Lab2 Individual Report

varsi146

2023-09-18

Question 1

```r
# Setting up the hidden and observed states as per question
num_states <- 10
hidden_states <- paste0('S', 1:num_states)
obs_states <- paste0('s', 1:num_states)

# Defining Initial probabilities
init_prob <- runif(10, min = 0, max = 1)

# Defining the transition matrix dimension: hidden_states x hidden_states

transMat <- matrix(0, nrow = num_states, ncol = num_states)
rownames(transMat) <- hidden_states
colnames(transMat) <- hidden_states

for (i in 1:num_states) {
  transMat[i, i] <- 0.5  # Probability of staying in the current sector
  if (i < num_states) {
    transMat[i, i + 1] <- 0.5  # Probability of moving to the next sector
  }else{
    transMat[i, 1] <- 0.5
  }
}

# Defining emission matrix with dimension: hidden_states x obs_symbols
emissMat <- 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),nrow = 10,byrow = TRUE)
rownames(emissMat) <- obs_states
colnames(emissMat) <- obs_states


hmm_model <- initHMM(States=hidden_states,
                       Symbols=obs_states,
                       startProbs=init_prob,
                       transProbs=transMat, emissionProbs=emissMat)
print(hmm_model)
```

```
## $States
##  [1] "S1"  "S2"  "S3"  "S4"  "S5"  "S6"  "S7"  "S8"  "S9"  "S10"
##
## $Symbols
##  [1] "s1"  "s2"  "s3"  "s4"  "s5"  "s6"  "s7"  "s8"  "s9"  "s10"
##
## $startProbs
##         S1         S2         S3         S4         S5         S6         S7
## 0.96339913 0.93554385 0.02883247 0.60103580 0.43600589 0.67632480 0.82844759
##         S8         S9        S10
## 0.06911952 0.47609352 0.75574709
##
## $transProbs
##      to
## from   S1  S2  S3  S4  S5  S6  S7  S8  S9 S10
##   S1  0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   S2  0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0
##   S3  0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0
##   S4  0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0
##   S5  0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0
##   S6  0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.0
##   S7  0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0
##   S8  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0
##   S9  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5
##   S10 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5
##
## $emissionProbs
##       symbols
## states  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10
##    S1  0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2
##    S2  0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2
##    S3  0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0 0.0
##    S4  0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0 0.0
##    S5  0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0 0.0
##    S6  0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0 0.0
##    S7  0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2 0.0
##    S8  0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2 0.2
##    S9  0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.2
##    S10 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.2 0.2 0.2
```

Question 2

```
simulate_HMM <- function(model, time_steps, alt_method = FALSE){
  # Function simulates an HMM for the the number of time-steps.
  # Has an alternative method of calculating smoothed distributions.
```

```r
  # Simulate the given HMM
  sim_timeStep <- simHMM(model, length = time_steps)

  # True values of Hidden States
  sim_actual <- sim_timeStep$states

  # Faulty/Noisy Observations
  sim_obs <- sim_timeStep$observation

  #parameters required to compute filtered and smoothed distributions
  # exp() because the forward and backward returns values on log-scale
  Alpha <- exp(forward(hmm = model, observation = sim_obs))
  Beta <- exp(backward(hmm = model, observation = sim_obs))

  # Normalizing the filter probabilities by column i.e., the filter
  # distribution is computed
  filtered <- apply(Alpha,2,prop.table)

  # Selecting the most probable state based on probability
  filter_pred <- apply(filtered, 2, which.max)
  filter_pred <- sapply(filter_pred, function(x) paste0('s',x))

  if (alt_method == FALSE) {
    # Computing smoothed distribution and normalizing as per the formula
    smoothed <- Alpha*Beta
    smoothed <- apply(smoothed, 2, prop.table)

    # Selecting the most probable state based on probability
    smoothed_pred <- apply(smoothed, 2, which.max)
    smoothed_pred <- sapply(smoothed_pred, function(x) paste0('s',x))
  }else{
    # Alternative method to calculate smoothed distribution. Correct?
    smoothed <- posterior(hmm = hmm_model, observation = sim_obs)
    smoothed_pred <- apply(smoothed, 2, which.max)
    smoothed_pred <- sapply(smoothed_pred, function(x) paste0('s',x))
  }

  # Computing the most probable state via the viterbi algorithm
  viterbi_path <- viterbi(hmm = model, observation = sim_obs)

  return(list('actual' = sim_actual, 'obs' = sim_obs,'filter_pred'= filter_pred,
              'smoothed_pred' = smoothed_pred,
              'viterbi' = viterbi_path, 'filter_mat' = filtered))
}

# Simulating above HMM for 100 time-steps
sim100_results <- simulate_HMM(model = hmm_model, time_steps = 100, alt_method = TRUE)
```

