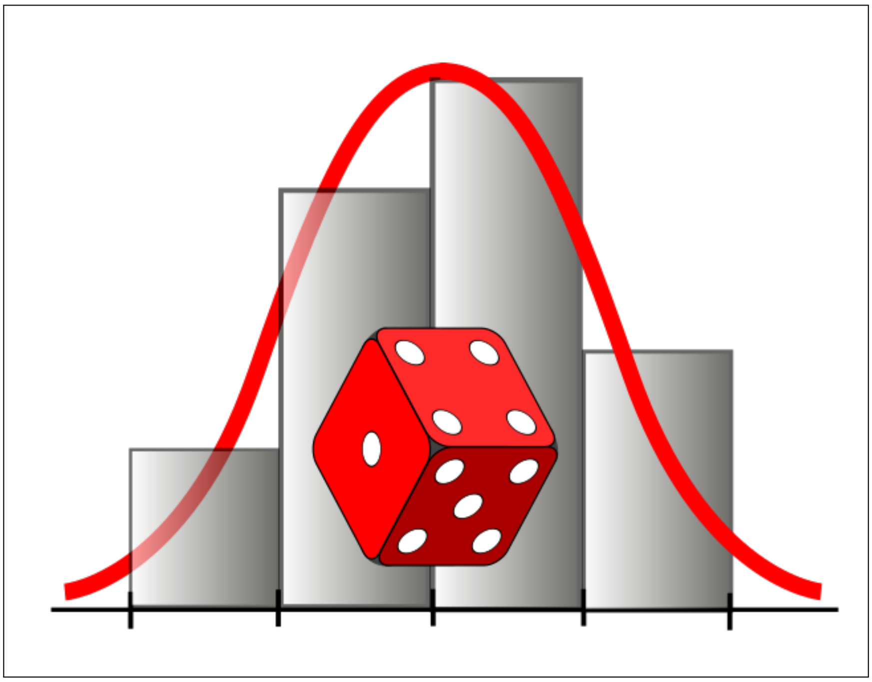
"Лабрадорная" работа 2 по Теории вероятностей

Мочеков Семён



1 Задача 1.1.

Пусть такое возможно и $X = X_1 + X_2$, где X_1, X_2 независимы и имеют невырожденное распределение.

В силу независимости, вероятность того, что X примет значение равное k^2, представляется по формуле свёртки:

$$P(X = k^2) = P(X_1 + X_2 = k^2) = \sum_j P(X_1 = j) \cdot P(X_2 = k^2 - j)$$

В связи с этим встаёт вопрос: Какие j могут быть в сумме?

Из условия имеем, что носителем X является множество квадратов натуральных чисел с нулём. То есть:

$$supp(X) = \{n^2 : n \in \mathbb{N}_0\}$$

Но тогда:

$$supp(X) = supp(X_1 + X_2) = \{n^2 : n \in \mathbb{N}_0\}$$

То есть все возможные суммы случайных величин X_1, X_2 должны давать в точности множество квадратов $\mathbb{N}_0$.

Теперь обратим внимание на второе свойство X_1, X_2. Их распределение не вырождено, то есть каждая из случайных величин принимает больше чем одно значение. Ну и дополнительно отметим, что носитель хотя бы одной должен быть счётен, иначе не выполнена счётность носителя X.

Так как эта пара случайных величин задаёт своей суммой все квадраты $\mathbb{N}_0$, то каждая из сумм свёртки не пуста.

Не умаляя общности будем отталкиваться от того, что некоторое значение принимает именно X_1, а дополняющее до квадрата X_2.

Тогда для некоторого m и $\hat{j}$:

$$P(X = m^2) = P(X_1 = \hat{j}) \cdot P(X_2 = m^2 - \hat{j}) + \sum_{j,\, j \neq \hat{j}} P(X_1 = j) \cdot P(X_2 = m^2 - j)$$

Аналогично для некоторого d и $\dot{j}$:

$$P(X = d^2) = P(X_1 = \dot{j}) \cdot P(X_2 = d^2 - \dot{j}) + \sum_{j,\, j \neq \dot{j}} P(X_1 = j) \cdot P(X_2 = d^2 - j)$$

При том $\hat{j}, \dot{j}$ таковы, что $\hat{j} \neq \dot{j}$ так как X_1 принимает хотя бы два значения.

