Advanced Machine Learning - Lab 2

Ali Etminan

9/23/2020

Question 1

In order to construct our model, we need to define transition probabilities, probability of moving from state **i** to state **j**, as well as the emission probabilities which denote the probability of a state given the observations.

```
# Question 1

states <- symbols <- 1:10
symbols <- c("A","B","C","D","E","F","G","H","I","J")
emitprob <- matrix(c(.2,.2,.2,0,0,0,0,0,.2,.2,
                   .2,.2,.2,.2,0,0,0,0,0,.2,
                   .2,.2,.2,.2,.2,0,0,0,0,0,
                   0,.2,.2,.2,.2,.2,0,0,0,0,
                   0,0,.2,.2,.2,.2,.2,0,0,0,
                   0,0,0,.2,.2,.2,.2,.2,0,0,
                   0,0,0,0,.2,.2,.2,.2,.2,0,
                   0,0,0,0,0,.2,.2,.2,.2,.2,
                   .2,0,0,0,0,0,.2,.2,.2,.2,
                   .2,.2,0,0,0,0,0,.2,.2,.2), ncol = 10)


transprob <- matrix(c(.5,0,0,0,0,0,0,0,0,.5,
                      .5,.5,0,0,0,0,0,0,0,0,
                     0,.5,.5,0,0,0,0,0,0,0,
                     0,0,.5,.5,0,0,0,0,0,0,
                     0,0,0,.5,.5,0,0,0,0,0,
                     0,0,0,0,.5,.5,0,0,0,0,
                     0,0,0,0,0,.5,.5,0,0,0,
                     0,0,0,0,0,0,.5,.5,0,0,
                     0,0,0,0,0,0,0,.5,.5,0,
                     0,0,0,0,0,0,0,0,.5,.5), ncol = 10)

hmm <- initHMM(States = states,
               Symbols = symbols,
               emissionProbs = emitprob,
               transProbs = transprob)
```

Question 2 - 3

Now we simulate our Hidden Markov Model 100 times and store the observations in a separate variable.

```r
# Question 2

simulated <- simHMM(hmm, 100)

obs <- simulated$observation
```

In this stage, the Forward-Backward algorithm is used to obtain forward and backward probabilities, which we use to compute the *Filtered* and *Smoothed* probability distributions.

```r
# Question 3

# the forward step probabilities (alpha)
frwrd <- forward(hmm, obs)

# the backward step probabilities (beta)
bkwrd <- backward(hmm, obs)

# Probabilities are converted back from log-scale
backward <- exp(bkwrd)
forward <- exp(frwrd)

# obtaining filtering distribution for each t
filtering <- matrix(0, 10, 100)

for (i in 1:100) {
  filtering[,i] <- forward[,i]/sum(forward[,i])
}

# obtaining smoothing distribution for each t
smoothing <- matrix(0, 10, 100)

for(i in 1:100) {
  smoothing[,i] <- (forward[,i] * backward[,i])/sum(forward[,i] * backward[,i])
}
```

Using the Viterbi algorithm on our model observations, we obtain the most probable path.

```
## The most probable path:
```

