Report for LAB3: Probabilistic Models
Bioinformatics

Ricardo Brancas	Margarida Ferreira	Felipe Gorostiaga	Benedict Schubert
83557	80832	95383	95034

25th October 2019

Group I. CpG Island characteristics

a CpG Islands, DNA stats and ORF Finder

CpG sites and CpG islands

CpG sites are regions of DNA where a cytosine nucleotide is followed by a guanine nucleotide (`CG`). Cytosines in CpG dinucleotides can be methylated to form 5-methylcytosines. In mammals, 70% to 80% of CpG cytosines are methylated, and such methylated cytosine has a high probability of becoming a thymine. Places where the cytosine is not methylated (and therefore have a high frequency of CpG sites) are called **CpG islands**.

CpG Islands tool from the Sequence Manipulation Suite (SMS) [1]

This tool allows us to search for potential CpG islands in a DNA sequence, using the following formula:

$$\frac{obs}{exp}$$

where *obs* is the number of observed `GC` occurrences, and *exp* is the expected number of `GC` occurrences in a window, computed as:

$$\frac{count(\texttt{C}) \cdot count(\texttt{G})}{window\ length}$$

CpG islands are then defined as sequence ranges where the $\frac{obs}{exp}$ value is greater than 0.6 and the `GC` content is greater than 50%. The calculation is performed using a moving window of size 200.

This tool detects only sequences of size up to 200, which makes the results hard to interpret – in the DNA sequence presented in the file `genome.fa`, 440 CpG islands of size 200 were identified.

[1] `https://www.bioinformatics.org/sms2/cpg_islands.html`

EMBOSS CpGPlot [2]

EMBOSS CpGPlot also allows us to identify CpG islands in nucleotide sequences. This tool, however, lets us set values for the following algorithm parameters:

- Window size (default: 100)
- Minimum sequence length (default: 200)
- Minimum observed to expected ratio (default: 0.6)
- Minimum percentage of `GC` occurrences (default: 50%)

With this tool, there is no upper limit to the length of the CpG islands detected.

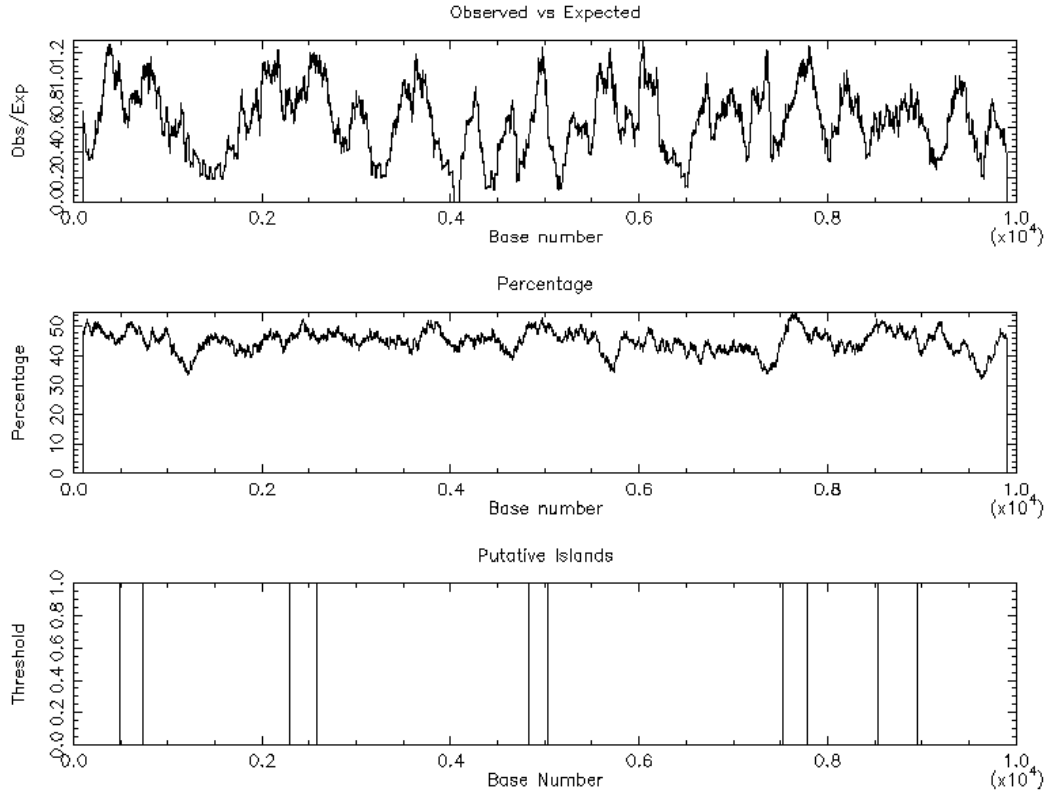
With the window size set to 200 and the rest of the settings set to default, we get a much more interpretable result than when using the SMS tool, as larger and fewer CpG islands are identified – 5 islands with lengths between 206 and 411 (figure 1).

