Advance Machine Learning Lab 2 Group lab 2

Omkar Bhutra (omkbh878), Obaid Ur Rehman (obaur539) ,Ruben Jared Muñoz Roquet (rubmu773)

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: $P(Z^t|Z^t - 1), t = 1, ..., 10$

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

startProbs = rep(1/10,10)
transProbs = emisProbs = matrix( nrow = 10, ncol = 10)
for(i in 1:10){

  ## For transition prpob
  rem = i %% 10
  transProbs[i,] = 0
  transProbs[i,i] = transProbs[i,rem+1] = 0.5

  ## For emission prob
  emisProbs[i,] = 0

  ##two forward
  f1 = (i+1)%%10
  f2 = (i+2)%%10

  #two backward
  b1 = (i+10-1)%%10
  b2 = (i+10-2)%%10

  ind = c(i,f1,f2,b1,b2)
  ind = replace(ind, ind==0, 10)
  emisProbs[i,ind]=0.2
}
hmm = initHMM(States = states,Symbols = symbols,transProbs = transProbs,emissionProbs = emisProbs)
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    0.5    0.0    0.0    0.0    0.0    0.0    0.0    0.0
##   state2     0.0    0.5    0.5    0.0    0.0    0.0    0.0    0.0    0.0
##   state3     0.0    0.0    0.5    0.5    0.0    0.0    0.0    0.0    0.0
##   state4     0.0    0.0    0.0    0.5    0.5    0.0    0.0    0.0    0.0
##   state5     0.0    0.0    0.0    0.0    0.5    0.5    0.0    0.0    0.0
##   state6     0.0    0.0    0.0    0.0    0.0    0.5    0.5    0.0    0.0
##   state7     0.0    0.0    0.0    0.0    0.0    0.0    0.5    0.5    0.0
##   state8     0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.5    0.5
##   state9     0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.5
##   state10    0.5    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
##          to
## from      state10
##   state1      0.0
##   state2      0.0
##   state3      0.0
##   state4      0.0
##   state5      0.0
##   state6      0.0
##   state7      0.0
##   state8      0.0
##   state9      0.5
##   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.

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

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

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: $f[X, t] := P(X|E_1 = e_1, ..., E_k = e_k)$

```r
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))
}
}
```

Let, HMMs are one of the most popular graphical models in real use (indeed, probably the most popular). They are characterized by variables X1,X2,..,Xn representing hidden states and variables E1,E2,..,En representing observations (i.e., evidence). The subscript i in Xi and Ei represents a discrete slice of time.

Given a series of observations, we want to determine the distribution over states at some time stamp. Concretely, we want to determine $P(X_t|E_1, E_2, .., E_n)$. The task is called filtering if $t = n$, smoothing if $t < n$, and predicting if $t > n$. Clearly, smoothing will give better estimates, and prediction the weakest (or most uncertain) estimates.

For most probable path, This is another common inference task. Given a series of observations, the Viterbi algorithm helps us to determine the most likely sequence of states the system went to produce those observations.

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 ?

