Lecture 6 – Part 2
Monte Carlo Simulation

Instructor: Assistant Lecturer Hadi Salah Hadi

1 Review: Idea of Monte Carlo Simulation

Monte Carlo simulation is a numerical method for approximating probabilities and expectations using random sampling.

1. Define the probabilistic model of the system or experiment.
2. Generate a large number of independent trials (simulated experiments).
3. For each trial, check whether the event of interest occurs.
4. Estimate the probability

$$\hat{p} = \frac{\text{Number of successes}}{\text{Total number of trials } N}.$$

As N increases, the estimate $\hat{p}$ converges to the true probability by the Law of Large Numbers.

2 Example 1: Sum of Two Dice Greater than 8

Problem

Two fair dice are thrown. Let

$$S = X_1 + X_2.$$

Find the probability that the sum is greater than 8:

$$P(S > 8).$$

Classical Solution

- Total number of outcomes: $6 \times 6 = 36$.
- Favourable outcomes with sum > 8:
  $(3,6), (4,5), (4,6), (5,4), (5,5), (5,6), (6,3), (6,4), (6,5), (6,6)$.
  Number of favourable outcomes is 10.

Thus,

$$P(S > 8) = \frac{10}{36} \approx 0.2778.$$

Monte Carlo Algorithm

1. Choose a large number of trials, e.g. $N = 100{,}000$.
2. For each trial, generate two random integers between 1 and 6.
3. Compute their sum and check whether it is greater than 8.
4. Let L be the number of trials with $S > 8$.
5. Estimate $\hat{p} = L/N$.

Python Code

Code

```
import numpy as np

N = 100_000

# Generate N pairs of random integers between 1 and 6
x = np.random.randint(1, 7, size=(2, N))   # shape: (2, N)

# Sum of each trial
sums = x.sum(axis=0)

# Count number of sums greater than 8
L = np.sum(sums > 8)

# Monte Carlo estimate
p_hat = L / N
print("Estimated P(S > 8) =", p_hat)
```

Sample Output (will vary slightly):

```
Estimated P(S > 8) = 0.27851
```

3 Example 2: At Least One 5 Appears

Problem

Two fair dice are thrown. Find the probability that at least one die shows 5.

Classical Solution

Let A be the event "at least one 5 appears". Consider the complement:

- A^c: "no 5 appears".
- Probability that a single die is not 5 is 5/6.

- For two independent dice,
$$P(A^c) = \left(\frac{5}{6}\right)^2 = \frac{25}{36}.$$
- Hence
$$P(A) = 1 - P(A^c) = 1 - \frac{25}{36} = \frac{11}{36} \approx 0.3056.$$

Monte Carlo Algorithm

1. Choose $N = 200{,}000$.
2. For each trial, generate two random integers between 1 and 6.
3. Check if at least one of them equals 5.
4. Let s be the number of successful trials.
5. Estimate $P(A) \approx s/N$.

Python Code (Loop Version)

Code

```
import numpy as np

N = 200_000
x = np.random.randint(1, 7, size=(2, N))

s = 0  # counter of successes
for k in range(N):
    if x[0, k] == 5 or x[1, k] == 5:
        s += 1

p_hat = s / N
print("Estimated P(at least one 5) =", p_hat)
```

Sample Output:

```
Estimated P(at least one 5) = 0.30489
```

Python Code (Vectorized Version)

Code

```
# Logical matrix: True where the die equals 5
y = (x == 5)

# For each trial j, c[j] is True if at least one die is 5
c = np.logical_or(y[0, :], y[1, :])

sm = np.sum(c)
p_hat_vec = sm / N
print("Vectorized estimate =", p_hat_vec)
```

Sample Output:

```
Vectorized estimate = 0.30512
```

4 Example 3: Conditional Probability with Dice

Problem

Two fair dice are thrown. What is the probability that their sum is more than 8, given that at least one die shows 6?

Let:

- A: "at least one die is 6",
- B: "sum $S > 8$".

We want $P(B \mid A)$.

Classical Solution

By the definition of conditional probability,

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}.$$

By counting outcomes:

- $P(A) = 11/36$ (all pairs with at least one 6),
- $P(A \cap B) = 7/36$ (pairs with at least one 6 and sum > 8).

