Advanced Machine Learning (732A96) Lab2

Anubhav Dikshit(anudi287)

29 September, 2019

Contents

Questions

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 to i+2 with equal probability.

Questions 1

1) Build a hidden Markov model (HMM) for the scenario described above.

```
set.seed(12345)

transition_mat <- matrix(data = c(0.5, 0.5, 0, 0, 0, 0, 0, 0, 0, 0,
                                  0, 0.5, 0.5, 0, 0, 0, 0, 0, 0, 0,
                                  0, 0, 0.5, 0.5, 0, 0, 0, 0, 0, 0,
                                  0, 0, 0, 0.5, 0.5, 0, 0, 0, 0, 0,
                                  0, 0, 0, 0, 0.5, 0.5, 0, 0, 0, 0,
                                  0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0,
                                  0, 0, 0, 0, 0, 0, 0.5, 0.5, 0, 0,
                                  0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0,
                                  0, 0, 0, 0, 0, 0, 0, 0, 0.5, 0.5,
                                  0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0.5),
                         nrow = 10,
                         ncol = 10)

sensor_mat <- matrix(data = c(0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0.2, 0.2,
                              0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0.2,
                              0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0,
                              0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0,
                              0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0,
                              0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0,
                              0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0,
                              0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2,
                              0.2, 0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2,
                              0.2, 0.2, 0, 0, 0, 0, 0, 0.2, 0.2, 0.2),
                              nrow = 10,
                              ncol = 10)

sector_10_model <- initHMM(States = c("1","2","3","4","5","6","7","8","9","10"),
                           Symbols = c("1","2","3","4","5","6","7","8","9","10"),
                           startProbs = rep(0.1, 10),
                           transProbs = transition_mat,
                           emissionProbs = sensor_mat)
```

Questions 2

2) Simulate the HMM for 100 time steps.

```r
set.seed(12345)

hmm_100 <- simHMM(sector_10_model, length=100)
hmm_100
```

```
## $states
##   [1] "9"  "9"  "9"  "9"  "8"  "7"  "6"  "5"  "5"  "5"  "4"  "3"  "2"  "2"
##  [15] "2"  "2"  "2"  "2"  "1"  "1"  "1"  "10" "9"  "9"  "8"  "8"  "8"
##  [29] "7"  "6"  "5"  "4"  "3"  "2"  "2"  "2"  "1"  "10" "9"  "8"  "8"  "7"
##  [43] "7"  "7"  "6"  "6"  "6"  "6"  "6"  "5"  "5"  "5"  "4"  "3"  "2"  "2"
##  [57] "1"  "1"  "1"  "1"  "10" "10" "9"  "9"  "9"  "8"  "8"  "7"  "7"  "7"
##  [71] "7"  "7"  "6"  "6"  "6"  "5"  "5"  "5"  "5"  "4"  "4"  "4"  "3"  "3"
##  [85] "3"  "3"  "2"  "1"  "10" "10" "9"  "9"  "8"  "8"  "8"  "8"  "8"  "8"
##  [99] "7"  "6"
##
## $observation
##   [1] "7"  "10" "8"  "10" "10" "6"  "4"  "6"  "3"  "3"  "2"  "4"  "10" "4"
##  [15] "10" "3"  "3"  "1"  "2"  "2"  "1"  "1"  "10" "10" "10" "6"  "6"  "8"
##  [29] "7"  "4"  "7"  "2"  "3"  "10" "3"  "3"  "10" "10" "9"  "6"  "10" "9"
##  [43] "7"  "9"  "8"  "6"  "6"  "5"  "6"  "3"  "6"  "5"  "2"  "4"  "4"  "10"
##  [57] "10" "1"  "3"  "10" "1"  "2"  "8"  "8"  "7"  "10" "6"  "8"  "6"  "7"
##  [71] "6"  "8"  "4"  "7"  "7"  "4"  "4"  "6"  "4"  "6"  "4"  "5"  "2"  "4"
##  [85] "3"  "3"  "4"  "3"  "10" "10" "10" "8"  "7"  "7"  "6"  "8"  "7"  "9"
##  [99] "7"  "5"
```

