Advanced Machine Learning Individual Report

Akshath Srinivas(akssr921)

2023-09-18

Question 1

```
# Define the number of states and symbols
num_states <- 10
num_symbols <- 10

# Initial state probabilities (uniform)
start_prob <- rep(1/num_states, num_states)

transition_probs <-  matrix(
  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),nrow = 10,byrow = TRUE
)

# Emission probabilities (higher for current sector, lower for adjacent sectors)
emission_probs <- matrix(
  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),nrow = 10,byrow = TRUE
)

states <- as.character(1:num_states)
symbols <- as.character(1:num_symbols)
# Create the HMM
hmm <- initHMM(states, symbols,start_prob, transition_probs,emission_probs)
```

Question 2

The code below will generate 100 steps.

```
sample = simHMM(hmm,100)
sample
```

```
## $states
##   [1] "9"  "9"  "10" "1"  "1"  "1"  "2"  "3"  "4"  "4"  "5"  "5"  "5"  "5"  "6"
##  [16] "7"  "7"  "8"  "8"  "8"  "8"  "8"  "9"  "9"  "9"  "10" "1"  "1"  "1"  "1"
##  [31] "1"  "2"  "2"  "3"  "3"  "4"  "5"  "6"  "7"  "7"  "8"  "9"  "10" "1"  "1"
##  [46] "2"  "2"  "2"  "3"  "3"  "3"  "3"  "3"  "3"  "4"  "5"  "5"  "5"  "6"  "7"
##  [61] "8"  "8"  "9"  "9"  "9"  "9"  "9"  "10" "10" "1"  "1"  "1"  "2"  "2"  "2"
##  [76] "3"  "3"  "3"  "3"  "4"  "5"  "6"  "7"  "8"  "8"  "8"  "8"  "9"  "9"  "9"
##  [91] "9"  "9"  "10" "10" "10" "10" "1"  "2"  "3"  "3"
##
## $observation
##   [1] "8"  "9"  "8"  "3"  "10" "3"  "4"  "1"  "4"  "6"  "5"  "5"  "5"  "7"  "7"
##  [16] "5"  "6"  "6"  "7"  "6"  "9"  "10" "9"  "7"  "1"  "8"  "10" "2"  "1"  "9"
##  [31] "3"  "10" "3"  "1"  "4"  "4"  "4"  "7"  "6"  "8"  "10" "10" "1"  "9"  "10"
##  [46] "3"  "4"  "2"  "2"  "2"  "4"  "2"  "3"  "3"  "4"  "4"  "6"  "3"  "4"  "5"
##  [61] "10" "9"  "7"  "7"  "10" "8"  "10" "8"  "1"  "10" "10" "2"  "4"  "10" "3"
##  [76] "2"  "2"  "5"  "2"  "5"  "3"  "5"  "5"  "7"  "8"  "10" "7"  "8"  "8"  "8"
##  [91] "8"  "7"  "10" "1"  "10" "9"  "3"  "3"  "1"  "5"
```

Question 3

Below is the code for the filtered and smoothed probability distributions for each of the 100 time point and also to compute the most probable path.

```
sample = simHMM(hmm,100)
fil_smooth <- function(sam){
  observed <- sam$observation
    #calculate alpha
  alpha <-  exp(forward(hmm, observed))
    #filtering
  fil <-  apply(alpha,2,prop.table)
    #calculating beta
  beta <-  exp(backward(hmm,observed))

    #smoothing
  smo <-  alpha*beta
  smooth <- apply(smo,2,prop.table)
  return(list(fil,smooth))
}


#filtering
filter <-  fil_smooth(sample)[[1]]

smooth <- fil_smooth(sample)[[2]]
```

```
#path of viterbi algorithm
path = viterbi(hmm, sample$observation)


cat('The most Probable Path is')
```

```
## The most Probable Path is
```

```
path
```

```
##   [1] "4"  "5"  "6"  "7"  "8"  "9"  "10" "1"  "1"  "1"  "1"  "1"  "2"  "3"  "4"
##  [16] "5"  "5"  "6"  "7"  "8"  "8"  "8"  "8"  "8"  "9"  "10" "1"  "1"  "1"  "1"
##  [31] "1"  "2"  "3"  "4"  "5"  "6"  "6"  "6"  "6"  "7"  "8"  "8"  "9"  "10" "1"
##  [46] "1"  "1"  "1"  "2"  "3"  "4"  "5"  "5"  "5"  "6"  "7"  "8"  "9"  "10" "1"
##  [61] "1"  "1"  "1"  "1"  "1"  "1"  "2"  "3"  "4"  "5"  "5"  "5"  "6"  "7"  "8"
##  [76] "8"  "8"  "9"  "10" "1"  "2"  "2"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"
##  [91] "10" "1"  "1"  "1"  "1"  "1"  "2"  "2"  "2"  "2"
```