Question 3

```r
print('The filtered distribution computed from the simulated HMM is:')
```

```
## [1] "The filtered distribution computed from the simulated HMM is:"
```

```r
sim100_results$filter_pred
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13 
##  "s7"  "s7"  "s8"  "s8"  "s9" "s10"  "s1"  "s1"  "s2"  "s2"  "s2"  "s2"  "s2" 
##    14    15    16    17    18    19    20    21    22    23    24    25    26 
##  "s3"  "s3"  "s4"  "s5"  "s5"  "s6"  "s7"  "s7"  "s8"  "s8"  "s8"  "s8"  "s9" 
##    27    28    29    30    31    32    33    34    35    36    37    38    39 
##  "s9"  "s8"  "s8"  "s9" "s10" "s10" "s10"  "s1"  "s2"  "s2"  "s2"  "s3"  "s3" 
##    40    41    42    43    44    45    46    47    48    49    50    51    52 
##  "s3"  "s3"  "s3"  "s4"  "s5"  "s5"  "s4"  "s4"  "s5"  "s5"  "s5"  "s5"  "s6" 
##    53    54    55    56    57    58    59    60    61    62    63    64    65 
##  "s7"  "s7"  "s7"  "s8"  "s8"  "s9"  "s9"  "s9" "s10" "s10"  "s1"  "s1"  "s1" 
##    66    67    68    69    70    71    72    73    74    75    76    77    78 
##  "s1"  "s2"  "s2"  "s3"  "s3"  "s4"  "s4"  "s5"  "s5"  "s5"  "s5"  "s5"  "s6" 
##    79    80    81    82    83    84    85    86    87    88    89    90    91 
##  "s6"  "s7"  "s7"  "s8"  "s8" "s10"  "s1" "s10"  "s1"  "s1"  "s1"  "s1"  "s1" 
##    92    93    94    95    96    97    98    99   100 
##  "s2"  "s2"  "s3"  "s4"  "s4"  "s4"  "s5"  "s7"  "s7"
```

```r
print('The smoothed distribution computed from the simulated HMM is:')
```

```
## [1] "The smoothed distribution computed from the simulated HMM is:"
```

```r
sim100_results$smoothed_pred
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13 
##  "s7"  "s8"  "s8"  "s9"  "s9" "s10"  "s1"  "s1"  "s1"  "s2"  "s2"  "s2"  "s3" 
##    14    15    16    17    18    19    20    21    22    23    24    25    26 
##  "s3"  "s4"  "s4"  "s5"  "s6"  "s6"  "s6"  "s6"  "s7"  "s7"  "s7"  "s7"  "s8" 
##    27    28    29    30    31    32    33    34    35    36    37    38    39 
##  "s8"  "s8"  "s9"  "s9" "s10" "s10" "s10"  "s1"  "s1"  "s2"  "s2"  "s2"  "s2" 
##    40    41    42    43    44    45    46    47    48    49    50    51    52 
##  "s3"  "s3"  "s3"  "s4"  "s4"  "s4"  "s4"  "s4"  "s4"  "s5"  "s5"  "s6"  "s6" 
##    53    54    55    56    57    58    59    60    61    62    63    64    65 
##  "s6"  "s7"  "s7"  "s7"  "s8"  "s8"  "s9"  "s9" "s10" "s10" "s10"  "s1"  "s1" 
##    66    67    68    69    70    71    72    73    74    75    76    77    78 
##  "s1"  "s2"  "s2"  "s2"  "s3"  "s3"  "s3"  "s4"  "s4"  "s5"  "s5"  "s5"  "s6" 
##    79    80    81    82    83    84    85    86    87    88    89    90    91 
##  "s6"  "s7"  "s8"  "s8"  "s9" "s10" "s10" "s10" "s10" "s10"  "s1"  "s1"  "s2" 
##    92    93    94    95    96    97    98    99   100 
##  "s2"  "s3"  "s3"  "s4"  "s4"  "s5"  "s6"  "s7"  "s7"
```

```r
print('The most probable path computed from the simulated HMM is:')
```

```
## [1] "The most probable path computed from the simulated HMM is:"
```

```r
sim100_results$viterbi
```

```
##   [1] "S7"  "S8"  "S9"  "S10" "S1"  "S1"  "S1"  "S1"  "S1"  "S1"  "S1"  "S1"
##  [13] "S2"  "S2"  "S3"  "S4"  "S4"  "S5"  "S6"  "S6"  "S6"  "S6"  "S6"  "S6"
##  [25] "S6"  "S6"  "S6"  "S7"  "S8"  "S9"  "S9"  "S10" "S1"  "S1"  "S1"  "S1"
##  [37] "S1"  "S2"  "S2"  "S2"  "S2"  "S3"  "S4"  "S4"  "S4"  "S4"  "S4"  "S4"
##  [49] "S4"  "S4"  "S5"  "S6"  "S6"  "S6"  "S6"  "S7"  "S7"  "S8"  "S9"  "S10"
##  [61] "S1"  "S1"  "S1"  "S1"  "S1"  "S1"  "S1"  "S1"  "S2"  "S2"  "S2"  "S2"
##  [73] "S2"  "S2"  "S3"  "S4"  "S5"  "S5"  "S6"  "S7"  "S7"  "S8"  "S9"  "S10"
##  [85] "S10" "S10" "S1"  "S1"  "S1"  "S1"  "S1"  "S1"  "S2"  "S3"  "S4"  "S4"
##  [97] "S5"  "S6"  "S7"  "S7"
```

