BioE241: Probabilistic Modelling in Computational Biology
Final Project: Exploring methods on identifying and predicting CpG islands on a given sequence

Prithiv Natarajan
14.05.2019

Preface

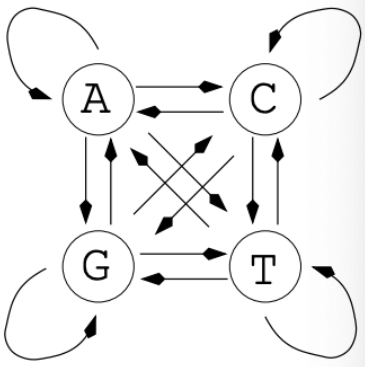
Genomic sequences consists of base pairs A,G,C,T as is common knowledge. The 'CG' combination is known to occur infrequently due to chemical properties. It has been shown that in the areas where the genes are codified, the occurrence of CG pairs is common due to alteration in chemical processes. High occurrence of CG combination is CpG island and they are important because they allow for faster search for genes in a genome sequence.

In this project I try to implement as well as review some methods that will allow one to make extensive analysis on CpG islands given a random sequences. Using Hidden Markov Models, Viterbi algorithm and some Deep Learning model, there is certainly more clarity gained in understanding how researchers go about addressing such large scale problems.

The data given is a long multi-fasta sequence file and a corresponding classification which tells if the base comes from a CpG island or not. Using HMM where the hidden variables are (base, sign) combination where the bases are A,G,C,T and the sign is '+' if the base is in a CpG island and '-' if not. Using HMM learning of emission and transition probabilities is done by counting the desired transition over the total number of transitions. For eg: P(Seq(x+1) = A+ | Seq(x) = C-) = probability of the next position coming from the island given the previous did not will be the summation of this desired transition over all transitions with the previous position as C-. The priors are based on frequencies of the base occurring in the non-CpG island parts of the training set. The first base does not belong to CpG island.

HMM

What leads to the Markov chain is the idea of the latter position of a sequence depending on its predecessor. The arrows in the image show possible transitions.



Suppose we are given a sequence of x1,x2,x3…xL the Probability

of that a Markov Chain will give out the exact sequence will depend on the successor and predecessor only which is the Markov Chain Property.

Using this we define the boundary states of start and end points. Following this, we determine e the transition states empirically by

$$a_{st}^{+} = \frac{c_{st}^{+}}{\sum_{t'} c_{st'}^{+}}$$

Where C(st) is the number if positions in CpG islands at which state s is
followed by state t which constitutes our + transition matrix and and the - transition is based on non-CpG islands. Using this we can ask if x = x1,x2…xL comes from a CpG island or not.

$$P(x \mid \text{Model}^{+}) = \prod_{i=0}^{L} a_{x_i x_{i+1}} \qquad S(x) = \log \frac{P(x \mid \text{Model}^{+})}{P(x \mid \text{Model}^{-})} = \sum_{i=0}^{L} \log \frac{a^{+}_{x_{i-1}x_i}}{a^{-}_{x_{i-1}x_i}}$$

Using HMMs we merge the two transition matrices along with emission probabilities to predict the most probable path in the sequence.

$$\pi = (\pi_1, \pi_2, \ldots, \pi_L)$$

A path is a sequence of states in a model

Given a sequence: $x = (x_1, \ldots, x_L)$ $\qquad P(x, \pi) = a_{0\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}}$

We do not usually know the path through the model so we try to decode the sequence. We compute the most probable path recursively:

$$\pi^{*} = \arg\max_{\pi} P(x, \pi).$$

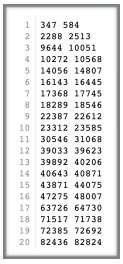
Using Viterbi algorithm: $v_l(i+1) = e_l(x_{i+1}) \max_{k \in Q}(v_k(i) a_{kl}).$

We obtain a dynamic programming matrix, using the HMM for CpG islands a set of possible values will look like this:

