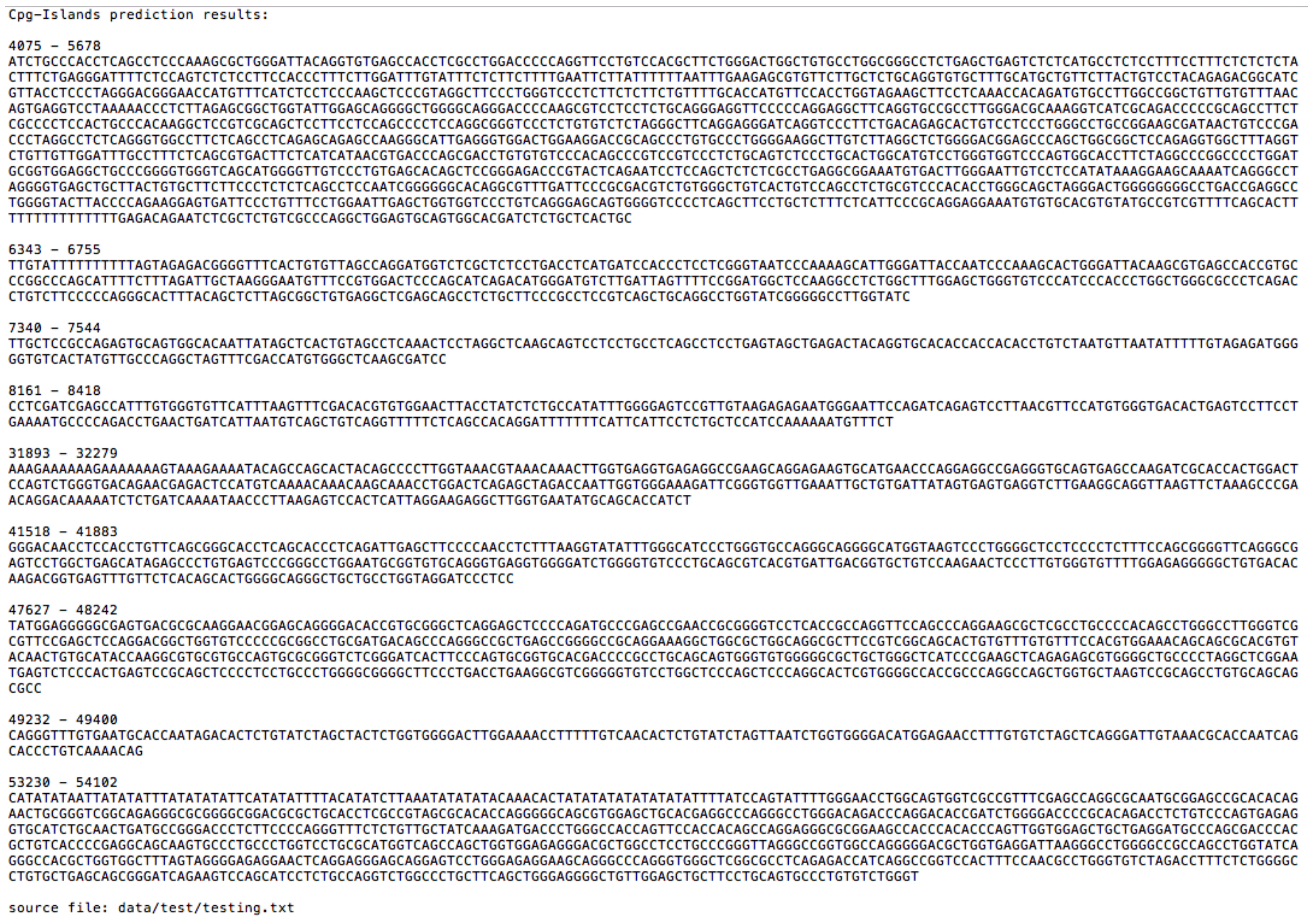
In the Viterbi algorithm where we need to calculate probabilities, we used log of the probabilities. The reason is that if we compute probabilities normally, the Viterbi values dropped to zero after a certain point because of the small values of the transition probabilities, which made the comparison hard and led to a wrong result. With the log function, we would be able to compare the values and choose the right state. Additionally, when multiplying emission probabilities in Viterbi, we changed every zero emission probability and transition probability to a small number 10^-30 since log(0) is not defined.

