Advance Machine Learning Lab 2

Omkar Bhutra (omkbh878)

26 September 2019

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.

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

A robot moves around a ring which divided into 10 sectors. The robot is in one sector at any given time step and it's equally probable for the robot to stay in the state as it is to move to the next state. The robot has a tracking device. If the robot is in sector i, the tracking device will report that the robot is in the sectors [i - 2, i + 2] with equal probability i.e P = 0.2 for being in each position of that range.

Create transition matrix where each row consists of: $P(Z^t|Z^t-1), t = 1, ..., 10$

```
states <- paste("state",1:10,sep="")
symbols <- paste("symbol",1:10,sep="")
transition_vector <- 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)

emission_vector <- 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)


transition_matrix <- matrix(data = transition_vector,nrow = 10,ncol = 10)

emission_matrix <- matrix(data = emission_vector,nrow = 10,ncol = 10)
# States are the hidden variables
# Symbols are the observable variables
hmm <- initHMM(States = states,
```

```
        Symbols = symbols,
        startProbs = rep(0.1,10),
        transProbs = transition_matrix,
        emissionProbs = emission_matrix)
hmm
```

```
## $States
##  [1] "state1"  "state2"  "state3"  "state4"  "state5"  "state6"  "state7"
##  [8] "state8"  "state9"  "state10"
##
## $Symbols
##  [1] "symbol1"  "symbol2"  "symbol3"  "symbol4"  "symbol5"  "symbol6"
##  [7] "symbol7"  "symbol8"  "symbol9"  "symbol10"
##
## $startProbs
##  state1  state2  state3  state4  state5  state6  state7  state8  state9
##     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
## state10
##     0.1
##
## $transProbs
##          to
## from      state1 state2 state3 state4 state5 state6 state7 state8 state9
##   state1     0.5    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
##   state2     0.5    0.5    0.0    0.0    0.0    0.0    0.0    0.0    0.0
##   state3     0.0    0.5    0.5    0.0    0.0    0.0    0.0    0.0    0.0
##   state4     0.0    0.0    0.5    0.5    0.0    0.0    0.0    0.0    0.0
##   state5     0.0    0.0    0.0    0.5    0.5    0.0    0.0    0.0    0.0
##   state6     0.0    0.0    0.0    0.0    0.5    0.5    0.0    0.0    0.0
##   state7     0.0    0.0    0.0    0.0    0.0    0.5    0.5    0.0    0.0
##   state8     0.0    0.0    0.0    0.0    0.0    0.0    0.5    0.5    0.0
##   state9     0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.5    0.5
##   state10    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.5
##          to
## from      state10
##   state1      0.5
##   state2      0.0
##   state3      0.0
##   state4      0.0
##   state5      0.0
##   state6      0.0
##   state7      0.0
##   state8      0.0
##   state9      0.0
##   state10     0.5
##
## $emissionProbs
##          symbols
## states    symbol1 symbol2 symbol3 symbol4 symbol5 symbol6 symbol7 symbol8
##   state1      0.2     0.2     0.2     0.0     0.0     0.0     0.0     0.0
##   state2      0.2     0.2     0.2     0.2     0.0     0.0     0.0     0.0
##   state3      0.2     0.2     0.2     0.2     0.2     0.0     0.0     0.0
##   state4      0.0     0.2     0.2     0.2     0.2     0.2     0.0     0.0
##   state5      0.0     0.0     0.2     0.2     0.2     0.2     0.2     0.0
```

```
##   state6      0.0     0.0     0.0     0.2     0.2     0.2     0.2     0.2
##   state7      0.0     0.0     0.0     0.0     0.2     0.2     0.2     0.2
##   state8      0.0     0.0     0.0     0.0     0.0     0.2     0.2     0.2
##   state9      0.2     0.0     0.0     0.0     0.0     0.0     0.2     0.2
##   state10     0.2     0.2     0.0     0.0     0.0     0.0     0.0     0.2
##          symbols
## states    symbol9 symbol10
##   state1      0.2      0.2
##   state2      0.0      0.2
##   state3      0.0      0.0
##   state4      0.0      0.0
##   state5      0.0      0.0
##   state6      0.0      0.0
##   state7      0.2      0.0
##   state8      0.2      0.2
##   state9      0.2      0.2
##   state10     0.2      0.2
```

