Advanced Machine Learning - Lab 2 - Individual report

Nicolas Taba (nicta839)

20/09/2021

```r
# Load libraries
library(HMM)
```

```
## Warning: package 'HMM' was built under R version 4.0.3
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```r
library(knitr)
library(entropy)
```

```
## Warning: package 'entropy' was built under R version 4.0.5
```

Question 1

```r
# Parameters
states <- as.character(c(1:10))
symbols <- as.character(c(1:10))
startprob <- rep((1/length(states)), length(states))

# transition matrix
transprob <- diag(x = 0.5, nrow = 10, ncol = 10)
index <- which(transprob == 0.5)
transprob[index[-length(index)] - 1] <- 0.5
transprob[9, 10] <- 0.5
transprob[10, 1] <- 0.5

colnames(transprob) <- as.character(c(1:10))
rownames(transprob) <- as.character(c(1:10))

# Emission Matrix (there's a smarter way of doing this for sure)
emissionprob <- matrix(data = 0, nrow = 10, ncol = 10)
emissionprob[1, c(9, 10, 1, 2, 3)] <- 0.2
emissionprob[2, c(10, 1, 2, 3, 4)] <- 0.2
emissionprob[3, c(1, 2, 3, 4, 5)] <- 0.2
emissionprob[4, c(2, 3, 4, 5, 6)] <- 0.2
emissionprob[5, c(3, 4, 5, 6, 7)] <- 0.2
emissionprob[6, c(4, 5, 6, 7, 8)] <- 0.2
emissionprob[7, c(5, 6, 7, 8, 9)] <- 0.2
emissionprob[8, c(6, 7, 8, 9, 10)] <- 0.2
emissionprob[9, c(7, 8, 9, 10, 1)] <- 0.2
emissionprob[10, c(8, 9, 10, 1, 2)] <- 0.2
```

```
colnames(emissionprob) <- as.character(c(1:10))
rownames(emissionprob) <- as.character(c(1:10))
```

```
# initialize HMM
HMM <- initHMM(States = states, Symbols = symbols, startProbs = startprob,
               transProbs = transprob, emissionProbs = emissionprob)
```

