Lab 2, Group Report

Linus Bergström, Isak Berntsson, Erik Jareman, Ludvig Knast

2022-09-17

```
#install.packages("HMM")
library(HMM)
```

```
## Warning: package 'HMM' was built under R version 4.1.3
```

```
#install.packages("entropy")
library(entropy)
```

Task 1

If the robot is in S1, it can stay or move to S2 with equal probability, giving the first transition matrix row: {.5, .5, 0, 0, 0, 0, 0, 0, 0, 0}. The same goes for the second row with the difference being that it starts in position 2: {0, .5, .5, 0, 0, ...}... For the emission matrix, we set the current state and the four closest states to 0.2 as the probabilities are equal (divided on five states). We execute the above mentioned steps for each row to create the transition- and emission matrices, and use these to build the HMM.

```
transition = matrix(0, 10, 10)
emission = matrix(0, 10, 10)
for (i in 0:9)
{
  # Set transition values to .5 for S(current) and S(current+1)
  transition[i+1, i:(i+1)%%10 +1] = 0.5
  # Set emission values to .2 for S(current) and the four closest columns
  emission[i+1, (i-2):(i+2)%%10 +1] = 0.2
}

transition
```

```
##       [,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
```

```
emission
```

```
##       [,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
```

```
#Build the HMM with the transition and emission matrices
states=1:10
symbols=1:10
HMM_model=initHMM(States=states, Symbols=symbols, transProbs=transition, emissionProbs=emission)
```

Task 2

Simulate a 100 timesteps for the Hidden Markov Model

```
sim = simHMM(HMM_model, 100)
sim
```

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

Task 3

```
observations = sim$observation

# Calculate alpha and beta to be used in filtering and smoothing (+log convert)
alpha = exp(forward(HMM_model, observations))
beta = exp(backward(HMM_model, observations))

#Calculate filtering- and smoothing probability distributions
filtered_dist = prop.table(alpha, 2)
```

```
smooth_dist = prop.table(alpha*beta, 2)

# Compute the most probable path using the Viterbi algorithm
most_probable_path = viterbi(HMM_model, observations)



round(head(t(filtered_dist)),3)
```

```
##      states
## index     1     2     3     4     5     6 7 8 9  10
##     1 0.200 0.200 0.200 0.200 0.000 0.000 0 0 0 0.2
##     2 0.222 0.222 0.222 0.222 0.111 0.000 0 0 0 0.0
##     3 0.000 0.000 0.000 0.500 0.375 0.125 0 0 0 0.0
##     4 0.000 0.000 0.000 0.364 0.636 0.000 0 0 0 0.0
##     5 0.000 0.000 0.000 0.000 0.611 0.389 0 0 0 0.0
##     6 0.000 0.000 0.000 0.000 1.000 0.000 0 0 0 0.0
```

```
round(head(t(smooth_dist)),3)
```

```
##      states
## index 1     2     3     4     5 6 7 8 9 10
##     1 0 0.182 0.455 0.364 0.000 0 0 0 0  0
##     2 0 0.000 0.364 0.545 0.091 0 0 0 0  0
##     3 0 0.000 0.000 0.727 0.273 0 0 0 0  0
##     4 0 0.000 0.000 0.364 0.636 0 0 0 0  0
##     5 0 0.000 0.000 0.000 1.000 0 0 0 0  0
##     6 0 0.000 0.000 0.000 1.000 0 0 0 0  0
```

```
most_probable_path
```

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

Task 4

```
guesses = matrix(nrow=100,ncol=3)
colnames(guesses) = c("filtered", "smoothed", "viterbi")
for (i in 1:100){
  filtered_guess = symbols[ which.max(filtered_dist[,i])]
  smooth_guess = symbols[ which.max(smooth_dist[,i])]
  guesses[i,1:2] = c(filtered_guess, smooth_guess)
}
guesses[,3] = most_probable_path
acc = colMeans(guesses == sim$states)
acc
```

```
## filtered smoothed  viterbi
##     0.50     0.70     0.55
```