Question 2: Simulate the HMM for 100 time steps.

```
hmm_sim <- simHMM(hmm = hmm,length = 100)
hmm_sim
```

```
## $states
##   [1] "state5"  "state5"  "state4"  "state3"  "state3"  "state3"  "state3"
##   [8] "state2"  "state2"  "state2"  "state1"  "state10" "state10" "state10"
##  [15] "state9"  "state8"  "state8"  "state7"  "state7"  "state7"  "state6"
##  [22] "state6"  "state6"  "state5"  "state4"  "state4"  "state4"  "state3"
##  [29] "state3"  "state3"  "state3"  "state3"  "state2"  "state1"  "state1"
##  [36] "state1"  "state1"  "state1"  "state10" "state9"  "state8"  "state8"
##  [43] "state8"  "state8"  "state8"  "state7"  "state6"  "state5"  "state5"
##  [50] "state4"  "state3"  "state2"  "state1"  "state10" "state10" "state9"
##  [57] "state8"  "state7"  "state7"  "state7"  "state6"  "state6"  "state6"
##  [64] "state5"  "state5"  "state4"  "state3"  "state2"  "state2"  "state1"
##  [71] "state10" "state10" "state9"  "state9"  "state9"  "state8"  "state8"
##  [78] "state8"  "state8"  "state8"  "state7"  "state7"  "state7"  "state6"
##  [85] "state6"  "state6"  "state6"  "state5"  "state5"  "state5"  "state4"
##  [92] "state3"  "state2"  "state2"  "state2"  "state2"  "state2"  "state2"
##  [99] "state2"  "state2"
##
## $observation
##   [1] "symbol7"  "symbol5"  "symbol2"  "symbol2"  "symbol1"  "symbol1"
##   [7] "symbol1"  "symbol2"  "symbol4"  "symbol2"  "symbol10" "symbol10"
##  [13] "symbol1"  "symbol9"  "symbol1"  "symbol10" "symbol7"  "symbol8"
##  [19] "symbol8"  "symbol5"  "symbol8"  "symbol7"  "symbol4"  "symbol4"
##  [25] "symbol2"  "symbol4"  "symbol2"  "symbol4"  "symbol1"  "symbol1"
##  [31] "symbol5"  "symbol1"  "symbol3"  "symbol3"  "symbol3"  "symbol9"
##  [37] "symbol10" "symbol2"  "symbol2"  "symbol1"  "symbol9"  "symbol7"
##  [43] "symbol10" "symbol8"  "symbol10" "symbol8"  "symbol7"  "symbol5"
##  [49] "symbol6"  "symbol4"  "symbol4"  "symbol2"  "symbol2"  "symbol8"
##  [55] "symbol1"  "symbol1"  "symbol7"  "symbol8"  "symbol6"  "symbol8"
##  [61] "symbol4"  "symbol6"  "symbol6"  "symbol5"  "symbol7"  "symbol6"
```

```
##  [67] "symbol1"  "symbol10" "symbol1"  "symbol3"  "symbol9"  "symbol9"
##  [73] "symbol1"  "symbol10" "symbol9"  "symbol10" "symbol6"  "symbol8"
##  [79] "symbol9"  "symbol10" "symbol8"  "symbol9"  "symbol8"  "symbol7"
##  [85] "symbol8"  "symbol4"  "symbol6"  "symbol3"  "symbol3"  "symbol4"
##  [91] "symbol5"  "symbol2"  "symbol10" "symbol4"  "symbol10" "symbol4"
##  [97] "symbol4"  "symbol2"  "symbol4"  "symbol2"
```

Question 3: Discard the hidden states from the sample obtained above. Use the remaining observationsto compute the filtered and smoothed probability distributions for each of the 100 time points. Compute also the most probable path. Question 4: Compute the accuracy of the filtered and smoothed probability distributions, and of themost 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.

