732A96 Advanced Machine Learning: Lab 2

Rebin Hosini, rebho 150

2017-09-13

Question 1

```
#Library
library(HMM)
library(knitr)

transmat <- matrix(0, ncol = 10, nrow = 10)
emissionmat <- matrix(0,ncol = 10, nrow =10)
emissionprob <- 1/5
transprob <- 1/2

for (i in seq_along(1:10)){
  for(j in seq_along(1:10)){
    transmat[i,j] <- ifelse(((i-j) <1 && (i-j) >= -1),transprob,
                            ifelse(j == 10 && i-j >= 9, transprob, 0))
    emissionmat[i,j] <- ifelse(abs(i-j)<=2 | abs(i-j)>=8,emissionprob,0)
  }
}

transmat[10,1] <- transprob

hmm <- initHMM(Symbols = 1:10, States = 1:10, transProbs = transmat,
               emissionProbs = emissionmat)
```

Question 2

```
set.seed(12345)
hmmsim <- simHMM(hmm, length = 100)
```

Question 3

```
prob_dist <- function(hSim = hmmsim){
  smoothing <- posterior(hmm, hSim$observation)
  filtering <- exp(forward(hmm, hSim$observation))
  #filtering <- apply(filtering, MARGIN = 2, exp)
  filtering <- prop.table(filtering, margin = 2)
  vit <- viterbi(hmm, hSim$observation)
  return(list("Viterbi" = vit,
              "Smoothing" = smoothing,
              "Filtering" = filtering))
}
```

Question 4

```
prob_percentage <- function(){
  set.seed(12345)
  #Initialize vectors
  accuracyFilter <- c()
  accuracySmoothing <- c()
  accuracyViterbi <- c()

  #Most probable
  probFilter <- apply(prob_dist()$Filtering, MARGIN = 2, which.max)
  probSmoothing <- apply(prob_dist()$Smoothing, MARGIN = 2, which.max)
  probViterbi <- prob_dist()$Viterbi

  for (i in 1:length(probFilter)){
    accuracyFilter[i] <- ifelse(hmmsim$states[i] == probFilter[i],1,0)
    accuracySmoothing[i] <- ifelse(hmmsim$states[i] == probSmoothing[i],1,0)
    accuracyViterbi[i] <- ifelse(hmmsim$states[i] == probViterbi[i],1,0)
  }

  fprop <- sum(accuracyFilter)/length(accuracyFilter)
  sprop <- sum(accuracySmoothing)/length(accuracySmoothing)
  vprop <- sum(accuracyViterbi)/length(accuracyViterbi)

  return(data.frame("Filter Percentage" =  fprop,
                    "Smoothing Percentage" = sprop,
                    "Viterbi Percentage" = vprop))
}

kable(prob_percentage(), caption = "Accuracy")
```

Table 1: Accuracy

Filter.Percentage	Smoothing.Percentage	Viterbi.Percentage
0.53	0.74	0.56

Question 5

```
iterations <- rep(100,5)
accuracyFun <- function(iter = iterations){

  #Initialize vectors
  accuracyFilter <- c()
  accuracySmoothing <- c()
  accuracyViterbi <- c()
  accuracyList <- list()
  count <- 1

  for(i in iter){
    simulatedHmm <- simHMM(hmm, length = i)
    probFilter <- apply(prob_dist(hSim = simulatedHmm)$Filtering,
                        MARGIN = 2, which.max)
    probSmoothing <- apply(prob_dist(hSim = simulatedHmm)$Smoothing,
                           MARGIN = 2, which.max)
    probViterbi <- prob_dist(hSim = simulatedHmm)$Viterbi

   for (j in 1:length(probFilter)){
     accuracyFilter[j] <- ifelse(simulatedHmm$states[j] == probFilter[j],1,0)
     accuracySmoothing[j] <- ifelse(simulatedHmm$states[j] == probSmoothing[j],1,0)
     accuracyViterbi[j] <- ifelse(simulatedHmm$states[j] == probViterbi[j],1,0)
   }
   fprop <- sum(accuracyFilter)/length(accuracyFilter)
   sprop <- sum(accuracySmoothing)/length(accuracySmoothing)
   vprop <- sum(accuracyViterbi)/length(accuracyViterbi)

   accuracyList[[count]] <- data.frame("Filter Percentage" =  fprop,
                                       "Smoothing Percentage" = sprop,
                                       "Viterbi Percentage" = vprop)
   count <- count + 1
  }
  return(accuracyList)
}

aclist <- accuracyFun()
accuracydf <- data.frame(matrix(unlist(aclist), nrow=length(iterations), byrow=T))
colnames(accuracydf) <- c("Filter", "Smoothing", "Viterbi")
#rownames(accuracydf) <- c(iterations)

kable(accuracydf, caption = "Accuracy")
```

