STA Ass6 12310401 王子恒

1\. (1) Assume X = number of heads in n tosses

$Y = n - X$

$Var(X) = np(1-p) = \frac{1}{4}n$

$Var(Y) = Var(X) = \frac{1}{4}n$

$Var(X-Y) = Var(2X-n) = 2^2 \times \frac{1}{4}n = n$

Thus the function is $f(n) = n$

(2) No. Because as the n get bigger, the Variance of $X-Y$ is getting bigger. which opposite the "difference approaches 0".
Actually, the difference will fluctuated arround 0.

2\. (1) $X_i \sim$ Poission (3)

$E(X) = 3$

According to the LLN

Y_n converges to $E(X) = 3$

(2) $X_i \sim U(-1,3)$

$E(X) = \frac{3-1}{2} = 1$

Y_n converges to 1

(3) $X_i \sim Exp(5)$

$E(X) = \frac{1}{5}$

Y_n converges to $\frac{1}{5}$

3. Assume $R=\sqrt{-2\ln(U_1)}$ $\quad \theta = 2\pi U_2$

then $Z_1 = R\cos\theta,$ $\quad Z_2 = R\sin\theta$

$R=\sqrt{Z_1^2+Z_2^2}$

$\Rightarrow U_1 = \exp\left(-\frac{1}{2}(z_1^2+z_2^2)\right)$ $\quad U_2 =$

$$\iint f_{Z_1,Z_2}(z_1,z_2) = \iint f_{U_1,U_2}(u_1(z_1,z_2), u_2(z_1,z_2))|J|$$

$$J = \begin{vmatrix} \frac{\partial z_1}{\partial u_1} & \frac{\partial z_1}{\partial u_2} \\ \frac{\partial z_2}{\partial u_1} & \frac{\partial z_2}{\partial u_2} \end{vmatrix} = \begin{vmatrix} \frac{\cos(2\pi U_2)}{\sqrt{-2\ln(U_1)}} \cdot \frac{1}{U_1} & -\sqrt{-2\ln(U_1)}\sin(2\pi U_2)\cdot 2\pi \\ -\frac{\sin(2\pi U_2)}{\sqrt{-2\ln(U_1)}} \cdot \frac{1}{U_1} & \sqrt{-2\ln U_1}\cos(2\pi U_2)\cdot 2\pi \end{vmatrix} = \frac{2\pi}{U_1}$$

$\because f_{U_1,U_2}(u_1,u_2) = 1$

$\therefore f_{Z_1,Z_2}(z_1,z_2) = \frac{U_1}{2\pi} = \frac{1}{2\pi}\cdot \exp\left(-\frac{1}{2}(z_1^2+z_2^2)\right)$

$f_{Z_1}(z_1) = \int_{-\infty}^{\infty} f_{Z_1,Z_2}(z_1,z_2)\,dz_2 = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{z_1^2}{2}\right) \sim N(0,1)$

$f_{Z_2}(z_2) = \int_{-\infty}^{\infty} f_{Z_1Z_2}(z_1,z_2)\,dz_1 = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{z_2^2}{2}\right). \sim N(0,1)$

thus $f_{Z_1,Z_2}(z_1,z_2) = f_{Z_1}(z_1)\cdot f_{Z_2}(z_2)$.

Hence, Z_1, Z_2 are a pair of independent standard normal random variables.

4. (1) $X \sim$ Geometric (p)

$P_k = P(X=k) = p(1-p)^{k-1}$ $\quad k=1,2,3\cdots$

Divide the $(0,1)$ as follows:

$I_1 = (0, p_1),\ I_2 = [p_1, p_1+p_2),\ I_3 = [p_1+p_2, p_1+p_2+p_3)$

Define $x_i = k$ if $u_i \in I_k$, then $x_1, x_2 \cdots x_n$ can be considered numbers generated from Geometric (p)

