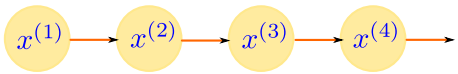
Lecture 6

Markov Chain Monte Carlo

Contents

1 Definitions

Typically, a stochastic process is a system which evolves in time while undergoing random fluctuations.[1] Formally, a **stochastic process** is a non-deterministic sequence of random variables, $x^{(i)}$, for $x^{(i)} \in \mathbf{X}$.



A particular x_i is called a **state**. It may be scalar/vector, discrete/continuous, or more complicated.

$\mathbf{X}$ is called the **state space** or the **phase space**. It is the set or space of all possible $\{x_1, x_2, x_3, ...\}$ (think *population*).[2]

A trivial example may be a series of independent coin tosses. There are two states, and the state space $\mathbf{X} = \{\text{H}, \text{T}\}$. A particular realization may be,

$$H \to T \to T \to H \to T \to \cdots$$

This is not a particularly interesting example because successive states are independent. In more interesting examples, there is some correlation between successive states.

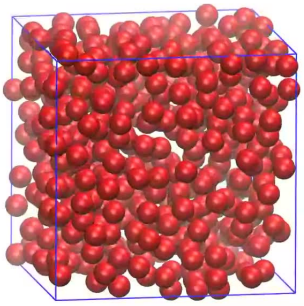
Process	**State Space**	**Realization**
number of customers in a store every hour	non-negative integers	5, 3, 0, 1, $\cdots$
stock price of Apple Inc. every second	non-negative real number	\$200.05, \$200.01, $\cdots$
daily low/high temperatures in Tallahassee	2D array of real numbers	[-1.1, 32], [2.5, 43], $\cdots$
score in a soccer game	2D array of integers	[0, 0], [1, 0], [2, 0] $\cdots$

We can also think of more complicated examples. For instance think about a molecular simulation with N similar particles.

A state x_i (or configuration) may correspond to the positions and velocities of all the particles in the system. Each particle has 6 "co-ordinates":

- position (r_x, r_y, r_z)
- velocity (v_x, v_y, v_z)

Thus x_i is a vector of size $6N$.



wikipedia.org

1.1 Markov Process

A Markov process is a special type of stochastic process.

[1] For simplicity, we shall assume that "time" increases in discrete units, say by making observations after fixed intervals of time. As we make this interval smaller and smaller, we can approach the continuous limit.

[2] In the notation used here, the superscript identifies the position of a state in a sequence, while the subscript identifies a particular state.

In a Markov process, the next state depends only on the current state. It is independent of past states. This can be mathematically represented as,

$$\pi(x^{(i)}|x^{(i-1)}, x^{(i-2)}, ..., x^{(0)}) = \pi(x^{(i)}|x^{(i-1)}) \tag{1}$$

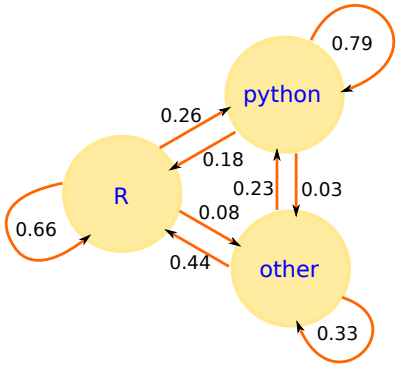
The dependence of successive states is the source of persistence or correlation. It has some "memory", but not too much.

Consider a state space with n distinct states, $\mathbf{X} = \{x_1, x_2, \cdots x_n\}$. A Markov process can be completely specified by prescribing the transition probabilities $W_{ij} = \pi(x_i|x_j)$ for all pairs (i, j) of states. These probabilities can be encoded in an $n \times n$ matrix $\mathbf{W} = W_{ij}$, which is called the **transition probability matrix** (TPM).[3]

The TPM is the heart and soul of a Markov process. Let us illustrate it with an example.

