Hidden Markov Models: Overview, Examples, and Applications

W. Bliss, J. Canfield, J. Lovato, D.R. Schmidt

March 13, 2023

1 Introduction to Hidden Markov Models (HMMs)

Many real-world scenarios can be characterized as sequences of events, i.e. transitions between states, where the probability of the scenario being in any state is solely dependent on the previous state. This condition of the probability of transitioning to a state depending only on the prior state is what defines a Markov Chain, or Markov Process. For example, say the weather on a given day can be either sunny or rainy. Then, the probability of tomorrow's weather is only dependent on the the weather today, and the weather on all previous days has no influence. Here, the weather can be modeled as a Markov Chain.

Building on the idea of a Markov Chain is a Hidden Markov Model (HMM). The 'backbone' of an HMM is just a regular Markov Chain. However in this case, the states are unobservable, or hidden. Each of these hidden states have some influence on the states that are observable, sometimes called signals or symbols. Hence, there are transition probabilities between the hidden states, as well as "emission" probabilities; the probabilities of an observation occurring given the current hidden state.

In this paper, we aim to:

1. Give an overview of HMMs, introduce basic notation and main problems studied using HMMs.
2. Give various examples to illustrate the theory of HMMs.
3. Use HMMs to study various problems in applications such as political science/voting systems, financial time series, and effect of mental state on video game performance.

We also provide R code that provides tools to compute the forward algorithm, the Viterbi algorithm, and code to create all figures provided in the examples.

1.1 Definitions

We will mainly follow notation in Rabiner (1989).

- Stochastic Process: A collection of random variables $\{X(t) : t \in T\}$. For each $t \in T$, $X(t)$ is a random variable. $X(t)$ is the state of the process at time t
- Markov Property: The conditional distribution of the future state given the current state and the past is independent of the past:

$$P(X_{n+1} = j | X_0 = i_0, X_1 = i_1, \dots, X_{n-1} = i_{n-1}, X_n = i) = P(X_{n+1} = j | X_n = i)$$

- Markov Chain: A type of stochastic process which has the Markov Property

- Hidden Markov Model: A model in which hidden states operate as a Markov Chain, where each state influences the probability of some observation occurring.
- Hidden States: The unobservable states that produce the observed sequence.
- Observed States: The observable events of the process, which are produced by the underlying hidden states.
- Transition Matrix: The matrix of transition probabilities from every hidden state to each other. If there are N hidden states, the Transition Matrix is NxN.
- Emission Matrix: The matrix of probabilities of each observation occurring given the underlying hidden state. If there are N hidden states and M possible observations, the Emission Matrix is NxM.
- Initial Probabilities: The probabilities of the model starting in each state. If there are N hidden states, the initial probabilities, notated π, is a vector of length N.

1.2 Symbols

- N: Number of hidden states in the model
  - Individual states are $S = \{S_1, S_2, \ldots, S_N\}$
  - State at time t is q_t
- M: Number of distinct observations "symbols" per state; the number of observable events
  - Individual symbols (observations) are $V = \{v_1, v_2, \ldots, v_n\}$
- a_{ij}: State transition probability
  - $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$, $i \geq 1$, $j \leq N$
- A: Transition probability matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1N} \\ a_{21} & a_{22} & & a_{2N} \\ \vdots & & \ddots & \\ a_{N1} & a_{N2} & & a_{NN} \end{bmatrix}$$

- $b_j(k)$: Emission probability; the observation symbol probability in state j
  - $b_j(k) = P(v_k \text{ at } t | q_t = S_j)$, $1 \leq j \leq N$, $1 \leq k \leq M$: Probability of observing v_k at time t given we are in state S_j at time t
- B: Emission Matrix; it gives the probability of observing any symbol v_k given the system is in any state S_k

$$B = \begin{bmatrix} b_1(1) & b_1(2) & \ldots & b_1(M) \\ b_2(1) & b_2(2) & & b_2(M) \\ \vdots & & \ddots & \\ b_N(1) & b_N(2) & & b_N(M) \end{bmatrix}$$

- π: Initial state distribution
  - $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$
  - $\pi_i = P(q_1 = S_i)$, $1 \leq i \leq N$
- $\lambda = (A, B, \pi)$: The complete parameter set of the HMM

1.3 Simple Example

This example illustrates the fundamental ideas behind a Hidden Markov Model.