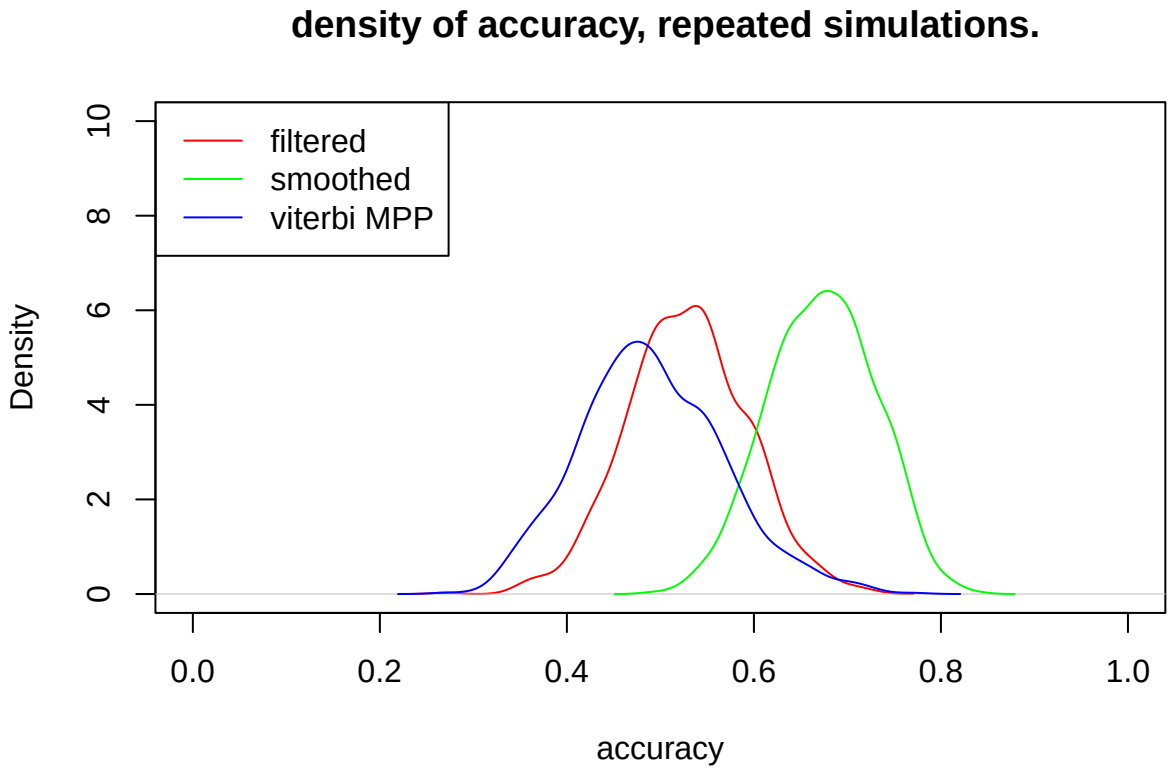
The smoothed distribution utilizes future observations as well as historical observations, whilst the filtered distribution only utilizes the latter. The smoothed distribution has more data to base its probability distribution on.

Smoothed distribution is more accurate than the Viterbi-path for both of the data sets. The reason is that the most probable path has to follow a path that is possible, meaning that the robot can not skip states, it can only stay or move forward for each predicted state. Due to this dependency, the most probable path approach can end up in situations where it would make sense to make a prediction that is two steps forward, but the dependency only allows for predictions one step forward.

Task 5

```
acc_mat = matrix(ncol=4, nrow=0)
colnames(acc_mat) = c("NumSteps", "filtered_acc", "smoothed_acc", "viterbi_acc")

nSimulations = 1000
simLength = 100
for  ( iter in 1:nSimulations){


  simulated_path = simHMM(HMM_model,simLength)
  obs = simulated_path$observation

  f = forward(HMM_model,obs)

  filtered = prop.table(t(exp(f)),margin=1)


  b = backward(HMM_model, obs)

  smoothed = prop.table(t(exp(f + b)) ,margin=1)




  guesses = matrix(nrow=simLength,ncol=3)
  colnames(guesses) = c("filtered", "smoothed", "viterbi")

  for (i in 1:simLength){
    filtered_guess = symbols[ which.max(filtered[i,])]

    smoothed_guess = symbols[ which.max(smoothed[i,])]

    guesses[i,1:2] = c(filtered_guess, smoothed_guess)
  }

  most_probable_path = viterbi(HMM_model,obs)

  guesses[,3] = most_probable_path
```

```
  acc = colMeans(guesses == simulated_path$states)

  acc_mat = rbind(acc_mat, c(simLength,acc))


  }


plot(density(acc_mat[,2]), col ="red",xlab="accuracy", xlim=c(0,1), ylim=c(0,10), main= " density of ac
lines(density(acc_mat[,3]), col ="green")
lines(density(acc_mat[,4]), col ="blue")

legend("topleft",legend = c("filtered", "smoothed", "viterbi MPP"), col=c("red", "green", "blue"), lty =
```

