Hidden Markov Lab2

Aman Kumar Nayak

9/20/2020

Task

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.

To build HMM, we have to specify emission and transition probability, where transition probability specify probability with which robot move from state i to state j. And Emission probability specify probability of state given observation.

```
#Building Hidden Markov Model
library(HMM)

States =  1:10
Symbols = 1:10

#Since
#the robot is in one of the sectors and decides with equal probability to stay in that sector or move t
#we get probability of staying as 0.5 and probability of moving as 0.5

# transProbs = 0.5*diag(10)
# for(i in 1:ncol(transProbs))
# {
#     if(i != ncol(transProbs))
#     {
#       transProbs[i+1 , i]  = 0.5
#     }else{
#       transProbs[1 , i]  = 0.5
#     }
# }

transProbs = 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
               )

transProbs = matrix(data =  transProbs , byrow =  TRUE , nrow = 10 , ncol = 10)

colnames(transProbs) = States
rownames(transProbs) = States


#emissionProbs
#If the robot is in the sector i, then the device will report that the robot is in the sectors [i - 2,
#so robot can be in any sector i with probability of 0.2 (5/10)

emissionProbs = 0.2 * diag(10)
n = ncol(emissionProbs)
for(i in 1:n){
  if(i-1 == 0) {
    Col_Indx = c((n-1) , n , i , i+1 , i+2)
  }else if(i-1 ==  1){
    Col_Indx = c( n , i-1 ,i , i+1 , i+2)
  }else if(i+1 == n){
    Col_Indx = c( i-2 ,i-1 ,i , i+1 , 1)
  }else if(i+1 > n) {
    Col_Indx = c( i-2 ,i-1 ,i , 1 , 2)
  }else{
    Col_Indx = c( i-2 , i-1 ,i , i+1 , i+2)
  }
  emissionProbs[i , Col_Indx] = 0.2
}#for

hmm = HMM::initHMM(States , Symbols , transProbs = transProbs , emissionProbs = emissionProbs)


## Hidden Markov Model for given scenario is

## $States
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $Symbols
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $startProbs
##   1   2   3   4   5   6   7   8   9  10
## 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##
## $transProbs
##      to
## from   1   2   3   4   5   6   7   8   9  10
##   1  0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   2  0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   3  0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0
##   4  0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0
```

```
##   5  0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0
##   6  0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0
##   7  0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0
##   8  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0
##   9  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5
##   10 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5
##
## $emissionProbs
##       symbols
## states   1   2   3   4   5   6   7   8   9  10
##     1  0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2
##     2  0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2
##     3  0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0
##     4  0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0
##     5  0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0
##     6  0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
##     7  0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0
##     8  0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2
##     9  0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2
##     10 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2
```

Question 2

Simulate the HMM for 100 time steps.

```
suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
nSim = 100
simulateHmm = simHMM(hmm , nSim)
```

```
## Simulated Values

## $states
##   [1]  9  9  9  9 10  1  2  2  2  2  3  3  4  4  4  4  4  4  4  5  6  6  7  8  9
##  [26] 10 10 10  1  2  2  3  3  4  4  4  5  5  5  6  7  7  8  9 10  1  2  3  3  4
##  [51]  5  5  6  6  7  7  8  8  8  8  9 10 10 10 10  1  1  2  2  2  2  2  3  3  3
##  [76]  4  5  5  5  6  7  8  8  8  8  8  9  9  9 10 10 10  1  1  1  1  1  1  2  3
##
## $observation
##   [1]  7 10  8 10  2  3 10  3  4  4  5  4  2  3  2  6  6  5  4  3  5  5  8  9 10
##  [26]  9  9 10  2 10  2  5  3  2  6  6  4  7  7  6  5  9  7 10 10  3  3  1  3  3
##  [51]  6  5  4  7  7  9  9 10  6  9 10  2  9  9  8  1  3  2  3  4  3  2  5  4  4
##  [76]  2  4  6  4  6  8 10  8  7  6  6  7  8  9 10  1  9  2  2  3  9  2 10  4  1
```

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.

Filtered and Smoothed Probability

Now in order to calculate filtered probability, I am using forward(hmm, observation)

In order to calculate, smoothed probabilities, we can use posterior(hmm, observation) where The posterior probability of being in a state X at time k can be computed from the forward and backward probabilities.