Filtered and smoothed probability distribution values can be seen below

```
filter[,1:10]
```

```
##       index
## states   1         2         3          4          5         6         7
##     1  0.0 0.0000000 0.0000000 0.00000000 0.00000000 0.0000000 0.0000000
##     2  0.2 0.1111111 0.0000000 0.00000000 0.00000000 0.0000000 0.0000000
##     3  0.2 0.2222222 0.0000000 0.00000000 0.00000000 0.0000000 0.0000000
##     4  0.2 0.2222222 0.2857143 0.14285714 0.00000000 0.0000000 0.0000000
##     5  0.2 0.2222222 0.2857143 0.28571429 0.00000000 0.0000000 0.0000000
##     6  0.2 0.2222222 0.2857143 0.28571429 0.00000000 0.0000000 0.0000000
##     7  0.0 0.0000000 0.1428571 0.21428571 0.58333333 0.3043478 0.1521739
##     8  0.0 0.0000000 0.0000000 0.07142857 0.33333333 0.4782609 0.3913043
##     9  0.0 0.0000000 0.0000000 0.00000000 0.08333333 0.2173913 0.3478261
##     10 0.0 0.0000000 0.0000000 0.00000000 0.00000000 0.0000000 0.1086957
##       index
## states          8          9         10
##     1  0.08333333 0.21666667 0.33750000
##     2  0.00000000 0.04166667 0.12916667
##     3  0.00000000 0.00000000 0.02083333
##     4  0.00000000 0.00000000 0.00000000
##     5  0.00000000 0.00000000 0.00000000
##     6  0.00000000 0.00000000 0.00000000
##     7  0.00000000 0.00000000 0.00000000
##     8  0.00000000 0.00000000 0.00000000
##     9  0.56666667 0.28333333 0.14166667
##     10 0.35000000 0.45833333 0.37083333
```

```
smooth[,1:10]
```

```
##       index
## states          1         2         3         4          5         6         7
```

```
##     1  0.00000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000 0.0000000
##     2  0.00000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000 0.0000000
##     3  0.03297055 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000 0.0000000
##     4  0.19078105 0.0659411 0.0000000 0.0000000 0.00000000 0.0000000 0.0000000
##     5  0.46702945 0.3156210 0.1318822 0.0000000 0.00000000 0.0000000 0.0000000
##     6  0.30921895 0.6184379 0.4993598 0.2637644 0.00000000 0.0000000 0.0000000
##     7  0.00000000 0.0000000 0.3687580 0.5512164 0.46158771 0.1075544 0.0000000
##     8  0.00000000 0.0000000 0.0000000 0.1850192 0.47119078 0.5563380 0.2765685
##     9  0.00000000 0.0000000 0.0000000 0.0000000 0.06722151 0.3361076 0.5633803
##     10 0.00000000 0.0000000 0.0000000 0.0000000 0.00000000 0.0000000 0.1600512
##       index
## states          8          9         10
##     1  0.06081946 0.24135723 0.51856594
##     2  0.00000000 0.01440461 0.08930858
##     3  0.00000000 0.00000000 0.00000000
##     4  0.00000000 0.00000000 0.00000000
##     5  0.00000000 0.00000000 0.00000000
##     6  0.00000000 0.00000000 0.00000000
##     7  0.00000000 0.00000000 0.00000000
##     8  0.00000000 0.00000000 0.00000000
##     9  0.52240717 0.16325224 0.02176697
##     10 0.41677337 0.58098592 0.37035851
```