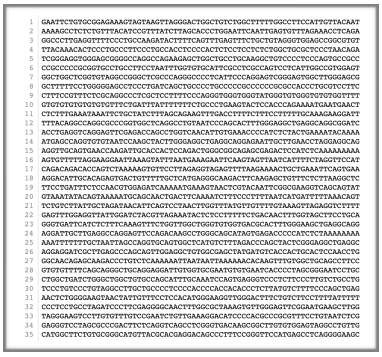
x_0	x_1	x_2	x_3	…	x_{i-2}	x_{i-1}	x_i	x_{i+1}
	A_+	A_+	A_+	…	$\mathbf{A}_+$	A_+	A_+	…
	C_+	$\mathbf{C}_+$	$\mathbf{C}_+$	…	C_+	$\mathbf{C}_+$	C_+	
	$\mathbf{G}_+$	G_+	G_+	…	G_+	G_+	$\mathbf{G}_+$	
	T_+	T_+	T_+	…	T_+	T_+	T_+	
0	A_-	A_-	A_-	…	A_-	A_-	A_-	
	C_-	C_-	C_-	…	C_-	C_-	C_-	
	G_-	G_-	G_-	…	G_-	G_-	G_-	
	T_-	T_-	T_-	…	T_-	T_-	T_-	

			sequence			
	v		C	G	C	G
	0	1	0	0	0	0
	A+	0	0	0	0	0
	C+	0	**.13**	0	**.012**	0
State	G+	0	0	**.034**	0	**.0032**
	T+	0	0	0	0	0
	A−	0	0	0	0	0
	C−	0	.13	0	.0026	0
	G−	0	0	.010	0	.00021
	T−	0	0	0	0	0

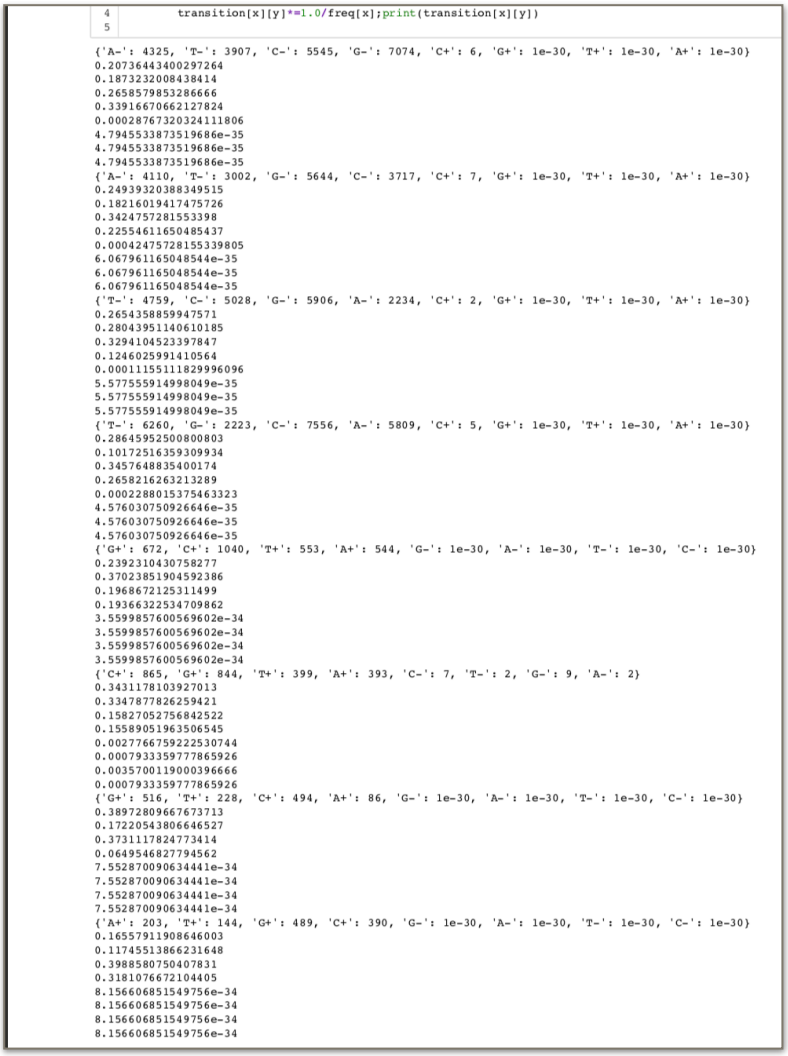
Method:



```
347 584
2288 2513
9644 10051
10272 10568
14056 14807
16143 16445
17368 17745
18289 18546
22387 22612
23312 23585
30546 31068
39033 39623
39892 40206
40643 40871
43871 44075
47275 48007
63726 64730
71517 71738
72385 72692
82436 82824
```



