Individual Lab report 2 - William Anzén

2020-09-23

Question 1:

Build a hidden Markov model (HMM) for the scenario described above.

We initiate the HMM by setting up by defining the states and setting up transition & emission matrices. Initial state is set to a uniform distribution since none is set in the initHMM function.

```
set.seed(12345)
states = 1:10
symbols = c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10")

tmatrix = matrix(0,10,10) #transmission matrix
ematrix = matrix(0,10,10) #emission matrix

tmatrix=0.5*diag(10)
ematrix=0.2*diag(10)
for(i in 1:10){
  tmatrix[i,i%%10+1]=0.5
  max=i+2
  min=i-2
  int=min:max%%10
  int[int=0]=10
  ematrix[i,int]=0.2
}
tmatrix
```

```
##       [,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
```

```
ematrix
```

```
##       [,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   0.0
##  [2,]  0.2  0.2  0.2  0.2  0.0  0.0  0.0  0.0  0.0   0.0
##  [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   0.0
##  [9,]  0.2  0.0  0.0  0.0  0.0  0.0  0.2  0.2  0.2   0.0
## [10,]  0.2  0.2  0.0  0.0  0.0  0.0  0.0  0.2  0.2   0.2
```

```
hmm <- initHMM(States=states, Symbols=symbols, transProbs=tmatrix, emissionProbs=ematrix)
```

Question 2:

Simulate the HMM for 100 time steps.

```
simuHMM <- simHMM(hmm, 100)
simuHMM$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
```

```
simuHMM$observation
```

```
##   [1] "X7"  "X1"  "X9"  "X1"  "X2"  "X9"  "X1"  "X2"  "X3"  "X4"  "X5"  "X4"
##  [13] "X2"  "X3"  "X2"  "X6"  "X6"  "X5"  "X4"  "X3"  "X5"  "X5"  "X8"  "X9"
##  [25] "X1"  "X9"  "X9"  "X10" "X3"  "X2"  "X4"  "X5"  "X3"  "X2"  "X6"  "X6"
##  [37] "X4"  "X7"  "X7"  "X6"  "X5"  "X9"  "X7"  "X1"  "X10" "X9"  "X2"  "X1"
##  [49] "X3"  "X3"  "X6"  "X5"  "X4"  "X7"  "X7"  "X9"  "X9"  "X6"  "X8"  "X9"
##  [61] "X1"  "X2"  "X9"  "X9"  "X8"  "X3"  "X9"  "X4"  "X2"  "X3"  "X2"  "X4"
##  [73] "X5"  "X4"  "X4"  "X2"  "X4"  "X6"  "X4"  "X6"  "X8"  "X6"  "X7"  "X6"
##  [85] "X8"  "X8"  "X7"  "X9"  "X8"  "X10" "X1"  "X9"  "X3"  "X3"  "X9"  "X2"
##  [97] "X3"  "X9"  "X4"  "X1"
```

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.

```
smoothingFunc <- function(hmm, obs, n){
  alpha <- exp(forward(hmm,obs))
  beta <- exp(backward(hmm,obs))
  smoothing = matrix(0,10,n)
  for(j in 1:n){
    smoothing[,j] <- beta[,j]*alpha[,j]/sum(beta[,j]*alpha[,j])
  }
  return(smoothing)
}

filteringFunc <- function(hmm, obs,n){
  alpha <- exp(forward(hmm,obs))
  filtering = matrix(0,10,n)
  for(j in 1:n){
    filtering[,j] <- alpha[,j]/sum(alpha[,j])
  }
  return(filtering)
}

smoothing = smoothingFunc(hmm,simuHMM$observation,100)
filtering = filteringFunc(hmm,simuHMM$observation,100)

probPath <- viterbi(hmm,simuHMM$observation)
```

Question 4:

Compute the accuracy of the filtered and smoothed probability distributions, and of the most probable path. That is, compute the percentage of the true hidden states that are guessed by each method.

```
smoothPath <- apply(smoothing, MARGIN=2,FUN=which.max)
filterPath <- apply(filtering, MARGIN = 2, FUN=which.max)

accuracyFunc <- function(data, inputdata){
    n=length(inputdata)
    nTrue <- sum(data == inputdata)
    return(nTrue/n)
  }

probPathAcc <- accuracyFunc(probPath,simuHMM$states)
smoothPathAcc <- accuracyFunc(smoothPath, simuHMM$states)
filterPathAcc <- accuracyFunc(filterPath, simuHMM$states)

probPathAcc
```

```
## [1] 0.53
```

```
smoothPathAcc
```

```
## [1] 0.69
```

```
filterPathAcc
```

```
## [1] 0.55
```

We see that the smoothing probability has the highest accuracy and the most probable path has the lowest accuracy. The filtering has a accuracy quite close the the most probable path.

