732A96: Advance Machine Learning

LAB 2: HIDDEN MARKOV MODELS

Arian Barakat, ariba405
Rebin Hosini, rebho150
Hao chi Kiang, haoki222
Joshua Hudson, joshu107
Carles Sans Fuentes, carsa564

Background

The purpose of the lab is to put in practice some of the concepts covered in the lectures. To do so, you are asked to model the behavior of a robot that walks around a ring. The ring is divided into 10 sectors. At any given time point, the robot is in one of the sectors and decides with equal probability to stay in that sector or move to the next sector. You do not have direct observation of the robot. However, the robot is equipped with a tracking device that you can access. The device is not very accurate though: If the robot is in the sector i, then the device will report that the robot is in the sectors $[i-2, i+2]$ with equal probability.

Questions

(1)

Question:
Build a HMM for the scenario described above

Answer:

See code in appendix

(2)

Question:
Simulate the HMM for 100 time steps.

Answer:

See code in appendix

(3)

Question:

Discard the hidden states from the sample obtained above. Use the remaining observations to compute the filtered and smoothed probability distributions for each of the 100 time points. Compute also the most probable path.

Answer:

See code in appendix

(4)

Question:

Compute the accuracy of the filtered and smoothed probability distributions, and of the most probable path. That is, compute the percentage of the true hidden states that are guessed by each method.

[Hint: Note that the function forward in the HMM package returns probabilities in log scale. You may need to use the functions exp and prop.table in order to obtain a normalized probability distribution. You may also want to use the functions apply and which.max to find out the most probable states. Finally, recall that you can compare two vectors A and B elementwise as A==B, and that the function table will count the number of times that the different elements in a vector occur in the vector.]

Answer:

Table 1: Accuracy given Method

Filtered	0.59
Smoothed	0.75
Viterbi	0.55

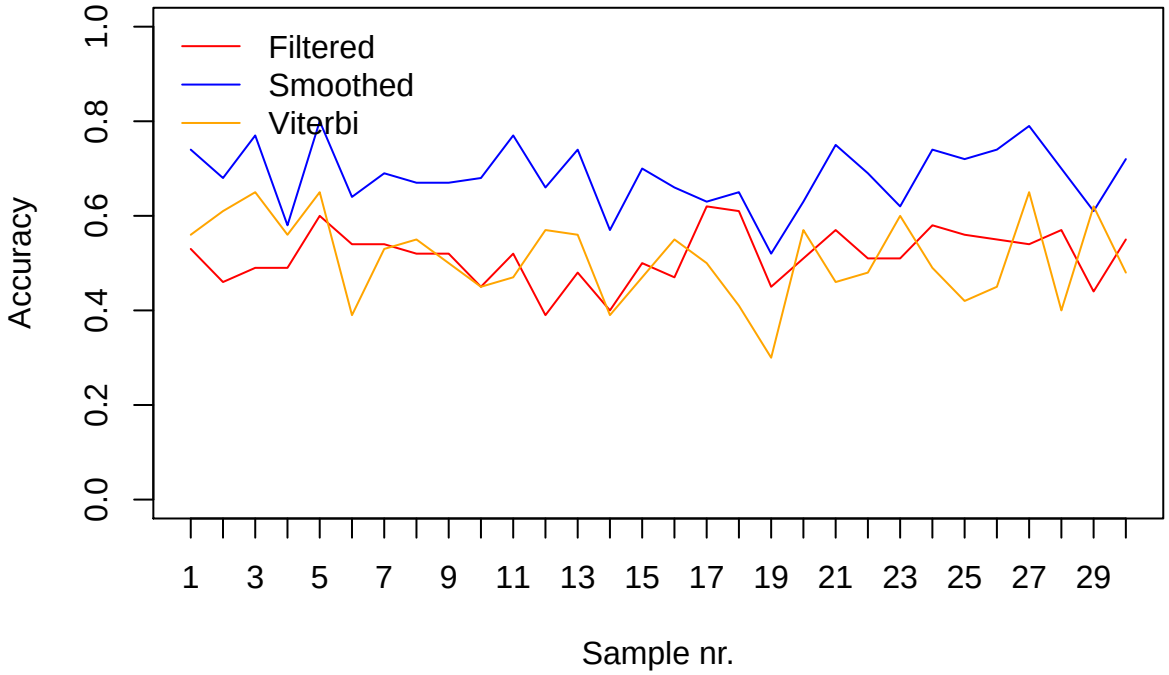

Figure 1: Accuracy with respect to Sample nr. and Method

(5)

Question:

Repeat the previous exercise with different simulated samples. In general, the smoothed distributions should be more accurate than the filtered distributions. Why ? In general, the smoothed distributions should be more accurate than the most probable paths, too. Why ?

