Lab 1

Kristian Sikiric (krisi211)

The lab

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.

Assignment 1

First we were to build a HMM for the scenario described above, to do so, the transition model and the emission model had to be defined. These were defined as stochastic matrices containing the transition/emission probabilities between the states, were rows are the current state and the columns the state to move to.

The following matrix is the transistion probabilites between states:

```
##       [,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
```

The following matrix is the emission probabilites for the 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
```

The following are the states and symbols of the model:

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

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

Assignment 2

The following are simulations of 100 time steps.

Simulated states:

```
##   [1]  3  4  5  5  5  5  6  6  7  8  9  9 10 10  1  1  2  3  3  3  3  3  3
##  [24]  3  3  4  5  5  5  6  6  6  6  7  7  8  8  8  9  9  9  9 10 10  1  1
##  [47]  1  2  3  4  4  4  5  5  6  7  7  8  8  9  9 10  1  2  3  4  4  5  6
##  [70]  7  7  8  8  9 10 10  1  2  3  4  5  6  6  7  7  8  8  8  8  9 10 10
##  [93]  1  2  3  4  4  4  4  5
```

Simulated emissions (observations):

```
##   [1]  2  5  3  3  7  3  8  8  5  8 10  9  9 10  2  3  3  4  2  5  5  3  3
##  [24]  3  1  3  3  5  5  7  6  4  8  5  6 10  6 10 10  8  1  8  8  2  2 10
##  [47] 10  1  3  5  5  3  7  6  7  6  9  6  7  1  1  2  1  4  2  2  5  7  6
##  [70]  7  9 10  9  1  8  1  1  1  2  3  5  7  4  8  6  8  6  9  7 10  1  9
##  [93]  9  3  3  6  6  2  6  4
```

Assignment 3

Discarding the hidden states from the simulation above, we were now to compute the filter and smoothing probability distributions as well as the most probable path. The filter was calculated with the formula $\frac{\alpha(Z^t)}{\sum_{z^t}(\alpha(z^t))}$, alpha was gained from the function 'forward' in the hmm package. The smoothing was calculated with $\frac{\alpha(Z^t)\beta(Z^t)}{\sum_{z^t}(\alpha(z^t)\beta(z^t))}$, were beta was gained from the backawrd function in the hmm package. The path was obtained using the viterbi algorithm.

The normalized probability distribution of the filter (first three time steps):

```
##       index
## states   1   2         3
##      1  0.2 0.0 0.0000000
##      2  0.2 0.0 0.0000000
##      3  0.2 0.4 0.2222222
##      4  0.2 0.4 0.4444444
##      5  0.0 0.2 0.3333333
##      6  0.0 0.0 0.0000000
##      7  0.0 0.0 0.0000000
##      8  0.0 0.0 0.0000000
##      9  0.0 0.0 0.0000000
##      10 0.2 0.0 0.0000000
```

The normalized probability distribution of the smoothing (first three time steps):

```
##       index
## states         1          2         3
##      1  0.0000000 0.00000000 0.0000000
##      2  0.2307692 0.00000000 0.0000000
##      3  0.4615385 0.46153846 0.1538462
##      4  0.3076923 0.46153846 0.6153846
##      5  0.0000000 0.07692308 0.2307692
##      6  0.0000000 0.00000000 0.0000000
##      7  0.0000000 0.00000000 0.0000000
##      8  0.0000000 0.00000000 0.0000000
```

```
##      9  0.0000000 0.00000000 0.0000000
##      10 0.0000000 0.00000000 0.0000000
```

The path:

```
##   [1]  2  3  3  4  5  5  6  6  7  8  9 10  1  1  1  1  1  2  2  3  3  3  3
##  [24]  3  3  3  3  3  4  5  5  5  6  6  7  8  8  8  8  8  9  9 10  1  1  1
##  [47]  1  1  2  3  3  4  5  5  5  6  7  8  9 10  1  1  1  2  2  3  4  5  5
##  [70]  6  7  8  8  9 10  1  1  1  2  3  4  5  5  6  6  6  7  8  9 10  1  1
##  [93]  1  2  3  4  4  4  4  4
```

Assignment 4

Now, the accuracy of the filter, smoothing and path were to be calculated. The results are as follows:

Accuracy of the path:

```
## [1] 0.49
```

Accuracy of filter:

```
## [1] 0.63
```

Accuracy of smoothing:

```
## [1] 0.75
```

```
## [1] "Accuracy of my implementaiton of forward:  0.63"
```

```
## [1] "Accuracy of my implementation of backward:  0.75"
```

Assignment 5

Since the smoothed distribution uses all the time steps for deciding were the robot is, it is logical that it usually is more efficient than the filter distribution and the path since they only use the time steps up to that point of prediction. The smoothed distribution hence have more information to use when predicting.

Assignment 6

The entropy for 100 simulated time steps.

