Advanced Machine Learning Lab 2

Simge Cinar

2024-09-23

Question 1

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

Answer 1:

```r
transition_matrix <- matrix(0, nrow = 10, ncol = 10)

for (i in 1:9){
  transition_matrix[i, i:(i+1)] <- 0.5
}
transition_matrix[10, 10] <- 0.5
transition_matrix[10, 1] <- 0.5
```

```r
emission_matrix <- matrix(0, nrow = 10, ncol = 10)

for (i in 1:10) {
  sectors = (i-2):(i+2)
  row_idx = sectors %% 10
  row_idx[row_idx == 0] <- 10
  emission_matrix[row_idx, i] <- 0.2
}
```

```r
hmm = initHMM(States = c("h1", "h2", "h3", "h4", "h5", "h6", "h7", "h8", "h9", "h10"), # hidden state
              Symbols = c("o1", "o2", "o3", "o4", "o5", "o6", "o7", "o8", "o9", "o10"), # observed stat
              transProbs = transition_matrix,
              emissionProbs = emission_matrix)


print(hmm)
```

```
## $States
##  [1] "h1"  "h2"  "h3"  "h4"  "h5"  "h6"  "h7"  "h8"  "h9"  "h10"
##
## $Symbols
##  [1] "o1"  "o2"  "o3"  "o4"  "o5"  "o6"  "o7"  "o8"  "o9"  "o10"
##
## $startProbs
##  h1  h2  h3  h4  h5  h6  h7  h8  h9 h10
## 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##
## $transProbs
##      to
## from   h1  h2  h3  h4  h5  h6  h7  h8  h9 h10
##   h1  0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   h2  0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   h3  0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0
##   h4  0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0
##   h5  0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0
##   h6  0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0
##   h7  0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0
##   h8  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0
##   h9  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5
##   h10 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5
##
## $emissionProbs
##       symbols
## states  o1  o2  o3  o4  o5  o6  o7  o8  o9 o10
##    h1  0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2
##    h2  0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2
##    h3  0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0
##    h4  0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0
##    h5  0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0
##    h6  0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
##    h7  0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0
##    h8  0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2
##    h9  0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2
##    h10 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.

Answer 2:

```r
set.seed(12345)
simulation_hmm <- simHMM(hmm, length = 100)
simulation_hmm
```

```
## $states
##   [1] "h9"  "h9"  "h9"  "h9"  "h10" "h1"  "h2"  "h2"  "h2"  "h2"  "h3"  "h3"
##  [13] "h4"  "h4"  "h4"  "h4"  "h4"  "h4"  "h4"  "h5"  "h6"  "h6"  "h7"  "h8"
##  [25] "h9"  "h10" "h10" "h10" "h1"  "h2"  "h2"  "h3"  "h3"  "h4"  "h4"  "h4"
##  [37] "h5"  "h5"  "h5"  "h6"  "h7"  "h7"  "h8"  "h9"  "h10" "h1"  "h2"  "h3"
##  [49] "h3"  "h4"  "h5"  "h5"  "h6"  "h6"  "h7"  "h7"  "h8"  "h8"  "h8"  "h8"
##  [61] "h9"  "h10" "h10" "h10" "h10" "h1"  "h1"  "h2"  "h2"  "h2"  "h2"  "h2"
##  [73] "h3"  "h3"  "h3"  "h4"  "h5"  "h5"  "h5"  "h6"  "h7"  "h8"  "h8"  "h8"
##  [85] "h8"  "h8"  "h9"  "h9"  "h9"  "h10" "h10" "h10" "h1"  "h1"  "h1"  "h1"
##  [97] "h1"  "h1"  "h2"  "h3"
##
## $observation
##   [1] "o7"  "o10" "o8"  "o10" "o2"  "o3"  "o10" "o3"  "o4"  "o4"  "o5"  "o4"
##  [13] "o2"  "o3"  "o2"  "o6"  "o6"  "o5"  "o4"  "o3"  "o5"  "o5"  "o8"  "o9"
##  [25] "o10" "o9"  "o9"  "o10" "o2"  "o10" "o2"  "o5"  "o3"  "o2"  "o6"  "o6"
##  [37] "o4"  "o7"  "o7"  "o6"  "o5"  "o9"  "o7"  "o10" "o10" "o3"  "o3"  "o1"
##  [49] "o3"  "o3"  "o6"  "o5"  "o4"  "o7"  "o7"  "o9"  "o9"  "o10" "o6"  "o9"
##  [61] "o10" "o2"  "o9"  "o9"  "o8"  "o1"  "o3"  "o2"  "o3"  "o4"  "o3"  "o2"
##  [73] "o5"  "o4"  "o4"  "o2"  "o4"  "o6"  "o4"  "o6"  "o8"  "o10" "o8"  "o7"
##  [85] "o6"  "o6"  "o7"  "o8"  "o9"  "o10" "o1"  "o9"  "o2"  "o2"  "o3"  "o9"
##  [97] "o2"  "o10" "o4"  "o1"
```

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.

Answer 3:

