INF552: Programming Assignment 7 [HMM]

Student: Jiashi Chen USCID: 4684194123

Part 1: Implementation

Output:
The coordinates of the most likely trajectory of the robot is `(5,3), (6,3), (7,3), (8,3), (8,2), (7,2), (7,1), (6,1), (5,1), (4,1), (3,1)`

Data structure:
I use a list of tuple to store nodes like free nodes or tower locations like
$[(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)]$
I use a list of list to store all noisy distance like
$[[Tower1_{t1}, Tower2_{t1}, Tower3_{t1}, Tower4_{t1}] \dots, [Tower1_{tn}, Tower2_{tn}, Tower3_{tn}, Tower4_{tn}]]$
Use dictionary to store the possible free nodes each time-step. Key is the time and value is the list of all possible nodes. $\{t_0 : [x_1, x_2, \dots, x_n], \dots, t_{10} : [x_1, x_2, \dots, x_m]\}$
And finaly during Viterbi Algorithm, the data structure I use to record probabilty and best previous node is like this
$\{t_0 : \{possible\_node : \{prob : \delta, pre : \psi\}\}, \dots, t_{10} : \{possible\_node : \{prob : \delta, pre : \psi\}\}\}$

Any code-level **optimizations** I perform is that I just store the possible node in each time -step. Instead of put all of probabilities into matrix at beginning, I only calulate the probabilities of free node like transition probability and emission probability when I need it. It could save a lot of memory to store data and speed up the whole algorithm.

One of the **challenges** I face is that there are 87 states that lead to a large data set and difficult to set up the matrix clearly for all probabilities. Also I spend a lot of time in figuring out how to design a data structure than could record the max probability of each node in each-time and which the previous node it has. What's more, when trying to calculate the emission matrix, I have no idea how to compute it at the beginning.

Part 2: Software Familiarization

hmmlearn package offers a good implementation of the Hidden Markov Models.

How to use

I used this package to fit the data in ***Jiashi_Chen_Homework_7_Code.ipynb*** file too.

API for Hidden Markov Models and Viterbi algorithms(https://hmmlearn.readthedocs.io/en/latest/api.html)

```
from hmmlearn import hmm  ## Obtain necessary package

model = hmm.MultinomialHMM(n_components=n_states)  #Set the number of hidden states
model.startprob_ = np.array(start_probability)   #Initial state occupation distribution.
model.transmat_ = np.array(transition_probability) #Set matrix of transition probabilities between states.
model.emissionprob_ = np.array(emission_probability) # Set probability of emitting a given symbol when in each state.

seen_list = [i for i in range(11)]
seen = np.array([seen_list]).T  #The real observation.

box2 = model.predict(seen)  #Find most likely state sequence corresponding to seen.
```

And the output of this module is the same as mine.

Comparsion and Improvement

After wathcing the document of this module and some examples of it, the module is much thoughtful than what I code and I could add some of functionality into my work. The module not only could solve the discrete data, but also it has designed something like GaussianHMM to solve continuous data. Besides *GaussianHMM*, module also include *GMMHMM* and *MultinomialHMM*. About the **decoder algorithm**, we can choose *Viterbi* or *Map* to solve the problem, while my program only contanins *Viterbi*. I could try to use other algorithm to see the differnce between each other. Also there is an argument called **covariance_type** that describe the type of covariance parameters to use. We could choose *spherical*, *diag*, *full* and *tied*, which make the user deal with the data more accurately.

Part 3: Applications

GeneMark.hmm: new solutions for gene finding[1]

The number of completely sequenced bacterial genomes has been growing fast. There are computer methods available for finding genes but yet there is a need for more accurate algorithms. The GeneMark. hmm algorithm presented in the paper was designed to improve the gene prediction quality in terms of finding exact gene boundaries. The idea was to embed the GeneMark models into naturally derived hidden Markov model framework with gene boundaries modeled as transitions between hidden states. They also used the specially derived ribosome binding site pattern to refine predictions of translation initiation codons. The algorithm was evaluated on several test sets including 10 complete bacterial genomes. It was shown that the new algorithm is significantly more accurate than GeneMark in exact gene prediction. Interestingly, the high gene finding accuracy was observed even in the case when Markov models of order zero, one and two were used. We present the analysis of false positive and false negative predictions with the caution that these categories are not precisely defined if the public database annotation is used as a control.

HMM-based architecture for face identification[2]

This paper describes an approach to the problem of face identification which uses Hidden Markov Models (HMM) to represent the statistics of facial images. HMMs have previously been used with considerable success in speech recognition. Here they describe how two-dimensional face images can be converted into one-dimensional sequences to allow similar techniques to be applied. They investigate the factors that affect the choice of model type and model parameters. They show how a HMM can be used to automatically segment face images and extract features that can be used for identification. Successful results are obtained when facial expression, face details and lighting vary. Small head orientation changes are also tolerated. Experiments are described which assess the performance of the HMM-based approach and the results are compared with the well-known Eigenface method. For the given test set of 50 images, the HMM approach performs favourably. They conclude by summarizing the benefits of using HMMs in this area, and indicate future directions of work.

References

[1] Lukashin, A. GeneMark.hmm: new solutions for gene finding[J]. Nuclc Acids Research, 1998, 26(4):1107-1115.

[2] Ferdinando, Samaria. HMM-based architecture for face identification[J]. Image & Vision Computing, 1994.