Question 4

```
#function to return the states name from probability table
fil_smooth_pred <- function(x){
  sta <- rep(NA,ncol(x))
  for(i in 1:ncol(x)){
    sta[i] <- names(which.max(x[,i]))
  }
  return(sta)
}

#getting the predicted states
filter_pred <- fil_smooth_pred(filter)
smooth_pred <- fil_smooth_pred(smooth)


#function to calculate accuracy
cal_accuracy <- function(pre,act){
  t <- table(pre,act)
  acc <- sum(diag(t))/ sum(t)
  return(acc)
}

#accuracy of filtering
acc_filter <- cal_accuracy(filter_pred,sample$states)

#accuracy of smoothing
acc_smooth <- cal_accuracy(smooth_pred,sample$states)
```

```r
#viterbi accuracy
acc_viterbi <- cal_accuracy(path,sample$states)

cat('Accuracy of filtered probability distributions is',acc_filter*100,'%')
```

```
## Accuracy of filtered probability distributions is 50 %
```

```r
cat('Accuracy of smoothed probability distributions is',acc_smooth*100,'%')
```

```
## Accuracy of smoothed probability distributions is 74 %
```

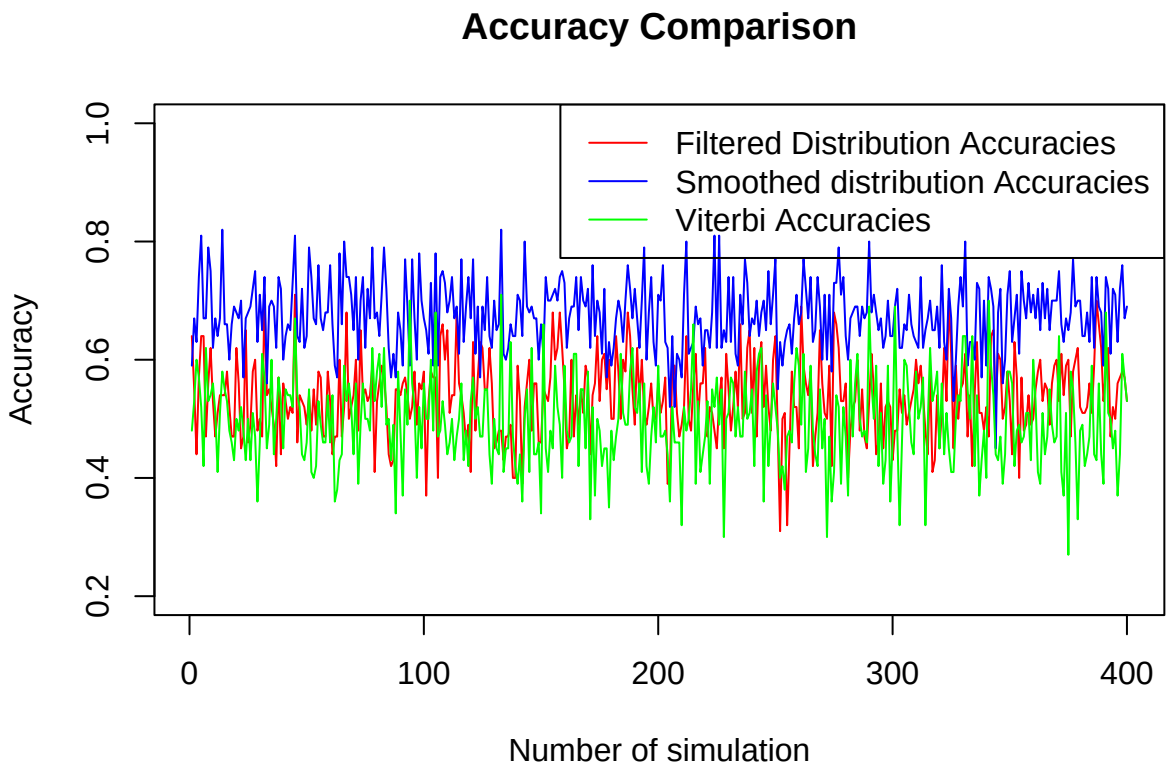
```r
cat('Accuracy of most probable path is',acc_viterbi*100,'%')
```

```
## Accuracy of most probable path is 55 %
```

