Advanced Machine Learning

Lab 2

Rasmus Holm

2017-09-21

1)

```r
library(HMM)
library(ggplot2)
library(entropy)

## Hidden variables (true positions)
states <- 1:10

transition_probs <- matrix(c(0.5, 0.5, 0, 0, 0, 0, 0, 0, 0, 0,
                             0, 0.5, 0.5, 0, 0, 0, 0, 0, 0, 0,
                             0, 0, 0.5, 0.5, 0, 0, 0, 0, 0, 0,
                             0, 0, 0, 0.5, 0.5, 0, 0, 0, 0, 0,
                             0, 0, 0, 0, 0.5, 0.5, 0, 0, 0, 0,
                             0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0,
                             0, 0, 0, 0, 0, 0, 0.5, 0.5, 0, 0,
                             0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0,
                             0, 0, 0, 0, 0, 0, 0, 0, 0.5, 0.5,
                             0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0.5),
                           byrow=TRUE, nrow=length(states), ncol=length(states))

## Emission variables (observed positions)
symbols <- 1:10

emission_probs <- matrix(c(0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0.2, 0.2,
                           0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0, 0.2,
                           0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0,
                           0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0,
                           0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0,
                           0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0,
                           0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2, 0,
                           0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2, 0.2,
                           0.2, 0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0.2,
                           0.2, 0.2, 0, 0, 0, 0, 0, 0.2, 0.2, 0.2),
                         byrow=TRUE, nrow=length(states), ncol=length(states))

start_probs <- rep(1, length(states)) / length(states)

robot_hmm <- initHMM(states, symbols,
                     startProbs=start_probs,
                     transProbs=transition_probs,
                     emissionProbs=emission_probs)
print(robot_hmm)
#> $States
#>  [1]  1  2  3  4  5  6  7  8  9 10
#>
#> $Symbols
#>  [1]  1  2  3  4  5  6  7  8  9 10
#>
#> $startProbs
#>   1   2   3   4   5   6   7   8   9  10
#> 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
#>
#> $transProbs
#>     to
#> from   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
#>
#> $emissionProbs
#>       symbols
#> states   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
```

