Review Classmates: Bioinformatics Application Challenge

Review by December 21, 11:59 PM PST

Reviews 3 left to complete

It looks like this is your first peer-graded assignment. Learn more

C. elegans project

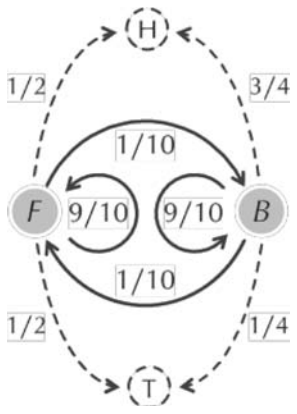
by Pantelis Katharios
Submitted on December 8, 2016

like Flag this submission

Analyzing the Crooked Casino

Given a string emitted by an HMM, the Viterbi algorithm solves the Decoding Problem, i.e., it finds the most likely sequence of hidden states that generated this string.

To solve the Decoding Problem for a given HMM, we will use Viterbi.py,, a Python script that we provide for this purpose. Viterbi.py simulates the HMM described by the following *HMM diagram* that models the crooked casino (with states *F* and *B* emitting symbols H and T).



In the crooked casino, you are placing bets on whether a dealer's coin will land on heads or tails. However, the dealer does not always use a fair coin with the probability of landing on heads equal to 0.5. Sometimes the dealer will switch out the coin and use a biased coin (with the probability of landing on head equal to 0.75) without you noticing. Here is the description of the HMM modelling the crooked dealer.

1. **∑:** Heads (H) and Tails (T)
2. *States:* Fair (F) and Biased (B)
3. *Transition* matrix:

	Transition To: **Fair**	Transition To: **Biased**
Current state: **Fair**	0.90	0.10
Current state: **Biased**	0.10	0.90

4. *Emission* matrix

	Emit: **Heads**	Emit: **Tails**
Current State: **Fair**	0.50	0.50
Current State: **Biased**	0.75	0.25

The starting probabilities are the same for all states.

The solution of the Decoding Problem is the most likely sequence of hidden states π that generated the emitted sequence *x* and the probability Pr(*x*, π) that the HMM emits *x* and follows the sequence of hidden states π. In addition to outputting this data, Viterbi.py also outputs the scores of nodes in the **Viterbi graph**.

To get started with Viterbi.py, first click here to download it. Save this file to a directory on your computer where you will work with it.

This file is written in Python, so in order to work with it, we will need to download Python from the Python website (choose version 3).

To make running Python as smooth as possible, we suggest downloading PyCharm Community Edition. Then open PyCharm, click "File" --> "Open" and select "Viterbi.py" from the directory where you saved it.

Note: You will not need to understand how to program in Python in order to complete this application challenge.

Run Viterbi.py (click the green triangle button in PyCharm). This will run Viterbi.py for the emitted string "THHHH" because of the line

observations = ('tails', 'heads', 'heads', 'heads', 'heads')

You can verify that the output is given by the following, which is divided into a table and a line following the table.

```
           0       1       2       3       4
Biased: 0.12500 0.08437 0.05695 0.03844 0.02594
Fair: 0.25000 0.11250 0.05062 0.02278 0.01025

(0.025949267578125004, ['Biased', 'Biased', 'Biased', 'Biased', 'Biased'])
```

What does the table signify?

It is the results obtained after running the Viterby algorithm. The number of columns is equal to the number of observations

The table is the dynamic programming table generated from running the Viterbi algorithm.

There are as many columns as there are observations. The values in each column are generated from left to right. In order to obtain each value, the largest value in the previous column is multiplied by the transition probability of ending up in the label of the row you are calculating (in this case, Biased or Fair) and then is multiplied by the emission probability of the observed symbol of the column (in this case, heads or tails). The largest value in the rightmost column is the most likely outcome.

- ○ 0 pts
  The learner did not answer.
- ○ 1 pt
  The learner gave a partial description (e.g. only number of observations).
- ○ 2 pts
  The learner gave a complete description.

Returning to the output of Viterbi.py, we will now examine the last line of output.

```
           0       1       2       3       4
Biased: 0.12500 0.08437 0.05695 0.03844 0.02594
Fair: 0.25000 0.11250 0.05062 0.02278 0.01025

(0.025949267578125004, ['Biased', 'Biased', 'Biased', 'Biased', 'Biased'])
```

Interpret the two items in the final line of output.

the first figure in the last row is the probability of the most likely sequences of states resulted in the emmited string. The second item is the most likely sequence of states occuring for the given symbols when using the biased coin

(1 point) The first component represents the probability (0.025949...) of the most likely sequence of states having generated the given string of emitted symbols.

(1 point) The second component represents the most likely sequence of states for the emitted symbols ("THHHH"), which occurs when the biased coin was used each time.

- ○ 0 pts
  (See above grading scheme.)
- ○ 1 pt
  (See above grading scheme.)
- ○ 2 pts
  (See above grading scheme.)