Тогда, возвращаясь к носителю случайной величины, мы приходим к тому, что на самом деле $supp(X_1 + X_2)$ не будет совпадать с квадратами натуральных чисел с нулём "почти наверное". Так как, то же $m^2 - \hat{j} + \dot{j}$ является квадратом в исключительных случаях.

Предположение привело нас к противоречию.

Но если допустить, что у одной величины вырожденное распределение, то есть $X_1 \in \{c\}$, тогда можно представить X как: $X = X_1 + X_2,\ X_1 \in \{c\},\ X_2 \in \{k^2 - c\}_{k=0}^{+\infty}$

2 Задача 2.3.

Определение стандартного гауссовского вектора:

Вектор $X = (X_1, \ldots, X_n)$ будем называть ***стандартным гауссовским****, если его плотность имеет вид*

$$p(x_1, \ldots, x_n) = \frac{1}{(2\pi)^{n/2}} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n} x_i^2\right\} = \frac{1}{(2\pi)^{n/2}} \exp\left\{-\frac{x^T x}{2}\right\}.$$

Покажем, что совместная плотность X, Y удовлетворяет этому определению.

$$X = \sqrt{-2\ln V}\cos(2\pi U), \quad Y = \sqrt{-2\ln V}\sin(2\pi U)$$

Введём:

$$\rho = \sqrt{-2\ln V}, \quad \varphi = 2\pi U$$

Тогда:

$$V = e^{-\frac{\rho^2}{2}}, \quad U = \frac{\varphi}{2\pi}$$

Что в принципе даёт выражение для полярных координат, так как $\varphi \in [0; 2\pi], \ \rho \in [0; +\infty)$. По области определения U, V Так как старые случайные величины имеющие равномерное распределение на $[0; 1]$, то плотность каждой равна $p_U = p_V = 1$, а их совместная плотность, в силу независимости $p_{UV} = 1 \cdot 1 = 1$.

Тогда по теореме о преобразовании плотности:

$$p_{XY} = p_{UV} \cdot \left|\frac{1}{\det(J)}\right|$$

$$J = \begin{pmatrix} \frac{\partial X}{\partial U} & \frac{\partial X}{\partial V} \\ \frac{\partial Y}{\partial U} & \frac{\partial Y}{\partial V} \end{pmatrix}$$

$$\frac{\partial X}{\partial U} = -2\pi\rho\sin(2\pi U), \quad \frac{\partial X}{\partial V} = -\frac{\cos(2\pi U)}{V\rho}$$

$$\frac{\partial Y}{\partial U} = 2\pi\rho\cos(2\pi U), \quad \frac{\partial Y}{\partial V} = -\frac{\sin(2\pi U)}{V\rho}$$

Тогда:

$$\left|\frac{1}{\det(J)}\right| = \frac{V}{2\pi} = \frac{1}{2\pi}e^{-\frac{\rho^2}{2}}$$

Ну и тогда переводя полярные координаты в обычные декартовы:

$$p_{XY} = 1 \cdot \frac{1}{2\pi}e^{-\frac{\rho^2}{2}} = \frac{1}{2\pi}e^{-\frac{\rho^2(\cos^2(\varphi)+\sin^2(\varphi))}{2}} = \frac{1}{2\pi}e^{-\frac{x^2+y^2}{2}}$$

Что в точности соответствует определению *стандартного гауссовского* вектора.