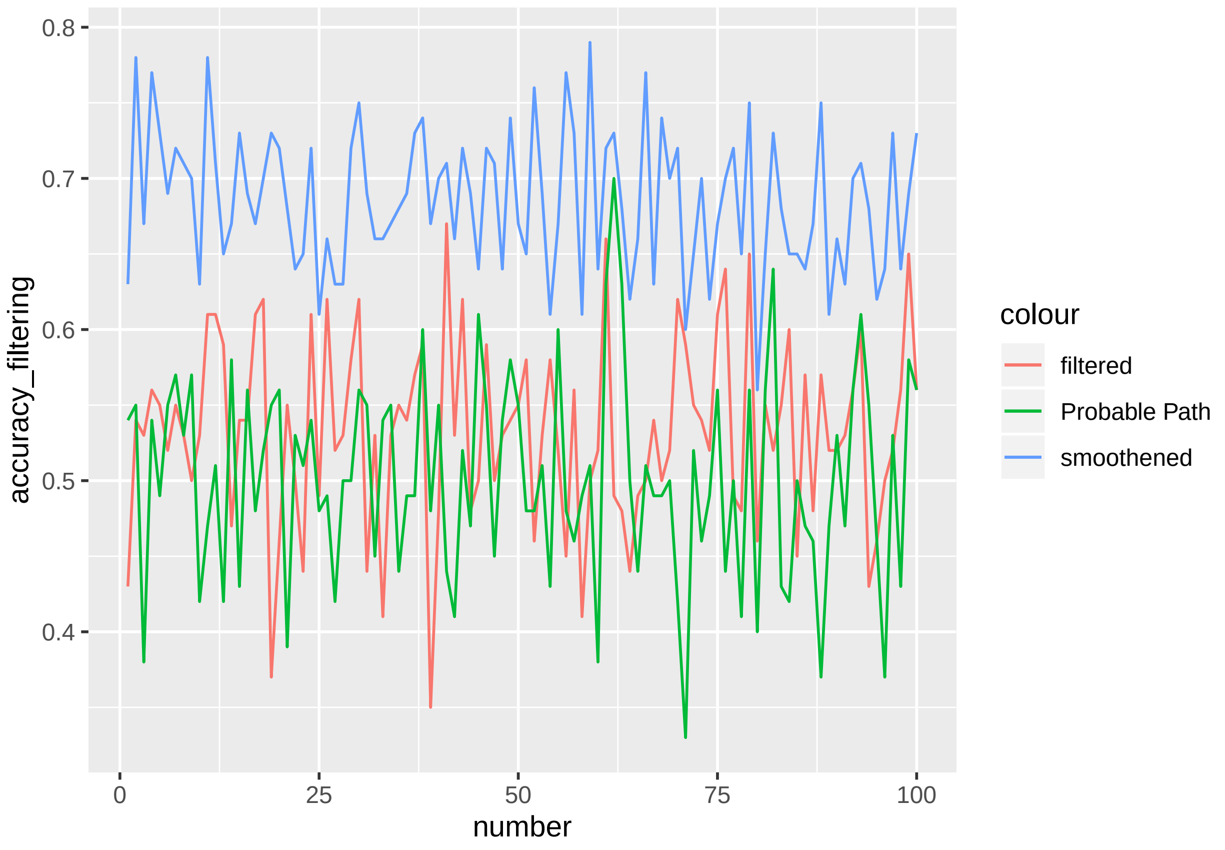
Smoothed distribution is better than the filtered distribution and also better than the most probable path.

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

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

```r
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
##                 0.5320                 0.6853                 0.5017
```

5a) Sample 1: length of 150

```
simHmm2 <- simHMM(hmm, 150)

obs = simHmm2$observation
logForward = forward(hmm,obs)
forwardProbs = exp(logForward)
forwardProbs <- prop.table(forwardProbs,margin = 2)
indFor <- apply(forwardProbs,FUN=which.max, MARGIN = 2)
filteredPath <- states[indFor]
confMat = table(simHmm2$states,filteredPath)
n = sum(confMat)
acc = sum(diag(confMat))/n
paste('Accuracy for filtered distribution:',round(acc,2)*100,'%')
```

```
## [1] "Accuracy for filtered distribution: 52 %"
```

```r
#--------------------------------------------------------------
smoothProbs = posterior(hmm,obs)
smoothProbs <- prop.table(smoothProbs,margin = 2)
indsmooth <- apply(smoothProbs,FUN=which.max, MARGIN = 2)
smoothProbs <- states[indsmooth]
confMat = table(simHmm2$states,smoothProbs)
n = sum(confMat)
acc = sum(diag(confMat))/n
paste('Accuracy for smoothed distribution:',round(acc,2)*100,'%')
```

```
## [1] "Accuracy for smoothed distribution: 69 %"
```

5b) Sample 1: length of 200

```r
simHmm3 <- simHMM(hmm, 200)

obs = simHmm3$observation
logForward = forward(hmm,obs)
forwardProbs = exp(logForward)
forwardProbs <- prop.table(forwardProbs,margin = 2)
indFor <- apply(forwardProbs,FUN=which.max, MARGIN = 2)
filteredPath <- states[indFor]
confMat = table(simHmm3$states,filteredPath)
n = sum(confMat)
acc = sum(diag(confMat))/n
paste('Accuracy for filtered distribution:',acc*100,'%')
```

```
## [1] "Accuracy for filtered distribution: 49 %"
```

```r
#--------------------------------------------------------------

smoothProbs = posterior(hmm,obs)
smoothProbs <- prop.table(smoothProbs,margin = 2)
indsmooth <- apply(smoothProbs,FUN=which.max, MARGIN = 2)
smoothProbs <- states[indsmooth]
confMat = table(simHmm3$states,smoothProbs)
n = sum(confMat)
acc = sum(diag(confMat))/n
paste('Accuracy for smoothed distribution:',acc*100,'%')
```