We now will change the second emitted symbol so that the emitted string is now "TTHHH" instead of "THHHH". To do so, change the line

```
observations = ('tails', 'heads', 'heads', 'heads', 'heads')
```

to the line

```
observations = ('tails', 'tails', 'heads', 'heads', 'heads')
```

Then run Viterbi.py again.

How did this change in the emitted symbols affect the probability of the most likely sequence of hidden states? How did it change the most likely sequence of hidden states? Interpret this result.

It decreased the probability of the most likely sequence of hidden states and changed the most likely sequence of hidden states from all "biased" to all "fair". The interpretation is that the probablity that the dealer used the fair coin is higher than using the biased coin and that applies for all observations

(1 point) The probability of the most likely sequence of hidden states has decreased from 0.025949... to 0.102515...

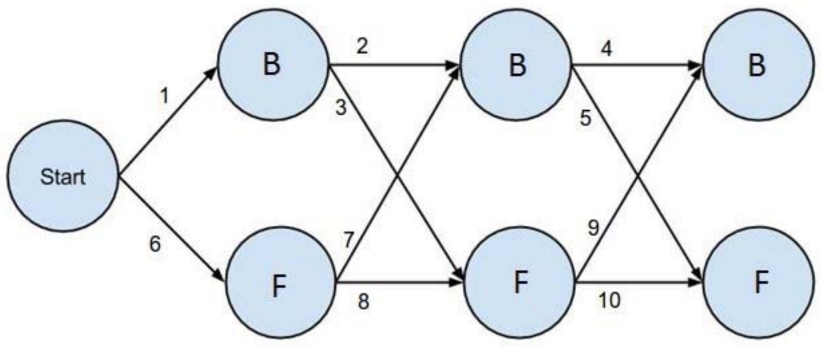
(1 point) The most likely sequence of states changed from ['Biased', 'Biased', 'Biased', 'Biased', 'Biased'] to ['Fair', 'Fair', 'Fair', 'Fair', 'Fair'].

(1 point) If we observe four heads in five flips, it is more likely that the coin is in the biased state than the fair state. However, this is not true if we observe three heads in five coin flips.

- 0 pts
  (See above grading scheme.)
- 1 pt
  (See above grading scheme.)
- 2 pts
  (See above grading scheme.)
- 3 pts
  (See above grading scheme.)

Help Center

The figure below is the Viterbi graph for the crooked casino HMM emitting HTH.



Compute the weights of enumerated edges in this graph (10 edges total).

1=0.375
2=0.225
3=0.05
4=0.675
5=0.05
6=0.25
7=0.025
9=0.45
9=0.075
10=0.45

1: 0.375

2: 0.225

3: 0.05

4: 0.675

5: 0.05

6: 0.25

7: 0.025

8: 0.45

9: 0.075

10: 0.45

○ 0 pts
The learner computed four or more values incorrectly.

○ 1 pt
The learner computed all but one to three values correctly.

○ 2 pts
The learner computed all weghts correctly.

Calculate (by hand) the dynamic programming table in the Viterbi algorithm for the emitted sequence HTH. Fill in the following table:

	0	**1**	**2**
Biased:			
Fair:			

```
        0      1          2
Biased:0,3750,084370,05695
Fair:     0,25  0,1125  0,05062
```

	0	**1**	**2**
Biased:	0.37500	0.08437	0.05695
Fair:	0.25000	0.11250	0.05062

○ 0 pts
The learner did not fill in the table correctly.

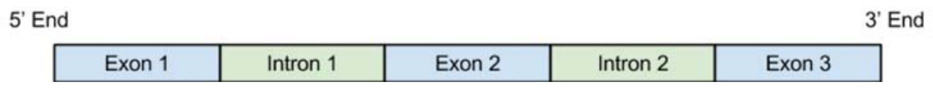
○ 1 pt
The learner filled in all but one or two values correctly.

○ 2 pts
The learner filled in the table correctly.

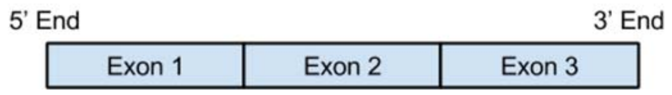
Finding Donor and Acceptor Splicing Sites

Various bioinformatics tools (such as GeneMark) use HMMs for gene prediction. Below, we will design a simple HMM aimed at finding exons in a nucleotide sequence.

In a **split gene**, before *messenger RNA* is translated into protein, it undergoes *splicing* to remove some parts of the RNA. The removed intervals are called **introns**, and the remaining intervals (which code for a protein in a protein-coding gene) are called **exons**. Before splicing, a strand of RNA may look like this:



And after splicing it will look like this:



The **donor site** is the region that separates the end of an exon from the beginning of an intron. The **acceptor site** is the region that separates the end of an intron from the beginning of an exon.