Now most probable path is calculated considering that robot can move from State X to State X $\pm$ 1 , so it can only move to next/previous stage and not like 2 or 3 stages ahead / behind.

```
#Since we have received probabilities in log, taking anti-log
filterdProb = exp(HMM::forward(hmm , simulateHmm$observation))

smoothedProb = HMM::posterior(hmm , simulateHmm$observation)

probablePath = HMM::viterbi(hmm, simulateHmm$observation)
```

Calculating Accuracy

```
fnAccuracy = function(prob , Margin = 2 , States){
  #prob = probability of filter or smoother
  #Margin = margin = 2
  #States : States obtained during simulation

  #normalized prob
  normalisedProb = prop.table(prob , margin = Margin)

  #predicted states
  PredStates = apply(normalisedProb, MARGIN = Margin, which.max)

  #Calculate Accuracy in percentage when compared with  States obtained during simulation
  percAcc = sum(PredStates == States) / length(States)

  return(percAcc)
}
```

Table 1: Accuracy Table

Filter	Smoothing	Probable.Path
0.53	0.74	0.18

Question 5

Generating Different Sample

```
suppressWarnings(RNGversion("3.5.1"))
set.seed(9999789)
nSim = 100
simulateHmm2 = simHMM(hmm , nSim)
```

```
filterdProb2 = exp(HMM::forward(hmm , simulateHmm2$observation))

smoothedProb2 = HMM::posterior(hmm , simulateHmm2$observation)
probablePath2 = HMM::viterbi(hmm, simulateHmm2$observation)

filterAccuracy2 = fnAccuracy(prob = filterdProb2 , Margin = 2 , States = simulateHmm2$states)
smoothAccuracy2 = fnAccuracy(prob = smoothedProb2 , Margin = 2 , States = simulateHmm2$states)
probPathAcc2 = sum(probablePath2 == simulateHmm2$observation)/length(simulateHmm2$observation)
```

Table 2: Accuracy Table with Different Sample

Filter	Smoothing	Probable.Path
0.66	0.73	0.18

Smoothing Vs Filtering

Now if we have observation $x_1, x_2, ...x_f, ...x_n$, and we apply filtering to it for predicting state Z_f, filtering distribution only uses observations till $x_1, x_2, ...x_f$ while smoothing distribution uses complete observations $x_1, x_2, ...x_f, ...x_n$, i.e. past and future values thus it is less influenced by sudden changes. Thus it is more accurate than filtered one.

Now it can be seen that, smoothed distribution is more accurate than filters distribution as filtered distribution is influenced only uses data till smoothing, because it have access to both past and future values thus it is less vulnerable to sudden changes and gives more accurate results.

Smoothing Vs Most Probable Path

Now in order to calculate most probable path, we are calculating we are using viterbi algorithm where we are calculating joint probability of $p(Z_{1:n}, x_{1:n})$ thus we are trying to calculate maximal probability of all events happening together which is always going to be less than marginal probability calculated for smoothing, thus Most probable path is less accurate.

Question 6

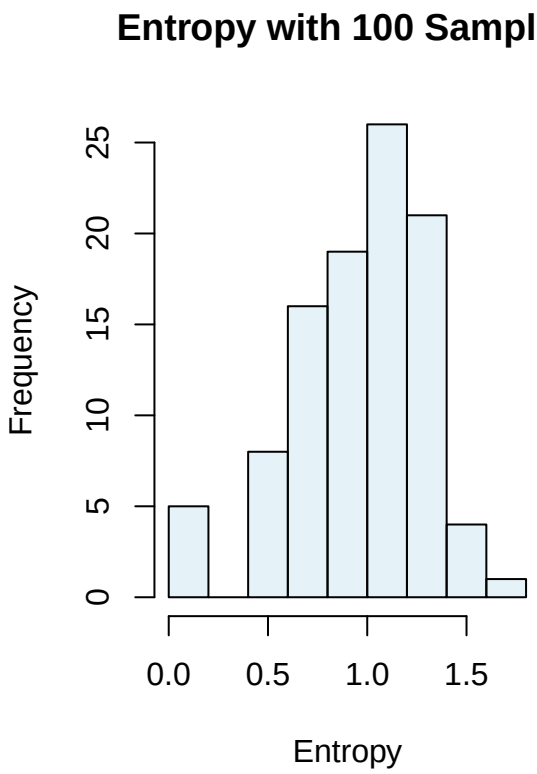
Entropy of a random variable is average level of "Information", "Surprise" or "Uncertainty" inherent in variables possible outcome.