Other experiences were made with this tool varying the considered window size. With window size of 50, we found no CpG islands of size 200. However, changing the minimum length to 100 bps. we were able to find 3 CpG islands with lengths up to 120 nucleotides (figure 2). These 3 islands were all included in the previous 5, found with a window size of 200.

We finally tried the tool with a window size of 100, which was the default value for this parameter. We were able to find 2 CpG islands of lengths of 221 to 232 nucleotides, which are also included in the first set of 5 found with a window size of 200 (figure 3).

Variations in the remaining two parameters (minimum observed to expected ratio and minimum percentage of `GC` occurrences) were also made, but the results were exactly as it would be expected – the higher these values, the more restricted our search and therefore fewer sequences are returned. We also observed that even small increases in these values made the tool unable to find any CpG island at all. On the other hand, if these values are lowered we obtain more and longer hits. However, we must take into consideration that these are less relevant results, as they have fewer CpG sites on average than in the previously obtained sequences.

[2] `https://www.ebi.ac.uk/Tools/seqstats/emboss_cpgplot`

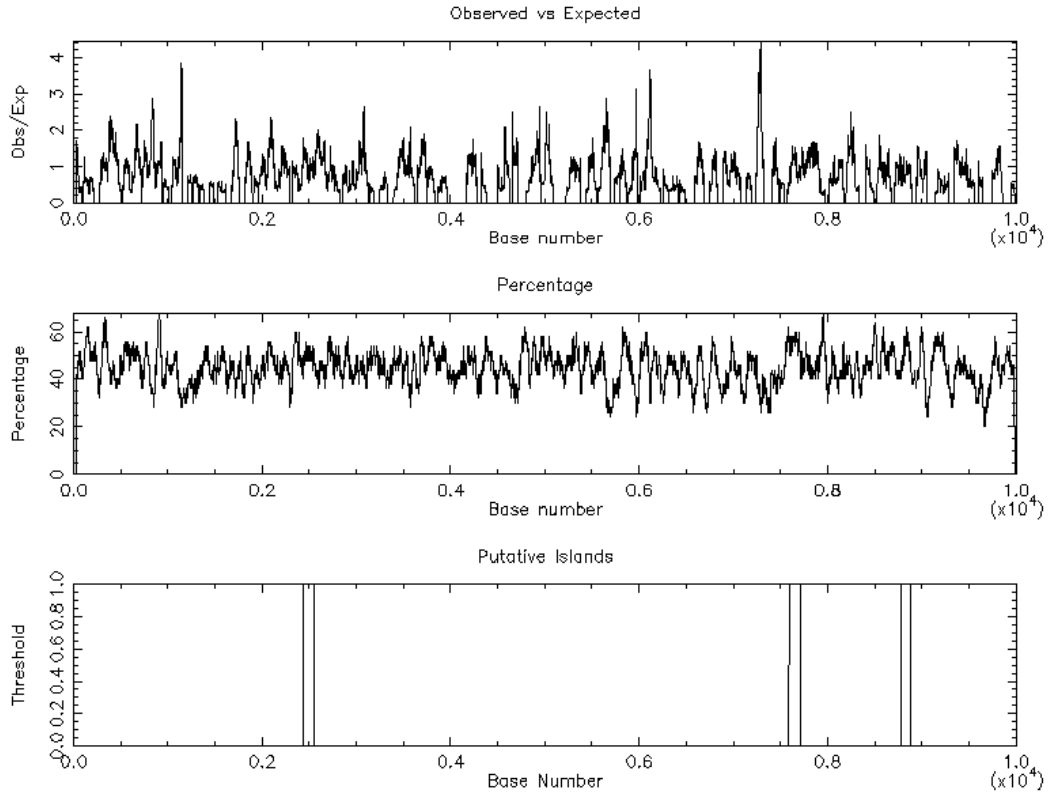



```
CPGPLOT islands of unusual CG composition
Sequence from 1 to 10009

     Observed/Expected ratio > 0.60
     Percent C + Percent G > 50.00
     Length > 200

 Length 247 (489..735)
 Length 293 (2287..2579)
 Length 206 (4830..5035)
 Length 256 (7531..7786)
 Length 411 (8538..8948)
```

