SỰ XÂM NHẬP VÀ HỢP THÀNH GIỮA LẬP TRÌNH PYTHON VÀ XÁC SUẤT THÔNG KÊ ỨNG DỤNG

THE INTERPRETATION AND COMPARISON OF APPLICATION STATISTICS PROBABILITY AND PYTHON PROGRAMMING

NGUYỄN VĂN LỘC(*)

THÔNG TIN

Ngày nhận bài: 17-7-2024
Ngày biên tập xong: 05-9-2024
Ngày duyệt đăng: 31-9-2024
Mã số: TCKH47-09-2024
ISSN: 2525 – 2429

Từ khóa: *phương pháp; ngôn ngữ lập trình Python; xác suất thống kê.*

Key words: *approach, Python programming language, and statistical likelihood.*

TÓM TẮT

Bài viết trình bày phương pháp ngôn ngữ lập trình Python để giải bài toán xác suất thống kê, thể hiện trên ví dụ minh họa là bài toán tìm số điện thoại đúng với 7 cách giải khác nhau. Các cách giải sử dụng thư viện hàm itertools; thư viện random; thư viện numpy; thư viện scipy; thư viện pandas; thư viện secrets.

ABSTRACT: *In addition to providing an example of a telephone problem with seven possible solutions, the article describes the process of integrating applied statistical probability with Python programming. The function library is used in the solutions. The libraries include itertools, random, numpy, scipy, pandas, and secrets.*

1. ĐẶT VẤN ĐỀ

Để nâng cao hiệu quả dạy học thực hành xác suất thống kê, tất yếu phải xác lập mối liên hệ giữa thuật toán giải các bài tập xác suất thống kê và code của mỗi thư viện hàm được sử dụng. Muốn vậy, cần xác định công dụng và sự khác biệt của các thư viện hàm phù hợp bối cảnh của bài toán. Với định hướng đó, bài viết mô tả sử dụng phương pháp lập trình Python để giải bài toán xác suất thống kê và minh họa trên ví dụ là bài toán tìm số điện thoại đúng khi sử dụng đa dạng các thư viện hàm Python trong các cách giải khác nhau.

2. NỘI DUNG

2.1. Phương pháp lập trình Python để giải bài toán xác suất thống kê

Quy trình gồm 5 bước sau:

Bước 1. Xác định thuật toán giải bài toán trên bằng lập trình Python;

Bước 2. Sử dụng một thư viện hàm Python viết code lập trình theo thuật toán đã xác định;

Bước 3. Xác định chi tiết về ý nghĩa các hàm và các phần quan trọng trong đoạn code đã viết;

Bước 4. Viết đoạn mã với các chú thích tương ứng;

Bước 5. Xác định sự khác biệt giữa các hàm trong các cách giải với các thư viện hàm khác nhau

2.2. Các ví dụ

Ví dụ: Một người gọi điện thoại nhưng lại quên hai số cuối của số điện thoại, chỉ biết rằng hai số đó là khác nhau. Tính xác suất để người đó chỉ quay một lần đúng số cần gọi.

(*) PGS.TS. Trường Đại học Văn Lang, *loc.nv@vlu.edu.vn*

Giải: Lựa chọn thư viện hàm khác nhau để giải bài toán và xác định thuật toán giải bài toán với thư viện đã chọn bằng lập trình Python.

Thuật toán

1. Tạo danh sách các chữ số từ 0 đến 9;

2. Tạo tất cả các cặp số khác nhau có thể có từ danh sách trên;

3. Tính tổng số các cặp số khác nhau có thể;

4. Xác định số trường hợp thành công (chỉ có một cặp số đúng);

5. Tính xác suất bằng cách chia số trường hợp thành công cho tổng số cặp số khác nhau.

Cách 1. Sử dụng thư viện hàm itertools để tạo ra các hoán vị của các số [1].

