Probability Theory tasks #6-9

Vusal Aliev

1. Problems to Solve

#6.16. Для того, чтобы воспользоваться формулой полной вероятности, необходимо ввести две гипотезы и посчитать их вероятности:

$$P(H_1) = \frac{25}{30} \cdot \frac{5}{29} + \frac{5}{30} \cdot \frac{25}{29} \text{ — студент знает ответ на один из вопросов}$$
$$P(H_2) = \frac{25}{30} \cdot \frac{24}{29} \text{ — студент знает ответы на два вопроса}$$

Также введем событие A — событие, при котором студент сдал экзамен. Тогда условные вероятности сдачи экзамена:

$$P(A \mid H_1) = \frac{24}{28} \text{ — при условии, что студент ответил на один из вопросов}$$
$$P(A \mid H_2) = 1 \text{ — при условии, что студент ответил на два вопроса}$$

Применив формулу полной вероятности, получим:

$$\begin{aligned} P(A) &= \sum_{j=1}^{2} P(H_j) \cdot P(A \mid H_j) \\ &= P(H_1) \cdot P(A \mid H_1) + P(H_2) \cdot P(A \mid H_2) \\ &= \left(\frac{25}{30} \cdot \frac{5}{29} + \frac{5}{30} \cdot \frac{25}{29}\right) \cdot \frac{24}{28} + \left(\frac{25}{30} \cdot \frac{24}{29}\right) \cdot 1 \\ &= \frac{190}{203} \approx \underline{0.935961} \end{aligned} \tag{1}$$

#7.17. Для того, чтобы определить вероятность того, что деталь была взята из партии, имеющей второсортные детали детали, необходимо воспользоваться формулой Байеса. Введём три гипотезы и посчитаем их вероятности:

$$P(H_1) = P(H_2) = P(H_3) = \frac{1}{3} \text{ — деталь взята из конкретной партии}$$

Также введем событие A — событие, при котором взята первосортная деталь. Тогда условные вероятности взятия первосортной детали из конкретной партии:

$$P(A \mid H_1) = \frac{2}{3} \text{ — при условии, что деталь взята из партии, где есть второсортные детали}$$
$$P(A \mid H_2) = P(A \mid H_3) = 1 \text{ — при условии, что деталь взята из других партий}$$

Применив формулу Байеса, получим:

$$P(H_1 \mid A) = \frac{P(H_1)P(A \mid H_1)}{\sum_{j=1}^{3} P(H_j)P(A \mid H_j)} = \frac{\frac{1}{3}}{\frac{1}{3}} \cdot \frac{\frac{2}{3}}{\frac{2}{3} + 1 + 1} = \frac{1}{4} \tag{2}$$

#8.7. Для того, чтобы определить вероятность того, что схема выйдет из строя, попробуем посчитать вероятность обратного события, когда схема **не** выйдет из строя.
Схема работает при условии, что:

1. Работают два элемента типа A
2. Работают один элемент типа B
3. Работает либо один из крайних элементов типа C, либо любые три элементы типа C, либо все элементы типа C, либо два элемента типа C (один из которых крайний)

Обозначим события A, B, C как события, при которых один элемент, соответсвующего типа, работает в цепи. Тогда их вероятности равны:

$$P(A) = 0.7, P(B) = 0.6, P(C) = 0.8$$

Используя вышеперечисленное, получаем вероятность цепи не выйти из строя:

$$\begin{aligned} p &= P(A)^2 \cdot P(B) \cdot \left(2P(C)(1-P(C))^3 + 5P(C)^2(1-P(C))^2 + 4P(C)^3(1-P(C)) + P(C)^4\right) \\ &= 0.7^2 \cdot 0.6 \cdot (2 \cdot 0.8 \cdot 0.2^3 + 5 \cdot 0.8^2 \cdot 0.2^2 + 4 \cdot 0.8^3 \cdot 0.2 + 0.8^4) = 0.28224 \end{aligned}$$

Коэффициенты при P(C) определены количеством благоприятных ситуаций, исходя из условий о работе схемы.

Искомая вероятность равна:

$$P = 1 - p = 1 - 0.28224 = 0.71776 \tag{3}$$

#9.3. По условию $\forall i \in \{1, 2, 3\} : p_i = \frac{1}{3}$. Тогда воспользуемся полиномиальным распределением, чтобы вычислить искомые вероятности:

a.) $$P_{9,3,3,3} = \frac{9!}{3! \cdot 3! \cdot 3!} \cdot p_1^3 \cdot p_2^3 \cdot p_3^3 = \frac{9!}{3! \cdot 3! \cdot 3!} \cdot \left(\frac{1}{3}\right)^{3+3+3} = \frac{560}{6561} \approx \underline{0.0853528} \tag{4}$$

b.) $$P_{9,4,3,2} = \tfrac{9!}{4! \cdot 3! \cdot 2!} \cdot p_1^4 \cdot p_2^3 \cdot p_3^2 = \tfrac{9!}{4! \cdot 3! \cdot 2!} \cdot \left(\tfrac{1}{3}\right)^{4+3+2} = \tfrac{140}{2187} \approx 0.0640146$$

В пункте (b) нужно также учесть перестановки ящиков, потому что шары распределяются в разном количестве. Конечная вероятность для второго случая равна:

$$p = P_{9,4,3,2} \cdot P_3 = 0.0640146 \cdot 3! = \underline{0.384088} \tag{5}$$

2. Problems to Model

#6.16. Аналитическое решение представлено в выражении (1). Для моделирования использовался язык программирования Python. Исходный код программы представлен в листинге 1. Создадим функцию, имитирующую экзамен из задания. Если студент его сдаёт, то возвращается True, иначе возвращается False. Запустим 1000000 экспериментов с помощью созданной функции *exam()* и выберем благоприятные исходы.