Figure 1: Output of `EMBOSS CpGPlot` with settings window size = 200, Minimum sequence length = 200, minimum observed to expected ratio = 0.6, minimum percentage of `GC` occurrences = 50%

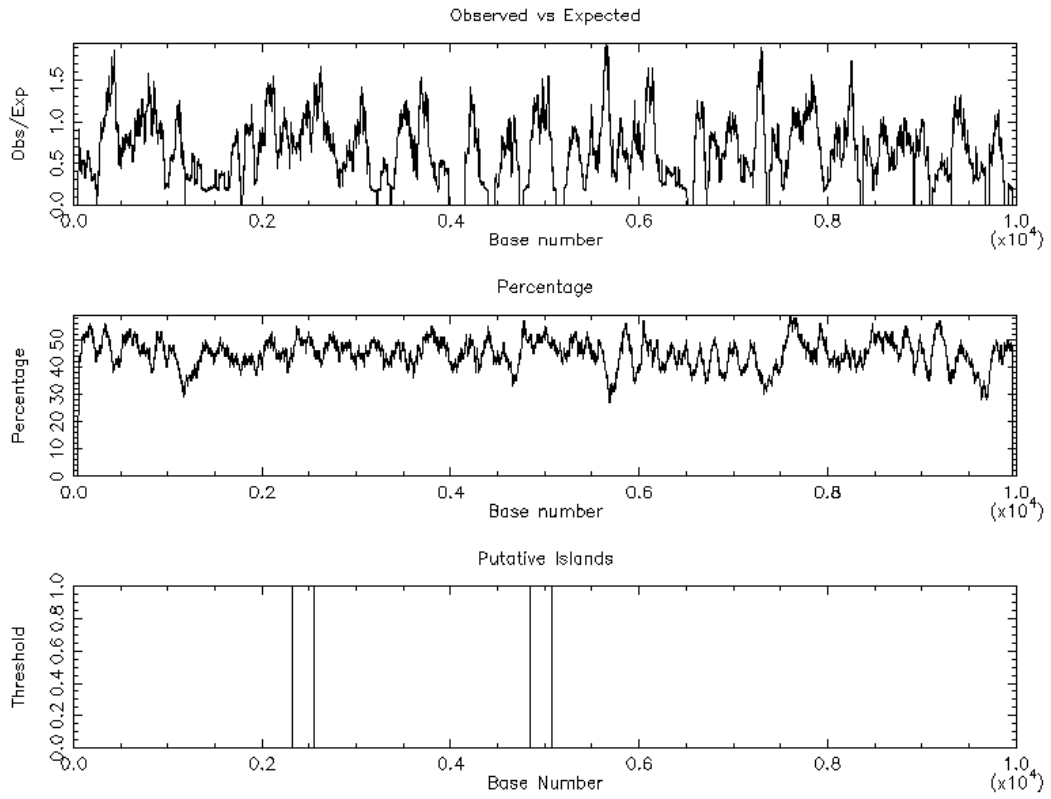



```
CPGPLOT islands of unusual CG composition
Sequence from 1 to 10009

     Observed/Expected ratio > 0.60
     Percent C + Percent G > 50.00
     Length > 100

 Length 119 (2440..2558)
 Length 119 (7592..7710)
 Length 101 (8781..8881)
```

