CS 230
Problem Set 6
Probability, Hidden Markov Models, and the Viterbi Algorithm

1 Background Information

1.1 Hidden Markov Model

In this problem set, you will be working with the Hidden Markov Model (HMM). To understand it, we first start by introducing the Markov Model. Mathematically, a model is a description of a system using mathematical concepts and language. A Markov Model is a stochastic model that has the Markov property. Given a set of $n$ states

$$X = \{ x_1, x_2, \ldots, x_n \},$$

stochastic means that each state $x_i$ depends on previous states in a non-deterministic way. As time goes on, the system changes from one state ($x_i$) to the next ($x_{i+1}$). Thus, if the system is currently in state $x_i$, the set of past states is $X_p(x_i) = \{ x_j | 1 \leq j < i, i, j \in \mathbb{N} \}$, and the set of future states is $X_f(x_i) = \{ x_j | i < j \leq n, i, j \in \mathbb{N} \}$. The Markov property states that future states (which, in theory, could depend on both current and past states) only depend on the current state, and not the sequence of events that preceded it. In other words, given the current state, the future does not depend on the past.

The Hidden Markov Model and the Viterbi algorithm (to be introduced below) are useful in medical diagnosis (physicians probably don’t actually know this much probability, but they should!). We will use a simplified example, but you can imagine how this would extend to a whole range of illnesses and symptoms.

Suppose that there are two different states of health: Healthy ($H$), and Sick ($S$), and a person’s state of health can potentially change every day. Our mathematically sophisticated physician can’t observe directly whether someone is healthy or sick on a given day, but he can observe symptoms. By observing symptoms on a sequence of days, the physician needs to decide on which days the patient was healthy, and on which days s/he was sick.

Assume that on the first day that the physician starts observing the patient, the patient is healthy with probability 60% and sick with probability 40%. The physician knows that if the patient is healthy, he will report feeling normal with probability 50%, cold with probability 40%, and dizzy with probability 10%. But if the patient is sick, he will report feeling dizzy with probability 30%, cold with probability 30%, and normal with probability 40%. Additionally, the physician knows that if a patient is healthy one day then he will be healthy the next day with 70% probability, and if the patient is sick one day then he will be sick the next day with probability 60%. Note that this illustrates the Markov property, since there is no dependence on the patient’s health more than one day before the day we are interested in. The probabilities are summarized in Table 1 below.
Table 1: Relevant probabilities for the medical diagnosis example

<table>
<thead>
<tr>
<th>State</th>
<th>Symptoms</th>
<th>Next State</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>Cold</td>
<td>Dizzy</td>
</tr>
<tr>
<td>Healthy</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>Sick</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

We can represent this model succinctly with a state diagram, as shown in Figure 1. Here, the blue edges are from the state Healthy ($H$), and the red edges are from the state Sick ($S$). The color of the probability values correspond to the color of the edges.

Figure 1: State diagram for the medical diagnosis example

In a Markov Model, the states are completely visible to you. The state which the system is in can be completely determined by observing the output of the system. In a Hidden Markov Model, however, the states are only partially visible to you. In other words, the outputs that you can see are related to the states, but they are not sufficient to completely determine the states. Here, the outputs are called emissions and we use $Y = \{ y_1, y_2, \ldots, y_n \}$ to denote the set of all emissions.
Figure 2 illustrates a general HMM. $x_1 \in X$ is the 1st state of the model, and $x_i \in X$ represents the $i$th state. All the states are “hidden”, which means that they are invisible. Each $y_i \in Y$, which is visible, is the emission of state $x_i \in X$. In our example, each $x_i$ is the patient’s health on day $i$, so it could take the value $H$ or $S$. Each $y_i$ is an emission, which in our example could be that the patient reports feeling dizzy, feeling cold, or feeling normal on day $i$.

The probability of changing from one state $x_i$ to the next $x_{i+1}$ i.e., $\Pr(x_{i+1}|x_i)$ is called a transition probability. In our example, the complete set of transition probabilities are

$$\begin{align*}
\Pr(x_{i+1} = S|x_i = S) &= 0.6 \\
\Pr(x_{i+1} = H|x_i = S) &= 0.4 \\
\Pr(x_{i+1} = S|x_i = H) &= 0.3 \\
\Pr(x_{i+1} = H|x_i = H) &= 0.7.
\end{align*}$$

Figure 2 and the above equations illustrate the Markov property: the state $x_{i+1}$ only depends on the state $x_i$. Transition probabilities thus provide enough information to move from state to state in a memoryless fashion.

