Lab 2 - Advanced Machine Learning

Phillip Hölscher & Andreas C Charitos & Zijie Feng & Pascal Fahrni

9/30/2019

Libraries

Question 1 Build a HMM

The transition probability:

	a	b	c	d	e	f	g	h	i	j
a	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
b	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
c	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0
d	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0
e	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0	0.0
f	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0	0.0
g	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0	0.0
h	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0
i	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.5
j	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

The emission probability:

	a	b	c	d	e	f	g	h	i	j
a	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2
b	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2
c	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0	0.0
d	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.0
e	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.0
f	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0	0.0
g	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2	0.0
h	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2	0.2
i	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2	0.2
j	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.2

Question 2 Simulate the HMM for 100 time steps

```
## $states
##   [1] "i" "i" "i" "i" "j" "a" "b" "b" "b" "b" "c" "c" "d" "d" "d" "d" "d"
##  [18] "d" "d" "e" "f" "f" "g" "h" "i" "j" "j" "j" "a" "b" "b" "c" "c" "d"
##  [35] "d" "d" "e" "e" "e" "f" "g" "g" "h" "i" "j" "a" "b" "c" "c" "d" "e"
##  [52] "e" "f" "f" "g" "g" "h" "h" "h" "h" "i" "j" "j" "j" "j" "a" "a" "b"
##  [69] "b" "b" "b" "b" "c" "c" "c" "d" "e" "e" "e" "f" "g" "h" "h" "h" "h"
##  [86] "h" "i" "i" "i" "j" "j" "j" "a" "a" "a" "a" "a" "a" "b" "c"
##
## $observation
##   [1] "g" "j" "h" "j" "b" "c" "j" "c" "d" "d" "e" "d" "b" "c" "b" "f" "f"
##  [18] "e" "d" "c" "e" "e" "h" "i" "j" "i" "i" "j" "b" "j" "b" "e" "c" "b"
##  [35] "f" "f" "d" "g" "g" "f" "e" "i" "g" "j" "j" "c" "c" "a" "c" "c" "f"
##  [52] "e" "d" "g" "g" "i" "i" "j" "f" "i" "j" "b" "i" "i" "h" "a" "c" "b"
##  [69] "c" "d" "c" "b" "e" "d" "d" "b" "d" "f" "d" "f" "h" "j" "h" "g" "f"
##  [86] "f" "g" "h" "i" "j" "a" "i" "b" "b" "c" "i" "b" "j" "d" "a"
```

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.

Result question 3

In this and next part of the task we will filtered probability, smoothed probability distributions and calculate the most probable path. For this we will use a certain function for the respective algorithm and then do some calculations to create a prediction for the respective algorithm. This is then compared with the hidden state to produce a prediction accuracy. For the *filtering* do I use the *forward* algorithm, to compute the smoothed do I use the **posterior** function. We generate the alpha and beta with this function, which we need to run the *forward-backward* algorithm. With the viterbi algorihum do we compute the most probable path, this algorithm does not allow to jump between the states and chooses the next state of the current state.

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.

Hint: Note that the function *forward* in the *HMM* package returns probabilities in log scale. You may need to use the functions *exp* and *prop.table* in order to obtain a normalized probability distribution. You may also want to use the functions *apply* and *which.max* to find out the most probable states. Finally, recall that you can compare two vectors A and B elementwise as A==B, and that the function *table* will count the number of times that the different elements in a vector occur in the vector.

Result question 4

In this task part I calculate a prediction accruacy for all three algorithms. The calculation is well commented to follow the steps.

- Forward
- Smoothing
- Most proable path

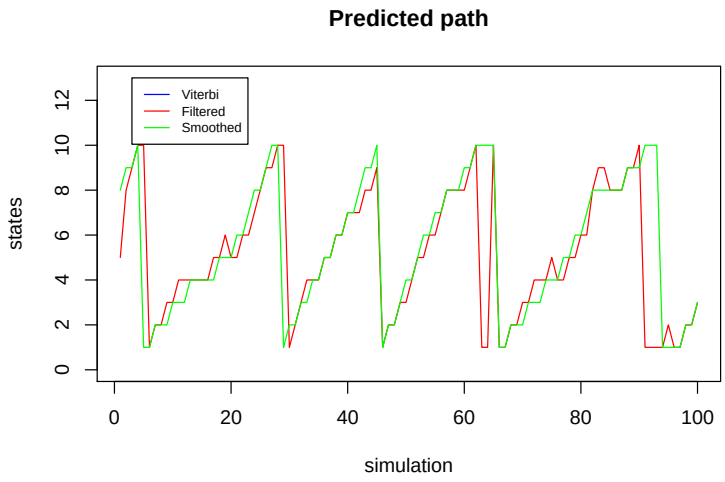
```
## The accuracy of the following methods
```

filtered	smoothed	viterbi
0.53	0.74	0.56

The prediction accuracy of the smoothed is highest and the other two methods are very similar.