Question 2

```
# We now simulate 100 timesteps
set.seed(12345)
simulation100 <- simHMM(HMM, 100)
```
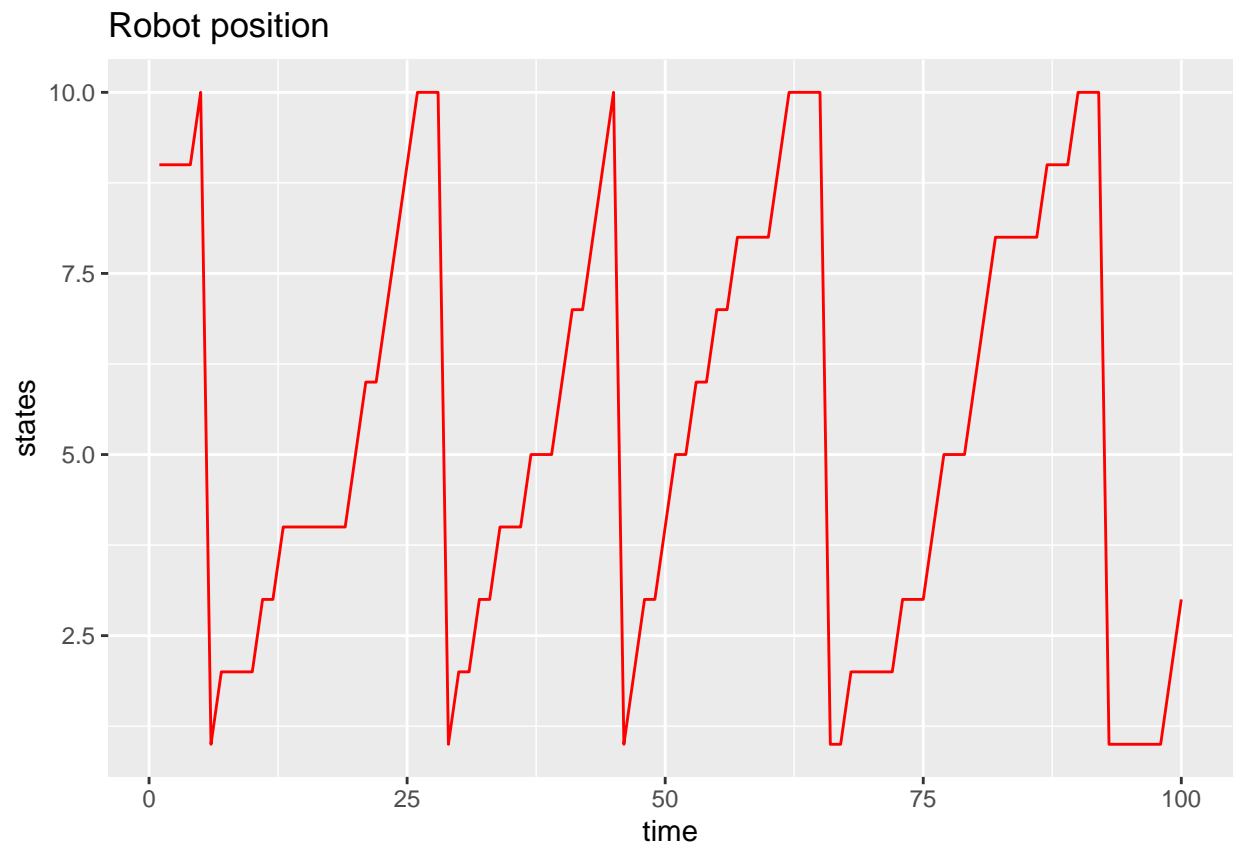
```
# plot results
time <- 1:100
states <- as.numeric(simulation100$states)
observations <- as.numeric(simulation100$observation)
df_plot <- data.frame(time, states, observations)

plot_states <- ggplot(df_plot, aes(x = time, y = states))+
  geom_line(col = "red")+
  ggtitle("Robot position")

plot_recorded <- ggplot(df_plot, aes(x = time, y = observations))+
  geom_line(col = "green")+
  ggtitle("Robot recorded position")

plot_states
```
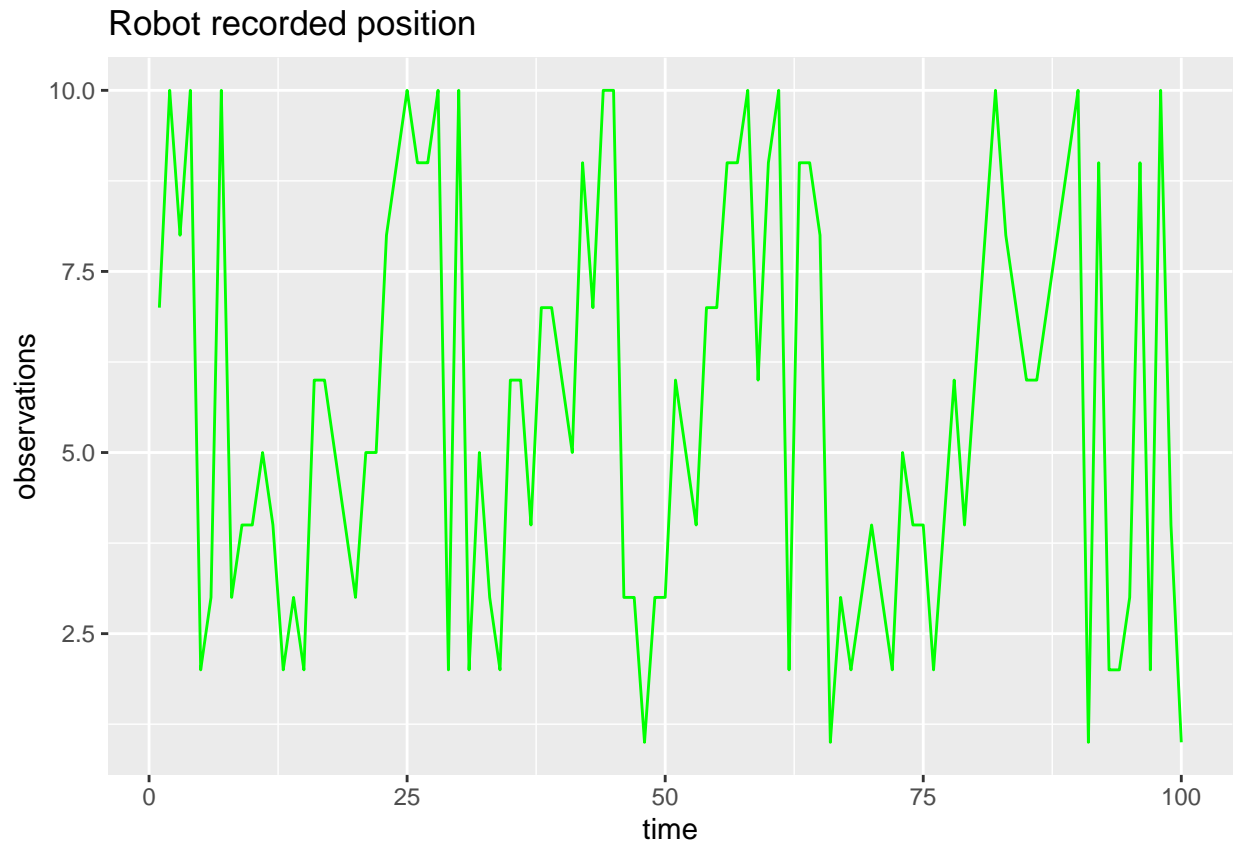
Robot position


```
plot_recorded
```