2 Transition Probability Matrix

I found this picture describing the traffic of programmers between R, python, and "other" languages for the year 2013.



Consider the "R" blob. The graphic implies that over a cycle, the probability that an R programmer switches to python is 0.26. The programmer continues to use R with probability 0.66.[4]

Let us represent the state space with $\mathbf{X} = \{r, p, o\}$ (R, Python, Other).

Let $\{\pi_r, \pi_p, \pi_o\}$ represent the fraction or probability of programmers currently using R, python, and "other" as their primary programming language, respectively.

Particular Markov chains might look like:

- rrpppprrpo...
- orrrpppppp...

In this example, one may think of a particular Markov chain as the trajectory of a particular programmer over their life.

[3] For most interesting problems n is large (if not infinite), and hence the matrix $\mathbf{W}$ is huge ($n \times n$), and is never directly stored in memory.

[4] I don't trust such numbers, but let us temporarily suspend our skepticism and take them seriously.

First, let us encode this graphic into a TPM,

$$\mathbf{W} = \begin{bmatrix} 0.66 & 0.18 & 0.44 \\ 0.26 & 0.79 & 0.23 \\ 0.08 & 0.03 & 0.33 \end{bmatrix} \tag{2}$$

Here we adopt the following convention for the order of the columns and rows:

$$\begin{array}{ccc} \text{R} & \rightarrow & 1 \\ \text{Python} & \rightarrow & 2 \\ \text{Other} & \rightarrow & 3 \end{array}$$

Thus, $W_{32} = W_{3\leftarrow 2} = W_{o\leftarrow p} = W_{op} = 0.03$.[5] Notice that none of the numbers are negative (zeros are allowed), and the columns all add up to one; more on this later.

Given this information, we may be interested in questions like:

- after n cycles, what is the relationship of π_r, π_p, π_o to the initial values
- after infinite cycles, what is the eventual "marketshare"?

Let $\pi_r^{(n)}$ represent the fraction of R programmers after n cycles.[6] After the first cycle:

$$\pi_r^{(2)} = \pi_r^{(1)} W_{rr} + \pi_p^{(1)} W_{rp} + \pi_o^{(1)} W_{ro} \tag{3}$$

The first term in the summation represents an R programmer who stays with R, the second term represents a python programmer in cycle 1 who transitioned to R, and the last term represents an "other" programmer switching to R.

We can write similar expressions for $\pi_p^{(2)}$ and $\pi_o^{(2)}$.

$$\begin{aligned} \pi_p^{(2)} &= \pi_r^{(1)} W_{pr} + \pi_p^{(1)} W_{pp} + \pi_o^{(1)} W_{po} \\ \pi_o^{(2)} &= \pi_r^{(1)} W_{or} + \pi_p^{(1)} W_{op} + \pi_o^{(1)} W_{oo}. \end{aligned} \tag{4}$$

Knowing the rules for matrix multiplication, we can combine these three equations as:

$$\boldsymbol{\pi}^{(2)} = \begin{bmatrix} \pi_r^{(2)} \\ \pi_p^{(2)} \\ \pi_o^{(2)} \end{bmatrix} = \begin{bmatrix} W_{rr} & W_{rp} & W_{ro} \\ W_{pr} & W_{pp} & W_{po} \\ W_{or} & W_{op} & W_{oo} \end{bmatrix} \begin{bmatrix} \pi_r^{(1)} \\ \pi_p^{(1)} \\ \pi_o^{(1)} \end{bmatrix} = \mathbf{W}\boldsymbol{\pi}^{(1)}. \tag{5}$$

If this is not immediately clear, dwell on the equation, until you "see" it.

Sure, matrix notation is compact. But what else is it good for? As we shall see, (i) it helps us generalize and compute, and (ii) through the magic of linear algebra we get insight into important properties of $\mathbf{W}$.

