STAT 201A - Introduction to Probability at an advanced level Problem Set

Fall 2023, UC Berkeley

Isabel Moreno

2023-11-21

1. Simulation of Markov Process.

a. Write the Markov process in matrix representation, that is, define the matrix $P \in \mathbb{R}^3 \times 3$ such that P_ij is the probability of transitioning from the node i to j.

$$P = \begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.2 & 0.5 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{bmatrix}$$

P_{ij} represents the probability of transitioning from state i to state j, and each row of the matrix sums to 1, as required in a Markov transition probability matrix.

b. Simulate one single realization of the chain, that is, starting from $X_0 = 1$, update the value of X_i using the probabilities defined by the process.

```
import numpy as np

def simulate_markov_chain(initial_state, transition_matrix, num_steps):
    current_state = initial_state
    states = [current_state]

    for _ in range(num_steps):
        probabilities = transition_matrix[current_state - 1, :]
        next_state = np.random.choice(
          np.arange(1, len(probabilities) + 1), p=probabilities)
        states.append(next_state)
```

```
        current_state = next_state

    return states

# Given transition matrix P
transition_matrix = np.array([[0.2, 0.7, 0.1],
                              [0.2, 0.5, 0.3],
                              [0.2, 0.4, 0.4]])

# Initial state
initial_state = 1

# Number of steps to simulate
num_steps = 20

# Simulate the Markov chain
result = simulate_markov_chain(initial_state, transition_matrix, num_steps)

# Print the result
print("Simulated Markov Chain:", result)
```

```
Simulated Markov Chain: [1, 2, 3, 3, 1, 2, 3, 1, 2, 2, 2, 3, 2, 2, 2, 3, 1, 2, 2, 3, 2]
```

2. Stationary Distribution. The goal of this section is to show the convergence of the probability distribution of the Markov process.

a. Calculate theoretically the stationary state of the process by finding the vector $\pi_\infty \in \mathbb{R}^3$ such that $\pi_\infty^T = \pi_\infty^T P$. Notice that this is the same as finding the eigenvector with eigenvalues equals one of the matrix P^T . This is the same as solving $(P^T - I)\pi_\infty = 0$. You can solve the linear system of equation numerically or analytically.

$$P = \begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.2 & 0.5 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{bmatrix}$$

We then transponse the matrix:

$$P^T = \begin{bmatrix} 0.2 & 0.2 & 0.2 \\ 0.7 & 0.5 & 0.4 \\ 0.1 & 0.3 & 0.4 \end{bmatrix}$$

$$(P^T - I)\pi_\infty = 0;$$

$$(P^T - I)\pi_\infty = \begin{bmatrix} -0.8 & 0.2 & 0.2 \\ 0.7 & -0.5 & 0.4 \\ 0.1 & 0.3 & -0.6 \end{bmatrix} \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

To solve this equation, we solve π_∞ numerically:

$$\begin{aligned} -0.8\pi_1 + 0.2\pi_2 + 0.2\pi_3 &= 0 \\ 0.7\pi_1 - 0.5\pi_2 + 0.4\pi_3 &= 0 \\ 0.1\pi_1 + 0.3\pi_2 - 0.6\pi_3 &= 0 \end{aligned}$$

Using gaussian elimination, we get that equation 2 and 3 are proportials:

$$\begin{aligned} -0.8\pi_1 + 0.2\pi_2 + 0.2\pi_3 &= 0 \\ 0.7\pi_1 - 0.5\pi_2 + 0.4\pi_3 = 0 \rightarrow (-2E_1) &\rightarrow 2.3\pi_1 - 0.9\pi_2 = 0 \\ 0.1\pi_1 + 0.3\pi_2 - 0.6\pi_3 = 0 \rightarrow (+3E_1) &\rightarrow -2.3\pi_1 + 0.9\pi_2 = 0 \end{aligned}$$

Then, we have a system of 2 equations and three variables, where we have infinite solutions:

$$\begin{aligned} -0.8\pi_1 + 0.2\pi_2 + 0.2\pi_3 &= 0 \\ 2.3\pi_1 - 0.9\pi_2 &= 0 \end{aligned}$$