Interesting Sample Session

output: output of the probabilities and the postions of CpG island detector

```
Welcome to our CpG islands Detector!
Please choose one of the testing files: ['.DS_Store', 'testing-chr1.txt', 'testing.txt']
Your input dataset: testing.txt


Disjoint counts: {'A+': 1226, 'G+': 2521, 'C+': 2809, 'T+': 1324, 'A-': 16480, 'G-': 20857, 'C-': 21853, 'T-': 17930}


Transition counts: {'A+': {'A+': 203, 'G+': 489, 'C+': 390, 'T+': 144, 'A-': 0, 'G-': 0, 'C-': 0, 'T-': 0}, 'G+': {'A+': 393, 'G+
': 844, 'C+': 865, 'T+': 399, 'A-': 2, 'G-': 9, 'C-': 7, 'T-': 2}, 'C+': {'A+': 544, 'G+': 672, 'C+': 1040, 'T+': 553, 'A-': 0, '
G-': 0, 'C-': 0, 'T-': 0}, 'T+': {'A+': 86, 'G+': 516, 'C+': 494, 'T+': 228, 'A-': 0, 'G-': 0, 'C-': 0, 'T-': 0}, 'A-': {'A+': 0,
 'G+': 0, 'C+': 7, 'T+': 0, 'A-': 4110, 'G-': 5644, 'C-': 3717, 'T-': 3002}, 'G-': {'A+': 0, 'G+': 0, 'C+': 6, 'T+': 0, 'A-': 432
5, 'G-': 7074, 'C-': 5545, 'T-': 3907}, 'C-': {'A+': 0, 'G+': 0, 'C+': 5, 'T+': 0, 'A-': 5809, 'G-': 2223, 'C-': 7556, 'T-': 6260
}, 'T-': {'A+': 0, 'G+': 0, 'C+': 2, 'T+': 0, 'A-': 2234, 'G-': 5906, 'C-': 5028, 'T-': 4759}}



Disjoint Probability: {'A+': 0.014423529411764707, 'G+': 0.029658823529411765, 'C+': 0.03304705882352941, 'T+': 0.015576470588235
294, 'A-': 0.19388235294117648, 'G-': 0.2453764705882353, 'C-': 0.2570941176470588, 'T-': 0.21094117647058824}



Transition Probability: {'A+': {'A+': 0.16557911908646003, 'G+': 0.398858075040783, 'C+': 0.3181076672104405, 'T+': 0.11745513866
231648, 'A-': 1e-30, 'G-': 1e-30, 'C-': 1e-30, 'T-': 1e-30}, 'G+': {'A+': 0.15589051963506545, 'G+': 0.3347877826259421, 'C+': 0.
3431178103927013, 'T+': 0.15827052756842522, 'A-': 0.0007933359777865926, 'G-': 0.0035700119000396666, 'C-': 0.002776675922253074
, 'T-': 0.0007933359777865926}, 'C+': {'A+': 0.19366322534709862, 'G+': 0.2392310430758277, 'C+': 0.3702385190459238, 'T+': 0.196
8672125311499, 'A-': 1e-30, 'G-': 1e-30, 'C-': 1e-30, 'T-': 1e-30}, 'T+': {'A+': 0.0649546827794562, 'G+': 0.38972809667673713, '
C+': 0.3731117824773414, 'T+': 0.17220543806646527, 'A-': 1e-30, 'G-': 1e-30, 'C-': 1e-30, 'T-': 1e-30}, 'A-': {'A+': 1e-30, 'G+'
: 1e-30, 'C+': 0.00042475728155339805, 'T+': 1e-30, 'A-': 0.24939320388349515, 'G-': 0.34247572815533983, 'C-': 0.225546116504854
37, 'T-': 0.1821601941747573}, 'G-': {'A+': 1e-30, 'G+': 1e-30, 'C+': 0.0002876732032411181, 'T+': 1e-30, 'A-': 0.207364434002972
64, 'G-': 0.33916670662127824, 'C-': 0.2658579853286666, 'T-': 0.1873232008438414}, 'C-': {'A+': 1e-30, 'G+': 1e-30, 'C+': 0.0002
288015375463323, 'T+': 1e-30, 'A-': 0.2658216263213289, 'G-': 0.10172516359309934, 'C-': 0.3457648835400174, 'T-': 0.286459525008
00803}, 'T-': {'A+': 1e-30, 'G+': 1e-30, 'C+': 0.00011155111829996096, 'T+': 1e-30, 'A-': 0.12460259914105638, 'G-': 0.3294104523
397847, 'C-': 0.28043951140610185, 'T-': 0.2654358859947571}}



Running Viterbi Algorithm...


[RESULTS] There are 9 CpG-islands in total.
Position pairs on the DNA sequence are:
4075 5678
6343 6755
7340 7544
8161 8418
31893 32279
41518 41883
47627 48242
49232 49400
53230 54102

End of the algorithm. Please check  "result/CpG-Island Results: 2019-12-02 22:18:29" for more information.
```