Generalizing eqn 5, after n steps.

$$\boldsymbol{\pi}^{(n+1)} = \mathbf{W}\boldsymbol{\pi}^{(n)} = \mathbf{W}^n \boldsymbol{\pi}^{(1)}. \tag{6}$$

[5] Take your time to get comfortable with this convention. Some people find it confusing that $W_{ij} = \pi(x_i|x_j)$ is the probability of transition from state j to i, and not the other way around.

[6] We use the labels fraction and probability loosely in this example.

2.1 Stationary Distribution

Let us crunch some numbers. Let us assume that currently, programmers are evenly split into the three camps ($\pi_r^{(1)} = \pi_p^{(1)} = \pi_o^{(1)} = 1/3$. Given,

$$\mathbf{W} = \begin{bmatrix} 0.66 & 0.18 & 0.44 \\ 0.26 & 0.79 & 0.23 \\ 0.08 & 0.03 & 0.33 \end{bmatrix} \text{ and } \boldsymbol{\pi}^{(1)} = \begin{bmatrix} 0.3333 \\ 0.3333 \\ 0.3333 \end{bmatrix},$$

we can perform repeated matrix-vector multiplication to track the evolution of the fraction of programmers in the three camps,

$$\boldsymbol{\pi}^{(2)} = \mathbf{W}^1\boldsymbol{\pi}^{(1)} = \begin{bmatrix} 0.4229 \\ 0.4817 \\ 0.0953 \end{bmatrix}, \qquad \boldsymbol{\pi}^{(5)} = \mathbf{W}^4\boldsymbol{\pi}^{(1)} = \begin{bmatrix} 0.3896 \\ 0.5383 \\ 0.0721 \end{bmatrix}$$

$$\boldsymbol{\pi}^{(10)} = \mathbf{W}^9\boldsymbol{\pi}^{(1)} = \begin{bmatrix} 0.3816 \\ 0.5483 \\ 0.0702 \end{bmatrix}, \qquad \boldsymbol{\pi}^{(50)} = \mathbf{W}^{49}\boldsymbol{\pi}^{(1)} = \begin{bmatrix} 0.3812 \\ 0.5487 \\ 0.0701 \end{bmatrix}$$

We notice that after a reasonably large n, the distribution settles to

$$\boldsymbol{\pi}^{(\infty)} = \begin{bmatrix} 0.3812 \\ 0.5487 \\ 0.0701 \end{bmatrix}$$

Before we analyze this further, let us try a different initial state, say $\boldsymbol{\pi}^{(1)} = [0 \quad 0 \quad 1]^T$, where all programmers start out in the "other" category. We get the sequence,

$$\boldsymbol{\pi}^{(2)} = \begin{bmatrix} 0.477 \\ 0.372 \\ 0.151 \end{bmatrix}, \boldsymbol{\pi}^{(5)} = \begin{bmatrix} 0.4034 \\ 0.5209 \\ 0.0757 \end{bmatrix}, \boldsymbol{\pi}^{(10)} = \begin{bmatrix} 0.3822 \\ 0.5475 \\ 0.0703 \end{bmatrix}, \boldsymbol{\pi}^{(50)} = \begin{bmatrix} 0.3812 \\ 0.5487 \\ 0.0701 \end{bmatrix}$$

This is interesting! Regardless of what state we start in $\boldsymbol{\pi}^{(1)}$, after a reasonably large number of cycles, we end up with the same **stationary** or **limiting distribution** $\boldsymbol{\pi}^{(\infty)}$.

Note that $\boldsymbol{\pi}^{(\infty)}$ is completely independent of $\boldsymbol{\pi}^{(1)}$. After enough cycles, the Markov chain forgets where it started from.[7] The stationary distribution is an intrinsic property of $\mathbf{W}$. We hurtle towards it, regardless of where we start.