Question 5

```r
#function to simulate samples and calculate accuracies
simulated_samples <- function(n) {
  filter_accs <- rep(NA,n)
  smooth_accs <- rep(NA,n)
  viterbi_accs <- rep(NA,n)

  for(i in 1:n){
    #generate samples for n iterations
    samp <- simHMM(hmm,100)
    #filtering and smoothing
    filter <- fil_smooth(samp)[[1]]
    smooth <- fil_smooth(samp)[[2]]
    #getting predictions
    fil_pred <- fil_smooth_pred(filter)
    smo_pred <- fil_smooth_pred(smooth)
    #getting and saving accuracies of filtering,smoothing and viterbi path
    filter_accs[i] <- cal_accuracy(fil_pred,samp$states)
    smooth_accs[i] <- cal_accuracy(smo_pred,samp$states)
    pth <- viterbi(hmm, samp$observation)
    viterbi_accs [i] <- cal_accuracy(pth,samp$states)
  }

  return(list(filter_accs,smooth_accs,viterbi_accs))
}

accuracies <- simulated_samples(400)
filter_accs <- accuracies[[1]]
smooth_accs <- accuracies[[2]]
viterbi_accs <- accuracies[[3]]
# Plotting accuracies
plot(filter_accs, type = 'l', col = 'red', ylim = c(0.2, 1),
     xlab = "Number of simulation", ylab = "Accuracy",
```

```
     main = "Accuracy Comparison")

lines(smooth_accs, col = 'blue')
lines(viterbi_accs, col = 'green')

# Adding Legend
legend("topright", legend = c("Filtered Distribution Accuracies", "Smoothed distribution Accuracies", "
       col = c("red", "blue", "green"), lty = 1)
```

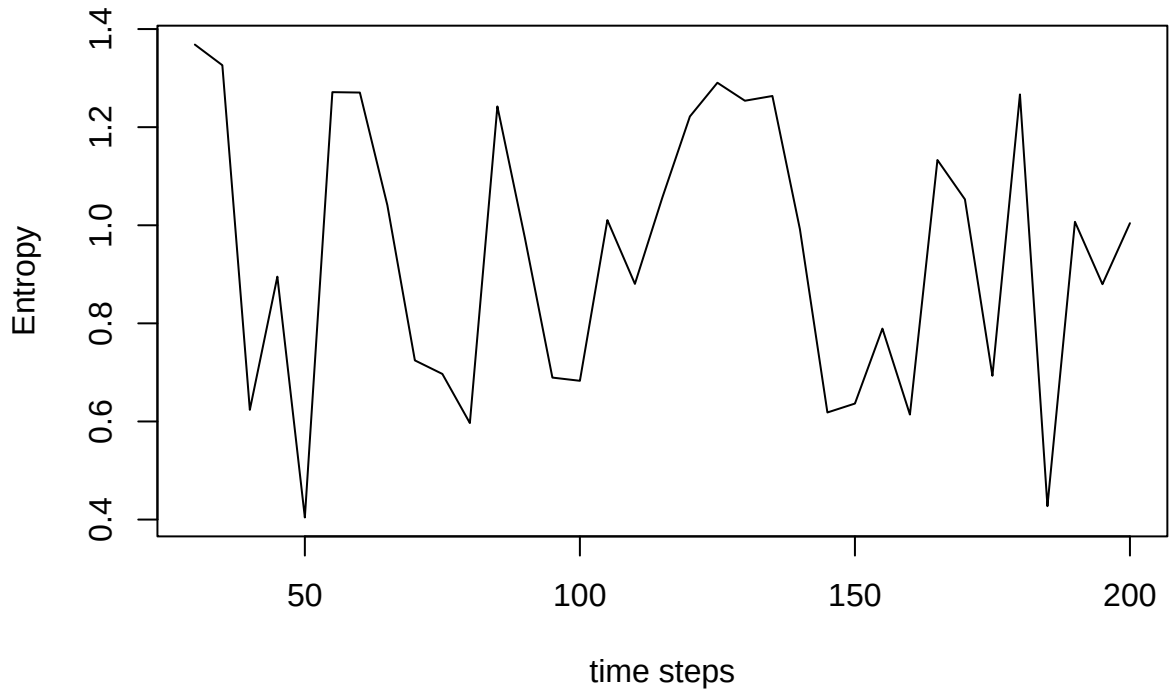


The smoothed distributions should generally be more accurate than the filtered distributions and most probable paths because of below reasons:

- The main difference between the filtered and smoothed estimates is that the smoother includes future observations. This means that when estimating past states, it has access to more information, including observations that occur after the time point of interest. In contrast, the filtered estimate only uses past observations up to the current time.By considering future observations, the smoother can reduce the impact of noisy or uncertain measurements. It can smooth out the estimates over time, leading to more stable and accurate state estimates. In contrast, the filtered measurements may be influenced by short-term fluctuations in the data.
- The Viterbi Algorithm operates by maximizing the joint distribution of all hidden states (from Z_0 to Z_T). This approach prioritizes selecting the hidden state that best fits the overall sequence of observations, not necessarily the one with the highest individual probability at each time step. As a result, it might not always choose the state with the highest local probability, aiming for a more overall path. However, this can sometimes lead to less accuracy compared to smoothing distributions.

Question 6

```
obs_samp <- simHMM(hmm,200)$observation
seque <- seq(30,200,5)

library(entropy)
alpha <-  exp(forward(hmm, obs_samp))
  #filtering
fil <-  apply(alpha,2,prop.table)
entropies_vec<-rep(NA,length(seque))
for(j in 1:length(entropies_vec)){
  entropies_vec[j] <- entropy.empirical(fil[,seque[j]])
}
plot(seque,entropies_vec,type='l',xlab='time steps'
     ,ylab='Entropy')
```



No, it's not always true that the later in time, with more observations received, we better know where the robot is. Entropy measures the uncertainty or randomness in a probability distribution. If the entropy is high, it means there is a lot of uncertainty, and if it's low, it means there is more certainty. From the plot above we can say that the fluctuation in entropy suggests that there are moments when the robot's position is relatively uncertain, even with more observations.

Question 7

We have to compute the probabilities of the hidden states for the time step 101 by deriving below expression.

$$P(Z_{101}|X_{1:100})$$

Above expression can be written like below, adding Z_{100} and marginalizing out Z_{100}.

$$\sum_{z_{100}} P(Z_{100}, Z_{101}|X_{1:100})$$

By rules of conditional probability.

$$\sum_{x}^{z_{100}} P(Z_{100}, Z_{101}|X_{1:100}) = \sum_{x}^{z_{100}} P(Z_{100}|X_{1:100})P(Z_{101}|Z_{100})$$

Notice that $P(Z_{100}|X_{1:100})$ is filter distribution at time stamp 100, $P(Z_{101}|Z_{100})$ is transition probability matrix. The $P(Z_{101}|X_{1:100})$ can be obtained like below in the code.

```
obs_samp <- simHMM(hmm,100)$observation
alpha <-  exp(forward(hmm, obs_samp))
#filtering
filt <-  apply(alpha,2,prop.table)
prob_101 <- transition_probs %*% filt[,ncol(filt)]

cat('The probabilities of the hidden states for the time step 101')
```

```
## The probabilities of the hidden states for the time step 101
```

```
prob_101
```

```
##       [,1]
##  [1,]  0.0
##  [2,]  0.0
##  [3,]  0.5
##  [4,]  0.5
##  [5,]  0.0
##  [6,]  0.0
##  [7,]  0.0
##  [8,]  0.0
##  [9,]  0.0
## [10,]  0.0
```