Figure 2: Output of `EMBOSS CpGPlot` with settings window size = 50, Minimum sequence length = 100, minimum observed to expected ratio = 0.6, minimum percentage of `GC` occurrences = 50%




```
CPGPLOT islands of unusual CG composition
Sequence from 1 to 10009

     Observed/Expected ratio > 0.60
     Percent C + Percent G > 50.00
     Length > 200

 Length 221 (2326..2546)
 Length 232 (4840..5071)
```

Figure 3: Output of `EMBOSS CpGPlot` with settings window size = 100, Minimum sequence length = 200, minimum observed to expected ratio = 0.6, minimum percentage of `GC` occurrences = 50%

Open Reading Frames (ORFs)

A **reading frame** is a way of dividing the sequence of nucleotides in a nucleic acid (DNA or RNA) molecule into a set of consecutive, non-overlapping triplets. Where these triplets equate to amino acids or stop signals during translation, they are called codons.

A **open reading frame (ORF)** is the part of a reading frame that has the ability to be translated. An ORF is a continuous stretch of codons that begins with a start codon (usually `AUG`) and ends at a stop codon (usually `UAA`, `UAG` or `UGA`).

ORF Finder tool from the Sequence Manipulation Suite [3]

ORF finder parameters:

- Allowed start codons
- Minimum length of ORF
- Reading frame (1, 2, 3) and strand (direct or reverse) on which to search
- Which genetic code to use

In general, independently of other parameters, the larger minimum length of ORF, the fewer sequences are returned.

When using the standard genetic code, only small ORFs are found for sequence X, none with length over 200 codons. However, using a vertebrate mitochondrial genetic code (as indicated by group I.b), we found 12 ORFs more than 200 codons long in reading frames 1, 2 and 3 on the direct strand. On the reverse strand, no ORFs were found.

DNA Stats tool from the Sequence Manipulation Suite [4]

The results of applying this tool to the entire string X are shown in table 1.

We also applied this tool to the CpG island identified in sequence X, between bps. 4840 and 5071. Results are shown in table 2. We can see, as expected, that the CpG island has a much higher percentage of cytosine (41%) than the sequence as a whole (28%). On the other hand, the percentage of thymine is lower on the CpG island (20%) than in the whole sequence X (25%), as the methylated cytosine nucleotides turn into thymines. Finally, we can note that the percentage of `CG` occurrences is higher in the CpG island (3.9%) than in the whole sequence (3.02%), and, complementarily, there is a significantly lower percentage of `TG` dinucleotides (0.43%) in the CpG island than considering the whole sequence (3.28%) – `TG` dinucleotides result from the methylation of cytosine in previously `CG` dinucleotides.

[3] `https://www.bioinformatics.org/sms2/orf_find.html`

[4] `https://www.bioinformatics.org/sms2/dna_stats.html`

Pattern:	Times found:	Percent
G	1682	16.80
A	2964	29.61
T	2550	25.48
C	2813	28.10
GG	367	3.67
GA	519	5.19
GT	328	3.28
GC	468	4.68
AG	587	5.87
AA	948	9.47
AT	670	6.69
AC	759	7.58
TG	425	4.25
TA	748	7.47
TT	759	7.58
TC	617	6.17
CG	302	3.02
CA	749	7.48
CT	793	7.92
CC	969	9.68
G, C	4495	44.91
A, T	5514	55.09

Table 1: Result of applying the DNA Stats tool from the SMS to the DNA sequence X. 'G', 'C', and 'GC' frequency values are shaded in gray. Zero values were omitted.