Once we arrive at the stationary distribution, we are stuck. Mathematically, this is an interesting property,

$$\boldsymbol{\pi}^{(\infty)} = \mathbf{W}\boldsymbol{\pi}^{(\infty)}. \tag{7}$$

If you have seen some linear algebra, this might remind you of an eigenvalue problem: $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$. Here $\lambda = 1$, $\mathbf{v} = \boldsymbol{\pi}^{(\infty)}$, and $\mathbf{A} = \mathbf{W}$. One can show (see appendix A.1), that the stationary distribution is the eigenvector of $\mathbf{W}$ corresponding to eigenvalue 1.[8]

[7] recall that a Markov process has some memory, but not too much

[8] If you have seen it before, the process of repeated matrix multiplication might remind you of Power Method for finding the eigenvector corresponding to the largest eigenvalue.

This means that we do not have to perform repeated matrix multiplication to find the stationary distribution. The stationary distribution is simply the "eigenvector" corresponding to the largest eigenvalue (see A.2.1 to see the eigenvalues and eigenvectors of $\mathbf{W}$).

2.2 Properties of Transition Property Matrix

A proper TPM $\mathbf{W}$ has three essential properties:

(i) ergodicity
(ii) normalization
(iii) balance

Notation Alert: In what follows, I shall drop the superscript (∞) and assume that π_i is the probability of state i in the stationary distribution,

$$\pi_i^{(\infty)} \to \pi_i.$$

Furthermore, I shall use the shorthand, $\boldsymbol{\pi} = [\pi_1 \ \ \pi_2 \ \ ...\pi_N]^T$, where N is the number of states in the state space.

Let us now look at the three essential properties in order.

2.2.1 Ergodicity

This condition guarantees that a Markov chain, if run sufficiently long, shall visit all states in the state space. For complicated problems, ergodicity is nearly impossible to prove.

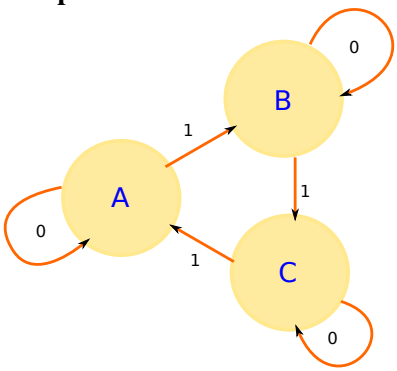
It is generally guaranteed if the Markov chain is *aperiodic* and *irreducible*.

It is aperiodic if it does not get stuck in cycles.

It is irreducible if it is possible to traverse between any two states (of non-zero probability) in a finite number of steps; mathematically, if $\pi_i > 0$ and $\pi_j > 0$, then for some n, $\mathbf{W}$ must satisfy the property, $W_{ij}^n > 0$.

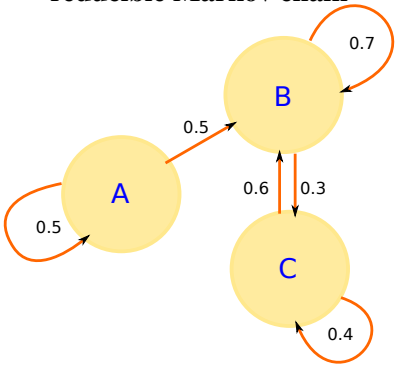
Perhaps a good way to illustrate these ideas is by considering counter-examples.

periodic Markov chain



A → B → C → A → B → ...

reducible Markov chain



A → B → B → C → B → B...

In a periodic Markov chain, such as the one shown in the figure on the left, the chain gets stuck in a cycle. If it enters this cycle anywhere, the same pattern repeats indefinitely. If this cycle is attached

to a larger system of states, then those states will be ignored once the chain finds its way into the cycle and gets trapped.

In the reducible Markov chain shown in the figure above on the right, there is was path back to state A from B or C.