Robot recorded position

Question 3

```r
# filtered distribution
# alpha
alpha <- exp(forward(hmm = HMM, observation = simulation100$observation))
# beta
beta <- exp(backward(hmm = HMM, observation = simulation100$observation))
# filtered distribution
filter_dist <- prop.table(alpha, margin = 2)
```

```r
# smoothed distribution
smooth_dist <- prop.table(alpha*beta, margin = 2)
```

```r
# most probable path (viterbi algo)
viterbi_path <- viterbi(hmm = HMM, observation = simulation100$observation)
```

Could probably plot the paths here. Overlap them maybe

Question 4

```r
# accuracy filter
path_filter <- as.character(apply(filter_dist, 2, which.max))
filter_mat <- table(path_filter == simulation100$states)
accuracy_filter <- filter_mat[2]/sum(filter_mat)
```

4

```
# accuracy smoothed
path_smooth <- as.character(apply(smooth_dist, 2, which.max))
smooth_mat <- table(path_smooth == simulation100$states)
accuracy_smooth <- smooth_mat[2]/sum(smooth_mat)

# accuracy most probable
viterbi_mat <- table(viterbi_path == simulation100$states)
accuracy_viterbi <- viterbi_mat[2]/sum(viterbi_mat)

accuracy_filter
```

```
## TRUE
## 0.53
```

```
accuracy_smooth
```

```
## TRUE
## 0.74
```

```
accuracy_viterbi
```

```
## TRUE
## 0.56
```

present results in a kable table

Question 5

```
# prepare data frame for experiment
exp_length <- 30
empt_matrix <- matrix(data = 0, nrow = exp_length, ncol = 3)
df_exp <- as.data.frame(empt_matrix)
colnames(df_exp) <- c("Filtered", "Smoothed", "Viterbi")
```

```
# experiment

for(i in 1:exp_length){
  #simulate data
  simul <- simHMM(HMM, length = 100)
  #get alpha, distribution and accuracy
  alpha <- exp(forward(hmm = HMM, observation = simul$observation))
  filterd <- prop.table(alpha, 2)
  path_filt <- as.character(apply(filterd, 2, which.max))
  filt_table <- table(path_filt == simul$states)
  filt_accuracy <- filt_table[2]/sum(filt_table)
  #get beta and distribution and accuracy
  beta <- exp(backward(hmm = HMM, observation = simul$observation))
  smoothd <- prop.table(alpha*beta, 2)
  path_smooth <- as.character(apply(smoothd, 2, which.max))
  smooth_table <- table(path_smooth == simul$states)
  smooth_accuracy <- smooth_table[2]/sum(smooth_table)
  # viterbi path and accuracy
  vit_path <- viterbi(hmm = HMM, observation = simul$observation)
  vit_table <- table(vit_path == simul$states)
  vit_accuracy <- vit_table[2]/sum(vit_table)
  # record accuracy in data frame
```

```
  df_exp[i, ] <- cbind(filt_accuracy, smooth_accuracy, vit_accuracy)
}
```

```
#  print results
kable(df_exp, caption = "Accuracies for the different methods")
```

Table 1: Accuracies for the different methods

| Filtered	Smoothed	Viterbi
0.46	0.68	0.61
0.49	0.77	0.65
0.49	0.58	0.56
0.60	0.79	0.65
0.54	0.64	0.39
0.54	0.69	0.53
0.52	0.67	0.55
0.52	0.67	0.50
0.45	0.68	0.45
0.52	0.77	0.47
0.39	0.66	0.57
0.48	0.74	0.56
0.40	0.57	0.39
0.50	0.70	0.47
0.47	0.66	0.55
0.62	0.63	0.50
0.61	0.65	0.41
0.45	0.52	0.30
0.51	0.63	0.57
0.57	0.75	0.46
0.51	0.70	0.48
0.51	0.62	0.60
0.58	0.74	0.49
0.56	0.72	0.42
0.55	0.74	0.45
0.54	0.79	0.65
0.57	0.69	0.40
0.44	0.61	0.62
0.55	0.72	0.48
0.53	0.69	0.46

