TDDE15 Lab 2

Cajsa Schöld

2024-09-22

```
#install.packages('HMM')
library(HMM)
set.seed(12345)


states = c(1:10)
symbols = c(1:10)

trans_probs = matrix(0, length(states), length(states))
for (i in 1:10) {
  trans_probs[i,i] = 0.5
  trans_probs[i, (i %% 10)+1] = 0.5
}
trans_probs
```

```
##       [,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
```

```
emission_probs = matrix(0, length(states), length(symbols))
for (i in 1:10) {
  emission_probs[i,i] = 0.2
  emission_probs[i, (i %% 10) + 1] = 0.2
  emission_probs[i, ((i - 2) %% 10) + 1] = 0.2
  emission_probs[i, (i + 1) %% 10 + 1] = 0.2
  emission_probs[i, ((i - 3) %% 10) + 1] = 0.2
}
emission_probs
```

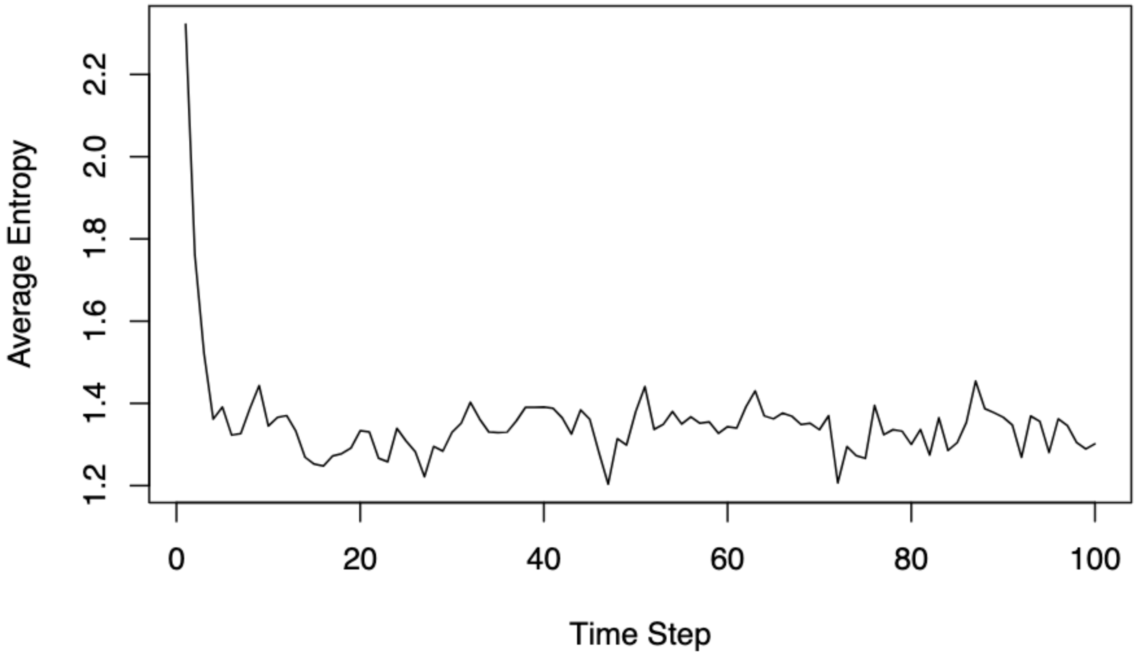
```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]  0.2  0.2  0.2  0.0  0.0  0.0  0.0  0.0  0.2   0.2
##  [2,]  0.2  0.2  0.2  0.2  0.0  0.0  0.0  0.0  0.0   0.2
##  [3,]  0.2  0.2  0.2  0.2  0.2  0.0  0.0  0.0  0.0   0.0
##  [4,]  0.0  0.2  0.2  0.2  0.2  0.2  0.0  0.0  0.0   0.0
##  [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
```

```r
hmm_model = initHMM(States = states, Symbols = symbols,
        transProbs=trans_probs, emissionProbs=emission_probs)


# 2 #########################
hmm_simulated = simHMM(hmm_model, length = 100)
hmm_states = hmm_simulated$states
hmm_obs = hmm_simulated$observation

# 3 #########################

filt_smooth_func <- function(model, obs) {
  alpha = exp(forward(model, obs))
  filtered = matrix(NA, nrow = nrow(alpha), ncol = ncol(alpha))
  for (t in 1:ncol(alpha)) {
    col_sum = sum(alpha[, t])
    filtered[, t] = alpha[, t] / col_sum
  }

  beta = exp(backward(model, obs))

  smooth = alpha * beta

  norm_smooth = matrix(0, nrow = nrow(smooth), ncol = ncol(smooth))

  for (t in 1:ncol(smooth)) {
    col_sum = sum(smooth[, t])
    norm_smooth[, t] = smooth[, t] / col_sum
  }
  return(list(filtered = filtered, norm_smooth = norm_smooth))
}


filtered_smooth = filt_smooth_func(hmm_model, hmm_obs)
filtered = filtered_smooth$filtered
smooth = filtered_smooth$norm_smooth

#most probable path
viterbi_path = viterbi(hmm_model, hmm_obs)

# 4 ########################

acc_func <- function(true, pred) {
  n = length(true)
  acc = sum(true==pred)/n
  return(acc)
}

acc_viterbi = acc_func(hmm_states, viterbi_path)

filtered_max = apply(filtered, 2, which.max)
smooth_max = apply(smooth, 2, which.max)

acc_filtered = acc_func(hmm_states, filtered_max)
acc_smooth = acc_func(hmm_states, smooth_max)

# 5 ###################

acc_matrix = matrix(NA, nrow = 100, ncol = 3)

for (i in 1:100) {
  hmm_simulated = simHMM(hmm_model, length = 100)
  hmm_states = hmm_simulated$states
  hmm_obs = hmm_simulated$observation
  filtered_smooth = filt_smooth_func(hmm_model, hmm_obs)
  filtered = filtered_smooth$filtered
  smooth = filtered_smooth$norm_smooth