We set then $\pi_1 = \lambda$ and solve for π_2 and π_3:

$$\begin{aligned} 2.3\lambda - 0.9\pi_2 &= 0 \\ \pi_2 &= \frac{2.3\lambda}{0.9} \end{aligned}$$

$$\begin{aligned} -0.8\lambda + 0.2\frac{2.3\lambda}{0.9} + 0.2\pi_3 &= 0 \\ -4\lambda + \frac{2.3\lambda}{0.9} + \pi_3 &= 0 \\ \frac{-1.3\lambda}{0.9} + \pi_3 &= 0 \\ \pi_3 &= \frac{1.3\lambda}{0.9} \end{aligned}$$

$$\pi_\infty = \begin{bmatrix} \lambda \\ \frac{2.3\lambda}{0.9} \\ \frac{1.3\lambda}{0.9} \end{bmatrix}$$

As we know that the π_∞ matrix must sum 1, then we solve for λ:

$$\begin{aligned} \lambda + \frac{2.3\lambda}{0.9} + \frac{1.3\lambda}{0.9} &= 1 \\ \frac{0.9\lambda}{0.9} + \frac{2.3\lambda}{0.9} + \frac{1.3\lambda}{0.9} &= 1 \\ \lambda &= \frac{0.9}{4.5} = 0.2 \end{aligned}$$

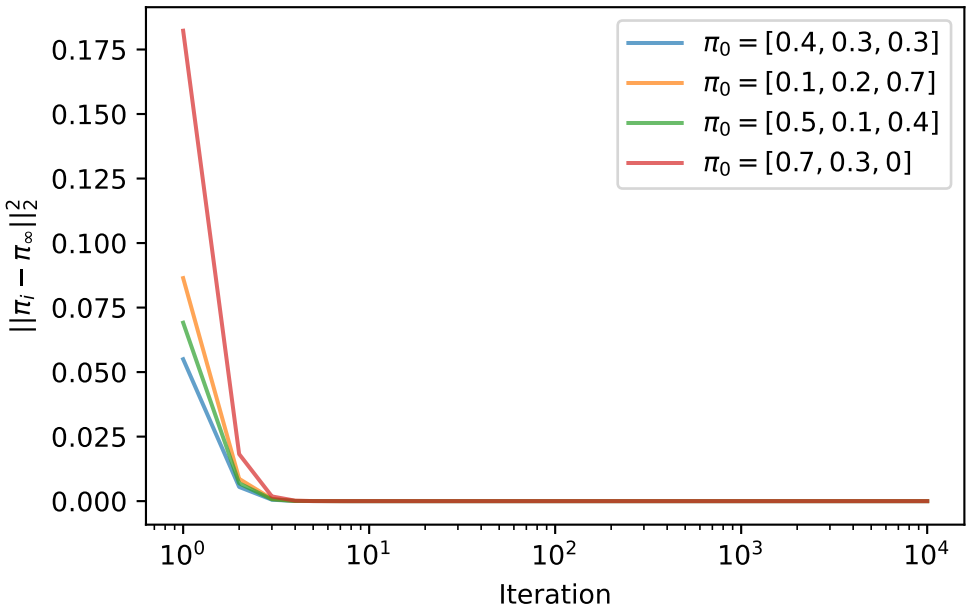
Then, the stationary distribution π_∞ is:

$$\pi_\infty \approx \begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix}$$