1. Read file for training (The training data contains a multi-fasta file with already specified CpG island ranges)
2. Construct a map that puts '1' when CpG island is present in that interval: [0,0,0,0,1,1,1,1,0,0,0...]
3. Construct a hidden layer array that puts '+' as a suffix to the nucleotide that is present in the island and '-' if it is not



```
transition[x][y]*=1.0/freq[x];print(transition[x][y])
```

```
{'A-': 4325, 'T-': 3907, 'C-': 5545, 'G-': 7074, 'C+': 6, 'G+': 1e-30, 'T+': 1e-30, 'A+': 1e-30}
0.20736443400297264
0.1873232008438414
0.2658579853286666
0.33916670662127824
0.00028767320324111806
4.79455338735196 86e-35
4.79455338735196 86e-35
4.79455338735196 86e-35
{'A-': 4110, 'T-': 3002, 'G-': 5644, 'C-': 3717, 'C+': 7, 'G+': 1e-30, 'T+': 1e-30, 'A+': 1e-30}
0.24939320388349515
0.18216019417475726
0.3424757281553398
0.22554611650485437
0.0004247572815533 9805
6.067961165048544e-35
6.067961165048544e-35
6.067961165048544e-35
{'T-': 4759, 'C-': 5028, 'G-': 5906, 'A-': 2234, 'C+': 2, 'G+': 1e-30, 'T+': 1e-30, 'A+': 1e-30}
0.2654358859947571
0.28043951140610185
0.3294104523397847
0.1246025991410564
0.00011155111829996096
5.5775559149980 49e-35
5.5775559149980 49e-35
5.5775559149980 49e-35
{'T-': 6260, 'G-': 2223, 'C-': 7556, 'A-': 5809, 'C+': 5, 'G+': 1e-30, 'T+': 1e-30, 'A+': 1e-30}
0.28645952500800803
0.10172516359309934
0.3457648835400174
0.2658216263213289
0.00022880153754633 23
4.5760307509266 46e-35
4.5760307509266 46e-35
4.5760307509266 46e-35
{'G+': 672, 'C+': 1040, 'T+': 553, 'A+': 544, 'G-': 1e-30, 'A-': 1e-30, 'T-': 1e-30, 'C-': 1e-30}
0.2392310430758277
0.37023851904592386
0.1968672125311499
0.19366322534709862
3.5599857600569602e-34
3.5599857600569602e-34
3.5599857600569602e-34
3.5599857600569602e-34
{'C+': 865, 'G+': 844, 'T+': 399, 'A+': 393, 'C-': 7, 'T-': 2, 'G-': 9, 'A-': 2}
0.3431178103927013
0.3347877826259421
0.15827052756842522
0.15589051963506545
0.00277667592225 30744
0.00079333359777865926
0.00357001190003 96666
0.00079333359777865926
{'G+': 516, 'T+': 228, 'C+': 494, 'A+': 86, 'G-': 1e-30, 'A-': 1e-30, 'T-': 1e-30, 'C-': 1e-30}
0.38972809667673713
0.17220543806646527
0.3731117824773414
0.0649546827794562
7.5528700906344 41e-34
7.5528700906344 41e-34
7.5528700906344 41e-34
7.5528700906344 41e-34
{'A+': 203, 'T+': 144, 'G+': 489, 'C+': 390, 'G-': 1e-30, 'A-': 1e-30, 'T-': 1e-30, 'C-': 1e-30}
0.16557911908646003
0.11745513866231648
0.3988580750407831
0.3181076672104405
8.1566606851549 756e-34
8.1566606851549 756e-34
8.1566606851549 756e-34
8.1566606851549 756e-34
```

