COMP 3240 Spring 2023
Programming Assignment 3
Weather Prediction Through Matrix Multiplication

Daniel Tauritz, Ph.D.

March 21, 2023

Synopsis

The goal of this assignment is for you to apply your understanding of matrix multiplication to implement in Python 3 or Java a weather prediction program. These are individual assignments and plagiarism will not be tolerated. You are free to use libraries, except ones for matrix multiplication.

Markov Chain weather prediction

A very simple method for weather prediction is to predict tomorrow's weather based solely on today's weather. For example, if you knew that there is a certain chance that a rainy day (R) will be followed by a rainy day and that there is a certain chance that a dry day (D) will be followed by a dry day, then you can give probabilities for what tomorrow's weather will be based on today's weather, as well as predict the weather on subsequent days.

A Markov Chain is a process in which the probability of a system being in a particular state at a given observation period depends only on its state at the preceding observation period. In other words, $S_t = f(S_{t-1})$. The probability that the system is in state j at observation period t is denoted by $p_j^{(t)}$. The set of probabilities at observation period t for a system with n states is denoted by $P^{(t)} = \begin{bmatrix} p_1^{(t)} \\ p_2^{(t)} \\ \vdots \\ p_n^{(t)} \end{bmatrix}$

Assuming that the weather is either dry (D) or raining (R), the first problem you need to solve in predicting weather in this fashion, is to obtain the transition probabilities based on an observation record. For example, given the following thirteen day observation record:
RRRRDDDDDRDRD
you can compute the following transition probabilities:

$R \rightarrow R$: $\frac{3}{6} = \frac{1}{2}$

$R \rightarrow D$: $\frac{3}{6} = \frac{1}{2}$

$D \rightarrow D$: $\frac{4}{6} = \frac{2}{3}$

$D \rightarrow R$: $\frac{2}{6} = \frac{1}{3}$

The associated transition matrix T is then:

$$T = \begin{matrix} \\ D \\ R \end{matrix} \begin{matrix} \begin{matrix} D & R \end{matrix} \\ \begin{pmatrix} \frac{2}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} \end{pmatrix} \end{matrix}$$

where element i, j is the estimated probability of transitioning from the state indicated by column header i to the state indicated by row header j.
Assuming that the current day, the starting point of the prediction, is dry, the initial state vector for this problem is $P^{(0)} = \begin{bmatrix} p_1^{(0)} \\ p_2^{(0)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

To predict tomorrow's weather based on today's, we need to calculate $P^{(1)} = f(P^{(0)})$ with f being the transition function such that $P^{(t)} = f(P^{(t-1)})$.

$$f(P^{(t-1)}) = T \cdot P^{(t-1)} = \begin{bmatrix} \frac{2}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} \end{bmatrix} \cdot P^{(t-1)}$$

Using the associative properties of matrices and scalars we have:
$P^{(t)} = T \cdot P^{(t-1)} = T \cdot T \cdot P^{(t-2)} \ldots = T^t \cdot P^{(0)}$
Conclusion: using this Markov Chain model, the weather prediction for observation period t depends only on the transition matrix T and the initial state vector $P^{(0)}$.

Tomorrow's weather prediction is $P^{(1)}$, the weather prediction for the day after tomorrow is $P^{(2)}$, etc. Here are some sample computations:

$$P^{(1)} = T \cdot P^{(0)} = \begin{bmatrix} \frac{2}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 0.6667 \\ 0.3333 \end{bmatrix}$$

$$P^{(2)} = T^2 \cdot P^{(0)} \approx \begin{bmatrix} 0.6111 & 0.5833 \\ 0.3889 & 0.4167 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 0.6111 \\ 0.3889 \end{bmatrix}$$

$$P^{(3)} = T^3 \cdot P^{(0)} \approx \begin{bmatrix} 0.6019 & 0.5972 \\ 0.3981 & 0.4028 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 0.6019 \\ 0.3981 \end{bmatrix}$$

$$P^{(4)} = T^4 \cdot P^{(0)} \approx \begin{bmatrix} 0.6003 & 0.5995 \\ 0.3997 & 0.4005 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 0.6003 \\ 0.3997 \end{bmatrix}$$