Pattern:	Times found:	Percentage:
g	20	8.62
a	70	30.17
t	46	19.83
c	96	41.38
gg	2	0.87
ga	6	2.6
gt	1	0.43
gc	11	4.76
ag	6	2.6
aa	20	8.66
at	10	4.33
ac	34	14.72
tg	3	1.3
ta	17	7.36
tt	12	5.19
tc	14	6.06
cg	9	3.9
ca	26	11.26
ct	23	9.96
cc	37	16.02
g,c	116	50
a,t	116	50

Table 2: Result of applying the DNA Stats tool from the SMS to CpG island from bp. 4840 to bp. 4071 of sequence X. 'G', 'C', and 'GC' frequency values are shaded in gray. Zero values were omitted.

b GenBank Nucleotide Match [5]

After running sequence X through the GenBank database, we got a 100% match with sequence `AP009202.1`. It is the complete genome of mitochondrial DNA of *Abalistes stellaris*, also known as starry triggerfish (figure 4).

The CpG islands identified before, from bp. 2326 to 2546 and from bp. 4840 to 5071 correspond to ribossomal RNA and the gene ND2, respectively. This gene encodes the protein *NADH dehydrogenase subunit 2*.

Figure 4: A happy starry triggerfish.

Group II. Probability and statistics concepts

a The sum of the probabilities of all possible sequences of states of length L is 1

Let us consider a simpler case for $L = 3$. We will observe the expressions' transformations in this simpler setting, and then apply the same ideas to the general setting. The sum of the probabilities of all possible sequences of states of length 3 is then:

[5] `https://blast.ncbi.nlm.nih.gov/Blast.cgi`

$$\sum_{X_1=x_1}\sum_{X_2=x_2}\sum_{X_3=x_3} P(x_1)\cdot\prod_{i=2}^{3} a_{x_{i-1}x_i} \tag{1}$$

$$= \sum_{X_1=x_1}\sum_{X_2=x_2}\sum_{X_3=x_3} P(x_1)\cdot a_{x_1}a_{x_2}\cdot a_{x_2}a_{x_3} \quad \text{[expand product]} \tag{2}$$

$$= \sum_{X_1=x_1}\sum_{X_2=x_2}\sum_{X_3=x_3} P(x_1)\cdot P(x_2|x_1)\cdot P(x_3|x_2) \tag{3}$$

$$= \sum_{X_1=x_1}\sum_{X_2=x_2}\sum_{X_3=x_3} P(x_1)\cdot P(x_2|x_1)\cdot P(x_3|x_2,x_1) \quad \text{[Markov property]} \tag{4}$$

$$= \sum_{X_1=x_1}\sum_{X_2=x_2}\sum_{X_3=x_3} P(x_1)\cdot \frac{P(x_2,x_1)}{P(x_1)}\cdot\frac{P(x_3,x_2,x_1)}{P(x_2,x_1)} \quad \text{[Conditional probability]} \tag{5}$$

$$= \sum_{X_1=x_1}\sum_{X_2=x_2}\sum_{X_3=x_3} P(x_3,x_2,x_1) \tag{6}$$

$$= 1 \tag{7}$$

In the first step (1 to 2), we simply expand the product. Then, in 3 we write the value $a_{x_{i-1}x_i}$ as the probability of going to state x_i coming from state x_{i-1}, according to its definition: $a_{x_{i-1}x_i} = P(x_i|x_{i-1})$. In 4 we make use of the Markov "memorylessness" property, which states that only the last visited state matters to determine the probability of being in the current state: $P(x_i|x_{i-1}) = P(x_i|x_{i-1},...,x_1)$. In 5 we apply the definition of conditional probability directly (see 17). In 6 we cut the denominators, and, finally, in 7 we apply the basic rule of probability that states that the sum of the probabilities of all possible events equals 1.

Now, we can generalize the calculation for the sum of the probabilities of all possible sequences of states of length L:

$$\sum_{X_1=x_1} \sum_{X_2=x_2} ... \sum_{X_L=x_L} P(x_1) \cdot \prod_{i=2}^{L} a_{x_{i-1}x_i} \tag{8}$$

$$= \sum_{X_1=x_1} \sum_{X_2=x_2} ... \sum_{X_L=x_L} P(x_1) \cdot \prod_{i=2}^{L} P(x_i|x_{i-1}) \tag{9}$$

$$= \sum_{X_1=x_1} \sum_{X_2=x_2} ... \sum_{X_L=x_L} P(x_1) \cdot \prod_{i=2}^{L} P(x_i|x_{i-1}, x_{i-2}, ..., x_2, x_1) \quad \text{[Markov property]} \tag{10}$$