The most conservative part of the donor site is the first two nucleotides of introns (usually GU). The most conservative part of the acceptor site is the last two nucleotides of introns (usually AG). We will refer to the first two nucleotides in introns as Donor1 and Donor2 and the last two nucleotides in introns Acceptor1 and Acceptor2. For example, for canonical donor and acceptor sites GU and AG, Donor1 = G, Donor2 = U, Acceptor1 = A and Acceptor2 = G.

Our goal is to determine whether the genomic region contains exons, and if so, to find them based on a simple (and not very practical) analysis of donor and acceptor sites. Suppose we have done prior analysis of known exons and introns and come up with the following HMM based on the statistical analysis of annotated genes:

1. **∑:** {A, C, G, T}
2. *States:* Exon, Donor1, Donor2, Intron, Acceptor1, Acceptor2
3. *Transition* matrix:

	Transition To:**Exon**	Transition To:**Donor1**	Transition To:**Donor2**	Transition To: **Intron**	Transition To: **Acceptor1**	Transition To: **Acceptor2**
Exon	0.9	0.1	0.0	0.0	0.0	0.0
Donor1	0.0	0.0	1.0	0.0	0.0	0.0
Donor2	0.0	0.0	0.0	1.0	0.0	0.0
Intron	0.0	0.0	0.0	0.9	0.1	0.0
Acceptor1	0.0	0.0	0.0	0.0	0.0	1.0
Acceptor2	1.0	0.0	0.0	0.0	0.0	0.0

4. *Emission* matrix:

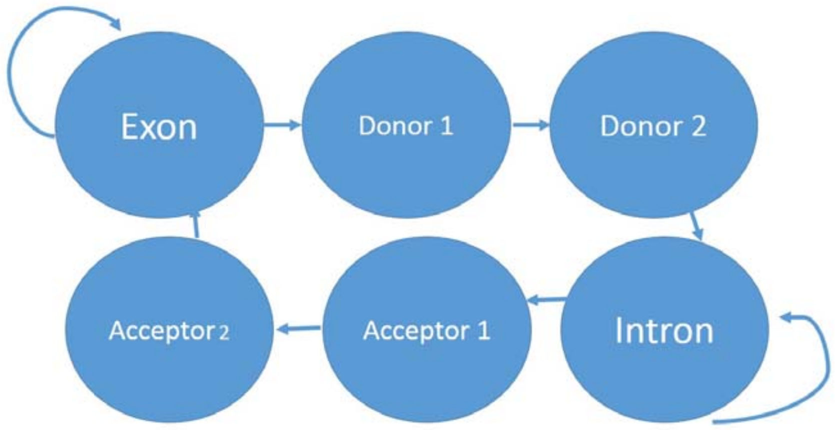
	Emit: **A**	Emit: **C**	Emit: **G**	Emit: **T**
Exon	0.25	0.25	0.25	0.25
Donor1	0.05	0.00	0.95	0.00
Donor2	0.00	0.05	0.00	0.95
Intron	0.40	0.10	0.10	0.40
Acceptor1	0.95	0.00	0.05	0.00
Acceptor2	0.05	0.00	0.95	0.00

This HMM starts in either the exon or intron state with the same probability. This means that the HMM's starting probabilities are as follows:

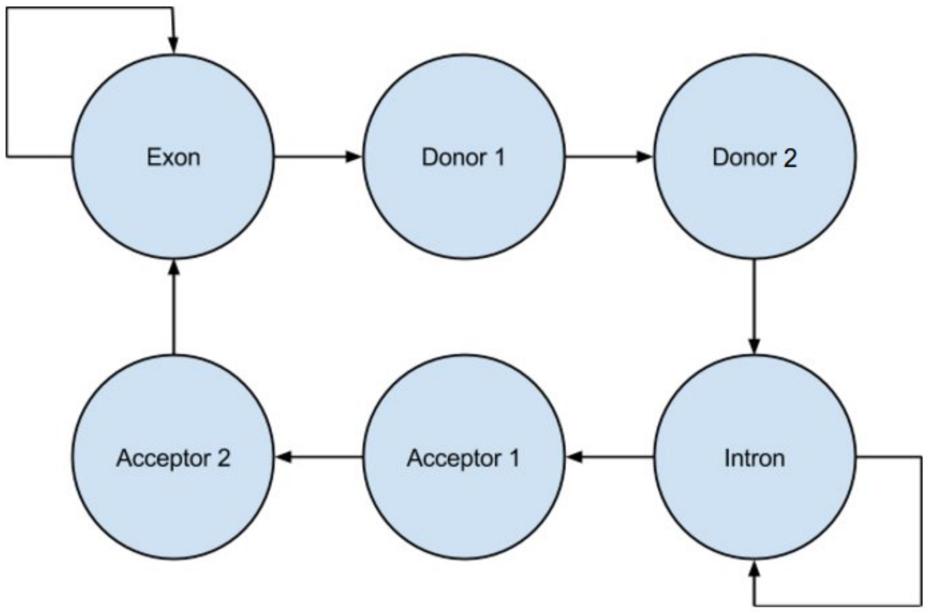
Exon	**Donor1**	**Donor2**	**Intron**	**Acceptor1**	**Acceptor2**
0.5	0.0	0.0	0.5	0.0	0.0

Draw the diagram of this HMM (only existing nodes and edges, without numbers).