b. Starting now from an initial probability distribution π_0 on the nodes, compute the value of $\pi_i^T = \pi_0^T p_i$ the probability distribution at time i. Show that $\pi_i \to \pi_\infty$ and make plot of i vs $||\pi_i - \pi_\infty||_2^2$. Generate this plot for at least two different initial conditions π_0 and compare.

```python
import numpy as np
import matplotlib.pyplot as plt

# Given transition matrix
P = np.array([[0.2, 0.7, 0.1],
              [0.2, 0.5, 0.3],
              [0.2, 0.4, 0.4]])

# Stationary distribution
pi_infinity = np.array([0.2, 0.2*2.3/0.9, 1.3*0.2/0.9])

# Initial conditions (Examples)
pi_0_1 = np.array([0.4, 0.3, 0.3])
pi_0_2 = np.array([0.1, 0.2, 0.7])
pi_0_3 = np.array([0.5, 0.1, 0.4])
pi_0_4 = np.array([0.7, 0.3, 0])
```

```
# Number of iterations
num_iterations = 10000

# Arrays to store L2 norm values for each iteration
l2_norm_1 = np.zeros(num_iterations)
l2_norm_2 = np.zeros(num_iterations)
l2_norm_3 = np.zeros(num_iterations)
l2_norm_4 = np.zeros(num_iterations)

# Calculate pi_i and L2 norm for each iteration
for i in range(num_iterations):
    pi_i_1 = np.dot(pi_0_1, np.linalg.matrix_power(P, i + 1))
    pi_i_2 = np.dot(pi_0_2, np.linalg.matrix_power(P, i + 1))
    pi_i_3 = np.dot(pi_0_3, np.linalg.matrix_power(P, i + 1))
    pi_i_4 = np.dot(pi_0_4, np.linalg.matrix_power(P, i + 1))

    # Calculate L2 norm
    l2_norm_1[i] = np.linalg.norm(pi_i_1 - pi_infinity)
    l2_norm_2[i] = np.linalg.norm(pi_i_2 - pi_infinity)
    l2_norm_3[i] = np.linalg.norm(pi_i_3 - pi_infinity)
    l2_norm_4[i] = np.linalg.norm(pi_i_4 - pi_infinity)

# Plotting
plt.plot(range(1, num_iterations + 1), l2_norm_1,
label=r'$\pi_0 = [0.4, 0.3, 0.3]$', alpha=0.7)
plt.plot(range(1, num_iterations + 1), l2_norm_2,
label=r'$\pi_0 = [0.1, 0.2, 0.7]$', alpha=0.7)
plt.plot(range(1, num_iterations + 1), l2_norm_3,
label=r'$\pi_0 = [0.5, 0.1, 0.4]$', alpha=0.7)
plt.plot(range(1, num_iterations + 1), l2_norm_4,
label=r'$\pi_0 = [0.7, 0.3, 0]$', alpha=0.7)
plt.xscale('log')
plt.xlabel('Iteration')
plt.ylabel(r'$||\pi_i - \pi_\infty||_2^2$')
plt.legend()
plt.show()
```



All lines in the plot are similar and differ only in their starting values along the y-axis, showing that the Markov chain is converging to the stationary distribution π_∞ regardless of the initial conditions. Therefore, as the number of iterations increases, the probability distribution π_i for different initial conditions tends to approach the stationary distribution π_∞ .

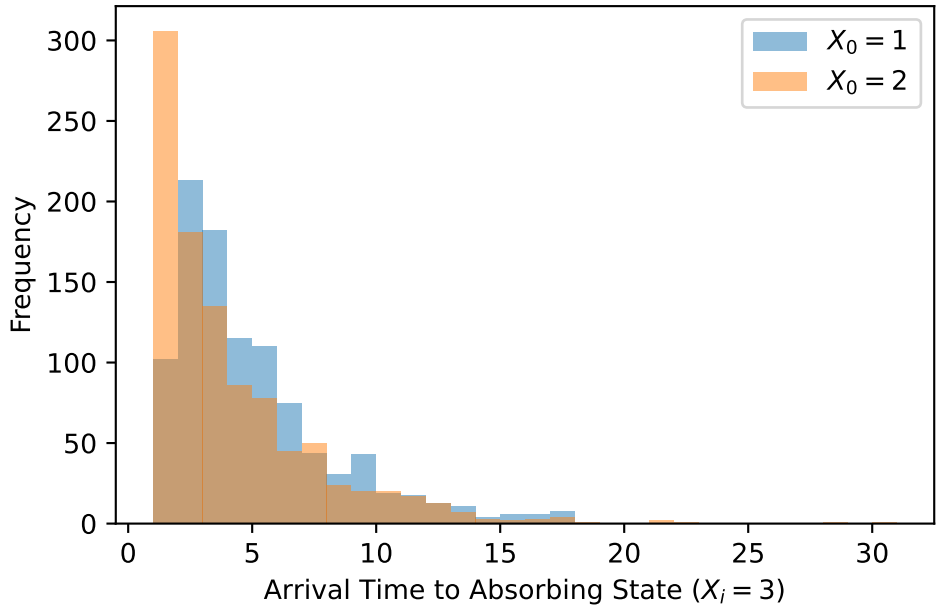
3. Absorbing state. Consider now that node 3 is an absorbing state and we want to estimate the waiting time until the process arrives at $X_i = 3$ from any other node.