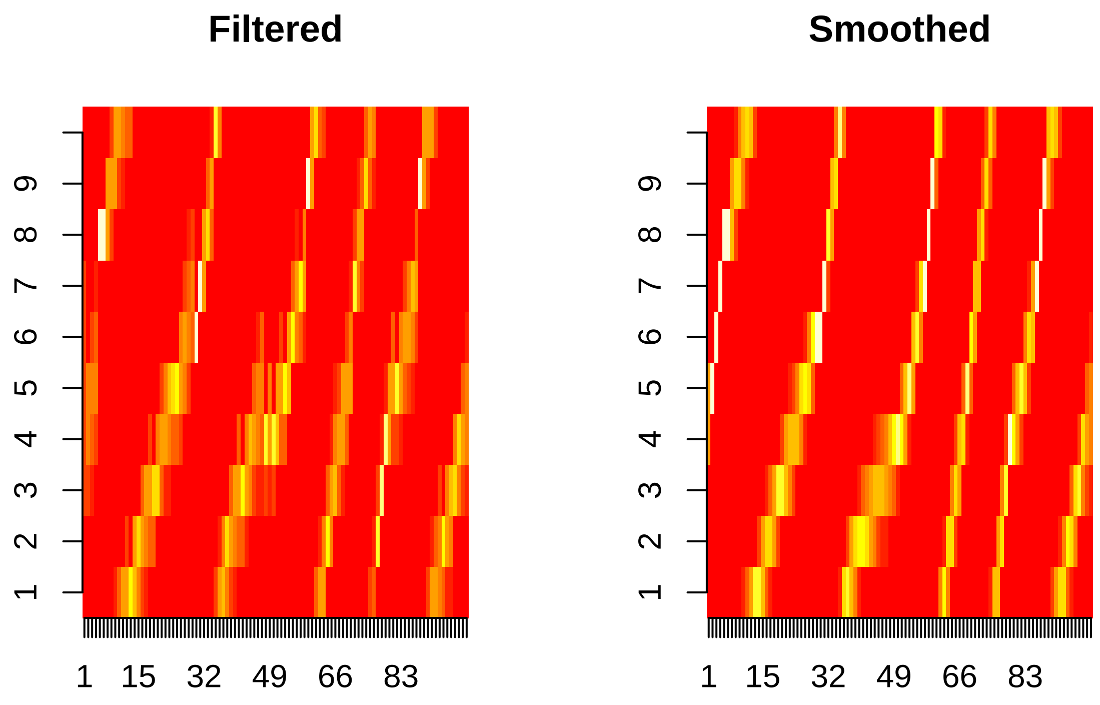
2)

```
set.seed(123)
samples_hmm <- simHMM(robot_hmm, 100)
print(samples_hmm)
#> $states
#>   [1]  4  5  6  7  8  8  9  9  9 10 10  1  1  1  2  2  3  3  3  4  5  5  5
#>  [24]  5  5  5  5  5  6  6  7  8  9  9 10  1  1  2  3  3  3  3  3  3  3  3
#>  [47]  3  3  3  4  4  4  5  6  7  7  7  8  9 10 10  1  2  3  4  4  5  5  5
#>  [70]  6  7  8  9 10  1  2  3  4  4  4  4  5  6  7  7  7  8  9  9 10  1  1
#>  [93]  2  2  3  3  4  4  4  5
#>
#> $observation
#>   [1]  5  3  5  6 10  6  8 10  1 10  9  2  9  3 10  2  1  3  1  5  4  3  3
#>  [24]  3  3  6  7  7  8  4  9  6 10 10  2 10  3 10  3  1  4  1  2  2  3  4
#>  [47]  4  2  4  2  6  5  3  7  8  7  5  7  1  8  2  2 10  4  3  5  4  3  5
#>  [70]  7  9  8  8  1 10 10  4  5  6  5  6  3  4  7  5  5  7  1  8  9  2  2
#>  [93]  3 10  3  1  5  2  4  5
```