Questions 3

3) 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.

```r
set.seed(12345)

# The library retruns the probabilities logged, we we have to de-log
alpha = exp(forward(sector_10_model, hmm_100$observation))
beta = exp(backward(sector_10_model, hmm_100$observation))

# Filtering which is defined as alpha column, divided by its col sum
filtered =  sweep(alpha, 2, colSums(alpha), FUN="/")

# Smoothing
smoothing = alpha * beta
smoothing =  sweep(smoothing, 2, colSums(smoothing), FUN="/")

# Path
hmm_viterbi = viterbi(sector_10_model, hmm_100$observation)
as.numeric(hmm_viterbi)
```

```
##   [1]  8  8  8  8  8  7  6  5  4  3  2  2  2  2  1  1  1  1  1 10  9  9  8
##  [24]  8  8  7  6  6  5  5  5  4  3  2  1  1 10  9  8  8  8  7  7  7  6  5
##  [47]  4  4  4  4  4  3  2  2  2  1  1  1  1  1  1 10  9  8  8  8  7  6  6
##  [70]  6  6  6  5  5  5  4  4  4  4  4  3  3  2  2  2  2  2  1 10  9  8  7
##  [93]  7  7  7  7  7  7  6  5
```

Questions 4

4) 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.

```
set.seed(12345)

# finding the max of each column for 100 entries for filtered values
filtered_path <- t(filtered)
filtered_path <- max.col(filtered_path, "first")

# finding the max of each column for 100 entries for smoothened values
smoothing_path <- t(smoothing)
smoothing_path <- max.col(smoothing_path, "first")

# viterbi path
viterbi_path <- as.numeric(hmm_viterbi)

# actual path
actual_state <- as.numeric(hmm_100$states)

# Accuracy of filtered path
acc_filered <- sum(actual_state == filtered_path)/length(filtered_path) * 100

# Accuracy of smoothened path
acc_smooth <- sum(actual_state == smoothing_path)/length(smoothing_path) * 100

# Accuracy of viterbi path
viterbi_smooth <- sum(actual_state == viterbi_path)/length(viterbi_path) * 100

cat("Accuracy of filtered, smoothing and viterbi are as follows:", acc_filered, "%,", acc_smooth, "% an
```

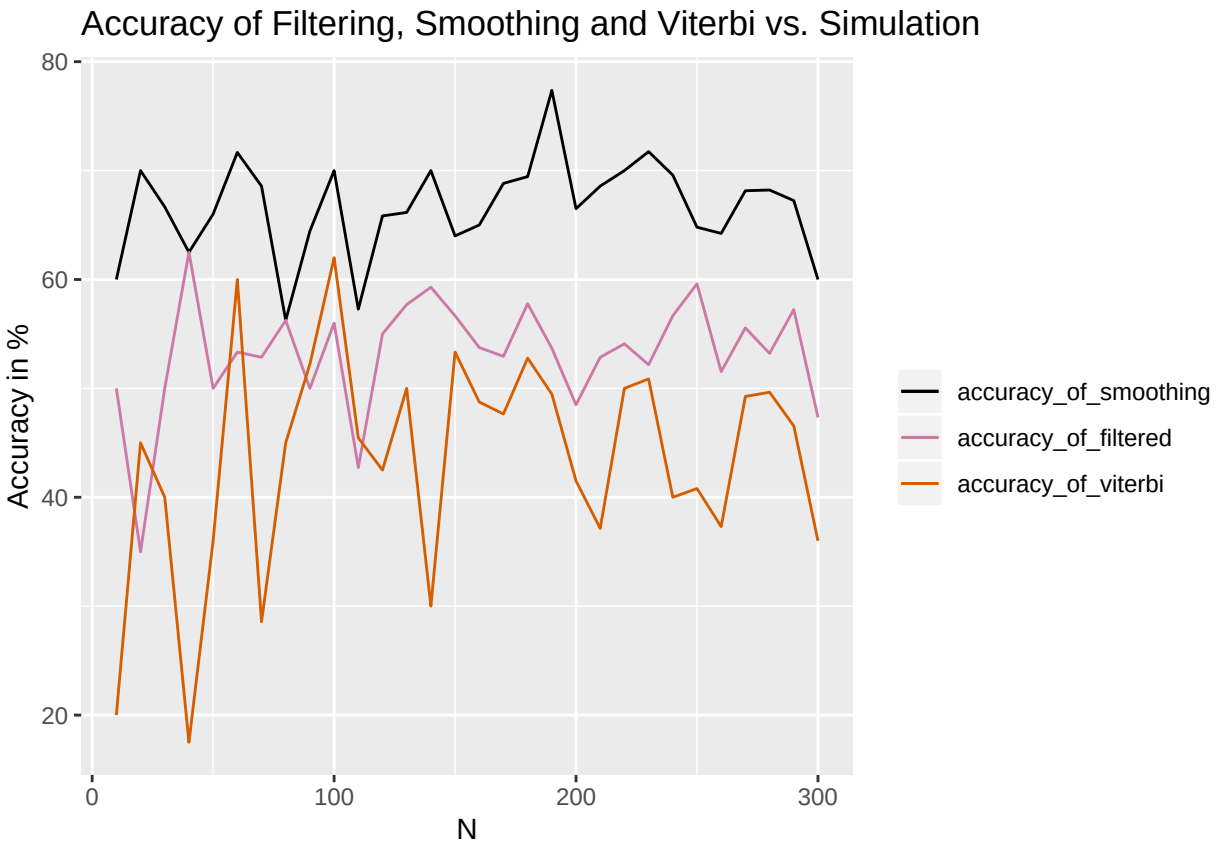
```
## Accuracy of filtered, smoothing and viterbi are as follows: 49 %, 63 % and  28 %
```