The probability of emitting $y_i$ in state $x_i$ i.e., $\Pr(y_i|x_i)$ is called an emission probability. In our example, the complete set of emission probabilities are

$$\begin{align*}
\Pr(y_i = \text{dizzy}|x_i = S) &= 0.3 \\
\Pr(y_i = \text{cold}|x_i = S) &= 0.3 \\
\Pr(y_i = \text{normal}|x_i = S) &= 0.4 \\
\Pr(y_i = \text{dizzy}|x_i = H) &= 0.1 \\
\Pr(y_i = \text{cold}|x_i = H) &= 0.4 \\
\Pr(y_i = \text{normal}|x_i = H) &= 0.5.
\end{align*}$$

Given the $i$th state $x_i$, the emission $y_i$ can take on multiple values with corresponding probabilities, and is only dependent on $x_i$. 
Now, given the symptoms on each day, what do we know about whether a person is healthy or not? Or more generally, given a sequence of emissions, what do we know about the hidden states? That’s where the Viterbi algorithm comes in.

1.2 Viterbi Algorithm

The Viterbi algorithm is used to find the most likely sequence of hidden states given a sequence of emissions. It is particularly useful when the state can change between emissions. This sequence of hidden states is usually called the (Viterbi) best path. Suppose that \( a \) is an element in the set \( W \) of all values that can be taken by random variable \( x_i \) (the \( i \)th state), the Viterbi algorithm states that

\[
V(a, i + 1) = \max \{ V(b, i) \cdot \Pr(x_{i+1} = a | x_i = b) \cdot \Pr(y_{i+1} | x_{i+1} = a) \mid b \in W \} \tag{1}
\]

We have used set theory to write Eq. (1), but note that most mathematicians would write

\[
V(a, i + 1) = \max_{b \in W} ( V(b, i) \cdot \Pr(x_{i+1} = a | x_i = b) \cdot \Pr(y_{i+1} | x_{i+1} = a)) \tag{2}
\]

instead. In Eq. (1),

- \( V(a, i + 1) \) is the probability of the best path that ends with \( a \) as the value of the \((i + 1)\)st state.
- \( \Pr(x_{i+1} = a | x_i = b) \) is the transition probability from state \( x_i \) to state \( x_{i+1} \) where \( x_i \) can take any value \( b \in W \) and \( x_{i+1} \) takes the value \( a \).
- \( \Pr(y_{i+1} | x_{i+1} = a) \) is the emission probability of \( y_{i+1} \) at the \((i + 1)\)st state given that it takes the value \( a \).

To interpret Eq. (1), consider the \( i \)th state and the \((i + 1)\)st state. The \( i \)th state can take on any value \( b \in W \), and for each \( b \), there is a corresponding best path that ends with the \( i \)th state being that \( b \) and its associated probability \( V(b, i) \). Also, for each \( b \), the probability of the path leading to the \((i + 1)\)st state being \( a \) can be computed by multiplying \( V(b, i) \) by \( \Pr(x_{i+1} = a | x_i = b) \), the transition probability from \( b \) to \( a \), and \( \Pr(y_{i+1} | x_{i+1} = a) \), the probability of emitting \( y_{i+1} \) given that the \((i + 1)\)st state is \( a \). After obtaining the probability of the path leading to \( a \) from each \( b \), the probability of the best path, then, is the maximum over the set of all probabilities. Note that \( b \) and \( a \) can be the same state, and if they are the same, the state stays the same between the \( i \)th state and the \((i + 1)\)st state.

For the initial state \( (i = 1) \), since there is no previous state, the Viterbi algorithm simplifies to

\[ V(a, 1) = \Pr(x_1 = a) \cdot \Pr(y_1 | x_1 = a) \]

To go back to our example, first notice that there are 2 possible values that each state can take, namely \( H \) and \( S \). Therefore, \( W = \{H, S\} \). Suppose that the physician observes a patient for three consecutive days. The patient reports that on the first day he feels normal,
on the second day he feels dizzy, and on the third day he feels dizzy again. This gives us the set of all emissions: $Y = \{ y_1, y_2, y_3 \}$ where $y_1 = \text{normal}$, $y_2 = \text{dizzy}$, and $y_3 = \text{dizzy}$. We want to know the most likely sequence of the patient’s actual health on these three days. The math that follows looks complicated, but we will break down each step of the algorithm’s computational procedure. You should go through this worked example carefully before moving on.