2.2.2 Normalization

The columns of $\mathbf{W}$ have to sum to 1, since they represent transition probabilities.

$$\sum_i W_{ij} = W_{1j} + W_{2j} + ... + W_{jj} + ... + W_{Nj} = 1. \tag{8}$$

Normalization is the mathematical codification of the intuition "in the next step, the Markov chain has to travel to *some* state." This includes staying put.

Note that the summation in eqn. 8 is over the first index. There are N such conditions one can write for each of the $j = 1, 2, ..., N$ columns.

2.2.3 Balance

This condition ensures that the equilibrium state corresponds to eigenvalue 1.

$$\sum_j W_{ij}\pi_j = \pi_i, \quad i = 1, 2, ..., N \tag{9}$$

Note that unlike eqn. 8 the summation here is over the second index. The equation makes more intuitive sense when written out in matrix form (and recognizing that the summation is just matrix multiplication)

$$\mathbf{W}\boldsymbol{\pi} = \boldsymbol{\pi}.$$

3 The Design Problem

If $\mathbf{W}$ is given to us, finding $\boldsymbol{\pi}$ is relatively easy. It is uniquely determined if $\mathbf{W}$ satisfies the three conditions listed above.

However, in a typical application, our goal is to sample $x_i \sim \boldsymbol{\pi}(x)$. Therefore, given a PDF $\boldsymbol{\pi}(x)$, our goal is to **design** or **construct** the TPM $\mathbf{W}$. Operationally, this is the inverse of what we have obsessed over in this chapter so far.

Like most inverse problems, we have to deal with the fact that the solution to the inverse problem may not be unique. In simpler terms, given a particular $\boldsymbol{\pi}(x)$ there may be many possible choices of $\mathbf{W}$. Mathematically, this is not difficult to comprehend. Given a TPM $\mathbf{W}$, we could find $\boldsymbol{\pi}$ – its principal eigenvector. However, given $\boldsymbol{\pi}$, one cannot "find" a unique matrix $\mathbf{W}$. There are too many matrices that can fit the bill.

Practically, this is a good thing! It means we have flexibility in setting up $\mathbf{W}$. There are many ways to skin a cat! Different MCMC algorithms make different choices for $\mathbf{W}$.

Usually, in designing a MCMC method,[9] the last condition (balance), is replaced with a stricter condition called **detailed balance**. The thinking is "we have too much freedom; it might be a good idea to give some of that up, especially if it facilitates algorithm design."

Note that the imposition of detailed balance still does not completely specify $\mathbf{W}$. It eliminates many choices, but still leaves many open.

3.1 Detailed Balance

This is a sufficient condition that is stricter than the balance condition (eqn. 9).

It states that the *net traffic* between any two states is zero.

$$W_{ij}\pi_j = W_{ji}\pi_i. \tag{10}$$

While more restrictive, it is easier to specify and enforce. It is useful to make the relationship between eqns 10 and 9 explicit. Starting with eqn 10, we can sum up over "j" on both sides of the equation. From the normalization condition (eqn 8), we find,

$$\begin{aligned}\sum_j W_{ij}\pi_j &= \sum_j W_{ji}\pi_i \quad \text{detailed balance} \\ &= \pi_i \sum_j W_{ji} \quad \text{normalization} \\ &= \pi_i.\end{aligned}$$

Thus detailed balance $\implies$ balance, which is a requirement for TPM to be legitimate. But balance $\not\Rightarrow$ detailed balance.

Let us do some order of magnitude analysis. In general, for a state space of size n, the number of TPM elements we need to specify is $\mathcal{O}(n^2)$. From the normalization and balance conditions, we get $\mathcal{O}(n)$ eqns or constraints each. However, these constraints are not necessarily independent (see Exercise ii). Detailed balance yields ${}^nC_2 - n = \mathcal{O}(n^2/2)$ constraints. For large n that still leaves $\mathcal{O}(n^2/2)$ degrees of freedom for large n.

