Lab 2 - Advanced Machine Learning

Gustav Sternelöv

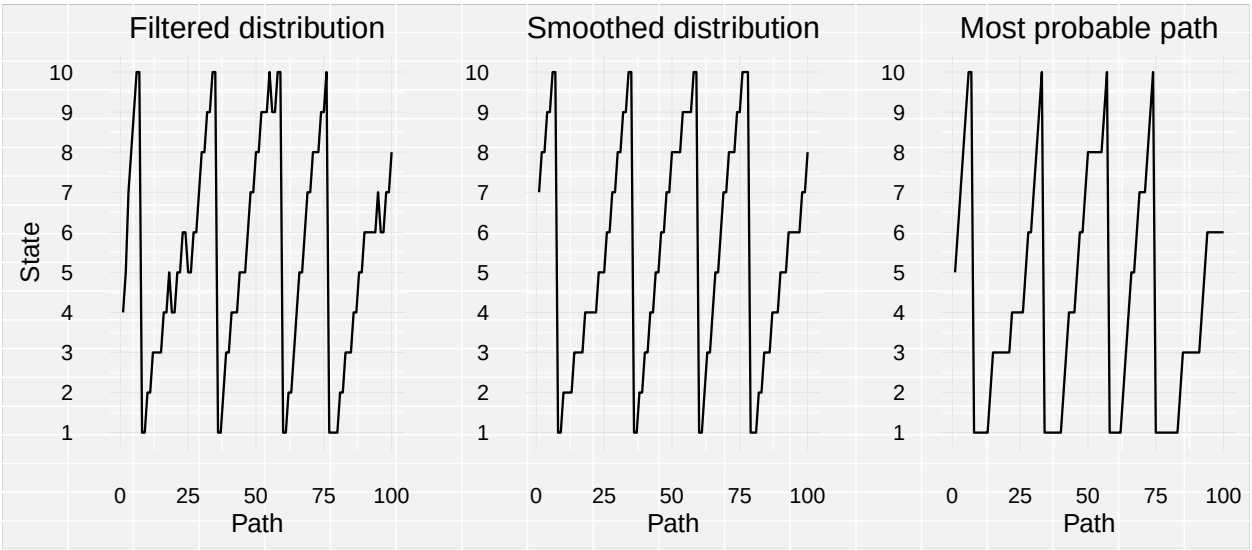
September 20, 2016

(1-2) Build a HMM for the scenario described above. Simulate the HMM for 100 time steps.

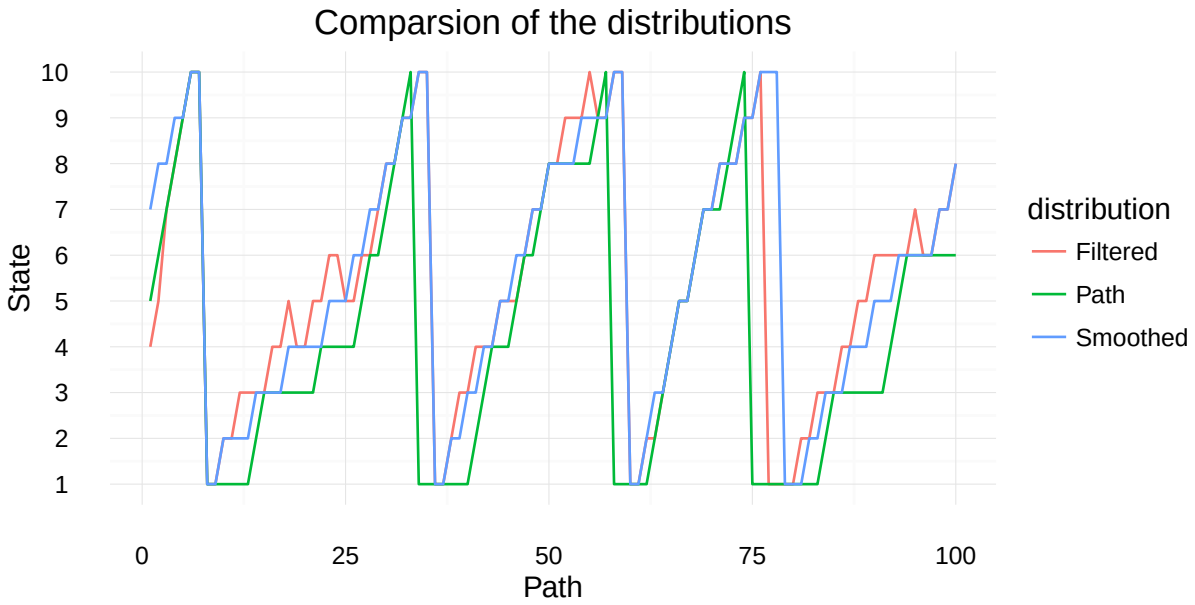
A HMM model with the described scenario is implemented with the *initHMM* function and in the model is the starting probability equal for all different states. For this model is then 100 time steps simulated.

