Markov Chains

Vasilis Tsioulos*

*Project 5: Computational mathematics II 2025

ABSTRACT

This study investigates the Markov Chains, specifically analyzes the recurrent and the transient states. In order to analyze the states, a transition matrix (probability matrix) is inputted and get the recurrent classes. The project itself is challenging, and in order to make it simple, the drunkard walk problem is deployed.

1 Introduction

1.1 Markov Chains

The Markov Chains[1] are a mathematical system that experiences transitions from one state to another according to probabilistic rules[2]. To model a stochastic process[3] with a Markov Chain, we need a set of states, for example a transition matrix. The matrix itself is providing the probabilities of every state. Moving from one state to another is called a transition and the states that cannot be escaped are the absorbing states or recurrent states. The transition matrix can appear as:

$$P = \begin{array}{cc|cc} & & pass & fail \\ \hline 1st & student & 0.8 & 0.2 \\ 2nd & student & 0.6 & 0.4 \end{array}$$

The transition matrix models student performance outcomes where the first column represents the probabilities of passing the class, while the second column shows the probabilities of failing. The first state corresponds to a student who studies consistently throughout the entire semester, while the second state represents a student who only crams during the final weeks before exams.

1.2 Drunkard's walk

A classic example in probability theory is the drunkard's walk, which models a random walk in one dimension. For simplicity, consider a drunkard moving along a straight line, taking steps either left or right with equal probability at each point in time. The process is random because each step is independent of previous steps. The walk terminates if the drunkard reaches either home or a pub, at which point the state becomes recurrent. The transition matrix of the problem appears to have the form of:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{array}{c|ccccc} & H & S_0 & S_1 & S_2 & P \\ \hline State0 & 1 & 0 & 0 & 0 & 0 \\ State1 & 0.5 & 0 & 0.5 & 0 & 0 \\ State2 & 0 & 0.5 & 0 & 0.5 & 0 \\ State3 & 0 & 0 & 0.5 & 0 & 0.5 \\ State4 & 0 & 0 & 0 & 0 & 1 \end{array}$$

2 Computational analysis

The analysis of the matrix was achieved with the help of the "networkx" library of Python which has the ability to make graph plots with weights and biases. When the transition matrix is defined, the most crucial part of the code is the part that loops through the components and divides the states to transient and recurrent classes.

```
sccs = list(nx.strongly_connected_components(G))

recurrent_classes = []
recurrent_class_ids = {}
transient_states = []

class_id = 0
for scc in sccs:
    is_closed = True
    for state in scc:
        for neighbor in G.successors(state):
            if neighbor not in scc:
                is_closed = False
                break
        if not is_closed:
            break

    if is_closed:
        for state in scc:
            recurrent_class_ids[state] = class_id
        recurrent_classes.append(scc)
        class_id += 1
    else:
        transient_states.extend(scc)
```

Listing 1. Recurrent and transient states search

The results of the code are:

```
State Analysis:
State 0: Recurrent (Class ID: 0)
State 1: Transient
State 2: Transient
State 3: Transient
State 4: Recurrent (Class ID: 1)
```

As predicted by the model, the states representing home and the pub form distinct recurrent classes, indicating that once the drunkard reaches either location, the process terminates. Graphically, the states appear as:

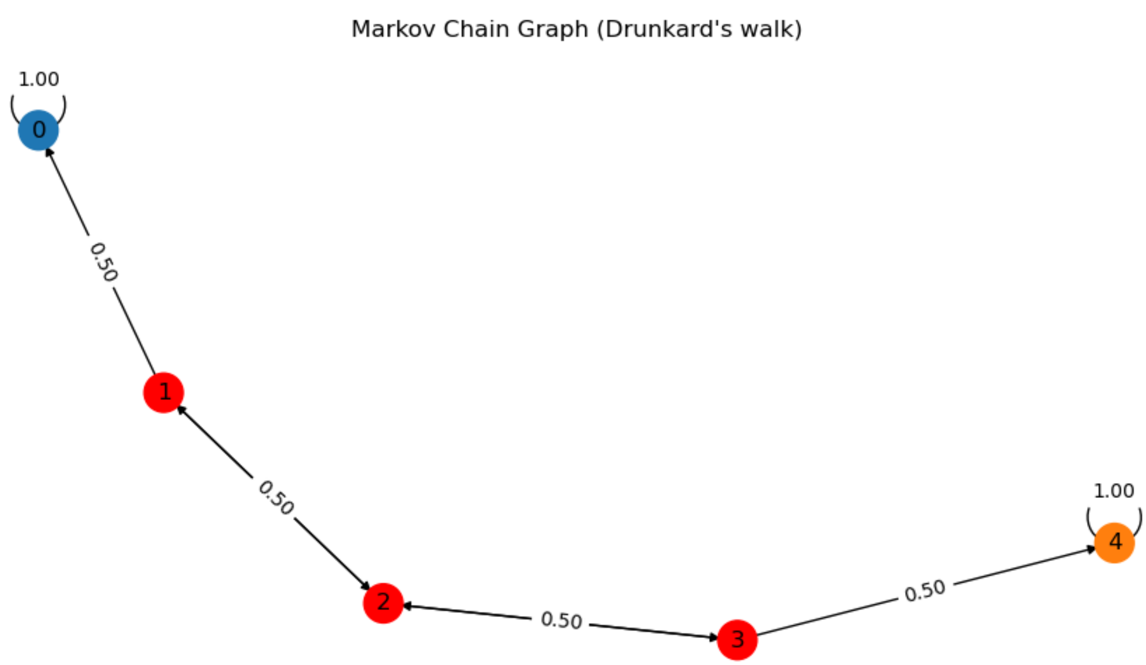

Figure 1. Drunkard's walk Markov Chain

3 Deepseek's answer

3.1 Question

Develop a program that takes the transition matrix of a Markov Chain as input. The program should analyze each state in the Markov Chain and determine whether it is recurrent or transient. If a state is recurrent, the program should provide the ID of the recurrent class to which the state belongs.

3.2 Problem Statement

Develop a program that:

- Takes the transition matrix of a Markov Chain as input
- Analyzes each state to determine whether it is recurrent or transient
- For recurrent states, provides the ID of the recurrent class to which the state belongs

3.3 Approach

The solution follows these steps:

3.3.1 Input the Transition Matrix

The user provides the transition probabilities between states in an $n \times n$ matrix where each row sums to 1.

3.3.2 Identify Communication Classes

States that communicate with each other (mutually reachable) form a class: [H] [1] Initialize empty list of classes Initialize visited array of size n to False each state i from 0 to $n-1$ not visited[i] Find all states mutually reachable with i Mark them as visited Add to classes list

3.3.3 Classify States

- A class is **recurrent** if no state in the class can transition to any state outside the class
- A class is **transient** if at least one state in the class can transition to a state outside the class

3.3.4 Output Results

For each state, indicate whether it's transient or recurrent, and if recurrent, specify its class ID.

3.4 Solution Implementation

The Python implementation consists of three main functions:

3.4.1 Main Function

Handles user input and output:

```
def main():
    # Get transition matrix from user
    # Validate matrix (rows sum to 1)
    # Call analysis function
    # Print classification results
```

3.4.2 *Analysis Function*

Core classification logic:

```
def analyze_markov_chain(transition_matrix):
    # Identify communication classes
    # Classify each class as recurrent/transient
    # Prepare classification output
```

3.4.3 *Reachability Helper Function*

```
def can_reach(transition_matrix, from_state, to_state):
    # Check if to_state is reachable from from_state
    # Using breadth-first search limited to 100 steps
```

3.5 Algorithm Explanation

The complete algorithm works as follows:

1. **Input Validation**: Verify each row of the transition matrix sums to 1
2. **Class Identification**:
    - For each unvisited state, find all states that are mutually reachable
    - These form a communication class
3. **Classification**:
    - For each class, check if any state can transition outside the class
    - If yes: transient class
    - If no: recurrent class
4. **Output**: Display classification for each state

3.6 Example Usage

For a transition matrix:

$$P = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

The program would identify:

- State 0: Transient
- State 1: Transient
- State 2: Recurrent (Class 0)

3.7 Conclusion

This program provides a systematic way to analyze Markov Chains by classifying states as recurrent or transient and identifying recurrent classes. The implementation uses graph traversal techniques to determine communication classes and reachability between states.

References

1. Kosmidis, K. Computational mathematics ii notes.

2. Brilliant.org. Markov chains (n.d.). Accessed: [Insert Date Accessed].

3. Mahfuz, F. Markov chains and their applications (2021). Math Theses, Paper 10.