  filtered_max = apply(filtered, 2, which.max)
  smooth_max = apply(smooth, 2, which.max)
  acc_matrix[i, 1] = acc_func(hmm_states, filtered_max)
  acc_matrix[i, 2] = acc_func(hmm_states, smooth_max)

  viterbi_path = viterbi(hmm_model, hmm_obs)
  acc_matrix[i, 3] = acc_func(hmm_states, viterbi_path)
}

acc_matrix
```

```
##        [,1] [,2] [,3]
##   [1,] 0.46 0.68 0.61
##   [2,] 0.49 0.77 0.65
##   [3,] 0.49 0.58 0.56
##   [4,] 0.60 0.79 0.65
##   [5,] 0.54 0.64 0.39
##   [6,] 0.54 0.69 0.53
##   [7,] 0.52 0.67 0.55
##   [8,] 0.52 0.67 0.50
##   [9,] 0.45 0.68 0.45
##  [10,] 0.52 0.77 0.47
##  [11,] 0.39 0.66 0.57
##  [12,] 0.48 0.74 0.56
##  [13,] 0.40 0.57 0.39
##  [14,] 0.50 0.70 0.47
##  [15,] 0.47 0.66 0.55
##  [16,] 0.62 0.63 0.50
##  [17,] 0.61 0.65 0.41
##  [18,] 0.45 0.52 0.30
##  [19,] 0.51 0.63 0.57
##  [20,] 0.57 0.75 0.46
##  [21,] 0.51 0.70 0.48
##  [22,] 0.51 0.62 0.60
##  [23,] 0.58 0.74 0.49
##  [24,] 0.56 0.72 0.42
##  [25,] 0.55 0.74 0.45
##  [26,] 0.54 0.79 0.65
##  [27,] 0.57 0.69 0.40
##  [28,] 0.44 0.61 0.62
##  [29,] 0.55 0.72 0.48
##  [30,] 0.53 0.69 0.46
##  [31,] 0.47 0.62 0.45
##  [32,] 0.61 0.71 0.60
##  [33,] 0.53 0.73 0.62
##  [34,] 0.65 0.64 0.49
##  [35,] 0.47 0.56 0.54
##  [36,] 0.47 0.67 0.53
##  [37,] 0.55 0.71 0.53
##  [38,] 0.45 0.66 0.49
##  [39,] 0.44 0.68 0.63
##  [40,] 0.61 0.68 0.60
##  [41,] 0.62 0.78 0.50
##  [42,] 0.46 0.71 0.59
##  [43,] 0.52 0.64 0.40
##  [44,] 0.60 0.69 0.41
##  [45,] 0.53 0.68 0.55
##  [46,] 0.53 0.72 0.52
##  [47,] 0.61 0.76 0.45
##  [48,] 0.65 0.65 0.49
##  [49,] 0.58 0.69 0.53
##  [50,] 0.51 0.65 0.55
##  [51,] 0.40 0.62 0.55
##  [52,] 0.55 0.68 0.54
##  [53,] 0.63 0.76 0.52
##  [54,] 0.48 0.69 0.44
##  [55,] 0.63 0.72 0.45
##  [56,] 0.39 0.66 0.51
##  [57,] 0.51 0.66 0.60
##  [58,] 0.58 0.71 0.33
##  [59,] 0.53 0.67 0.42
##  [60,] 0.50 0.71 0.61
##  [61,] 0.54 0.63 0.50
##  [62,] 0.61 0.72 0.39
##  [63,] 0.55 0.61 0.52
##  [64,] 0.50 0.63 0.41
##  [65,] 0.53 0.72 0.50
##  [66,] 0.52 0.67 0.45
##  [67,] 0.56 0.65 0.40
##  [68,] 0.48 0.71 0.46
##  [69,] 0.47 0.73 0.58
##  [70,] 0.61 0.72 0.46
##  [71,] 0.55 0.65 0.45
##  [72,] 0.66 0.65 0.39
##  [73,] 0.66 0.69 0.43
##  [74,] 0.46 0.62 0.36
##  [75,] 0.49 0.63 0.59
##  [76,] 0.58 0.73 0.48
##  [77,] 0.52 0.75 0.43
##  [78,] 0.49 0.57 0.37
##  [79,] 0.48 0.62 0.43
##  [80,] 0.48 0.70 0.43
##  [81,] 0.46 0.68 0.46
##  [82,] 0.62 0.73 0.52
##  [83,] 0.55 0.62 0.56
##  [84,] 0.56 0.65 0.50
##  [85,] 0.54 0.65 0.49
##  [86,] 0.53 0.67 0.33
##  [87,] 0.52 0.72 0.56
##  [88,] 0.45 0.60 0.43
##  [89,] 0.66 0.68 0.46
##  [90,] 0.56 0.73 0.53
##  [91,] 0.54 0.65 0.60
##  [92,] 0.57 0.68 0.46
##  [93,] 0.56 0.64 0.47
##  [94,] 0.56 0.69 0.30
##  [95,] 0.41 0.64 0.44
##  [96,] 0.49 0.67 0.47
##  [97,] 0.49 0.65 0.44
##  [98,] 0.49 0.70 0.50
##  [99,] 0.57 0.72 0.50
## [100,] 0.58 0.79 0.54
```