output file: output file containing the CpG islands' sequences, their positions, and the source of testing file



```
Cpg-Islands prediction results:

4075 - 5678
ATCTGCCCACCTCAGCCTCCCAAAGCGCTGGGATTACAGGTGTGAGCCACCTCGCCTGGACCCCCAGGTTCCTGTCCACGCTTCTGGGACTGGCTGTGCCTGGCGGGCCTCTGAGCTGAGTCTCTCATGCCTCTCCTTTCCTTTCTCTCTCTA
CTTTCTGAGGGATTTTCTCCAGTCTCTCCTTCCACCCTTTCTTGGATTTGTATTTCTCTTCTTTTGAATTCTTATTTTTTAATTTGAAGAGCGTGTTCTTGCTCTGCAGGTGTGCTTTGCATGCTGTTCTTACTGTCCTACAGAGACGGCATC
GTTACCTCCCTAGGGACGGAACCATGTTTCATCTCCTCCCAAGCTCCCGTAGGCTTCCCTGGGTCCCTCTTCTCTTCTGTTTTGCACCATGTTCCACCTGGTAGAAGCTTCCTCAAACCACAGATGTGCCTTGGCCGGCTGTTGTGTTTAAC
AGTGAGGTCCTAAAAACCCTCTTAGAGCGGCTGGTATTGGAGCAGGGGCTGGGGCAGGGACCCCAAGCGTCCTCCTCTGCAGGGAGGTTCCCCCAGGAGGCTTCAGGTGCCGCCTTGGGACGCAAAGGTCATCGCAGACCCCCGCAGCCTTCT
CGCCCCTCCACTGCCCACAAGGCTCCGTCGCAGCTCCTTCCTCCAGCCCCTCCAGGCGGGTCCCTCTGTGTCTCTAGGGCTTCAGGAGGGATCAGGTCCCTTCTGACAGAGCACTGTCCTCCCTGGGCCTGCCGGAAGCGATAACTGTCCCGA
CCCTAGGCCTCTCAGGGTGGCCTTCTCAGCCTCAGAGCAGAGCCAAGGGCATTGAGGGTGGACTGGAAGGACCGCAGCCCTGTGCCCTGGGGAAGGCTTGTCTTAGGCTCTGGGGACGGAGCCCAGCTGGCGGCTCCAGAGGTGGCTTTAGGT
CTGTTGTTGGATTTGCCTTTCTCAGCGTGACTTCTCATCATAACGTGACCCAGCGACCTGTGTGTCCCACAGCCCGTCCGTCCCTCTGCAGTCTCCCTGCACTGGCATGTCCTGGGTGGTCCCAGTGGCACCTTCTAGGCCCGGCCCCTGGAT
GCGGTGGAGGCTGCCCGGGGTGGGTCAGCATGGGGTTGTCCCTGTGAGCACAGCTCCGGGAGACCCGTACTCAGAATCCTCCAGCTCTCTCGCCTGAGGCGGAAATGTGACTTGGGAATTGTCCTCCATATAAAGGAAGCAAAATCAGGGCCT
AGGGGTGAGCTGCTTACTGTGCTTCTTCCCTCTCTCAGCCTCCAATCGGGGGGCACAGGCGTTTGATTCCCGCGACGTCTGTGGGCTGTCACTGTCCAGCCTCTGCGTCCCACACCTGGGCAGCTAGGGACTGGGGGGGGCCTGACCGAGGCC
TGGGGTACTTACCCCAGAAGGAGTGATTCCCTGTTTCCTGGAATTGAGCTGGTGGTCCCTGTCAGGGAGCAGTGGGGTCCCCTCAGCTTCCTGCTCTTTCTCATTCCCGCAGGAGGAAATGTGTGCACGTGTATGCCGTCGTTTTCAGCACTT
TTTTTTTTTTTTTGAGACAGAATCTCGCTCTGTCGCCCAGGCTGGAGTGCAGTGGCACGATCTCTGCTCACTGC