First, we need to find $V(w, 1)$ for each $w \in W$. That is, we need $V(H, 1)$ and $V(S, 1)$:

$$V(H, 1) = \Pr(x_1 = H) \Pr(y_1 | x_1 = H)$$
$$= 0.6 \cdot 0.5$$
$$= 0.3,$$

$$V(S, 1) = \Pr(x_1 = S) \Pr(y_1 | x_1 = S)$$
$$= (1 - 0.6) \cdot 0.4$$
$$= 0.16.$$

Hence, just taking into account that the patient feels normal on the first day, the most likely state, i.e., the best path (recall the definition above) is $H$, that the patient is, in fact, healthy.

Now let’s find $V(H, 2)$:

$$V(H, 2) = \max \{ V(H, 1) \Pr(x_2 = H | x_1 = H) \Pr(y_2 | x_2 = H), V(S, 1) \Pr(x_2 = H | x_1 = S) \Pr(y_2 | x_2 = H) \}$$
$$= \max \{ 0.3 \cdot 0.7 \cdot 0.1, 0.16 \cdot 0.4 \cdot 0.1 \}$$
$$= 0.021;$$

this corresponds to the case where the patient is healthy on both the first and second days, i.e., the sequence of states $(H, H)$.

Doing the same to find $V(S, 2)$:

$$V(S, 2) = \max \{ V(H, 1) \Pr(x_2 = S | x_1 = H) \Pr(y_2 | x_2 = S), V(S, 1) \Pr(x_2 = S | x_1 = S) \Pr(y_2 | x_2 = S) \}$$
$$= \max \{ 0.3 \cdot 0.3 \cdot 0.3, 0.16 \cdot 0.6 \cdot 0.3 \}$$
$$= 0.0288;$$

this corresponds to the case where the patient is sick on both the first and second days, i.e., the sequence of states $(S, S)$. Since $0.0288 > 0.021$, this is the most likely sequence of states, i.e., the best path given the fact that the patient feels normal on the first day and dizzy on the second day.

Important note: the first state in the best path considering the first 2 days could be different from the first state in the best path considering only the first day, as in this case. Therefore, best paths are NOT necessarily successive.
So now that we have $V(S, 2)$ and $V(H, 2)$ we can go ahead and find $V(S, 3)$ and $V(H, 3)$:

$$
V(H, 3) = \max \{ V(H, 2)\Pr(x_3 = H|x_2 = H)\Pr(y_3|x_3 = H), \\
V(S, 2)\Pr(x_3 = H|x_2 = S)\Pr(y_3|x_3 = H) \} \\
= \max \{ 0.021 \cdot 0.7 \cdot 0.1, 0.0288 \cdot 0.4 \cdot 0.1 \} \\
= 0.00147,
$$
corresponding to the sequence of states $(H, H, H)$.

$$
V(S, 3) = \max \{ V(H, 2)\Pr(x_3 = S|x_2 = H)\Pr(y_3|x_3 = S), \\
V(S, 2)\Pr(x_3 = S|x_2 = S)\Pr(y_3|x_3 = S) \} \\
= \max \{ 0.021 \cdot 0.3 \cdot 0.3, 0.0288 \cdot 0.6 \cdot 0.3 \} \\
= 0.005184,
$$
corresponding to the sequence of states $(S, S, S)$.

Since $V(S, 3) > V(H, 3)$, we deduce that the patient is most likely sick on the third day. Therefore, the Viterbi best path for our example is $(S, S, S)$.

With the Viterbi algorithm, we can obtain the best path and its probability for any number of states. Of course, this algorithm can be generalized to cases where each state can take on more than 2 values (the set $W$ will thus have more than 2 elements).

2 Assignment

It is winter break in Las Vegas, and you have decided to spend the night at Anna’s casino. A disgruntled ex-employee of the casino told you that it uses two kinds of coins in its games: fair coins and biased coins with probability 0.65 of producing a head, i.e., $P(\text{head}|\text{biased}) = 0.65$.

2.1 Part 1: Warm up!

1. You are playing a game at Anna’s Casino where you guess how many heads there will be after tossing a coin 12 times. The casino promises that it will use a fair coin. Suppose you noticed that the casino did not switch coins, and you observed the following sequence of heads and tails:

```
H H T T H T H T H T T T
```

Based on this result, if you were to make your best guess, do you think the casino cheated, i.e., actually used a biased coin? What if you observed the following sequence instead:
Now, do you think the casino cheated?