Forward probabilities can be found in a hidden markov model with hidden states X upto observation at time t defined as the probability of observing the sequence of observations $e_1, ..., e_k$ and that the state at time t is X. That is: $f[X,t] := P(X|E_1 = e_1, ..., E_k = e_k)$

```
robot <- function(hiddenmarkovmodel,pars){
  hmm_sim <- simHMM(hmm = hmm,length = 100)
  hmm_obs <- hmm_sim$observation
  hmm_states <- hmm_sim$states
  #filter: forward function does the filtering and returns log probabilities
  log_filter = forward(hmm = hmm,observation = hmm_obs)
  filter = exp(log_filter)
  #normalised probability distribution
  filternormalised <- prop.table(filter,margin = 2)
  # find out the most probable states
  filternormalised_probable <- apply(filternormalised,MARGIN = 2, FUN = which.max)
  accuracy_filtering <- sum(paste("state", filternormalised_probable, sep ="")
                            ==hmm_states)/length(hmm_states)
  #filternormalised
  #accuracy_filtering
  #smoothing using function posterior
  smooth <- posterior(hmm,hmm_obs)
  smoothnormalised <- prop.table(smooth, margin = 2)
  smoothnormalised_probable <- apply(smoothnormalised,MARGIN = 2, FUN = which.max)
  accuracy_smooth <- sum(paste("state", smoothnormalised_probable, sep ="")
                            ==hmm_states)/length(hmm_states)
  #smoothnormalised
  #accuracy_smooth
  #Finding the most probable path using viterbi algorithm
  probable_path <- viterbi(hmm = hmm, observation = hmm_obs)
  accuracy_probable_path <- sum(probable_path == hmm_states)/ length(hmm_states)
  probable_path #most probable path
  #accuracy_probable_path
  if(pars == "filter"){
  return(filternormalised)
}
```

```
if(pars == "smooth"){
  return(smoothnormalised)
}
if(pars == "ProbablePath"){
  return(probable_path)
}
if(pars == "accuracy"){
  return(c(accuracy_filtering = accuracy_filtering,
           accuracy_smooth = accuracy_smooth,
           accuracy_probable_path = accuracy_probable_path))
}
}
```

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 ?

```
robot(hmm,"ProbablePath")
```

```
##   [1] "state1"  "state10" "state9"  "state8"  "state8"  "state8"  "state8"
##   [8] "state8"  "state8"  "state8"  "state8"  "state7"  "state6"  "state6"
##  [15] "state6"  "state6"  "state5"  "state5"  "state5"  "state4"  "state3"
##  [22] "state3"  "state3"  "state2"  "state1"  "state10" "state9"  "state9"
##  [29] "state8"  "state8"  "state7"  "state6"  "state6"  "state5"  "state5"
##  [36] "state5"  "state4"  "state3"  "state3"  "state3"  "state3"  "state3"
##  [43] "state2"  "state1"  "state1"  "state1"  "state1"  "state1"  "state10"
##  [50] "state9"  "state8"  "state7"  "state6"  "state5"  "state4"  "state4"
##  [57] "state4"  "state3"  "state2"  "state1"  "state1"  "state10" "state9"
##  [64] "state8"  "state8"  "state8"  "state8"  "state8"  "state7"  "state7"
##  [71] "state6"  "state6"  "state5"  "state5"  "state4"  "state3"  "state2"
##  [78] "state1"  "state10" "state9"  "state9"  "state8"  "state7"  "state6"
##  [85] "state5"  "state5"  "state4"  "state4"  "state4"  "state4"  "state3"
##  [92] "state2"  "state1"  "state10" "state9"  "state9"  "state9"  "state8"
##  [99] "state8"  "state8"
```

```
acc <- sapply(1:100,FUN = function(x){robot(hmm,"accuracy")})
acc <- data.frame(t(acc))
acc$number <- 1:100

ggplot(data = acc) + geom_line(aes(x=number,y=accuracy_filtering,col="filtered"))+
                     geom_line(aes(x=number,y=accuracy_smooth,col="smoothened"))+
                     geom_line(aes(x=number,y=accuracy_probable_path,col="Probable Path"))
```


```
colMeans(acc[,-4])
```

