

# 732A96 Lab 2

*Emil K Svensson*

*14 September 2017*

## Question 1

```
states <- paste("z",1:10,sep = "")
symbols <- paste("s",1:10,sep = "")
start <- rep(0.1,10)

#Just to see the structure
#initHMM(states, symbols)

# The transition probabilities, since we only can move
trans <- matrix(0,ncol = 10, nrow = 10)
diag(trans) <- 0.5
diag(trans[, -1]) <- 0.5
trans[10, 1] <- 0.5
#trans[1,10] <- 0.5

# Making sure the probabilities sum to 1 in each row
#apply(trans, MARGIN = 1, FUN = sum)

emission <- matrix(0,ncol = 10, nrow = 10)
diag(emission) <- 0.5

# The Emission probabilities are our uncertainty in the position of the robot.

# Brute force set up, Nick, Rasmus and Sascha probably has a nicer ways of solving this
diag(emission) <- 0.2
diag(emission[, -1]) <- 0.2
diag(emission[-1,]) <- 0.2
diag(emission[c(-1:-2),]) <- 0.2
diag(emission[, c(-1:-2)]) <- 0.2
emission[10, 2] <- 0.2
emission[2, 10] <- 0.2
emission[2, 10] <- 0.2
emission[1, 10] <- 0.2
emission[9:10, 1] <- 0.2
emission[1, 9] <- 0.2
```

```

# The different places the robot can be in
symbols

## [1] "s1" "s2" "s3" "s4" "s5" "s6" "s7" "s8" "s9" "s10"

# The Hidden States
states

## [1] "z1" "z2" "z3" "z4" "z5" "z6" "z7" "z8" "z9" "z10"

# Where they start
start

## [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

# Transition matrix is the probabilities the robot will move
trans

##      [,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 matrix states the uncertainties we have about the robots position
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

```

With everything defined we can initialize the HMM-robot.

```
robot <- initHMM(states,symbols,start,trans,emission)
```

## Question 2

The function below is built for answering Questions 2 to 7

```

mrRobot <- function(hmm,simobs = 100, what = "acc"){

  # Simulation the robot simobs times
  sim_rob <- simHMM(hmm,length = simobs)

```

```

#extract the observations (x_t)
observations <- sim_rob$observation

#extract the hidden states z_t
state_place <- sim_rob$states

### Filter
# Forward function computes the filtering with log
log_filter <- forward(hmm = hmm, observation = observations)

# Remove the log-transformation
filter <- exp(log_filter)

# Normalizing
norm_filter <- prop.table(filter, margin = 2)

# Checking which probability in each column is the highest
most_prob_filter <- apply(norm_filter, MARGIN = 2, FUN = which.max)

# Accuracy for the filter
accuracy_filter <- sum(paste("z",most_prob_filter, sep = "")
                      == state_place) / length(state_place)

### Smoothed (in this package called the posterior)
smoothed <- posterior(hmm=hmm,observations)

# Normalizing
norm_smoothed <- prop.table(smoothed, margin = 2)

most_prob_smoothed <- apply(norm_smoothed, MARGIN = 2, FUN = which.max)

# Accuracy for the smoothed
accuracy_smoothed <- sum(paste("z",most_prob_smoothed, sep = "")
                       == state_place) / length(state_place)

#cat("The accuracy of the smoothed is",accuracy_smoothed)

### Most probable path (Viterbi)
mpp <- viterbi(hmm,observations)

accuracy_mpp <- sum(mpp == state_place) / length(state_place)
#cat("The accuracy of the Viterbi is",accuracy_mpp)

#Just logical statements on what to return.
if(what == "acc"){
return( c(accuracy_filter = accuracy_filter,
          accuracy_smoothed = accuracy_smoothed,

```

```

        accuracy_mpp = accuracy_mpp))
}

if(what == "filter"){
  return(norm_filter)
}

if(what == "smooth"){
  return(norm_smoothed)
}

if(what == "mpp"){
  return(mpp)
}
}