Questions 5

5) 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?

```
set.seed(12345)

get_hmm_accuracy <- function(N, model)
{
hmm_N <- simHMM(model, length=N)
alpha = exp(forward(model, hmm_N$observation))
beta = exp(backward(model, hmm_N$observation))

filtered =  sweep(alpha, 2, colSums(alpha), FUN="/")

smoothing = alpha * beta
smoothing =  sweep(smoothing, 2, colSums(smoothing), FUN="/")
```

```
hmm_viterbi = viterbi(model, hmm_N$observation)

filtered_path <- t(filtered)
filtered_path <- max.col(filtered_path, "first")


smoothing_path <- t(smoothing)
smoothing_path <- max.col(smoothing_path, "first")

viterbi_path <- as.numeric(hmm_viterbi)
actual_state <- as.numeric(hmm_N$states)


acc_filered <- sum(actual_state == filtered_path)/length(filtered_path) * 100
acc_smooth <- sum(actual_state == smoothing_path)/length(smoothing_path) * 100
viterbi_smooth <- sum(actual_state == viterbi_path)/length(viterbi_path) * 100

df <- data.frame(N=N, accuracy_of_filtered = acc_filered, accuracy_of_smoothing = acc_smooth, accuracy_
return(df)
}

final <- NULL
for (i in seq(10, 300, 10)) {
temp <- get_hmm_accuracy(i, sector_10_model)
final <- rbind(temp, final)
}

ggplot(final, aes(x = N)) +
    geom_line(aes(y=accuracy_of_smoothing, color="accuracy_of_smoothing")) +
    geom_line(aes(y=accuracy_of_filtered, color="accuracy_of_filtered")) +
  geom_line(aes(y=accuracy_of_viterbi, color="accuracy_of_viterbi")) +
  ggtitle("Accuracy of Filtering, Smoothing and Viterbi vs. Simulation") +
  ylab("Accuracy in %") +
    scale_colour_manual("", breaks = c("accuracy_of_smoothing", "accuracy_of_filtered", "accuracy_of_vi
                        values = c("#CC79A7", "#000000", "#D55E00"))
```