```r
log_forward_probs <- forward(hmm, simulation_hmm$observation)
log_backward_probs <- backward(hmm, simulation_hmm$observation)

forward_probs <- exp(log_forward_probs) # alpha(z_t)
backward_probs <- exp(log_backward_probs) # beta(z_t)
```

The formula for smoothing and filtering can be seen below, z represents hidden state and x represent observed states

Smoothing:

$$p(z^t|x^{0:T}) = \frac{\alpha(z^t)\beta(z^t)}{\sum_{z^t} \alpha(z^t)\beta(z^t)}$$

Filtering:

$$p(z^t|x^{0:t}) = \frac{\alpha(z^t)}{\sum_{z^t} \alpha(z^t)}$$

```r
# 1. Smoothing
smoothed_probs <- forward_probs * backward_probs
smoothed_probs <- smoothed_probs / colSums(smoothed_probs) # Normalize

# 2. Filtering
filtered_probs <- matrix(nrow = dim(forward_probs)[1], ncol = dim(forward_probs)[2])
for (t in 1:dim(filtered_probs)[2]){
  filtered_probs[,t] <- forward_probs[,t] / sum(forward_probs[,t])
}

# 3. Finding most probable path
most_probable_path <- viterbi(hmm, simulation_hmm$observation)
```

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.

Answer 4:

```r
# Get the predictions for three methods
pred_smoothed_probs <- as.vector(apply(smoothed_probs, 2, which.max))
pred_filtered_probs <- as.vector(apply(filtered_probs, 2, which.max))
pred_most_probable_path <- as.integer(gsub("h", "", most_probable_path))
```

```r
# Get the true hidden state
true_hidden_state <- as.integer(gsub("h", "", simulation_hmm$states))

# Calculate confusion matrix and accuracy
```

```r
conf_matrix1 <- table(pred_smoothed_probs, true_hidden_state)
acc1 <- sum(diag(conf_matrix1)) / sum(conf_matrix1)
cat("Smoothed distribution accuracy:", acc1, "\n")
```

```
## Smoothed distribution accuracy: 0.74
```

```r
conf_matrix2 <- table(pred_filtered_probs, true_hidden_state)
acc2 <- sum(diag(conf_matrix2)) / sum(conf_matrix2)
cat("Filtered distribution accuracy:", acc2, "\n")
```

```
## Filtered distribution accuracy: 0.53
```

```r
conf_matrix3 <- table(pred_most_probable_path, true_hidden_state)
acc3 <- sum(diag(conf_matrix3)) / sum(conf_matrix3)
cat("Most probable path accuracy:", acc3, "\n")
```

```
## Most probable path accuracy: 0.56
```

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 ?

Answer 5:

```r
hidden_state_function <- function(hmm, length = 100){
  simulation <- simHMM(hmm, length) # run the simulation
  forward_probs <- exp(forward(hmm, simulation$observation)) # alpha
  backward_probs <- exp(backward(hmm, simulation$observation)) # beta

  # 1. smoothing
  smoothed_probs <- forward_probs * backward_probs
  smoothed_probs <- smoothed_probs / colSums(smoothed_probs)
  pred_smoothed_probs <- as.vector(apply(smoothed_probs, 2, which.max))

  # 2. filtering
  filtered_probs <- matrix(nrow = dim(forward_probs)[1], ncol = dim(forward_probs)[2])
  for (t in 1:dim(filtered_probs)[2]){
    filtered_probs[,t] <- forward_probs[,t] / sum(forward_probs[,t])
  }
  pred_filtered_probs <- as.vector(apply(filtered_probs, 2, which.max))

  # 3. most probable path
  most_probable_path <- viterbi(hmm, simulation$observation)
  pred_most_probable_path <- as.integer(gsub("h", "", most_probable_path))

