BIOSTAT615 Homework Assignment #4

Due : November 16th, 2017 by 8:30am (before class)

Send the following files as separare attachments (do not compress or bundle) within a single email to BIOSTAT.o5cnt679fbn3zzqu@u.box.com

- **[your_uniqname]HW4.tar.gz**
- **[your_uniqname]HW4.pdf**

Note that the email address above is DIFFERENT from the address used for Homework 0 to 3, and it will be different next time too.

Read carefully the problems below to understand how to prepare each file for submission.

Multiple submissions are allowed before the due, but only the last submission will be considered valid. Make sure that your submission was sent by checking the confirmation email. You must send the email from your [your_uniqname]@umich.edu email account.

Note that there no late submission will be allowed, unless late submission is pre-approved by documented reasons (with medical records, doctor's notes, or requests from advisor)

A Symmetric Continuous-time Markov Chain HMM (15 pts)

Implement a generic Hidden Markov Model algorithm for the following continuous-time Markov Chain hidden Markov models. (Note that you do not need to know what continuous-time Markov chain is to solve this problem)

- States: There are *n* possible states
- Initial distribution: $\pi_i = \Pr(q_t = S_i) = \frac{1}{n}$ (i.e. a uniform initial states)
- Observations: There are *m* observed data. Instead of specifying observed data, the users need to specify the *n x m* likelihood matrix of $\Pr(o_j|S_i)$ where $i \in \{1,2 \ldots, n\}$ represents row index and $j \in \{1,2 \ldots, m\}$ represents a column index.
- Timestamp: Each observed outcome is associated with a timestamp t_j
- Transition probability: Transition probability only depends on ***t*** and a transition parameter θ More specifically, the transition probability is defined as

$$\Pr\left(q_{j+1} = S_k \middle| q_j = S_i\right) = \begin{cases} \frac{1}{n}\left(1 - e^{-(t_{j+1}-t_j)\theta}\right) & k \neq i \\ 1 - \frac{n-1}{n}\left(1 - e^{-(t_{j+1}-t_j)\theta}\right) & k = i \end{cases}$$

First, you need to write `ctmcViterbi()` and `ctmcForwardBackward()` algorithms in Rcpp to perform Viterbi algorithm (to infer MLE path) and forward-backward algorithm (to calculate conditional probability of each observations) with the following criteria:

- The three input arguments are (1) `ts` – a `Rcpp::NumericVector` of size `m`, containing the timestamp of each observation (sorted in increasing order), (2) `theta` – a `double` type value, representing the parameter θ in the model of transition probability, (3) `obs` – a `n` by `m` `Rcpp::NumericMatrix` containing likelihood of observed data $\Pr(o_j|S_i)$ at `obs(i,j)`
- The requirement of Viterbi algorithm is to return a Viterbi path, and the forward-backward algorithm should return a *m x n* matrix of $\Pr(q_j = S_i|\boldsymbol{o})$, in a similar format of `obs`.
- The time complexity of your algorithms (both Viterbi and forward-backward) must be *O(nm)*.
- The algorithm should scale well to very many (possibly millions of) timestamps and (possibly thousands of) states.
- If you scale the algorithm while avoiding log calculation, you will receive an extra point.

Second, you need to document your method (preferably using `roxygen2`), and make a package that contains these two functions, and make a package named "`[youruniqname]_hw4`", and submit the R package file. You need to make a decent effort to make your documentation comprehensive.

In your PDF submission, you need to provide new equations describing how the forward and backward probabilities (and any additional intermediate values) are calculated. The document should also explain how the equations are implemented in your code, by matching the variables in the equations and the code. You need to clearly justify how the algorithm can be implemented in *O(nm).*

A skeleton Rcpp code is given below. Additional functions may be defined, if needed. Do not modify the existing code.

```
#include <Rcpp.h>
using namespace std;
using namespace Rcpp;

// Do not forget to add documentation for your package using roxygen2
// [[Rcpp::export]]
IntegerVector ctmcViterbi(NumericVector ts, double theta, NumericMatrix obs)
{
  int m = (int)ts.size();
  int n = (int)obs.nrow();
  if ( obs.ncol() != m )
    stop("The input matrix does not conform to the other parameters");

  IntegerVector viterbiPath(m);

  // TODO : Implement your function here



  return viterbiPath;
}

// Do not forget to add documentation for your package using roxygen2
// [[Rcpp::export]]
NumericMatrix ctmcForwardBackward(NumericVector ts, double theta,
NumericMatrix obs) {
  int m = (int)ts.size();
  int n = (int)obs.nrow();
  if ( obs.ncol() != m )
    stop("The input matrix does not conform to the other parameters");

  NumericMatrix condProb(n,m);
  // TODO : Implement your function here



  return condProb;
}
```

A running example is shown below. You may use **hmm615** package in the lecture material to further check the sanity of the code.

```
> obs <- matrix(c(0.88,0.10,0.88,0.10,0.02,0.30,0.02,0.30,0.10,0.60),2,5)
> obs
     [,1] [,2] [,3] [,4] [,5]
[1,] 0.88 0.88 0.02 0.02  0.1
[2,] 0.10 0.10 0.30 0.30  0.6

> theta <- log(2)

> ts <- c(1,2,3,4,5)

> ctmcViterbi(ts,theta,obs)
[1] 0 0 1 1 1

> ctmcForwardBackward(ts,theta,obs)
           [,1]      [,2]       [,3]       [,4]       [,5]
[1,] 0.93709508 0.8787157 0.06015362 0.01499567 0.05684089
[2,] 0.06290492 0.1212843 0.93984638 0.98500433 0.94315911
```

```
> ts <- c(1,2.95,3,4,5)

> ctmcViterbi(ts,theta,obs)
[1] 0 1 1 1 1

> ctmcForwardBackward(ts,theta,obs)
          [,1]      [,2]      [,3]       [,4]       [,5]
[1,] 0.8735049 0.3561437 0.2073607 0.02620133 0.05998634
[2,] 0.1264951 0.6438563 0.7926393 0.97379867 0.94001366
```