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

Question 4

The following two functions first find the argument (state) that maximizes the probability in each time t and then output the confusion matrix and misclassification rate for both the filtered and smoothed probabilities.

```r
# Question 4
accuracy <- function(method_prob,sim) {
  conf <- table(sim, method_prob)
  misclass <- 1 - sum(diag(conf))/sum(conf)
  return(list(ConfusionMatrix=conf, MisclassificationRate=misclass))
}
```

```
most_probable <- function(mat,sim) {
  method_prob <- apply(mat, 2, FUN = which.max)
  accuracy(method_prob, sim)
}
```

```
## Confusion matrix and miscalssification rate for smoothed distributions:
```

```
## $ConfusionMatrix
##     method_prob
## sim  1 2 3 4 5 6 7 8 9 10
##   1  6 1 0 0 0 0 0 0 0  3
##   2  2 5 4 2 0 0 0 0 0  0
##   3  0 2 4 3 0 0 0 0 0  0
##   4  0 0 0 4 2 0 0 0 0  0
##   5  0 0 0 2 6 2 0 0 0  0
##   6  0 0 0 0 3 5 1 0 0  0
##   7  0 0 0 0 1 2 7 0 0  0
##   8  0 0 0 0 0 0 3 7 0  0
##   9  0 0 0 0 0 0 0 3 8  1
##   10 3 1 0 0 0 0 0 0 2  5
##
## $MisclassificationRate
## [1] 0.43
```

```
## Confusion matrix and miscalssification rate for filtered distributions:
```

```
## $ConfusionMatrix
##     method_prob
## sim  1 2 3 4 5 6 7 8 9 10
##   1  5 1 0 0 0 0 0 0 1  3
##   2  1 7 4 0 0 0 0 0 0  1
##   3  1 2 4 1 0 0 0 0 0  1
##   4  0 0 1 4 1 0 0 0 0  0
##   5  0 0 0 3 6 1 0 0 0  0
##   6  0 0 0 0 5 3 1 0 0  0
##   7  0 0 0 0 1 7 2 0 0  0
##   8  0 0 0 0 0 1 4 3 2  0
##   9  0 0 0 0 0 0 0 4 7  1
##   10 3 1 0 0 0 0 0 0 3  4
##
## $MisclassificationRate
## [1] 0.55
```

The misclassification rate for the most probable path is:

```
## [1] 0.56
```

Question 5

Now we simulate our model 200 times and repeat the above steps to get the following output.

```
## The most probable path:
```

```
##   [1]  1  1  1  1  1  2  3  3  4  5  5  6  7  7  8  9 10  1  1  1  1  1  1  1  1
##  [26]  1  2  3  4  5  6  7  7  8  9 10  1  1  1  1  1  1  1  2  3  3  4  4  5  6
##  [51]  7  7  8  9 10 10 10  1  1  1  1  1  2  2  2  3  4  5  6  6  6  6  7  7  8
##  [76]  9 10  1  1  1  1  2  3  4  4  4  4  4  4  4  4  5  6  7  7  7  8  9 10  1
## [101]  1  1  1  2  3  3  3  4  5  5  5  6  6  6  7  7  8  9  9  9 10  1  1  1  1
## [126]  1  1  2  3  3  3  3  3  3  3  4  4  5  6  7  8  9 10  1  1  1  2  3  4  4
## [151]  4  4  4  5  5  6  7  7  7  7  7  7  7  7  8  9 10  1  1  2  3  4  5  6  7
## [176]  8  9 10  1  1  1  1  1  1  1  1  1  1  2  3  4  4  4  5  6  7  8  9 10  1
```

```
## Confusion matrix and miscalssification rate for smoothed distributions:
```

```
## $ConfusionMatrix
##     method_prob
## sim   1  2  3  4  5  6  7  8  9 10
##   1  16  0  0  0  0  0  0  0  0  0
##   2   3 13  2  0  0  0  0  0  0  0
##   3   0 10 13  4  0  0  0  0  0  0
##   4   0  0  5 18  1  0  0  0  0  0
##   5   0  0  0  2 15  4  0  0  0  0
##   6   0  0  0  0  3 12  3  0  0  0
##   7   0  0  0  0  0  2 17  4  0  0
##   8   0  0  0  0  0  0  3 13  4  0
##   9   0  0  0  0  0  0  0  1 12  4
##   10  2  0  0  0  0  0  0  0  1 13
##
## $MisclassificationRate
## [1] 0.29
```

```
## Confusion matrix and miscalssification rate for filtered distributions:
```

```
## $ConfusionMatrix
##     method_prob
## sim   1  2  3  4  5  6  7  8  9 10
##   1  11  3  0  0  0  0  0  0  0  2
##   2   4 10  2  0  0  0  0  0  0  2
##   3   2  7 14  4  0  0  0  0  0  0
##   4   0  1  3 15  5  0  0  0  0  0
##   5   0  0  2  2 11  5  1  0  0  0
##   6   0  0  0  0  5  8  5  0  0  0
##   7   0  0  0  0  0  3 14  5  1  0
##   8   0  0  0  0  0  0  3 11  6  0
##   9   0  0  0  0  0  0  0  4 10  3
##   10  3  0  0  0  0  0  0  0  3 10
##
## $MisclassificationRate
## [1] 0.43
```

```
## Misclassification for the most probable path:
```

```
## [1] 0.56
```