Say you have 3 urns, each containing balls of S colors: pink, blue, purple, green, and orange. A person draws one ball at each time step. The urn they choose from is hidden from you. The urns are the states, so N=3 and the transition probability matrix A is

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

For example, if they pick from urn 1 at time t, they pick from urn 3 at time $t+1$ with probability a_{13}.

There are 5 colors of balls, so $M = 5$. Let $v_1 =$ "you see pink ball", $v_2 =$ "you see blue ball", $v_3 =$ "you see purple ball", $v_4 =$ "you see green ball", and $v_5 =$ "you see orange ball". Then

$$B = \begin{bmatrix} b_1(1) & b_1(2) & b_1(3) & b_1(4) & b_1(5) \\ b_2(1) & b_2(2) & b_2(3) & b_2(4) & b_2(5) \\ b_3(1) & b_3(2) & b_3(3) & b_3(4) & b_3(5) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{2}{9} & \frac{1}{9} & \frac{1}{9} & \frac{3}{9} & \frac{2}{9} \\ \frac{1}{8} & \frac{1}{8} & \frac{2}{8} & \frac{3}{8} & \frac{1}{8} \\ \frac{3}{10} & \frac{2}{10} & \frac{1}{10} & \frac{1}{10} & \frac{3}{10} \end{bmatrix}$$

Thus, $b_2(5) = \frac{3}{10}$ is the probability of picking an orange ball given the person is picking from urn 3.

1.4 The Three Basic Problems for HMMs

Problem 1:
Compute $P(V|\lambda)$. That is, given the observation sequence $V = v_1, v_2, \ldots, v_T$ and model $\lambda = (A, B, \pi)$, compute the probability of observing the observation sequence. This is done using forward procedure.

Problem 2:
Given the observation sequence $V = v_1, v_2, \ldots, v_T$ and model λ, choose the corresponding hidden state sequence $Q = q_1, q_2, \ldots, q_T$ that best explain the observations. This can be done using the Viterbi algorithm.

Problem 3:
Choose model parameters $\lambda = (A, B, \pi)$ to maximize $P(V|\lambda)$. This can be done using Expectation-Maximization algorithms such as the Baulm-Welch algorithm.

2 Applications Involving HMMs

Hidden Markov Models have many real world applications and can model natural phenomena. We will consider problems involving financial time series analysis of stock market movement, political parties and survey data, and analysis of psychological states to predict video game performance.

2.1 Prediction of Financial Time Series with Hidden Markov Models

The following application is derived from Zhang (2004).

2.1.1 A Markov Chain of Stock Market Movement

Observe a five state model with state space $S = \{\text{large rise}, \text{ small rise}, \text{ no change}, \text{ small drop}, \text{ large drop}\}$. Without loss of generality, let $S = \{1, 2, 3, 4, 5\}$.

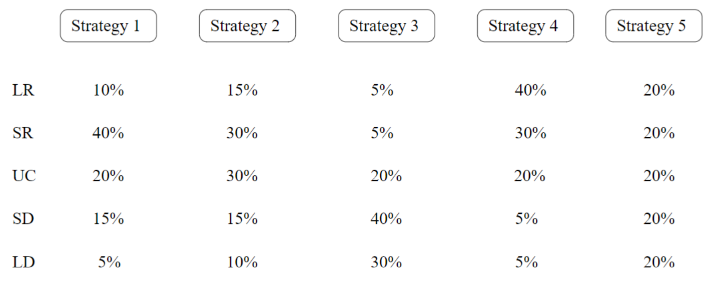
Our MC has the following main idea: What is the probability of seeing a certain pattern of events? For example, what is the probability of seeing "small drop, no change, small rise, no change, no change, large rise" in six consecutive days?