6343 - 6755
TTGTATTTTTTTTTTAGTAGAGACGGGGTTTCACTGTGTTAGCCAGGATGGTCTCGCTCTCCTGACCTCATGATCCACCCTCCTCGGGTAATCCCAAAAGCATTGGGATTACCAATCCCAAAGCACTGGGATTACAAGCGTGAGCCACCGTGC
CCGGCCCAGCATTTTCTTTAGATTGCTAAGGGAATGTTTCCGTGGACTCCCAGCATCAGACATGGGATGTCTTGATTAGTTTTCCGGATGGCTCCAAGGCCTCTGGCTTTGGAGCTGGGTGTCCCATCCCACCCTGGCTGGGCGCCCTCAGAC
CTGTCTTCCCCCAGGGCACTTTACAGCTCTTAGCGGCTGTGAGGCTCGAGCAGCCTCTGCTTCCCGCCTCCGTCAGCTGCAGGCCTGGTATCGGGGGCCTTGGTATC

7340 - 7544
TTGCTCCGCCAGAGTGCAGTGGCACAATTATAGCTCACTGTAGCCTCAAACTCCTAGGCTCAAGCAGTCCTCCTGCCTCAGCCTCCTGAGTAGCTGAGACTACAGGTGCACACCACCACACCTGTCTAATGTTAATATTTTTGTAGAGATGGG
GGTGTCACTATGTTGCCCAGGCTAGTTTCGACCATGTGGGCTCAAGCGATCC

8161 - 8418
CCTCGATCGAGCCATTTGTGGGTGTTCATTTAAGTTTCGACACGTGTGGAACTTACCTATCTCTGCCATATTTGGGGAGTCCGTTGTAAGAGAGAATGGGAATTCCAGATCAGAGTCCTTAACGTTCCATGTGGGTGACACTGAGTCCTTCCT
GAAAATGCCCCAGACCTGAACTGATCATTAATGTCAGCTGTCAGGTTTTTCTCAGCCACAGGATTTTTTCATTCATTCCTCTGCTCCATCCAAAAAATGTTTCT

31893 - 32279
AAAGAAAAAAGAAAAAAAGTAAAGAAAATACAGCCAGCACTACAGCCCCTTGGTAAACGTAAACAAACTTGGTGAGGTGAGAGGCCGAAGCAGGAGAAGTGCATGAACCCAGGAGGCCGAGGGTGCAGTGAGCCAAGATCGCACCACTGGACT
CCAGTCTGGGTGACAGAACGAGACTCCATGTCAAAACAAACAAGCAAACCTGGACTCAGAGCTAGACCAATTGGTGGGAAAGATTCGGGTGGTTGAAATTGCTGTGATTATAGTGAGTGAGGTCTTGAAGGCAGGTTAAGTTCTAAAGCCCGA
ACAGGACAAAAATCTCTGATCAAAATAACCCTTAAGAGTCCACTCATTAGGAAGAGGCTTGGTGAATATGCAGCACCATCT

41518 - 41883
GGGACAACCTCCACCTGTTCAGCGGGCACCTCAGCACCCTCAGATTGAGCTTCCCCAACCTCTTTAAGGTATATTTGGGCATCCCTGGGTGCCAGGGCAGGGGCATGGTAAGTCCCTGGGGCTCCTCCCCTCTTTCCAGCGGGGTTCAGGGCG
AGTCCTGGCTGAGCATAGAGCCCTGTGAGTCCCGGGCCTGGAATGCGGTGTGCAGGGTGAGGTGGGGATCTGGGGTGTCCCTGCAGCGTCACGTGATTGACGGTGCTGTCCAAGAACTCCCTTGTGGGTGTTTTGGAGAGGGGGCTGTGACAC
AAGACGGTGAGTTTGTTCTCACAGCACTGGGGCAGGGCTGCTGCCTGGTAGGATCCCTCC

47627 - 48242