4. Create a dictionary where the nucleotides not in an island will have their frequencies along [A-,G-,C-,T-]
   4.1. Assign a random small number to [A+,C+,T+,G+] along with summing their frequency
5. Calculate the transition matrix for each of the 8 states and combination of those
6. Calculate the emission matrix
7. Use the Viterbi algorithm to back trace the best path possible using best (or max) resulting product of probability of each state

Results:

Using the Viterbi algorithm where log of probabilities was used the log(probability of

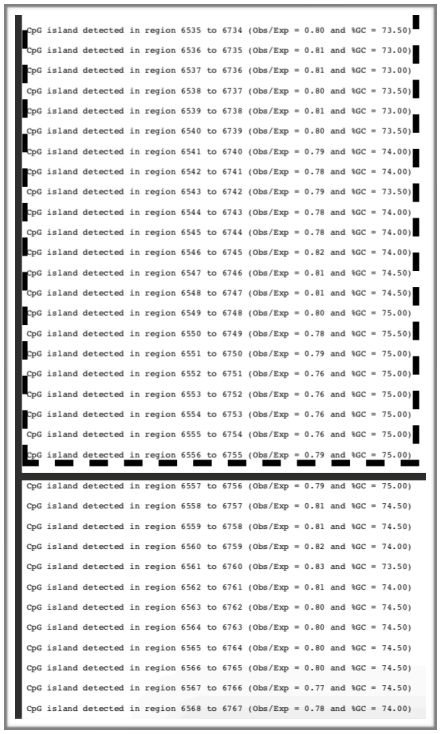
$$\log(prob) = argmax_{x \in S} V_{L-1,x}$$

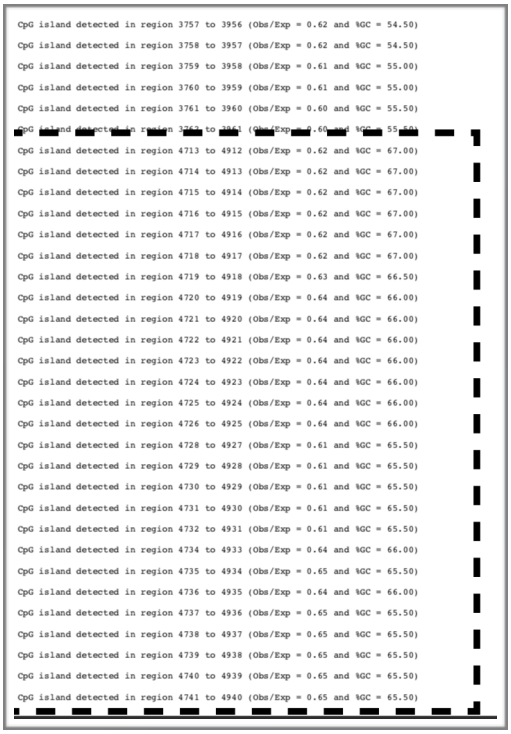
```
4075 5678
6343 6755
7340 7544
8161 8418
31893 32279
41518 41883
47627 48242
49232 49400
53230 54102
<ipykernel.iostream.OutStream object at 0x10dad0be0> Completed 2.7578141689300537 seconds
```

most likely sequence was found to be -100094.5894 for the test data. The image shows the ranges found for the CpG island. To validate the results I used http://bioinformatics.org/sms2/cpg_islands.html which is a site developed to identify CpG islands in a sequence.

The results of the site were huge in size, however, I have marked the ranges the code using HMM has correctly predicted.

‘





There are more CpG islands present as mentioned in the site, however, HMM model could predict about 7 out of 12 or 13 islands where the rest of the islands were present in the range above 60000 sequence position while the HMM code stops at 54000.

