732A96 Advanced Machine Learning

LAB 2: HIDDEN MARKOV MODELS

Johannes Hedström

STIMA
Department of Computer and Information Science
Linköpings universitet

2024-09-17

Contents

```r
library(HMM)
library(knitr)
```

Description

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.

1 Question 1

Build a hidden Markov model (HMM) for the scenario described above. Note that the documentation for the initHMM function says that the emissionProbs matrix should be of size [number of states]x[number of states]. This is wrong. It should be of size [number of states]x[number of symbols]. The package works correctly, though. It is just the documentation that is wrong.

```r
# States
states <- as.character(1:10)
# Movements for the robot
sym <- paste0('State_',states)
# starting probabilities, equal for all states
startprobs <- rep(1/10,10)

# Transitions probabilities 1/2 probability to stay or change

t_prob <- matrix(0,nrow=10,ncol=10) # empty matrix
diag(t_prob) <- 0.5  # 0.5 to stay
diag(t_prob[,-1]) <- 0.5 # 0.5 to move to NEXT sector
t_prob[10,1] <- 0.5 # left bottom corner(step from 10 to 1)

# emissionprobs
e_prob <- matrix(0,nrow=10, ncol=10) # empty matrix

for (i  in 1:10) { # filling the matrix with probabilities, each row sums to 1
  j <- (i-2):(i+2)
  if(11 %in% j){ # if we are going over 10 then go back to 1 and upwards

    j_1 <-j[1:which(j==10)]
    j <- c(1:(length(j) - length(j_1)),j_1)
    }
  if(0 %in% j){ # if we are getting a value under 1 then go from 10 and downwards
    j_1 <-j[which(j==1):length(j)]
    j <- c((10-(length(j) - length(j_1))+1):10,j_1)
```

```
  }
  e_prob[i,j] <- 1/5
}

hmm <-initHMM(states,sym, startprobs,transProbs=t_prob,e_prob)

rownames(t_prob) <- sym
colnames(t_prob) <- states
rownames(e_prob) <- sym
colnames(e_prob) <- states
knitr::kable(t_prob, caption='Transitions probabilities')
```

Table 1: Transitions probabilities

	1	2	3	4	5	6	7	8	9	10
State_1	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
State_2	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
State_3	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0
State_4	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0
State_5	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0
State_6	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0
State_7	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0
State_8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0
State_9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
State_10	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

```
knitr::kable(e_prob, caption='Emission probabilities[i - 2, i + 2]')
```

Table 2: Emission probabilities[i − 2, i + 2]

	1	2	3	4	5	6	7	8	9	10
State_1	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2
State_2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2
State_3	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0
State_4	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0
State_5	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0
State_6	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0
State_7	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0
State_8	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2
State_9	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2
State_10	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2

2 Question 2

Simulate the HMM for 100 time steps.

```r
sims <- simHMM(hmm,100) # simulate 100 steps

sim_state <- sims$states
sim_obs <- sims$observation

knitr::kable(sim_obs[1:10],caption='First ten simulated observations')
```

Table 3: First ten simulated observations

x
State_6
State_8
State_5
State_6
State_7
State_6
State_10
State_8
State_2
State_2

3 Question 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.

$$\text{Filtering: } p\left(z^t \mid x^{0:t}\right) = \frac{\alpha\left(z^t\right)}{\sum_{z^t} \alpha\left(z^t\right)}$$

$$\text{Smoothing: } p\left(z^t \mid x^{0:T}\right) = \frac{\alpha\left(z^t\right)\beta\left(z^t\right)}{\sum_{z^t} \alpha\left(z^t\right)\beta\left(z^t\right)}$$

```r
# probabilities are given on the logaritmic scale and must therefore be transformed
# with exp
# forward for filter
filt_pd <- exp(forward(hmm,sim_obs))

# backwards for smoothed
# alpha*beta/ sum over states alpha*beta
beta <- exp(backward(hmm, sim_obs))
smooth_pd <-  (filt_pd*beta)/colSums(filt_pd*beta)

# Viterbi algorithm for the most likely path
most_prob_path <- viterbi(hmm,sim_obs)
```

4 Question 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.

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 element wise 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

```
# picking out the highest probability from the filter pd
filt_path <- as.character(unname(apply(prop.table(filt_pd,2),2,which.max)))

# picking out the highest probability from the smoothed pd
smooth_path <-  as.character(unname(apply(prop.table(smooth_pd,2),2,which.max)))

# number of correct devided by total(100
acc_f <- sum(filt_path==sim_state)/100
acc_s <- sum(smooth_path==sim_state)/100
acc_mpp <- sum(most_prob_path==sim_state)/100

# dataframe with the accuracy
df_acc <- data.frame('Filter'=acc_f, 'Smoothed'=acc_s, 'Viterbi' = acc_mpp)

knitr::kable(df_acc, caption='the filtered and smoothed probability distributions ')
```

Table 4: the filtered and smoothed probability distributions

Filter	Smoothed	Viterbi
0.7	0.74	0.54

5 Question 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 ?

```
# a function to run simulation, filer, smooth etc
# returns a dataframe with the accuracy for each probabilitie distribution
# and the most probable path
hmm_sims <- function(nr_of_sims){

  sims <- simHMM(hmm,nr_of_sims) # simulate 100 steps
  sim_obs <- sims$observation
  sim_state <- sims$states
```

```r
  filt_pd <- exp(forward(hmm,sim_obs))
  # backwards for smoothed
  beta <- exp(backward(hmm, sim_obs))
  # alpha*beta/ sum over states alpha*beta
  smooth_pd <-  (filt_pd*beta)/colSums(filt_pd*beta)

  # Viterbi algorithm for the most likely path
  most_prob_path <- viterbi(hmm,sim_obs)

  # picking out the highest probability from the filter pd
  filt_path <- as.character(unname(apply(prop.table(filt_pd,2),2,which.max)))
  # picking out the highest probability from the smoothed pd
  smooth_path <-  as.character(unname(apply(prop.table(smooth_pd,2),2,which.max)))

  # number of correct devided by total(100
  acc_f <- sum(filt_path==sim_state)/100
  acc_s <- sum(smooth_path==sim_state)/100
  acc_mpp <- sum(most_prob_path==sim_state)/100

  # craeting a dataframe with each accuracy
  df_acc <- data.frame('Filter'=acc_f, 'Smoothed'=acc_s, 'Viterbi' = acc_mpp)
  df_acc # returning the dataframe
}

rownames(df_acc) <-paste0('Simulation: ',1)
# running the simulation 5 times and saving the results in the dataframe from before
for(i in 2:5){

  df_acc <- rbind(df_acc,hmm_sims(100))
  rownames(df_acc)[i] <-paste0('Simulation: ',i)
}

knitr::kable(df_acc, )
```

	Filter	Smoothed	Viterbi
Simulation: 1	0.70	0.74	0.54
Simulation: 2	0.49	0.62	0.43
Simulation: 3	0.52	0.70	0.57
Simulation: 4	0.57	0.65	0.46
Simulation: 5	0.54	0.68	0.49

6 Question 6

Is it always true that the later in time (i.e., the more observations you have received) 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.

7 Question 7

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