3 Задача 3.4

$$p(x) = \frac{e^{-|x|}}{2} \cdot (1 - e^{-1})^{-1} \cdot \mathbb{I}(|x| \leq 1)$$

```python
from scipy import stats
import numpy as np
from time import time

class CustomDistribution(stats.rv_continuous):
    def _pdf(self, x):
        return np.exp(-np.abs(x)) / (2 * (1 - np.exp(-1))) * (np.abs(x) <= 1)

    def _cdf(self, x):
        result = np.zeros_like(x, dtype=float)

        cond_first = x < -1
        result[cond_first] = 0

        cond_second = (x >= -1) & (x <= 0)
        result[cond_second] = (np.exp(x[cond_second])) / (2 * (1 - np.exp(-1)))

        cond_third = (x > 0) & (x <= 1)
        result[cond_third] = -(np.exp(-x[cond_third])) / (2 * (1 - np.exp(-1)))

        cond_fourth = x > 1
        result[cond_fourth] = 1

        return result

def inverse_cdf_sampling(n):
    """делаем семплс по обратной функции распределения"""
    u = np.random.uniform(0, 1, size=n)
    result = np.zeros_like(u)

    cond1 = u <= 0.5
    result[cond1] = np.log(2 * u[cond1] * (1 - np.exp(-1)) + np.exp(-1))

    cond2 = u > 0.5
    result[cond2] = -np.log(1 - 2 * (u[cond2] - 0.5) * (1 - np.exp(-1)))

    return result

def rejecting_sampling(n):
    """делаем семплс rejecting sampling"""
    samples = []
    pdf_bound = 1 / (1 - np.exp(-1))

    while len(samples) < n:
        batch_size = min(2 * (n - len(samples)), n)
        x_proposals = np.random.uniform(0, 1, size=batch_size)
        pdf_values = np.exp(-np.abs(x_proposals)) / (2 * (1 - np.exp(-1)))
        u = np.random.uniform(0, pdf_bound, size=batch_size)
        accepted = x_proposals[u <= pdf_values]

        samples.extend(accepted[:n - len(samples)])

    return np.array(samples)


custom_dist = CustomDistribution()
```

```
def method1_test(n_values):
    """проверяем ООП"""
    results = {}

    for n in n_values:
        start_time = time()
        samples = custom_dist.rvs(size=n)
        elapsed = time() - start_time

        results[n] = {'samples': samples,
                      'time': elapsed}

    return results

def method2_test(n_values):
    """проверяем обратную функцию распределения"""
    results = {}

    for n in n_values:
        start_time = time()
        samples = inverse_cdf_sampling(n)
        elapsed = time() - start_time

        results[n] = {'samples': samples,
                      'time': elapsed}

    return results

def method3_test(n_values):
    """проверяем rejecting sampling"""
    results = {}

    for n in n_values:
        start_time = time()
        samples = rejecting_sampling(n)
        elapsed = time() - start_time

        results[n] = {'samples': samples,
                      'time': elapsed}

    return results

def eherements():
    """экспереметы"""
    n_values_method1 = [10, 500, 1000]
    n_values_method23 = [10, 500, 1000, 5000, 10000, 100000]

    results_method1 = method1_test(n_values_method1)
    results_method2 = method2_test(n_values_method23)
    results_method3 = method3_test(n_values_method23)

    makefile(results_method1, results_method2, results_method3,
                             n_values_method1, n_values_method23)

def makefile(results_method1, results_method2, results_method3,
                             n_values_method1, n_values_method23, filename="teorver_stat.txt"):
    with open(filename, "w") as f:
        f.write("{:<10} {:<15} {:<12}\n".format(
            "Method", "Sample Size", "Time (s)"))
        f.write("-" * 40 + "\n")

        for n in n_values_method1:
```

```python
            f.write("{:<10} {:<15d} {:<12.6f}\n".format(
                "OOP", n, results_method1[n]['time']
            ))
        f.write("-" * 40 + "\n")

        for n in n_values_method23:
            for method_name, results in [("Inverse CDF", results_method2),
                                         ("Rejection", results_method3)]:
                f.write("{:<10} {:<15d} {:<12.6f}\n".format(
                    method_name, n, results[n]['time']
                ))
            f.write("-" * 40 + "\n")

if __name__ == "__main__":
    eherements()
```

```
Method      Sample Size     Time (s)
----------------------------------------
OOP         10              0.080208
OOP         500             4.034261
OOP         1000            6.558655
----------------------------------------
Inverse CDF 10              0.000000
Rejection   10              0.000000
----------------------------------------
Inverse CDF 500             0.000000
Rejection   500             0.000000
----------------------------------------
Inverse CDF 1000            0.000000
Rejection   1000            0.000000
----------------------------------------
Inverse CDF 5000            0.001000
Rejection   5000            0.001034
----------------------------------------
Inverse CDF 10000           0.000000
Rejection   10000           0.001196
----------------------------------------
Inverse CDF 100000          0.004676
Rejection   100000          0.015585
----------------------------------------
```

Rejecting sampling

Метод отбраковки используется для генерации случайных чисел из целевого распределения $p(x)$, если напрямую сгенерировать из него сложно. Для этого выбирается вспомогательное распределение $q(x)$, из которого выборка генерируется легко, и константа M, удовлетворяющая условию $p(x) \leq Mq(x)$ для всех x. Это гарантирует, что график $Mq(x)$ полностью покрывает график $p(x)$.