(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.

The filtered and the smoothed probability distribution and the most probable path for the 100 time points are visualised with the plots below.



All the distributions seem to be rather similar as the general pattern is the same in the plots for each distribution. To compare the respective distributions may be easier when all the distributions are in the same plot.



It is notable that the line for the most probable path seem to get stuck at state 1 for relatively large periods. The filtered and the smoothed distribution appears to be rather similar for this set of 100 simulated time steps. Althoguh, there is one notable difference between the two distributions. At four time steps the filtered distribution moves backwards, something that not is possible according to the specified scenario for the model. This does not occur at any time point for the smoothed distribution.

(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.

The accuracy for the respective distributions is presented in the table below. For the actual sample of 100 time steps is the smoothed probability the most accuarate one with a margin of 9 percentages down to the filtered distribution. The least accurate of the methods for this set of simulated time steps was the most probable path which was right in 44 % of the time steps.

```
##   AccuracyFiltered AccuracySmoothed AccuracyPath
## 1             0.57             0.66         0.44
```

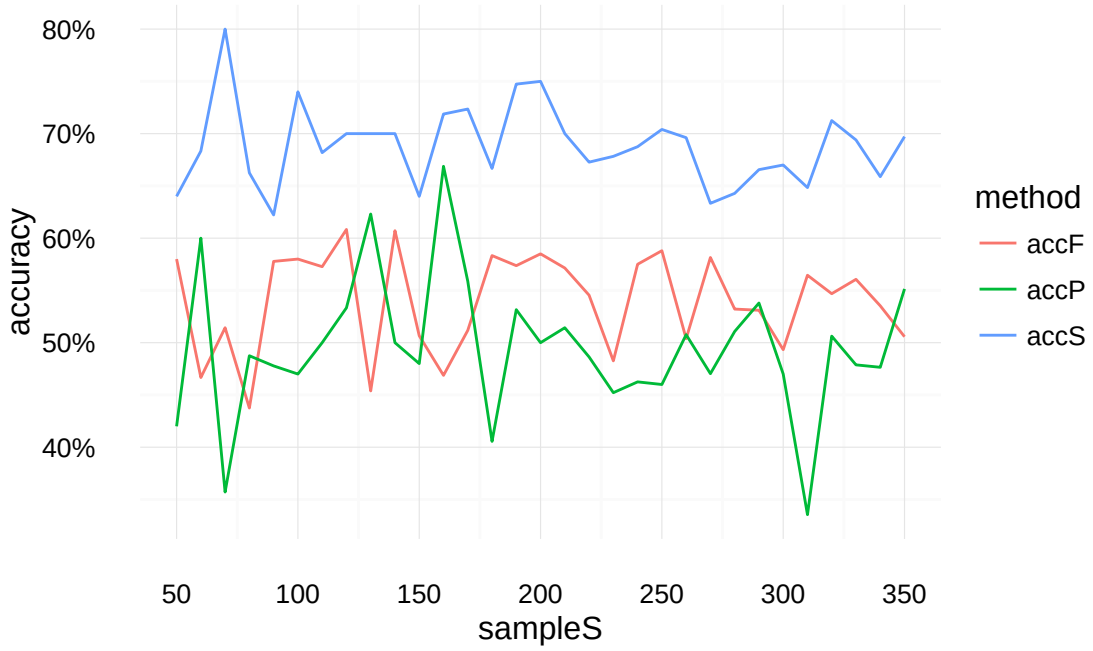
The probability for each state for the respective distributions is shown in the following table. The smoothed distribution has the most values that are 0.1 or close to 0.1. For the filtered distribution is the values also relatively close to 0.1 but for the most probable path there are som values that are quite far from 0.1. Especially for state 1 is the probability, 0.27, very high.

```
##    Filtered Smoothed Path
## 1      0.10     0.09 0.27
## 2      0.07     0.09 0.04
## 3      0.10     0.11 0.16
## 4      0.11     0.11 0.10
## 5      0.13     0.10 0.06
## 6      0.13     0.10 0.13
```

```
## 7       0.09      0.09 0.06
## 8       0.09      0.12 0.09
## 9       0.10      0.10 0.04
## 10      0.08      0.09 0.05
```

(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 path, too. Why ?