TATGGAGGGGGCGAGTGACGCGCAAGGAACGGAGCAGGGGACACCGTGCGGGCTCAGGAGCTCCCCAGATGCCCGAGCCGAACCGCGGGGTCCTCACCGCCAGGTTCCAGCCCAGGAAGCGCTCGCCTGCCCCACAGCCTGGGCCTTGGGTCG
CGTTCCGAGCTCCAGGACGGCTGGTGTCCCCCGCGGCCTGCGATGACAGCCCAGGGCCGCTGAGCCGGGGCCGCAGGAAAGGCTGGCGCTGGCAGGCGCTTCCGTCGGCAGCACTGTGTTTGTGTTTCCACGTGGAAACAGCAGCGCACGTGT
ACAACTGTGCATACCAAGGCGTGCGTGCCAGTGCGCGGGTCTCGGGATCACTTCCCAGTGCGGTGCACGACCCCGCCTGCAGCAGTGGGTGTGGGGGCGCTGCTGGGCTCATCCCGAAGCTCAGAGAGCGTGGGGCTGCCCCTAGGCTCGGAA
TGAGTCTCCCACTGAGTCCGCAGCTCCCCTCCTGCCCTGGGGCGGGGCTTCCCTGACCTGAAGGCGTCGGGGGTGTCCTGGCTCCCAGCTCCCAGGCACTCGTGGGGCCACCGCCCAGGCCAGCTGGTGCTAAGTCCGCAGCCTGTGCAGCAG
CGCC

49232 - 49400
CAGGGTTTGTGAATGCACCAATAGACACTCTGTATCTAGCTACTCTGGTGGGGACTTGGAAAACCTTTTTGTCAACACTCTGTATCTAGTTAATCTGGTGGGGACATGGAGAACCTTTGTGTCTAGCTCAGGGATTGTAAACGCACCAATCAG
CACCCTGTCAAAACAG

53230 - 54102
CATATATAATTATATATTTATATATATTCATATATTTTACATATCTTAAATATATATACAAACACTATATATATATATATATTTTATCCAGTATTTTGGGAACCTGGCAGTGGTCGCCGTTTCGAGCCAGGCGCAATGCGGAGCCGCACACAG
AACTGCGGGTCGGCAGAGGGCGCGGGGCGGACGCGCTGCACCTCGCCGTAGCGCACACCAGGGGGCAGCGTGGAGCTGCACGAGGCCCAGGGCCTGGGACAGACCCAGGACACCGATCTGGGGACCCCGCACAGACCTCTGTCCCAGTGAGAG
GTGCATCTGCAACTGATGCCGGGACCCTCTTCCCCAGGGTTTCTCTGTTGCTATCAAAGATGACCCTGGGCCACCAGTTCCACCACAGCCAGGAGGGCGCGGAAGCCACCCACACCCAGTTGGTGGAGCTGCTGAGGATGCCCAGCGACCCAC
GCTGTCACCCCGAGGCAGCAAGTGCCCTGCCCTGGTCCTGCGCATGGTCAGCCAGCTGGTGGAGAGGGACGCTGGCCTCCTGCCCGGGTTAGGGCCGGTGGCCAGGGGGACGCTGGTGAGGATTAAGGGCCTGGGGCCGCCAGCCTGGTATCA
GGGCCACGCTGGTGGCTTTAGTAGGGGAGAGGAACTCAGGAGGGAGCAGGAGTCCTGGGAGAGGAAGCAGGGCCCAGGGTGGGCTCGGCGCCTCAGAGACCATCAGGCCGGTCCACTTTCCAACGCCTGGGTGTCTAGACCTTTCTCTGGGGC
CTGTGCTGAGCAGCGGGATCAGAAGTCCAGCATCCTCTGCCAGGTCTGGCCCTGCTTCAGCTGGGAGGGGCTGTTGGAGCTGCTTCCTGCAGTGCCCTGTGTCTGGGT

source file: data/test/testing.txt
```