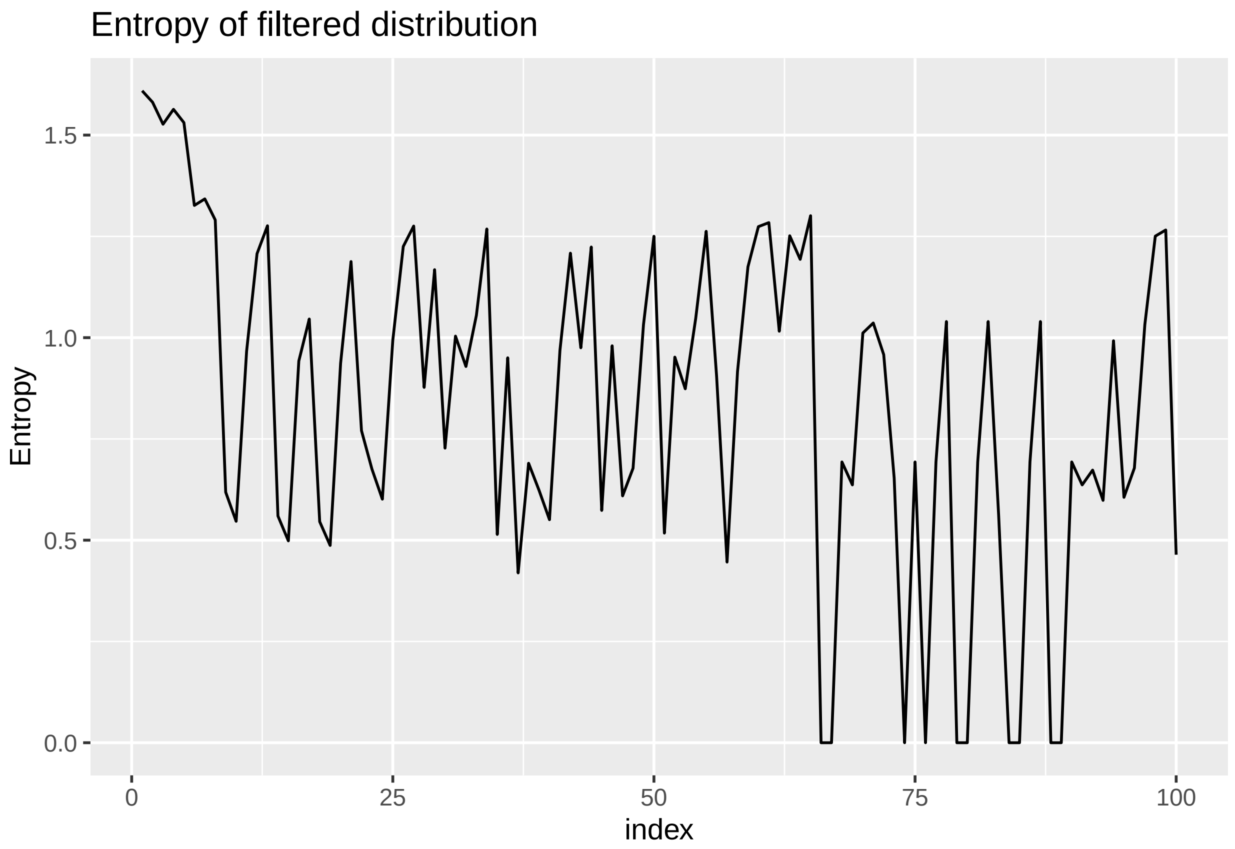
```
## [1] "Accuracy for smoothed distribution: 59.5 %"
```

The smoothed distibution has higher accuracy than filtered and vertebi because it uses all of the past, present and future data. And filtered distribution uses only the past data and vertebi (most probable path) imposes the contraint that the path should be valid which might result in decreased accuracy.