In this visualization is the state prediction for each algorithm. In the first simulation it can already be seen that the start value is different. It has to be said that a similar process can be seen with all methods. In the case of the forward algorithm (filtered) it can be seen that this also goes back to states. That's one, you can see that the Foward algorithm goes by probabilities and doesn't consider the rules, because in the case description at the beginning of the lab you can see that the robot can only stop or move forward, this was initialized in the *transProbs*. Furthermore, jumps from forward from state 9 to state 1 can be seen, thus the forward skips the state 10. These phenomena do not occur with the viterbi algorithm, because it follows the most prob path which does excludes jumps.

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): NAs introduced by coercion
```



Question 5 Repeat the previous exercise with different simulated samples

The mean accuracies of filtering and smoothing paths with 100 observations and 50 iterations are

	filtering_accuracy	smoothing_accuracy	viterbi_accuracy
Mean	0.582549	0.6864706	0.5141176

The general accuracy of smoothing probability distribution is higher than the one of filtering's. Filtering is calculated by

$$p(Z_t|x_{0:t}) = \frac{\alpha(Z_t)}{\sum_{Z_t}\alpha(Z_t)} \quad ,$$

which means that the goal is to calculate the conditional probability of state Z_t given by the previous time series $x_{1:t}$, $t \leq 100$. On the other hand, smoothing is calculated by

$$p(Z_t|x_{0:t}) = \frac{\alpha(Z_t)\beta(Z_t)}{\sum_{Z_t}\alpha(Z_t)\beta(Z_t)}$$

which is the conditional probability of state Z_t given by the whole time series $x_{1:100}$. This might be the reason why smoothing distribution has such nice preformance.

We also predict the path by Viterbi algorithm for comparison, but the result from smoothed distribution is the highest as well. Another reason is that smoothed distribution predicts each hidden state which is the optimal result given by all the observations. However, it ignores the relation between hidden states and assumes that all time states are independent.

Question 6

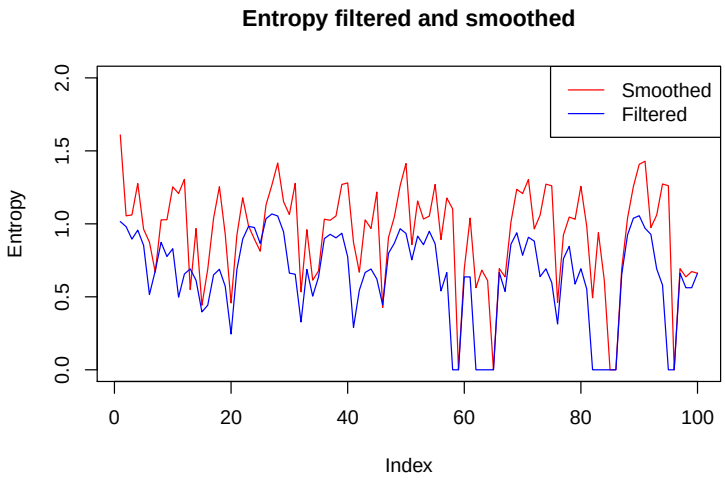
Is it true that the more observations you have the better you know where the robot is ?

Hint: You may want to compute the entropy of the filtered distributions with the function *entropy.empirical* of the package *entropy*.

Result question 6

The empirical entropy estimator is a plug-in estimator: in the definition of the Shannon entropy the bin probabilities are replaced by the respective empirical frequencies.

The empirical entropy estimator is the maximum likelihood estimator. If there are many zero counts and the sample size is small it is very inefficient and also strongly biased.



At first it´s important to explain what the entropy means, Entropy is a measure of uncertainty which goes up when the uncertainty is high and goes down when the uncertainty goes down. If the entropy is at 0, this means that there is virtually no uncertainty. This is the case several times with the filter method, less with the smoothing method. However, the question is whether increasing iterations makes us safer where the robot is. Due to the plot, this statement cannot be confirmed. For this a steadily sinking entropy would have to be present. However, even after 100 iterations it can still be seen that the entropy fluctuates strongly. This makes sense, because our problem case is stochastic, as we initialized at the transition probabilities. So desent matter which method we look at, the entropy will not convert to 0.

Question 7

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

I use the last state, which is the state 100, our prior inforamtion, to compute the prediction with the given transition probabilities.

Result question 7

```
## Probabilities of the hidden states for the time step 101:
```

```
##     [,1]
## a 0.0000
## b 0.1875
## c 0.5000
## d 0.3125
## e 0.0000
## f 0.0000
## g 0.0000
## h 0.0000
## i 0.0000
## j 0.0000
```