It appears that our system is approaching an equilibrium state, describing the long-term behavior of the system. This fixed vector the system converges to is called its *steady-state vector*. A transition matrix T of a Markov Chain is called *regular* if all the entries in some power of T are positive. One can prove that if a Markov Chain has a regular transition matrix, then it also has a steady-state vector. Our weather prediction transition matrix is regular, therefore a steady-state vector exists. There are two ways to obtain the steady-state vector. The first is through numerical approximation like we have started to do, the other is by solving the equation $T \cdot \vec{V} = \vec{V}$ with the sum of the elements of $\vec{V}$ being equal to 1. This programming assignment focuses on the numerical approximation, but if you're interested in how to compute the exact steady-state vector (or would like to validate your implementation), then check out the appendix at the bottom of this document.

Problem statement

Write in Python 3 or Java, a Markov Chain prediction program which uses two parameters defined in your code as shown in the provided template: (I) a positive floating point number no larger than 0.1 indicating the desired precision of the climate prediction, and (II) a string consisting solely of D's and R's to specify the observation record, with the final observation specifying the current (initial state) weather, and with the constraint that all four possible transitions have to occur at least once; if the user tries to enter a level of precision violating the stated range, then the program should tell the user what the violation was and

have them reenter the precision until they enter a valid one; if the user tries to enter an observation record which violates the constraint, then the program should tell the user what the violation was and have them reenter the record until they have entered a valid one. Once a valid precision level and a valid record have been entered, the program should compute the weather prediction for the following 7 days and output that to the user, as well as compute the climate prediction with the desired precision. To estimate the latter, continue raising the transition matrix to increasingly larger powers until none of the matrix elements differ by more than the desired precision from their value in the previous power (i.e., they may be considered to have converged within the desired precision).

To summarize, the phases your program will execute are:

1. Obtain validated user input
2. Compute the transition matrix
3. Compute and output weather prediction for the following 7 days
4. Compute and output the climate prediction with the desired precision

Note that the second and third phase both rely on matrix multiplication, so you'll want to create a matrix multiplication function to use in both phases. Also note that you should output your 7-day predictions and steady-state vector to the command line using the given `print_predictions()` and `print_steady_state()` methods.

Resubmissions, penalties, documents, and bonuses

If you submit before the deadline, then you may resubmit up to a reasonable number of times till the deadline but not thereafter, your last on time submission will be graded. If you do not submit before the deadline, then your first late submission will be graded.

The penalty for late submission is a 5% deduction for the first 24 hour period and a 10% deduction for every additional 24 hour period. So 1 hour late and 23 hours late both result in a 5% deduction. 25 hours late results in a 15% deduction, etc. Not following submission guidelines can be penalized for up to 5%, which may be in addition to regular deduction due to not following the assignment guidelines.

Some assignments may offer bonus points for extra work, but note that the max grade for the average of all assignments is capped at 100%.

Deliverables & Due Date

The deliverables of this assignment are:

1. your source code with your name and the string "COMP 3240 Spring 2023 Programming Assignment 3" at the top of the file

You must name your file `PA3_Python.py` or `PA3_Java.java`. Submit your file directly to Gradescope, don't compress, zip, etc. The due date for this assignment is 10:00 PM on Thursday April 6, 2023.

Grading

The maximum number of regular points you can get is 50. The point distribution is as follows:

Algorithmic (e.g., does it provide the correct output for a given input)	30
Good programming practices including code reliability/efficiency/readability and commenting	15
Output to user (e.g., clearly state solution found, provide helpful error messages for invalid user input)	5

Appendix

Here's how you'd compute the exact steady-state vector for the number example shown earlier in this document:

$$\begin{bmatrix} \frac{2}{3} & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \text{ and } x + y = 1$$

$\frac{2}{3}x + \frac{1}{2}y = x$
$\frac{1}{3}x + \frac{1}{2}y = y$
$\frac{1}{3}x = \frac{1}{2}y$
$\frac{2}{3}x = y$
$\frac{2}{3}x = 1 - x$
$\frac{5}{3}x = 1$
$x = \frac{3}{5}$
$y = 1 - \frac{3}{5} = \frac{2}{5}$

Steady-state vector $= \begin{bmatrix} \frac{3}{5} \\ \frac{2}{5} \end{bmatrix}$

In other words, our long term climate prediction is that about 60% of the time we'll have dry days and about 40% of the time rainy days.