HW3

```r
library(tidyverse)
```

Given Quantities

- Transition Matrix

```r
transition_m =
  matrix(c(0.9, 0,  # first column - transition from states 1 and 2 to 1
           .1, 1),  # second column - transition from states 1 and 2 to 2
         nrow = 2, ncol = 2)


transition_m
```

```
     [,1] [,2]
[1,]  0.9  0.1
[2,]  0.0  1.0
```

- Starting Probabilities

```r
prob_X1_eq_1 = 0.8
prob_X1_eq_2 = 0.2
```

- Emission Probabilities

```r
emission_m =
  matrix(c(.99, .01,  # first column - transition from states 1 and 2 to 1
           .96, .04),  # second column - transition from states 1 and 2 to 2
```

```r
         nrow = 2, ncol = 2)

rownames(emission_m) = c("a", "u")

emission_m
```

```
  [,1] [,2]
a 0.99 0.96
u 0.01 0.04
```

Forward Algorith

```r
forward_HMM_k <- function(

    ## returns: joint pmf for a given observed signal
    ##             and possible hidden states of a markov chain

    TRANSITION_M,  ## obv. a matrix
    EMISSION_M, ## matrix of emitted probabilities
    EMITTED_SIGNALS,  ## needs to be a vector
    STARTING_P, ## needs to be a vector
    DESIRED_K
){

  ## initialize
  first_signal = EMITTED_SIGNALS[1] # first signal
  STARTING_P * EMISSION_M[rownames(EMISSION_M) == first_signal] -> working_F

  if(DESIRED_K > 1){
    ## future states 2, 3, etc...
    for( STEPS in 2:DESIRED_K){

      current_F = rep(NA, length(STARTING_P))

      for( STATES in 1:length(STARTING_P)){

        current_F[STATES] <-
          EMISSION_M[
            ## get conditional probs given potential HMM state
```

```r
            rownames(EMISSION_M) ==
              ## get probability for the Hidden state we work with here
                      EMITTED_SIGNALS[STEPS]][STATES]   *

            ## multiply previous Forward F's  with transition probabilities
           sum(working_F * TRANSITION_M[,STATES] )

       }

       working_F = current_F
     }
   }else{
       current_F = working_F
     }

   return(current_F)
 }
```

```r
### test the algo

forward_HMM_k(
  TRANSITION_M = transition_m,
  EMISSION_M = emission_m,
  EMITTED_SIGNALS = c("a", "u", "a"),
  STARTING_P = c(0.8, 0.2),
  DESIRED_K = 3
) %>% round(., 6)
```

```
[1] 0.006351 0.011098
```

Backward Algo

```r
backward_HMM_k <-
  function(
    TRANSITION_M,
    EMISSION_M,
    EMITTED_SIGNALS,
    STARTING_P,
    LAST_K
```

```r
  ){

  current_B = rep(1, length(STARTING_P))
  ## past states states 2, 1, etc... {2 and 1 is all we get in this example}

  backward_state_sequence = seq(from = length(EMITTED_SIGNALS) - 1, to = LAST_K, by = -1)

  for( STEPS in backward_state_sequence){

    working_B = rep(NA, length(STARTING_P))

    for( STATES in 1:length(STARTING_P)){

      working_B[STATES] <-
        sum(
          EMISSION_M[rownames(EMISSION_M) ==
                       EMITTED_SIGNALS[STEPS + 1]] *
          TRANSITION_M[STATES,] *
          current_B
        )
    }

    current_B = working_B

  }

  return(current_B)
  }
```

```r
backward_HMM_k(
  TRANSITION_M = transition_m,
  EMISSION_M = emission_m,
  EMITTED_SIGNALS = c("a", "u", "a"),
  STARTING_P = c(0.8, 0.2),
  LAST_K = 1
) %>% round(. ,4)
```

```
[1] 0.0127 0.0384
```

```r
backward_HMM_k(
  TRANSITION_M = transition_m,
  EMISSION_M = emission_m,
  EMITTED_SIGNALS = c("a", "u", "a"),
  STARTING_P = c(0.8, 0.2),
  LAST_K = 2
) %>% round(. ,4)
```

```
[1] 0.987 0.960
```