Code:

```
#Cách 1_Sử dụng thư viện hàm itertools tạo các hoán vị
import itertools
def calculate_probability_correct_call():
digits = list(range(10))
# Số các cặp số khác nhau
total_combinations=len(list(itertools.permutations(digits, 2)))
# Chỉ có 1 cặp số đúng
favorable_outcome = 1
# Xác suất là tỷ lệ giữa số trường hợp đúng và tổng số trường hợp có thể
probability=favorable_outcome/total_combinations
return probability
# Tính và in xác suất
probability=calculate_probability_correct_call()
print(f"Xác suất để người đó chỉ quay một lần đúng số cần gọi là: {probability:.4f}")
```

Xác suất để người đó chỉ quay một lần đúng số cần gọi là: 0.0111

Các hàm và logic chính

1. import itertools: Thư viện itertools cung cấp các công cụ hiệu quả để làm việc với các tổ hợp và hoán vị;

2. itertools.permutations(iterable, r): Tạo ra tất cả các hoán vị dài r của các phần tử trong iterable;

3. len(list(itertools.permutations(digits,2))): Tạo một danh sách từ các hoán vị được tạo ra và tính độ dài của danh sách đó để xác định tổng số cặp số khác nhau;

4. favorable_outcome = 1: Số trường hợp đúng là 1;

5. probability = favorable_outcome / total_combinations: Tính xác suất bằng cách chia số trường hợp đúng cho tổng số trường hợp có thể.

Dưới đây là toàn bộ mã với các chú thích chi tiết tương ứng:

```
#Cách 1_Sử dụng thư viện hàm itertools với chú thích chi tiết
import itertools  # Thư viện itertools để làm việc với các tổ hợp và hoán vị
def calculate_probability_correct_call():
digits = list(range(10))  # Tạo danh sách các chữ số từ 0 đến 9
# Số các cặp số khác nhau
total_combinations=len(list(itertools.permutations(digits, 2)))
# Chỉ có 1 cặp số đúng
favorable_outcome = 1
# Xác suất là tỷ lệ giữa số trường hợp đúng và tổng số trường hợp có thể
probability=favorable_outcome/total_combinations
return probability
# Tính và in xác suất
probability=calculate_probability_correct_call()
print(f"Xác suất để người đó chỉ quay một lần đúng số cần gọi là: {probability:.4f}")
```

Xác suất để người đó chỉ quay một lần đúng số cần gọi là: 0.0111

Nhận xét. Đoạn mã này tính toán xác suất trực tiếp mà không cần phải mô phỏng các cuộc gọi, nhờ vào việc sử dụng các hoán vị từ itertools. Điều này làm cho mã ngắn gọn và hiệu quả hơn.

Cách 2. Sử dụng thư viện random để mô phỏng việc gọi điện thoại nhiều lần và đếm số lần thành công [4].

Thuật toán

Xác định các khả năng có thể xảy ra: Có 10 chữ số có thể cho mỗi vị trí (từ 0 đến 9); Hai

số cuối khác nhau, tức là mỗi khả năng đều khác nhau.

Xác định tổng số trường hợp có thể: Chọn số đầu tiên từ 10 số (0-9) có 10 cách; Chọn số thứ hai từ 9 số còn lại (khác với số đã chọn trước đó) có 9 cách; Tổng cộng có 10x9 =90; 10×9=90 khả năng khác nhau cho hai số cuối.

Xác định số trường hợp đúng: Có đúng 1 trường hợp là đúng số cần gọi.

Tính xác suất: Xác suất là tỷ số giữa số trường hợp đúng và tổng số trường hợp có thể xảy ra: 1/90

Code:

```
#Cách 2_Sử dụng thư viện hàm random
import random
def simulate_phone_call():
correct_number = random.randint(0, 9), random.randint(0, 9)
while correct_number[0]== correct_number [1]:
correct_number=random.randint(0,9), random. randint(0, 9)
attempts = [(i, j) for i in range(10) for j in range(10) if i != j]
success = sum(1 for attempt in attempts if attempt == correct_number)
probability = success / len(attempts)
return probability
# Tính xác suất
probability = simulate_phone_call()
print(f"Xác suất để người đó chỉ quay một lần đúng số cần gọi là: {probability:.5f}")
```

Xác suất để người đó chỉ quay một lần đúng số cần gọi là: 0.01111

Các hàm và logic chính

1. import random: Thư viện random được sử dụng để tạo ra các số ngẫu nhiên;

2. random.randint(a, b): Tạo ra một số nguyên ngẫu nhiên từ a đến b;

3. while correct_number[0] == correct_number [1]: Đảm bảo rằng hai số cuối là khác nhau;

4. List Comprehension [(i, j) for i in range(10) for j in range(10) if i != j]: Tạo danh sách tất cả các cặp số khác nhau;

5. sum(1 for attempt in attempts if attempt == correct_number): Đếm số lần cặp số đúng xuất hiện trong danh sách các cặp số;

6. probability = success / len(attempts): Tính xác suất bằng cách chia số lần thành công cho tổng số khả năng.

