Machine Learning - Project

Adventures of Sherlock Holmes in Royale Casino

The goal of the proposed project is to understand the use of Viterbi's algorithm, when a system is modelled as a hidden Markov process. An overview of the problem is initially given. This is followed by its mathematical formulation. The proposed solution via Viterbi's algorithm is also outlined. Finally, the input string of data is given. Fortunately, the expected output is also provided. We use $P(\cdot)$ to denote the probability function. Also, $\mathbb{P} = \{1, 2, 3, \ldots\}$ is the set of positive integers.

1. Overview

You play the role of Sherlock Holmes in solving the mystery of fair and loaded die in Royale Casino.

In a casino, they use a fair die most of the time, but occasionally they switch to a loaded die. The casino switches from a fair to a loaded die with probability 0.05 before each roll, and that the probability of switching back is 0.1. The switch between the die is a Markov process. In each state of the Markov process the outcomes of a roll have different probabilities, and thus the whole process is an example of a hidden Markov model (HMM).

What is hidden in the model? You can just see a sequence of rolls (the sequence of observations) you do not know which rolls used a loaded die and which used a fair one, because that is kept secret by the casino; that is the state sequence is hidden. In a Markov chain you always know exactly in which state a given observation belongs. Obviously the casino wouldn't tell you that they use loaded dice and what the probabilities are. It is possible to estimate the probabilities in the above HMM (once you have a suspicion that they use two different dice).

The Model

A casino uses a fair die most of the time, but occasionally switches to a loaded one.

Let $P(k)$ = probability that a k shows up when a dice is rolled, where $k = 1, 2, 3, 4, 5,$ and 6.

- Fair die: $P(1) = P(2) = P(3) = P(4) = P(5) = P(6) = 1/6$.
- Loaded die: $P(1) = P(2) = P(3) = P(4) = P(5) = 1/10; P(6) = 1/2$.

These are the *emission probabilities* at the two states: fair (F), and loaded (L). Transitions between states obey a Markov process. The states of the discrete time Markov process are: fair and loaded. The *transition probabilities* are:

- $P(\text{Fair} \rightarrow \text{Loaded}) = 0.05$.
- $P(\text{Loaded} \rightarrow \text{fair}) = 0.1$.

The state transition diagram of the Markov process is shown below in Figure 1.

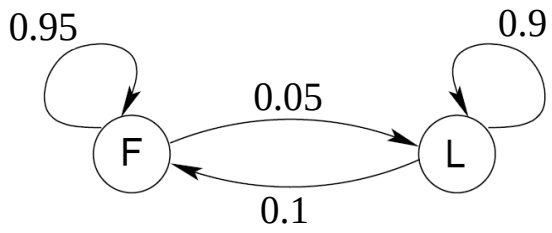

Figure 1. State transition diagram of the Markov process.

Statement of the Problem for Sherlock Holmes

The casino won't tell you when they use the fair or loaded die.

- *Known*:

  The structure of the model

  The transition probabilities

- *Hidden*: What the casino did

  FFFFFLLLLLLLFFFF...

- *Observable*: The series of die tosses

  3415256664666153...

- *What Holmes must infer*:

  When the fair die was used?

  When the loaded die was used?

- A possible answer is a sequence of states FFFFFFFLLLLLLFFF...

2. Mathematical Formulation

The problem is stated in terms of the transition matrix of the underlying Markov process, the emission probabilities, and the initial distribution.

- The underlying Markov process is specified by the transition matrix $A = [a_{ij}]$, where a_{ij} is the transition probability from state i to state j. Denote the 'fair' (F) state by 1, and the 'loaded' (L) state by 2. Thus the state transition matrix is

$$A = \begin{bmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{bmatrix}$$

  Note that the sum of elements in each row is equal to unity.

- The emission matrix $B = [b_{jk}]$ specifies the observable probabilities in each state. The states are $j = 1, 2$; where 1 denotes a fair die, and 2 denotes a loaded die. The observable symbols of the HMM are $k = 1, 2, 3, 4, 5$, and 6. We also use the notation $b_{jk} \triangleq b_j(k)$. For example, $b_2(4)$ is the probability of getting a 4 on a loaded die (which is eqaul to 1/10). Thus

$$B = \begin{bmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/2 \end{bmatrix}$$

  Note that the sum of elements in each row is equal to unity.

