Report_lab2

Gustav Wahlquist

2020-09-22

Libraries used:

```
library("HMM")
library("entropy")
```

Background

The purpose of the lab is to put in practice some of the concepts covered in the lectures. To do so, you are asked to model the behavior of a robot that walks around a ring. The ring is divided into 10 sectors. At any given time point, the robot is in one of the sectors and decides with equal probability to stay in that sector or move to the next sector. You do not have direct observation of the robot. However, the robot is equipped with a tracking device that you can access. The device is not very accurate though: If the robot is in the sector i, then the device will report that the robot is in the sectors [i minus 2, i plus 2] with equal probability.

Question 1

Build a hidden Markov model (HMM) for the scenario described above.

```
set.seed(1)

# Number of readings
n = 100

# The states are numbered from 0-9 and a reading is anotated with a "'"
states = c("0","1","2","3","4","5","6","7","8","9")
observable_readings = c("0'","1'","2'","3'","4'","5'","6'","7'","8'","9'")

# Transmission model where the probability to stay in a given state is the same as
# moving to next state.
transmission_model = t(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
  ),10,10))

# A emisson model where the probability of reading a certain state is uniformly distrubuted
# over the two previous states, the true state and the two following states.
emission_model = t(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
  ),10,10))


hmm = initHMM(states,
              observable_readings,
              transProbs = transmission_model,
              emissionProbs = emission_model)
```

Question 2

Simulate the HMM for 100 time steps.

```
simed_hmm = simHMM(hmm, n)

print(simed_hmm)
```

```
## $states
##   [1] "3" "3" "4" "4" "5" "6" "7" "8" "8" "9" "0" "1" "1" "2" "3" "3" "4" "4"
##  [19] "5" "6" "7" "7" "8" "9" "0" "1" "2" "2" "3" "3" "3" "4" "5" "5" "6" "7"
##  [37] "8" "9" "9" "0" "0" "0" "0" "0" "0" "0" "1" "2" "3" "4" "5" "6" "6" "6"
##  [55] "6" "6" "6" "7" "8" "9" "9" "0" "1" "2" "3" "3" "3" "4" "5" "6" "6" "7"
##  [73] "7" "7" "7" "8" "8" "9" "9" "9" "0" "0" "1" "2" "3" "3" "4" "5" "5" "5"
##  [91] "5" "5" "6" "7" "8" "8" "9" "0" "0" "0"
##
## $observation
##   [1] "1'" "2'" "2'" "5'" "3'" "6'" "7'" "0'" "7'" "1'" "2'" "9'" "3'" "0'" "3'"
##  [16] "3'" "3'" "6'" "4'" "8'" "5'" "9'" "0'" "9'" "9'" "0'" "0'" "3'" "2'" "4'"
##  [31] "4'" "6'" "7'" "3'" "5'" "9'" "0'" "1'" "8'" "0'" "9'" "9'" "1'" "1'" "9'"
##  [46] "0'" "1'" "4'" "3'" "5'" "3'" "4'" "6'" "4'" "4'" "7'" "4'" "7'" "6'" "7'"
##  [61] "7'" "2'" "0'" "4'" "5'" "4'" "3'" "2'" "3'" "6'" "8'" "5'" "5'" "6'" "6'"
##  [76] "7'" "9'" "0'" "0'" "8'" "1'" "8'" "2'" "0'" "5'" "3'" "3'" "3'" "5'" "4'"
##  [91] "3'" "6'" "7'" "5'" "6'" "0'" "9'" "2'" "1'" "9'"
```

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.

```
filtering <- function(hmm, observations) {

  alpha = exp(forward(hmm, observations))
  beta = exp(backward(hmm, observations))

  # Creating empty matrix to be filled in following loop
  filtered = matrix(, nrow = 10, ncol = n)

  # Computing the filtered distribution
  for (i in 1:length(observations)) {
    filtered[,i] = alpha[,i]/sum(alpha[,i])
  }
  return(filtered)
}

smoothing <- function(hmm, observations) {

  alpha = exp(forward(hmm, observations))
  beta = exp(backward(hmm, observations))

  # Creating empty matrix to be filled in following loop
  smoothed = matrix(, nrow = 10, ncol = n)

  # Computing the smoothed distribution
  for (i in 1:length(observations)) {
    smoothed[,i] = alpha[,i]*beta[,i]/sum(alpha[,1:i]*beta[,1:i]) # Double check if this is correct?
  }


  return(smoothed)
}

observations = simed_hmm$observation

filtered_distribution = filtering(hmm, observations)
smoothed_distribution = smoothing(hmm, observations)

# Computing the most probable path with the viterbi algorithm from the HMM-package
most_prob_path = viterbi(hmm, observations)
```