2. Prove by induction that the Viterbi algorithm described above gives the best path of a Hidden Markov Model with \( n \geq 1 \) states and \( m \geq 1 \) emissions. (Note: your proof must work in general for any Hidden Markov Model, not just for the examples here, nor just for head and tail sequences. Induction should be done on \( m \).)

### 2.2 Part 2: Game time!

1. Now suppose you noticed that during another round of the same game, the casino switched coins, and you want to guess which coin the casino used for each toss. Assume that after each toss, the probability of switching coins for the next toss is 0.45. Draw a state transition diagram that shows all the possible states in this model and the probability of changing from one state to another. Add to your state transition diagram the possible emissions of each state and their associated emission probabilities. Use notations similar to Figure 1 (i.e., use circles to denote the states and arrows labeled with corresponding probabilities to show how the states could change from one to another with what probability, and use circles as emissions and arrows directed from states to emissions labeled with the corresponding probability).

Comment: This problem is designed to make sure you understand the model before proceeding to using it to calculate probabilities.

Note: For the following problems, assume that it is equally probable for the casino to use a fair or a biased coin for the 1st coin toss. For problems 2 and 3, when we say "calculate by hand the probability given by the Viterbi best path that the casino used a fair coin for the \( k \)th toss", please ignore the results of tosses after the \( k \)th toss and don’t count them as useful information. In other words, do these problems using the Viterbi algorithm as instructed in the Background Information, and take into consideration only the best path up until the \( k \)th toss.

2. You observed the following sequence of heads and tails in one game:

\[
H H T T H T
\]

Using the Viterbi algorithm described above, calculate by hand the probability given by the Viterbi best path that the casino used a fair coin for the 5th coin toss.

3. Write a Scheme procedure called `viterbiProb` that takes the following parameters and returns the probability given by the Viterbi best path that the casino used a fair coin for the \( k \)th coin toss.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type and Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>emission</td>
<td>a list of 0's and 1's representing head/tail emissions (where 0 represents head and 1 represents tail, e.g., '(1 0 1 0))</td>
</tr>
<tr>
<td>k</td>
<td>an integer representing the state of interest, e.g., 2</td>
</tr>
<tr>
<td>switchPr</td>
<td>a number representing the switching probability, e.g., 0.45</td>
</tr>
<tr>
<td>p1</td>
<td>a number representing the probability of generating a head when the state takes one value, e.g., 0.5 (Remember that a state can take any value in the set W)</td>
</tr>
<tr>
<td>p2</td>
<td>a number representing the probability of generating a head when the state takes another value, e.g., 0.65</td>
</tr>
</tbody>
</table>

The header of the procedure has been given to you:

```
(define viterbiProb
  (lambda (emission k switchPr p1 p2)
    ...))
```

Use your answer to Problem 2 as a test case for your procedure. Here is another test case:

```
(viterbiProb '(1 0 1 0) 2 0.45 0.5 0.65)
=> 0.06875
```

**Hint:** Problem 3 is closely related to Problem 5 below. You may want to consider both problems before you start solving either.

4. You observed the following sequence of heads and tails in one game:

```
T H H H T
```

Using the Viterbi algorithm described above, calculate by hand the best path and the probability associated with it.

5. Write a Scheme procedure called `viterbiPath` that takes the following parameters and returns a list of 2 elements, where the first element is the best path represented as a list of symbols `Fair` and `Biased`, where `Fair` represents a fair coin and `Biased` represents a biased coin, and the second element is the probability of this path.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type and Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>emission</td>
<td>same as in viterbiProb</td>
</tr>
<tr>
<td>switchPr</td>
<td>same as in viterbiProb</td>
</tr>
<tr>
<td>s1</td>
<td>a symbol representing one possible value that can be taken by a state e.g., 'Fair'</td>
</tr>
<tr>
<td>s2</td>
<td>a symbol representing another possible value that can be taken by a state e.g., 'Biased'</td>
</tr>
<tr>
<td>p1</td>
<td>a number representing the probability of generating a head when the state takes the value s1 e.g., 0.5</td>
</tr>
<tr>
<td>p2</td>
<td>a number representing the probability of generating a head when the state takes the value s2 e.g., 0.65</td>
</tr>
</tbody>
</table>

The header of the procedure has been given to you:

```lisp
(define viterbiPath
  (lambda (emission switchPr s1 s2 p1 p2)
    ...))
```