Question 4

```
accuracy <- function(actual, pred){
  accuracy <- sum(diag(table(actual, pred)))/sum(table(actual, pred))
  return(accuracy)
}

filter_accuracy <- accuracy(actual = sim100_results$actual, pred = sim100_results$filter_pred)

smoothed_accuracy <- accuracy(actual = sim100_results$actual, pred = sim100_results$smoothed_pred)

viterbi_accuracy <- accuracy(actual = sim100_results$actual, pred = sim100_results$viterbi)

cat('The accuracy of filtered distribution is', filter_accuracy)
```

```
## The accuracy of filtered distribution is 0.56
```

```
cat('The accuracy of smoothed distribution is', smoothed_accuracy)
```

```
## The accuracy of smoothed distribution is 0.66
```

```
cat('The accuracy of most probable paths distribution is',viterbi_accuracy)
```

```
## The accuracy of most probable paths distribution is 0.41
```

Question 5

```
sim100_samplesRes <- list(
  filter_accs = NULL,
  smooth_accs = NULL,
  viterbi_accs = NULL
)
# Computing filtered, smoothed and viterbi accuracies of simulated samples.
for (i in 1:50) {
  simulate_hmm_vals <- simulate_HMM(model = hmm_model, time_steps = 100, alt_method = FALSE)
  sim100_samplesRes$filter_accs[i] <- accuracy(actual = simulate_hmm_vals$actual,
                                               pred = simulate_hmm_vals$filter_pred)
  sim100_samplesRes$smooth_accs[i] <- accuracy(actual = simulate_hmm_vals$actual,
                                               pred = simulate_hmm_vals$smoothed_pred)

  sim100_samplesRes$viterbi_accs[i] <- accuracy(actual = simulate_hmm_vals$actual,
                                                pred = simulate_hmm_vals$viterbi)
}


cat('The mean accuracy of filtered distribution is', mean(sim100_samplesRes$filter_accs), '\n')
```

```
## The mean accuracy of filtered distribution is 0.538
```