Answer:

From figure 1 it can be observed that the accuracy of the Smoothed approach is relatively higher than the other two methods. The reason that the Smoothed approach is more accurate than the other two methods is due to that it uses all data at once when drawing inference about the position of the robot. The filtering and the Viterbi use the data sequentially. It can be assumed that the Smoothing method is relatively more accurate in general, due to the fact that it uses all data, but it might be an unrealistic approach depending on the application.

It should also be noted that the Viterbi algorithm always returns a consistent path, while the other two can return a path which has multi-step jump, which is impossible according to our assumption that the robot can only move one step at a time. This constraint in the Viterbi algorithm, in fact, leads to a worse result point-wise, since a wrong prediction in a particular step would make it more likely that the next step's prediction is wrong as well. On the other hand, we do not have this problem if we do not insist that the path is consistent.

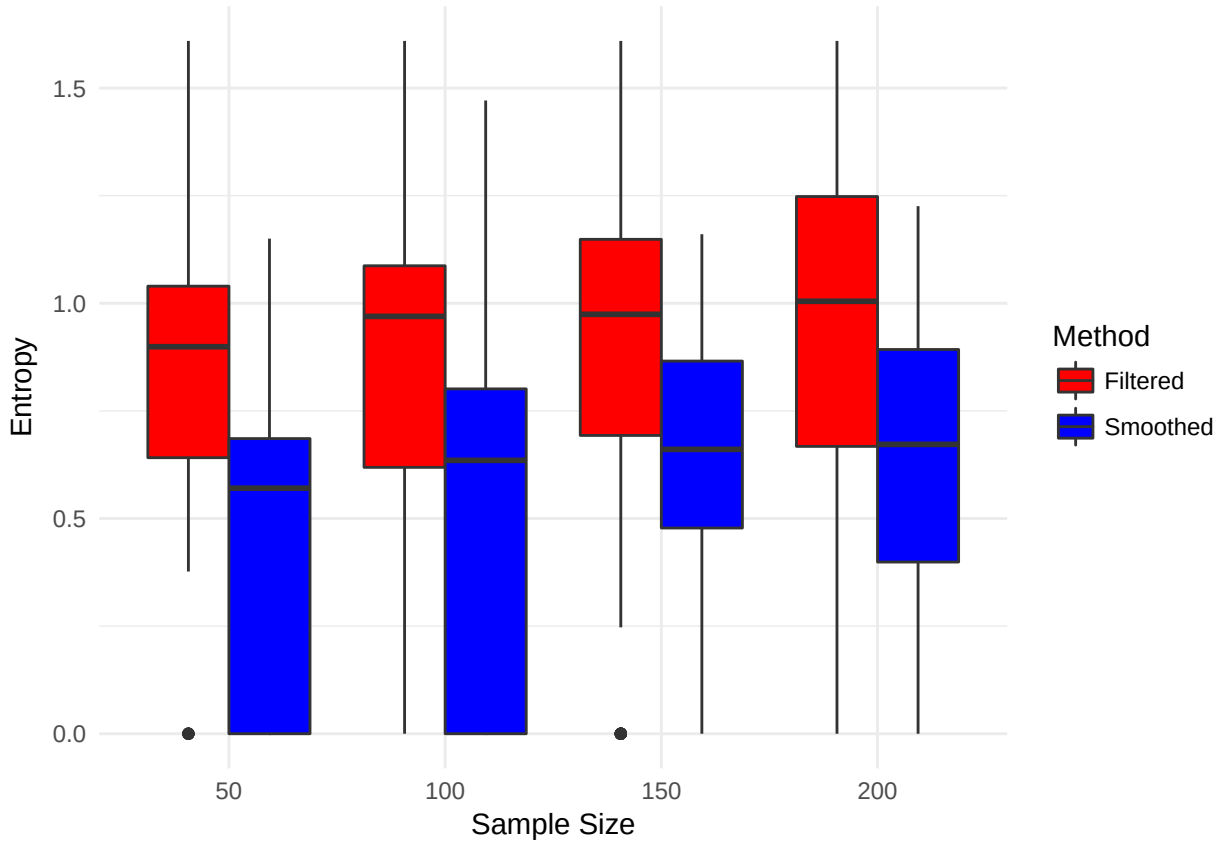

Figure 2: Boxplot of Entropy with respect to Sample Size and Method

(6)

Question:

Is it true that the more observations you have the better you know where the robot is?

[Hint: You may want to compute the entropy of the filtered distributions with the function entropy.empirical of the package entropy.]

Answer:

Figure 2 displays boxplots of the entropies for different methods with respect to sample size. As seen in the figure, the median for each method seems to stay relatively the same as the number of observation increase. This can be interpreted as that our certainty about the whereabouts of the robot does not change as the number of observation increase.

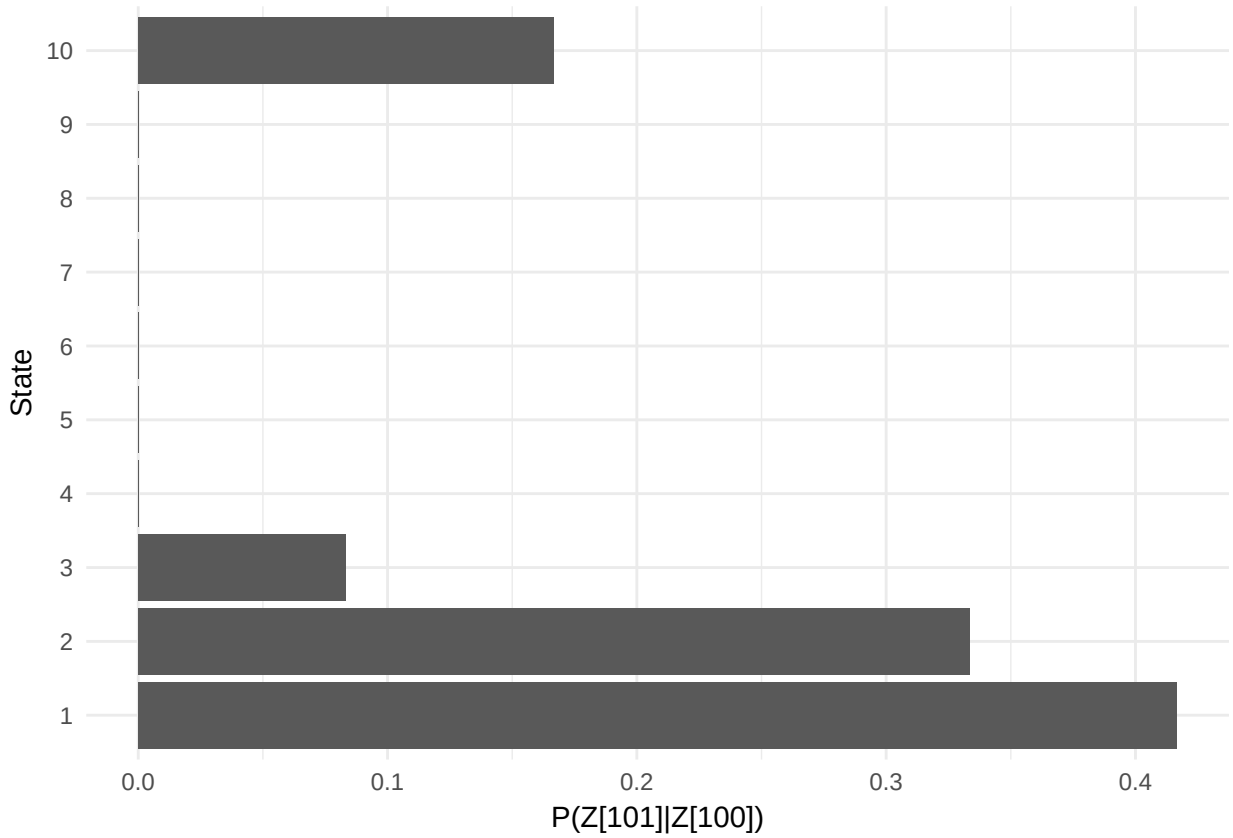

Figure 3: Probabilities of the hidden states for the time step 101

(7)

Question:

Consider any of the samples above of length 100. Compute the probabilities of the hidden states for the time step 101.