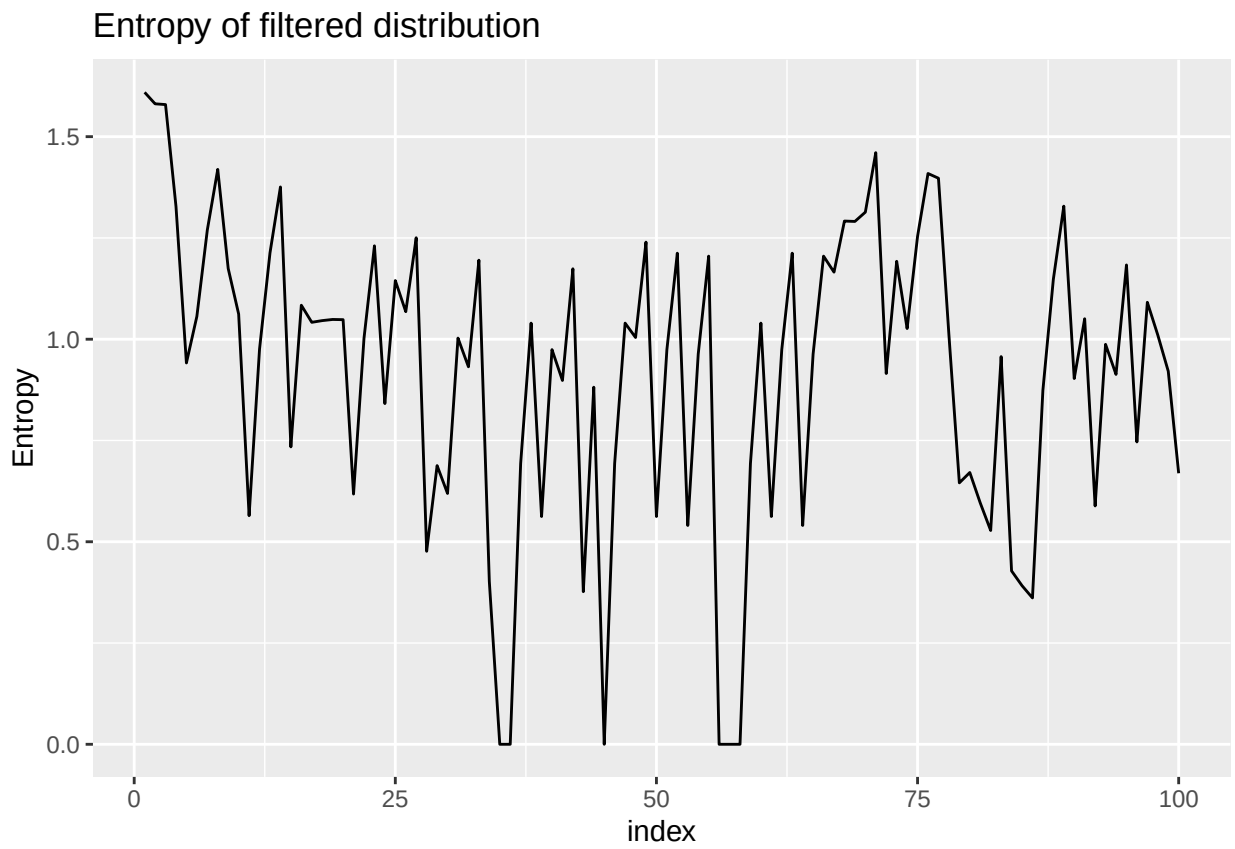
```
##     accuracy_filtering        accuracy_smooth accuracy_probable_path
##                 0.5246                 0.6736                 0.4499
```

The smoothed distribution is using more information i.e the whole set of observed emission x[0:T] whereas filtered only uses data emitted up to that point x[0:t] The most probable path generated by the viterbi algorithm is constrained by the transitions between states in the hidden variables. The smooth distribution approximates the most probable state and hence can make jumps from one state to another when predicting current state.

Question 6: 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.