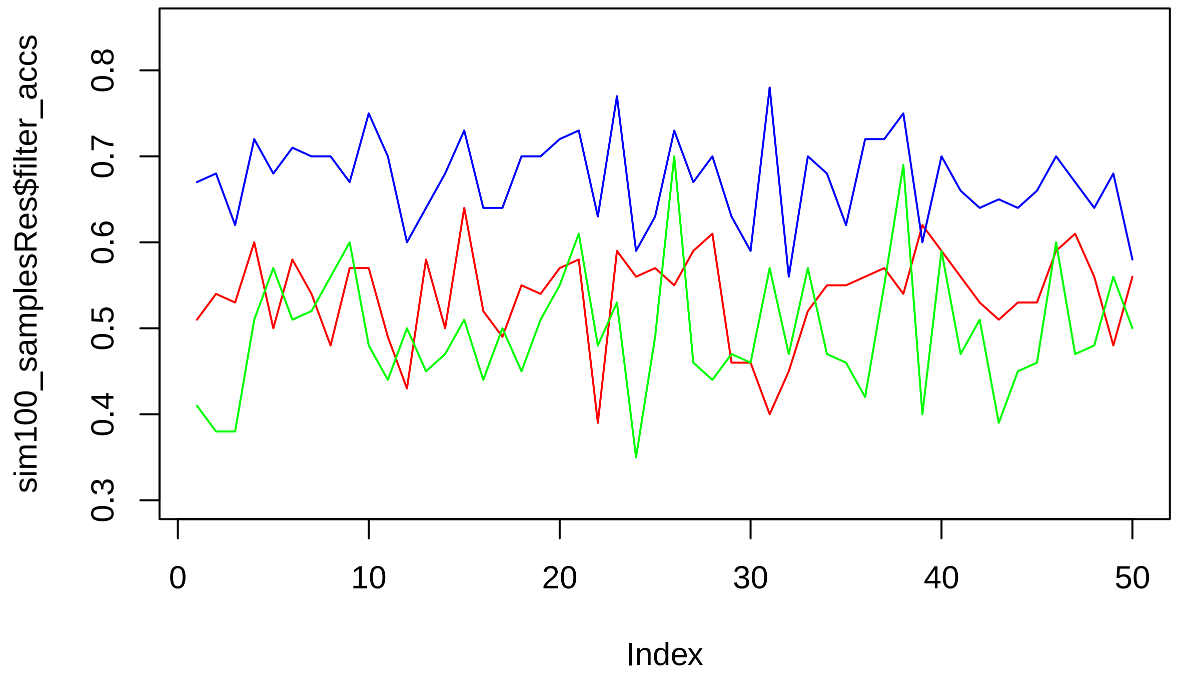
```
cat('The mean accuracy of smoothed distribution is', mean(sim100_samplesRes$smooth_accs), '\n')
```

```
## The mean accuracy of smoothed distribution is 0.6734
```

```
cat('The mean accuracy of most probable paths distribution is', mean(sim100_samplesRes$viterbi_accs), '
```

```
## The mean accuracy of most probable paths distribution is 0.4962
```

```
plot(sim100_samplesRes$filter_accs, type = 'l', col = 'red', ylim = c(0.3,0.85))
lines(sim100_samplesRes$smooth_accs, type = 'l', col = 'blue')
lines(sim100_samplesRes$viterbi_accs, type = 'l', col = 'green')
```



From the results above, we can see that in general, smoothed distributions are more accurate compared to filtered distributions, and this can be represented mathematically as well.

Smoothing is given by the following expression: $p(z^t|x^{0:T})$

While, filtering is given by the following expression: $p(z^t|x^{0:t})$

T in the smoothing expression means that the entire process has finished running and we calculate the probability of the hidden states given all the observations from start to the end of the process. While, the expression for filtering has t implying that the probability of hidden states are calculated only using the observations upto time step t and is hence more sensitive to short-term fluctuations or noisy data. In general, smoothed distributions are more accurate because they consider a broader context and hence are less sensitive to short-term fluctuations.

The Viterbi algorithm aims to find the single most probable sequence of states, by trying to maximize the joint probability of the observations and the states i.e., $p(x_1, ..., x_t, z_1, ..., z_t)$.

Viterbi algorithm is a type of greedy algorithm and hence is bound to get stuck in a local maxima.

Smoothed distributions are more accurate in general because they consider multiple possible state sequence rather than the single most probable one by considering the past and the future observations i.e., the forward and the backward pass.

