CS 369 2020 Assignment 4

Due Wednesday June 10 10:00 pm

In the first part of this assignment, we use a Hidden Markov Model to model secondary structure in protein sequences and implement a couple of algorithms we saw in lectures.

In the second part, we simulate sequences down a tree according to the Jukes-Cantor model then use distance methods to try to reconstruct the tree.

Write your code in Python and present your code embedded in a report in a Jupyter Notebook. Make sure you test your code thoroughly and write clear, commented code that others can understand.

Submit two files to Canvas: the .ipynb and .html both showing code and results by 10pm on the due date.

There are 30 marks in total for this assessment.

1. *[14 marks total]* Suppose we wish to estimate basic secondary structure in protein (amino acid) sequences. The model we consider is a simplistic rendition of the model discussed in S C. Schmidler et al. (2004) Bayesian Segmentation of Protein Secondary Structure, doi:10.1089/10665270050081496

   We assume that at each point of the sequence, the residue is associated with one of three secondary structures: α-helix, β-strand and loops which we label H, S and T, respectively. To simplify the problem, we classify the amino acids as either hydrophobic, hydrophilic or neutral (B, I or N, respectively) so a sequence can be represented by this 3-letter alphabet.

   In a α-helix, the residues are 10% neutral, 30% hydrophobic and 60% hydrophilic. In a β-strand, they are 30%, 55%, 15% and in a loop they are 70%, 10%, 20%.

   Assume that all secondary structures have geometrically distributed length with α-helices having mean 15 residues, β-strands having a mean of 8 residues and loops a mean of 6 residues. A β-strand is followed by an α-helix 40% of the time and a loop 60% of the time. An α-helix is followed by a β-strand 30% of the time and a loop 70% of the time and a loop is equally likely to be followed by a strand or a helix. At the start of a sequence, any structure is equally likely.

   (a) *[3 marks]* Sketch a diagram of the HMM (a hand-drawn and scanned picture is fine). In your diagram, show only state nodes and transitions. Show the emission probabilities using a separate table.

   Derive the transition probabilities of a state to itself (e.g., a_{HH}) by considering that if L is geometrically distributed with parameter p then $E[L] = 1/p$. Make sure you use the correct parametrisation of the geometric distribution (noting that you can't have a secondary structure of length 0) and remember that $\sum_l a_{kl} = 1$ for any state k.

   (b) *[3 marks]* Write a method to simulate state and symbol sequences of arbitrary length from the HMM. Your method should take sequence length as an argument. Simulate and print out a state and symbol sequence of length 150.

(c) *[3 mark]* Write a method to calculate the natural logarithm of the joint probability $P(x, \pi)$. Your method should take x and π as arguments.

Use your method to calculate $P(x, \pi)$ for π and x given below and for the sequences you simulated in Q1b.

$\pi =$ `S,S,H,H,H,T,T,S,S,S,H,H,H,H,H,H,S,S,S,S,S,S`

$x =$ `B,I,N,B,N,I,N,B,N,I,N,B,I,N,B,I,I,N,B,B,N,B`

(d) *[5 marks]* Implement the forward algorithm for HMMs to calculate the natural logarithm of the probability $P(x)$. Your method should take x as an argument.

Use your method to calculate $\log(P(x))$ for π and x given above and for the sequences you simulated in Q1b.

How does $P(x)$ compare to $P(x, \pi)$ for the examples you calculated? Does this relationship hold in general? Explain your answer.

2. *[16 marks total]* In this question you will write a method that simulates random trees, simulates sequences using a mutation process on these trees, calculate a distance matrix from the simulated sequences and then, using existing code, reconstruct the tree from this distance matrix.

(a) *[5 marks]* Write a method that simulates trees according to the Yule model (described below) with takes as input the number of leaves, n, and the branching parameter, λ. Use the provided Python classes.

The Yule model is a branching process that suggests a method of constructing trees with n leaves. From each leaf, start a lineage going back in time. Each lineage coalesces with others at rate λ. When there k lineages, the total rate of coalescence in the tree is $k\lambda$. Thus, we can generate a Yule tree with n leaves as follows:

Set $k = n, t = 0$.

Make n leaf nodes with time t and labeled from 1 to n. This is the set of available nodes.

While $k > 1$, iterate:

Generate a time $t_k \sim \text{Exp}\,(k\lambda)$. Set $t = t + t_k$.

Make a new node, m, with height t and choose two nodes, i and j, uniformly at random from the set of available nodes. Make i and j the child nodes of m.

Add m to the set of available nodes and remove i and j from this set.

Set k = k-1.

Simulate 1000 trees with $\lambda = 0.5$ and $n = 10$ and check that the mean height of the trees (that is, the time of the root node) agrees with the theoretical mean of 3.86.

Use the provided `plot_tree` method to include a picture of a simulated tree with 10 leaves and $\lambda = 0.5$ in your report. To embed the plot in your report, include in the first cell of your notebook the command `%matplotlib inline`

(b) *[5 marks]* The Jukes-Cantor model of DNA sequence evolution is simple: each site mutates at rate μ and when a mutation occurs, a new base is chosen uniformly at random from the four possible bases, $\{A, C, G, T\}$. If we ignore mutations from base X to base X, the mutation rate is $\frac{3}{4}\mu$. All sites mutate independently of each other. A sequence that has evolved over time according to the Jukes-Cantor model has each base equally likely to occur at each site.

The method `mutate` is provided to simulate the mutation process.

Write a method to simulate sequences down a simulated tree according to the Jukes-Cantor model.

Your method should take a tree with n leaves, sequence length L, and a mutation rate μ. It should return either a matrix of sequences corresponding to nodes in the tree or the tree with sequences stored at the nodes.

Your method should generate a uniform random sequence of length L at the root node and recursively mutate it down the branches of the tree, using the node heights to calculate branch length.

In your report, include a simulated tree with $n = 10$ and $\lambda = 0.5$ and a set of sequences of length $L = 20$ and mutation parameter $\mu = 0.5$ simulated on that tree.

(c) *[3 marks]* Write a method to calculate the Jukes-Cantor distance matrix, d, from a set of sequences, where d_{ij} is the distance between the ith and the jth sequences. Recall that the Jukes-Cantor distance for sequences x and y is defined by

$$d_{xy} = -\frac{3}{4}\log\left(1 - \frac{4f_{xy}}{3}\right)$$

where f_{xy} is the fraction of differing sites between x and y. Since we will be dealing with short sequences, use the following definition of f_{xy} so that the distances are well-defined:

$$f_{xy} = \min\left(\frac{D_{xy}}{L}, 0.75 - \frac{1}{L}\right)$$

where D_{xy} is the number of differing sites between x and y and L is the length of x.

Include a simulated set of sequences of length $L = 20$ from the tree *leaves* and corresponding distance matrix in your report for a tree with $n = 10$, $\lambda = 0.5$ and mutation parameter $\mu = 0.5$.

(d) *[3 marks]* Now simulate a tree with $n = 10$ and $\lambda = 0.5$ and on that tree, simulate three sets of sequences with lengths $L = 20$, $L = 50$ and $L = 200$, respectively, with fixed $\mu = 0.1$. For each simulated set of sequences, calculate the distance matrix and print it out.

Then reconstruct the tree using the provided `compute_upgma_tree` method. Use the `plot_tree` method to include a plot of the original tree and a plot of the reconstructed tree for each distance matrix.

Comment on the quality of the reconstructions and the effect that increasing the sequence length has on the accuracy of the reconstruction.