The smoothing distribution is generally more accurate since it uses all the available data to update the predictions on the hidden states. This is while for making prediction on the hidden state at time t using filtering, we use only use observations up to time t.

The most probable path, obtained through the Viterbi algorithm, tries to find the hidden states that were most likely to occur. For this, the algorithm maximizes the likelihood of the robot being in a state at time t z_t given the next state z_{t+1}. In smoothing however, the states are updated after observing all the data (into the future), decreasing the error and thus resulting in a higher accuracy.

Question 6

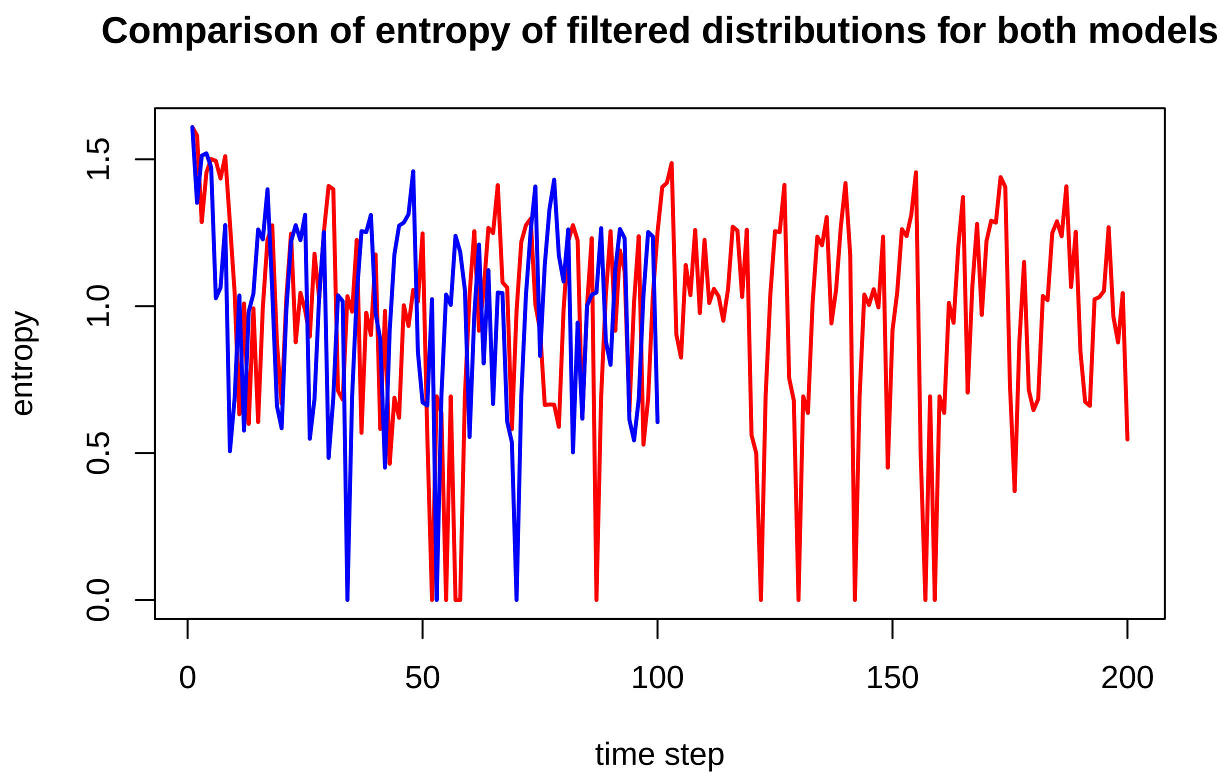
Shannon's entropy indicates how certain or uncertain we are about our obtained value which here refers to the hidden states. A value of 0 suggest the highest certainty (probability 1) of the robot being in a particular state and vice versa.

