Holly Do - COMP 614

Homework 3

The URL for your final CodeSkulptor file:

https://py3.codeskulptor.org/#user310_8ADUqc58oZ_7.py

Question 3.A:

Before you dive into the code, you must answer the following "pre-write" questions and include your answers in your writeup, labeled with the question number. **For these questions, you must refer to the length of the input list as m and the input number as n.** Note that these questions are similar to the moving averages questions from the previous assignment. :)

Question 3.A.i: In terms of m and/or n, how many pieces of data should comprise each state in the resultant Markov chain?

In terms of input length m and order n, each state should be comprised of n pieces of data.

Question 3.A.ii: In terms of m and/or n, what is the maximum possible number of states that you could end up with in your markov chain (*i.e.*, in the case where there are no repeats)? Why? Please be as clear and specific as possible in your explanation. You should only count states for which there is a "next" value in the input list.

In case m>n:
The number of states you end up with is m-n states. Because prediction is made one by one, the total valid states with length n is m-n+1. But, because we only count the states for which there is a next value in the input list, the last element, since there is no next state, is not included in the states' list. This the number of states is m-n states.
In case m=n:
A state is not found, then return a random number among the tuple (0, 1, 2, 3)
In case m<n:
A state is not found, then return a random number among the tuple (0, 1, 2, 3)

Question 3.A.iii: Without using iteration, how could you extract a single state of the correct length? Provide a Python expression for extracting the sequence of elements

that will be used to comprise the state starting at position i in the original list. Your expression should work for any arbitrary value of i that is greater than or equal to zero and less than your answer to Question 3.A.ii.

Instead of iteration we could use slicing list/set method. First, we need to make sure that the data is the list or set structure, then utilize the slicing method as follow:
data[i : (i+order)]
This works with any value of i that is greater than or equal to zero or less than m-n. Since m-n+n = m which is an edge case, meaning data[m] will be out of range. Thus, the maximum valid value for i is *m - n – 1*.

Question 3.B:

Before you dive into the code, you must answer the following "pre-write" questions and include your answers in your writeup, labeled with the question number. **For these questions, you must refer to the number of previous pieces of data as m, the order of the model as n, and the number of pieces of future data to be predicted as p.**

1. **Question 3.B.i:** Explain how you will use the inputs (model and last) to predict the first piece of future data. In particular, you should address how you will access the appropriate probabilities and how you will use those probabilities to select the future datum. Please be as clear and precise as possible!

   To find the first piece of prediction, I will compare the key last to the keys existed in the Markov chain model. If a match is not found, then return a random number among the tuple (0, 1, 2, 3). Else, meaning there is a first match, then I will find the dictionary associated with that key. Then, in this inner dictionary, I will use the function probability-weighted selection to select a prediction key based on that probability distribution.

   The probability-weighted selection will be implemented by generating a random float number between 0 and 1 using random.random(). Then, this number will be translated into a bucket among (0, 1, 2, 3) depending on the probability distribution saved in the dictionary retrieved earlier using a cumulative probability value as follow:

   ```
   rand = random.random()

   cum_val = 0.0
   ```

```
for item, val in model[last_tuple].items():
    cum_val = val + cum_val
    if rand < cum_val:
        next_val = item
```

2. **Question 3.B.ii:** Imagine that is greater than or equal to 2. How will you construct the state that will be used to make the second prediction? As a part of your explanation, please include a concrete example (*i.e.*, sample inputs + the first prediction generated + the state used to make the second prediction).

   After I find the first prediction key, I will append that key to last_tuple, then increment 1 space ahead to create a new list using slicing method as follow:

   ```
   last_tuple = last_tuple + (next_val, )

   last_tuple = last_tuple[1:]
   ```

   Then I loop for num of times to get future predictions.

   Example:

   sample input list: [1,2,2,1]
   order: 2
   sample previous state: (1,2)
   first prediction generated: 2
   append that key to sample previous state: (1,2,2)
   slicing the previous state to make the next state: (2,2)
   use next state to make second prediction: 1

   repeat num times to get num future predictions

3. **Question 3.B.iii:** In the event that a state is not found in the model, what Python function call could you make to generate a random integer between 0 and 3, inclusive? Please indicate not only the name of the function, but the input(s) that you will use.