Help Center



The diagram should resemble the following:



- 0 pts
  Three or more edges are drawn incorrectly.
- 1 pt
  One or two edges are drawn incorrectly.
- 2 pts
  The learner's diagram is correct.

We now have a different set of states, observations, transition and emission probability matrices, but the Viterbi algorithm is the same. As a result, we only need to change the input variables located at the top of Viterbi.py. You can copy the lines from this file to replace the preamble with (or, if you are interested in programming in Python, try to figure out how to change these lines yourself).

The DNA string that we will consider as the emitted symbols of the HMM is found here.

How many exons and introns do you find in this sequence?

exons 4, introns 3

4 Exons, 3 Introns

○ 0 pts
Answer is incorrect

○ 2 pts
Answer is correct

Let's imagine that our colleague conducted an experiment and showed that our test sequence should have many more introns than we found.

Given this piece of information, how would you change the transition matrix to create an HMM that more accurately finds exons and introns? Be specific about which transitions you would change and why.

In order for introns to occur more often we should increas the transition probability from exon to donor 1 and from intron to acceptor 1 since all the other transitions cannot be changed

Because we think that we have missed introns, we need to modify the probabilities in our model so that introns occur more often.We know that we always have a donor site at the beginning of an intron (i.e., the transitions *donor1 -> donor2 -> intron* always have probability 1), and we know that we always have an acceptor site at the end of intron (i.e. the transitions *acceptor1 -> acceptor2 -> exon* also have probability 1), so we can't change these. However, to raise sensitivity to introns, we can increase the transition probability from exon -> donor1 and from intron -> acceptor1.

○ 0 pts
The learner did not answer about increasing transition probabilities

○ 1 pt
The learner answered about increasing transition probabilities but did not provide reasonable justification.

○ 2 pts
The learner answered about increasing transition probabilities and provided reasonable justification.

Help Center

Finding CpG Islands

DNA methylation typically adds a methyl (CH3) group to the cytosine nucleotide within a CG dinucleotide, and methylated C spontaneously deaminate into T over time. As a result, CG is the least frequent dinucleotide in many genomes.

However, methylation is often suppressed near **promoter regions**, where gene transcription is initiated. As a result, these regions form **CG-islands**, where CG appears relatively frequently.

The following **dinucleotide frequency** matrices are obtained based on the statistics of annotated genomic sequences. (It is worth keeping in mind that obtaining these probabilities presents a catch-22; we need the transition probabilities to find the CG-islands, but we must first find the CG-islands in order to determine these frequencies.) Your task is to design an HMM based on these matrices that will help you find the CG-islands.

	A	**C**	**G**	**T**
A	0.053	0.079	0.127	0.036
C	0.037	0.058	**0.058**	0.041
G	0.035	0.075	0.081	0.026
T	0.024	0.105	0.115	0.050

-	**A**	**C**	**G**	**T**
A	0.087	0.058	0.084	0.061
C	0.067	0.063	**0.017**	0.063
G	0.053	0.053	0.063	0.042
T	0.051	0.070	0.084	0.084

Figure: Dinucleotide frequencies for a collection of CG-islands (top) and non-CG-islands (bottom) in the human genome computed for a single strand of the X chromosome. Frequencies of CG are shown in bold.

Your colleague suggests building a simple HMM that consists of just 2 states "+" (CG-island) and "–" (non-CG-island), one emitting symbols in CG-islands, and the other emitting symbols in non-CG-islands. The figure below shows a sequence of emitted symbols (top) and a possible sequence of hidden states (bottom) for this HMM:

```
C A C C T T T A T C G C C C T A A T C G C T A G C C G G C
- - - - - - - - + + + + + + + - - + + + + + + + + + + + +

```

Can you utilize the matrices above to derive emission and transition probabilities for the proposed two-state HMM? Why?

no

The tables tell us about the dinucleotide frequencies inside and outside of CG-islands, but they don't tell us anything about the frequencies of individual nucleotides, which will make defining an emission matrix difficult.

Furthermore, the tables only provide us with information about the likely of dinucleotides either within a CG-island or outside of a CG-island and do not help us define transition probabilities for the likelihood of changing from a CG-island to a non-CG-island or vice-versa.

○ 0 pts
The learner answered incorrectly

○ 1 pt
The learner answered correctly, but did not provided reasonable justification.

○ 2 pts
The learner answered correctly and provided reasonable justification.

After some thought, we decide to use an 8-state HMM rather than a 2-state HMM. The hidden path for this HMM is shown below.