```
## the estimated position for our robot is:
```

```
## [1] "c"
```

**Contribution

- Q1-Zijie
- Q2-Zijie
- Q4-Phillip
- Q5-Phillip
- Q6-Pascal
- Q7-Andreas

Appendix

```r
knitr::opts_chunk$set(echo = F, out.height = "200px")
# if packages is not installed, eval = T
install.packages("HMM")
install.packages("entropy")
# used libraries in this lab
library(HMM)
library(entropy)
library(ggplot2)
library(dplyr)
library(latex2exp)
library(kableExtra)
# state Z: sector the robot might locate
sta <- strsplit("abcdefghij","")[[1]]
# observation x: sector the device reports
sym <- sta
# transition probability matrix
A <- matrix(0, nrow = 10, ncol = 10, dimnames = list(sta,sta))
# possible locations the device reports with different states
B <- matrix(0, nrow = 10, ncol = 10, dimnames = list(sta,sta))
for (i in 1:10) {
  f <- function(i){ifelse(i%%10==0,10,i%%10)}
  A[i,i] <- 0.5
  A[i,f(i+1)] <- 0.5
  B[i,f(i-2)] <- 0.2
  B[i,f(i-1)] <- 0.2
  B[i,f(i)] <- 0.2
  B[i,f(i+1)] <- 0.2
  B[i,f(i+2)] <- 0.2
}
hmm <- initHMM(States = sta, Symbols = sym,
               transProbs = A,
               emissionProbs = B)
cat("The transition probability:")
kable(A) %>%
  kable_styling(latex_options="basic")
cat("The emission probability:")
kable(B) %>%
  kable_styling(latex_options="basic")
set.seed(12345)
n <- 100
res <- simHMM(hmm, n)
states <- res$states
observation <- res$observation
res
simulation=res
# compute filtered probability distributions
forwar_method_log = forward(hmm = hmm, observation = simulation$observation)
# compute smoothed probability distributions
smoothed = posterior(hmm = hmm, observation = simulation$observation)
# Compute most probable path
viterbi_method = viterbi(hmm = hmm, observation = simulation$observation)
```

```r
##### compute filtered proability distributions
# Remove the log-transformation
forwar_method = exp(forwar_method_log)
# Normalizing
forwar_method_norm = prop.table(forwar_method, margin = 2)
# margin over the columns -> 2
# Checking which probability in each column is the highest
forwar_method_prob = apply(forwar_method_norm, MARGIN = 2, FUN = which.max)
# margin over the columns -> 2
# Accuracy for the filter
accuracy_filter = sum(sta[forwar_method_prob] == simulation$states)/
  length(simulation$states)
#####  compute smoothed proability distributions
# 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(sta[most_prob_smoothed] == simulation$states) /
  length(simulation$states)
#####  most probable path
accuracy_viterbi = sum(viterbi_method == simulation$states) /
  length(simulation$states)
cat("The accuracy of the following methods")
accruacy_table = data.frame("filtered" = accuracy_filter,
                            "smoothed" = accuracy_smoothed,
                            "viterbi" = accuracy_viterbi)
knitr::kable(accruacy_table)
plot(x = 1:n, y = viterbi_method,
     type = "l", col = "blue",  ylim = c(0,13),
     xlab = "simulation", ylab = "states", main = "Predicted path")
lines(x =1:n, y = forwar_method_prob, col =  "red")
lines(x =1:n, y = most_prob_smoothed, col = "green")
legend(3, 13, legend=c("Viterbi", "Filtered", "Smoothed"),
       col=c("blue", "red", "green"), lty=1, cex=0.7)
general_rate<-data.frame(filtering_accuracy=0.5825490,smoothing_accuracy=0.6864706,
                         viterbi_accuracy=0.5141176)
rownames(general_rate) <- "Mean"
kable(general_rate) %>%        # knitr::kable(general_rate)
  kable_styling(latex_options="basic")
#?entropy.empirical()
# use the result from question 1-2-3-4
# not results from 5
entropy_filtered = apply(X = forwar_method_norm, MARGIN = 2, FUN = entropy.empirical)
entropy_smoothed = apply(X = norm_smoothed, MARGIN = 2, FUN = entropy.empirical)
plot(entropy_filtered,main = "Entropy filtered and smoothed",
     type = "l", col = "red",
     ylab = "Entropy", ylim = c(0,2))
lines(entropy_smoothed, col = "blue")
legend(x = "topright", lty=1,
       legend = c("Smoothed","Filtered"),
       col = c("red","blue"))
# Generate probabilities of the hidden state
# for time step 101
```

```r
posterior = forwar_method_norm[,100] # The last information of the robot aka the prior
pred_101 = posterior %*% A
cat("Probabilities of the hidden states for the time step 101: ")
t(pred_101)
cat("the estimated position for our robot is: ")
sta[which.max(pred_101)]
```