TDDE15 – Lab 2: Hidden Markov Models

hanma409 – Hannes Bengtsson

Exercise 1

To initiate the Hidden Markov Model for the robot we need to specify all the parameters for the *initHMM* function based on the lab-description. Please see the code below for further details.

Exercise 2

To simulate 100 time steps for the HMM we run the *simHMM* function for our Hidden Markov Model from exercise 1. Please see the code below for further details.

Exercise 3

To compute the filtered probability distributions we first need to calculate the α:s with the help of the *forward* function. Once we've done that we can calculate the filtered probability distributions. To compute the smoothed probability distributions we need to calculate the β:s with the help of the *backward* function. Once we've done that we can calculate the smoothed probability distributions based on α and β. Please see the code below for further details.

Exercise 4

By setting the seed to 12345, simulating 100 time steps, and predicting the robots true state we obtain the accuracies below. We note that the smoothed probability distributions results in the highest accuracy and that filtered probability distributions results in the lowest.

Description	Accuracy
Filtered probability distributions	0.53
Smoothed probability distributions	0.74
Most probable path	0.56

Exercise 5

By setting the seed to 67890, simulating 100 time steps, and predicting the robots true state we obtain the accuracies below. Here we note that the smoothed probability distributions results in the highest accuracy and that the most probable path results in the lowest.

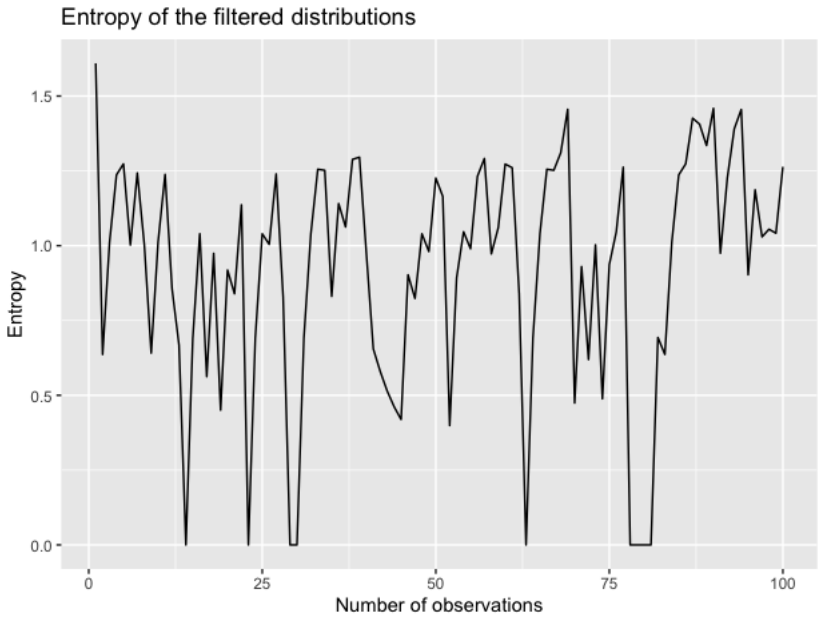
Description	Accuracy
Filtered probability distributions	0.57
Smoothed probability distributions	0.66
Most probable path	0.44

The filtering probability distribution for a given time step are calculated based on previous observations up to that given time step. The smoothed probability distribution for a given time step however, are calculated based on all available observations and thus also takes observations after the given time step into consideration. As a result smoothed probability distributions take more datapoints into consideration and thus have the opportunity to capture more information about the overall data structure and the distribution. Therefore, in general, the smoothed distributions are more accurate than the filtered distributions.

The smoothed distributions maximises the likelihood for each observations while the most probable path identifies the most probable path and thus needs to take the restrictions of the robots movement into consideration. Because of this, the smoothed distributions are more likely to be correct on individual predictions and thus in general gets in a higher accuracy.

Exercise 6

Entropy is a measure of the degree of randomness, therefore as entropy is higher it is more difficult to predict where the robot is. Plotting the entropy of the filtered distributions for different numbers of observations (1-100), we don't note any correlation between an increased number of observations and the entropy of the filtered distributions. In the graph below we note that at certain time steps the entropy is zero and at others it's not. At time steps when the entropy is zero, the true state of the robot is predicted with 100% certainty. Increased entropy corresponds to greater spread in the filtered probability distribution, and thus a greater uncertainty of the robots true state.



Exercise 7

To compute the probabilities of the hidden states for the time step 101 we use the filtered probabilities for the 100th time step in combination with a 50/50 chance of the robot moving or staying. If we look at the filtered probabilities for the 100th time step we observe the following probabilities for the second sample (from Exercise 5):

S1	S2	S3	S4	S5	S6	S7	S8
0.00	0.00	0.00	0.19	0.41	0.31	0.09	0.00

There is a 50/50 chance of the robot moving or staying and thus we get the following probabilities of the hidden states for the time step 101:

S1	S2	S3	S4	S5	S6	S7	S8
0.00	0.00	0.00	0.09	0.30	0.36	0.20	0.05

Code

Exercise 1

```
# Actual states
states <- c("S1","S2","S3","S4","S5","S6","S7","S8","S9","S10")
# Observed states via the device
symbols <- c("S1","S2","S3","S4","S5","S6","S7","S8","S9","S10")
# Starting probabilities for the states
start_probs <- rep(0.1,10)
# Create a 10x10 matrix with probability of movement
trans_matrix <- matrix(0, nrow = 10, ncol = 10)
diag(trans_matrix) <- 0.5
diag(trans_matrix[,-1]) <- 0.5
trans_matrix[10,1] <- 0.5
```

```
# Rows from, col to, i.e., trans_matrix[from,to], [row,col]
trans_matrix
trans_probs <- matrix(trans_matrix, nrow=10, ncol=10, byrow=FALSE,
dimnames = list(states,
states))
# Create a 10x10 matrix with probability of device result
device_matrix <- matrix(0, nrow = 10, ncol = 10)
emit_prob <- rep(0.2,5)
device_matrix[1,-4:-8] <- emit_prob
device_matrix[2,-5:-9] <- emit_prob
device_matrix[3,-6:-10] <- emit_prob
device_matrix[4,2:6] <- emit_prob
device_matrix[5,3:7] <- emit_prob
device_matrix[6,4:8] <- emit_prob
device_matrix[7,5:9] <- emit_prob
device_matrix[8,6:10] <- emit_prob
device_matrix[9,-2:-6] <- emit_prob
device_matrix[10,-3:-7] <- emit_prob
# Rows state, col emission, i.e., trans_matrix[Actual state, Emit state], [row,col]
device_probs <- matrix(device_matrix, nrow=10, ncol=10, byrow=FALSE,
dimnames = list(states,
states))
# Initiation of the Hidden Markov Model
robot <- initHMM(States = states, Symbols = symbols, startProbs = start_probs,
transProbs = trans_probs, emissionProbs = device_probs)
```

Exercise 2

```
# Robot simulations of 100 time steps
set.seed(12345)
robot_simulation <- simHMM(robot,100)
observations <- robot_simulation$observation
```

Exercise 3

```
# Calculating the filtered probability distributions using the forward function
# Exp takes the logarithmic output from the forward function and convert them into natural numbers
# Prob.table with margin 2 normalizes the filtered probability distributions for each time step
# i.e., normalizes each column in the filtered matrix
# Using prop.table returns the same result as when looping over all columns and taking each value
# in a column and divide it by the sum of all values in that column (col if margin = 2)
filtered <- prop.table(exp(forward(robot, observations)),margin=2)
# To compute the smoothed probability distributions we can use the posterior function
# or we can combine the forward and backward functions, multiplying alpha and beta
# from the forward function alpha is obtained and from backward beta is obtained
smoothed <- posterior(robot,observations)
smoothed <- prop.table(exp(forward(robot, observations))*exp(backward(robot,observations)),margin=2)
# Most probable path using the Viterbi algorithm
most_prob_path <- viterbi(robot, observations)
```

Exercise 4

```
# Obtain the state with the highest probability for each observation
filtered_pred <- rownames(filtered)[apply(filtered,2,which.max)]
smoothed_pred <- rownames(smoothed)[apply(smoothed,2,which.max)]
# Accuracy function
accuracy <- function(X,X1){
n <- length(X)
return(sum(diag(table(X,X1)))/n)
}
# Accuracy calculations
filtered_acc <- accuracy(robot_simulation$states,filtered_pred)
smoothed_acc <- accuracy(robot_simulation$states,smoothed_pred)
viterbi_acc <- accuracy(robot_simulation$states,most_prob_path)
```

Exercise 5

```
# Robot simulations of 100 time steps
set.seed(67890)
robot_simulation <- simHMM(robot,100)
observations <- robot_simulation$observation
# Calculating the filtered probability distributions using the forward function
# Exp takes the logarithmic output from the forward function and convert them into natural numbers
# Prob.table with margin 2 normalizes the filtered probability distributions for each time step
# i.e., normalizes each column in the filtered matrix
# Using prop.table returns the same result as when looping over all columns and taking each value
# in a column and divide it by the sum of all values in that column (col if margin = 2)
filtered <- prop.table(exp(forward(robot, observations)),margin=2)
# To compute the smoothed probability distributions we can use the posterior function
# or we can combine the forward and backward functions, multiplying alpha and beta
# from the forward function alpha is obtained and from backward beta is obtained
smoothed <- posterior(robot,observations)
smoothed <- prop.table(exp(forward(robot, observations))*exp(backward(robot,observations)),margin=2)
# Most probable path using the Viterbi algorithm
most_prob_path <- viterbi(robot, observations)
# Obtain the state with the highest probability for each observation
filtered_pred <- rownames(filtered)[apply(filtered,2,which.max)]
smoothed_pred <- rownames(smoothed)[apply(smoothed,2,which.max)]
# Accuracy calculations
filtered_acc <- accuracy(robot_simulation$states,filtered_pred)
smoothed_acc <- accuracy(robot_simulation$states,smoothed_pred)
viterbi_acc <- accuracy(robot_simulation$states,most_prob_path)
```

Exercise 6

```
# Computation of the entropy for the filtered distributions
e_filtered <-numeric(ncol(filtered))
for(i in 1:100){
e_filtered[i] <- entropy.empirical(filtered[,i])
}
# Save the entropy of the filtered distributions with the corresponding time step
data <- data.frame(entropy=e_filtered, observation=seq(1,100))
# Plot the entropy of the filtered distributions
ggplot(data = data)+
geom_line(aes(y=entropy,x=observation))+
ylab("Entropy") + xlab("Number of observations") +
labs(colour="Data type", title="Entropy of the filtered distributions")
```

Exercise 7

```
# Probabilities for the 100th time step combined with 50/50 chance of moving
# If the trans_matrix contains NA instead of 0 we get NA in the matrix multiplication
# Therefore we need to use 0 when initiating the trans_matrix for the HMM
filtered[,100]
prob_101 <- as.vector(filtered[,100] %*% trans_matrix)
```