Therefore,

$$P(B \mid A) = \frac{7/36}{11/36} = \frac{7}{11} \approx 0.6364.$$

Monte Carlo Algorithm

1. Generate N pairs of dice outcomes.
2. Event A: create a mask that is true when at least one die equals 6.
3. Select only those trials where A occurs.
4. Among these selected trials, count how many also satisfy B: sum > 8.
5. If the number of selected trials is n and the number of successes is s, estimate

$$\widehat{P(B \mid A)} = \frac{s}{n}.$$

Python Code

Code

```
import numpy as np

N = 100_000
x = np.random.randint(1, 7, size=(2, N))

# Event A: at least one die equals 6
has_six = np.logical_or(x[0, :] == 6, x[1, :] == 6)

# Select only trials where A occurs
y = x[:, has_six]      # shape: (2, n)
n = y.shape[1]         # number of selected trials

# Event B within the selected trials: sum > 8
sums_y = y.sum(axis=0)
success = np.sum(sums_y > 8)

p_cond = success / n
print("Estimated P(S > 8 | at least one 6) =", p_cond)
```

Sample Output:

```
Estimated P(S > 8 | at least one 6) = 0.63742
```

5 Characteristics of Monte Carlo Simulation in Medicine

Monte Carlo methods are very important in medical and health applications.

Main Characteristics

- **Handles Uncertainty:** Patient parameters (age, weight, organ function, etc.) are not fixed; Monte Carlo allows sampling these quantities from probability distributions.
- **Complex Systems:** Many medical systems are too complex for closed-form solutions (radiation transport, pharmacokinetics, patient flow). Monte Carlo can simulate these systems numerically.
- **Full Distribution of Outcomes:** Instead of giving a single number, Monte Carlo provides a distribution (e.g. distribution of dose, waiting times, or blood pressure after treatment).
- **What-if Analysis:** Easy to change input parameters and repeat the simulation for different scenarios.
- **Decision Support:** Helps physicians and planners to analyze risks, compare treatment strategies, and allocate resources.

6 Medical Example: Drug Effect on Blood Pressure

Model

Consider a simple model for the effect of a drug on systolic blood pressure.

- Initial systolic blood pressure:

$$\mathrm{BP}_0 = 150 \text{ mmHg}.$$

- The reduction in blood pressure due to the drug is a random variable

$$R \sim \mathcal{N}(\mu = 15,\ \sigma = 8) \text{ mmHg},$$

  where $\mu = 15$ is the mean reduction and $\sigma = 8$ is the standard deviation.

- Final blood pressure after treatment:

$$\mathrm{BP}_{\text{after}} = \mathrm{BP}_0 - R.$$

Goal: Estimate

$$P(110 \leq \mathrm{BP}_{\text{after}} \leq 130),$$

the probability that the final blood pressure lies in the normal range $[110, 130]$ mmHg.

Monte Carlo Algorithm

1. Choose $N = 100{,}000$ simulated patients.
2. For each patient:
   - Generate a random response R_i from $\mathcal{N}(15, 8^2)$.
   - Compute $\mathrm{BP}_i = 150 - R_i$.
   - Check whether $110 \leq \mathrm{BP}_i \leq 130$.
3. Let L be the number of simulated patients with blood pressure in the normal range.
4. Estimate $\hat{p} = L/N$.

Python Code

Code

```
import numpy as np

N     = 100_000         # number of simulated patients
BP0   = 150.0           # initial systolic blood pressure (mmHg)
mu    = 15.0            # mean decrease (mmHg)
sigma = 8.0             # standard deviation of decrease

# Generate N random responses: R ~ Normal(mu, sigma^2)
R = np.random.normal(loc=mu, scale=sigma, size=N)
```

```
# Final blood pressure after treatment
BP_after = BP0 - R

# Check normal range: 110 <= BP_after <= 130
normal_mask = (BP_after >= 110) & (BP_after <= 130)
L = np.sum(normal_mask)

# Monte Carlo estimate
p_normal = L / N
print("Estimated probability of normal BP =", p_normal)
```

Sample Output:

```
Estimated probability of normal BP = 0.62137
```