C- A- C- C- T- T- T- A- T+ C+ G+ C+ C+ C+ T+ A- A- T+ C+ G+ C+ T+ A+ G+ C+ C+ G+ G+ C+

The complete description of the 8-state HMM for finding CG-islands is given as follows:

1. ∑ = {A, C, G, T}
2. *States* = { A+, C+, G+, T+, A-, C-, G-, T-} We partition the set of states into "+" and "-" groups and assume that the starting probabilities across all states are equal to each other.
3. An 8x8 *Transition* matrix. There are two types of transition probabilities. There are transition probabilities *within* the same group (between states with the same sign, e.g., from A+ to C+ or A- to C-) and transition probabilities *between* groups (between states different signs, e.g., from A- to C+). The transition probabilities within groups were calculated from the tables above.

+	**A**	**C**	**G**	**T**
A	0.180	0.268	0.430	0.122
C	0.191	0.299	0.299	0.211
G	0.161	0.346	0.373	0.120
T	0.082	0.357	0.391	0.170

-	**A**	**C**	**G**	**T**
A	0.300	0.200	0.290	0.210
C	0.319	0.300	0.081	0.300
G	0.251	0.251	0.299	0.199
T	0.176	0.242	0.291	0.291

Explain how we transformed the dinucleotide frequency matrices given previously into the transition probability matrices shown above. (Hint: what is the sum of all transition probabilities for a single state in a single group?)

the sum of values of teh matrices is one. TO convert the dinucleotide matrices to transition matrices we must devide each entry by the sum of all probabilities in the row

In the dinucleotide matrices, the sum of the values in each matrix is 1.

However, with transition states, each row should sum to 1, since each row represents the probability of transitioning from the symbol represented by that row to each of A, C, G, and T.

Therefore, to convert the dinucleotide matrices shown previously to the transition matrices given now, we can simply normalize each row by dividing each entry by the sum of all probabilities in the row.

○ 0 pts
The learner answered incorrectly and provided a limited answer.

○ 1 pt
The learner answered correctly, but did not provided reasonable justification, or attempted a well-justified response that was incorrect.

○ 2 pts
The learner answered correctly and provided reasonable justification.

As for transition probabilities, we will assume that the probability of transitioning between + and - groups is be the same for each nucleotide, as is described in the following tables:

+ to -	**A-**	**C-**	**G-**	**T-**
A+	0.25	0.25	0.25	0.25
C+	0.25	0.25	0.25	0.25
G+	0.25	0.25	0.25	0.25
T+	0.25	0.25	0.25	0.25

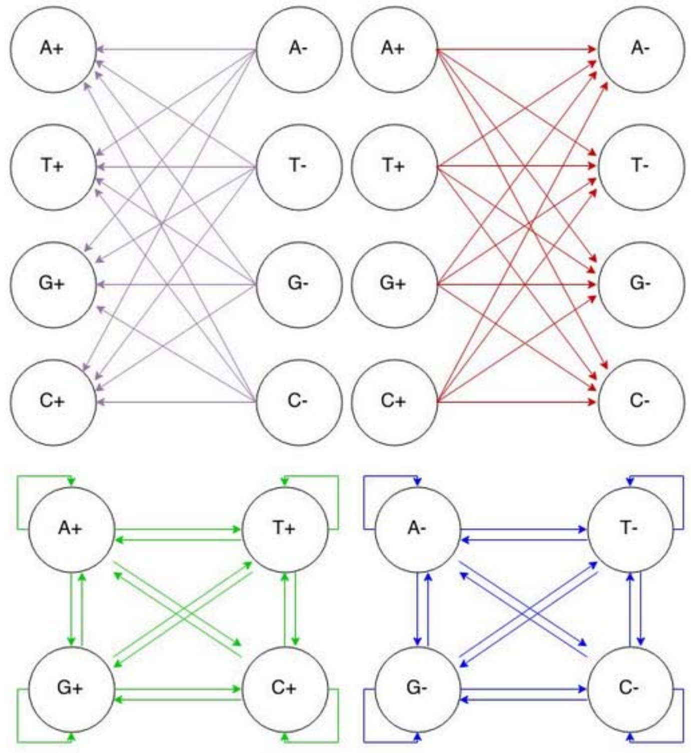
- to +	**A+**	**C+**	**G+**	**T+**
A-	0.25	0.25	0.25	0.25
C-	0.25	0.25	0.25	0.25
G-	0.25	0.25	0.25	0.25
T-	0.25	0.25	0.25	0.25

Help Center

The 4x4 matrices that we already considered regarding transitions within CG-islands (and within non-CG-islands) are used to construct an 8x8 *Transition* matrix (below) using a parameter *p*. This parameter is used to ensure that the sum of each row is 1 (without it, the sum of each row would be equal to 2). By varying the value of p, we weight the preference of group transitions within a goup (+ to + **or** - to -) versus transitions that change group (+ to - **or** - to +).