Let $X = \{4, 3, 2, 3, 3, 1\}$, then

$$\begin{aligned} P(X|A, \pi) &= P(4, 3, 2, 3, 3, 1|A, \pi) \\ &= P(4)P(3|4)P(2|3)P(3|2)P(3|3)P(1|3) \\ &= \pi \cdot a_{43} \cdot a_{32} \cdot a_{23} \cdot a_{33} \cdot a_{31} \end{aligned}$$

2.1.2 A Hidden Markov Model of Stock Market Movement

We can extend the process above into a Hidden Markov Model with discrete observations. In an HMM of Stock Market Movement, one does not know anything about what generates the observation sequence. Thus, the number of states, transition probabilities, and from which state an observation is generated is unknown. Instead of combining states with deterministic output (large rise, small drop, etc.), each state of an HMM is associated with a probabilistic function. At time t, an observation v_k is generated by a probabilistic function $b_j(k)$. The hidden states of the HMM are five strategies that a financial professional can take. Each strategy is associated with a probability of generating the possible stock price movement. These probabilities are hidden to the public. Each strategy has a different initial probability distribution π for all movement symbols:



	Strategy 1	Strategy 2	Strategy 3	Strategy 4	Strategy 5
LR	10%	15%	5%	40%	20%
SR	40%	30%	5%	30%	20%
UC	20%	30%	20%	20%	20%
SD	15%	15%	40%	5%	20%
LD	5%	10%	30%	5%	20%

Figure 1: The set of strategies that a financial professional can take. These probabilities are hidden to the public.

The HMM will pick a strategy based on the observation probability. The HMM can pick the best overall sequence of strategies based on an observed sequence. We have the following transition probability matrix A:

Step 1: Choose an initial state (a strategy) $q_1 = S_i$ according to the initial state distribution π.

Step 2: Let $t = 1$, since we are at the first day of the sequence.

Step 3: Get the prediction according to the strategy of the current state i, i.e. $b_j(k)$. Then, we get the

Probability	Strategy1	Strategy2	Strategy3	Strategy4	Strategy5
Strategy1	10%	10%	20%	30%	30%
Strategy2	10%	20%	30%	20%	20%
Strategy3	20%	30%	10%	20%	20%
Strategy4	30%	20%	20%	15%	15%
Strategy5	30%	20%	20%	15%	15%

Figure 2: State transition probability matrix of the HMM.

prediction for the current day. For example, the probability of seeing a large rise prediction on the first day using Strategy 2 is 15%.

Step 4: Transition to new state q_{t+1} according to the state transition probability distribution

Step 5: Let $t = t + 1$. Return to Step 3 if $t \leq T$, otherwise terminate.

What is the probability of seeing the prediction sequence small drop, small rise if the prediction starts by taking strategy 2 on the first day?

Sum up all probabilities the movement sequence may have through different state sequences. We know the prediction starts with strategy 2, so $q_1 = 2$. Then there are 5 possible states for the second day. So, the probability of seeing the movement sequence small drop, small rise is

$$0.15 + 0.1 \cdot 0.4 + 0.2 \cdot 0.3 + 0.3 \cdot 0.05 + 0.2 \cdot 0.3 + 0.2 \cdot 0.2 = 0.365$$

2.2 A Hidden Markov Model in Political Science

Consider a U.S. political policy questionnaire used to determine which political party a user most aligns with. The political parties are the hidden states and their answers to the questionnaire are the observed states. For the sake of simplicity, we shall consider there to be three parties in the U.S. : Republican, Democrat, and Independent. We shall define the initial state distribution to be:

$$\pi = [\pi_1, \pi_2, \pi_3] = [0.26, 0.32, 0.42]$$

Where π_1, π_2, π_3 represents the initial probabilities of being a Republican (R), Democrat (D), or Independent (I) respectively.