Filtered method uses only past data to fit the best model "with the data so far". Viterbi has the constrain of making the most probable possible path and the smoothing performs better because it uses all of the data (not necessarily a valid path)

Question 6

```
# entropy is calculated over each column using entropy.empirical
entropy <- apply(filter_dist, 2, entropy.empirical)
plot(entropy, type = "l")
```
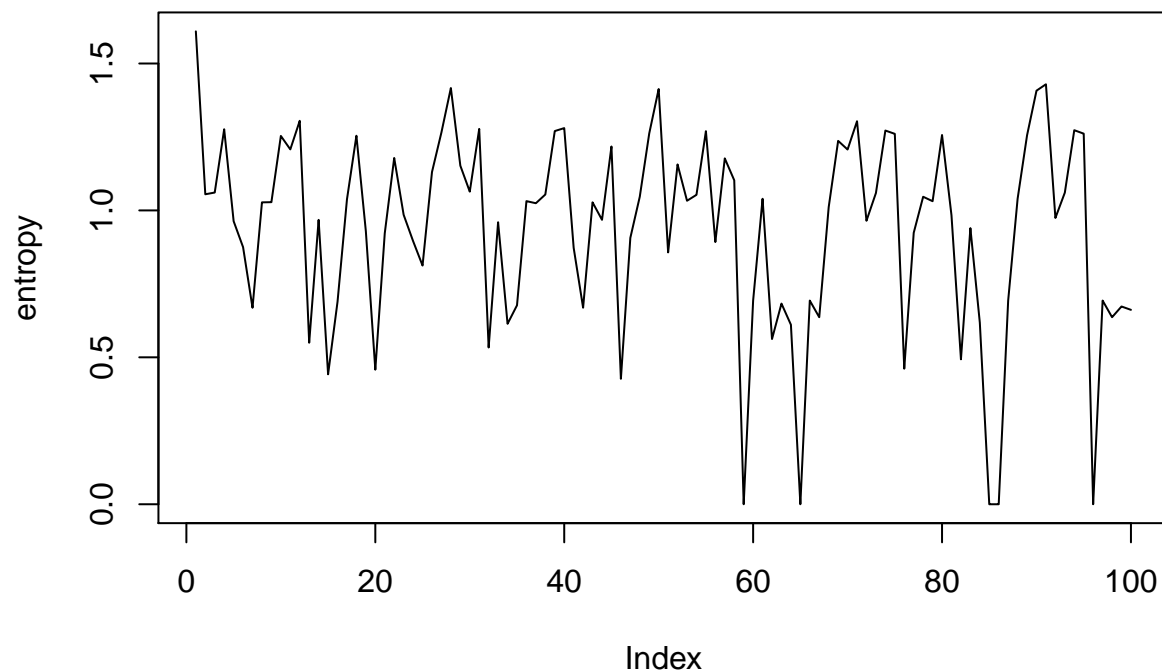
Entropy is not monotonically decreasing with the increase of data so we do not gain more certainty as we add more time steps.

Question 7

we are trying to compute z_101|X_1:100

We can add Z_100 in the conditional as long as we add p(Z_100|X_1:100). Using the proper independences, we get $\sum$ p(Z_101|Z_100)p(Z_100|X_100)

The second factor is the last point of the filtering distribution and the first factor is the transition matrix. We just have to fo the matrix multiplication.

```
# We simply have to multiply the last filter_distribution with the transition matrix
dist_101 <- filter_dist[, 100] %*% transprob
dist_101
```

```
##      1      2   3      4 5 6 7 8 9 10
## [1,] 0 0.1875 0.5 0.3125 0 0 0 0 0  0
```