```

## Filter

```
mrRobot(hmm = robot,simobs = 100, what = "filter")
```

To much to print out, so run the code if you want to see the distribution

## Smoothed

```
mrRobot(hmm = robot,simobs = 100, what = "smooth")
```

To much to print out, so run the code if you want to see the distribution

## Most probable path

```
mrRobot(hmm = robot,simobs = 100, what = "mpp")
```

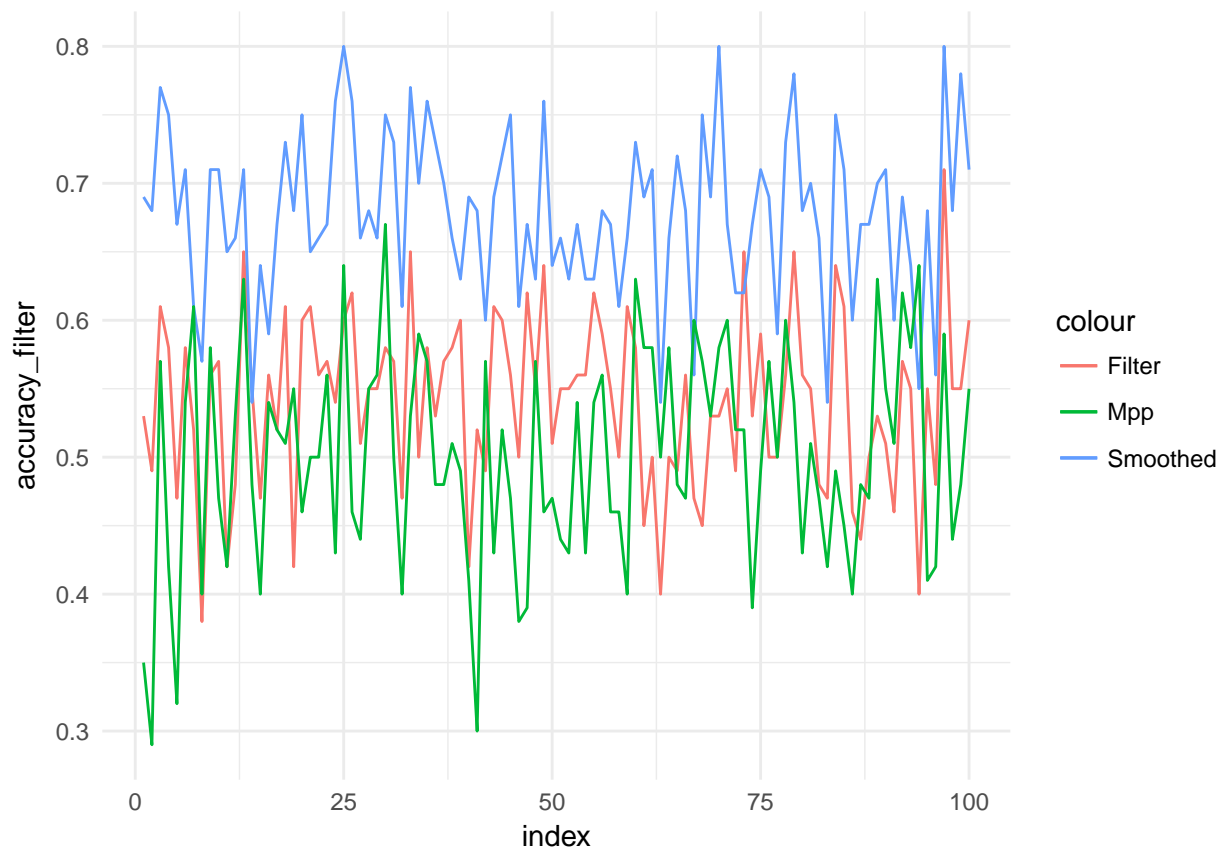
```
## [1] "z1" "z2" "z3" "z4" "z4" "z4" "z4" "z4" "z4" "z4" "z4" "z5"
## [12] "z5" "z6" "z7" "z8" "z9" "z10" "z1" "z1" "z1" "z2" "z3"
## [23] "z4" "z4" "z5" "z5" "z6" "z7" "z7" "z7" "z8" "z9" "z10"
## [34] "z1" "z1" "z1" "z1" "z2" "z3" "z4" "z5" "z5" "z5" "z6"
## [45] "z6" "z7" "z8" "z9" "z10" "z1" "z1" "z1" "z1" "z2" "z3"
## [56] "z4" "z5" "z5" "z6" "z7" "z8" "z9" "z10" "z1" "z1" "z1"
## [67] "z1" "z1" "z1" "z1" "z1" "z2" "z3" "z4" "z4" "z5" "z5"
## [78] "z6" "z7" "z7" "z7" "z8" "z8" "z9" "z10" "z1" "z1" "z1"
## [89] "z2" "z3" "z4" "z4" "z5" "z6" "z7" "z8" "z9" "z10" "z1"
## [100] "z1"
```

```
total_acc <- sapply(1:100,FUN = function(x){mrRobot(robot,100, what = "acc")})
```

```
total_acc <- as.data.frame(t(total_acc))
```

```
total_acc$index <- 1:100
```

```
ggplot(data = total_acc) + geom_line(aes(x=index,y=accuracy_filter , col = "Filter")) + geom_line(aes(
```



```
colMeans(total_acc[,-4])
```