Let us denote our transition probability matrix as follows:

$$A = \begin{array}{c|ccc} & R & D & I \\ \hline R & 0.8 & 0.05 & 0.15 \\ D & 0.05 & 0.8 & 0.15 \\ I & 0.1 & 0.1 & 0.8 \end{array}$$

Let us consider a 5 question policy questionnaire where users can answer "Agree", "Neutral", or "Disagree" to each policy statement. We shall consider a users entire answer sequence to be one observed state. (For example, one answer sequence may look like "Agree, Disagree, Disagree, Agree, Neutral"). Thus, we shall have a total of $3^5 = 243$ possible observed states. Let us denote each sequence s_k, for $k \in [1, 243]$.

Suppose 100 people took this questionnaire, indicating which party they currently affiliate with. We can use these results to create our emission probability matrix.

$$A = \begin{array}{c} \\ R \\ D \\ I \end{array} \begin{array}{ccccccc} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & \dots \\ \end{array}$$

$$A = \begin{array}{c|ccccccc} & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & \dots \\ \hline R & 0.02 & 0.03 & 0.05 & 0.08 & 0.3 & 0.2 & \dots \\ D & 0.3 & 0.25 & 0.08 & 0.06 & 0.01 & 0.02 & \dots \\ I & 0.05 & 0.09 & 0.4 & 0.1 & 0.04 & 0.03 & \dots \end{array}$$

(note that all rows would sum to one)

Thus, we have the following Hidden Markov Model Diagram:

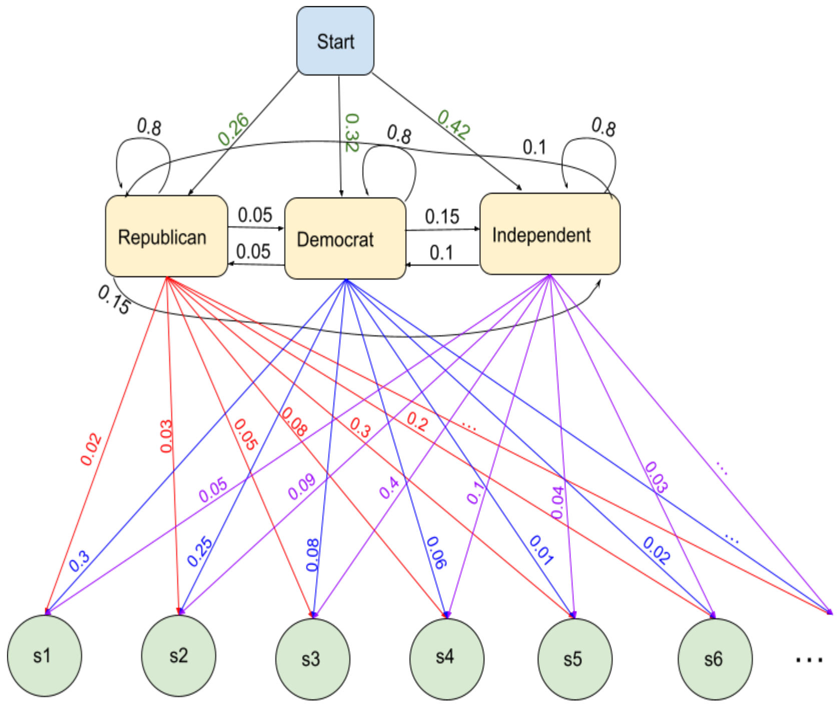


Figure 3: HMM transition diagram where there are three hidden states: Republican, Democrat, and Independent and 243 observed states (s1...s243) representing the answers to the questionnaire.

Suppose a given user answered s_1 in 2016, s_2 in 2018, and s_3 in 2020, and we wanted to know what is the most likely party they voted for in each of those years. This is related to question 2 of the 3 important questions of Hidden Markov models, which concerns computing the most likely series of states that generated a given observation sequence. We can use the Viterbi Algorithm to answer this question. This can be done using the 'hmm' package in R (See Appendix).

2.3 Hidden Markov Model to Predict Mental State

Suppose one tracks their activities over a period of time and also notes their mental state during the activities. Then, given enough data, the Viterbi Algorithm can be used to predict the (hidden) sequence of one's mental state given the observed future sequence of activities.