```
## [1] 5.520148
```

The entropy for 200 simulated time steps.

```
## [1] 6.21909
```

Since the entropy score is higher for 200 simulated time steps than for 100 simulated time steps, meaning that more simulated time steps gives more uncertainty. Therefor more observations does not mean that we better know were the robot is.

Assignment 7

```
##        [,1]
##  [1,] 0.000
##  [2,] 0.000
##  [3,] 0.125
##  [4,] 0.375
##  [5,] 0.375
##  [6,] 0.125
##  [7,] 0.000
##  [8,] 0.000
##  [9,] 0.000
## [10,] 0.000
```

Appendix - Code

```
library(HMM)
###### Functions ########
my_forward = function(hmm, obs){
  startProbs = rep(0.1,10)
  a = matrix(NA,nrow =100,ncol = 10)
  for(i in states){
    a[1,i] = emissionprob[i,obs[1]]*startProbs[i]
  }
  for(t in 2:100){
    for(i in states){
      a[t,i] = emissionprob[i,obs[t]]*sum(a[t-1,]*transprob[,i])
    }
  }
  return(a)
}

my_backward = function(obs){
  b = matrix(NA,nrow =100,ncol = 10)
  for(i in states){
    b[100,i]=1
  }
  for(t in 99:1){
    for(i in states){
      b[t,i]=sum(b[t+1,]*emissionprob[,obs[t+1]]*transprob[i,])
    }
  }
  return(b)
}
###### Assignment 1 ######
transprob = matrix(1:100,ncol = 10)*0
for(i in 1:10)
  for(j in 1:10)
    if(i == j | j == i +1)
    transprob[i,j] = 0.5
#Special case to make the path a ring
```

```
transprob[10,1] = 0.5

emissionprob = matrix(1:100,ncol = 10)*0
for(i in 1:10)
  for(j in 1:10)
    if(i == j | j == i +1 | j == i+2 | j == i-1 | j == i -2)
      emissionprob[i,j] = 0.2
#Some special cases
emissionprob[10,1] = 0.2
emissionprob[10,2] = 0.2
emissionprob[9,1] = 0.2
emissionprob[1,10] = 0.2
emissionprob[1,9] = 0.2
emissionprob[2,10] = 0.2

states = c("S1","S2","S3","S4","S5","S6", "S7", "S8", "S9", "S10")
symbols = c("s1","s2","s3","s4","s5","s6", "s7", "s8", "s9", "s10")

hmm = initHMM(States = states,
              Symbols = symbols,
              transProbs = transprob,
              emissionProbs = emissionprob)

###### Assignment 2 ######
sim = simHMM(hmm,100)
sim$states
sim$observation

###### Assignment 3 ######
f = exp(forward(hmm,sim$observation))
b = exp(backward(hmm,sim$observation))

filtering = f/sum(f)
smoothing = (f*b)/sum(f*b)

#Normalized probability distributions
prob.dist.filter = prop.table(filtering,2)
prob.dist.smoothing = prop.table(smoothing,2)

path = viterbi(hmm, sim$observation)

###### Assignment 4 ######
path.tab = table(Actual = sim$states,Predicted = path)
path.acc = sum(diag(path.tab))/sum(path.tab)

predict.filter = apply(prob.dist.filter,2,which.max)
filter.bool = states[predict.filter] == sim$states
filter.acc = sum(filter.bool == TRUE)/length(filter.bool)

predict.smoothing = apply(prob.dist.smoothing,2,which.max)
smoothing.bool = states[predict.smoothing] == sim$states
smoothing.acc = sum(smoothing.bool == TRUE)/length(smoothing.bool)
```

```
###### Compare built in forward/backward with my forward/backward #####
a = my_forward(hmm,sim$observation)
a = a/sum(a)
maxa = apply(a,1,which.max)
maxa.bool = states[maxa] == sim$states
acc = sum(maxa.bool == TRUE)/length(maxa.bool)
print(paste("Accuracy of my implementaiton of forward: ",acc))

b = my_backward(sim$observation)
b = (b*a)/sum(b*a)
maxb = apply(b,1,which.max)
maxb.bool = states[maxb] == sim$states
acc = sum(maxb.bool == TRUE)/length(maxb.bool)
print(paste("Accuracy of my implementation of backward: ",acc))

###### Assignment 5 ######
#Rerun 1-4 with different sample sizes...
###### Assignment 6 ######
library(entropy)
sim = simHMM(hmm,200)
f.200 = exp(forward(hmm,sim$observation))
filtering.200 = f.200/sum(f.200)
prob.dist.filter.200 = prop.table(filtering.200,2)


entropy.empirical(prob.dist.filter)
entropy.empirical(prob.dist.filter.200)
###### Assignment 7 ######
res = transprob %*% prob.dist.filter[,100]
```