Conditional PMF

```r
conditional_pmf <-
  function(
    DESIRED_STATE,
    DESIRED_K_H,

    EMITTED_SIGNALS_H,
    TRANSITION_M_H,
    EMISSION_M_H,
    STARTING_P_H,

    LAST_K_H

  ){

    forward_HMM_k(
      TRANSITION_M = TRANSITION_M_H,
      EMISSION_M = EMISSION_M_H,
      EMITTED_SIGNALS = EMITTED_SIGNALS_H,
      STARTING_P = STARTING_P_H,
      DESIRED_K = DESIRED_K_H
    ) -> forward_part

    backward_part <-

      ifelse(
        ## if we want to predict X at the time of last observed
        ##    signal, then we will assign B_k all 1
```

```r
        ##    otherwise, we will use the fucntion to calculate our stuff
        DESIRED_K_H == length(EMITTED_SIGNALS_H),

        rep(1, length(STARTING_P_H)),

        backward_HMM_k(
          TRANSITION_M = TRANSITION_M_H,
          EMISSION_M = EMISSION_M_H,
          EMITTED_SIGNALS = EMITTED_SIGNALS_H,
          STARTING_P = STARTING_P_H,
          LAST_K = LAST_K_H
        )
      )

    ### now we need to recreate

    prod = forward_part * backward_part

    results = prod[DESIRED_STATE]/sum(prod)

    return(results)
  }
```

```r
conditional_pmf(
    DESIRED_STATE = 1,
    DESIRED_K_H = 3,
    TRANSITION_M_H = transition_m,
    EMISSION_M_H = emission_m,
    STARTING_P_H = c(0.8, 0.2),
    EMITTED_SIGNALS_H = c("a", "u", "a") ,

    LAST_K_H = 1
  ) %>% round(., 3)
```

```
[1] 0.364
```