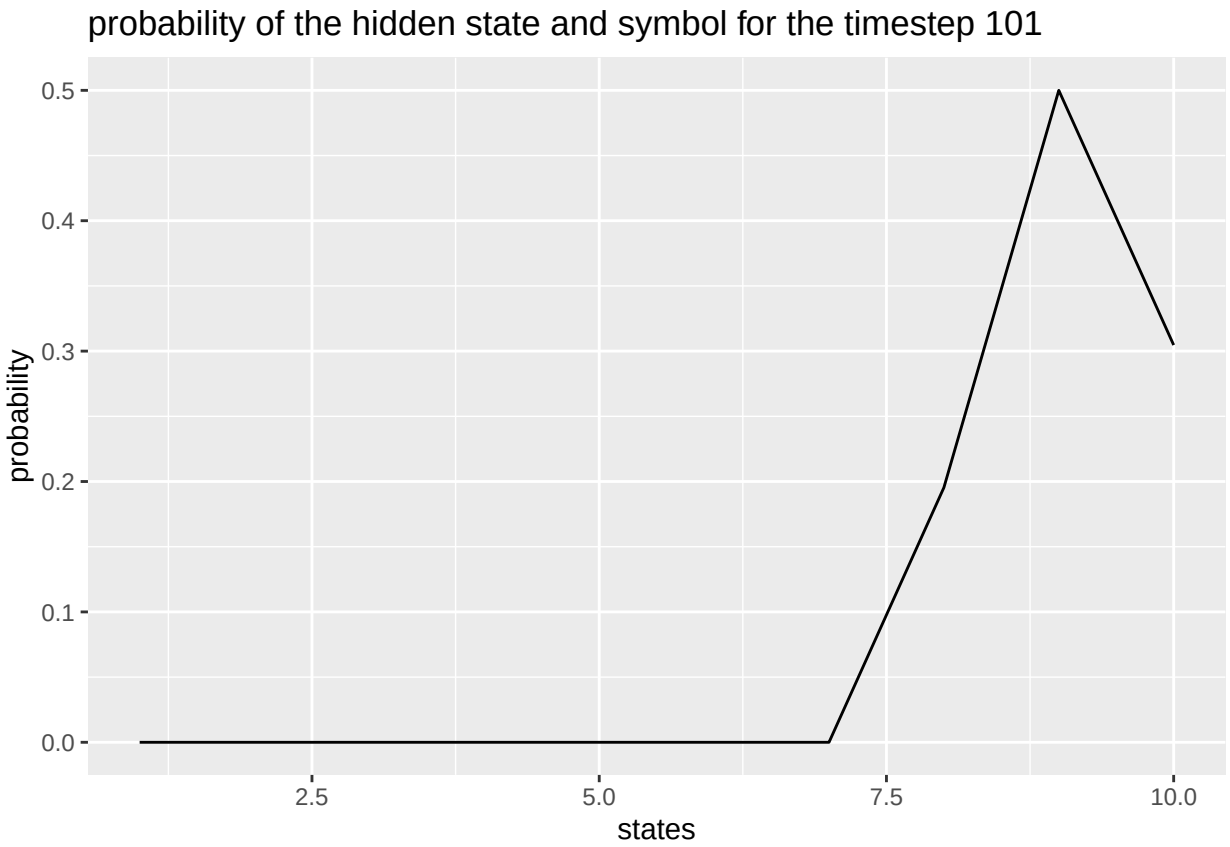
```
hmm_filter <- robot(hmm,"filter")
hmm_filter_entropy <- data.frame(index = 1:100, Entropy = apply(hmm_filter, MARGIN = 2,
                                 FUN = entropy::entropy.empirical))
ggplot(hmm_filter_entropy,aes(x = index, y = Entropy))+
  geom_line()+ggtitle("Entropy of filtered distribution")
```



The entropy is random even when we increase the number of observations added to the hidden markov model. This is because the HMM is Markovian and only depends on the previous observation.

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

```
posterior <- hmm_filter[,100]
# matrix multiplication
probability <- as.data.frame(hmm$transProbs %*% posterior)
ggplot(probability)+geom_line(aes(x=1:10,y=V1))+ggtitle("probability of the hidden state and symbol for
```