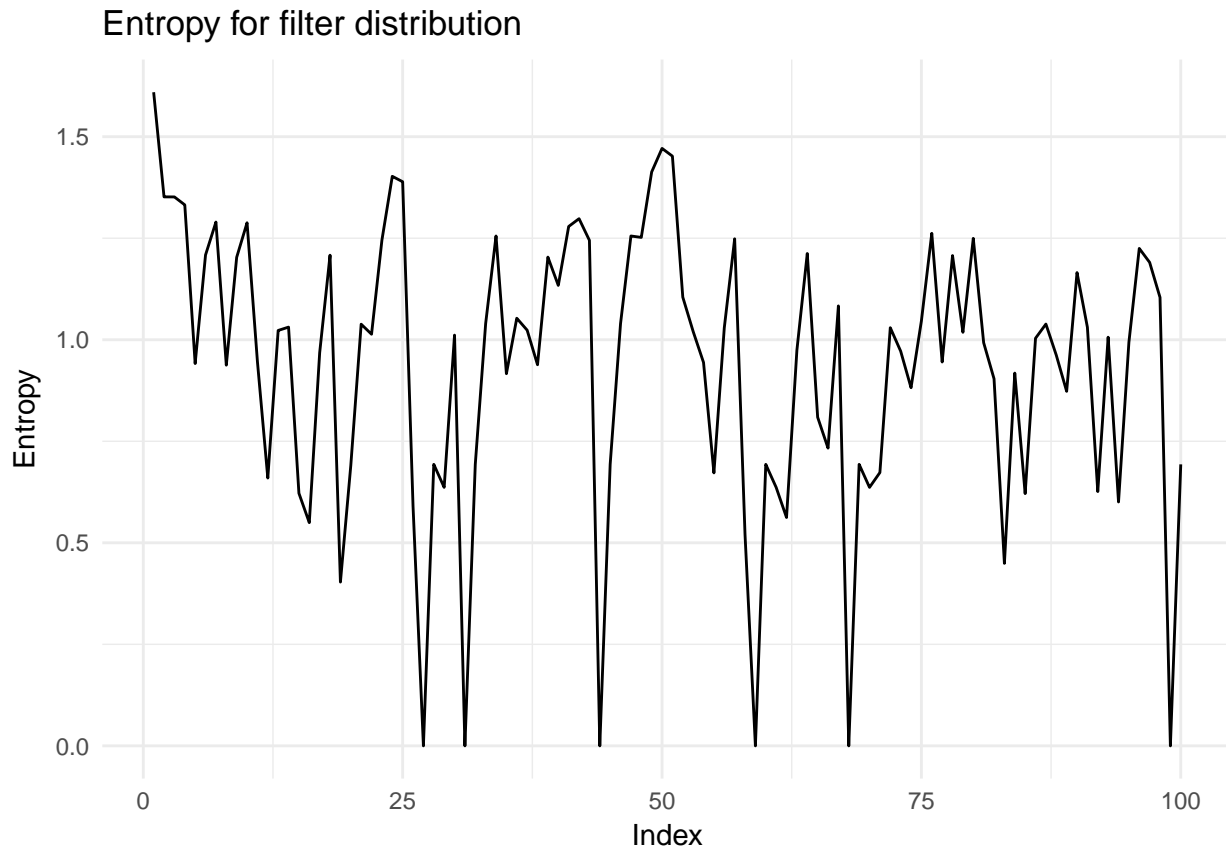
```
## accuracy_filter accuracy_smoothed accuracy_mpp
##          0.5407          0.6780          0.5005
```

## Question 6

```
# filter_data <- sapply(1,FUN = function(x){
#   mrRobot(robot,100, what = "filter")
# })

filter_data <- mrRobot(robot,100, what = "filter")
filter_entropy <- data.frame(Index = 1:100 ,
                             Entropy =
                               apply(filter_data, MARGIN = 2, FUN =entropy.empirical))

ggplot(data = filter_entropy, aes(x = Index, y = Entropy)) + geom_line() + ggtitle("Entropy for filter")
```



No, the entropy remains random even while increasing the number of observations added to the hmm. This is because it is markovian and only depends on the previous observation.

## Question 7

```
prior <- filter_data[,100] # The last information of the robot aka the prior
transition <- robot$transProbs
transition %*% prior
```

```
##
## from  [,1]
##  z1  0.00
##  z2  0.00
##  z3  0.00
##  z4  0.00
##  z5  0.00
##  z6  0.00
##  z7  0.25
##  z8  0.50
##  z9  0.25
##  z10 0.00
```

Since we have a markovian assumption that the only relevant state is the one we are in now and since all previous states feeds forward through  $z_{100}$  and provides it with information about previous observations and states we can just multiply  $z_{100}$  probabilities for each state with the transition probabilities to get at prediction of how  $z_{101}$  and  $s_{101}$  will be like. In this case we were asked to get the hidden states probabilities

but if you want the observation you can just do a  $\text{argmax}$  over these probabilities to get were the next state should be.