- The initial state description of the DTMC is specified by $\Pi = \{\pi_1, \pi_2\}$, where, $\pi_1 = \pi_2 = 1/2$. This means that, initially at time $t = 1$ the die is either fair or loaded with equal probability.

3. Viterbi's Algorithm

For a given observation sequence o, the corresponding state sequence q, has to be determined which is optimal in some predetermined sense. That is, this problem requires the determination of the hidden-state sequence that will most likely produce the specified observation sequence. Let the length of each of the hidden-state and output sequences be T. Note that

$$o = (o_1, o_2, \ldots, o_T), \quad \text{and} \quad q = (q_1, q_2, \ldots, q_T)$$

Also let Q be the random hidden-state sequence vector and O be the corresponding random output sequence vector. Each of these random vectors is of length T. A popular criterion is to find a hidden-state sequence (path) which maximizes the probability $P(Q = q \mid O = o)$. This is equivalent to maximizing $P(Q = q, O = o)$. A technique to finding such sequence uses the methods of dynamic programming. The actual technique is called the Viterbi algorithm, named after its inventor, Andrew J. Viterbi.

Let S be the set of states, where $|S| = N$. That is, the total number of possible states is equal to N. Define $\delta_t(i)$ as the probability of the highest probability sequence (path) at time $t \in \mathbb{P}$, which accounts for the first t output symbols, and which ends in state $s_i \in S$. Thus

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_{t-1}, q_t = s_i; o_1, o_2, \ldots, o_t)$$

Use of induction yields

$$\delta_{t+1}(j) = \left\{ \max_i \delta_t(i)\, a_{ij} \right\} b_j(o_{t+1})$$

The state sequence can be kept track of, via an array $\psi_t(j)$. That is, $\psi_t(j)$ keeps track of the state that maximizes $\delta_{t-1}(i)\, a_{ij}$ over i, which is the optimal previous state. Thus $\delta_{t+1}(i)$ can be computed recursively, and the optimal path can be determined via backtracking from time T. The Viterbi algorithm is as follows.

Step 1: Initialization:

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(o_1), \quad 1 \le i \le N \\ \psi_1(i) &= 0, \quad 1 \le i \le N \end{aligned}$$

Step 2: Recursion:

$$\begin{aligned} \delta_t(i) &= \max_{1 \le k \le N} \{\delta_{t-1}(k)\, a_{ki}\}\, b_i(o_t), \quad 2 \le t \le T, \quad 1 \le i \le N \\ \psi_t(i) &= \arg \max_{1 \le k \le N} \{\delta_{t-1}(k)\, a_{ki}\}, \quad 2 \le t \le T, \quad 1 \le i \le N \end{aligned}$$

Step 3: Termination:

$$\begin{aligned} P^* &= \max_{1 \le k \le N} \delta_T(k) \\ q_T^* &= \arg \max_{1 \le k \le N} \delta_T(k) \end{aligned}$$

Step 4: Optimal state sequence (path) backtracking:

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right), \quad t = (T-1), (T-2), \ldots, 1$$

□

Observe that, the above algorithm involves multiplication of probabilities (which are numbers between zero and unity). This might result in underflow in the computations. The above Viterbi algorithm can be modified by taking logarithms. This eliminates the need for multiplications.