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.

```
predict <- function(dist, states) {
```

```
  predictions = c()

  for (i in 1:dim(dist)[2]) {

  # Predicts the state that has the highest probability, if there are multiple states with
  # equal probability, the state with the lowest number/index is predicted i.e if ["1","2"] is
  # two states with equal probability in a certain point, "1" is predicted.
  predictions[i] = states[which(dist[,i] == max(dist[,i]))]

  }

  return(predictions)
}

calc_acc <- function(hmm, pred) {

  # Fetching ture hidden states and creating vectors to store predictions in
  true_states = simed_hmm$states


  acc = sum(pred == true_states)/n

  return(acc)
}

# Computing the accuracy for the filtering method
filtered_pred = predict(filtered_distribution, states)
filtered_acc = calc_acc(simed_hmm, filtered_pred)

# Computing the accuracy for the smoothing method
smoothing_pred = predict(smoothed_distribution, states)
smoothing_acc = calc_acc(simed_hmm, smoothing_pred)

# Computing the accuracy for the smoothing method
mpp_acc = calc_acc(simed_hmm, most_prob_path)

filtered_acc
```

```
## [1] 0.56
```

```
smoothing_acc
```

```
## [1] 0.72
```

```
mpp_acc
```

```
## [1] 0.51
```

Question 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 paths, too. Why ?

```r
# Repeating earlier calculations with different seed
set.seed(12345)
simed_hmm = simHMM(hmm, n)
observations = simed_hmm$observation

filtered_distribution = filtering(hmm, observations)
smoothed_distribution = smoothing(hmm, observations)
most_prob_path = viterbi(hmm, observations)

# Computing the accuracy for the filtering method
filtered_pred = predict(filtered_distribution, states)
filtered_acc_seed2 = calc_acc(simed_hmm, filtered_pred)

# Computing the accuracy for the smoothing method
smoothing_pred = predict(smoothed_distribution, states)
smoothing_acc_seed2 = calc_acc(simed_hmm, smoothing_pred)

# Computing the accuracy for the smoothing method
mpp_acc_seed2 = calc_acc(simed_hmm, most_prob_path)

filtered_acc_seed2
```

```
## [1] 0.53
```

```r
smoothing_acc_seed2
```

```
## [1] 0.74
```

```r
mpp_acc_seed2
```

```
## [1] 0.56
```

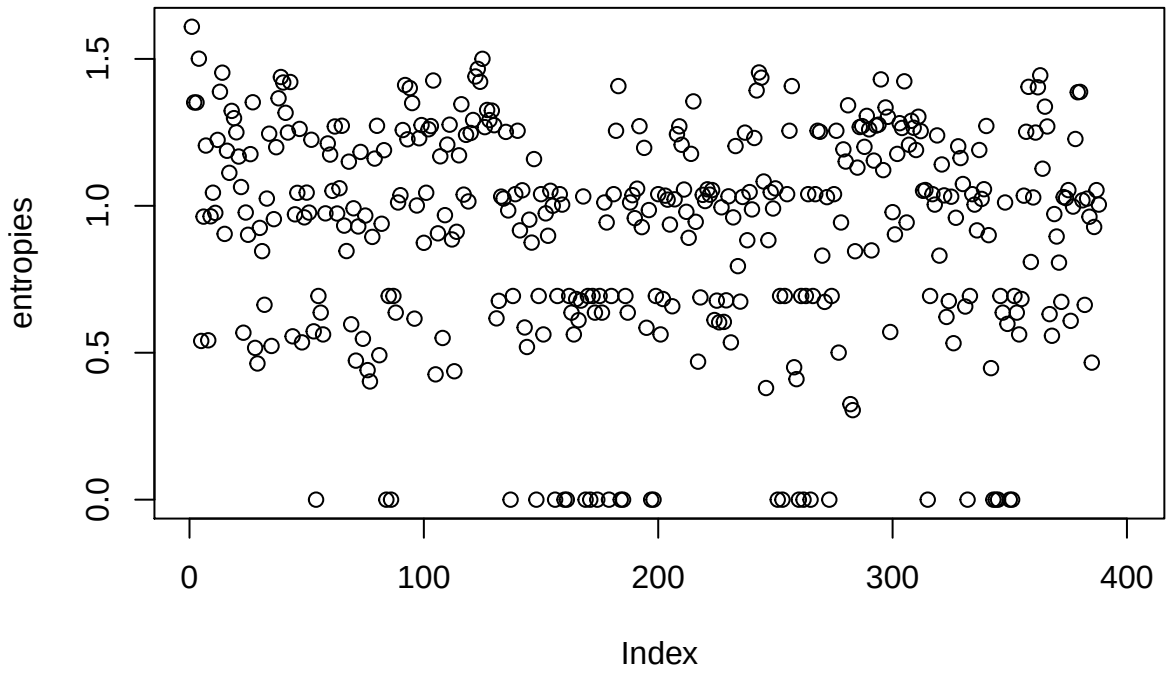
As seen in the results above, the smoothening method performs the best accuracy compared to the filtering method and the most probable path method. This is since the smoothed method takes future observations into account.

Question 6

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

```r
n = 400

simed_hmm = simHMM(hmm, n)
observations = simed_hmm$observation

filtered_distribution_1000_obs = filtering(hmm, observations)

entropies = c()

for (i in 1:n) {
```

```
  entropies[i] = entropy.empirical(filtered_distribution_1000_obs[,i])
}

plot(entropies)
```



It is hard to see any concrete trends in the entropy which intuitively makes sense. This is since the first reading does not really contribute with any information about what the 100th hidden state is.

Question 7

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

```
next_predicted_dist = filtered_distribution[,100]%*%transmission_model
prop.table(next_predicted_dist)
```

```
##      [,1]   [,2] [,3]   [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    0 0.1875  0.5 0.3125    0    0    0    0    0     0
```