Answer:

Table 2: Probabilities of the hidden states for the time step 101

States	Pr
1	0.4166667
2	0.3333333
3	0.0833333
4	0.0000000
5	0.0000000
6	0.0000000
7	0.0000000
8	0.0000000
9	0.0000000
10	0.1666667

Appendix

```
knitr::opts_chunk$set(echo = FALSE,
                      warning = FALSE,
                      message = FALSE)

library(HMM)
library(knitr)
library(entropy)
library(ggplot2)
library(gridExtra)


# Question 1

# The transition Matrix

nrSectors <- 10
intervalLength <- 5
prTransChange <- 1/2 # Uniform
prEmiss <- 1/5 # Uniform


# Creating Trans.Matrix and filling it w/ pr
# + We assumes that the robot can go backward as well
A <- matrix(nrow = nrSectors,
            ncol = nrSectors)

# Creating Emiss.Matrix and filling it w/ pr
E <- matrix(nrow = nrSectors,
            ncol = nrSectors)


for(rows in 1:nrSectors){
  for(cols in 1:nrSectors){
    A[rows, cols] <- ifelse(((rows-cols) <1 && (rows-cols) >= -1),prTransChange,
                            ifelse(cols == 10 && rows-cols >= 9, prTransChange, 0))
    E[rows, cols] <- ifelse(abs(rows - cols) <= 2 || abs(rows - cols) >= 8,
                            prEmiss,
                            0)
  }
}

A[10,1] <- prTransChange


HMM_model <- initHMM(States = as.character(1:nrSectors),
                     Symbols = as.character(1:nrSectors),
                     transProbs = A,
                     emissionProbs = E)
```

```r
set.seed(123456)

HMM_model_sim<- simHMM(HMM_model,
                       length = 100)




# Question 3
simulated_states <- HMM_model_sim$states
simulated_obs <- HMM_model_sim$observation


logFilteredPr <- forward(HMM_model,
                         observation =  simulated_obs)

SmoothPr <- posterior(HMM_model,
                      observation = simulated_obs)


mostProbPath <- viterbi(HMM_model,
                        observation = simulated_obs)

methodPr <- list("Filtered" = logFilteredPr,
                 "Smoothed" = SmoothPr,
                 "Viterbi" = mostProbPath)




# Question 4

predictState <- function(methodList){

  predicted <- list()

  for(i in 1:length(methodList)){

    if(names(methodList)[i] == "Filtered"){
      predicted[[i]] <- as.vector(apply(exp(methodList[[i]]), MARGIN = 2, FUN = function(x){
        which.max(prop.table(x))
      }))
    }

    if(names(methodList)[i] == "Smoothed"){
      predicted[[i]] <- as.vector(apply(methodList[[i]], 2, which.max))
    }

    if(names(methodList)[i] == "Viterbi"){
      predicted[[i]] <- as.numeric(methodList[[i]])
    }

  }
```

```
  predicted <- lapply(predicted, as.character)
  names(predicted) <- c("Filtered",
                        "Smoothed",
                        "Viterbi")

  return(predicted)

}

probableState <- predictState(methodPr)

calcAccur <- function(predictedState, trueStates){
  accurMethod <- lapply(predictedState, FUN = function(x, trueVal){
    sum(x == trueVal)/length(x)
  }, trueVal = trueStates)

  return(unlist(accurMethod))
}


estimatedAccur <- calcAccur(probableState, simulated_states)


kable(as.matrix(estimatedAccur), caption = "Accuracy given Method")




itSamples <- rep(100, 30)
accurMatrix <- matrix(nrow = length(itSamples),
                      ncol = 3)
set.seed(12345)
for(iter in 1:length(itSamples)){

  HMM_model_sim_temp <- simHMM(HMM_model,
                               length = itSamples[iter])

  simulated_states_temp <- HMM_model_sim_temp$states
  simulated_obs_temp <- HMM_model_sim_temp$observation


  logFilteredPr_temp <- forward(HMM_model,