4 Summary

A Markov process is a stochastic process in which *next* state depends only on the *current* state. This dependence is specified through the transition probability $\pi(x^{(k+1)}|x^{(k)})$. The transition probability matrix describes the transition probabilities between all pairs of states in the state space.

$$W_{ij} = W_{i\leftarrow j} = \pi(x_i^{(k+1)}|x_j^{(k)}).$$

It has three essential properties: (i) ergodicity, (ii) normalization, and (iii) balance. These imply,

$$\mathbf{W}\boldsymbol{\pi} = \boldsymbol{\pi},$$

where $\boldsymbol{\pi}$ is the stationary distribution. This can be exploited by in designing $\mathbf{W}$ that helps us sample from $\boldsymbol{\pi}$.

[9] or equivalently the TPM

In practice the balance condition, is replaced by detailed balance.

$$W_{ij}\pi_j = W_{ji}\pi_i.$$

Even so, there are many $\mathbf{W}$ that correspond to a particular target $\boldsymbol{\pi}$. These different TPMs correspond to different MCMC methods.

5 Problems

5.1 Thought Exercises

(i) True or False: The TPM is symmetric.

(ii) Consider a 2 state Markov chain with stationary distribution $\mathbf{p} = [p_1,\ p_2]^T$. Our goal is to consider the design problem of building a suitable 2×2 TPM $\mathbf{W}$.

- Show that the normalization condition (eqn 8) leaves us with two degrees of freedom,

$$\mathbf{W} = \begin{bmatrix} a & b \\ 1-a & 1-b \end{bmatrix}.$$

- Show how the balance condition (eqn 9) leads to an under-determined system, with

$$a = 1 - \frac{p_2}{p_1}b,$$

that still leaves one degree of freedom unresolved.

- Let $\mathbf{p} = [0.3 \quad 0.7]^T$. Find two different TPMs which yield $\mathbf{p}$ as the stationary distribution.
- Show how in this case, detailed balance leads to the same relation between a and b.

5.2 Numerical Questions

(i) Markov Chain from TPM

Consider the transition probability matrix,

$$\mathbf{W} = \begin{bmatrix} 0.66 & 0.18 & 0.44 \\ 0.26 & 0.79 & 0.23 \\ 0.08 & 0.03 & 0.33 \end{bmatrix}.$$

- Write an algorithm to simulate a Markov chain starting from a given initial state $x^{(1)}$ (python) over N cycles. The output is an array of N elements such as $[2, 2, 3, \cdots, 1]$.
- Implement the algorithm above starting from $x^{(1)} = 2$ (python) for $N = 100$, and report the sequence.
- Repeat the exercise above for $N = 10000$, but do not report the sequence. Instead consider the last half of the chain (states $x^{(N/2)}$ through $x^{(N)}$), and find the fraction of R, python, and other programmers. How do these fractions compare with the stationary distribution $\boldsymbol{\pi}^{(\infty)} = [0.3812, 0.5487, 0.0701]^T$?

(ii) 2D Random Walk

A random walk is an example of a Markov process. A constant step-size random walk of N steps in 2D can be implemented as follows:

- Let $\mathbf{r}_0 = (0, 0)$ be the starting position
- For steps i = 1 to N
  - pick $\theta \sim U[0, 2\pi]$
  - $\mathbf{r}_i = \mathbf{r}_{i-1} + (\cos\theta, \sin\theta)$
  - $i = i + 1$

(a) Implement this algorithm, and visualize 5 different random walks with $N = 1000$.

(b) Vary N between 10^1 and 10^4. At each N perform 100 replicas and find the average end-to-end distance $R^2 = \langle \mathbf{r}_N \cdot \mathbf{r}_N \rangle$. Plot R^2 as a function of N on a log-log plot. Show error bars.

(iii) Ladder Climbing Example

Consider a game of "ladder climbing".[10] There are 5 levels (states) in the game, level 1 is the lowest (bottom) and level 5 is the highest (top).