In one such instance, a man played the video game Fortnite, in which 100 players are dropped onto a virtual island and fight to be the last remaining player. The man tracked his placement for each round of Fortnite he played (i.e. did he place 1st, 2nd, 15th, etc.) as well as his corresponding mental state during each round (focused or distracted). Break the placements into deciles (Decile 1: Placed top 10, Decile 2: Placed 11-20, etc.) and note the man never placed lower than 66th place. The transition probability matrix A could be calculated from finding the probability of the mental state changing from subsequent rounds using the function createSequenceMatrix() from the 'markovchain' R package:

$$A = \begin{array}{r} \\ Distracted \\ Focused \end{array}\begin{array}{c} \begin{array}{cc} Distracted & Focused \end{array} \\ \begin{bmatrix} 0.8684211 & 0.1315789 \\ 0.1219512 & 0.8780488 \end{bmatrix} \end{array}$$

Similarly, the emission matrix B can be found by calculating the probability of placing in each decile given the man's mental state:

$$B = \begin{array}{r} \\ Distracted \\ Focused \end{array}\begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0.1315789 & 0.2894737 & 0.3421053 & 0.15789474 & 0.02631579 & 0.02631579 & 0.02631579 \\ 0.2142857 & 0.2142857 & 0.3809524 & 0.07142857 & 0.11904762 & 0.00000000 & 0.00000000 \end{bmatrix} \end{array}$$

From the A and B matrices (as well as initial state probabilities set to 0.5), the HMM package in R was used to simulate 2000 theoretical rounds of Fortnite, and then the Viterbi algorithm was used to predict the most likely mental state the man was in during the sequence of games. See plot below. The first seven rows plot the placement for each simulated round of Fortnite. The top colored bar plots the corresponding mental state for each simulated rounnd, where green is focused and red is distracted. The bottom colored bar plots the Viterbi algorithm's predicted path of mental states, which was calculated from the simulated round placements. Approximately 66% of the sequence of states was correctly predicted by the Viterbi Algorithm. See appendix for R code.