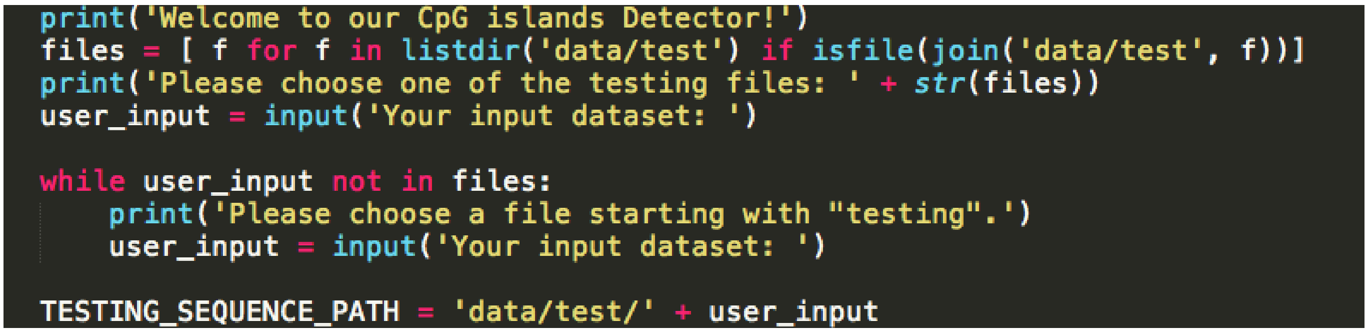
Demo Instructions

1. Unzip the zip file and put the decompressed folder in a directory that you are familiar with
2. Open up a shell window (Linux) / terminal window (macOS) and change the directory into the upzipped folder
3. Check the content of the folder:
    1. There is a folder called `result/`. This will be where the output files locate
    2. There is a `data/` folder. Inside `data/`, there should be 3 text files and a `test/` folder containing two text files. These are the training and testing data for this program.
4. Back to the console. Try with the command: `python3 HMM.py`
5. Seeing the prompted message, choose a testing file from the message and type in the console. Hit `enter` to continue.
6. Intermediate results, such as disjoint probability, transition probability would be prinred in the console.
7. The final results would be in two parts: a message in the console showing the starting and ending position pairs of the CpG Islands; a file in the result folder have whole sequences of islands and some other details.

> You can also add your own DNA sequence files to the `test\` folder and choose it to be executed as the program begins.