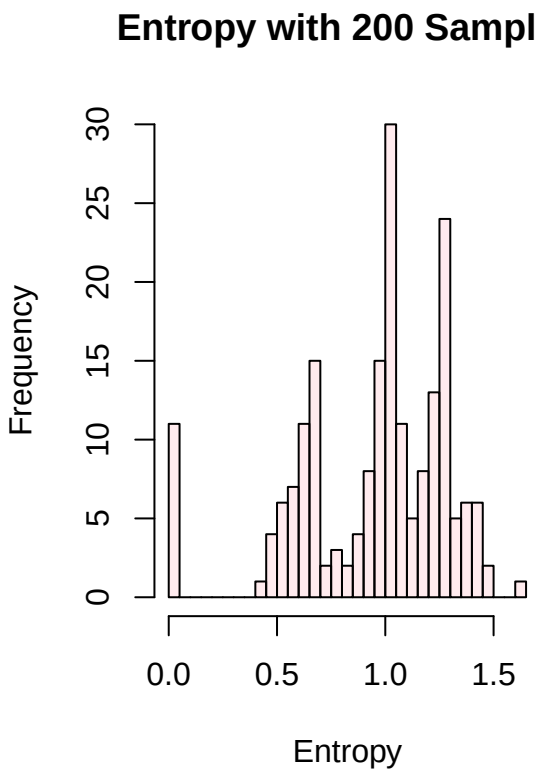
Now a random variable X, with possible outcome $ x_1,x_2,....x_n $ which occur with probability of $P(x_1), P(x_2), ..., P(x_n)$, the entropy of X is formally defined as:

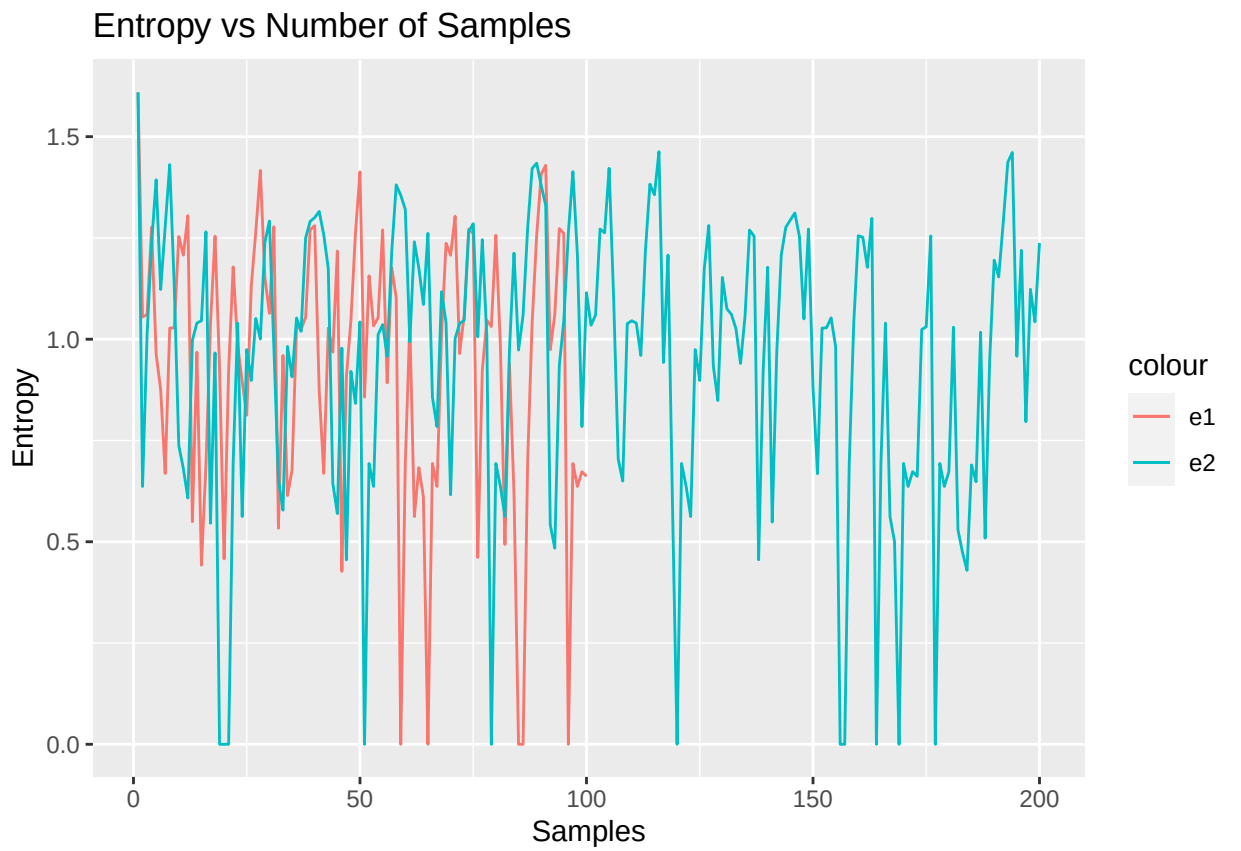
$$H(X) = -\sum_{i=1}^{n} P(x_i) log P(x_i)$$