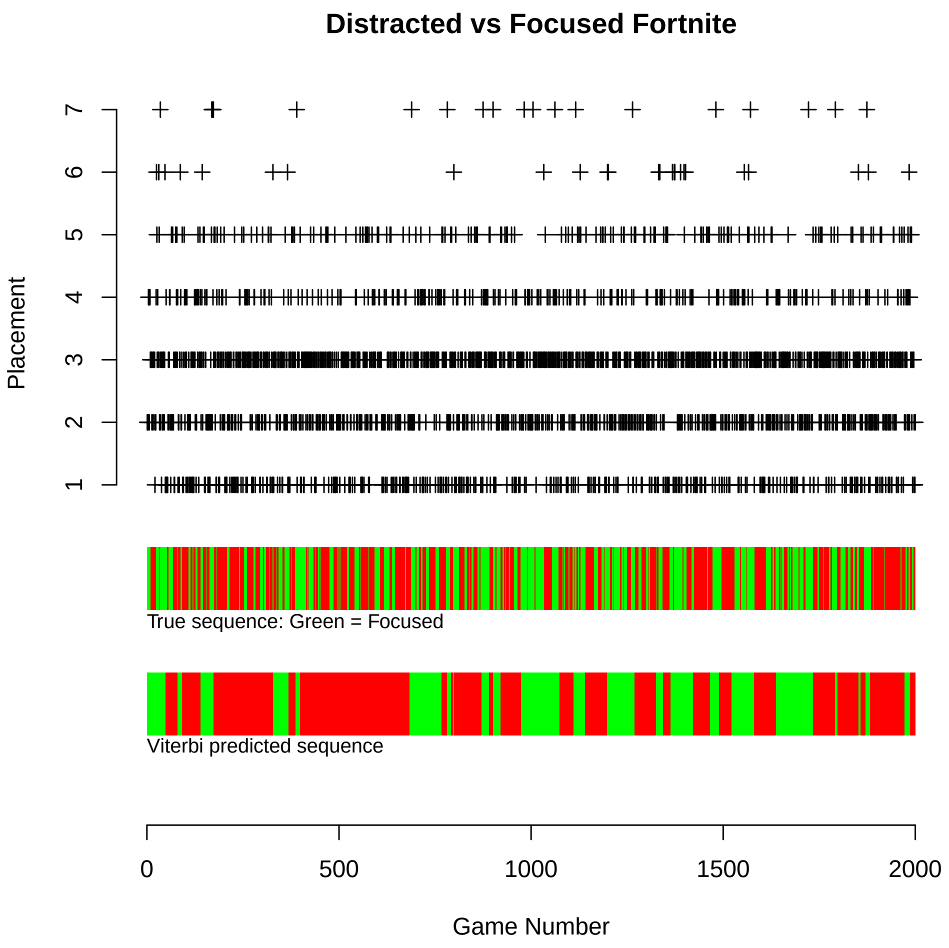


Figure 4: Simulation of 2000 Rounds of Fortnite with Accompanying True Hidden States and Predicted States from Viterbi Algorithm

3 Algorithms

3.1 Forward Algorithm

This code was written based on the algorithm in Rabiner (1989). The forward algorithm is useful in calculating $P(O|\lambda)$, where $O = O_1, O_2, ...$ is a sequence of observations, and $\lambda = (\pi, A, B)$ is the set of parameters that define the HMM. Hence, $P(O|\lambda)$ tells us the probability of some sequence of observations occurring, given some HMM.

The forward algorithm calculates forward probabilities, $\alpha_t(i)$, i.e. the probability of the partial sequence $O_1, O_2, ..., O_t$ until time t and state q_i given the model λ. After computing all partial sequence probabilities, these values are summed up to return $P(O|\lambda)$.

```
forward_alg <- function(A, B, pi, N, O){
  #Parameters:
  #A is hidden state transition probability matrix
  #B is emission probability matrix
  #pi is vector of initial state probabilities
  #N a scalar representing the number of hidden states
  #O is vector of observations that the function finds the probability
  #of occurring given the model

  T = length(O)

  alpha <- matrix(numeric(0), nrow = T, ncol = N)

  #Step 1: initialize
  alpha[1, ] <- pi * B[, O[1]]

  #Step 2: induction
  for(t in 1:(T - 1)){
    prev_alpha = alpha[t, ]
    alpha[t + 1, ] = (prev_alpha %*% A) * B[, O[t + 1]]
  }

  return(alpha)
}

#To find P(O|lambda), i.e. the probability of getting observation sequence O, sum
#the last row of alpha, the outputt of the forward algorithm
sum(alpha[nrow(alpha), ])
```

3.2 Viterbi Algorithm

The Viterbi Algorithm takes in a sequence of observations and returns the most likely sequence of hidden states that corresponds to the sequence of observations. We use the function 'viterbi' provided by the 'HMM' R package. A simple implementation is as follows: Say the weather is either rainy or sunny. If it is sunny today, tomorrow will be rainy with probability 0.5 and sunny with probability 0.5. If it is rainy today, tomorrow will be rainy with probability 0.3 and sunny with probability 0.7. Each day, you either get a car wash or do not get a car wash. If it is sunny, you get a wash with probability 0.5 and no wash with probability 0.5. If it is rainy, you get a wash with probability 0.1 and no wash with probability 0.9. The following R code simulates 10 days of observations; that is, whether you get a car wash or no car wash each day for 10 days. That sequence is then input into the Viterbi algorithm and the most likely sequence of weather is returned.

```
library(HMM)

states <- c("sunny", "rainy")
symbols <- c("wash", "no wash")
A <- matrix(c(0.5, 0.5,
              0.7, 0.3), nrow = 2, byrow = TRUE)
B <- matrix(c(0.7, 0.3,
              0.1, 0.9), nrow = 2, byrow = TRUE)
rownames(B) <- states
colnames(B) <- symbols
pi <- c(0.5, 0.5)

hmm <- initHMM(states,
               symbols,
               transProbs = A,
               emissionProbs = B,
               startProbs = pi)

sim <- simHMM(hmm, 10)
vit <- viterbi(hmm, sim$observation)
sim
vit
#sim = "wash" "wash" "no wash" "wash" "no wash" "wash" "no wash" "wash" "no wash" "wash"
#vit "sunny" "sunny" "rainy" "sunny" "rainy" "sunny" "rainy" "sunny" "rainy" "sunny"
```