Question 5:

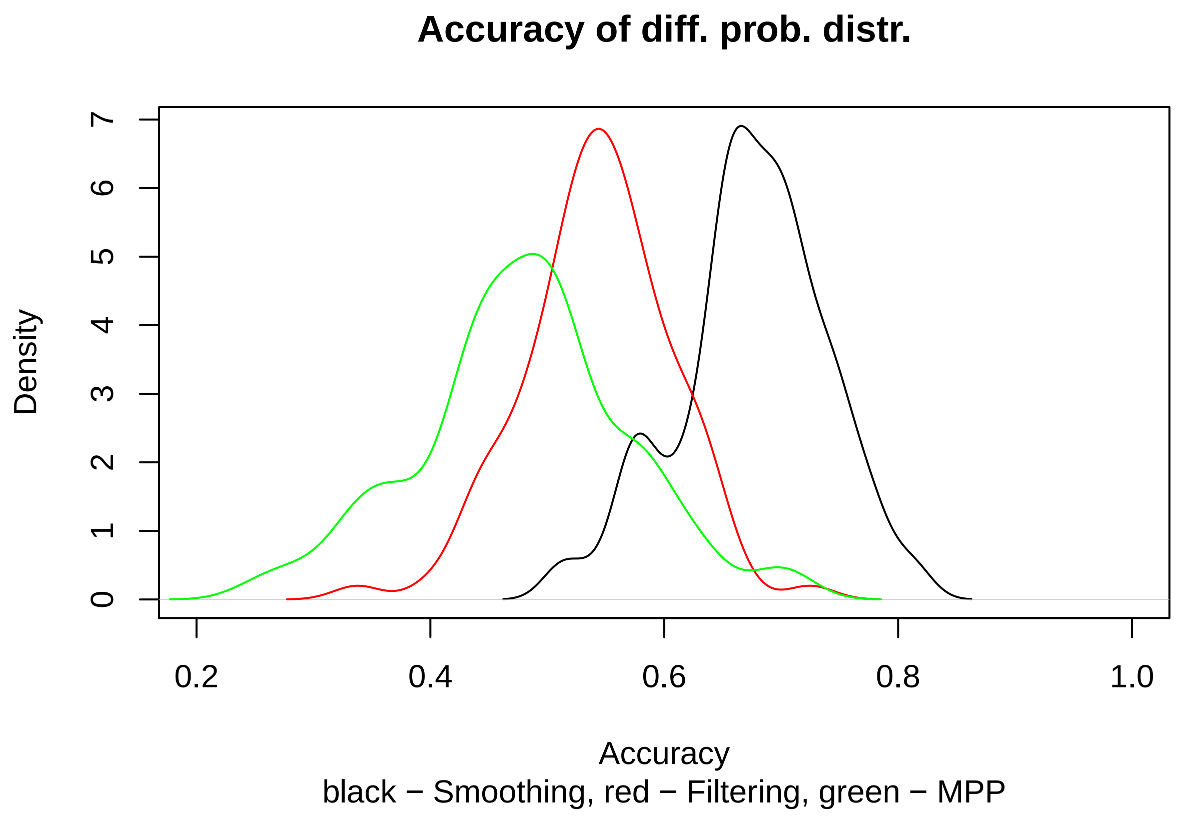
Repeat the previous exercise with different simulated samples. In general, the smoothed distributions should be more accurate than the filtered distributions. Why ? In general, the smoothed distributions should be more accurate than the most probable paths, too. Why ?

```r
AccSim <- function(hmm, nobs){
  sim <- simHMM(hmm, nobs)
  smooths <- smoothingFunc(hmm, sim$observation, nobs)
  filts <- filteringFunc(hmm, sim$observation, nobs)
  probpath <- viterbi(hmm, sim$observation)
  pathSmooth <- apply(smooths, MARGIN = 2, FUN=which.max)
  pathfilts <- apply(filts, MARGIN = 2, FUN=which.max)

  accpp <- accuracyFunc(probpath,sim$states)
  accfilts <- accuracyFunc(pathfilts,sim$states)
  accSmooth <- accuracyFunc(pathSmooth, sim$states)
  tempMatrix = matrix(c(accpp,accfilts,accSmooth),1,3)
  return(tempMatrix)
}

set.seed(1234566)
nSim <- 100
AccMatrix20<- matrix(0,nSim,3)
AccMatrix40 <- matrix(0,nSim,3)
AccMatrix80 <- matrix(0,nSim,3)
for(i in 1:nSim){
  AccMatrix40[i,] <- AccSim(hmm, 40)
  AccMatrix20[i,] <- AccSim(hmm,20)
  AccMatrix80[i,] <- AccSim(hmm,80)
}

plot(density(AccMatrix80[,3]) ,type="l", xlim=c(0.2,1),
     main="Accuracy of diff. prob. distr." ,xlab="Accuracy",
     sub="black - Smoothing, red - Filtering, green - MPP")
lines(density(AccMatrix80[,2]), type="l", col="red")
lines(density(AccMatrix80[,1]),type="l", col="green")
```