a. Starting from each one of $X_0 = 1$ and $X_0 = 2$, run multiple simulation of the Markov chain (Problem 1, part b) until $X_i = 3$ and store the arrival time untilthis happens. Make a histogram of the arrival time for both $X_0 = 1$ and $X_0 = 2$ and compute the mean.

```python
import numpy as np
import matplotlib.pyplot as plt

def simulate_markov_chain_until_absorbing(initial_state, transition_matrix):
    current_state = initial_state
    time_steps = 0

    while current_state != 3:
        probabilities = transition_matrix[current_state - 1, :]
        next_state = np.random.choice(
```

```
            np.arange(1, len(probabilities) + 1), p=probabilities)
        current_state = next_state
        time_steps += 1

    return time_steps

# Given transition matrix P
transition_matrix = np.array([[0.2, 0.7, 0.1],
                              [0.2, 0.5, 0.3],
                              [0.2, 0.4, 0.4]])

# Initial states
initial_state_1 = 1
initial_state_2 = 2

# Number of simulations
num_simulations = 1000

# Simulate and record arrival times
arrival_times_1 = [simulate_markov_chain_until_absorbing(
  initial_state_1, transition_matrix
  )
for _ in range(num_simulations)]
arrival_times_2 = [simulate_markov_chain_until_absorbing(
  initial_state_2, transition_matrix
  )
for _ in range(num_simulations)]

# Plot histograms
plt.hist(arrival_times_1, bins=range(min(arrival_times_1), max(arrival_times_1) + 1),
 alpha=0.5, label=r'$X_0 = 1$')
plt.hist(arrival_times_2, bins=range(min(arrival_times_2), max(arrival_times_2) + 1),
 alpha=0.5, label=r'$X_0 = 2$')

plt.xlabel('Arrival Time to Absorbing State ($X_i = 3$)')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# Compute and print mean arrival times
mean_arrival_time_1 = np.mean(arrival_times_1)
```

```
mean_arrival_time_2 = np.mean(arrival_times_2)

print("Mean Arrival Time (X_0 = 1):", mean_arrival_time_1)
print("Mean Arrival Time (X_0 = 2):", mean_arrival_time_2)
```




```
Mean Arrival Time (X_0 = 1): 4.546
Mean Arrival Time (X_0 = 2): 3.762
```

b. Compute theoretically the mean arrival time to the absorbing state and compare it with part a. To do so, notice that if T_i denotes the random variable associated to the arrival time starting from $X_0 = i$, then:

$$\mu_i = 1 + \sum_{j=1}^{3} p_{ij}\mu_j,$$

with $\mu_i = \mathbb{E}[T_i]$ This is a linear system of equations that you can solve. Notice $T_3 = 0$.

$$\mu_i = 1 + \sum_{j=1}^{3} p_{ij}\mu_j,$$

This equation represents a system of linear equations for the mean arrival time μ_i starting from state $X_0 = i$, where $T_3 = 0$ since state 3 is an absorbing state.

Then for the 3 states (i=1,2,3) the system of equations is as follows:

$$\begin{aligned} \mu_1 &= 1 + p_{11}\mu_1 + p_{12}\mu_2 + p_{13}\mu_3 \\ \mu_2 &= 1 + p_{21}\mu_1 + p_{22}\mu_2 + p_{23}\mu_3 \\ \mu_3 &= 0 \end{aligned}$$

Substituting the values for the transition matrix P:

$$\begin{aligned} \mu_1 &= 1 + 0.2\mu_1 + 0.7\mu_2 + 0.1\mu_3 \\ \mu_2 &= 1 + 0.2\mu_1 + 0.5\mu_2 + 0.3\mu_3 \\ \mu_3 &= 0 \end{aligned}$$

Then, we solve the following system in order to find μ_1 and μ_2:

$$\begin{aligned} \mu_1 &= 1 + 0.2\mu_1 + 0.7\mu_2 \\ \mu_2 &= 1 + 0.2\mu_1 + 0.5\mu_2 \end{aligned}$$

$$\begin{aligned} 0.8\mu_1 - 0.7\mu_2 &= 1 \\ -0.2\mu_1 + 0.5\mu_2 = 1 \rightarrow (+4E_1) &\rightarrow 1.3\mu_2 = 5 \\ \mu_2 &= 3.846 \\ 0.8\mu_1 - 0.7 * 3.769 &= 1; \\ \mu_1 &= 4.548 \end{aligned}$$

Theoretically the mean arrival time to the absorbing state are $\mu_1 = 4.548$ and $\mu_2 = 3.846$ that are pretty closed to the simulated means which underscores the accuracy of the estimates for the waiting time until the process reaches the absorbing state.