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.

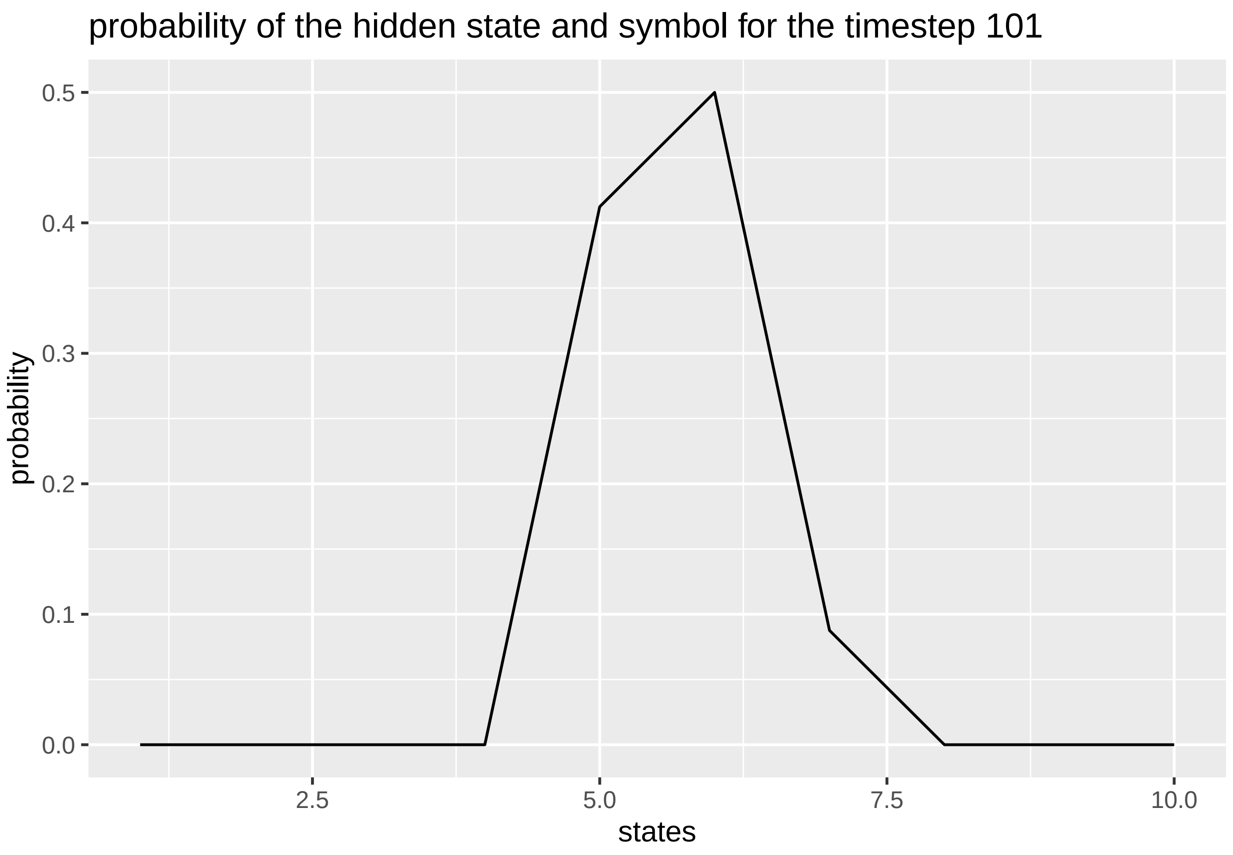
```
hmm_filter <- robot(hmm,"filter")
hmm_filter_entropy <- data.frame(index = 1:100, Entropy = apply(hmm_filter, MARGIN = 2,
                                 FUN = entropy::entropy.empirical))
ggplot(hmm_filter_entropy,aes(x = index, y = Entropy))+
  geom_line()+ggtitle("Entropy of filtered distribution")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



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
```


probability of the hidden state and symbol for the timestep 101
probability
states
0.0
0.1
0.2
0.3
0.4
0.5
2.5
5.0
7.5
10.0