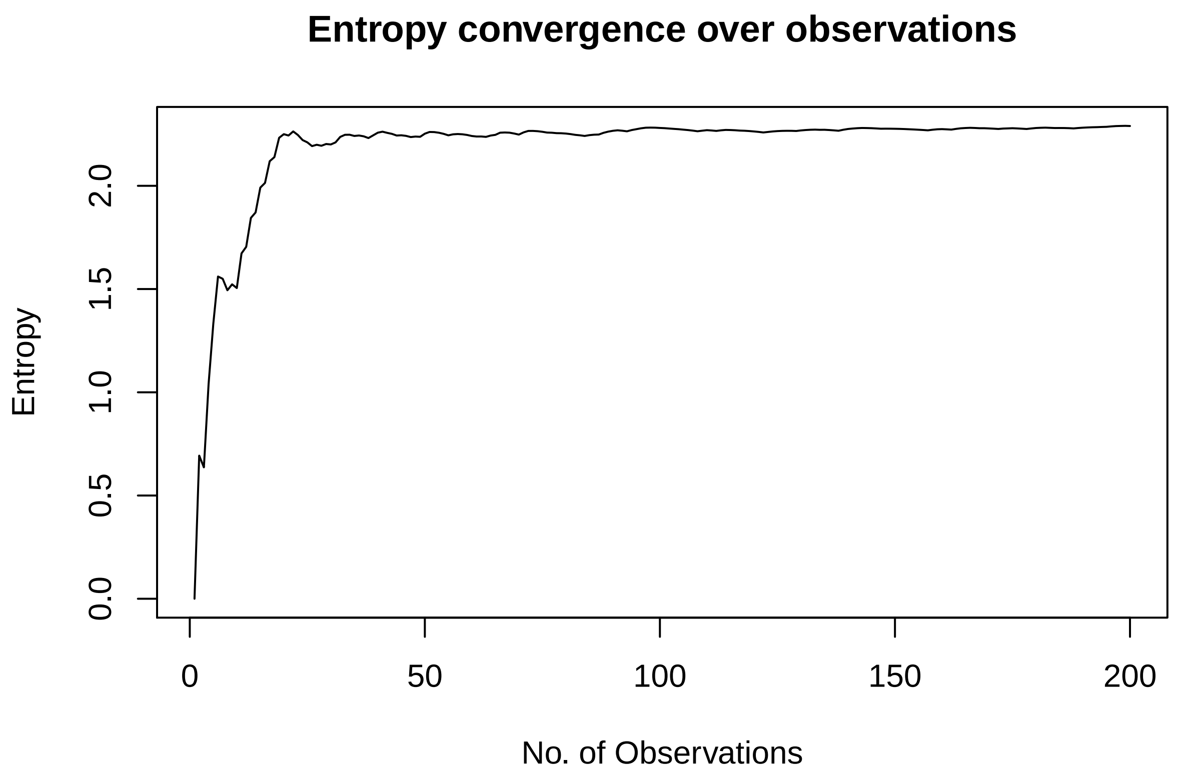


Since the smoothing takes all observations into account (backward and forward) the distribution has more information and is better at predicting the correct state. It makes sense that the filtering has a lower accuracy than the smoothing because it only relies on alpha, compared to smoothing relying on alpha and beta. As for the MPP which calculates the most probable path X timesteps will almost always perform worse than e.g. the smoothing which maximizes the marginal probability in each step.