Appendix: All code for this report

```r
knitr::opts_chunk$set(echo = TRUE)
# Load libraries
library(HMM)
library(ggplot2)
library(knitr)
library(entropy)
# Parameters
states <- as.character(c(1:10))
symbols <- as.character(c(1:10))
startprob <- rep((1/length(states)), length(states))

# transition matrix
transprob <- diag(x = 0.5, nrow = 10, ncol = 10)
index <- which(transprob == 0.5)
transprob[index[-length(index)] - 1] <- 0.5
transprob[9, 10] <- 0.5
transprob[10, 1] <- 0.5

colnames(transprob) <- as.character(c(1:10))
rownames(transprob) <- as.character(c(1:10))

# Emission Matrix (there's a smarter way of doing this for sure)
emissionprob <- matrix(data = 0, nrow = 10, ncol = 10)
emissionprob[1, c(9, 10, 1, 2, 3)] <- 0.2
emissionprob[2, c(10, 1, 2, 3, 4)] <- 0.2
emissionprob[3, c(1, 2, 3, 4, 5)] <- 0.2
emissionprob[4, c(2, 3, 4, 5, 6)] <- 0.2
emissionprob[5, c(3, 4, 5, 6, 7)] <- 0.2
emissionprob[6, c(4, 5, 6, 7, 8)] <- 0.2
emissionprob[7, c(5, 6, 7, 8, 9)] <- 0.2
emissionprob[8, c(6, 7, 8, 9, 10)] <- 0.2
emissionprob[9, c(7, 8, 9, 10, 1)] <- 0.2
emissionprob[10, c(8, 9, 10, 1, 2)] <- 0.2

colnames(emissionprob) <- as.character(c(1:10))
rownames(emissionprob) <- as.character(c(1:10))

# initialize HMM
HMM <- initHMM(States = states, Symbols = symbols, startProbs = startprob,
               transProbs = transprob, emissionProbs = emissionprob)

# We now simulate 100 timesteps
set.seed(12345)
simulation100 <- simHMM(HMM, 100)
# plot results
time <- 1:100
states <- as.numeric(simulation100$states)
observations <- as.numeric(simulation100$observation)
df_plot <- data.frame(time, states, observations)

plot_states <- ggplot(df_plot, aes(x = time, y = states))+
```

```r
  geom_line(col = "red")+
  ggtitle("Robot position")

plot_recorded <- ggplot(df_plot, aes(x = time, y = observations))+
  geom_line(col = "green")+
  ggtitle("Robot recorded position")

plot_states
plot_recorded
# filtered distribution
# alpha
alpha <- exp(forward(hmm = HMM, observation = simulation100$observation))
# beta
beta <- exp(backward(hmm = HMM, observation = simulation100$observation))
# filtered distribution
filter_dist <- prop.table(alpha, margin = 2)
# smoothed distribution
smooth_dist <- prop.table(alpha*beta, margin = 2)
# most probable path (viterbi algo)
viterbi_path <- viterbi(hmm = HMM, observation = simulation100$observation)

# accuracy filter
path_filter <- as.character(apply(filter_dist, 2, which.max))
filter_mat <- table(path_filter == simulation100$states)
accuracy_filter <- filter_mat[2]/sum(filter_mat)

# accuracy smoothed
path_smooth <- as.character(apply(smooth_dist, 2, which.max))
smooth_mat <- table(path_smooth == simulation100$states)
accuracy_smooth <- smooth_mat[2]/sum(smooth_mat)
# accuracy most probable
viterbi_mat <- table(viterbi_path == simulation100$states)
accuracy_viterbi <- viterbi_mat[2]/sum(viterbi_mat)
accuracy_filter
accuracy_smooth
accuracy_viterbi
# prepare data frame for experiment
exp_length <- 30
empt_matrix <- matrix(data = 0, nrow = exp_length, ncol = 3)
df_exp <- as.data.frame(empt_matrix)
colnames(df_exp) <- c("Filtered", "Smoothed", "Viterbi")

# experiment

for(i in 1:exp_length){
  #simulate data
  simul <- simHMM(HMM, length = 100)
  #get alpha, distribution and accuracy
  alpha <- exp(forward(hmm = HMM, observation = simul$observation))
  filterd <- prop.table(alpha, 2)
  path_filt <- as.character(apply(filterd, 2, which.max))
  filt_table <- table(path_filt == simul$states)
  filt_accuracy <- filt_table[2]/sum(filt_table)
```

```r
  #get beta and distribution and accuracy
  beta <- exp(backward(hmm = HMM, observation = simul$observation))
  smoothd <- prop.table(alpha*beta, 2)
  path_smooth <- as.character(apply(smoothd, 2, which.max))
  smooth_table <- table(path_smooth == simul$states)
  smooth_accuracy <- smooth_table[2]/sum(smooth_table)
  # viterbi path and accuracy
  vit_path <- viterbi(hmm = HMM, observation = simul$observation)
  vit_table <- table(vit_path == simul$states)
  vit_accuracy <- vit_table[2]/sum(vit_table)
  # record accuracy in data frame
  df_exp[i, ] <- cbind(filt_accuracy, smooth_accuracy, vit_accuracy)
}

#  print results
kable(df_exp, caption = "Accuracies for the different methods")

# entropy is calculated over each column using entropy.empirical
entropy <- apply(filter_dist, 2, entropy.empirical)
plot(entropy, type = "l")


# We simply have to multiply the last filter_distribution with the transition matrix
dist_101 <- filter_dist[, 100] %*% transprob
dist_101
```