The smoothed probabilities are more accurate than the filtered ones since they consider both the past and future observations, while the filtered ones only consider the past. Regarding the viterbi algorithm it makes an global optimization over the whole path which makes it possible for the algoritm to preform badly at specific timesteps.

```
# 6 ################################

#install.packages("entropy")
library(entropy)

compute_entropy <- function(filtered) {
  entropies = matrix(nrow = ncol(filtered), ncol = 1)
  for (i in 1:ncol(filtered)) {
    entropies[i] = entropy.empirical(filtered[, i], unit = "log2")
  }
  return(entropies)
}

entropy_matrix = matrix(nrow = 100, ncol = 100)

for (i in 1:100) {
  hmm_simulated = simHMM(hmm_model, length = 100)
  hmm_states = hmm_simulated$states
  hmm_obs = hmm_simulated$observation

  filtered_smooth = filt_smooth_func(hmm_model, hmm_obs)
  filtered = filtered_smooth$filtered

  entropy_matrix[i, ] = compute_entropy(filtered)
}

average_entropy = apply(entropy_matrix, 2, mean)
plot(1:100, average_entropy, type = "l", xlab = "Time Step", ylab = "Average Entropy")
```



It is not always true that the later in time, the better you know where the robot is. Usually the entropy decreases with time as we get more data to use to make predictions but then we can see from the plot that it reaches a time where the entropy stop decreasing and more more observations apperently does not increase the accuracy.

```
# 7 ##########################
filtered_t101 = hmm_model$transProbs %*% filtered[, 100]
filtered_t101
```

```
##
## from          [,1]
##    1   0.00000000
##    2   0.00000000
##    3   0.00000000
##    4   0.03061224
##    5   0.13775510
##    6   0.28061224
##    7  0.31632653
##    8  0.18877551
##    9  0.04591837
##    10 0.00000000
```