Advanced AI Assignment-1 HMM

Team : Susim Mukul Roy(B20AI043), Jaisidh Singh(B20AI015)

The assignment has been coded in Python. All the given assumptions along with the observation tuples and states have been integrated in the code as the transition, emission and the Pi matrices.

In this problem there are 5 states, namely Planning, Scouting, Burglary, Migrating and Miscellaneous. The observations here are presented as a tuple of Daytime and Action. The daytime can be Day, Evening or Night and the Action can be Roaming, Eating, Home or Untracked which makes a total of 12 possible observations/tuples.

Model Setup: -

The HMM() class has three functions A, B and Pi. A takes two arguments, s^i and s^j, and returns the probability of the transition from $s^i \to s^j$. B takes two arguments, s^i and o_i, and returns the probability of emitting o_i from state s^i. Pi takes one argument, s^i, and returns the probability of the HMM starting in that state. The transition matrix is a 5x5 matrix which is a sparse matrix with non-zero values at those (i, j) whose transition is mentioned in the assumptions listed in the question. The emission matrix is a 4x5x3 matrix whose initial values are given as per the assumptions. Higher values of probabilities have been given to those cells which are mentioned as *most* or *preferably* in the question and the other non-zero probabilities are equally divided.

Part I :-

We use the forward backward algorithm to train our HMM model λ, that is to estimate the transitions and emissions from the observations. To do this, we utilize a backward probability array defined as $\beta_t(j)$ given by:

$$\beta_t(j) = P(o_{t+1}, o_{t+2}, ...o_T | q_t = i, \lambda)$$

Here, $q_t = i$ means the state at time step t in the sequence of states is state i.It is computed inductively in a dynamic programming fashion as show:

Initialization:

$$\beta_T(i) = 1, \quad i \in \{1, ..., N\}$$

Recursion:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \qquad i \in \{1, ..., N\}, t \in \{1, ..., T\}$$

Termination:

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j b_j(o+1) \beta_1(j)$$

Next, we want to estimate the new transition matrix. We do this by defining $\xi_t(i, j)$ as:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)}$$

The new transition matrix $\hat{a}_{ij}$ is then found as:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i, k)}$$

We also require a new emission matrix for which we define $\gamma_t(j) = P(q_t = j|O, \lambda)$

$$\gamma_t(j) = \frac{\alpha_t(j) \beta_t(j)}{P(O|\lambda)}$$

where $P(O|\lambda) = \sum_{j=1}^{N} \alpha_t(j) \beta_t(j)$. Finally, we estimate the new emission matrix $\hat{b}_j(v_k)$ as:

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 \ s.t. \ O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

The $LearnModel()$ function in the code calculates the new estimates and returns the HMM model.

Part II : -

We are given a list of observations and our task is to find the likelihood of a sequence of observations given our model parameters, i.e., the emission and transition matrices, A and B respectively and the HMM model λ. In order to solve this, we apply a dynamic programming approach where we define $\alpha_t(i)$ as:

$$\alpha_t(i) = P(o_1, o_2, ..., o_t, q_t = j|\lambda)$$

Here, $q_t = j$ means the state at time step t in the sequence of states is state j. The algorithm implemented is called the forward algorithm which is as follows:

- Initialization

$$\alpha_1(j) = \pi_j b_j(o_1) \quad 1 \leq j \leq N$$

- Recursion

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \leq j \leq N, \ \ 1 < t \leq T$$

- Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The $Liklihood()$ function in the code returns the probability of the sequence of observations.

Part III : -

We are given a list of observation $(o_1, o_2, ..., o_t)$ along with the transition and emission probabilities. Our task is to find the most probable sequence of states which produces these observations. In order to solve this too, we apply a dynamic programming approach where we define $v_t(j)$ as:

$$v_t(i) = \max_{q_1,..,q_{t-1}} P(q_1, ..., q_{t-1}, o_1, o_2, ..., o_t, q_t = j|\lambda)$$

The implemented algorithm is known as the Viterbi algorithm which uses the concept of *back-pointers* to store the best state sequence by keeping track of the path of hidden states that led to each state and then at the end backtracking the best path to the beginning. This is also known as the Viterbi back-trace. The implemented algorithm is as follows:

- Initialization

$$v_1(j) = \pi_j b_j(o_1) \ \ 1 \le j \le N$$
$$bt_1(j) = 0 \ \ 1 \le j \le N$$

- Recursion

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \ \ 1 \le j \le N, \ \ 1 < t \le T$$
$$bt_t(j) = \arg\max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \ \ 1 \le j \le N, \ \ 1 < t \le T$$

- Termination

$$P = \max_{i=1}^{N} v_T(i)$$
$$q_T = \arg\max_{i=1}^{N} v_T(i)$$

The $GetHiddenStates()$ function returns the list of state sequences in the Enum format as required.