Алгоритм

1. Сгенерировать точку x из распределения $q(x)$
2. Сгенерировать число u из равномерного распределения $U[0;1]$
3. Если $u \leq \dfrac{p(x)}{Mq(x)}$, принять x как образец из $p(x)$. В противном случае — отбраковать x и повторить шаги

Обоснование

Вероятность принятия точки x пропорциональна отношению $\dfrac{p(x)}{Mq(x)}$. Совместная плотность принятых точек равна $q(x) \cdot \dfrac{p(x)}{Mq(x)} = \dfrac{p(x)}{M}$.

После нормировки на вероятность принятия $\displaystyle\int \frac{p(x)}{M} dx = \frac{1}{M}$, получаем, что принятые точки имеют распределение $p(x)$.

Вывод

Как видно из результатов, ООП проигрывает абсолютно. А наиболее эффективным себя показывает метод обратной функции распределения.

4 Задача 4.1

Имеем:

$$X_i \in \mathrm{Bern}(\theta \leftrightarrow p),\ \theta \leftrightarrow p \in (0;1)$$

$$P(|\overline{X}_{n_{\varepsilon\delta}} - \mu_\theta| \le \varepsilon) \ge 1 - \delta \Longrightarrow P(|\overline{X}_{n_{\varepsilon\delta}} - \mu_\theta| \ge \varepsilon) \le \delta$$

Наибольшая дисперсия $\sigma^2 = \mathrm{Var}(X) = p(1-p) \to_{\max} 0.25$. Это следует из анализа производной $(p(1-p))'_p = 1 - 2p$.

Чтобы оценки были верны для всех p, будем пользоваться этим далее.

- Тогда неравенство Чебышёва даёт:

$$P(|\overline{X}_{n_{\varepsilon\delta}} - \mu_\theta| \ge \varepsilon) \le \frac{\mathrm{Var}(\overline{X}_{n_{\varepsilon\delta}})}{\varepsilon^2} = \frac{\sigma^2}{n_{\varepsilon\delta}\varepsilon}$$

Хотим:

$$\frac{\sigma^2}{n_{\varepsilon\delta}\varepsilon} \le \delta \Longrightarrow \frac{\sigma^2}{\varepsilon\delta} \le n_{\varepsilon\delta}$$

Тогда получим оценку:

$$\frac{\sigma^2}{\varepsilon\delta} \le n_{\varepsilon\delta} \Longrightarrow \frac{0.25}{\varepsilon\delta} \le n_{\varepsilon\delta}$$

- Неравенство Хёфдинга, от части нр-в типа Чернова, даст нам:

$X_i \in \{a_i; b_i\} = \{0; 1\}$

$$P(|\overline{X}_n - \mu_\theta| \ge \varepsilon) \le 2\mathrm{exp}\left(-\frac{2\varepsilon^2 n^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

$$P(|\overline{X}_{n_{\varepsilon\delta}} - \mu_\theta| \ge \varepsilon) \le 2\mathrm{exp}(-2\varepsilon^2 n_{\varepsilon\delta})$$

Тогда чтобы получить, то что требуется:

$$\exp(-2\varepsilon^2 n_{\varepsilon\delta}) \le \delta \Longrightarrow \frac{\ln(\frac{2}{\delta})}{2\varepsilon^2} \le n_{\varepsilon\delta}$$

- Чтобы получить оценку по ЦПТ:

$$\frac{n(\overline{X}_{n_{\varepsilon\delta}} - \mu_\theta)}{\sqrt{n}\sigma} \longrightarrow_{n\to+\infty} \mathcal{N}(0,1)$$

Что даст нам:

$$\overline{X}_{n_{\varepsilon\delta}} \approx \mathcal{N}(\mu_\theta, \frac{\sigma^2}{n})$$