$$= \sum_{X_1=x_1} \sum_{X_2=x_2} ... \sum_{X_L=x_L} P(x_1) \cdot \prod_{i=2}^{L} \frac{P(x_i, x_{i-1}, ..., x_2, x_1)}{P(x_{i-1}, x_{i-2}, ..., x_2, x_1)} \quad \text{[Conditional prob.]} \tag{11}$$

$$= \sum_{X_1=x_1} \sum_{X_2=x_2} ... \sum_{X_L=x_L} P(x_i, x_{i-1}, ..., x_2, x_1) \quad \text{[induction step from (6)]} \tag{12}$$

$$= 1 \tag{13}$$

b Most probable path of HMM

$$\pi^* = \arg\max_{\pi} P(x, \pi) \tag{14}$$

$$= \arg\max_{\pi} P(\pi|x) \cdot P(x) \tag{15}$$

$$= \arg\max_{\pi} P(\pi|x) \tag{16}$$

From (14) to (15) we directly apply the conditional probability definition (note that $P(x, \pi) = P(\pi, x)$):

$$P(A|B) = \frac{P(A, B)}{P(B)} \iff P(A, B) = P(A|B) \cdot P(B) \tag{17}$$

From (15) to (16), we take into consideration the fact that the result of the $argmax$ function is not changed by the multiplication of a positive constant ($P(x)$ is constant in π). In fact, for any strictly monotonic function h:

$$\arg\max\ u = \arg\max\ h(u) \tag{18}$$

In the points where the first derivative of u is zero, the first derivative of $h(x)$ must be zero as well. This property of $argmax$ (and $argmin$) is also used very often when doing

calculations involving probabilities, since the logarithm – a strictly monotonic function – transforms multiplication of probabilities into addition of log-probabilities:

$$\arg\max \prod_{i=1}^{N} P(x) = \arg\max \sum_{i=1}^{N} logP(x) \tag{19}$$

c Probability of a 2-step transition

$$P(x_{i+2} = u | x_i = s) = \sum_{t \in A} P(u, t|s) \tag{20}$$

$$= \sum_{t \in A} P(u|t, s)P(t|s) \qquad \text{[Chain rule]} \tag{21}$$

$$= \sum_{t \in A} P(u|t)P(t|s) \qquad \text{[Markov property]} \tag{22}$$

$$= \sum_{t \in A} a_{st}a_{tu} \qquad \text{[transition probability def.]} \tag{23}$$

In the first step (20) we transform the probability of seeing u in state x_{i+2} given s by explicitly iterating over all symbols $t \in A$ to get the probability $\sum_{t \in A} P(u, t|s)$. We get from (20) to (21) by applying the conditional probability chain rule:

$$P(B, C|A) = P(B|A)P(C|A, B)$$

After using the Markov property in step (22) to let go of symbol s we can directly use the state transition definition to get to the desired term in (23).

Group III. Hidden Markov Decision Process

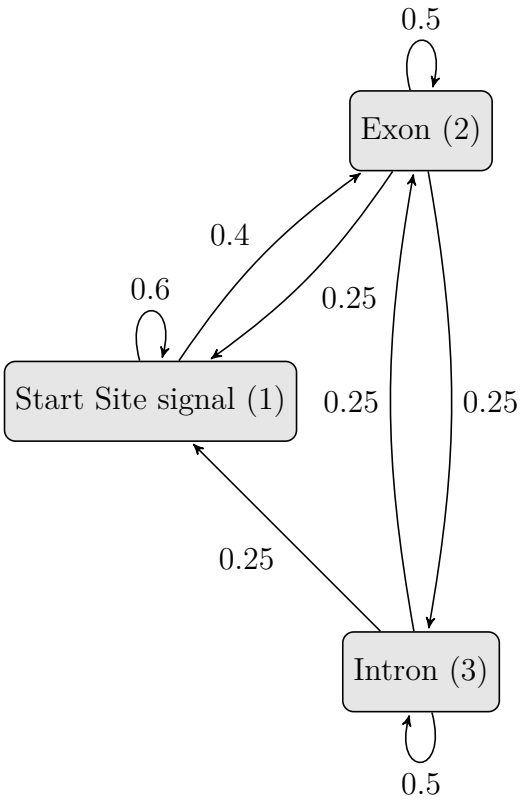
a Hidden Markov Model – graphical representation