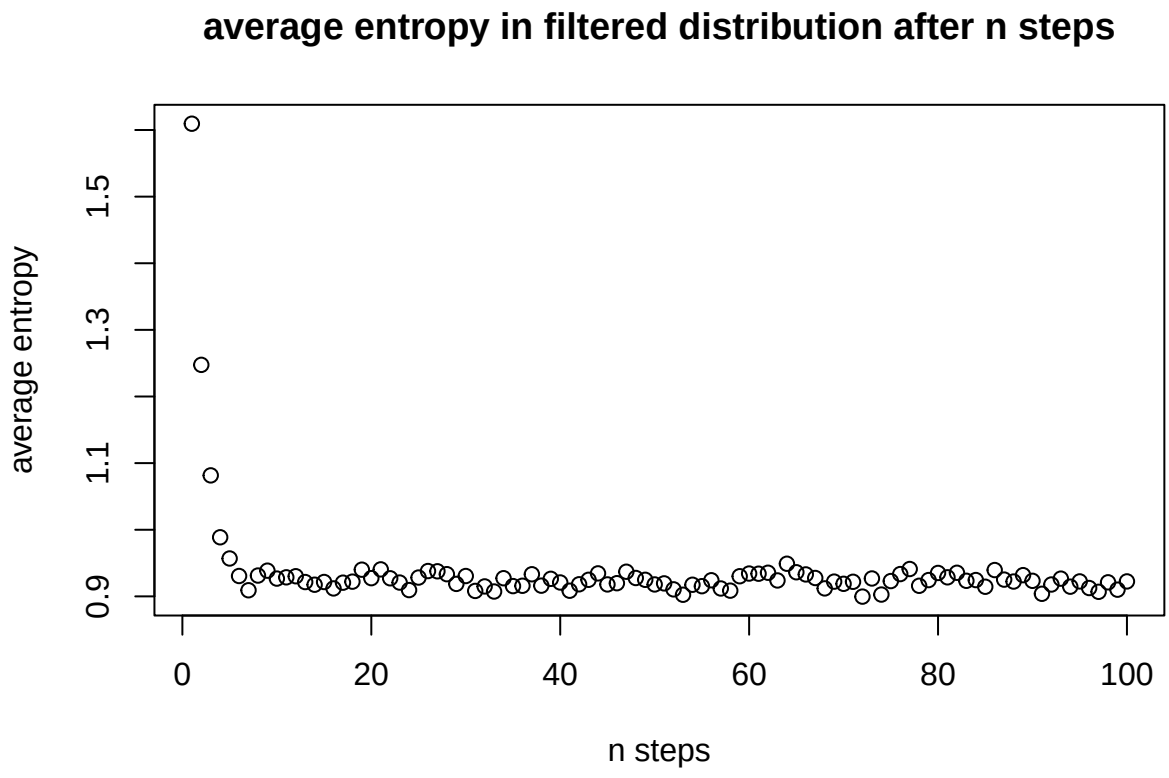


The smoothed distribution takes future as well as historical data into account and thus has more information available than the the filtered which only looks backwards. This means the smoothed distribution is more accurate.

The most probable path has the be a valid path which is a restriction not placed on the smoothed or filtered distributions.

Task 6

```r
nSimulations = 1000
simLength = 100

#could have reused simulations from previous task
mat_ents = matrix(nrow=simLength, ncol=nSimulations)

for ( i in 1:nSimulations){
  simulated_path = simHMM(HMM_model,simLength)
  obs = simulated_path$observation

  f = forward(HMM_model,obs)

  filtered = prop.table(t(exp(f)),margin=1)
  entropy_vec = apply(filtered,1, entropy.empirical )
  mat_ents[,i] = entropy_vec



}

ent_means = apply(mat_ents,1,mean)
plot(x=1:simLength, y = ent_means, main="average entropy in filtered distribution after n steps", xlab=
```



There is a sharp decline in entropy in the first (~3) steps. After that the entropy seems the be relatively stable. Lower entropy means there is less uncertainty regarding the state of the robot, i.e. our predictions are more confident. This means that it does not always become easier to infer the location of the robot as

time passes.

Task 7

```
sim = simHMM(HMM_model, 100)

f = forward(HMM_model, sim$observation)
filtered = prop.table(exp(t(f)),1)

prob_101 = filtered[100,] %*% transition


prob_101 = filtered[100,] %*% transition
prob_101 = round(rbind(prob_101,filtered[100,]), digits=3)
rownames(prob_101)= c("Inferred Probabilty. 101", "Filtered dist. 100")
colnames(prob_101) = 1:10
prob_101[2:1,]
```

```
##                          1 2 3 4 5     6     7     8     9    10
## Filtered dist. 100       0 0 0 0 0 0.056 0.251 0.444 0.249 0.000
## Inferred Probabilty. 101 0 0 0 0 0 0.028 0.153 0.347 0.347 0.125
```

Task	Contributors
1	Linus Bergström
2	Linus Bergström & Ludde Eriksson-Knast
3	Linus Bergström & Isak Berntsson
4	Linus Bergström
5	Isak Berntsson & Erik Jareman
6	Isak Berntsson & Erik Jareman & Ludde Eriksson-Knast
7	Isak Berntsson