                                observation =  simulated_obs_temp)

  SmoothPr_temp <- posterior(HMM_model,
                             observation = simulated_obs_temp)


  mostProbPath_temp <- viterbi(HMM_model,
                               observation = simulated_obs_temp)

  methodPr_temp <- list("Filtered" = logFilteredPr_temp,
```

```
                          "Smoothed" = SmoothPr_temp,
                          "Viterbi" = mostProbPath_temp)

  probableState_temp <- predictState(methodPr_temp)

  accurMatrix[iter,] <- calcAccur(probableState_temp, simulated_states_temp)
}


xGrid <- 1:length(itSamples)
plot(x = xGrid,
     y = accurMatrix[,1],
     ylab = "Accuracy",
     xlab = "Sample nr.",
     col = "red",
     type = "l",
     ylim = c(0,1),
     xaxt = "n")
axis(1, at = xGrid)
lines(x = xGrid,
     y = accurMatrix[,2],
     col = "blue",
     type = "l")
lines(x = xGrid,
     y = accurMatrix[,3],
     col = "orange",
     type = "l")
legend("topleft",
       legend = names(methodPr),
       col = c("red", "blue", "orange"),
       lty = rep(1,3),
       bty = "n")



# Question 6

calcEntropy <- function(methodList){

  entrop <- list()

  for(i in 1:length(methodList)){

    if(names(methodList)[i] == "Filtered"){
      entrop[[i]] <- as.vector(apply(exp(methodList[[i]]), MARGIN = 2, FUN = function(x){
        entropy.empirical(prop.table(x))
      }))
    }

    if(names(methodList)[i] == "Smoothed"){
      entrop[[i]] <- as.vector(apply(methodList[[i]], 2, entropy.empirical))
    }
```

```
  }

  names(entrop) <- c("Filtered","Smoothed")

  return(entrop)

}


itSamples <- seq(50,200, 50)
entropies <- list()

set.seed(123456)
for(iter in 1:length(itSamples)){

  HMM_model_sim_temp <- simHMM(HMM_model,
                               length = itSamples[iter])

  simulated_states_temp <- HMM_model_sim_temp$states
  simulated_obs_temp <- HMM_model_sim_temp$observation


  logFilteredPr_temp <- forward(HMM_model,
                                observation =  simulated_obs_temp)

  SmoothPr_temp <- posterior(HMM_model,
                             observation = simulated_obs_temp)


  methodPr_temp <- list("Filtered" = logFilteredPr_temp,
                        "Smoothed" = SmoothPr_temp)

  entropies[[iter]] <- calcEntropy(methodPr_temp)

}


dfList <- list()


for(iter in 1:length(itSamples)){

  dfList[[iter]] <- data.frame(Entropy = c(entropies[[iter]][[1]],entropies[[iter]][[2]]),
                               Method =  as.factor(c(rep("Filtered", itSamples[iter]),
                                                     rep("Smoothed", itSamples[iter]))),
                               Iteration = as.character(itSamples[iter]))

}


mergedDF <- Reduce(rbind, dfList)

ggplot(data = mergedDF) +
```

```
  geom_boxplot(aes(y = Entropy, fill = Method, x = Iteration)) +
  xlab("Sample Size") +
  scale_fill_manual(values = c("red", "blue")) +
  theme_minimal()



# Question 7


# Will be using the table from the posterior function at step 100

hiddenStateStep101 <- SmoothPr[,100] %*% A


ggplot(data = data.frame(pr = as.vector(hiddenStateStep101),
                         states = as.factor(1:length(hiddenStateStep101)))) +
  geom_bar(aes(y = pr, x = states), stat = "identity") +
  xlab("State") + ylab(expression("P(Z[101]|Z[100])")) +
  coord_flip() +
  theme_minimal()

kable((data.frame(States = as.factor(1:length(hiddenStateStep101)),
                  Pr = as.vector(hiddenStateStep101))),
      caption = "Probabilities of the hidden states for the time step 101")
```