Chapter 5 - Solutions to exercises

Exercises: 1,4,6

Exercise 1

For $t = 2, 3, \ldots, T$, $i = 1, 2, \ldots, m$ we have the following recursion

$$\xi_{tj} = \{\max_i(\xi_{t-1,i}\,\gamma_{ij})\}p_j(x_t).$$

For $t = 1$:

$$\xi_{1i} = \Pr(C_1 = i, X_1 = x_1) = \delta_i p_i(x_1).$$

For $t = 2$:

$$\begin{aligned}
\xi_{2j} &= \max_i \Pr(C_1 = i, C_2 = j, X_1 = x_1, X_2 = x_2) \\
&= \max_i \Pr(C_2 = j, X_2 = x_2 | C_1 = i, X_1 = x_1)\Pr(C_1 = i, X_1 = x_1) \\
&= \max_i \Pr(C_2 = j, X_2 = x_2 | C_1 = i)\xi_{1i} \\
&= \max_i \Pr(X_2 = x_2 | C_2 = j, C_1 = i)\Pr(C_2 = j | C_1 = i)\xi_{1i} \\
&= \{\max_i \xi_{1i}\Pr(C_2 = j | C_1 = i)\}\Pr(X_2 = x_2 | C_2 = j) \\
&= \{\max_i(\xi_{1i}\,\gamma_{ij})\}p_j(x_2).
\end{aligned}$$

Following this principle this can be analogously extended to hold for all $t \in \{1, \ldots, T\}$.

Exercise 4

We have two Poisson-HMMs with equal probability transition matrix

$$\mathbf{\Gamma} = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{pmatrix},$$

and $\boldsymbol{\lambda}_1 = (10, 20, 30)$ and $\boldsymbol{\lambda}_2 = (15, 20, 25)$ respectively. Generate two sequences of length 1000 (with same seed) and estimate the underlying state sequence for both sequences with the Viterbi algorithm.

a)

Note that there is an error in pois.HMM.local_decoding in A2.txt in Zucchini09. For a function that works see the final page of the supplementary slides for chapter 5.

```
# Chapter 5, R-code for exercise 4, mwp 1/2-2011
source("A2.txt")
statdist <- function(gamma){
  m = dim(gamma)[1]
  matrix(1,1,m) %*% solve(diag(1,m) - gamma + matrix(1,m,m))
}

m = 3
gamma = rbind(c(0.8,0.1,0.1),c(0.1,0.8,0.1),c(0.1,0.1,0.8))
delta = statdist(gamma)
lambda1 = c(10,20,30)
lambda2 = c(15,20,25)
n = 1000

# Generate hidden state sequence
mvect <- 1:m
state <- numeric(n)
state[1] <- sample(mvect,1,prob=delta)
for (i in 2:n){
  state[i]<-sample(mvect,1,prob=gamma[state[i-1],])
}


r <- .Random.seed # Store the random seed

x1 <- rpois(n,lambda=lambda1[state]) # Generate data for lambda1

.Random.seed <- r # Restore the random seed to make data comparable

x2 <- rpois(n,lambda=lambda2[state]) # Generate data for lambda2

# Calculate the most probable state sequences for lambda1 and lambda2
global1 <- pois.HMM.viterbi(x1,m,lambda1,gamma,delta)
global2 <- pois.HMM.viterbi(x1,m,lambda2,gamma,delta)
```

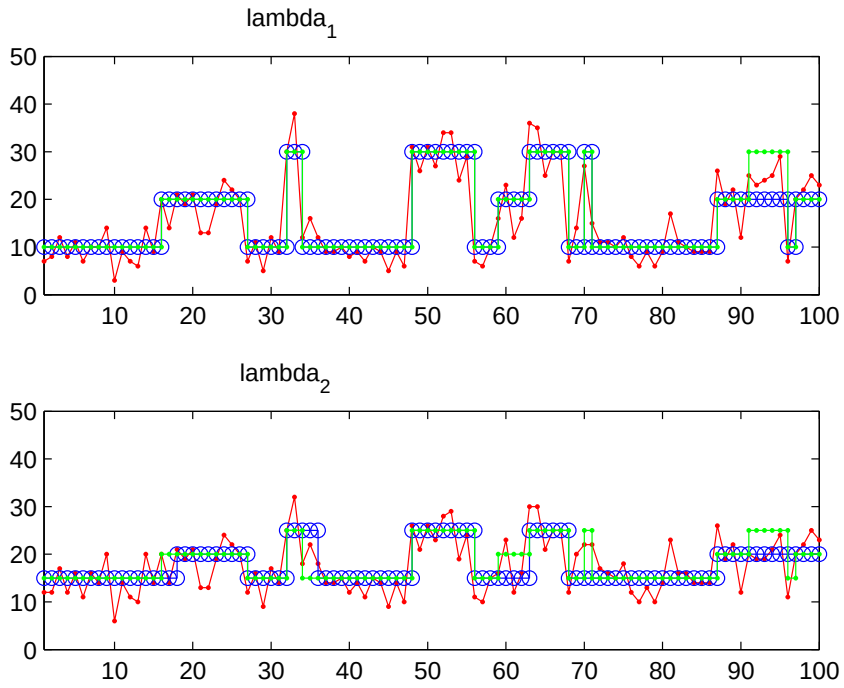
b)

Comparing the decoded states with the true states we observe the following number of wrongly classified states:

```
> n-sum(state == global1)
[1] 72
> n-sum(state == global2)
[1] 173
```

c)

From the above results we conclude that it is easier for the Viterbi algorithm to distinguish between states the more different the parameter values are in the states. In case 1 there is a larger difference between the λ's than in case 2, and therefore did we see fewer wrongly classified states in case 1. This is also intuitively clear from the below figure.



Exercise 6

We have that

$$\mathbf{\Gamma}^h(j,i) = \Pr(C_{t+h} = i | C_t = j),$$

and that

$$\alpha_T(j) = \Pr(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}, C_T = j).$$

Then

$$\begin{aligned}
\frac{\boldsymbol{\alpha}_T \mathbf{\Gamma}^h(,i)}{L_T} &= \frac{1}{L_T} \sum_j \alpha_T(j) \mathbf{\Gamma}^h(j,i) \\
&= \frac{1}{\Pr(\mathbf{X}^{(T)} = \mathbf{x}^{(T)})} \sum_j \Pr(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}, C_T = j) \Pr(C_{T+h} = i | C_T = j) \\
&= \frac{1}{\Pr(\mathbf{X}^{(T)} = \mathbf{x}^{(T)})} \sum_j \Pr(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}, C_T = j, C_{T+h} = i) \\
&= \frac{\Pr(\mathbf{X}^{(T)} = \mathbf{x}^{(T)}, C_{T+h} = i)}{\Pr(\mathbf{X}^{(T)} = \mathbf{x}^{(T)})} \\
&= \Pr(C_{T+h} = i | \mathbf{X}^{(T)} = \mathbf{x}^{(T)}).
\end{aligned}$$