  # Get the hidden states as integers
  true_hidden_state <- as.integer(gsub("h", "", simulation$states))

  # Get the accuracy
  conf_matrix1 <- table(pred_smoothed_probs, true_hidden_state)
  acc1 <- sum(diag(conf_matrix1)) / sum(conf_matrix1)

  conf_matrix2 <- table(pred_filtered_probs, true_hidden_state)
  acc2 <- sum(diag(conf_matrix2)) / sum(conf_matrix2)

  conf_matrix3 <- table(pred_most_probable_path, true_hidden_state)
  acc3 <- sum(diag(conf_matrix3)) / sum(conf_matrix3)

  return (c(acc1, acc2,acc3))
}
```

```
cat("Accuracy results for different simulations:\n")
```

```
## Accuracy results for different simulations:
```

```
for (i in 1:10){
  acc_results <- hidden_state_function(hmm, 100)
  cat("Smoothed:", acc_results[1], ";",
      "Filtered:", acc_results[2], ";",
      "Most probable path:", acc_results[3], "\n")
}
```

```
## Smoothed: 0.68 ; Filtered: 0.46 ; Most probable path: 0.61
## Smoothed: 0.77 ; Filtered: 0.49 ; Most probable path: 0.65
## Smoothed: 0.56 ; Filtered: 0.49 ; Most probable path: 0.56
## Smoothed: 0.81 ; Filtered: 0.6 ; Most probable path: 0.65
## Smoothed: 0.64 ; Filtered: 0.54 ; Most probable path: 0.39
## Smoothed: 0.69 ; Filtered: 0.54 ; Most probable path: 0.53
## Smoothed: 0.67 ; Filtered: 0.52 ; Most probable path: 0.55
## Smoothed: 0.68 ; Filtered: 0.52 ; Most probable path: 0.5
## Smoothed: 0.68 ; Filtered: 0.45 ; Most probable path: 0.45
## Smoothed: 0.77 ; Filtered: 0.52 ; Most probable path: 0.47
```

Filtering uses only past data whereas smoothing uses both path and future data to make prediction. Note that in the formula in question 3, the condition of z^t is different.

Question 6

Is it always true that the later in time (i.e., the more observations you have received) the better you know where the robot is ?

Answer 6:

```
for (i in 1:20) {
  i_new <- i * 5
  entropy <- entropy.empirical(filtered_probs[ ,i_new])
  cat("Observation length:", i_new, "; entropy:", entropy, "\n")
}
```

```
## Observation length: 5 ; entropy: 0.9632493
## Observation length: 10 ; entropy: 1.253531
## Observation length: 15 ; entropy: 0.4422675
## Observation length: 20 ; entropy: 0.4579084
## Observation length: 25 ; entropy: 0.8123793
## Observation length: 30 ; entropy: 1.06391
## Observation length: 35 ; entropy: 0.6770096
## Observation length: 40 ; entropy: 1.280137
## Observation length: 45 ; entropy: 1.217609
## Observation length: 50 ; entropy: 1.413187
## Observation length: 55 ; entropy: 1.26979
## Observation length: 60 ; entropy: 0.6931472
## Observation length: 65 ; entropy: 0
## Observation length: 70 ; entropy: 1.207243
## Observation length: 75 ; entropy: 1.260608
## Observation length: 80 ; entropy: 1.256652
## Observation length: 85 ; entropy: 0
## Observation length: 90 ; entropy: 1.407532
## Observation length: 95 ; entropy: 1.261251
## Observation length: 100 ; entropy: 0.6615632
```

No, entropy is a measure of uncertainty and it fluctuates between 0 and 1.5 in this example. Even though we get more observation, we don't gain more information.

Question 7

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

Answer 7:

```
forward_probs100 <- as.matrix(prop.table(forward_probs, margin = 2)[,100])

pred101 <- t(hmm$transProbs) %*% forward_probs100
cat("Hidden state for timestamp 101:\n")
```

```
## Hidden state for timestamp 101:
```

```
print(pred101)
```

```
##
## to      [,1]
##   h1  0.0000
##   h2  0.1875
##   h3  0.5000
##   h4  0.3125
##   h5  0.0000
##   h6  0.0000
##   h7  0.0000
##   h8  0.0000
##   h9  0.0000
##   h10 0.0000
```