So higher entropy refer to more uncertainty, thus in order to check if we increase number of observations, do we have better understanding of where robot is, we would calculate entropy for filtered probabilities as they give result for robot location at time t with 1:t time information.

```
#genrating more samples with same seed values
suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
nSim = 200
simulateHmm3 = simHMM(hmm , nSim)
filterdProb3 = exp(HMM::forward(hmm , simulateHmm3$observation))

library(entropy)

e1Fwd = apply(X = filterdProb , MARGIN = 2 , FUN =  entropy.empirical)
#e1FwdBack = apply(X = smoothAccuracy , MARGIN = 2 , FUN =  entropy.empirical)

e2Fwd = apply(X = filterdProb3 , MARGIN = 2 , FUN =  entropy.empirical)
```


Entropy with 100 Samples
Frequency
0
5
10
15
20
25
0.0
0.5
1.0
1.5
Entropy


Entropy with 200 Samples
Frequency
0
5
10
15
20
25
30
0.0
0.5
1.0
1.5
Entropy



It can be seen from above plot of entropy (i.e. uncertainty) is not improving over time with more and more data. As robot current position x_t only depend on its previous position such that given x_{t-1} and it conditionally independent of all remaining positions given x_{t-1}.

Question 7

1 Step Prediction

Using prior knowledge of its position as time t = 100 and transition probability, we can estimate probabilities of hidden state at time stamp t = 101.

To find probability of hidden state at time stamp 101 with given prior observations i.e. $p(Z_{101}|x_{1:100})$,

$$p(Z_{101}|x_{1:100}) = \sum_{Z_{100}} p(Z_{101}, Z_{100}|x_{1:100}) = \sum_{Z_{100}} p(Z_{101}|Z_{100})P(Z_{100}|x_{1:100})$$

Here $p(Z_{101}|Z_{100})$ is our transition probability and $P(Z_{100}|x_{1:100})$ is our filtered probability for state Z_{100}.

```
#normalized prob
normalisedProb = prop.table(filterdProb , margin = 2)

hiddenStates101 = t(transProbs) %*% normalisedProb[,100]
cat("\n")
```

```
cat("Probabilities of the hidden states for the time step 101 is")
```

```
## Probabilities of the hidden states for the time step 101 is
```

```r
cat("\n")
```

```r
hiddenStates101
```

```
##       [,1]
## 1  0.0000
## 2  0.1875
## 3  0.5000
## 4  0.3125
## 5  0.0000
## 6  0.0000
## 7  0.0000
## 8  0.0000
## 9  0.0000
## 10 0.0000
```

```r
cat("\n")
```

```r
cat("The most probable state at t= 101 is : " , as.character(which.max(hiddenStates101)))
```

```
## The most probable state at t= 101 is :  3
```

```r
cat("\n")
```

References

1. Package "HMM"
2. Entropy

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
#Building Hidden Markov Model
library(HMM)

States =  1:10
Symbols = 1:10

#Since
#the robot is in one of the sectors and decides with equal probability to stay in that sector or move t
#we get probability of staying as 0.5 and probability of moving as 0.5

# transProbs = 0.5*diag(10)
# for(i in 1:ncol(transProbs))
# {
#     if(i != ncol(transProbs))
#     {
#       transProbs[i+1 , i]  = 0.5
#     }else{
#       transProbs[1 , i]  = 0.5
```

```
#     }
# }

transProbs = 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
               )

transProbs = matrix(data =  transProbs , byrow =  TRUE , nrow = 10 , ncol = 10)

colnames(transProbs) = States
rownames(transProbs) = States


#emissionProbs
#If the robot is in the sector i, then the device will report that the robot is in the sectors [i - 2,
#so robot can be in any sector i with probability of 0.2 (5/10)

emissionProbs = 0.2 * diag(10)
n = ncol(emissionProbs)
for(i in 1:n){
  if(i-1 == 0) {
    Col_Indx = c((n-1) , n , i , i+1 , i+2)
  }else if(i-1 ==  1){
    Col_Indx = c( n , i-1 ,i , i+1 , i+2)
  }else if(i+1 == n){
    Col_Indx = c( i-2 ,i-1 ,i , i+1 , 1)
  }else if(i+1 > n) {
    Col_Indx = c( i-2 ,i-1 ,i , 1 , 2)
  }else{
    Col_Indx = c( i-2 , i-1 ,i , i+1 , i+2)
  }
  emissionProbs[i , Col_Indx] = 0.2
}#for

hmm = HMM::initHMM(States , Symbols , transProbs = transProbs , emissionProbs = emissionProbs)

