Hidden Markov Models

MACHINE INTELLIGENCE LABORATORY

Teaching Assistants
Sarasharish
Abaksy
Vighnesh kamath

In this week's experiment, you will implement the Viterbi algorithm for decoding a sequence of observations to find the most probable sequence of internal states that generated the observations.
Remember that a Hidden Markov Model (HMM) is characterized by the set of internal states, the set of observations, the transition matrix (between the internal states), the emission matrix (between the internal states and the emitted observations) and the initial distribution of the internal states.

You are provided with the following files:

1. **week8.py**
2. **SampleTest.py**

Note: These sample test cases are just for your reference.

Important Points:

1. Please do not make changes to the function definitions that are provided to you, as well as the functions that have **already been implemented**. Use the skeleton as it has been given. Also do not make changes to the sample test file provided to you.
2. You are free to write any helper functions that can be called in any of these predefined functions given to you. Helper functions must be only in the file named 'YOUR_SRN.py'.
3. Your code will be auto evaluated by our testing script and our dataset and test cases will not be revealed. Please ensure you take care of all edge cases!
4. **The experiment is subject to zero tolerance for plagiarism. Your code will be tested for plagiarism against every code of all the sections and if found plagiarized both the receiver and provider will get zero marks without any scope for explanation.**
5. **Kindly do not change variable names or use any other techniques to escape from plagiarism, as the plagiarism checker is able to catch such plagiarism**
6. Hidden test cases will not be revealed post evaluation.

week8.py

- ✓ You are provided with the structure of class HMM.
- ✓ The class Tensor contains one constructor and two methods, one of which is already implemented
- ✓ Your task is to write code for the viterbi_algorithm() method

- The **make_states_dict()** method is already implemented. It creates two dictionaries, one of them being a mapping between state names and indices, and the other being a mapping between the possible observations and the indices (e.g.: Suppose the HMM has internal states as 'Sunny and 'Rainy', and the possible observations are 'Happy' and 'Grumpy', then the make_states_dict() function generates the following dicts: {'Sunny': 0, 'Rainy': 1} and {'Happy': 0, 'Grumpy': 1})

- The **viterbi_algorithm(obs)** method is to be implemented by you. The transition matrix A, the emission matrix B and the initial distribution pi are class variables. Your task is to implement the Viterbi Algorithm and return the sequence of hidden states that is most likely to result in the given observation sequence. The sequence of hidden states is to be returned in the form of a Python List.
  Note: The Viterbi Algorithm is a dynamic programming problem who's solution is given below, and can be found in the course slides as well

Initialization:

$$\nu_1(j) = \pi_j b_j(o_1)$$
$$bt_1(j) = 0$$

Recursion:

$$\nu_t(j) = \max_{i=1}^{N} \nu_{t-1}(i)\ a_{ij}\ b_j(o_t)$$
$$bt_1(j) = \operatorname{argmax}_{i=1}^{N} \nu_{t-1}(i)\ a_{ij}\ b_j(o_t)$$

Termination:

$$\text{Best Score: } P* = \max_{i=1}^{N} \nu_T(i)$$
$$\text{Best Path Start: } q_T* = \operatorname{argmax}_{i=1}^{N} \nu_T(i)$$

Viterbi Algorithm

1. **You may write your own helper functions <u>if needed</u>**
2. **You can import libraries that come built-in with python 3.7**
3. **You cannot change the skeleton of the code**
4.

<u>SampleTest.py</u>

1. This will help you check your code.
2. Passing the cases in this does not ensure full marks, you will need to take care of edge cases
3. Name your code file as YOUR_SRN.py
4. Run the command

   python3 SampleTest.py --SRN YOUR_SRN

   if import error occurs due to any libraries that is mentioned in the skeleton code try:

   python3.7 SampleTest.py --SRN YOUR_SRN