Appendix

```
knitr::opts_chunk$set(warning = FALSE,echo = TRUE)
#question 1
library(HMM)

# Define the number of states and symbols
num_states <- 10
num_symbols <- 10
```

```r
# Initial state probabilities (uniform)
start_prob <- rep(1/num_states, num_states)

transition_probs <-  matrix(
  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),nrow = 10,byrow = TRUE
)

# Emission probabilities (higher for current sector, lower for adjacent sectors)
emission_probs <- matrix(
  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),nrow = 10,byrow = TRUE
)

states <- as.character(1:num_states)
symbols <- as.character(1:num_symbols)
# Create the HMM
hmm <- initHMM(states, symbols,start_prob, transition_probs,emission_probs)


sample = simHMM(hmm,100)
sample


sample = simHMM(hmm,100)
fil_smooth <- function(sam){
  observed <- sam$observation
    #calculate alpha
  alpha <-  exp(forward(hmm, observed))
    #filtering
  fil <-  apply(alpha,2,prop.table)
    #calculating beta
  beta <-  exp(backward(hmm,observed))

    #smoothing
  smo <-  alpha*beta
  smooth <- apply(smo,2,prop.table)
```

```
  return(list(fil,smooth))
}


#filtering
filter <-  fil_smooth(sample)[[1]]

smooth <- fil_smooth(sample)[[2]]

#path of viterbi algorithm
path = viterbi(hmm, sample$observation)


cat('The most Probable Path is')
path
filter[,1:10]
smooth[,1:10]
#function to return the states name from probability table
fil_smooth_pred <- function(x){
  sta <- rep(NA,ncol(x))
  for(i in 1:ncol(x)){
    sta[i] <- names(which.max(x[,i]))
  }
  return(sta)
}

#getting the predicted states
filter_pred <- fil_smooth_pred(filter)
smooth_pred <- fil_smooth_pred(smooth)


#function to calculate accuracy
cal_accuracy <- function(pre,act){
  t <- table(pre,act)
  acc <- sum(diag(t))/ sum(t)
  return(acc)
}

#accuracy of filtering
acc_filter <- cal_accuracy(filter_pred,sample$states)

#accuracy of smoothing
acc_smooth <- cal_accuracy(smooth_pred,sample$states)

#viterbi accuracy
acc_viterbi <- cal_accuracy(path,sample$states)

cat('Accuracy of filtered probability distributions is',acc_filter*100,'%')

cat('Accuracy of smoothed probability distributions is',acc_smooth*100,'%')

cat('Accuracy of most probable path is',acc_viterbi*100,'%')
#function to simulate samples and calculate accuracies
```

```r
simulated_samples <- function(n) {
  filter_accs <- rep(NA,n)
  smooth_accs <- rep(NA,n)
  viterbi_accs <- rep(NA,n)

  for(i in 1:n){
    #generate samples for n iterations
    samp <- simHMM(hmm,100)
    #filtering and smoothing
    filter <- fil_smooth(samp)[[1]]
    smooth <- fil_smooth(samp)[[2]]
    #getting predictions
    fil_pred <- fil_smooth_pred(filter)
    smo_pred <- fil_smooth_pred(smooth)
    #getting and saving accuracies of filtering,smoothing and viterbi path
    filter_accs[i] <- cal_accuracy(fil_pred,samp$states)
    smooth_accs[i] <- cal_accuracy(smo_pred,samp$states)
    pth <- viterbi(hmm, samp$observation)
    viterbi_accs [i] <- cal_accuracy(pth,samp$states)
  }

  return(list(filter_accs,smooth_accs,viterbi_accs))
}

accuracies <- simulated_samples(400)
filter_accs <- accuracies[[1]]
smooth_accs <- accuracies[[2]]
viterbi_accs <- accuracies[[3]]
# Plotting accuracies
plot(filter_accs, type = 'l', col = 'red', ylim = c(0.2, 1),
     xlab = "Number of simulation", ylab = "Accuracy",
     main = "Accuracy Comparison")

lines(smooth_accs, col = 'blue')
lines(viterbi_accs, col = 'green')

# Adding Legend
legend("topright", legend = c("Filtered Distribution Accuracies", "Smoothed distribution Accuracies", "
       col = c("red", "blue", "green"), lty = 1)


obs_samp <- simHMM(hmm,200)$observation
seque <- seq(30,200,5)

library(entropy)
alpha <-  exp(forward(hmm, obs_samp))
  #filtering
fil <-  apply(alpha,2,prop.table)
entropies_vec<-rep(NA,length(seque))
for(j in 1:length(entropies_vec)){
  entropies_vec[j] <- entropy.empirical(fil[,seque[j]])
}
plot(seque,entropies_vec,type='l',xlab='time steps'
```

```
     ,ylab='Entropy')

obs_samp <- simHMM(hmm,100)$observation
alpha <-  exp(forward(hmm, obs_samp))
#filtering
filt <-  apply(alpha,2,prop.table)
prob_101 <- transition_probs %*% filt[,ncol(filt)]

cat('The probabilities of the hidden states for the time step 101')
prob_101
```