Alternative Viterbi's Algorithm

Step 0: Preprocessing:

$$\begin{aligned} \widehat{\pi}_i &= \log(\pi_i), \quad 1 \le i \le N \\ \widehat{b}_i(o_t) &= \log\{b_i(o_t)\}, \quad 1 \le i \le N, 1 \le t \le T \\ \widehat{a}_{ij} &= \log(a_{ij}), \quad 1 \le i, j \le N \end{aligned}$$

Step 1: Initialization:

$$\begin{aligned} \widehat{\delta}_1(i) &= \log\{\delta_1(i)\} = \widehat{\pi}_i + \widehat{b}_i(o_1), \quad 1 \le i \le N \\ \psi_1(i) &= 0, \quad 1 \le i \le N \end{aligned}$$

Step 2: Recursion:

$$\begin{aligned} \widehat{\delta}_t(i) &= \log(\delta_t(i)) = \max_{1 \le k \le N} \left\{ \widehat{\delta}_{t-1}(k) + \widehat{a}_{ki} \right\} + \widehat{b}_i(o_t), \quad 2 \le t \le T, \quad 1 \le i \le N \\ \psi_t(i) &= \arg \max_{1 \le k \le N} \left\{ \widehat{\delta}_{t-1}(k) + \widehat{a}_{ki} \right\}, \quad 2 \le t \le T, \quad 1 \le i \le N \end{aligned}$$

Step 3: Termination:

$$\begin{aligned} P^* &= \max_{1 \le k \le N} \widehat{\delta}_T(k) \\ q_T^* &= \arg \max_{1 \le k \le N} \widehat{\delta}_T(k) \end{aligned}$$

Step 4: Optimal state sequence (path) backtracking:

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right), \quad t = (T-1), (T-2), \ldots, 1$$

□

4. Input and Expected Output

The input data and the expected output is shown in Figure 2.

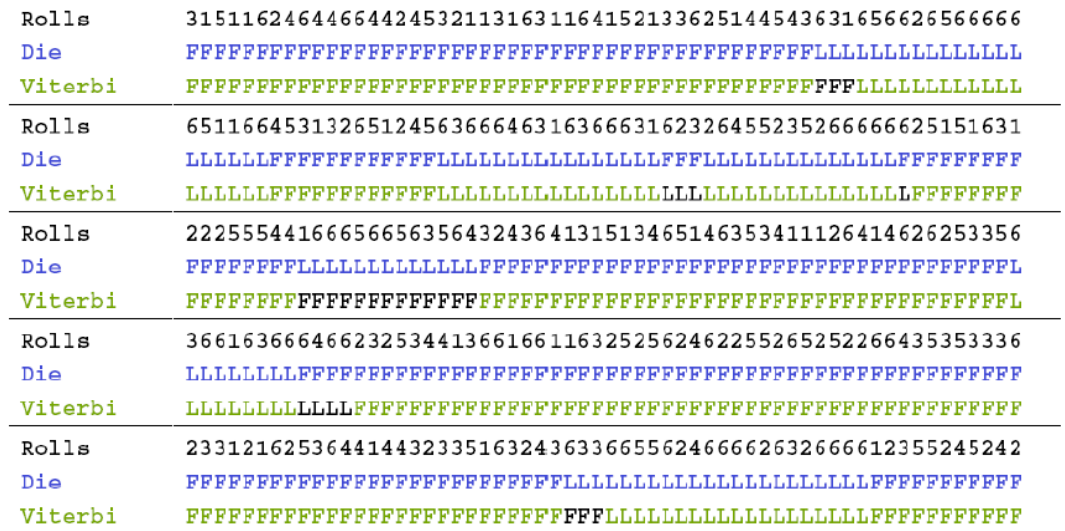

Figure 2. Input strings and the corresponding output strings.

Notice that there are five sets of data. Each data-set has

- Rolls: This is the string of rolls. It is the input to your computer program.
- Die: This is the string of hidden states. The states are: F (Fair) and L (Loaded).
- Viterbi: This is the output string produced by the Viterbi's algorithm.

Project Report

You can use any language to implement the algorithm. You should submit the following items.

1. Printed code.
2. For each of the five *Rolls* strings, print:
   (a) The given string of *Rolls*.
   (b) The string of *hidden states* produced by your Viterbi-code. It should match the given string *Viterbi*.

Hopefully, your output matches with the strings shown in Figure 2.

5. References

1. Durbin, R., Eddy, S., Krogh, A., and Mitchison, G., 1998. *Biological Sequence Analysis - Probabilistic models of proteins and nucleic acids*, Cambridge University Press, Cambridge, New York, N.Y..