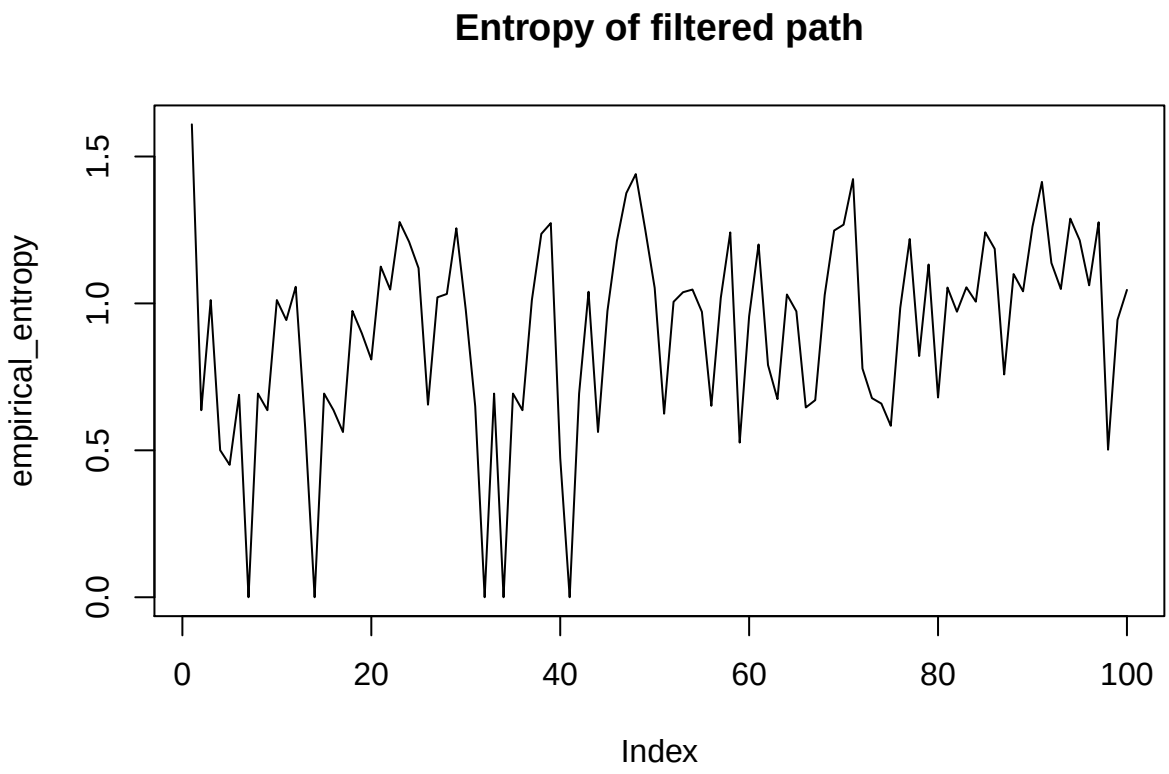


Analysis: The smoothed distribution gives better accuracy because it takes into account the whole series from 0 till N, on the filtred is a lot of more constraint in terms of sampling.

Questions 6

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

```
set.seed(12345)
empirical_entropy = apply(filtered, 2, entropy.empirical)
plot(empirical_entropy, type= 'l', main = "Entropy of filtered path")
```



Analysis: We know that entropy is a measure of the chaos in any system, from the plot we can see that its varying throughout of the position. Thus no its not clear that the more observation we have the more we know of the robots position, this is in agreement with Markov's property that current state depends only on the previous state.

Questions 7

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

```
set.seed(12345)
position = transition_mat %*% filtered[,100]
rownames(position) = names(filtered[,100])
position
```

```
##         [,1]
## 1  0.0000000
## 2  0.0000000
## 3  0.0000000
## 4  0.0000000
## 5  0.1051418
## 6  0.3419030
## 7  0.3948582
## 8  0.1580970
## 9  0.0000000
```

```
## 10 0.0000000
```

Analysis: The probabilities of each of the state is given above.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)

library("tidyverse") #ggplot and dplyr
library("gridExtra") # combine plots
library("knitr") # for pdf
library("bnlearn") # ADM
library("gRain") # ADM
library("entropy")
library("HMM") #Hidden Markov Models


if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
#BiocManager::install("gRain")


# The palette with black:
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
                "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
set.seed(12345)
set.seed(12345)

transition_mat <- matrix(data = c(0.5, 0.5, 0, 0, 0, 0, 0, 0, 0, 0,
                                  0, 0.5, 0.5, 0, 0, 0, 0, 0, 0, 0,
                                  0, 0, 0.5, 0.5, 0, 0, 0, 0, 0, 0,
                                  0, 0, 0, 0.5, 0.5, 0, 0, 0, 0, 0,
                                  0, 0, 0, 0, 0.5, 0.5, 0, 0, 0, 0,
                                  0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0,
                                  0, 0, 0, 0, 0, 0, 0.5, 0.5, 0, 0,
                                  0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0,
                                  0, 0, 0, 0, 0, 0, 0, 0, 0.5, 0.5,
                                  0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0.5),
                         nrow = 10,
                         ncol = 10)

sensor_mat <- matrix(data = c(0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0.2, 0.2,
                              0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0.2,
                              0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0,
                              0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0,
                              0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0,
                              0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0,
                              0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0,
                              0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2,
                              0.2, 0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2,
```