The accuracy is computed for samples of 50 to 350 by steps of 50 and the results are presented in the plot below.

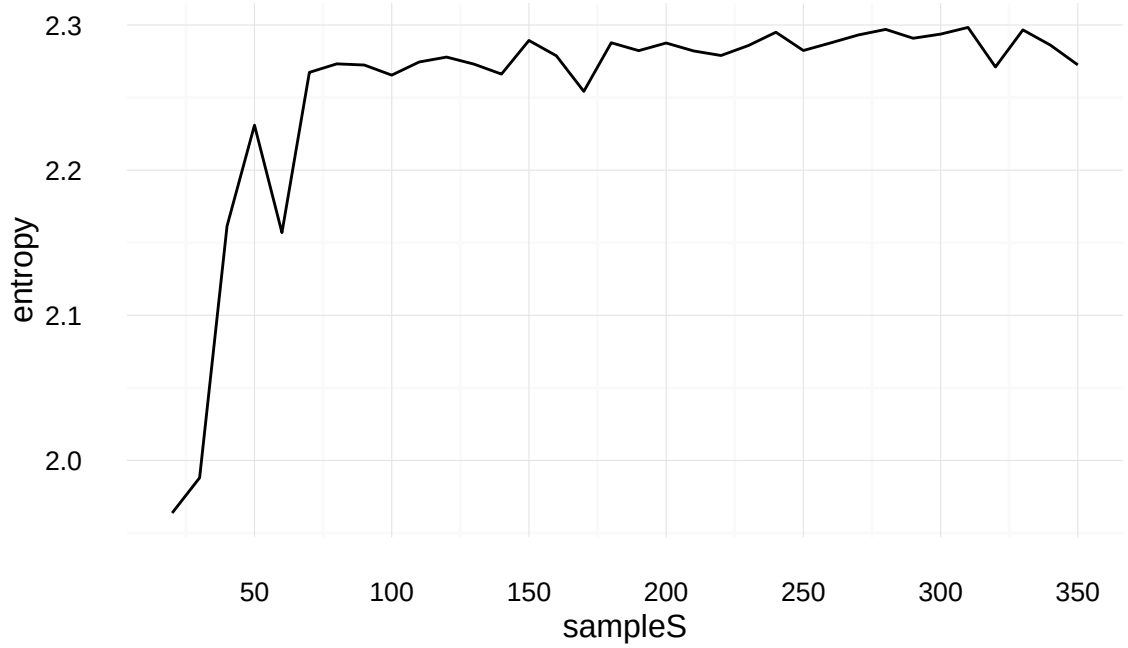


The blue line is the accuracy for the smoothed distribution and it can be seen that this distribution has the highest accuracy for all samples. The filtered distribution, red line, and most probable path, green line, has a relatively similar accuracy for most of the samples.
In general the smoothed distribution should be more accurate than the filtered since the former uses more data when it simulates the observation. More exactly, it makes use of both the forward and the backward step in the forward-backward algorithm whilst the filtered distribution only is a result of the forward step.

The smoothed distribution should also in the general case be more accurate than the most probable path. This is also a logical result since the aim with the viterbi algorithm which returns the most probable path is to return the path with the highest probability. A result of this aim is that the individually most probable state at each time step not always is chosen with the viterbi algorithm. The forward-backward algorithm on the other hand is constructed so that it at each time step choses the most probable state, but this may lead to paths with very low probability. Therefore, the smoothed distribution in general is more accurate than the most probable path.

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

The entropy for samples of size 20 to 350 by steps of 10 for the filtered distribution is plotted.



As the sample size increases the entropy increases as well. This result indicates that we not do know better where the roboit is as the number of observations increases. When the sample size grows from 25 to 50 a relatively large increase for the entropy is returned and after that it keeps growing for sample sizes up to 100 before it settles a bit. The explanation to this lies in the fact that for the filtered distribution we can at a specific time step be quite sure where the robot is. However, it does not necessarily helps us very much to be sure about this time point when we try to discover where it is at the next time point.

