人工智能程序设计


python
import turtle
turtle.setup(650,350,200,200)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.pensize(25)
turtle.pencolor("purple")
turtle.seth(-40)
for i in range(4):
turtle.circle(40, 80)
turtle.circle(-40, 80)
turtle.fd(40)
turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16, 180)
turtle.fd(40 * 2/3)



人工智能程序设计

4.4 随机函数库RANDOM

北京石油化工学院 人工智能研究院

刘 强

随机数

随机数是指在一定范围内随机产生的数值，其核心特征是无法预测下一个数值会是什么

在现实世界中，掷骰子、抽奖、洗牌等都是产生随机结果的典型例子。在计算机程序中，随机数让程序具有不确定性和多样性，使得每次运行都可能产生不同的结果

随机数在编程中有着广泛的应用：游戏开发中的随机事件、数据科学中的采样分析、机器学习中的参数初始化、密码学中的加密算法、模拟仿真中的不确定性建模等。

4.4.1 生成随机数

Python的random模块提供了生成伪随机数的各种函数，这些函数在游戏开发、数据分析、科学计算和模拟等领域有着广泛的应用。

基本随机数生成

random模块提供了多种生成随机数的函数，每种函数都有其特定的用途：

- random.random()：生成0到1之间的随机浮点数
- random.uniform(a, b)：生成指定范围内的随机浮点数
- random.randint(a, b)：生成指定范围内的随机整数
- random.choice(seq)：从序列中随机选择一个元素
- random.shuffle(list)：随机打乱列表中元素的顺序

4.4.1 生成随机数

```
import random

## 生成0到1之间的随机浮点数
print(random.random())  # 输出：0.37444887175646646

## 生成指定范围的随机浮点数
print(random.uniform(1.5, 8.7))  # 输出：5.234567891234567
## 生成指定范围的随机整数（包括两个端点）
print(random.randint(1, 6))  # 输出：4（模拟掷骰子）

## 从序列中随机选择一个元素
fruits = ["苹果", "香蕉", "橙子", "葡萄", "草莓"]
print(random.choice(fruits))  # 输出：香蕉

## 打乱序列顺序
numbers = [1, 2, 3, 4, 5]
random.shuffle(numbers)
print(numbers)  # 输出：[3, 1, 5, 2, 4]
```

4.4.1 生成随机数

随机种子：随机种子是随机数生成器的起始值。计算机生成的随机数实际上是"伪随机数"，它们是通过数学算法产生的确定性序列。通过设置相同的随机种子，我们可以让随机数序列变得可重现

```
import random
## 设置随机种子
random.seed(42)
print(random.randint(1, 100))  # 每次运行都输出相同的数字
## 不设置种子，每次运行结果不同
print(random.randint(1, 100))  # 输出随机数字
```

4.4.2 蒙特卡洛模拟计算π

蒙特卡洛方法是一种基于随机抽样的数值计算方法。计算π值的基本思路是：

1. 在一个正方形内画一个内切圆

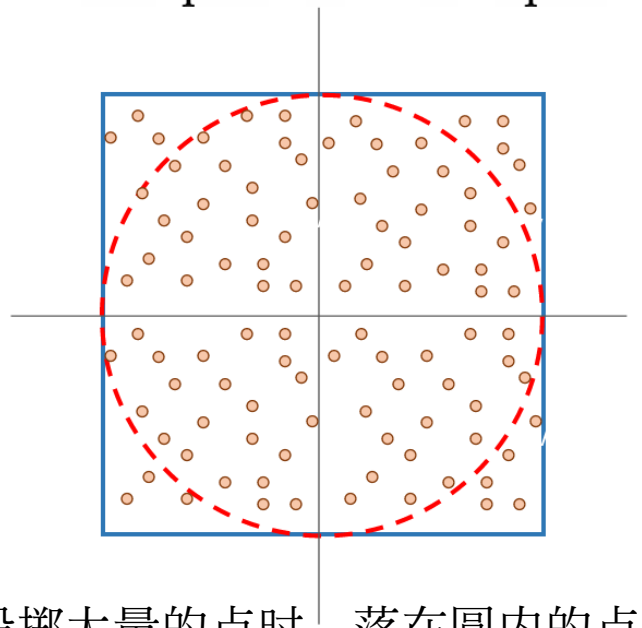
2. 随机生成点，统计落在圆内的点数

3. 利用面积比例关系计算π值：$\pi \approx 4 \times$ (圆内点数 / 总点数)

为什么面积比等于π/4?