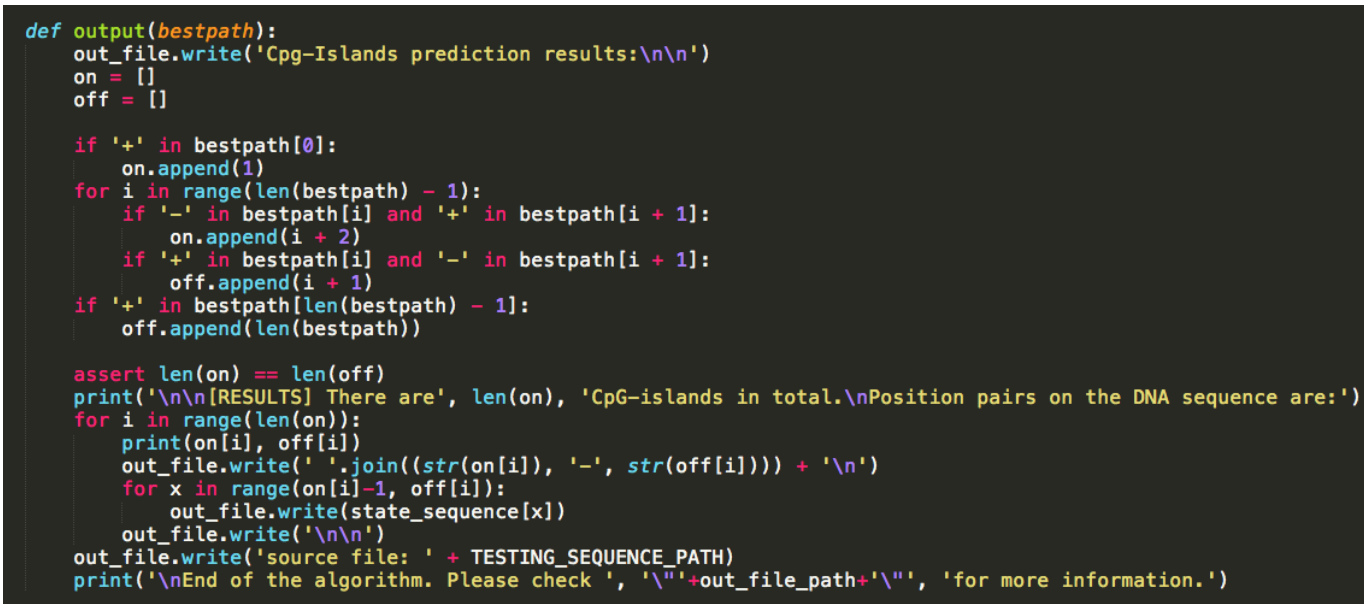
Code Excerpt

user interaction: allows user to choose the testing file



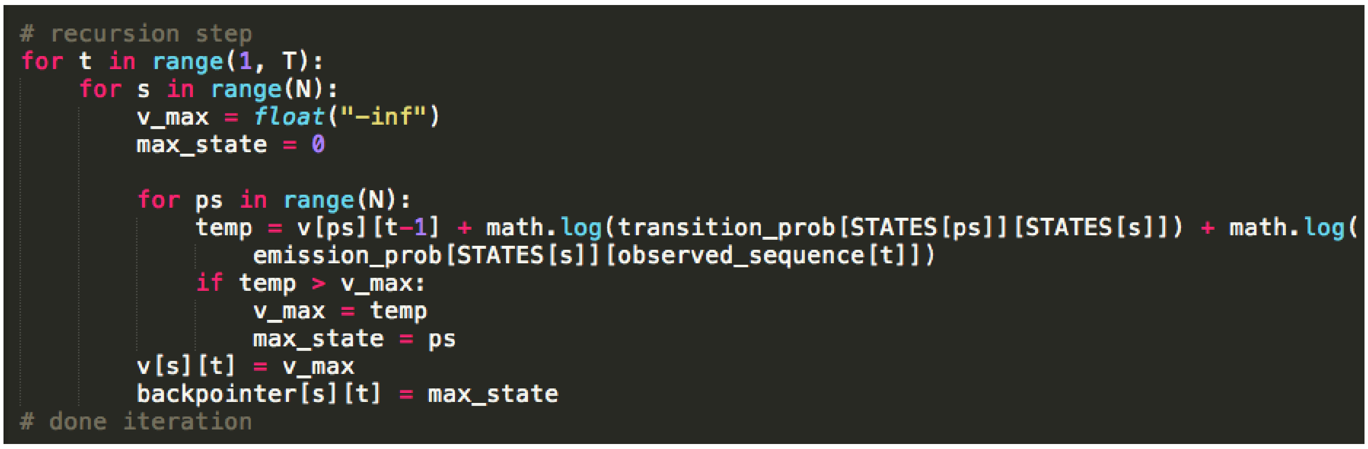
```
print('Welcome to our CpG islands Detector!')
files = [ f for f in listdir('data/test') if isfile(join('data/test', f))]
print('Please choose one of the testing files: ' + str(files))
user_input = input('Your input dataset: ')

while user_input not in files:
    print('Please choose a file starting with "testing".')
    user_input = input('Your input dataset: ')

TESTING_SEQUENCE_PATH = 'data/test/' + user_input
```