We see that if the sequence of observations is

wash, wash, no wash, wash, no wash, wash, no wash, wash, no wash, wash

then the most likely sequence of weather that caused this sequence of observations is

sunny, sunny, rainy, sunny, rainy, sunny, rainy, sunny, rainy, sunny

A R Code

A.1 Political Science Example

The following code uses the Viterbi algorithm to answer the question posited in 2.2.

```
#load hmm package
library(HMM)

#define hidden states, observed states (symbols), starting probabilities,
#transition probability matrix, and emission probability matrix

States <- c('Rep', 'Dem', 'Ind')
Symbols <- c('s1','s2','s3','s4','s5','s6')
start_p <- c(0.26,0.32,0.42) #R D I
trans_p <- matrix(c(0.8,0.05,0.15,
                    0.05,0.8,0.15,
                    0.1,0.1,0.8),nrow=3,ncol=3,byrow=TRUE)
emit_p <- matrix(c(0.02,0.03,0.05,0.1,0.45,0.35,
                   0.5,0.3,0.08,0.09,0.01,0.02,
                   0.1,0.115,0.4,0.2,0.115,0.07),nrow=3,ncol=6,byrow=TRUE)

#create the hidden markov model
hmm = initHMM(States,Symbols,start_p,trans_p,emit_p)

#test a few observation sequences
observation = c('s1','s2','s3')
observation2 = c('s4','s2','s6')
observation3 = c('s6','s1','s1')

#Run the viterbi function to see what the most likely path of hidden states were for each
#observation sequence

viterbi(hmm,observation)
viterbi(hmm,observation2)
viterbi(hmm,observation3)

#create all possible observation sequences (of size 3)
combinations <- combn(Symbols,m=3)

#Run the viterbi function for all the sequences
viterbi_paths <- matrix(data=NA,nrow=3,ncol=20)
for (i in 1:20){
  viterbi_paths[,i] <- viterbi(hmm,combinations[,i])
}

#label sequences in matrix
comb_names <- c()
for (i in 1:20){
  comb_names[i] <- c(paste(combinations[,i],collapse='-'))
}
```

```
colnames(viterbi_paths) <- comb_names

#print matrix containing all the paths
viterbi_paths
```

A.2 Problem 3

```
library(HMM)

## Make a known model
states <- c(1,2,3)
symbols <- c(1,2)
startProb <- c(0.5,0.25,0.25)
transProb <- matrix(c(0.8,0.05,0.15,
                      0.2,0.6,0.2,
                      0.2,0.3,0.5),3,3,TRUE)

emissionProb <- matrix(c(0.9,0.1,
                         0.2,0.8,
                         0.7,0.3), 3,2,TRUE)

hmmTrue <- initHMM(states, symbols, startProb, transProb , emissionProb)

## Create a simulation
observation <- simHMM(hmmTrue, 1000)
obs <- observation$observation

## Estimate parameters based off of simulation
hmmInit <- initHMM(states, symbols, c(4,2,1)/7)
hmmFit <- baumWelch(hmmInit, obs)
hmmFit$hmm$emissionProbs
hmmFit$hmm$transProbs
```

