Advanced Machine Learning

Lab 2

Rasmus Holm

2017-09-18

1)

```
library(HMM)

## 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)
```

2)

```
set.seed(12345)
samples_hmm <- simHMM(robot_hmm, 100)
```

3)

```
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)
```

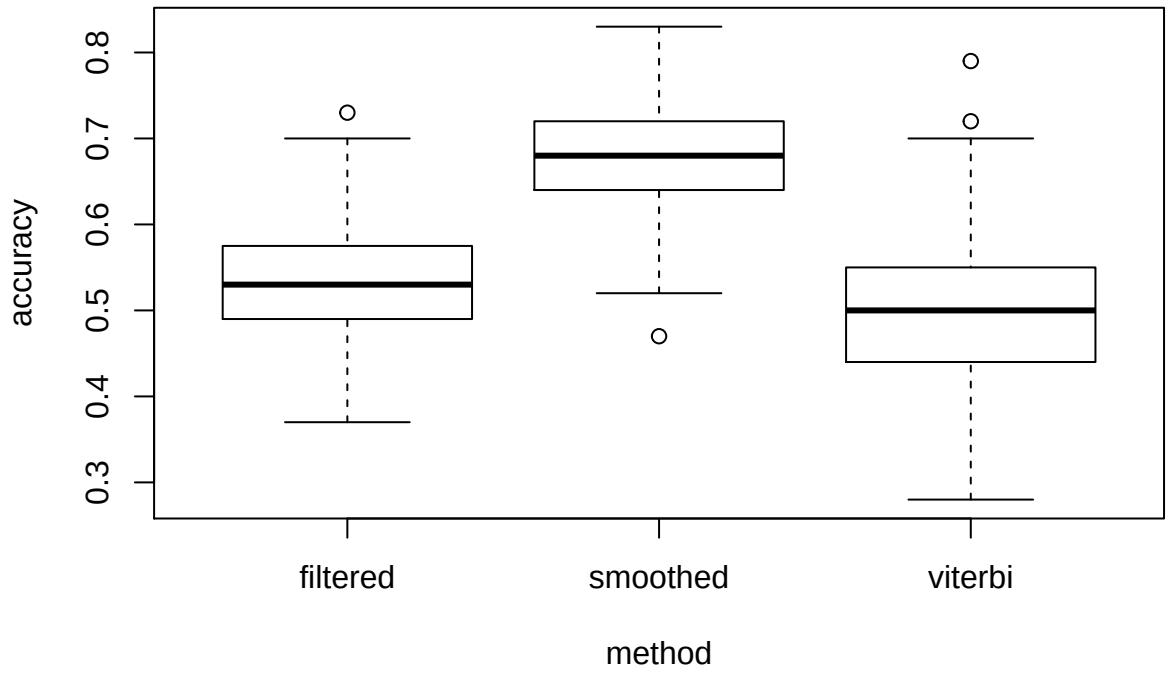
```
    sum(predicted_states == samples$states) / length(predicted_states)
}

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)
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)
```

4)

```
get_accuracy_filtered(robot_hmm, samples_hmm, states)
#> [1] 0.53
get_accuracy_smoothed(robot_hmm, samples_hmm, states)
#> [1] 0.74
get_accuracy_viterbi(robot_hmm, samples_hmm, states)
#> [1] 0.56
```

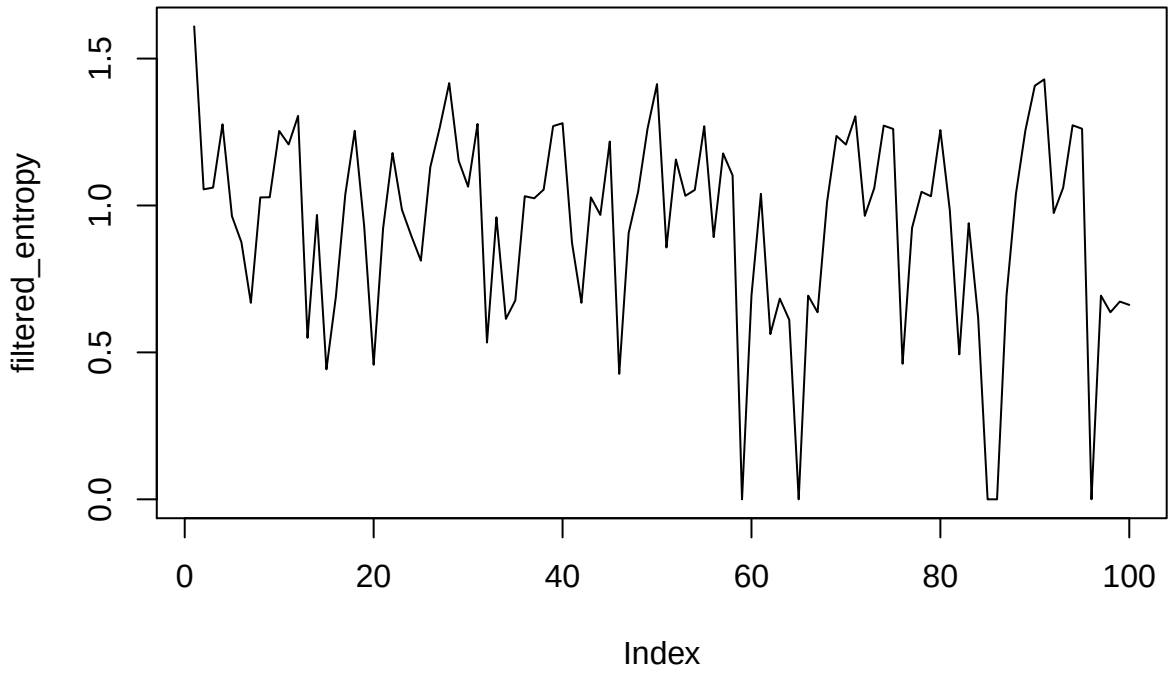
5)

```
nsamples <- 100
niters <- 1000

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)
}

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



6)

```
library(entropy)

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



7)

```
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 0.1875 0.5 0.3125 0 0 0 0 0  0
```