Viterbi

```r
#initial

viterbi_algo <-
  function(
    STARTING_P,
    TRANSITION_M,
    EMITTED_SIGNALS,
    EMISSION_M
  ){

    ## Store two probabilities to chose from for V_{t}(j)
    v_t_j_probs = matrix(rep(NA, length(EMITTED_SIGNALS) * length(STARTING_P) * 2),
                        nrow = 4)

    ## Here we will store X_{t} states picked on the highest value of V_t(j) in each step
    x_star = rep(NA, length(EMITTED_SIGNALS))

    ## Store probabilities that were used to pick Xai_t_j at each step
    xai_t_j_probs = matrix(rep(NA, length(EMITTED_SIGNALS) * length(STARTING_P) * 2),
                        nrow = 4)

    ## here we will store values of Xai function to retrieve most likely values of X_{t}
    xai_star = matrix(rep(NA, length(EMITTED_SIGNALS) * length(STARTING_P)),
                        nrow = 2)

    #### **** STEP 1: Initialize ###
    initial_V = STARTING_P * EMISSION_M[rownames(EMISSION_M) == EMITTED_SIGNALS[1]]

    x_star[1] = which(initial_V == max(initial_V))

    ## loop over time points
    for(TIMES in 2:length(EMITTED_SIGNALS)){

      working_V = rep(NA, length(STARTING_P))
      working_X = rep(NA, length(STARTING_P))

      ## loop over the number of hidden states
      for(J in 1:length(STARTING_P)){
```

```r
    ## Calculate V's
    iter_Vs <-
      EMISSION_M[rownames(EMISSION_M) == EMITTED_SIGNALS[TIMES]][J] *
      TRANSITION_M[J,] * initial_V

    ## store both probabilities
    v_t_j_probs[c(J + (J-1), J + J) ,TIMES] = iter_Vs

    ## pick the highest of the two probabilities and store it as value of V_{t}(j)
    working_V[J] = max(iter_Vs)

    ## Calculate Xai's
    iter_Xs <- TRANSITION_M[,J] * initial_V
    ## store these probabilities
    xai_t_j_probs[c(J + (J-1), J + J) ,TIMES] = iter_Xs

    ## store index of state that corresponds to the highest probability
    xai_star[J, TIMES] <- which(iter_Xs == max(iter_Xs))

  }
  x_star[TIMES] = which(working_V == max(working_V))

  initial_V = working_V

}

recovered_states = rep(NA, length(EMITTED_SIGNALS))

# last recovered state is just the one picked based on the
# higest probability for the last state
recovered_states[length(EMITTED_SIGNALS)] = x_star[length(EMITTED_SIGNALS)]

## backward loop
for(j in seq(from = length(EMITTED_SIGNALS) - 1,
             to  = 1,
             by = -1)){

  ## current X_t is based on previous X_{t+1} and previous Xai_{t+1}
  ## X_{t+1} serves as an index for Xai_{t+1}
  ## value retrieved based on this index is assigned to X_t
  prev_x = x_star[j + 1]
```

```
      recovered_states[j] = xai_star[prev_x,(j+1)]
    }

    ## make matrices nicer for printing
    rownames(v_t_j_probs) = c("j = 1, index = 1", "j = 1, index = 2",
                              "j = 2, index = 1", "j = 2, index = 2")
    colnames(v_t_j_probs) = paste0("t = ", seq(from = 1, to = dim(v_t_j_probs)[2], by = 1)

    rownames(xai_t_j_probs) = c("j = 1, index = 1", "j = 1, index = 2",
                              "j = 2, index = 1", "j = 2, index = 2")
    colnames(xai_t_j_probs) = paste0("t = ", seq(from = 1, to = dim(xai_t_j_probs)[2], by

    ## print all results
    print("INTERMEDIATE RESULTS OF ESTIMATION")

    options(scipen = 999)
    print("")
    print("Estiamted Probabilties for chosing value of V_{t}(j) at each time point t")
    print(v_t_j_probs %>% round(., 4))

    print("")
    print("Indexes of higer value of V_{j} at each time point t")
    print(x_star)

    print("")
    print("Estiamted Probabilties for chosing value of Xai_{t}(j) at each time point t")
    print(xai_t_j_probs %>% round(., 4))

    print("")
    print("Indexes of higer value of Xai_{j} at each time point t")
    print(xai_star)

    return(recovered_states)
  }

viterbi_algo(
  STARTING_P = c(.8, .2),
  TRANSITION_M = transition_m,
  EMITTED_SIGNALS = c("a", "u", "a", 'u', 'a'),
  EMISSION_M = emission_m
) -> recovered_states_results
```

```
[1] "INTERMEDIATE RESULTS OF ESTIMATION"
[1] ""
[1] "Estiamted Probabilties for chosing value of V_{t}(j) at each time point t"
                 t = 1  t = 2  t = 3  t = 4  t = 5
j = 1, index = 1    NA 0.0071 0.0064 0.0001 0.0001
j = 1, index = 2    NA 0.0002 0.0008 0.0000 0.0000
j = 2, index = 1    NA 0.0000 0.0000 0.0000 0.0000
j = 2, index = 2    NA 0.0077 0.0074 0.0003 0.0003
[1] ""
[1] "Indexes of higer value of V_{j} at each time point t"
[1] 1 2 2 2 2
[1] ""
[1] "Estiamted Probabilties for chosing value of Xai_{t}(j) at each time point t"
                 t = 1  t = 2  t = 3  t = 4  t = 5
j = 1, index = 1    NA 0.7128 0.0064 0.0057 0.0001
j = 1, index = 2    NA 0.0000 0.0000 0.0000 0.0000
j = 2, index = 1    NA 0.0792 0.0007 0.0006 0.0000
j = 2, index = 2    NA 0.1920 0.0077 0.0074 0.0003
[1] ""
[1] "Indexes of higer value of Xai_{j} at each time point t"
     [,1] [,2] [,3] [,4] [,5]
[1,]   NA    1    1    1    1
[2,]   NA    2    2    2    2
```

Final Answer

```
Sequence of states that resulted in the higest
likelihood of emitted signals (0,1,0,1,0) is:

[1] 2 2 2 2 2
```

Just Final Answers

$F_n(j)$ from forward equations:

- $F_1(j)$:

```
[1] 0.792 0.192
```

- $F_2(j)$:

```
[1] 0.007128 0.010848
```

- $F_3(j)$:

```
[1] 0.006351 0.011098
```

$B_k(j)$ **from backward equations for k = 1,2:**

- $B_1(i)$:

```
[1] 0.0127 0.0384
```

- $B_2(i)$:

```
[1] 0.987 0.960
```

- $B_3(i)$:

```
[1] 1 1
```

$P(X_3 = 1|s_3 = (0, 1, 0))$ **using forward and backward equations:**

```
[1] 0.364
```

Likely sequence of states given the sequence of signals s_5 **= (0, 1, 0, 1, 0) is:**

```
[1] 2 2 2 2 2
```