Не уходя далеко от стандартного нормального закона, применяя $T = \overline{X}_{n_{\varepsilon\delta}} - \mu_\theta$:

$$P(|\overline{X}_{n_{\varepsilon\delta}} - \mu_\theta| \ge \varepsilon) \approx P(|T| \ge \frac{\varepsilon\sqrt{n_{\varepsilon\delta}}}{\sigma})$$

$$P(|T| \ge \frac{\varepsilon\sqrt{n_{\varepsilon\delta}}}{\sigma}) = 1 - P(|T| \le \frac{\varepsilon\sqrt{n_{\varepsilon\delta}}}{\sigma}) = 2\Phi(-\frac{\varepsilon\sqrt{n_{\varepsilon\delta}}}{\sigma}) \le \delta \Longrightarrow \frac{\varepsilon\sqrt{n_{\varepsilon\delta}}}{\sigma} \ge Q_{\mathcal{N}}(1 - \frac{\delta}{2})$$

Выражая через квантили ($Q_{\mathcal{N}}$) стандартного нормального закона, получаем в итоге оценку на $n_{\varepsilon\delta}$:

$$\frac{\sigma^2 Q^2_{\mathcal{N}}(1-\frac{\delta}{2})}{\varepsilon^2} \le \frac{0.25 \cdot Q^2_{\mathcal{N}}(1-\frac{\delta}{2})}{\varepsilon^2} \le n_{\varepsilon\delta}$$

Теперь программно проверим наши оценки.

```python
import numpy as np
from scipy.stats import norm

def compute_n_chebyshev(eps, delta, var=0.5):
    """Вычисляем n из неравенства Чебышёва"""
    return int(np.ceil(var**2 / (delta * eps**2)))

def compute_n_chernoff(eps, delta):
    """Вычисляем n из неравенства типа Чернова"""
    return int(np.ceil((1 / (2 * eps**2)) * np.log(2 / delta)))

def compute_n_cpt(eps, delta, var=0.5):
    """Вычисляем n по ЦПТ"""
    Q = norm.ppf(1 - delta / 2)
    return int(np.ceil((Q * var / eps)**2))

def generate_sample(n, p):
    """Генерируем выборку из Bern(p)."""
    return np.random.binomial(1, p, size=n)

def is_successful(sample_mean, true_mean, eps):
    """Проверяем удовлетворены ли условия"""
    return abs(sample_mean - true_mean) <= eps

def run_experiment(n, p, epsilon, num_samples=100):
    successful_count = 0
    for i in range(num_samples):
        sample = generate_sample(n, p)
        sample_mean = np.mean(sample)
        if is_successful(sample_mean, p, epsilon):
            successful_count += 1
    success_rate = successful_count / num_samples
    return successful_count, success_rate

if __name__ == "__main__":
    eps = 0.01
    delta = 0.05
    p = 0.5 # Точка с наибольшей дисперсией для Bern(p)
    N = 100

    n_cheb = compute_n_chebyshev(eps, delta)
    succ_cheb, rate_cheb = run_experiment(n_cheb, p, eps, N)

    n_cher = compute_n_chernoff(eps, delta)
    succ_cher, rate_cher = run_experiment(n_cher, p, eps, N)

    n_cpt = compute_n_cpt(eps, delta)
    succ_cpt, rate_cpt = run_experiment(n_cpt, p, eps, N)

    with open("sammmples.txt", "w", encoding="utf-8") as f:
        f.write("Итог:\n")
        f.write(f"Чебышёв:  n = {n_cheb}, Успешные выборки: {succ_cheb}/{N}, Доля: {rate_cheb:.2%}\n")
        f.write(f"Чернов:   n = {n_cher}, Успешные выборки: {succ_cher}/{N}, Доля: {rate_cher:.2%}\n")
        f.write(f"ЦПТ:      n = {n_cpt}, Успешные выборки: {succ_cpt}/{N}, Доля: {rate_cpt:.2%}\n")
```

```
Итог:
Чебышёв:  n = 50000, Успешные выборки: 100/100, Доля: 100.00%
Чернов:   n = 18445, Успешные выборки: 99/100, Доля: 99.00%
ЦПТ:      n = 9604, Успешные выборки: 95/100, Доля: 95.00%
```

Оценка Чебышёва в силу грубости даёт наибольший успех в выборках, но требует очень большой объём.

Оценка по неравенству типа Чернова сильно улучшает границу, сохраняя хороший процент доли успеха.

Но всё же ЦПТ справляется наиболее лучшим образом. Теряя лишь пять процентов в точности.