	A+	**C+**	**G+**	**T+**	**A-**	**C-**	**G-**	**T-**
A+	0.180 * (1-p)	0.268 * (1-p)	0.430 * (1-p)	0.122 * (1-p)	0.25 * p	0.25 * p	0.25 * p	0.25 * p
C+	0.191 * (1-p)	0.299 * (1-p)	0.299 * (1-p)	0.211 * (1-p)	0.25 * p	0.25 * p	0.25 * p	0.25 * p
G+	0.161 * (1-p)	0.346 * (1-p)	0.373 * (1-p)	0.120 * (1-p)	0.25 * p	0.25 * p	0.25 * p	0.25 * p
T+	0.082 * (1-p)	0.357 * (1-p)	0.391 * (1-p)	0.170 * (1-p)	0.25 * p	0.25 * p	0.25 * p	0.25 * p
A-	0.25 * p	0.25 * p	0.25 * p	0.25 * p	0.300 * (1-p)	0.200 * (1-p)	0.290 * (1-p)	0.210 * (1-p)
C-	0.25 * p	0.25 * p	0.25 * p	0.25 * p	0.319 * (1-p)	0.300 * (1-p)	0.081 * (1-p)	0.300 * (1-p)
G-	0.25 * p	0.25 * p	0.25 * p	0.25 * p	0.251 * (1-p)	0.251 * (1-p)	0.299 * (1-p)	0.199 * (1-p)
T-	0.25 * p	0.25 * p	0.25 * p	0.25 * p	0.176 * (1-p)	0.242 * (1-p)	0.291 * (1-p)	0.291 * (1-p)

4. *Emission.* This is an unusual HMM in which all emission probabilities are either 1 or 0. Each state emits its “own” nucleotide with probability 1, i.e., G+ always emits G, T- always emits T, etc.The HMM diagram for this HMM is the complete directed graph on 8 nodes, i.e., each node is connected to each other node by an edge (yielding 64 edges overall). The figure below illustrates 16 of these edges corresponding to transitions from “+” to “-“ states (left) and 16 edges corresponding to transitions from “-” to “+“ states (right). The transition probability for each of these 16+16=32 edges is 0.25* *p*, and our goal is to choose the parameter *p* that is the most biologically sensible.



To simulate this HMM, we will once again use Viterbi.py. As before, we can use Viterbi.py by replacing the preamble with the content of this file.

We will assume that the HMM emits a nucleotide string chr_22.txt (which contains the first 1,000 base pairs of human chromosome 22). You need to download this file and save it in the same directory as your (modified) Viterby.py. This will ensure that when you run Viterbi.py, it is able to read the contents of chr_22.txt.

Run your Python code on the emitted string to get the sequence of hidden states. Do you notice anything troubling about the output?

most of the probabilities are zero

The probability of the most likely path is 0. Additionally, many entries in the output table are 0.

- 0 pts
  Answer is incorrect
- 1 pt
  Answer is correct

The problem you are likely to encounter when running the previous Python segment is called **underflow**, which occurs when a variable becomes too small for the computer to store in memory (which caused the computer to round it down to zero).

To address the issue of underflow, we will use the Python code Islands.py. Note that all numeric variables were converted (the technical term is "type casted") to decimal, which tells the computer to allocate more memory for a number. Using the decimal type effectively tells Python to store more places after the decimal, since the probabilities (scores of nodes) in the Viterbi graph for long sequences can become very small.

In Islands.py, you may change the variable *p*, which controls how much weight is given to the transition probabilities within a group versus the transition probabilities between groups. Experiment with different values of *p*.

How many islands do you find with *p* = 0.2? What about *p* = 0.4? Can you guess the number of islands for *p* = 0.9 without running a program and explain why?

p=0.2: 4
p=0.4: 36
p=0.9: 500, The weight of transition probability is so high that the output contains all dinucleotides of the sequence which is 500 of 1000 bases

(1 point) Identifying a value of p equal to 0.2 to produce about 4 CG-islands.

(1 point) Identifying a value of p equal to 0.4 to produce about 36 CG-islands.

(1 point) The number of islands found changes rapidly with respect to the value of p, and for a large value of p, the HMM will change state in every turn. Thus, for 1000 nucleotides with p=0.9, we predict 400 - 500 islands.

- 0 pts
  (See above grading scheme.)
- 1 pt
  (See above grading scheme.)
- 2 pts
  (See above grading scheme.)
- 3 pts
  (See above grading scheme.)

Classifying Worm Proteins

We mentioned that there are over 7,500 known orthologs between *C. elegans* and humans, but biologists are constantly searching for more elusive similarities between other *C. elegans* and human genes. Several large gene families in *C. elegans*, which were first thought to be nematode-specific, have since been classified as putative **G-protein coupled receptors** (**GPCRs**), These proteins, which are likely to have orthologs in other species, are receptors weaved into the cell membrane that sense extracellular stimuli and activate inside signal transduction pathways that trigger cellular responses. The human genome encodes at least 350 GPCRs, and the function of most of these proteins remains unknown.

Because protein function largely depends on protein structure (which is often difficult to infer from amino acid sequence alone), detecting function similarity between these nematode sequences and known GPCRs in other organisms is a nontrivial task. However, if we can find human proteins that are similar in function to these nematode sequences, then we will be able to study the nematode counterpart in a laboratory and extrapolate the experimental results to humans (with some further experiments, of course).