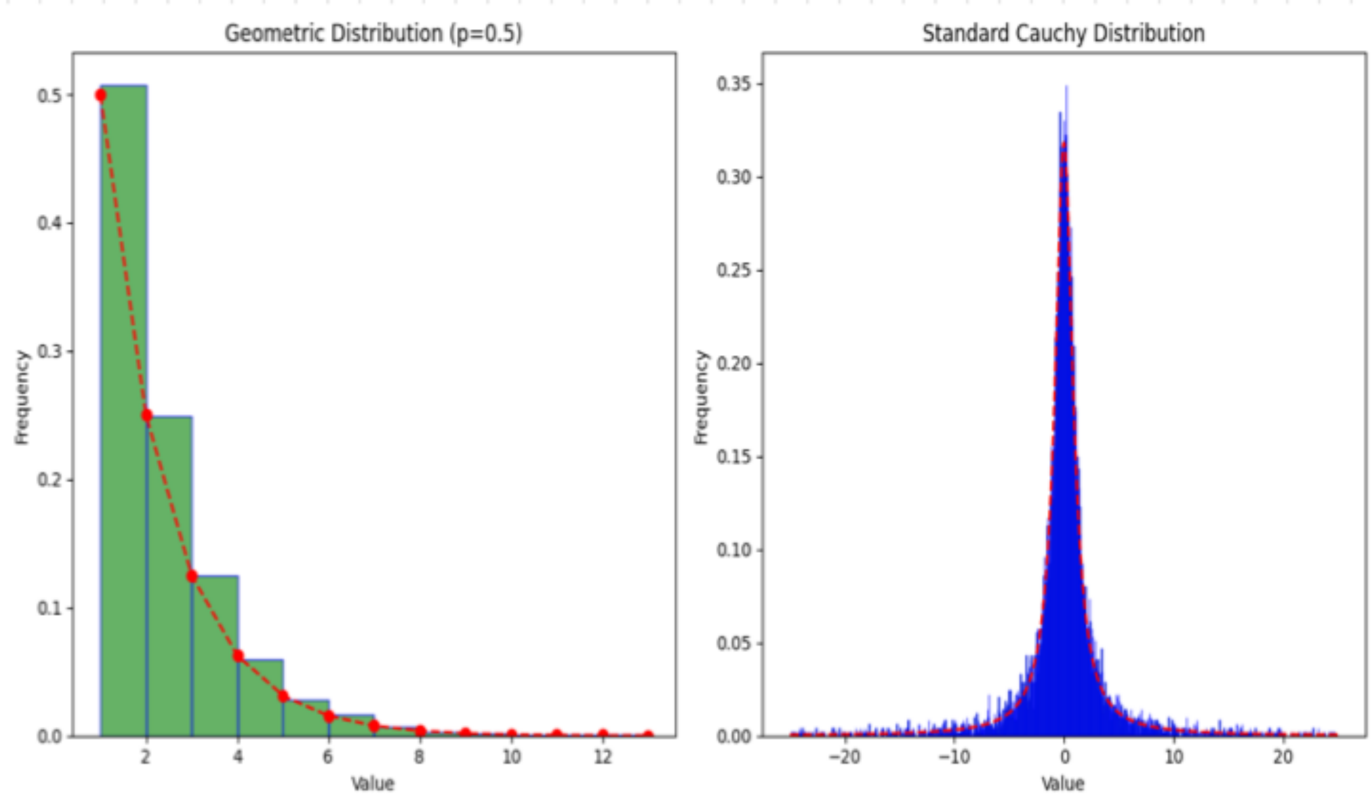
(2) $F(x) = \int_{-\infty}^{x} f(t)\,dt = \frac{1}{\pi}\left(\arctan x - \frac{1}{2}\right).$

$x = F^{-1}(u) = \tan\left(\pi u + \frac{\pi}{2}\right).$

For u_i from uniform distribution, $x_i = \tan\left(\pi u + \frac{\pi}{2}\right)$ can be generated from standard Cauchy distribution.

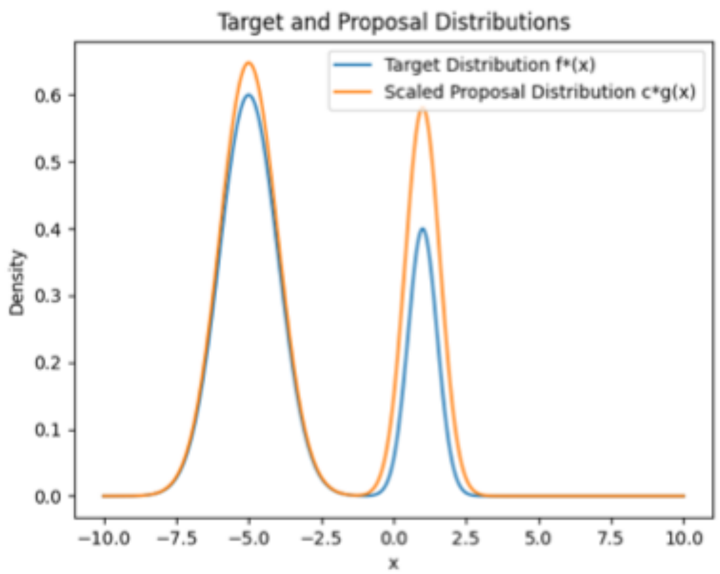
5、

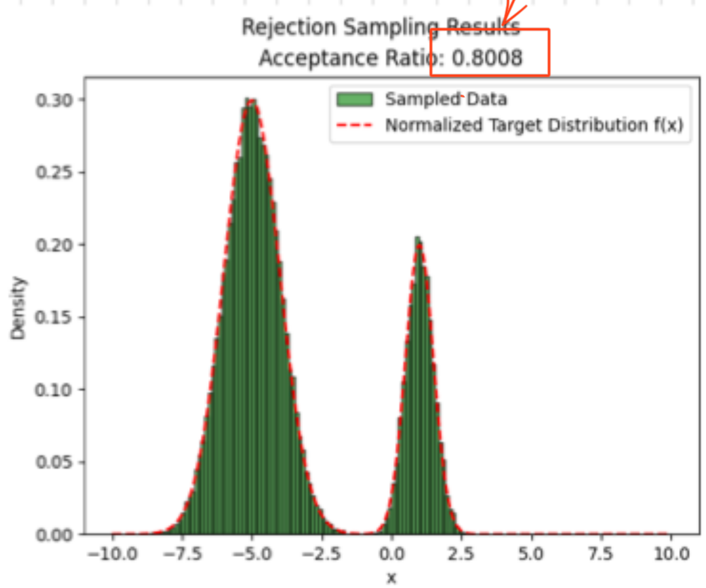
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import geom, cauchy