#Hidden Markov Model


cat("\n")
cat("Hidden Markov Model for given scenario is")
cat("\n")
cat("\n")
hmm
```

```r
suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
nSim = 100
simulateHmm = simHMM(hmm , nSim)
cat("\n")
cat("Simulated Values")
cat("\n")
simulateHmm
cat("\n")

#Since we have received probabilities in log, taking anti-log
filterdProb = exp(HMM::forward(hmm , simulateHmm$observation))

smoothedProb = HMM::posterior(hmm , simulateHmm$observation)

probablePath = HMM::viterbi(hmm, simulateHmm$observation)


fnAccuracy = function(prob , Margin = 2 , States){
  #prob = probability of filter or smoother
  #Margin = margin = 2
  #States : States obtained during simulation

  #normalized prob
  normalisedProb = prop.table(prob , margin = Margin)

  #predicted states
  PredStates = apply(normalisedProb, MARGIN = Margin, which.max)

  #Calculate Accuracy in percentage when compared with  States obtained during simulation
  percAcc = sum(PredStates == States) / length(States)

  return(percAcc)
}


filterAccuracy = fnAccuracy(prob = filterdProb , Margin = 2 , States = simulateHmm$states)
smoothAccuracy = fnAccuracy(prob = smoothedProb , Margin = 2 , States = simulateHmm$states)
probPathAcc = sum(probablePath == simulateHmm$observation)/length(simulateHmm$observation)

accDf = data.frame("Filter" = filterAccuracy ,
                   "Smoothing" = smoothAccuracy,
                   "Probable Path" = probPathAcc)

knitr::kable(accDf , caption = "Accuracy Table")

suppressWarnings(RNGversion("3.5.1"))
set.seed(9999789)
nSim = 100
simulateHmm2 = simHMM(hmm , nSim)
filterdProb2 = exp(HMM::forward(hmm , simulateHmm2$observation))

smoothedProb2 = HMM::posterior(hmm , simulateHmm2$observation)
```

```r
probablePath2 = HMM::viterbi(hmm, simulateHmm2$observation)

filterAccuracy2 = fnAccuracy(prob = filterdProb2 , Margin = 2 , States = simulateHmm2$states)
smoothAccuracy2 = fnAccuracy(prob = smoothedProb2 , Margin = 2 , States = simulateHmm2$states)
probPathAcc2 = sum(probablePath2 == simulateHmm2$observation)/length(simulateHmm2$observation)


accDf = data.frame("Filter" = filterAccuracy2 ,
                   "Smoothing" = smoothAccuracy2,
                   "Probable Path" = probPathAcc2)

knitr::kable(accDf , caption = "Accuracy Table with Different Sample")

#genrating more samples with same seed values
suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
nSim = 200
simulateHmm3 = simHMM(hmm , nSim)
filterdProb3 = exp(HMM::forward(hmm , simulateHmm3$observation))

library(entropy)

e1Fwd = apply(X = filterdProb , MARGIN = 2 , FUN =  entropy.empirical)
#e1FwdBack = apply(X = smoothAccuracy , MARGIN = 2 , FUN =  entropy.empirical)

e2Fwd = apply(X = filterdProb3 , MARGIN = 2 , FUN =  entropy.empirical)




c1 <- rgb(173,216,230,max = 255, alpha = 80, names = "lt.blue")
c2 <- rgb(255,192,203, max = 255, alpha = 80, names = "lt.pink")
par(mfrow = c(1,2))
hist(e1Fwd , main = "Entropy with 100 Samples" , breaks = 10  ,col = c1 , xlab = "Entropy")
hist(e2Fwd , main = "Entropy with 200 Samples" , breaks = 50 ,col = c2 ,  xlab = "Entropy")


library(ggplot2)
ggplot()+
  geom_line(aes(x = 1:length(e1Fwd) , y = e1Fwd , color = "e1"))+
  geom_line(aes(x = 1:length(e2Fwd) , y = e2Fwd , color = "e2"))+
  xlab("Samples")+
  ylab("Entropy")+
  ggtitle("Entropy vs Number of Samples")


#normalized prob
normalisedProb = prop.table(filterdProb , margin = 2)

hiddenStates101 = t(transProbs) %*% normalisedProb[,100]
cat("\n")
cat("Probabilities of the hidden states for the time step 101 is")
```

```r
cat("\n")
hiddenStates101
cat("\n")
cat("The most probable state at t= 101 is : " , as.character(which.max(hiddenStates101)))
cat("\n")
```