Table 2: Accuracy

Filter	Smoothing	Viterbi
0.53	0.74	0.56
0.46	0.68	0.61
0.49	0.77	0.65
0.49	0.58	0.56
0.60	0.80	0.65

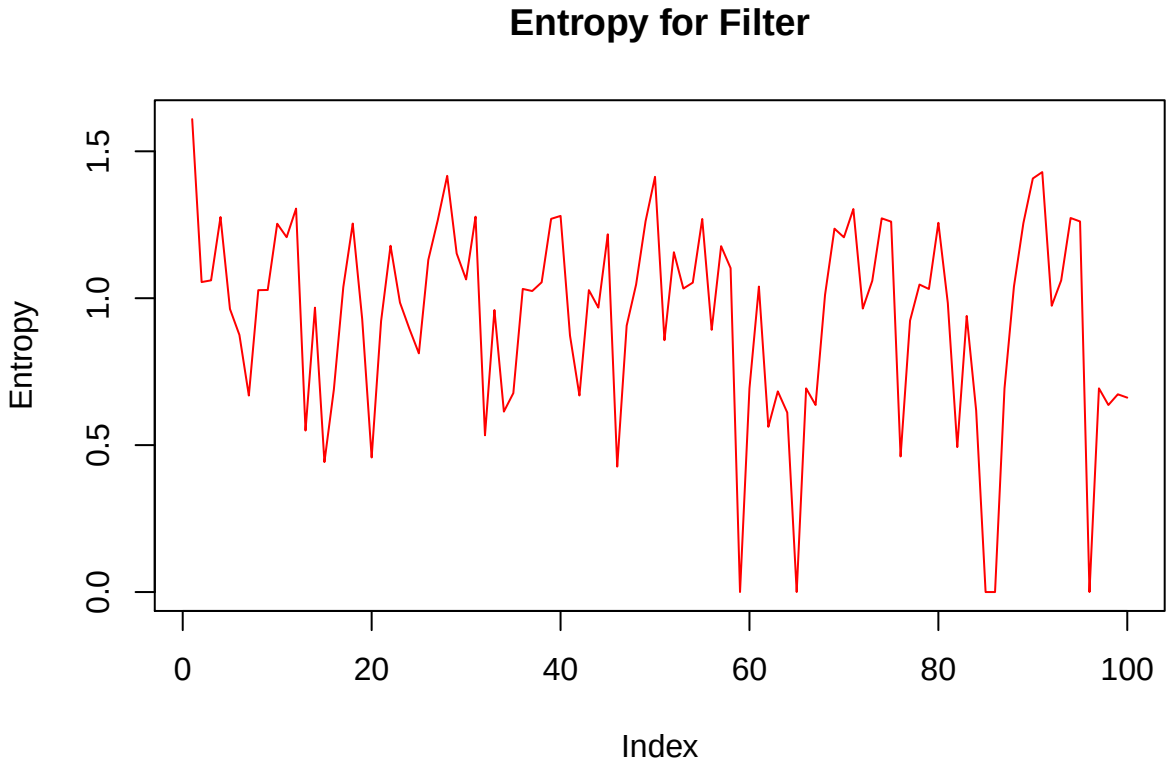
Question 6

The accuracy of the actual predictions. if we have a vector of 1 followed by 0 we will have a entropy value of 0 since we are certain about the result.

```
#Library
library(entropy)

entropy_fun <- function(){
  apply(prob_dist()$Filtering, MARGIN = 2, entropy.empirical)
}

plot(entropy_fun(), x = 1:length(entropy_fun()),
     col = "red", xlab = "Index", ylab = "Entropy",
     type = "l", main = "Entropy for Filter")
```



Question 7

```
prediction <- function(){
  res <- transmat%*%prob_dist()$Filter[,100]
  res <- as.data.frame(res)
  res
}

dfPred <- prediction()

colnames(dfPred) <- c("Prediction Probability")

rownames(dfPred) <- c("Z1","Z2","Z3","Z4","Z5","Z6","Z7","Z8","Z9","Z10")

kable(dfPred, caption = "Probability of state")
```

Table 3: Probability of state

	Prediction Probability
Z1	0.1875
Z2	0.5000
Z3	0.3125
Z4	0.0000
Z5	0.0000
Z6	0.0000
Z7	0.0000
Z8	0.0000
Z9	0.0000
Z10	0.0000