Here we choose the putative GPCR gene **sra-4** in *C. elegans* (**Serpentine Receptor A**, Wormpep AH6.8; UniProt Q09206; 329 aa) as an example. Your goal is to find a protein of known function in humans, which may serve as a possible ortholog of sra-4.

Before we can analyze sra-4, we must first retrieve its sequence. UniProt, the “Universal Protein Resource”, is a comprehensive resource for protein sequence and annotation data. From UniProt, we can obtain the sequence of a protein as well as its annotation.

Go to UniProt and search for the protein with ID Q09206. Read the "Function" section - it contains terms from the Gene Onthology (GO) database, applied to the protein. Check if this protein is a GPCR, and determine the processes in which it is involved.

Then, download the protein in FASTA format.

What is the length of this protein (in amino acids)?

329

329

- 0 pts
  Answer is incorrect
- 1 pt
  Answer is correct

Initial BLAST search of sra-4 protein.

“BLAST,” the “Basic Local Alignment Search Tool,” is a tool for finding similarities between a query sequence and sequences in a protein database. It is effectively the “Google search” of biological sequences: much like how we can “Google” a key term of interest, we can “BLAST” a biological sequence of interest to find similar sequences. BLAST performs a local alignment of the query sequences against sequences in the target database and returns the results in the order of ascending **E-values**. The “E-value” (or “Expected value”) is the number of hits with the same or higher score one would "expect" to see by chance when searching a randomly generated (decoy) database of the same size. Therefore, a smaller E-value corresponds to a more statistically significant match.

To quickly find any proteins similar to sra-4, you can run a BLASTP search (where the “P” stands for “Protein”) against NCBI BLAST’s “non-redundant (nr) protein sequences” database using the NCBI BLAST server.

Considering the first 10 matches, how many hits do you find with an E-value equal to zero? How many among the first ten matches are *C.elegans* genes? Did you find any non-worm proteins (other than the *Caenorhabditis* or *Ancylostoma* genus) among the first 50 hits?

there are 2 hits with E-values 0. Seven matches C. elegans genes in the 10 first hits. Of these 7, three are indicated as chemosensory receptors without mention about the source organism which is showin in the corresponding fasta file as C. elegans. There is no non-worm proteins among the first 50 hits

(1 point) Two of the first ten hits have zero E-value

(1 point) Seven of the first ten hits belong to *C.elegans*

(1 point) There are no non-worm proteins among the first 50 hits.

Note: There are several proteins described as "chemosensory receptor", but with closer examination (e.g. clicking on the protein ID) it turns out that this is also a worm protein. If the learner reports the chemosensory receptor as a possible non-worm protein, please assign full credit.

○ 0 pts
(See above grading scheme)

○ 1 pt
(See above grading scheme)

○ 2 pts
(See above grading scheme)

○ 3 pts
(See above grading scheme)

Trying to find human orthologs directly

In the previous exercise, we identified a large number of proteins similar to sra-4, and almost all of the top hits belonged to the same species or to close relatives. This is a fairly natural result, but our goal is to find human proteins. Recall that BLAST's "non-redundant (nr) protein database" contains proteins from various species, including humans, and we can limit our search within certain taxonomic groups. To restrict search, run BLAST for sra-4 again, filling in the optional "Organism" field with "Homo sapiens".

15. How many significantly similar sequences (e-value < 0.01) do you find?

Help Center

none

(1 point) No significant similarities were found.

Note: there are several hits of length 20-30 nucleotides which were found purely by chance.

○ 0 pts
Answer is incorrect

○ 1 pt
Answer is correct

Constructing a profile HMM from the multiple alignment and using it to search a database

We can see that we need a different approach to search for human orthologs of worm genes. As we already know, we can build a HMM to find more distant similarities by looking for specific signatures in proteins.

We will try to use **HMMER**, which is a toolkit that performs protein analysis using profile HMMs. We will use HMMER to construct a profile HMM of the multiple alignment and to search it against the protein database. HMMER contains the following search tools from which we must choose:

- phmmer: Searching a protein sequence against a protein database by building a profile HMM from your single query sequence and searching this profile HMM against a protein sequence database
- hmmscan: Searching a protein sequence against a database of profile HMMs
- hmmsearch: Searching a protein alignment or profile HMM against a protein sequence database. If a protein alignment is input, a profile HMM will be generated from the alignment that will be used to search against the protein sequence database.
- jackhammer: Iterative version of phmmer, but iterative: similar sequences detected in a previous round are incorporated into a new profile, and the new profile is searched again until there are no improvements (or little improvements) in score.

Because we have only one sra-4 sequence, we will use phmmer. Paste the downloaded sequence of sra-4 in FASTA format (or upload the file) in the main window. In the field "Sequence Database" select UniProtKB (Universal Protein Resource KnowledgeBase). Select the field "Restrict by taxonomy" and enter *Homo sapiens* here.