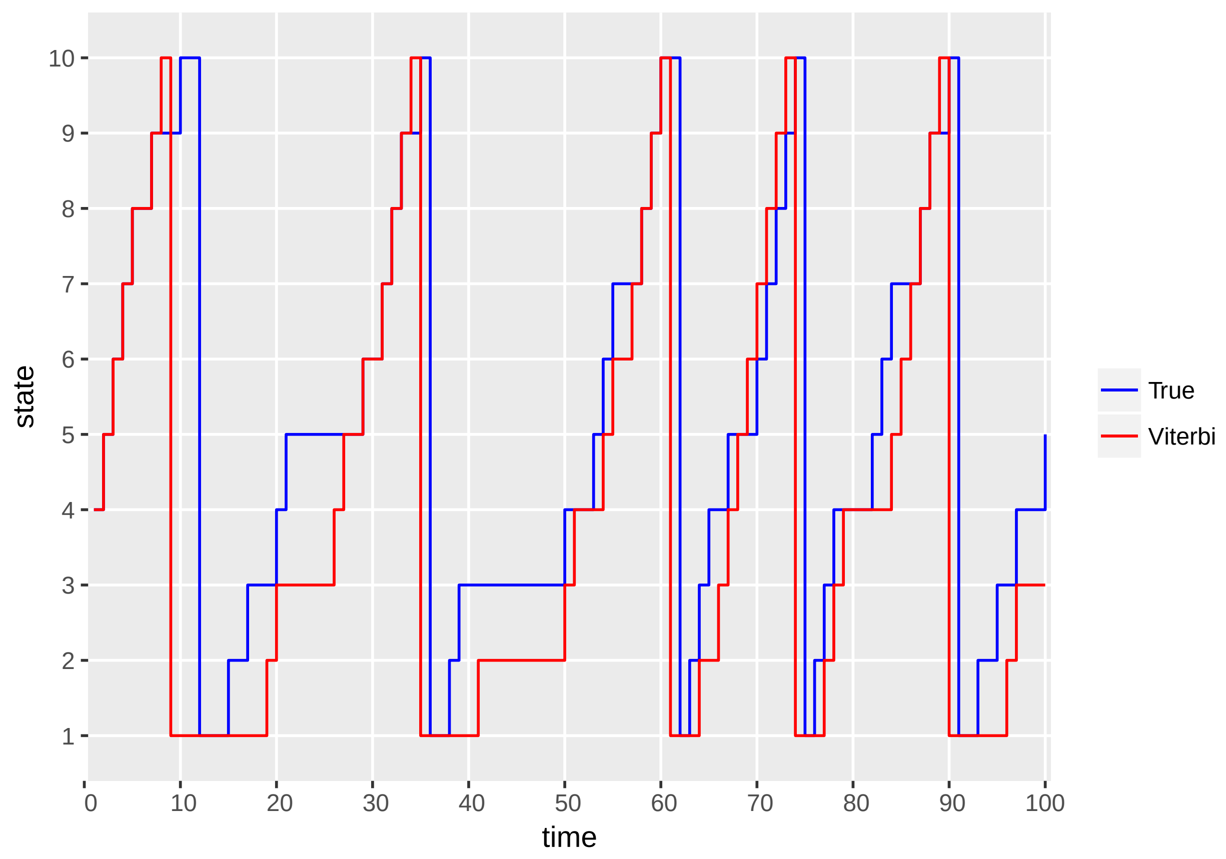
3)

```r
compute_filtered_probs <- function(hmm, observations) {
    log_probs <- forward(hmm, observations)
    probs <- prop.table(exp(log_probs), 2)
    probs
}

get_most_probable_states_by_filtered <- function(hmm, observations, states) {
    probs <- compute_filtered_probs(hmm, observations)
    most_probable_states <- as.numeric(apply(probs, 2, function(x) {
        states[which.max(x)]
    }))
    most_probable_states
}

compute_smoothed_probs <- function(hmm, observations) {
    probs <- posterior(hmm, observations)
    probs
}

get_most_probable_states_by_smoothed <- function(hmm, observations, states) {
    probs <- compute_smoothed_probs(hmm, observations)
    most_probable_states <- as.numeric(apply(probs, 2, function(x) {
        states[which.max(x)]
    }))
    most_probable_states
}

get_most_probable_path_by_viterbi <- function(hmm, observations) {
    most_probable_path <- viterbi(hmm, observations)
    most_probable_path
}

get_accuracy_filtered <- function(hmm, samples, states) {
    predicted_states <- get_most_probable_states_by_filtered(hmm, samples$observation, states)
    sum(predicted_states == samples$states) / length(predicted_states)
}

get_accuracy_smoothed <- function(hmm, samples, states) {
    predicted_states <- get_most_probable_states_by_smoothed(hmm, samples$observation, states)
    sum(predicted_states == samples$states) / length(predicted_states)
}

get_accuracy_viterbi <- function(hmm, samples, states) {
    predicted_states <- get_most_probable_path_by_viterbi(hmm, samples$observation)
    sum(predicted_states == samples$states) / length(predicted_states)
}

sample_states <- samples_hmm$states
sample_obs <- samples_hmm$observation

most_probable_states_filtered <- get_most_probable_states_by_filtered(robot_hmm, sample_obs, states)
```

```r
most_probable_states_smoothed <- get_most_probable_states_by_smoothed(robot_hmm, sample_obs, states)
most_probable_path <- get_most_probable_path_by_viterbi(robot_hmm, sample_obs)
```



The heatmaps show the state distributions in each timestep over 100 timesteps. A red color means low probability and the brigther the color, the higher the probability. We can clearly see that the smoothed probabilities have higher concentration than those found by filtered method. This is as we expect since smoothed uses the whole dataset for each estimation and is therefore more certain of its estimation.



The plot above shows the true states versus the most probable path found by the Viterbi algorithm. The general pattern is very similar for both but the predicted path is not quite right.