   I would use random.randint(0,3) because this function will select an integer from 0 to 3 inclusive.

Homework Reflection:

1. What were the two most important concepts and/or skills that were reinforced by this assignment? Why do you believe that these concepts are important? (Your response must be 3-6 sentences.)

   Two most important concepts I've learned from this assignment is working with nested data structures and translating a stock data time series price into a Markov chain model. These skills are important in introducing me to nested data structures which are built inside/outside of other data structures.

2. How do the skills and concepts that you applied in this assignment transcend the specific application/problem that you were tackling? How could you envision applying these skills and concepts again in the future? (Your response must be 2-4 sentences.)

   Seeing how stock data can be turned into python code helps me to visualize the problem-solving process for other real-world problems. For example, defining the problems and sub-problems, translating word problems into math problems, and implementing these math problems using python code and python data structures. I will apply this skill in the future when organizing and translating the overarching problems and the sub-problems into variables while keeping track of the edge cases that are greater/smaller/equal or the max/min of the variables.

3. What do you believe you did well on this assignment? If you could do this assignment over, would you do anything differently? Why or why not? (Your response must be 2-4 sentences.)

   I think I've got the results correctly. If I could work on this assignment over, I would make sure to read and understand the data types of the expected inputs, outputs, parameters, as well as the edge cases because I spent a lot of time figuring out why I tested successfully locally but failed to pass the Owl tests. Another important error that I made was the logical placement of some statements. Tracing each line has helped me to figure out the bugs I created.

4. Do you think you're comfortable enough with the concepts covered in this assignment that you would be capable of teaching them to a peer? Why or why not? (Your response must be 2-4 sentences.)

   I am comfortable enough to teach to a peer the lessons I learned. This assignment is important in getting us students familiar with data simulating and modeling process using python.

Discussion questions:

a. Which stock/index can you predict with the lowest error? Based on the plots of the day-to-day change in stocks and the histogram of bins (which should pop up when you run your code), can you guess why that stock/index is easiest to predict?

   The stock/index that I can predict with the lowest error is DJIA.
   The reason that DJIA is easiest to predict could be that there is less variance over time with DJIA change in price than GOOG or FSLR change in price. Thus DJIA's changes are more predictable.

b. Given that we have divided the day-to-day price change into 4 bins, how many possible states are there in an nth order Markov chain for predicting the change in stock price?

   In an nth order Markov chain, a state has n numbers in it, representing the memory the state holds. Each of these positions can be either 0,1,2, or 3. So 4^n is the maximum possible combinations that an nth order Markov state can be.

c. The training data we gave you covers two years of data, with 502 data points per stock/index. With that data, is it possible to see all of the possible states in an nth order Markov chain? What are the constraints on n? How do you think it would affect the accuracy of the model if there were not enough data?

   4^5 is 1024 while 4^4 is 256. With 502 days, it is possible to see all the combinations of an nth order Markov chain if the n is less than or equal 4.
   If there were not enough data, the predictions would be not accurate because there are less previous states to consider. The chain will then rely on limited states to make predictions thus leave out the consideration for other missing possibilities that can affect the outcomes.

The print output the appears when you uncomment the call to the run() function and run your code:

```
FSLR
====
Actual: [3, 0, 0, 1, 0]
Order 1 : 3.419599999999992
Order 3 : 3.205200000000002
Order 5 : 3.367199999999999
Order 7 : 3.128800000000002
Order 9 : 3.073600000000001

GOOG
====
Actual: [1, 3, 3, 1, 1]
Order 1 : 2.104000000000001
Order 3 : 1.471199999999999
Order 5 : 1.873599999999998
Order 7 : 2.269999999999999
Order 9 : 2.3172

DJIA
====
Actual: [2, 2, 2, 2, 1]
Order 1 : 0.9620000000000002
Order 3 : 0.9559999999999995
Order 5 : 0.7984000000000001
Order 7 : 1.138399999999997
Order 9 : 1.458399999999999
```