Do you find any significant hits?

No

No significant hits are identified.

○ 0 pts
Yes

○ 1 pt
No

Should we give up? Of course not! Just because the methods that we have used have not identified a similarity does not mean that one does not exist.

There is a large collection of protein families and corresponding multiple sequence alignments, called **Pfam** (**P**rotein **Fam**ilies), which we can check for similarity with sra-4.

Use hmmscan to search against the Pfam database of profile HMMs using the sra-4 sequence. Paste the sequence or upload the file in the main window, and leave all other parameters as their default values. Check for the significance of identified matches (be sure that e-value < 0.01). Select the most significant result (click on its Id), and see the protein description page.

What is the name of the corresponding family?

7TM GPCR
Nematode chemoreceptor

7TM_GPCR_Sra, or Serpentine type 7TM GPCR chemoreceptor Sra

- 0 pts
  The learner answered incorrectly.
- 1 pt
  The learner answered correctly.

Help Center

Now we that we have identified the most related family, we need to obtain some representatives of this family to align and use as a model for the subsequent steps.Fortunately, an alignment for each Pfam family has already been constructed. On the the same page, select "Alignment" (from the menu on the left side of the page), go to "Download options" on the bottom of page, and download the set of NCBI proteins in raw Stockholm format.

How many proteins are in this dataset?

497

497 or 508. Both answers are correct, depends on what version you use - raw of gzipped. Yes, we know it's weird.

- 0 pts
  Answer is incorrect
- 1 pt
  Answer is correct

Now that we have an alignment of sequences from a protein family, it is time to use **hmmsearch**. This tool takes a protein alignment as an input, and it generates a profile HMM from the alignment that it uses to search against a specified protein sequence database.

As with many bioinformatics tools, hmmsearch is very sensitive to the input file format. In the Pfam database, the alignment is stored along with additional information, so in order to avoid errors, open the downloaded file in any text editor and remove all strings before the second occurence of "# STOCKHOLM 1.0". The beginning of the new file should look like the following:

```
# STOCKHOLM 1.0

17535745/1-328 <
```

Open hmmsearch, paste or upload your edited alignment in the main window, and restrict your search by taxonomy with *Homo sapiens* and select UniProtKB as the database (as before).

Do you find any significant matches (e-value <0.01)? If yes, specify the name of the most significant gene.

Yes, the name of the most significant gene is CCR8, encoding the C-C chemokine receptor type 8 protein

The first two hits are isoforms of the same gene - CCR8_HUMAN, C-C chemokine receptor type 8.

- 0 pts
  The learner did not identify the CCR8_Human gene
- 2 pts
  The learner identified the CCR8_Human gene

Since we found the human gene that fits the HMM profile based on the sra-4 sequence, we can investigate its structure and function.Click on the protein name, which will open the corresponding page in the UniProt database.

Try to find additional information about this gene and try to answer the following:

- **where in the cell is it located, and how does its location relate to its structure and function?**
- **in which human organ can it be found predominantly?**
- **in which process in the organism is it involved?**

The protein is located in cell membrane. It is a transmembrane protein and its location and structure is such that it allos to span across the cell membrane and relay signals between the cell's internal and external environments.
It is mainly found in thymus.
It is involved in the immune system and specifically in inflammatory response. It regulates monocyte chemotaxis but also thymic cell line apoptosis

(1 point) CCR8 (chemokine (C-C motif) receptor 8) is a member of the beta chemokine receptor family. As a receptor, it is located in the cellular membrane, and it permits signal transduction from the outer cell surface inside. It contains seven transmembrane domains (which pass seven times through the cell membrane).

(1 point) It is located predominantly in the thymus.

(1 point) As well as other chemokines, CCR8 is important for the migration of various cells into inflammatory sites (chemotaxis) and immune response.

- 0 pts
  (See above grading scheme.)
- 1 pt
  (See above grading scheme.)
- 2 pts
  (See above grading scheme.)
- 3 pts
  (See above grading scheme.)

Help Center

Conclusion

The power of HMMs has allowed us to compare very distant organisms, which were separated so long ago that direct alignment fails to find any matches between them. From the set of aligned sequences we obtained a profile HMM, which contains the signatures of the protein of interest. We then got the opportunity to look for proteins with similar signatures in any organism, even very distant ones.

We have determined that sra-4, which is a member of the seven-transmembrane G-protein-coupled receptor class (7TM GPCRs), has a close relative in the human proteome. Sra-4 belongs to a family of chemoreceptors, which are a very important class of molecules for the worms. Chemoperception is one of the central "senses" of nematodes like *C. elegans,* which are otherwise blind and deaf. We have more advanced senses, but we still use almost the same structure to transduce chemical signals inside our bodies.

Note: no response needed.

(optional) Please provide any additional feedback that you would like to give here.

Submit Review

Comments
Visible to classmates

share your thoughts...

Help Center