output: output a file showing the CpG islands and the testing file path



```
def output(bestpath):
    out_file.write('Cpg-Islands prediction results:\n\n')
    on = []
    off = []

    if '+' in bestpath[0]:
        on.append(1)
    for i in range(len(bestpath) - 1):
        if '-' in bestpath[i] and '+' in bestpath[i + 1]:
            on.append(i + 2)
        if '+' in bestpath[i] and '-' in bestpath[i + 1]:
            off.append(i + 1)
    if '+' in bestpath[len(bestpath) - 1]:
        off.append(len(bestpath))

    assert len(on) == len(off)
    print('\n\n[RESULTS] There are', len(on), 'CpG-islands in total.\nPosition pairs on the DNA sequence are:')
    for i in range(len(on)):
        print(on[i], off[i])
        out_file.write(' '.join((str(on[i]), '-', str(off[i]))) + '\n')
        for x in range(on[i]-1, off[i]):
            out_file.write(state_sequence[x])
        out_file.write('\n\n')
    out_file.write('source file: ' + TESTING_SEQUENCE_PATH)
    print('\nEnd of the algorithm. Please check ', '\"'+out_file_path+'\"', 'for more information.')
```

log function: provides more accrate comparison between Viter values



```
# recursion step
for t in range(1, T):
    for s in range(N):
        v_max = float("-inf")
        max_state = 0

        for ps in range(N):
            temp = v[ps][t-1] + math.log(transition_prob[STATES[ps]][STATES[s]]) + math.log(
                emission_prob[STATES[s]][observed_sequence[t]])
            if temp > v_max:
                v_max = temp
                max_state = ps
        v[s][t] = v_max
        backpointer[s][t] = max_state
# done iteration
```

What Each Team Member Learned

Tianyuan Fu: HMM is a model that is not formally taught in class, and that is the point of chanllenges and attractiveness. Understanding the model is not the easy thing, involing many concepts from statistics. But during the process of accepting the concepts, I am happy to see that I can apply concepts previously learnt in lectures, most of which are high-level views, and that eased the carryout of the whole project. It would be so helpful to have such an experience since new models of AI and ML are emerging. Having the experience of this project makes me confident to confront projects with more difficulty in the future.

Jinghan Lu: I have learned the Hidden Markov Models which allows us to think about observed and hidden events as causal factors in a probabilistic model and the Viterbi algorithm which allows us to find the most possible state sequence given an HMM and an observed sequence. Additionally, I have learned how HMM can be also applied to many different applications, such as sequence recognition and speech recogition.

Possible Add-ons

- Input side: support to more biological data format like FASTA.
- Processing side: the current method for calculating the probability can be improved to have less complexity
- Output side: a graphical user interface (GUI) can be implemented to display the results.

References

1. GitHub Repository: https://github.com/devanshdalal/cpg-island-prediction-HMM A smaple implementation of CpG Island Detector with HMM and Viterbi Algorithm. We have our training data set from this project, but different concepts of implementing the program.
2. HMM slides from CMU: https://www.cs.cmu.edu/~02710/Lectures/HMMs.pdf Provided some hint and insights to HMM.
3. National Center for Biotechnology Information: https://www.ncbi.nlm.nih.gov/genome/gdv/browser/ NCBI provides us a dataset of human DNA, Chromesome 1, which we used the first 1110 lines of this dataset as one of the testing file.
4. HMM Chapter from Standford: https://web.stanford.edu/~jurafsky/slp3/A.pdf We used this chapter of HMM as a tutorial for learning HMM and implmenting Viterbi algorithm.