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

The probabilities of the hidden states for time step 101 are obtained by calculating the following probabilities.

$$p(Z^{101}|X^{0:100}) = \sum_{Z^{100}} p(Z^{101}, Z^{100}|X^{0:100})$$

$$p(Z^{101}|Z^{100}, X^{0:100})p(Z^{100}|X^{0:100}) = p(Z^{101}|Z^{100})p(Z^{100}|X^{0:100})$$

Z^{100} is simulated to be 7, so in the transition matrix is the probability for state 7 and 8 set to 0.5 and for the rest of the states is the probability 0. This tranisition matrix is multiplicated with the 100th row of the filtering matrix and normalized so that the sum of the probabilties is equal to 1. The result of these computations are presented in the table below and it can be noted that the probability for Z^{101} to be in state 7 is 0.4 and the probability to be in state 8 is 0.6.

```
##       [,1]
##  [1,]  0.0
##  [2,]  0.0
```

```
##  [3,]  0.0
##  [4,]  0.0
##  [5,]  0.0
##  [6,]  0.0
##  [7,]  0.4
##  [8,]  0.6
##  [9,]  0.0
## [10,]  0.0
```

R code

```
## ---- echo=FALSE, warning=FALSE, message=FALSE----------------------------
library(HMM)
library(ggplot2)
library(scales)
library(gridExtra)

## ---- echo=FALSE---------------------------------------------------------- 1
seq_eP <- rep(1:10, 10)
transP <- matrix(data = 0, nrow = 10, ncol = 10)
for (i in 1:10) {
    transP[i, c(seq_eP[i], seq_eP[i + 1])] <- 0.5
}
emissionP <- matrix(data = 0, nrow = 10, ncol = 10)
for (i in 1:10) {
    j <- i + 10
    emissionP[c(seq_eP[j - 2], seq_eP[j - 1], seq_eP[j], seq_eP[j + 1], seq_eP[j +
        2]), i] <- 0.2
}

HMM1 <- initHMM(States = c(1:10), Symbols = c(1:10), startProbs = rep(0.1, 10),
    transProbs = transP, emissionProbs = emissionP)
# 2
set.seed(311015)
Sim1 <- simHMM(HMM1, 100)

## ---- echo=FALSE, fig.width=8, fig.height=3.5----------------------------- 3
## Filtered prob dist
options(scipen = 99)
alpha <- exp(forward(HMM1, Sim1$observation))
filtering1 <- matrix(0, ncol = 100, nrow = 10)
for (k in 1:100) {
    filtering1[, k] <- alpha[, k]/colSums(alpha)[k]
}
filter <- data.frame(state = 0, obs = 1:100)
for (h in 1:100) {
    filter[h, 1] <- which.max(filtering1[, h])
}
filter_p <- ggplot(filter, aes(x = obs, y = state)) + geom_path() + labs(x = "Path",
    y = "State", title = "Filtered distribution") + scale_y_continuous(breaks = c(1:10)) +
    theme_minimal()
```

```r
# Smoothing
smoothing <- posterior(HMM1, Sim1[[2]])
smooth <- data.frame(state = 0, obs = 1:100)
for (h in 1:100) {
    smooth[h, 1] <- which.max(smoothing[, h])
}
smooth_p <- ggplot(smooth, aes(x = obs, y = state)) + geom_path() + labs(x = "Path",
    y = "", title = "Smoothed distribution") + scale_y_continuous(breaks = c(1:10)) +
    theme_minimal()
# Most probable path
path <- data.frame(state = as.numeric(viterbi(HMM1, Sim1[[2]])), obs = 1:100)
path_p <- ggplot(path, aes(x = obs, y = state)) + geom_path() + labs(x = "Path",
    y = "", title = "Most probable path") + scale_y_continuous(breaks = c(1:10)) +
    theme_minimal()
plots <- list(filter_p, smooth_p, path_p)
plots1 <- arrangeGrob(grobs = plots, nrow = 1)
plot(plots1)

## ---- echo=FALSE, fig.width=7, fig.height=3.5----------------------------
filter$distribution <- "Filtered"
smooth$distribution <- "Smoothed"
path$distribution <- "Path"
allDist <- rbind(filter, smooth, path)
