CS 369 Assignment 5 2017

Due Tuesday June 6 9:00 am

Two notebooks, `HMM.ipynb` and `tree.ipynb`, with helpful code are on the course resources page `https://www.cs.auckland.ac.nz/courses/compsci369s1c/resources/`.

Write your submission in a Jupyter notebook and write any code in Python 3. You should submit the .ipynb file and a .html file with all output displayed. The primary document the markers will look at is the .html file.

In your report, include explanations of what you are doing and comments in the code. Where you are making mathematical derivations, show your working.

Submit your notebook as an .ipynb file and as an .html file with all output showing to `https://adb.auckland.ac.nz/` by 6:00 pm on the due date.

1. *[8 marks total]* In Assignment 4, we introduced a simple HMM to model secondary structure in proteins with states H, S, T and symbols B, I, N. Refer to that assignment for more detail about the model and its meaning. It has transition probabilities

$$\mathbf{a} = \begin{array}{cccc} & H & S & T \\ H & 0.95 & 0.01 & 0.04 \\ S & 0.0333 & 0.9167 & 0.05 \\ T & 0.05 & 0.05 & 0.90 \end{array}$$

and emission probabilites

$$\mathbf{e} = \begin{array}{cccc} & B & I & N \\ H & 0.35 & 0.55 & 0.10 \\ S & 0.55 & 0.15 & 0.30 \\ T & 0.10 & 0.10 & 0.80 \end{array}$$

.

(a) Use the supplied function `simulate_HMM` to simulate a state and symbol sequence pair (π, x) of length 100. Print π and x to output. Make sure that π has at least two runs of Hs (you can set the random seed using `numpy.random.seed()` until you are happy with your sequence). You will use this simulated pair for the remainder of this question.

(b) By modifying the supplied forward and backward algorithm functions to return matrices, calculate the matrix $\mathbf{P}$ where the (k, i)th entry of $\mathbf{P}$ is $\mathbf{P}(k, i) = \Pr(\pi_i = k|x)$, the posterior probability of being in state k at step i. Notice that $\mathbf{P}(k, i)$ is **not** a log probability. Show that your method is correct by checking the column sums of $\mathbf{P}$ are all 1.

(c) Make a plot of $\Pr(\pi_i = H|x)$ against the index i (that is, plot i on the horizontal axis and $\Pr(\pi_i = H|x)$ on the vertical axis). To embed the plot in your report, include in the first cell of your notebook the command `%matplotlib inline`

(d) By comparing your plot to the true state sequence π, discuss whether or not you think you could accurately find the α-helices in a given protein sequence using this model.

2. *[16 marks total]* In this question you will write a method that simulates random trees, extend the sequence simulator you wrote in Assignment 3 to work with these trees, calculate a distance matrix from the simulated sequences and then reconstruct the tree from this distance matrix.

   (a) 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 to represent a tree (see resources page).

   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 $t = 0$ 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 distinct 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.
   ```

   To check the correctness of your implementation, 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.

   (b) Write a method to simulate sequences down a simulated tree according to the Jukes-Cantor model (as described in Assignment 3). 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. Pseudocode methods for random sequence generation and mutation down a lineage are provided at the end of this problem.

   (c) 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 $\mu = 0.1$.

(d) 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 from the leaves, calculate the distance matrix and print it out.

Now for each distance matrix, 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 each reconstructed tree.

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

Pseudocode methods

The method randseq simulates a uniform random sequence. The method mutate mutates a given sequence according to the Jukes-Cantor model of mutation over a given length.

```
randseq(L)
  for (i in 1 to L)
    seq[i] = choice(['A','C','G','T'], [0.25,0.25,0.25,0.25])
  return seq
```

The mutate method below allows “mutations” from a base to itself. The uncorrected mutation rate (μ rather than $\frac{3}{4}\mu$) can therefore be used.

```
mutate(X,t,mu)
  L= X.length()
  \\ the number of mutations is Poisson with total rate L*mu*t
  numMutation = randpoiss(L*mu*t)
  \\ for each mutation, choose a site to mutate and mutate it
  for (i in 1 to numMutation)
    \\ choose a site
    site = ceiling(random()*L)
    \\ mutate that site
    X[site] =  choice(['A','C','G','T'], [0.25,0.25,0.25,0.25])
  return X
```