INF552: Programming Assignment 7 [Hidden Markov Model]

Priyambada Jain (priyambj@usc.edu)
Sai Sree Kamineni (skaminen@usc.edu)
Varad Kulkarni (vdkulkar@usc.edu)

Part 1 Implementation

Command to run – python hmm.py

Input Files – hmm-data.txt

Data Structures Used - Python Dictionary, Math module

Output- The coordinates of the most likely trajectory of the robot for 11 time-stamps.

Code Level Optimizations- Enumerate function in python is used quite often in the program to retain the index as well as the values in the data structures while traversing them.

Algorithm-

1. First the possible states are spotted (All the cells with zero are omitted).
2. States whose distances from each of the tower destination lie in the range of 0.7*d to 1.3*d are considered as valid states.
3. After this step, the state transition matrix which consists of the probability by which the robot will go from one state to the other from the current to the next observation is calculated.
4. After the construction of the probability matrix, the Viterbi algorithm is called such that the variable reduction is executed, and the states for every observation are traced out as an output.

Part 2: Software Familiarization

For technical implementation of HMM we used hmmlearn which is a set of algorithms for **unsupervised** learning and inference of Hidden Markov Models in python.

Here is the working principle behind the library: The HMM is a generative probabilistic model, in which a sequence of observable **X**X variables is generated by a sequence of internal hidden states **Z**Z. The hidden states are not being observed directly. The transitions between hidden states are assumed to have the form of a (first-order) Markov chain. They can be specified by the start probability vector ***π***π and a transition probability matrix **A**A. The emission probability of an observable can be any distribution with parameters ***ϑ***θ conditioned on the current hidden state. The HMM is completely determined by ***π***π, **A**A and ***ϑ***θ.

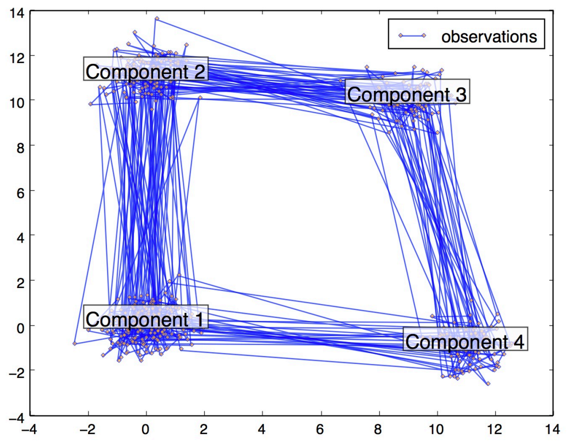
We wrote a script that shows how to sample points from a Hiden Markov Model (HMM):
For this we used a 4-components with specified mean and covariance.

model = hmm.GaussianHMM(n_components=4, covariance_type="full")

# Instead of fitting it from the data, we directly set the estimated
# parameters, the means and covariance of the components

model.startprob_ = startprob
model.transmat_ = transmat
model.means_ = means
model.covars_ = covars

The plot shows the sequence of observations generated with the transitions between them. We can see that, as specified by our transition matrix, there is no transition between component 1 and 3.



Part 3: Applications

Due to the powerfulness that Markov models provide, it can be used anywhere sequential information exists:

1. **Finance:** Model non-stationary and non-linearity of financial data to predict the direction of the time series.

2. **Biology:** DNA is composed if 4 bases (A, G, T, C) which pair together form nucleotides. Markov models can compute likelihoods of an DNA sequence.

3. **Tracking systems**: Markov models can be used to estimate the position in a tracking system

4. Speech processing: Speech recognition has been the most exploited area for use of Markov models.

5. **Image processing**: Human action recognition can be modeled with Markov models

   They perform very well for many applications if they are applied in the correct way:

   - Signal segmentation and classification
   - Clustering of signals
   - Prediction

Part 4: References

1. http://hmmlearn.readthedocs.io/en/latest/tutorial.html
2. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.378.1564&rep=rep1&type=pdf
3. http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf
4. https://en.wikipedia.org/wiki/Hidden_Markov_model#Applications
5. https://www.cs.umb.edu/~rvetro/vetroBioComp/HMM/Rabiner1986%20An%20Introduction%20to%20Hidden%20Markov%20Models.pdf