In this step, we calculate the entropy for each time step of the filtered distributions for both simulations above.

```r
entropy1 <- c()

for (i in 1:ncol(filtering)) {
  entropy1[i] <- entropy.empirical(filtering[,i])
}

entropy2 <- c()
for (i in 1:ncol(filtering2)) {
  entropy2[i] <- entropy.empirical(filtering2[,i])
}
```



```
## Average entropy for model 1 filtered distributions:  0.9821944
```

```
## Average entropy for model 2 filtered distributions:  0.9689585
```

Plot of the entropy for each time step for model 1 in *blue* and model 2 in *red* suggest that more observations do not lead to a higher accuracy in our predictions. We can observe from the plot and the average entropy for each model that the level of uncertainty remains constant regardless of the number of simulations.

Question 7

In order to make one-step ahead prediction, we can use the last step of the filtered distribution of the states, namely $Z_{t=100}$ as the coefficient for our AR(1) model and calculate its dot product with the prior transition probabilities to predict $Z_{t=101}$

```
coefs <- filtering[,100]
pred <- t(coefs) %*% transprob
```

```
## 1 step ahead prediction of the distribution is:
```

```
##            [,1]
##  [1,] 0.0000000
##  [2,] 0.0000000
##  [3,] 0.3532478
##  [4,] 0.5000000
##  [5,] 0.1467522
##  [6,] 0.0000000
##  [7,] 0.0000000
##  [8,] 0.0000000
##  [9,] 0.0000000
## [10,] 0.0000000
```

