DV1614, Assignment 2

Shahryar Eivazzadeh, September 2022

Introduction

In this assignment you are supposed to write a program that simulates transition between weather states. It is considered that there are four different states for the weather:

1. 'sunny'
2. 'cloudy'
3. 'rainy'
4. 'snowy'

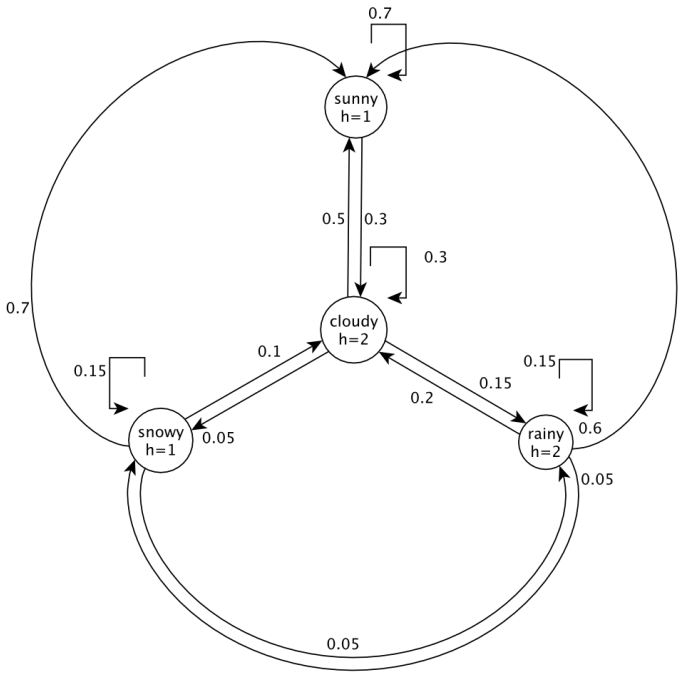
The weather is only in one of these states at each specific time (hour).

The weather changes between these four states by two factors:

1. Holding time: When in a state the weather stays in that state for a specific amount of time (hours) before changing to another state.
2. Transition probabilities: After spending the *holding time* in one state the weather can change its state to one of other states (or again back to the same state). This changing is specified by a set of *transition probabilities*. For each state, the *transition probabilities* indicates the probability of changing to another state (or even the current state itself). In another word, the new state will be chosen by chance, however the chance (probability) is specified in the *transition probabilities* list.

> *Scientific Hint*
>
> The model required in this assignment is actually a type of Continuous-Time Markov Chain (with fixed holdings). For the purpose of this assignment you do not need to know about this concept, but for your own curiosity or deeper understanding of similar situations you may want to learn about that.

Look at the following image to better understand the problem.



Now consider these notes:

1. The time unit in this simulation is *hour*.
2. The holding time for each state is a fixed value (in hours).
3. Since there are four states then there are four sets of transition probabilities. Each set corresponds to a transition from a state (eg 'cloudy') to all other states (including itself). In other words, there will be 4*4 =16 individual transition probability values.
4. The transition probabilities can be a value from 0 to 1 (eg: `[0.2,0.55,0,0.25]`).
5. The sum of all values in each list of transition is always 1 (apparently).
6. A value of 0 in the transition list means there is no chance of transition from current state to that specific state.
7. A value of 1 in the transition list means the current state will always change to that specific state (after spending the holding time).

Program Specification

1. You need to write a program in a file named `assignment2.py`
2. Your code will be imported to a `test.py` that is provided to you (you do not write test.py yourself)
3. You need to install `numpy` libraries in your python environment. (if you want to check Pylint score, you need to install `pylint` also, though not mandatory for assignment 2)
4. You can use only these libraries: `numpy`.
5. There are some features that you need to use by specification of the instruction (such as class, dictionary, generator functions …)
6. It is good to try to use `map`, `zip`

Class Specification

Consider these instructions when create the class:

1. You should implement a class called `WeatherSimulation`.
2. The constructor of the class receives two values:
   a. A dictionary type argument called `transition_probabilities`.
   b. A dictionary type argument called `holding_times`.
3. The `transition_probabilities` keys represent the state names as string type (i.e `sunny`, `cloudy`, `rainy`, `snowy`). However, your program should not make assumption about the keys' names.
4. The `transition_probabilities` values for the keys are dictionaries themselves, where the keys are strings representing state names. The values in these nested keys are probabilities. So, you already know that the `transition_probabilities` is a nested dictionary type. Look at the example code below and see `my_transitions`.
5. The program should check if the sum of the `transition_probabilities` is 1 or not, if it is not it should through exception of type `RuntimeError`, with some message.
6. The initial state for the simulation is 'sunny'.
7. A method, called `get_states()` should return a *list* of all states (*string*) (Do not put state names in your code, this should be extracted from the initialization data)
8. A method, called `current_state()` should return the current state (*string*)
9. A method, called `next_state()` should change the current state to a new state, based on the values in the `transition_probabilities`.
   a. TIP: Use `numpy.random.choice()`
10. A generator function should be implemented with name `iterable()` that yields `current_states()` in each call (so, it will go further in simulation step with each `next()` (the Python standard method))
11. A method called `simulate(hours)` should receive the number of simulation rounds (hours) as input argument and run that many of steps of the simulation (i.e. `next_state()`). It should accumulate all incidents of each state and at the end return the relative percentage of each state occurrence in a *list*. The order of values of the list corresponds to the order of keys in the input *transition probabilities*.
12. Hint: Try it for numbers above 10000, and you should see that the results will be almost stable (lets say +-10%)

List of all methods (except the constructor):

1. `get_states()` returns a list of all states
2. `current_state()` returns the current state
3. `next_state()` moves to the next state in simulation
4. `set_state(new_state)` changes the current state to the `new_state`
5. `current_state_remaining_hours()` returns the remaining hours in the current state.
6. `iterable()` a generator function that yields the current state (and moves to the next one in the simulation with standard `next()` function)
7.

`simulate(hours)` runs the simulation for the amount of `hours` and returns a list of percentages that each value shows the percent of times that a state occurred in the simulation. Hint: assume that the implicit order of states in list is the same as the order states in the initial dictionary which pass to the class.

PYTHON

```
#Sample of input data types
my_transitions = {
    'sunny':{'sunny':0.7, 'cloudy':0.3, 'rainy':0, 'snowy':0},
    'cloudy':{'sunny':0.5, 'cloudy': 0.3, 'rainy':0.15, 'snowy':0.05},
    'rainy':{'sunny':0.7, 'cloudy':0.2, 'rainy':0.15, 'snowy':0.05},
    'snowy':{'sunny':0.7, 'cloudy':0.1, 'rainy':0.05, 'snowy':0.15}
    }
my_holding_times = {'sunny':1, 'cloudy':2, 'rainy':2, 'snowy':1}
```

Submission Instructions

1. ONLY submit your file when your code passes the test.py (it will show **OK!**, otherwise it will show **NOK!**)
2. Follow exactly the naming (otherwise your code will fail), including upper or lower letter cases.
3. You need to upload both the `assignment2.py` and a file of the same content with `.txt` extension (that is `assignment2.txt`). DO NOT zip the files.
4. Care about avoiding plagiarism situation
5. You may need to install `numpy` in your python environment
6. You can use only these libraries: `numpy`, `functool`