Programming Assignment 7: Hidden Markov Models

Kavya Sethuram(ksetura@usc.edu); Rasika Guru (rguru@usc.edu); Roshani Mangalore (rmangalo@usc.edu)

1. Implementation of Hidden Markov Models in python v2.7:

A. DataStructures used in the Implementation are:

- ***Lists:*** Python has a great built-in list type named "list". List literals are written within square brackets [ ]
- ***Dictionary:*** Dictionaries in python are unordered data structures that are used to store unique keys and their corresponding values.

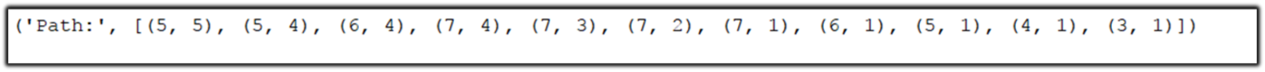
B. Explanation:

Hidden Markov Model Algorithm:

- Store the free cells, the 4 tower locations, the given distances to each the towers
- Calculate the Euclidean distance of the each of the tower from the free cells.
- Compute the possible cells which the towers could probably traverse: This is done by comparing the Euclidean distance calculated in the previous step with the given distance.
- The cells which have the distance lesser than the given distance are added to the list of possible cells in the path
- The transition path/trajectory is then calculated based on the Viterbi algorithm.
- Here the most likely sequence of the trajectory is calculated by keeping track of the arguments that maximize the probability for each tower and free cell, storing them in a matrix which is used to retrieve the optimal state sequence at the backtracking step.

Output of the HMM is as below:

The coordinates of the most likely trajectory of the robot for 11 time-steps is as follows:



```
('Path:', [(5, 5), (5, 4), (6, 4), (7, 4), (7, 3), (7, 2), (7, 1), (6, 1), (5, 1), (4, 1), (3, 1)])
```

C. Code Level Optimizations:

- Modularizing parts of code into methods, making code more readable.
- Used xrange instead of range for better performance because range() requires more memory than xrange()

D. Challenges:

- Computation of Probability Distribution table was tricky

2. Software Familiarization:

Existing Libraries:

Scikit-Learn

- It's a machine learning library. It includes various machine learning algorithms.
- It uses the following inbuilt library to implement Hidden Markov Model algorithm. Below is the scikit implement of this algorithm.

```
import numpy as np
from hmm import MultinomialHMM
hmm = MultinomialHMM(n_states=3, verbose=0, n_repeats=20)
X = ['a'] * 5 + ['b'] * 5 + ['c'] * 5 # build a simple sequence of characters
X = np.array(X)
hmm.fit(X) # learn the parameters of the model with EM
for i in range(3):
   print(''.join(hmm.generate(15))) # generate a sequence of 15 characters from the
model
```

- Scikit's HMMlearn implement HMM with emission probabilities determined by multimomial distributions, Gaussian distributions and mixtures of Gaussian distributions. Our program deals with the given matrix and one type of emission.
- We use the Viterbi algorithm whereas the HMMlearn offers three types of algorithm, namely – Viterbi, Forward-Backward algorithm and Baum-Welch algorithm.
- HMMLearn also has a base class called **hmmlearn.base** which allows for easy evaluation of, sampling from, and maximum a posteriori estimation of the parameters of a HMM. It allows us to deal with various other kinds of emission data unlike the single type of data that our algorithm deals with.
- HMMlearn allows to implement other emission probability (e.g. Poisson), by implementing a new HMM class by inheriting the **_BaseHMM** and overriding the methods __init__, _compute_log_likelihood, _set and _get for additional parameters.

3. Applications:

1. Protein- Profile HMMs

Protein structural similarities make it possible to create a statistical model of a protein family which is called a profile. The idea is, given a single amino acid target sequence of unknown structure, we want to infer the structure of the resulting protein. The profile HMM is built by analyzing the distribution of amino-acids in a training set of related proteins. This HMM in a natural way can model positional dependant gap penalties.

[Reference:
https://pdfs.semanticscholar.org/9d0e/c3f52be23cbff7b430726f606831d497c96c.pdf]

2. Creating Multiple sequence alignment:
HMMs can be used to automatically create a multiple alignment from a group of unaligned sequences. By taking a close look at the alignment, we can see the history of evolution. One great advantage of HMMs is that they can be estimated from sequences, without having to align the sequences first. The sequences used to estimate or train the model are called the training sequences, and any reserved sequences used to evaluate the model are called the test sequences. The model estimation is done with the forward-backward algorithm, also known as the Baum-Welch algorithm. It is an iterative algorithm that maximizes the likelihood of the training sequences.

[Reference:
https://pdfs.semanticscholar.org/9d0e/c3f52be23cbff7b430726f606831d497c96c.pdf]

3. **Prediction of protein secondary structure using HMM's**:

Prediction of secondary structures is need for the prediction of protein function. As an alternative method to direct X-ray analysis, a HMM is used to ß Analyze the amino-acid sequences of proteins ß Learn secondary structures such as helix, sheet and turn ß Predict the secondary structures of sequences The method is to train the four HMMs of secondary structure – helix, sheet, turn and other – by training sequences. The Baum-Welch method is used to train the HMMs. So, the HMM of helix is able to produce helix-like sequences with high probabilities. Now, these HMMs can be used to predict the secondary structure of the test sequence. The forward-backward algorithm is used to compute the probabilities of these HMMs outputting the test sequence. The sequence has the secondary structure whose HMM showed the highest probability to output the sequence.

[References:
https://pdfs.semanticscholar.org/9d0e/c3f52be23cbff7b430726f606831d497c96c.pdf]

4. Individual Contribution

Implementation of HMM	Kavya Sethuraman Rasika Guru Roshani Mangalore