A player starts at the bottom (initial state $x = 1$). Each time, a fair coin is tossed. If it turns up heads, the player moves up one rung. If tails, the player moves down to the very bottom. Once at the top level, the player moves to the very bottom if a tail turns up, and stays at the top if head turns up.

- Find the transition probability matrix, $\mathbf{W}$.
- Find the eigenvalues and eigenvectors of $\mathbf{W}$.
- What is the steady-state distribution $\pi(x)$ of the Markov chain?
- What is the expected value $E(x) = \sum x\pi(x)$?

A Appendices

A.1 Eigenvalues of Transition Probability Matrices

Theorem

A transition probability or Markov matrix $\mathbf{A}$ always has an eigenvalue 1. All other eigenvalues $|\lambda_i| \leq 1$.

Proof

Let a_{ij} be the element of $\mathbf{A}$ in row i and column j. Each column sums to 1.

$$\sum_i a_{ij} = 1 \tag{11}$$

[10] Example adapted from https://www.utdallas.edu/~mbaron/

An equivalent way of saying this is: the sum of the elements of each row of $\mathbf{A}^T$ adds up to 1. We can write this property as a matrix equation,

$$\mathbf{A}^T \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{12}$$

Therefore, 1 is an eigenvalue of $\mathbf{A}^T$, with eigenvector $[1, 1, \cdots, 1]^T$.

The determinant/characteristic polynomials of $\mathbf{A}$ and $\mathbf{A}^T$ are identical,

$$|\mathbf{A}^T - \lambda \mathbf{I}| = |\mathbf{A} - \lambda \mathbf{I}| \tag{13}$$

So they share the same eigenvalues. Thus, 1 is also an eigenvalue of $\mathbf{A}$.

Let us now show that all other eigenvalues $|\lambda_i| \leq 1$.

Suppose one of the other eigenvalues $|\lambda| > 1$, and $\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$.

Since rows of $\mathbf{A}^T$ are non-negative and add up to 1, each element of the vector $\mathbf{A}\mathbf{v}$ is less than the maximum component $v_{\max}$ of $\mathbf{v}$. However, if $\lambda > 1$, then for the row containing $v_{\max}$, $\lambda v_{\max} > v_{\max}$. This is a contradiction. Thus, $|\lambda| \leq 1$.

A.2 Python Programs

A.2.1 TPM Example

We first encode the matrix,

$$\mathbf{W} = \begin{bmatrix} 0.66 & 0.18 & 0.44 \\ 0.26 & 0.79 & 0.23 \\ 0.08 & 0.03 & 0.33 \end{bmatrix}$$

```
W = np.matrix([[0.66, 0.18, 0.44],
               [0.26, 0.79, 0.23],
               [0.08, 0.03, 0.33]])
```

Let's find the eigenvalues and eigenvectors of $\mathbf{W}$ directly. We can use the `eig` command from `numpy.linalg`.

```
lam, eigv = np.linalg.eig(W)
print(eigv)
[ 1.          0.5353  0.2447]

print(eigv)
[[-0.5674 -0.6305 -0.7397]
 [-0.8168  0.7645  0.0704]
 [-0.1043 -0.1340  0.6693]]
```

The eigenvalues are $(1, 0.5353, 0.2447)$. The first column of the matrix is the eigenvector that corresponds to the eigenvalue of 1. Eigenvectors are only known to a multiplicative constant, so lets re-normalize it - so that the elements add up to 1.

```
ev1 = eigv[:,0] # pick out the first column of the matrix
ev1 = np.abs(ev1)/np.linalg.norm(ev1,1)  # 1 norm is the absolute sum
print(ev1)
[[ 0.3812]
 [ 0.5487]
 [ 0.0701]]
```

Compare this with

$$\boldsymbol{\pi}^{(\infty)} = \begin{bmatrix} 0.3812 \\ 0.5487 \\ 0.0701 \end{bmatrix}.$$