Use your answer of Problem 4 as a test case for your procedure. Here are some other test cases:

```lisp
(viterbiPath '(1 0 1) 0.45 'Fair 'Biased 0.5 0.55)
=> ((Fair Fair Fair) 0.018906250000000003)
```

```lisp
(viterbiPath '(1 1 0 0 1) 0.45 'Fair 'Biased 0.5 0.65)
=> ((Fair Fair Biased Biased Fair) 0.0016175478515625004)
```

**Hints:**

(a) A good way to make sure that you understand the problem is to think about what your guess should be if the switching probability (switchPr) is 0 or 1. In these cases, what can the actual sequence of coins be? What should your guess be when you only know the results of the first 1 or 2 tosses, and how should it change after you have information about more and more tosses? After you think it through, you can also use these scenarios as test cases for your procedure and see if your procedure produces the expected output.

(b) In writing your procedure, it might help to consider the following two sub-problems first:

i. What is the best path assuming that the last coin the casino used was a fair coin?

ii. What is the best path assuming that the last coin the casino used was a biased coin?

Incorporating your answers from i and ii, what is the overall best path?
2.3 Part 3: Probability

Below are some probability problems. Choose any five of them and turn in the answers. For the ones you do not do, study those as practice for the final.

1. A North Carolina licence plate consists of a sequence of seven symbols: number, letter, letter, letter, number, number, number, where a letter is any one of 26 letters and a number is one among 0, 1, ..., 9. Assume that all license plates are equally likely. (a) What is the probability that all symbols are different? (b) What is the probability that all symbols are different and the first number is the largest among the numbers?

2. A tennis tournament has $2n$ participants, $n$ Swedes and $n$ Norwegians. First, $n$ people are chosen at random from the $2n$ (with no regard to nationality) and then paired randomly with the other $n$ people. Each pair proceeds to play one match. An outcome is a set of $n$ (ordered) pairs, giving the winner and the loser in each of the $n$ matches. (a) Determine the number of outcomes. (b) What do you need to assume to conclude that all outcomes are equally likely? (c) Under this assumption, compute the probability that all Swedes are the winners.

3. A group of Scandinavians consists of 5 Swedes, 6 Finns, and 7 Norwegians. They are seated at random around a table. Compute the following probabilities: (a) that all Swedes sit together, (b) that all Swedes and all the Finns sit together, and (c) that all the Norwegians, all the Swedes, and all the Finns sit together.

4. Choose each digit of a 5 digit number at random from digits 1, ..., 9. Compute the probability that no digit appears more than twice.

5. Roll a fair die 10 times. (a) Compute the probability that at least one number occurs exactly 6 times. (b) Compute the probability that at least one number occurs exactly once.

6. A chocolate egg either contains a toy or is empty. Assume that each egg contains a toy with probability $p$, independently of other eggs. You have 5 eggs; open the first one and see if it has a toy inside, then do the same for the second one, etc. Let $E_1$ be the event that you get at least 4 toys and let $E_2$ be the event that you get at least two toys in succession. Compute $P(E_1)$ and $P(E_2)$. Are $E_1$ and $E_2$ independent?

7. You have 16 balls, 3 blue, 4 green, and 9 red. You also have 3 urns. For each of the 16 balls, you select an urn at random and put the ball into it. (Urns are large enough to accommodate any number of balls.) (a) What is the probability that no urn is empty? (b) What is the probability that each urn contains 3 red balls? (c) What is the probability that each urn contains all three colors?

8. Assume that you have an $n$ element set $U$ and that you select $r$ independent random subsets $A_1, \ldots, A_r \subseteq U$. All $A_i$ are chosen so that all $2^n$ choices are equally likely. Compute (in a simple closed form) the probability that the $A_i$ are pairwise disjoint.
2.4 Extra Credit

Here is a frequent job interview question in data science and Computer Science in Silicon Valley.

Q: What is the difference between probability and statistics?

A: Probability is (gerund<sub>1</sub>) the (noun<sub>1</sub>) from the (noun<sub>2</sub>), whereas Statistics is (gerund<sub>2</sub>) the (noun<sub>3</sub>) from the (noun<sub>4</sub>).

Fill in the blanks to give the best answer. Each blank contains exactly one English word.

Hint: Nouns can be singular or plural.

Hint: I like to make math jokes, but this is not a joke. It is a serious question and will be graded as such.

(gerund<sub>1</sub>) = ____________________________

(noun<sub>1</sub>) = ____________________________

(noun<sub>2</sub>) = ____________________________

(gerund<sub>2</sub>) = ____________________________

(noun<sub>3</sub>) = ____________________________

(noun<sub>4</sub>) = ____________________________