In figure 5 we present a graphical representation of the hidden Markov model.

b Optimal State Sequence

The code in the file `viterbi.py` was run to use the Viterbi algorithm to find the optimal sequence of states π^* that generates the sequence `S = CATGCGGGTTATAAC`, in this case `211222221111112`. Note the option `--prob` can be used so that the program also outputs the probability of observing this sequence P[`S`]. The command to run the program as well as the output produced are shown in figure 6.

In algorithm 1 we present the pseudocode for the developed implementation. In this code $\boldsymbol{E}$ and $\boldsymbol{A}$ represent the matrices containing the emission probabilities and transition



(a) Transition model.

	A	T	G	C
Start S.s.	0.4	0.3	0.3	0
Exon	0.1	0.1	0.4	0.4
Intron	0.4	0.3	0	0.3

(b) Emission probabilities.

Figure 5: Hidden Markov model described in group III. The starting probabilities for each state are the same and equal to $\frac{1}{3}$.

probabilities, respectively. Furthermore, $\boldsymbol{\mu}_0$ represents the initial likelihoods of the different hidden states (in this case an uniform distribution). Note also the notation $\boldsymbol{M}_{:,i}$ which represents the i-th column vector of the matrix $\boldsymbol{M}$.

c Hidden State Sequence Probability

The probability $P[\mathtt{S}]$ can be computed by using the forward algorithm. For this sequence $P[\mathtt{S}] \approx 9.3865 \cdot 10^{-10}$.

The forward algorithm is a step of the Viterbi algorithm, and as such it would normally require no further operations to calculate this probability. However, since we took steps in order to ensure that the values do not underflow, that probability is lost, and needs to be re-computed at the end. Note that although the hidden state sequence, π^*, is always correct, the probability may still underflow.

As referred before, this functionality is implemented in the file `viterbi.py` and it is only needed to pass `--prob` as an argument in order to activate it.

```
$ python3 viterby.py --prob CATGCGGGTTATAAC
S    = CATGCGGGTTATAAC
π*   = 211222221111112
P[S] = 9.386459970616897e-10
```

Figure 6: Output of the implemented algorithm on the sequence `CATGCGGGTTATAAC`.

Algorithm 1: Viterbi Algorithm

Input : A sequence of observed states $\boldsymbol{X}_{0:L}$

Output: The most likely sequence of hidden states $\boldsymbol{\pi}^*$

$\boldsymbol{m}_0 \leftarrow \text{diag}(\boldsymbol{E}_{:,X_0})\boldsymbol{\mu}_0^\top$

for $i \in \{1, ..., L\}$ **do**

  $\boldsymbol{m}_i \leftarrow \text{diag}(\boldsymbol{E}_{:,X_i}) \max\{\boldsymbol{A}^\top \text{diag}(\boldsymbol{m}_{i-1})\}$

  $\boldsymbol{trace}_i \leftarrow \arg\max\{\boldsymbol{A}^\top \text{diag}(\boldsymbol{m}_{i-1})\}$

$\pi_L^* \leftarrow \arg\max\{\boldsymbol{m}_L\}$

for $i \in \{L-1, ..., 0\}$ **do**

  $\pi_i^* \leftarrow \boldsymbol{trace}_{t+1}[\pi_{i+1}^*]$

return $\boldsymbol{\pi}^*$

d Posterior probabilities

Using the backward algorithm we can compute the values $P[\pi_4 \mid X] \approx [0.5371, 0.4629, 0]$ and $P[\pi_9 \mid X] \approx [0.5371, 0.4629, 0]$. This means that $\hat{\pi}_4 = 1$ and $\hat{\pi}_9 = 1$ while $\pi_4^* = 2$ and $\pi_9^* = 1$. The difference between $\hat{\pi}$ and π^* in state 4 is due to the Viterbi algorithm considering the whole sequence, while the backward algorithm only takes into account the single state.