Dưới đây là toàn bộ mã với các chú thích chi tiết [2]

```
#Cách 2_Sử dụng thư viện hàm random với chú thích chi tiết
import random  # Thư viện random để tạo số ngẫu nhiên
def simulate_phone_call():
# Chọn ngẫu nhiên hai số cuối của số điện thoại
correct_number = random.randint(0, 9), random.randint(0, 9)
# Đảm bảo rằng hai số này khác nhau
while correct_number[0]==correct_number [1]:
correct_number = random.randint(0, 9), random.randint(0, 9)
# Tạo danh sách tất cả các cặp số có thể, với điều kiện hai số khác nhau
attempts = [(i, j) for i in range(10) for j in range(10) if i != j]
# Đếm số lần cặp số đúng xuất hiện trong danh sách các khả năng
success = sum(1 for attempt in attempts if attempt == correct_number)
# Tính xác suất bằng cách chia số lần thành công cho tổng số khả năng
probability = success / len(attempts)
return probability
# Tính xác suất
probability = simulate_phone_call()
print(f"Xác suất để người đó chỉ quay một lần đúng số cần gọi là: {probability:.5f}")
```

Xác suất để người đó chỉ quay một lần đúng số cần gọi là: 0.01111

Sự khác biệt chính của các hàm lập trình trong hai cách giải trên:

1) Phương pháp:

Cách 1: Sử dụng mô phỏng ngẫu nhiên để chọn các số và đếm số lần thành công. Đây là phương pháp tiếp cận dựa trên mô phỏng và sử dụng random;

Cách 2: Sử dụng toán học và hoán vị để tính toán trực tiếp xác suất. Đây là phương pháp tiếp cận lý thuyết xác suất và sử dụng itertools.

2) Hiệu suất và độ phức tạp

Cách 1: Có thể không hiệu quả đối với các bài toán phức tạp hơn, nhưng dễ hiểu và triển khai;

Cách 2: Hiệu quả hơn vì nó tính toán trực tiếp mà không cần phải mô phỏng nhiều lần.

3) Độ chính xác

Cách 1: Có thể có sự sai lệch nhỏ do tính ngẫu nhiên trong mô phỏng;

Cách 2: Đưa ra kết quả chính xác vì sử dụng phương pháp tính toán trực tiếp.

Cả hai cách đều giải quyết cùng một bài toán nhưng sử dụng các phương pháp khác nhau. Cách thứ hai là phương pháp tính toán trực tiếp nên sẽ luôn đưa ra kết quả chính xác và nhanh hơn so với cách thứ nhất.

Ngoài hai cách sử dụng các thư viện itertools và thư viện random, ta có thể sử dụng các thư viện khác như sau:

Cách 3. Sử dụng thư viện numpy để tính toán [5]

Code:

```
# Cach 3_Sử dụng thư viện hàm numpy
import numpy as np
def calculate_probability_with_numpy():
digits = np.arange(10)
# Số các cặp số khác nhau
total_combinations=len(np.array(np.mesh grid(digits, digits)).T.reshape(-1,2))
total_combinations=total_combinations - 10 # Loại bỏ các cặp có hai số giống nhau
# Chỉ có 1 cặp số đúng
favorable_outcome = 1
# Xác suất là tỷ lệ giữa số trường hợp đúng và tổng số trường hợp có thể
probability=favorable_outcome/total_combinations
return probability
# Tính và in xác suất
probability=calculate_probability_with_numpy()
print(f"Xác suất để người đó chỉ quay một lần đúng số cần gọi là: {probability:.4f}")
```

Xác suất để người đó chỉ quay một lần đúng số cần gọi là: 0.011

Cách 4. Sử dụng thư viện itertools để tính toán

```
#Cách 4_ Sử dụng thư viện itertools để tính toán
import itertools
def comb(n, k):
return len(list(itertools.combinations(range (n), k)))
# Tính số các cặp số khác nhau
total_pairs = comb(10, 2) * 2  # Vì mỗi cặp có thể xuất hiện theo hai thứ tự
# Tính xác suất quay đúng số cần gọi
probability = 1 / total_pairs
print(probability)
0.011111111111111112
```

