Computational Physics

Topic 03 : Computational Problems involving Marko Chains

Lecture 01 : Review of Markov Chains

Dr Kieran Murphy

Computing and Mathematics, SETU (Waterford).
(kieran.murphy@setu.ie)

Autumn Semester, 2022/23

Outline

- Some simple models
- Terminology and definitions

Example (Land of Oz)

Description

The Land of Oz is blessed by many things, but not by good weather. They never have two nice days in a row. If they have a nice day, they are just as likely to have snow as rain the next day. If they have snow or rain, they have an even chance of having the same the next day. If there is change from snow or rain, only half of the time is this a change to a nice day.

Example (Land of Oz)

Description

The Land of Oz is blessed by many things, but not by good weather. They never have two nice days in a row. If they have a nice day, they are just as likely to have snow as rain the next day. If they have snow or rain, they have an even chance of having the same the next day. If there is change from snow or rain, only half of the time is this a change to a nice day.

Formalisation . . . as a transition matrix

$$\begin{array}{cc} & \text{. . . to . . .} \\ & \begin{array}{ccc} \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \text{. . . from . . .} \begin{array}{c} \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} & \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \end{array}$$

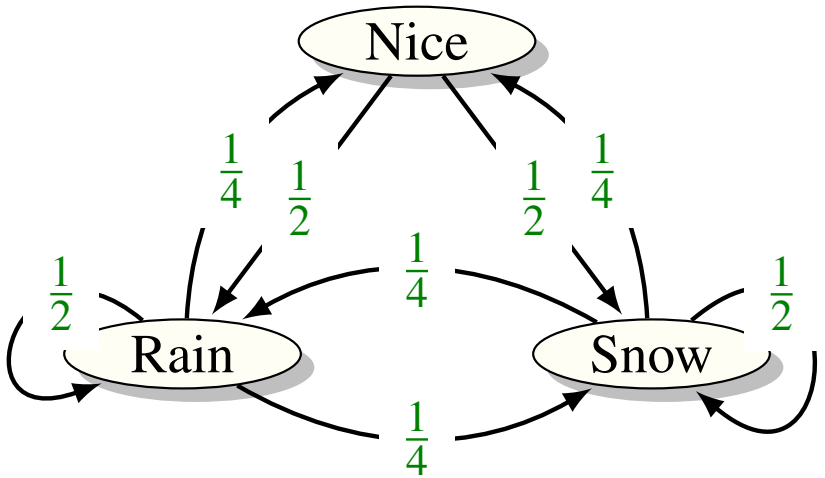
Example (Land of Oz)

Description

The Land of Oz is blessed by many things, but not by good weather. They never have two nice days in a row. If they have a nice day, they are just as likely to have snow as rain the next day. If they have snow or rain, they have an even chance of having the same the next day. If there is change from snow or rain, only half of the time is this a change to a nice day.

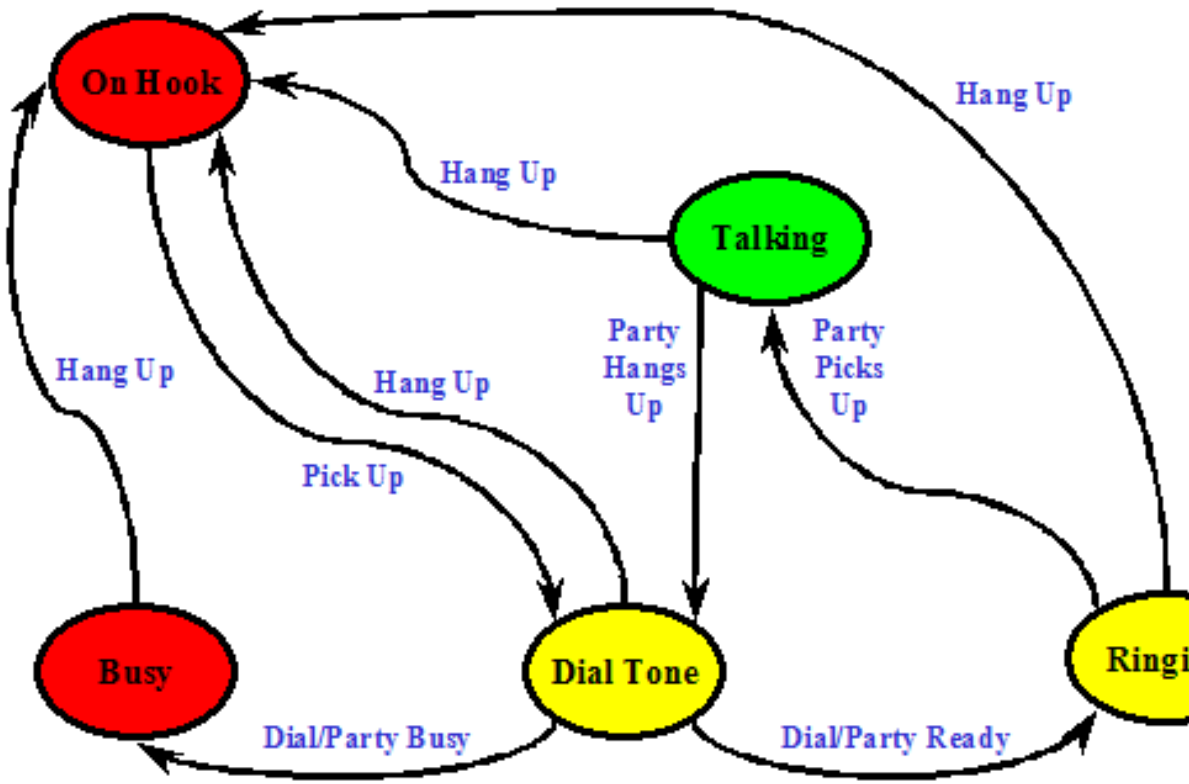
Formalisation . . . as a transition matrix . . . and finite state machine