*The Viterbi algorithm implemented for calculating most likely states of the hidden sequence has a complexity of O(|S| * |S| * N) where N is the length of the sequence and S is the set of all possible*

states. Since we have |S|=8 for all indexes of the sequence. Therefore final complexity will be O(kN) ~ O(N)

To explore into CpG islands and detecting them I tried to implement an already implemented code through GitHub using the hmmlearn package in Python.

```
states = ["0", "A+", "C+", "G+", "T+", "A-", "C-", "G-", "T-"]
```

```
start_probability = np.array([0,0.0725193,0.1637630,0.1788242,0.0754545,0.1322050,0.1267006,0.1226380,0.1278950])
```

```
emission_probability = np.array([
  [0.0,0.0,0.0,0],
  [1,0,0,0],
  [0,1,0,0],
  [0,0,1,0],
  [0,0,0,1],
  [1,0,0,0],
  [0,1,0,0],
  [0,0,1,0],
  [0,0,0,1]
])
```

```
transition_probability = np.array([
    [0, 0.0725193, 0.163763, 0.1788242, 0.0754545, 0.1322050, 0.1267006, 0.1226380, 0.1278950],
    [0.001, 0.1762237, 0.2682517, 0.4170629, 0.1174825, 0.0035964, 0.0054745, 0.0085104, 0.0023976],
    [0.001, 0.1672435, 0.3599201, 0.267984, 0.1838722, 0.0034131, 0.0073453, 0.005469, 0.0037524],
    [0.001, 0.1576223, 0.3318881, 0.3671328, 0.1223776, 0.0032167, 0.0067732, 0.0074915, 0.0024975],
    [0.001, 0.0773426, 0.3475514, 0.375944, 0.1781818, 0.0015784, 0.0070929, 0.0076723, 0.0036363],
    [0.001, 0.0002997, 0.0002047, 0.0002837, 0.0002097, 0.2994005, 0.2045904, 0.2844305, 0.2095804],
    [0.001, 0.0003216, 0.0002977, 0.0000769, 0.0003016, 0.3213566, 0.2974045, 0.0778441, 0.3013966],
    [0.001, 0.0001768, 0.000238, 0.0002917, 0.0002917, 0.1766463, 0.2385224, 0.2914165, 0.2914155],
    [0.001, 0.0002477, 0.0002457, 0.0002977, 0.0002077, 0.2475044, 0.2455084, 0.2974035, 0.2075844]
])
```

```
(1154, 1205)
(1881, 1925)
(3233, 3383)
(7128, 7176)
(10026, 10301)
(13238, 13394)
(15868, 15939)
(18828, 18965)
(19711, 19764)
(20306, 20363)
(20699, 20797)
(21672, 21733)
(22511, 22664)
(24119, 24279)
(27086, 27252)
(30928, 31109)
(32900, 33072)
(35841, 35878)
(40718, 40785)
(41394, 41496)
```

```
('log-liklihood:', -72250.99892943806)
(10029, 10302)
(13241, 13412)
(<ipykernel.iostream.OutStream object at 0x10533d3d0>, 'Completed', 3.6185669898986816, 'seconds')
```

Another sequence for this. I used NCBI to attain some new data on Human genome sequence (only a part of it) to see how the HMM model compares with the hmmlearn package. It can be seen that the HMM model could predict only two ranges while the hmmlearn could predict 20 CpG islands. The start probability was taken from different experimental values tested by the author.

Discussion and Conclusion:

CpG islands are important to find patterns in a genomic sequence. There are many models present right now in this computing field with DeepCpG and Pysster that allow you to perform different analysis on CpG islands using Bidirectional RNN architectures. The methods I have tried to implement is by far simple and basic as compared to the said models. However, the learning through this has been immense as a firm understanding is required to build better and more complex models.

Future work is to implement a Posterior probabilities as a way to calculate the probability of finding sequence belonging to an HMM model M (P(x|M) and using the Baum-Welch algorithm to train the parameters of model M.

Further studies can be done in characterising protein families along side adding insert and delete states to improve the HMM topology.

Reference:

1. Markov chains and Hidden Markov Models, Grundlagen der Bioinformatik, SS'08, D. Huson, June 16, 2008
2. Redefining CpG islands using hidden Markov models, Hao Wu, Brian Caffo, Harris A. Jaffee, and Rafael A. Irizarry*, 2010
3. GitHub repository for hmmlearn: https://github.com/Wishmitha/cpg-island-detection-hmm/blob/master/cpg_island_detector.py
4. Sequence Manipulation Site: http://bioinformatics.org/sms2/cpg_islands.html