p = 0.5
geom_data = np.random.geometric(p, 10000)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(geom_data, bins=range(1, max(geom_data)+1), density=True, alpha=0.6, color='g', edgecolor='blue')

x = np.arange(1, max(geom_data)+1)
pmf = geom.pmf(x, p)
plt.plot(x, pmf, 'r', marker='o', linestyle='--')

plt.title('Geometric Distribution (p=0.5)')
plt.xlabel('Value')
plt.ylabel('Frequency')

cauchy_data = np.random.standard_cauchy(10000)

plt.subplot(1, 2, 2)
plt.hist(cauchy_data, bins=1000, density=True, alpha=0.6, color='g', edgecolor='blue', range=(-25, 25))

x = np.linspace(-25, 25, 1000)
pdf = cauchy.pdf(x)
plt.plot(x, pdf, 'r', linestyle='--')

plt.title('Standard Cauchy Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



b.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm


def target_distribution(x):
    return 0.6 * np.exp(-(x + 5)**2 / 2) + 0.4 * np.exp(-(x - 1)**2 / 0.5)


class MixtureNormal:
    def __init__(self, weights, means, stds):
        self.weights = weights
        self.means = means
        self.stds = stds

    def pdf(self, x):
        return sum(w * norm.pdf(x, m, s) for w, m, s in zip(self.weights, self.means, self.stds))

    def rvs(self, size=1):
        component = np.random.choice(len(self.weights), size=size, p=self.weights)
        return np.random.normal(self.means[component[0]], self.stds[component[0]], size=size)

weights = [0.65, 0.35]
means = [-5, 1]
stds = [1,0.6]
proposal_distribution = MixtureNormal(weights, means, stds)

c = 2.5

x = np.linspace(-10, 10, 1000)
plt.plot(x, target_distribution(x), label='Target Distribution f*(x)')
plt.plot(x, c * proposal_distribution.pdf(x), label='Scaled Proposal Distribution c*g(x)')
plt.legend()
plt.title('Target and Proposal Distributions')
plt.xlabel('x')
plt.ylabel('Density')
plt.show()


def rejection_sampling(target, proposal, c, num_samples):
    samples = []
    count = 0

    while len(samples) < num_samples:
        x = proposal.rvs()
        u = np.random.uniform(0, c * proposal.pdf(x))
        if u < target(x):
            samples.append(x)
        count += 1
    return np.array(samples), count


num_samples = 50000

samples, total_count = rejection_sampling(target_distribution, proposal_distribution, c, num_samples)

acceptance_ratio = num_samples / total_count
print(f'Acceptance Ratio: {acceptance_ratio:.4f}')

plt.hist(samples, bins=100, density=True, alpha=0.6, color='g', edgecolor='black', label='Sampled Data')

normalized_target = target_distribution(x) / np.trapz(target_distribution(x), x)
plt.plot(x, normalized_target, 'r', linestyle='--', label='Normalized Target Distribution f(x)')

plt.legend()
plt.suptitle('Rejection Sampling Results')
plt.title('Acceptance Ratio: {:.4f}'.format(acceptance_ratio))
plt.xlabel('x')
plt.ylabel('Density')
plt.show()
```





7. $Z_{\frac{\alpha}{2}} = 1.96$

$$n = \left(\frac{1.96 \times 0.5}{0.005}\right)^2 = 38416$$

```python
import numpy as np

rows, cols = 20, 50
total_trees = rows * cols

prob_left = 0.8
prob_above = 0.3

num_simulations = 38416

def simulate_fire():
    forest = np.zeros((rows, cols), dtype=bool)
    forest[0, 0] = True

    for i in range(rows):
        for j in range(cols):
            if i > 0 and forest[i-1, j] and np.random.rand() < prob_above:
                forest[i, j] = True
            if j > 0 and forest[i, j-1] and np.random.rand() < prob_left:
                forest[i, j] = True

    return np.sum(forest)

burned_trees = np.array([simulate_fire() for _ in range(num_simulations)])

threshold = 0.3 * total_trees
probability = np.mean(burned_trees > threshold)

mean_burned_trees = np.mean(burned_trees)
std_burned_trees = np.std(burned_trees)

print(f"超过30%树木燃烧的概率: {probability:.5f}")
print(f"受影响的树木总数的预测值: {mean_burned_trees:.2f}")
print(f"受影响的树木总数的标准差: {std_burned_trees:.2f}")
```

```
超过30%树木燃烧的概率: 0.00307
受影响的树木总数的预测值: 43.20
受影响的树木总数的标准差: 62.59
```