Cách 5. Sử dụng thư viện scipy [3]

```
#Cách 5_ Sử dụng thư viện scipy
from scipy.special import comb
# Tính số các cặp số khác nhau
total_pairs = comb(10, 2) * 2  # Vì mỗi cặp có thể xuất hiện theo hai thứ tự
# Tính xác suất quay đúng số cần gọi
probability = 1 / total_pairs
print(probability)
0.011111111111111112
```

Cách 6. Sử dụng thư viện pandas

```
#Cach 6_ Sử dụng thư viện pandas
import pandas as pd
# Tạo DataFrame chứa tất cả các cặp số có thể có (từ 00 đến 99)
digits = range(10)
pairs = [(x, y) for x in digits for y in digits if x != y]
df=pd.DataFrame(pairs, columns=['Digit1', 'Digit2'])
# Tính tổng số các cặp số khác nhau
total_pairs = len(df)
# Tính xác suất quay đúng số cần gọi
probability = 1 / total_pairs
print(probability)
0.011111111111111112
```

Cách 7. Sử dụng thư viện secrets

```
#Cach 7_ sử dụng thư viện secrets
import secrets
def calculate_probability():
total_cases = 100  # Số lần gọi điện thoại ngẫu nhiên
successful_cases = 0  # Số lần gọi điện thoại thành công
for _ in range(total_cases):
correct_last_two_digits=secrets.randbelow(100)  # Sinh ngẫu nhiên số điện thoại đúng
guess = secrets.randbelow(100)  # Sinh ngẫu nhiên số điện thoại gọi
if guess == correct_last_two_digits:
successful_cases += 1
probability=successful_cases / total_cases
return probability
probability = calculate_probability()
print(f"Xác suất để người đó chỉ quay một lần đúng số cần gọi là: {probability:.2f}")
```

Xác suất để người đó chỉ quay một lần đúng số cần gọi là: 0.01

Tất cả các phương pháp trên đều giải quyết bài toán bằng cách xác định số các cặp số khác nhau và tính xác suất của một cặp số đúng. Lựa chọn phương pháp và thư viện phù hợp sẽ phụ thuộc vào nhu cầu và bối cảnh cụ thể của chúng ta.

Luyện tập. Bài toán của Samuel-Pepys: Pepys đã đặt bài toán sau cho Newton: Biến cố nào trong các biến cố sau đây có xác suất lớn nhất:

a. Có ít nhất một lần xuất hiện mặt 6 khi tung con xúc xắc 6 lần?

b. Có ít nhất hai lần xuất hiện mặt 6 khi tung con xúc xắc 12 lần?

c. Có ít nhất ba lần xuất hiện mặt 6 khi tung con xúc xắc 18 lần?

Hướng dẫn giải. Để giải bài toán của Samuel Pepys bằng Python, chúng ta sẽ tính toán xác suất của mỗi biến cố bằng cách sử dụng định lý nhị thức hoặc mô phỏng Monte Carlo.

3. KẾT LUẬN

Phương pháp sử dụng ngôn ngữ lập trình Python để giải bài toán xác suất thống kê, cho phép xác lập sự liên hệ chặt chẽ giữa thuật toán và quy trình giải bằng lập trình Python. Xác lập sự dịch chuyển giữa ngôn ngữ thuật toán và các ngôn ngữ hàm Python, giúp nâng cao khả năng giải Toán bằng lập trình Python không chỉ trong xác suất thống kê mà cả trong các môn học thực hành toán cao cấp ở các trường đại học.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Thành Cả (2014), *Xác suất và Thống kê Toán*, Nxb Kinh tế Thành phố Hồ Chí Minh.

[2] David R. Anderson, Dennis J. Sweeny, Thomas A. Williams, (Hoàng Trọng (Chủ biên dịch)) (2020), *Thống kê trong kinh tế và kinh doanh,* Nxb Kinh tế Thành phố Hồ Chí Minh.

[3] Michael Baron (2014), *Probability and statistics for computer scientists*, CRC Press.

[4] Jaan Kiusalaas (2005), *Numerical Methods in Engineering With Python*, Cambridge University press.

[5] Kevin Sheppard (2019), *Introduction to Python for Econometrics, Statistics and Data Analysis*, University of Oxford.