$$\begin{array}{cc} & \begin{array}{ccc} \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \begin{array}{c} \text{from} \ldots \\ \vdots \end{array} & \begin{array}{c} \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \end{array} \begin{array}{c} \ldots \text{ to } \ldots \\ \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \end{array}$$



Example (Model Based Testing)

Model-Based Testing*is the automatic generation of efficient test procedure sequences — using finite state machines or Markov chains – to represent a system's requirements and specified functionality.



*Google "Model Based Testing" or read Harry Robinson's (Microsoft) paper http://www.ecs.csun.edu/ rlingard/COMP595VAV/GraphTheory Techniques In Model-Based Testing.pdf

Example (Worker Employment Model) I

Example 1

Consider a worker who, at any given month, t, is either unemployed (state 0) or employed (state 1). Suppose that, over a one month period:

- An unemployed worker finds a job with probability $\alpha \in (0, 1)$.
- An employed worker loses their job and becomes unemployed with probability $\beta \in (0, 1)$.

Formulate model as a Markov chain[†]. Then:

1. What is the average duration of unemployment?
2. Over the long-run, what fraction of time does a worker find themself unemployed?
3. Conditional on employment, what is the probability of becoming unemployed at least once over the next 12 months?

[†]Since this model has only two states we could model this using the geometric distribution.

Specifying a Markov Chain I

Any Markov process/chain can be described as follows:

- We have a set of states, $S = \{s_1, s_2, \ldots, s_r\}$.

  For the *Land of Oz* example the states were

  $$S = \{\underbrace{\text{Rain}}_{s_1}, \underbrace{\text{Nice}}_{s_2}, \underbrace{\text{Snow}}_{s_3}\}$$

- The process starts (at step/stage 0) in one of these states and moves successively from one state to another, one step/stage at a time.
- Each move is called a step, and after n steps the process is at stage n, generating a run.

  For the *Land of Oz* example some possible runs are:
  - Rain, Rain, Rain, Nice, Snow, Rain, Snow, Snow, Nice, ...
  - Rain, Rain, Rain, Rain, Rain, Rain, Rain, Rain, ...

  while the following sequence is not possible (Why?)
  - Snow, Snow Snow, Nice, Nice, Snow, Snow, Snow, ...

Specifying a Markov Chain II

- If the chain is currently in state s_i, then it moves to state s_j at the next step/stage with a transition probability denoted by p_{ij}.

The probability does not depend upon which states the chain was in bef current state $\Longrightarrow$ The next state only depends on the current state and past states (Markov memory less property).

For the *Land of Oz* example the matrix of transition probabilities is

$$\begin{array}{cc} & \begin{array}{ccc} \multicolumn{3}{c}{\ldots \text{ to } \ldots} \\ \text{Rain} & \text{Nice} & \text{Snow} \end{array} \\ \begin{array}{c} \text{from} \ldots \\ \vdots \end{array} & \begin{array}{c} \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \end{array} = P$$

- The Markov chain can remain in the state it is in, and this occurs with probability p_{ii}.