ggplot(allDist, aes(x = obs, y = state, col = distribution)) + geom_path() +
    labs(x = "Path", y = "State", title = "Comparsion of the distributions") +
    scale_y_continuous(breaks = c(1:10)) + theme_minimal()

## ---- echo=FALSE---------------------------------------------------------- 4
data.frame(AccuracyFiltered = sum(Sim1[[1]] == filter$state)/100, AccuracySmoothed = sum(Sim1[[1]] ==
    smooth$state)/100, AccuracyPath = sum(Sim1[[1]] == path$state)/100)

## ---- echo=FALSE----------------------------------------------------------

data.frame(Filtered = prop.table(table(filter$state))[1:10], Smoothed = prop.table(table(smooth$state))
    Path = prop.table(table(path$state))[1:10])


## ---- echo=FALSE, fig.width=6, fig.height=3.5---------------------------- 5
## Repeat the code above but with different sample sizes
accuracy <- data.frame(sampleS = seq(50, 350, 10), accF = 0, accS = 0, accP = 0)
set.seed(814)
for (t in 1:nrow(accuracy)) {
    Sim2 <- simHMM(HMM1, accuracy[t, 1])
    # Filtering
    alpha <- exp(forward(HMM1, Sim2[[2]]))
    filtering <- matrix(0, ncol = accuracy[t, 1], nrow = 10)
    for (k in 1:accuracy[t, 1]) {
        filtering[, k] <- alpha[, k]/colSums(alpha)[k]
    }
    filter <- data.frame(state = 0, obs = 1:accuracy[t, 1])
    for (l in 1:accuracy[t, 1]) {
        filter[l, 1] <- which.max(filtering[, l])
    }
```

```
    # Smoothing
    smoothing <- posterior(HMM1, Sim2[[2]])
    smooth <- data.frame(state = 0, obs = 1:accuracy[t, 1])
    for (h in 1:accuracy[t, 1]) {
        smooth[h, 1] <- which.max(smoothing[, h])
    }
    # Path
    path <- data.frame(state = as.numeric(viterbi(HMM1, Sim2[[2]])), obs = 1:accuracy[t,
        1])
    # Accuracy
    accuracy[t, 2] <- sum(Sim2[[1]] == filter$state)/accuracy[t, 1]
    accuracy[t, 3] <- sum(Sim2[[1]] == smooth$state)/accuracy[t, 1]
    accuracy[t, 4] <- sum(Sim2[[1]] == path$state)/accuracy[t, 1]
}
accuracy2 <- tidyr::gather(accuracy, "method", "accuracy", 2:4)
ggplot(accuracy2, aes(x = sampleS, y = accuracy, col = method)) + geom_line() +
    theme_minimal() + scale_y_continuous(labels = percent) + scale_x_continuous(breaks = seq(50,
    350, 50))

## ---- echo=FALSE, fig.width=6, fig.height=3.5, message=FALSE,
## warning=FALSE---- 6
library(entropy)
entrF <- data.frame(sampleS = seq(20, 350, 10), entropy = 0)
set.seed(311015)
for (t in 1:nrow(entrF)) {
    Sim3 <- simHMM(HMM1, entrF[t, 1])
    # Filtering
    alpha <- exp(forward(HMM1, Sim3[[2]]))
    filtering <- matrix(0, ncol = entrF[t, 1], nrow = 10)
    for (k in 1:entrF[t, 1]) {
        filtering[, k] <- alpha[, k]/colSums(alpha)[k]
    }
    filter <- data.frame(state = 0, obs = 1:entrF[t, 1])
    for (l in 1:entrF[t, 1]) {
        filter[l, 1] <- which.max(filtering[, l])
    }
    entrF[t, 2] <- entropy.empirical(table(filter$state))
}
ggplot(entrF, aes(x = sampleS, y = entropy)) + geom_line() + theme_minimal() +
    scale_x_continuous(breaks = seq(0, 350, 50))

## ---- echo=FALSE----------------------------------------------------------
TP <- matrix(c(0, 0, 0, 0, 0, 0, 0.5, 0.5, 0, 0), ncol = 1)
Alph <- matrix(filtering1[, 100])

TP * Alph/sum(TP * Alph)

## ----code=readLines(knitr::purl('C:\\Users\\Gustav\\Documents\\AdvML\\Lab2\\Lab2.Rmd',documentation
## = 1)), eval = FALSE, tidy=TRUE----
```