Report_Lab2_Milda

Milda Poceviciute

27 September 2018

Question 1

```
# Create states
grid_states <- c(paste0("Sector ",1:10))
Readings <- grid_states

# Building states probabilities
Tprobs1 <- c(rep.int(0.5,2),rep.int(0,8))
Tprobs2 <- c(0,rep.int(0.5,2),rep.int(0,7))
Tprobs3 <- c(rep.int(0,2),rep.int(0.5,2),rep.int(0,6))
Tprobs4 <- c(rep.int(0,3),rep.int(0.5,2),rep.int(0,5))
Tprobs5 <- c(rep.int(0,4),rep.int(0.5,2),rep.int(0,4))
Tprobs6 <- c(rep.int(0,5),rep.int(0.5,2),rep.int(0,3))
Tprobs7 <- c(rep.int(0,6),rep.int(0.5,2),rep.int(0,2))
Tprobs8 <- c(rep.int(0,7),rep.int(0.5,2),rep.int(0,1))
Tprobs9 <- c(rep.int(0,8),rep.int(0.5,2))
Tprobs10 <- c(0.5,rep.int(0,8),0.5)

transProbs <- matrix(c(Tprobs1,Tprobs2,Tprobs3,Tprobs4,Tprobs5,Tprobs6,Tprobs7,Tprobs8,Tprobs9,Tprobs10)

# Building Observed readings probabilities
probs1 <- c(rep.int(0.2,3),rep.int(0,5),rep.int(0.2,2))
probs2 <- c(rep.int(0.2,4),rep.int(0,5),rep.int(0.2,1))
probs3 <- c(rep.int(0.2,5),rep.int(0,5))
probs4 <- c(0,rep.int(0.2,5),rep.int(0,4))
probs5 <- c(0,0,rep.int(0.2,5),rep.int(0,3))
probs6 <- c(rep.int(0,3),rep.int(0.2,5),rep.int(0,2))
probs7 <- c(rep.int(0,4),rep.int(0.2,5),rep.int(0,1))
probs8 <- c(rep.int(0,5),rep.int(0.2,5))
probs9 <- c(0.2,rep.int(0,5),rep.int(0.2,4))
probs10 <- c(rep.int(0.2,2),rep.int(0,5),rep.int(0.2,3))

emissProbs <- matrix(c(probs1,probs2,probs3,probs4,probs5,probs6,probs7,probs8,probs9,probs10),ncol=10,

# Initiate HMM

Robot_hmm <- initHMM(States=grid_states, Symbols=Readings, startProbs=rep.int(0.1,10), transProbs=transP
```

Question 2

```
Robot_path <- simHMM(Robot_hmm,100)
```

Question 3

$P(Z^t | X^{1:t})$

```
robot_probs <- function(hmm,xs){
    # Filtering
    forward_probs <- forward(hmm = hmm, observation = xs)

    # Smoothing - in wFALSEFALSEikipedia it says it should be forward-backward algorithm, but hmm packa
    # Instead it has backward algorithm


    backward_probs <- backward(hmm = hmm, observation = xs)

    # Normalising the probabilities
    e_forw <- exp(forward_probs)
    filter_prob <- prop.table(e_forw,2)
    # to prevent the problems in case all probabilities in a column is 0
    filter_prob[is.nan(filter_prob)] <- 0

    e_back <- exp(backward_probs)
    smooth_prob <- prop.table(e_forw*e_back,2)
    # to prevent the problems in case all probabilities in a column is 0
    smooth_prob[is.nan(smooth_prob)] <- 0

    return(list(filter_prob = filter_prob, smooth_prob = smooth_prob))
}

path_finder <- function(line){
    # select the path with the highest probability
    m <- max(line)
    list_m <- which(line == m)

    if (length(list_m) != 1){
        # if some hav ethe same highest prob, randomly select one of them
        index <- sample(list_m,1)
    } else {
        index <- list_m
    }
    return(grid_states[index])
}



result3 <- robot_probs(hmm=Robot_hmm,xs=Robot_path$observation)
```