4)

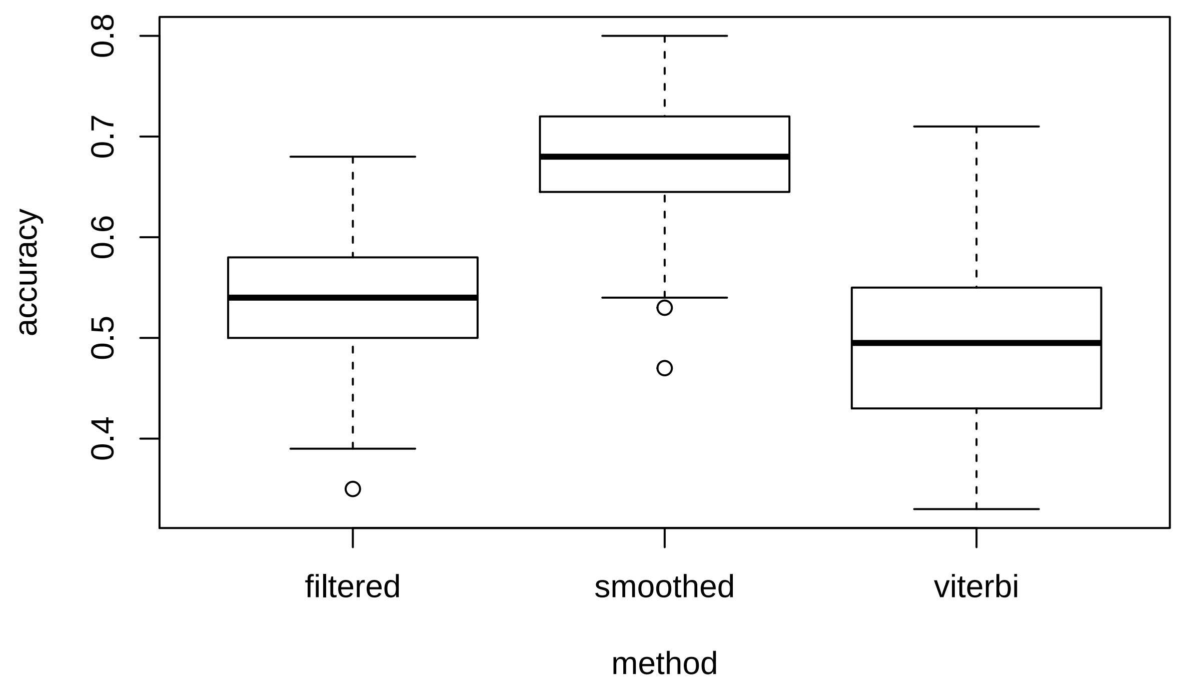
```
get_accuracy_filtered(robot_hmm, samples_hmm, states)
#> [1] 0.53
get_accuracy_smoothed(robot_hmm, samples_hmm, states)
#> [1] 0.64
get_accuracy_viterbi(robot_hmm, samples_hmm, states)
#> [1] 0.36
```

As we expect, the accuracy is higher for smoothed/filtered methods compared to Viterbi. And the smoothed method is superior.

5)

```r
nsamples <- 100
niters <- 100

filtered_acc <- rep(0, niters)
smoothed_acc <- rep(0, niters)
viterbi_acc <- rep(0, niters)

for (i in 1:niters) {
    samples <- simHMM(robot_hmm, nsamples)
    filtered_acc[i] <- get_accuracy_filtered(robot_hmm, samples, states)
    smoothed_acc[i] <- get_accuracy_smoothed(robot_hmm, samples, states)
    viterbi_acc[i] <- get_accuracy_viterbi(robot_hmm, samples, states)
}
```