Question 6:

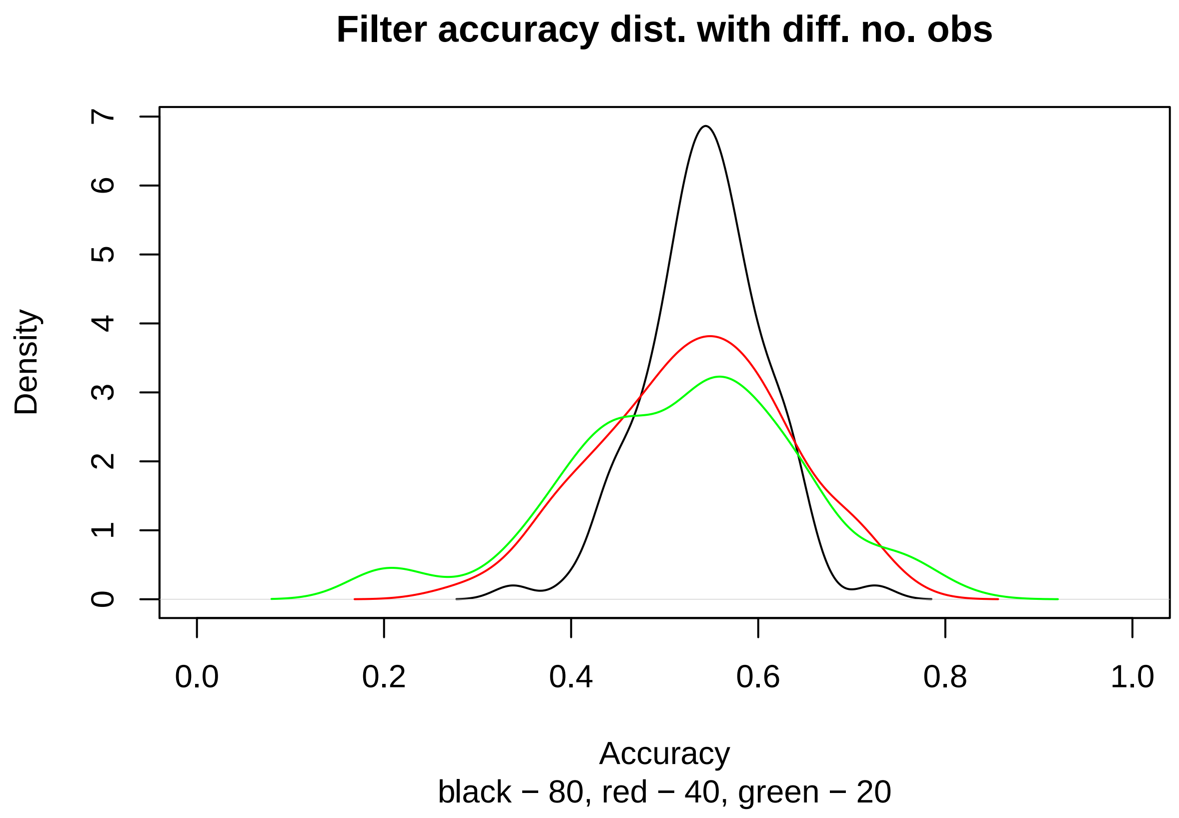
Is it true that the more observations you have the better you know where the robot is?

```
simHMM.200 <- simHMM(hmm,200)
filter.200 <- filteringFunc(hmm, simHMM.200$observation, 200)
filteredPath.200 <- apply(filter.200, MARGIN = 2 , FUN = which.max)
entropyFilt.200=rep(0,200)
for(i in 1:length(filteredPath.200)){
 entropyFilt.200[i]<- entropy.empirical(table(factor(filteredPath.200[1:i])))
}
plot(entropyFilt.200, type="l", ylab="Entropy", xlab="No. of Observations",
     main="Entropy convergence over observations")
```



As seen in the plot the entropy increases until approx. 30 observations where it converges toward 2.28. It seems that the more observations we have, the less we know about where the robot actually is because the entropy (disorder) increases. The plot below, containing accuracy distributions for different number of observations, confirms that we are not more certain about where the robot is as more observations does not increase the accuracy.

```r
plot(density(AccMatrix80[,2]), type="l", xlim=c(0,1),
     main="Filter accuracy dist. with diff. no. obs", xlab="Accuracy",
     sub="black - 80, red - 40, green - 20")
lines(density(AccMatrix40[,2]), type="l", col="red")
lines(density(AccMatrix20[,2]), type="l", col="green")
```



Question 7:

Consider any of the samples above of length 100. Compute the probabilities of the hidden states for the time step 101.

To compute step 101 simply multiply the prob. dist. for step 100 with the transition matrix.

```
smooth.101 <- t(smoothing[,100]%*%tmatrix)
smooth.101
```

```
##       [,1]
##  [1,] 0.00
##  [2,] 0.25
##  [3,] 0.50
##  [4,] 0.25
##  [5,] 0.00
##  [6,] 0.00
##  [7,] 0.00
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
```