Question 6

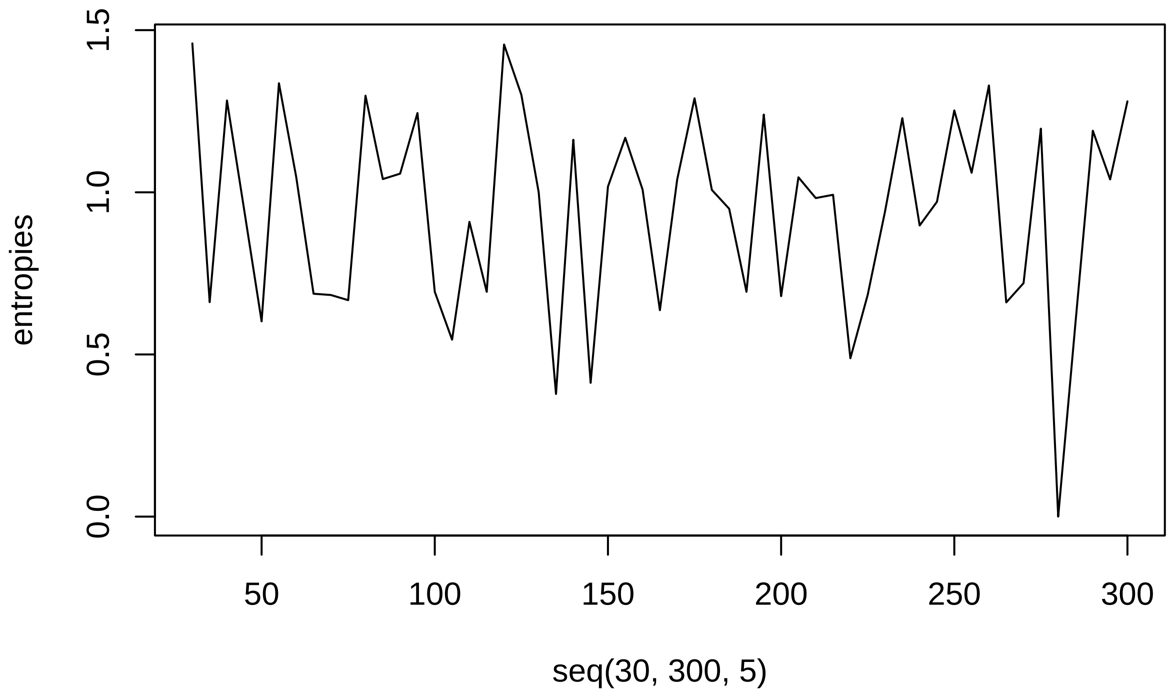
In general, one would expect that later in time, as the number of observations increase we know better about the robot's position.

To do so, we are checking the entropy of the filtered distribution. However, since we have faulty/noisy observations(recall that there is only 20% probability that a given observation is correct) there is bound to be considerable amount of uncertainty even as the number of observations increase.

```r
entropies <- c()
new_simHMM <- simHMM(hmm = hmm_model, length = 300)
new_obs <- new_simHMM$observation
Alpha_new <- exp(forward(hmm = hmm_model, observation = new_obs))
filtered <- apply(Alpha_new,2,prop.table)
for (i in seq(30, 300, 5)) {

  entropies <- c(entropies, entropy.empirical(filtered[,i]))
}

plot(seq(30, 300, 5),entropies, type = 'l')
```



Question 7

To compute the probability of hidden states for time step 101 given time step 100, we can write it as:

$$p(z^{T+1}|x^{1:T}) = \sum_{z^T} p(z^{T+1}, z^T|x^{1:T})$$

that is,

$$p(z^{101}|x^{1:100}) = \sum_{z^{100}} p(z^{101}, z^{100}|x^{1:100})$$
$$= \sum_{z^{100}} p(z^{100}|x^{1:100}) \cdot p(z^{101}|z^{100})$$

That is, the probability distribution of the hidden state at time step 101 is given by the product of the 100th time step of the filtered distribution and the transition matrix.

```
print('The probabilities of the hidden states of time step 101 given time step 100 is:')
```

```
## [1] "The probabilities of the hidden states of time step 101 given time step 100 is:"
```

```
transMat %*% sim100_results$filter_mat[,100]
```

```
##     [,1]
## S1  0.00
## S2  0.00
## S3  0.00
## S4  0.00
## S5  0.00
## S6  0.25
## S7  0.50
## S8  0.25
## S9  0.00
## S10 0.00
```