

Programming Assignment 7 Report

PART 1: IMPLEMENTATION

Hidden Markov Models:

Data Structures:

- **Lists:** We have used lists in order to store the grid-world, tower locations and noisy data in separate lists in the order they are given.
- **Numpy arrays:** I have used numpy arrays throughout the program for manipulating the input and calculating the transition probability matrix and emission matrix. I have also used numpy zeros and ones to initialize various variables for using in the calculations.
- **Dictionary:** I have used dictionaries in the program to compute and store different variables related to transition matrix and for storing paths and the point from which that particular path was possible.

Code-level optimizations:

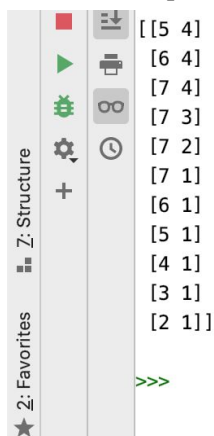
- We used numpy arrays along with various numpy functions along with enumerate function in calculating distances, finding empty cells, finding maximum probabilities and initializing variables so as to get quicker and more accurate results in an iterative fashion.

Challenges:

- While we were writing a program, there was a lot of confusion in using the datatypes as we had to convert the coordinates into a 2-D array to find the position each time. It took efforts to debug the program too as there were several possibilities while calculating the path in a 10 * 10 grid area and performing manual calculations to check any doubtful move seemed difficult.

Output:

- Our final computed path is:



```
[5 4]
[6 4]
[7 4]
[7 3]
[7 2]
[7 1]
[6 1]
[5 1]
[4 1]
[3 1]
[2 1]]
```

The screenshot shows a Jupyter Notebook interface. On the left, there is a sidebar with '2: Favorites' and '2: Structure'. The main area displays a list of coordinates: [5 4], [6 4], [7 4], [7 3], [7 2], [7 1], [6 1], [5 1], [4 1], [3 1], and [2 1]. Below the list, there are three green arrows pointing to the right.

PART 2: SOFTWARE FAMILIARIZATION

Hidden Markov Models

Libraries

- **hmmlearn:** We found a library called hmmlearn to implement the Hidden Markov Model with discrete observation data: `hmmlearn.hmm.MultinomialHMM(n_components)`. `n_components` is the number of states in each time stamp. To configure the model, set the model variables:
 - `model.startprob_` = numpy matrix with dimensions [number of states x 1] → initial start probabilities

- `model.transmat_` = numpy matrix with dimensions [number of states x number of states] → transition probabilities
- `model.emissionprob_` = numpy matrix with dimensions [number of states x number of states] → emission probabilities
- To find the most likely sequence of states corresponding to `X` from the model with the Viterbi algorithm: `model.decode(X, algorithm='viterbi')`
- Example:

```
model = hmm.MultinomialHMM(n_components=2)
model.startprob_ = np.array([0.8, 0.2])
model.transmat_ = np.array([[0.9, 0.1], [0.4, 0.6]])
model.emissionprob_ = np.array([[0.9, 0.1], [0.5, 0.5]])
X = np.atleast_2d([3, 4]).T
print(model.decode(X, algorithm='viterbi'))
```

Implementation Comparison

- We had some issues when trying to implement the `hmmlearn` library. Even when researching into other existing HMM libraries, they all required multiple parameters that we needed to figure out to pass into the arguments.
 - To create the HMM model, we need the start probability, transition probability, and emission probability. However, that information is not readily available through any libraries. Finding the transition probability requires iterating through the grid and checking the valid moves. In order to calculate the emission probabilities, we have to consider the noisy distances and towers as well as maximum probabilities.
 - We would need to manually calculate the transition and emission probabilities to pass it into the `hmmlearn MultinomialHMM` model parameters.
- The `HMMlearn` library has a base class called `hmmlearn.base` for evaluating, sampling, and maximizing the posteriori estimation of a HMM with different types of emission data, but our program only deals with a single type of data.

Possible Improvements

- The `hmmlearn` library can estimate the optimal sequence of hidden sequences given the model parameters and observed data. This is what our program achieves by outputting the robot's most likely trajectory for 11 time steps. However, the library can also calculate the likelihood of the data from given model parameters and observed data. It can also estimate model parameters if given just the observed data. These are some improvements we can add to our program for more accuracy and details.

PART 3: APPLICATIONS

- **GeneMark:** Gene prediction program using a Hidden Markov Model. DNA sequences can be considered by using four alphabetic letters to represent the nucleotides. Genes can be found from the difference in stochastic properties of Markov chain models between coding and non-coding regions of DNA.
- **Human Identification using Gait:** Human Gait is often studied as a collection of gait cycles. The two components of a gait cycle are the structural, physical component and dynamic, body motion component. Gait cycle boundaries are identified for a walking sequence and partitioned in

6 groups of temporarily adjacent stances. The averages of all stances from a particular partition is computed so that system states are identified with the HMM.

- **Facial expression identification:** Facial Action Coding Systems (FACS) is the most comprehensive way to characterize facial expressions. It uses discrete deformations of face regions called Action Units and divides facial expressions into upper and lower regions. The hidden state of the HMM is the hidden emotional state of an individual. And the observable symbols of the HMM are the feature vectors from the FACS.

Group Members and Contributions

- Our group members for this programming assignment were Sudeeptha Mouni Ganji (ID: 2942771049) and Vanessa Tan (ID: 4233243951).
- Vanessa and Sudeeptha worked together to research and understand the algorithms.
- After implementation, Sudeeptha and Vanessa worked together to write the report.

Programming:

Algorithm Implementation : Sudeeptha Mouni Ganji, Vanessa Tan

Library Implementation: Sudeeptha Mouni Ganji, Vanessa Tan

Report Writing:

Part 1: <ul style="list-style-type: none">● Hidden Markov Models	Sudeeptha Mouni Ganji Vanessa Tan
Part 2: <ul style="list-style-type: none">● Hidden Markov Models	Vanessa Tan Sudeeptha Mouni Ganji
Part 3: <ul style="list-style-type: none">● Hidden Markov Models	Vanessa Tan Sudeeptha Mouni Ganji