Specifying a Markov Chain III

- The random variable X_n represents the state of the Markov chain at stage n. It assumes values $S = \{s_1, s_2, \ldots, s_r\}$ with probability distribution

$$\boldsymbol{u}_n = (u_{n1}, u_{n2}, \ldots, u_{nr})$$

- The Markov memoryless property can be expressed in terms of conditional probability as follows

$$Pr(X_{n+1} = s | X_n, X_{n-1}, \ldots, X_0) = Pr(X_{n+1} = s | X_n)$$

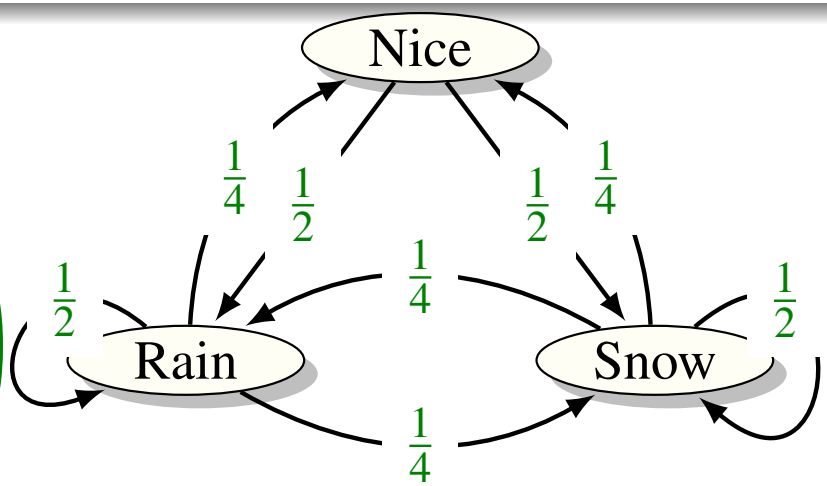
(No extra uncertainty is removed by knowing what has happened in the past.)

Of interest are:

- Given the initial state (i.e. given $\boldsymbol{u}_0$) what is the probability distribution for the state at some later stage n?
- What is the long term behaviour of the chain?

Transition Matrix

$$P = \begin{array}{c} \\ \text{from} \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} \multicolumn{3}{c}{\ldots \text{ to } \ldots} \end{array} \\ \begin{array}{c} \\ \text{Rain} \\ \text{Nice} \\ \text{Snow} \end{array} \begin{pmatrix} \text{Rain} & \text{Nice} & \text{Snow} \\ 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} \end{array}$$



- From the total law of probability the **sum of the entries along every row equals one**. (Sum of probabilities of outgoing arcs on each node equals one).‡
- If given the probability distribution for some stage n, then the probability distribution for some later stage $n + m$, is

$$\boldsymbol{u}_{n+m} = \boldsymbol{u}_n P^m$$

In other words, to move the distribution forward m units of time, we post-multiply by P^m.

‡The sum down the columns need not be one (sum of probabilities of incoming arcs), but if the column totals are also one then this special type of process is called a doubly stochastic

Example (Land of Oz, cont)

Question

Today it is raining in the Land of Oz. What is the probability distribution for tomorrow?

Solution

Since $s_1 = \text{Rain}$, we have $\boldsymbol{u}_0 = (1, 0, 0)$ and we want $\boldsymbol{u}_1$.

Using $\boldsymbol{u}_1 = \boldsymbol{u}_0 P$ we have

$$(1, 0, 0) \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/2 \end{pmatrix} = (1/2, 1/4, 1/4)$$

So the probability of rain tomorrow is 1/2, nice tomorrow is 1/4, and snow tomorrow is 1/4.

Example (Land of Oz, cont)

Question

Today it is nice in the Land of Oz. What is the probability of it raining in two days time?

Solution

Since $s_2 = \text{Nice}$, we have $\boldsymbol{u}_0 = (0, 1, 0)$ and we want the first component of $\boldsymbol{u}_2$.

Using $\boldsymbol{u}_2 = \boldsymbol{u}_0 P^2$ we have

$$(0,1,0)\begin{pmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0 & 0.5 \\ 0.25 & 0.25 & 0.5 \end{pmatrix}^2 = (0,1,0)\begin{pmatrix} 0.438 & 0.188 & 0.375 \\ 0.375 & 0.250 & 0.375 \\ 0.375 & 0.188 & 0.438 \end{pmatrix}$$

$$= (0.375, 0.250, 0.375)$$

So the probability of rain in two days time is 0.375, probability of being nice is 0.250 and probability of snow is 0.375.

Python Implementation I

Python setup

```
1 import numpy as np
  import matplotlib.pyplot as plt
  from numpy.random import default_rng
  rng = default_rng(12)
```

Define states

```
2 # label states for output
  labels = np.array(["Rain","Nice","Show"
  states = np.arange(0, len(labels))
  print("Labels:", labels)
  print("States:", states)
```

```
Labels: ['Rain' 'Nice' 'Show']
States: [0 1 2]
```