Листинг 1. Моделирование задачи 6.16

```python
from random import *

def exam():
    questions = [1] * 25 + [0] * 5
    shuffle(questions)

    bilet1 = questions[:2]
    bilet2 = questions[2:4]

    if bilet1.count(1) == 2:
        return True

    if bilet1.count(1) == 1 and bilet2[0] == 1:
        return True

    return False

if __name__ == '__main__':
    tries_number = 1000000
    good_tries = 0
    for i in range(tries_number):
        result = exam()
        if result:
            good_tries += 1

    model = good_tries / tries_number
    print("Смоделированный результат:", model)
```

Смоделированное решение равно $P_M = \frac{\text{good\_tries}}{1000000} = \frac{935744}{1000000} = \underline{0.935744}$. С аналитическим решением совпадение в 3 значащие цифры.

#7.17. Аналитическое решение представлено в выражении (2). Для моделирования использовался язык программирования Python. Исходный код программы представлен в листинге 2. Создадим функцию, имитирующую вытаскивание первосортной детали из задания. Если вытащенная деталь из партии с второсортными деталями, то возвращается True, иначе возвращается False. Запустим 10000000 экспериментов с помощью созданной функции *take_part()* и выберем благоприятные исходы.

Листинг 2. Моделирование задачи 7.17

```python
from random import *

def take_part():
    batch1 = [1, 1, 1]
    batch2 = [1, 1, 1]
    batch3 = [1, 1, 2]

    batches = [batch1, batch2, batch3]

    while True:
        batch_number, part_number = randint(0, 2), randint(0, 2)

        if batches[batch_number][part_number] != 2:
            break

    return batches[batch_number].count(2) == 1

if __name__ == '__main__':
    tries_number = 1000000
    good_tries = 0
    for i in range(tries_number):
        result = take_part()
        if result:
            good_tries += 1

    model = good_tries / tries_number
    print("Смоделированный результат:", model)
```

Смоделированное решение равно $P_M = \frac{\text{good\_tries}}{10000000} = \frac{2504430}{10000000} = \underline{0.250443}$. С аналитическим решением совпадение в 2 значащие цифры.

#8.7. Аналитическое решение представлено в выражении (3). Для моделирования использовался язык программирования Python. Исходный код программы представлен в листинге 3. Создадим функцию, имитирующую работу цепи из задания. Если цепь выходит из строя, то возвращается True, иначе возвращается False. Запустим 10000000 экспериментов с помощью созданной функции *circuit_fail()* и выберем благоприятные исходы. Моделирование для элементов типа C можно минимизировать, потому что на работу цепи влияют лишь крайние элементы.

Листинг 3. Моделирование задачи 8.7

```python
from random import *

def circuit_fail():
    if randint(1, 10) <= 3:
        return True
    if randint(1, 10) <= 3:
        return True
    if randint(1, 10) <= 4:
        return True
    # c1 - c2 - c3 - c4
    c1 = randint(1, 10) <= 8
    c2 = randint(1, 10) <= 8
    c3 = randint(1, 10) <= 8
    c4 = randint(1, 10) <= 8
    return not (c1 or c4)

if __name__ == '__main__':
    tries_number = 1000000
    good_tries = 0
    for i in range(tries_number):
        result = circuit_fail()
        if result:
            good_tries += 1

    model = good_tries / tries_number
    print("Смоделированный результат:", model)
```

Смоделированное решение равно $P_M = \frac{\text{good\_tries}}{10000000} = \frac{7176756}{10000000} = \underline{0.7176756}$. С аналитическим решением совпадение в 3 значащие цифры.

#9.3. Аналитическое решение представлено в выражениях (4) и (5). Для моделирования использовался язык программирования Python. Исходный код программы представлен в листинге 4. Создадим функцию, имитирующую распределение шаров из задания. В результате возвращается распределение шаров в виде массива. Запустим 1000000 экспериментов с помощью созданной функции *balls_distribution()* и выберем благоприятные исходы.

Листинг 4. Моделирование задачи 9.3

```
from random import *

def balls_distribution():
    balls = [[], [], []]

    for _ in range(9):
        picked_ball = randint(0, 2)
        balls[picked_ball].append(1)

    return balls

if __name__ == '__main__':
    tries_number = 1000000
    a_good_tries, b_good_tries = 0, 0
    for _ in range(tries_number):
        a_result = balls_distribution()
        if a_result[0].count(1) == 3
          and a_result[1].count(1) == 3
          and a_result[2].count(1) == 3:
            a_good_tries += 1

    for _ in range(tries_number):
        b_result = balls_distribution()
        b_set = set([box.count(1) for box in b_result])
        if {4, 3, 2} == b_set:
            b_good_tries += 1

    model_a = a_good_tries / tries_number
    print("Смоделированный результат для пункта A:", model_a)

    model_b = b_good_tries / tries_number
    print("Смоделированный результат для пункта B:", model_b)
```

Смоделированное решение для пункта А равно $P_M(\text{А}) = \frac{\text{good\_tries}}{1000000} = \frac{85389}{1000000} = \underline{0.085389}$, для пункта Б равно $P_M(\text{Б}) = \frac{\text{good\_tries}}{1000000} = \frac{385134}{1000000} = \underline{0.385134}$. С аналитическим решением совпадение в 2 значащие цифры.