Appendix

```
library(HMM)
library(entropy)
knitr::opts_chunk$set(echo = TRUE)

# Question 1

states <- symbols <- 1:10
symbols <- c("A","B","C","D","E","F","G","H","I","J")
emitprob <- matrix(c(.2,.2,.2,0,0,0,0,0,.2,.2,
                   .2,.2,.2,.2,0,0,0,0,0,.2,
                   .2,.2,.2,.2,.2,0,0,0,0,0,
                   0,.2,.2,.2,.2,.2,0,0,0,0,
                   0,0,.2,.2,.2,.2,.2,0,0,0,
                   0,0,0,.2,.2,.2,.2,.2,0,0,
                   0,0,0,0,.2,.2,.2,.2,.2,0,
                   0,0,0,0,0,.2,.2,.2,.2,.2,
                   .2,0,0,0,0,0,.2,.2,.2,.2,
                   .2,.2,0,0,0,0,0,.2,.2,.2), ncol = 10)


transprob <- matrix(c(.5,0,0,0,0,0,0,0,0,.5,
                      .5,.5,0,0,0,0,0,0,0,0,
                    0,.5,.5,0,0,0,0,0,0,0,
                    0,0,.5,.5,0,0,0,0,0,0,
                    0,0,0,.5,.5,0,0,0,0,0,
                    0,0,0,0,.5,.5,0,0,0,0,
                    0,0,0,0,0,.5,.5,0,0,0,
                    0,0,0,0,0,0,.5,.5,0,0,
                    0,0,0,0,0,0,0,.5,.5,0,
```

```r
                     0,0,0,0,0,0,0,0,.5,.5), ncol = 10)

hmm <- initHMM(States = states,
               Symbols = symbols,
               emissionProbs = emitprob,
               transProbs = transprob)


# Question 2

simulated <- simHMM(hmm, 100)

obs <- simulated$observation


# Question 3

# the forward step probabilities (alpha)
frwrd <- forward(hmm, obs)

# the backward step probabilities (beta)
bkwrd <- backward(hmm, obs)

# Probabilities are converted back from log-scale
backward <- exp(bkwrd)
forward <- exp(frwrd)

# obtaining filtering distribution for each t
filtering <- matrix(0, 10, 100)

for (i in 1:100) {
  filtering[,i] <- forward[,i]/sum(forward[,i])
}

# obtaining smoothing distribution for each t
smoothing <- matrix(0, 10, 100)

for(i in 1:100) {
  smoothing[,i] <- (forward[,i] * backward[,i])/sum(forward[,i] * backward[,i])
}


# Using the viterbi algorithm we find the most probable path
mostProb <- viterbi(hmm, obs)
cat("The most probable path: \n")
mostProb


# Question 4
accuracy <- function(method_prob,sim) {
  conf <- table(sim, method_prob)
  misclass <- 1 - sum(diag(conf))/sum(conf)
  return(list(ConfusionMatrix=conf, MisclassificationRate=misclass))
```

```r
}


most_probable <- function(mat,sim) {
  method_prob <- apply(mat, 2, FUN = which.max)
  accuracy(method_prob, sim)
}


cat("Confusion matrix and miscalssification rate for smoothed distributions: \n")
most_probable(smoothing,simulated$states)
cat("Confusion matrix and miscalssification rate for filtered distributions: \n")
most_probable(filtering,simulated$states)


# Misclassification of the most probable path
1 - sum(mostProb == simulated$states)/length(simulated$states)


nSim <- 200

simulated2 <- simHMM(hmm, nSim)
obs2 <- simulated2$observation

frwrd2 <- forward(hmm, obs2)
bkwrd2 <- backward(hmm, obs2)

backward2 <- exp(bkwrd2)
forward2 <- exp(frwrd2)

filtering2 <- matrix(0, 10, nSim)

for (i in 1:nSim) {
  filtering2[,i] <- forward2[,i]/sum(forward2[,i])
}

smoothing2 <- matrix(0, 10, nSim)

for(i in 1:nSim) {
  smoothing2[,i] <- (forward2[,i] * backward2[,i])/sum(forward2[,i] * backward2[,i])
}

# Using the viterbi algorithm we find the most probable path

mostProb2 <- viterbi(hmm, obs2)
cat("The most probable path: \n")
mostProb2

####


cat("Confusion matrix and miscalssification rate for smoothed distributions: \n")
most_probable(smoothing2,simulated2$states)
```

```r
cat("Confusion matrix and miscalssification rate for filtered distributions: \n")
most_probable(filtering2,simulated2$states)

cat("Misclassification for the most probable path: \n")
1 - sum(mostProb2 == simulated2$states)/length(simulated2$states)


entropy1 <- c()

for (i in 1:ncol(filtering)) {
  entropy1[i] <- entropy.empirical(filtering[,i])
}

entropy2 <- c()
for (i in 1:ncol(filtering2)) {
  entropy2[i] <- entropy.empirical(filtering2[,i])
}



plot(1:200, entropy2, type="l", col="red", lwd=2,
     main = "Comparison of entropy of filtered distributions for both models",
     xlab = "time step", ylab = "entropy")
lines(1:100, entropy1, type="l", col="blue", lwd=2)


cat("Average entropy for model 1 filtered distributions: ", mean(entropy1),"\n")
cat("Average entropy for model 2 filtered distributions: ", mean(entropy2),"\n")


coefs <- filtering[,100]
pred <- t(coefs) %*% transprob



cat("1 step ahead prediction of the distribution is: \n")
t(pred)
```