```
                              0.2, 0.2, 0, 0, 0, 0, 0, 0.2, 0.2, 0.2),
                              nrow = 10,
                              ncol = 10)

sector_10_model <- initHMM(States = c("1","2","3","4","5","6","7","8","9","10"),
                           Symbols = c("1","2","3","4","5","6","7","8","9","10"),
                           startProbs = rep(0.1, 10),
                           transProbs = transition_mat,
                           emissionProbs = sensor_mat)

set.seed(12345)

hmm_100 <- simHMM(sector_10_model, length=100)
hmm_100
set.seed(12345)

# The library retruns the probabilities logged, we we have to de-log
alpha = exp(forward(sector_10_model, hmm_100$observation))
beta = exp(backward(sector_10_model, hmm_100$observation))

# Filtering which is defined as alpha column, divided by its col sum
filtered =  sweep(alpha, 2, colSums(alpha), FUN="/")

# Smoothing
smoothing = alpha * beta
smoothing =  sweep(smoothing, 2, colSums(smoothing), FUN="/")

# Path
hmm_viterbi = viterbi(sector_10_model, hmm_100$observation)
as.numeric(hmm_viterbi)

set.seed(12345)

# finding the max of each column for 100 entries for filtered values
filtered_path <- t(filtered)
filtered_path <- max.col(filtered_path, "first")

# finding the max of each column for 100 entries for smoothened values
smoothing_path <- t(smoothing)
smoothing_path <- max.col(smoothing_path, "first")

# viterbi path
viterbi_path <- as.numeric(hmm_viterbi)

# actual path
actual_state <- as.numeric(hmm_100$states)

# Accuracy of filtered path
acc_filered <- sum(actual_state == filtered_path)/length(filtered_path) * 100

# Accuracy of smoothened path
acc_smooth <- sum(actual_state == smoothing_path)/length(smoothing_path) * 100
```

```r
# Accuracy of viterbi path
viterbi_smooth <- sum(actual_state == viterbi_path)/length(viterbi_path) * 100

cat("Accuracy of filtered, smoothing and viterbi are as follows:", acc_filered, "%,", acc_smooth, "% an

set.seed(12345)

get_hmm_accuracy <- function(N, model)
{
hmm_N <- simHMM(model, length=N)
alpha = exp(forward(model, hmm_N$observation))
beta = exp(backward(model, hmm_N$observation))

filtered =  sweep(alpha, 2, colSums(alpha), FUN="/")

smoothing = alpha * beta
smoothing =  sweep(smoothing, 2, colSums(smoothing), FUN="/")

hmm_viterbi = viterbi(model, hmm_N$observation)

filtered_path <- t(filtered)
filtered_path <- max.col(filtered_path, "first")


smoothing_path <- t(smoothing)
smoothing_path <- max.col(smoothing_path, "first")

viterbi_path <- as.numeric(hmm_viterbi)
actual_state <- as.numeric(hmm_N$states)


acc_filered <- sum(actual_state == filtered_path)/length(filtered_path) * 100
acc_smooth <- sum(actual_state == smoothing_path)/length(smoothing_path) * 100
viterbi_smooth <- sum(actual_state == viterbi_path)/length(viterbi_path) * 100

df <- data.frame(N=N, accuracy_of_filtered = acc_filered, accuracy_of_smoothing = acc_smooth, accuracy_
return(df)
}

final <- NULL
for (i in seq(10, 300, 10)) {
temp <- get_hmm_accuracy(i, sector_10_model)
final <- rbind(temp, final)
}

ggplot(final, aes(x = N)) +
    geom_line(aes(y=accuracy_of_smoothing, color="accuracy_of_smoothing")) +
    geom_line(aes(y=accuracy_of_filtered, color="accuracy_of_filtered")) +
  geom_line(aes(y=accuracy_of_viterbi, color="accuracy_of_viterbi")) +
  ggtitle("Accuracy of Filtering, Smoothing and Viterbi vs. Simulation") +
  ylab("Accuracy in %") +
    scale_colour_manual("", breaks = c("accuracy_of_smoothing", "accuracy_of_filtered", "accuracy_of_vi
                        values = c("#CC79A7", "#000000", "#D55E00"))
```

```r
set.seed(12345)
empirical_entropy = apply(filtered, 2, entropy.empirical)
plot(empirical_entropy, type= 'l', main = "Entropy of filtered path")


set.seed(12345)
position = transition_mat %*% filtered[,100]
rownames(position) = names(filtered[,100])
position
```