The Markovian assumption is that only relavant state is the current state. All previous states feeds though the 100 states and provides us information about the states and observations. On multiplication with transition probabilities to get the probability of the hidden state and symbol for the timestep 101. That the entropy of the filtered distributions does not decrease monotonically. It has to do with the fact that you get noisy observations and, thus, you will more often than not be uncertain as to where the robot is.

```
knitr::opts_chunk$set(echo = TRUE)
library("dplyr")
library("ggplot2")
library("gRain")
library("bnlearn")
library("HMM")
library("entropy")
states <- paste("state",1:10,sep="")
symbols <- paste("symbol",1:10,sep="")
transition_vector <- 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)
```

```
emission_vector <- 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)


transition_matrix <- matrix(data = transition_vector,nrow = 10,ncol = 10)

emission_matrix <- matrix(data = emission_vector,nrow = 10,ncol = 10)
# States are the hidden variables
# Symbols are the observable variables
hmm <- initHMM(States = states,
        Symbols = symbols,
        startProbs = rep(0.1,10),
        transProbs = transition_matrix,
        emissionProbs = emission_matrix)
hmm
hmm_sim <- simHMM(hmm = hmm,length = 100)
hmm_sim
robot <- function(hiddenmarkovmodel,pars){
  hmm_sim <- simHMM(hmm = hmm,length = 100)
  hmm_obs <- hmm_sim$observation
  hmm_states <- hmm_sim$states
  #filter: forward function does the filtering and returns log probabilities
  log_filter = forward(hmm = hmm,observation = hmm_obs)
  filter = exp(log_filter)
  #normalised probability distribution
  filternormalised <- prop.table(filter,margin = 2)
  # find out the most probable states
  filternormalised_probable <- apply(filternormalised,MARGIN = 2, FUN = which.max)
  accuracy_filtering <- sum(paste("state", filternormalised_probable, sep ="")
                            ==hmm_states)/length(hmm_states)
  #filternormalised
  #accuracy_filtering
  #smoothing using function posterior
  smooth <- posterior(hmm,hmm_obs)
  smoothnormalised <- prop.table(smooth, margin = 2)
  smoothnormalised_probable <- apply(smoothnormalised,MARGIN = 2, FUN = which.max)
  accuracy_smooth <- sum(paste("state", smoothnormalised_probable, sep ="")
                            ==hmm_states)/length(hmm_states)
  #smoothnormalised
  #accuracy_smooth
  #Finding the most probable path using viterbi algorithm
  probable_path <- viterbi(hmm = hmm, observation = hmm_obs)
  accuracy_probable_path <- sum(probable_path == hmm_states)/ length(hmm_states)
  probable_path #most probable path
  #accuracy_probable_path
```

```
  if(pars == "filter"){
  return(filternormalised)
}
if(pars == "smooth"){
  return(smoothnormalised)
}
if(pars == "ProbablePath"){
  return(probable_path)
}
if(pars == "accuracy"){
  return(c(accuracy_filtering = accuracy_filtering,
           accuracy_smooth = accuracy_smooth,
           accuracy_probable_path = accuracy_probable_path))
}
}
robot(hmm,"filter")
robot(hmm,"smooth")
robot(hmm,"ProbablePath")

acc <- sapply(1:100,FUN = function(x){robot(hmm,"accuracy")})
acc <- data.frame(t(acc))
acc$number <- 1:100

ggplot(data = acc) + geom_line(aes(x=number,y=accuracy_filtering,col="filtered"))+
                     geom_line(aes(x=number,y=accuracy_smooth,col="smoothened"))+
                     geom_line(aes(x=number,y=accuracy_probable_path,col="Probable Path"))

colMeans(acc[,-4])
hmm_filter <- robot(hmm,"filter")
hmm_filter_entropy <- data.frame(index = 1:100, Entropy = apply(hmm_filter, MARGIN = 2,
                                 FUN = entropy::entropy.empirical))
ggplot(hmm_filter_entropy,aes(x = index, y = Entropy))+
  geom_line()+ggtitle("Entropy of filtered distribution")
posterior <- hmm_filter[,100]
# matrix multiplication
probability <- as.data.frame(hmm$transProbs %*% posterior)
ggplot(probability)+geom_line(aes(x=1:10,y=V1))+ggtitle("probability of the hidden state and symbol for
# Build the HMM.
library(HMM)
#set.seed(123)
States<-1:100
Symbols<-1:2 # 1=door

transProbs<-matrix(rep(0,length(States)*length(States)), nrow=length(States), ncol=length(States), byrow
for(i in 1:99){
  transProbs[i,i]<-.1
  transProbs[i,i+1]<-.9
}

emissionProbs<-matrix(rep(0,length(States)*length(Symbols)), nrow=length(States), ncol=length(Symbols),
for(i in States){
  if(i %in% c(10,11,12,20,21,22,30,31,32)){
    emissionProbs[i,1]<-.9
```

```
    emissionProbs[i,2]<-.1
  }
  else{
    emissionProbs[i,1]<-.1
    emissionProbs[i,2]<-.9
  }
}

startProbs<-rep(1/100,100)
hmm<-initHMM(States,Symbols,startProbs,transProbs,emissionProbs)

# If the robot observes a door, it can be in front of any of the three doors. If it then observes a lon
# sequence of non-doors, then it know that it was in front of the third door.

obs<-c(1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2)
pt<-prop.table(exp(forward(hmm,obs)),2)

which.maxima<-function(x){ # This function is needed since which.max only returns the first maximum.
  return(which(x==max(x)))
}

apply(pt,2,which.maxima)
```