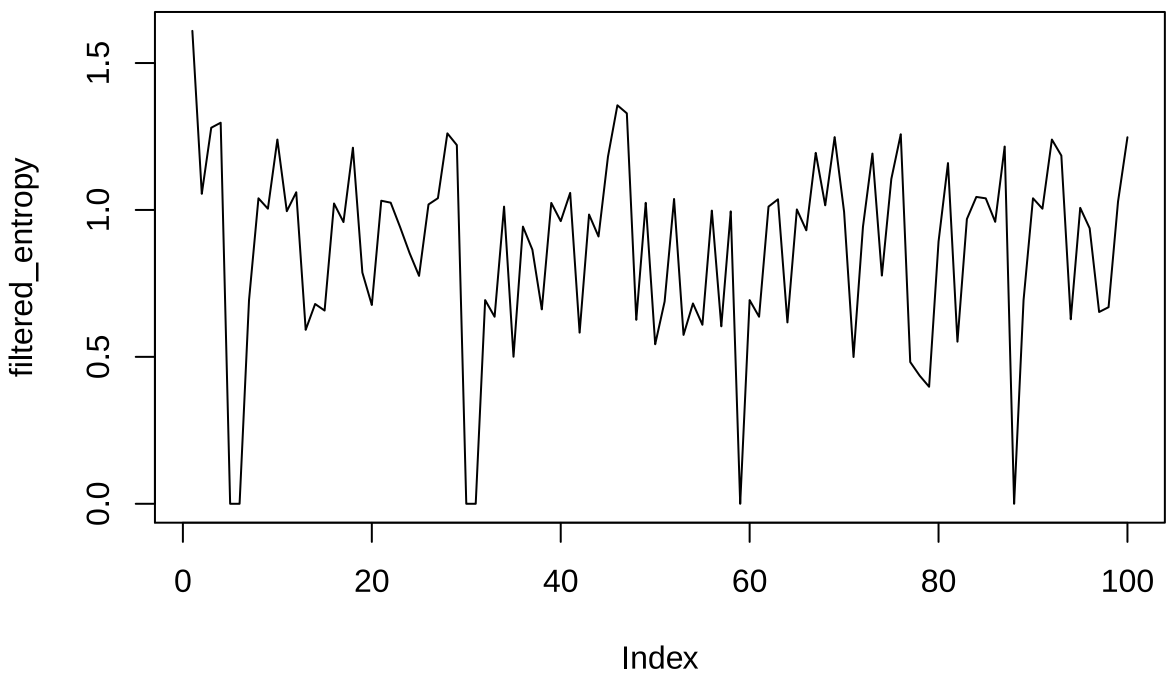
```r
plot_data <- data.frame(filtered=filtered_acc,
                        smoothed=smoothed_acc,
                        viterbi=viterbi_acc)
boxplot(plot_data, ylab="accuracy", xlab="method")
```



The accuracy of the smoothed probabilities is higher due to the fact that is uses all the data, past, present, and future, in its calculations. The Vitebi has the worst accuracy because it puts an extra contraint in its predictions, the path must be a valid path and thus mistakes early on propagate over the whole timespan. The smoothed/filtered probabilities are rather point estimates and does not have to follow a valid path.

6)

```r
filtered_probs <- compute_filtered_probs(robot_hmm, samples_hmm$observation)
filtered_entropy <- apply(filtered_probs, 2, entropy.empirical)
plot(filtered_entropy, type="l")
```



The plot above shows the empirical entropy of the distribution estimated by the filetered method for each step. We can clearly see that the entropy does not decrease over time, which indicates that our estimates do not increase in certainty. This is because the underlying state also changes for each timestep, thus the distribution is not statitionary.

7)

We can compute the probability of next state as

$$p(x_{t+1}) = p(x_{t+1}|x_t)p(x_t),$$

where $p(x_{t+1}|x_t)$ is specified by the transition matrix and $p(x_t)$ is the distribution estimated by the filtered or smoothed method.

```
probs <- compute_filtered_probs(robot_hmm, samples_hmm$observation)[, length(samples_hmm$observation)]
prediction <- probs %*% robot_hmm$transProbs
prediction
#>       to
#>        1 2          3         4     5         6          7 8 9 10
#>   [1,] 0 0 0.04883721 0.2226744 0.375 0.2773256 0.07616279 0 0  0
```