A.3 Fortnite Example

```
#Analysing Fortnite player data using Hidden Markov Model
#Data from:
#https://www.kaggle.com/datasets/johnharshith/fortnite-statistics-corresponding-to-mental-state

library(dplyr)
library(magrittr)
library(markovchain)
library(HMM)

# Want to predict whether the player was distracted or focused (hidden states)
# given their placement in the match

fortnite_full <- read.csv("FortniteStatistics.csv",header = T)

fortnite <- fortnite_full %>%
  select("Mental.State", "Placed") %>%
  mutate(Placed = floor(Placed / 10) + 1)
```

```
# Need to calculate A: transition probability matrix of focused and distracted
# The data is arranged in order of time
A <- createSequenceMatrix(fortnite$Mental.State, toRowProbs = TRUE)
A
# Create B: emission matrix
num_focused <- nrow(fortnite[which(fortnite$Mental.State == "focused"),])
num_distracted <- nrow(fortnite[which(fortnite$Mental.State == "distracted"),])

focused <- list()
for(i in 0:max(fortnite$Placed)){
  focused[i] <- nrow(fortnite[which(fortnite$Mental.State == "focused" & fortnite$Placed == i),])
}

distracted <- list()
for(i in 0:max(fortnite$Placed)){
  distracted[i] <- nrow(fortnite[which(fortnite$Mental.State == "distracted" & fortnite$Placed == i),])
}

B <- matrix(c(unlist(distracted[1]) / num_distracted,
unlist(distracted[2]) /  num_distracted,
unlist(distracted[3]) / num_distracted,
unlist(distracted[4]) / num_distracted,
unlist(distracted[5]) / num_distracted,
unlist(distracted[6]) / num_distracted,
unlist(distracted[7]) / num_distracted,

unlist(focused[1]) / num_focused,
unlist(focused[2]) /  num_focused,
unlist(focused[3]) / num_focused,
unlist(focused[4]) / num_focused,
unlist(focused[5]) / num_focused,
unlist(focused[6]) / num_focused,
unlist(focused[7]) / num_focused),

nrow = 2, byrow = TRUE)
rownames(B) <- c("distracted", "focused")
colnames(B) <- c(1, 2, 3, 4, 5, 6, 7)
B

# Initial probabilities
pi <- c(num_distracted / (num_distracted + num_focused),
1 - (num_distracted / (num_distracted + num_focused)))

states <- c("distracted", "focused")
symbols <- c("1", "2", "3", "4", "5", "6", "7")



hmm <- initHMM(states,
               symbols,
               transProbs = A,
               emissionProbs = B,
               startProbs = pi)
```

```
### Simulate HMM - Modified code from HMM package function dishonestCasino()
nSim <- 2000
sim <- simHMM(hmm, nSim)
vit <- HMM::viterbi(hmm, sim$observation)
f <- forward(hmm, sim$observation)
b <- backward(hmm, sim$observation)

f[1,nSim]->i
f[2,nSim]->j
probObservations = (i + log(1+exp(j-i)))
posterior = exp((f+b)-probObservations)
x = list(hmm=hmm,sim=sim,vit=vit,posterior=posterior)

#Plot simulated games
mn = "Distracted vs distracted Fortnite"
xlb = "Game Number"
ylb = "                                    Placement"
plot(x$sim$observation,ylim=c(-4,7),pch=3,main=mn,xlab=xlb,ylab=ylb,bty="n",yaxt="n")
axis(2,at=1:7)

#Plot true hidden states
text(0,-1.2,adj=0,cex=.8,col="black","True sequence: Green = distracted")
for(i in 1:nSim)
{
  if(x$sim$states[i] == "distracted")
    rect(i,-1,i+1,0, col = "green", border = NA)
  else
    rect(i,-1,i+1,0, col = "red", border = NA)
}

#Plot most probable path
text(0,-3.2,adj=0,cex=.8,col="black","Viterbi predicted sequence")
for(i in 1:nSim)
{
  if(x$vit[i] == "distracted")
    rect(i,-3,i+1,-2, col = "green", border = NA)
  else
    rect(i,-3,i+1,-2, col = "red", border = NA)
}
```

References

Rabiner, L. R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77:257–286.

Zhang, Y. 2004. Prediction of financial time series with hidden markov models. Ph.D. thesis. Applied Sciences: School of Computing Science.