Question 4

```
accuracy <- function(pathA, trueP){
    compare <- pathA==trueP
    t <- table(compare)
    accur <- t[2]/(sum(t))
    return(list(accuracy=accur,table=t))
```

```r
}

# Path calculated by hand based on the forward probabilities
filter_path <- apply(result3$filter_prob,2,path_finder)
# Path calculated by hand based on the backward probabilities
smooth_path <- apply(result3$smooth_prob,2,path_finder)
True_path <- Robot_path$states
# The most likely path calculated by the Viterbi Algorithm
viterbi_path <- viterbi(hmm = Robot_hmm, observation = Robot_path$observation)

# plotdf1 <- data.frame(time = 1:100,vpath=result3$viterbiP,real=True_path)
# library(ggplot2)
# ggplot(plotdf1, aes(x=time))+
#     geom_point(aes(y=vpath,colour="Viterbi"))+
#     geom_point(aes(y=real,colour="Actual"), size = 2)
#
# plot(x=plotdf1$time,y=plotdf1$vpath,type="l",col="red")
# points(x=plotdf1$time,y=plotdf1$vpath,col="red")
# lines(x=plotdf1$time,y=plotdf1$real,col="blue")
# points(x=plotdf1$time,y=plotdf1$real,col="blue")

result_f <- accuracy(filter_path,True_path)
result_v <- accuracy(viterbi_path,True_path)
result_s <- accuracy(smooth_path,True_path)
```

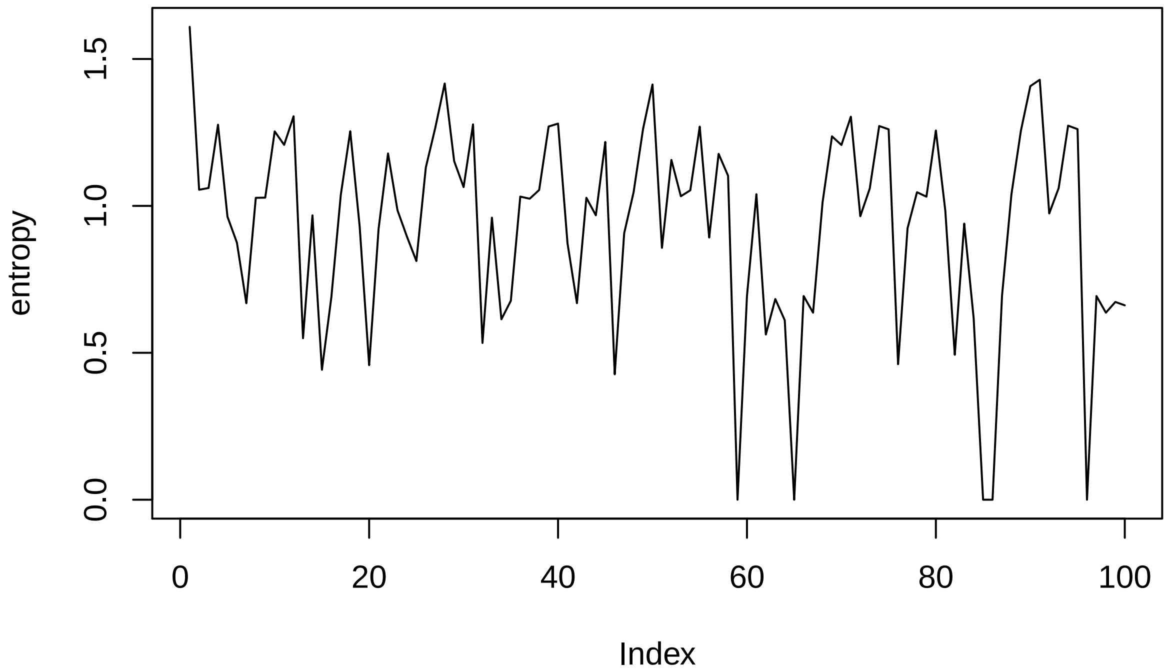
Question 5

```r
sim_probs <- function(hmm,sampleN){

    simulation <- simHMM(hmm=hmm,length=sampleN)
    result_probs <- robot_probs(hmm=hmm,xs=simulation$observation )

    # Path calculated by hand based on the forward probabilities
    filter_path2 <- apply(result_probs$filter_prob,2,path_finder)
    # Path calculated by hand based on the backward probabilities
    smooth_path2 <- apply(result_probs$smooth_prob,2,path_finder)
    True_path2 <- simulation$states
    # The most likely path calculated by the Viterbi Algorithm
    viterbi_path2 <- viterbi(hmm = hmm, observation = simulation$observation)

    result_f2 <- accuracy(filter_path2,True_path2)
    result_v2 <- accuracy(viterbi_path2,True_path2)
    result_s2 <- accuracy(smooth_path2,True_path2)

    return(list(result_f=result_f2,result_v=result_v2,result_s=result_s2))
}


question5 <- lapply(rep.int(100,5),sim_probs,hmm=Robot_hmm)
```

Question 6

The lower the Shannon entropy, the more information distribution contains.

```
entropy <- apply(result3$filter_prob,2,entropy.empirical)

plot(entropy, type="l")
```



Question 7

$P(Z^{t+1}|X^{1:t})$

```
Z100 <- matrix(rep(result3$filter_prob[,100],10),nrow=10,ncol=10)
Z_101 <- t(transProbs) %*% Z100[,1]
```