4.4.2 蒙特卡洛模拟计算π

$$\frac{N_{\text{circle}}}{N_{\text{square}}} \approx \frac{A_{\text{circle}}}{A_{\text{square}}} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}$$



$$\pi \approx \frac{4N_{\text{circle}}}{N_{\text{square}}}$$

当我们随机投掷大量的点时，落在圆内的点数与总点数的比例会接近这个面积比：

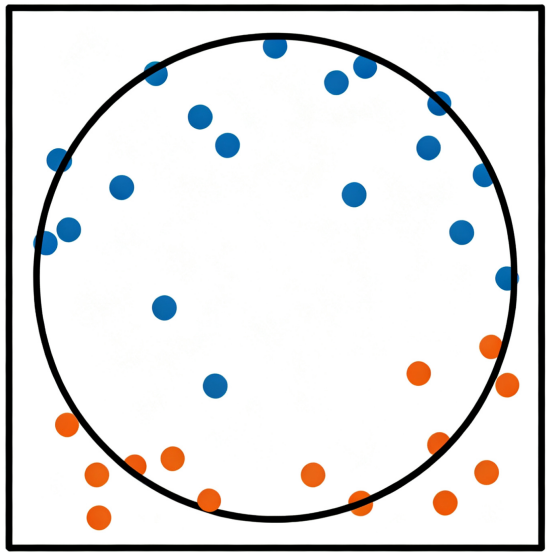
- 圆内点数 / 总点数 ≈ π / 4
- 因此：π ≈ 4 × (圆内点数 / 总点数)

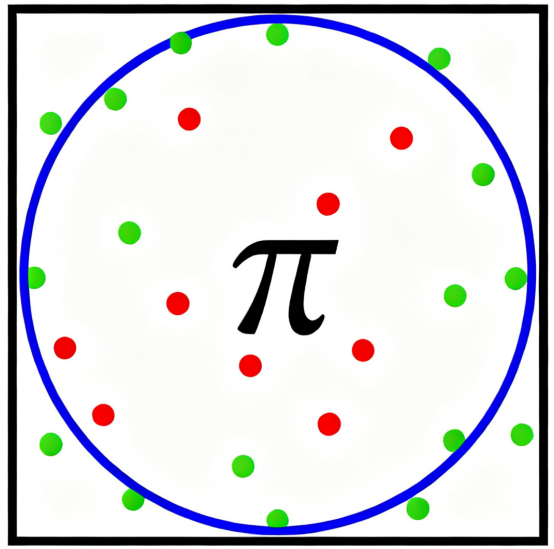
随机点数越多，估计值就越接近真实的π值。这就是蒙特卡洛方法的核心思想：通过大量随机采样来估计数学问题的答案。

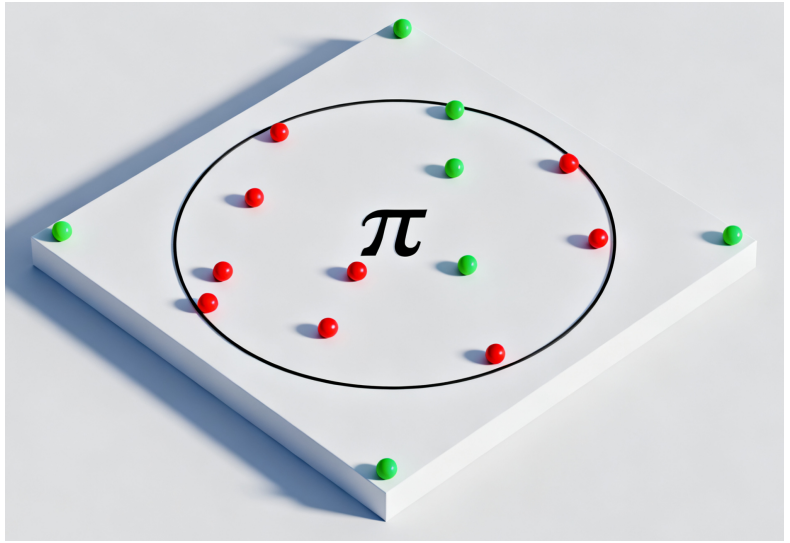
4.4.2 蒙特卡洛模拟计算π

Ask AI：想要更直观地理解蒙特卡洛方法吗?

你可以向 “AI图像生成助手”提问： "请画一个蒙特卡洛计算π值的示意图”







4.4.2 蒙特卡洛模拟计算π

```
import random
import math

def monte_carlo_pi(n_points):
    """使用蒙特卡洛方法计算π"""
    inside_circle = 0

    for _ in range(n_points):
        # 生成(-1, 1)范围内的随机点
        x = random.uniform(-1, 1)
        y = random.uniform(-1, 1)

        # 检查点是否在单位圆内
        if x*x + y*y <= 1:
            inside_circle += 1

    # 计算π的估计值
    pi_estimate = 4 * inside_circle / n_points
    return pi_estimate
```

4.4.2 蒙特卡洛模拟计算π

```
## 测试不同样本数量的精度
for n in [10000, 100000, 1000000]:
    pi_est = monte_carlo_pi(n)
    error = abs(pi_est - math.pi)
    print(f"样本数: {n}, π估计: {pi_est:.6f}, 误差: {error:.6f}")


## 输出结果：
## 样本数:    10000, π估计: 3.131600, 误差: 0.009993
## 样本数:   100000, π估计: 3.146040, 误差: 0.004447
## 样本数:  1000000, π估计: 3.143556, 误差: 0.001963
```

4.4.3 Ask AI：探索更多随机函数应用

当你想要深入探索random模块的更多功能时，可以向AI助手提出以下问题：

- "如何使用random模块进行数据采样和统计分析？"
- "除了计算π，蒙特卡洛方法还能解决哪些问题？"
- "如何生成符合特定分布的随机数（正态分布、指数分布等）？"
- "在游戏开发中如何使用随机数创造更好的体验？"
- "如何确保随机数的质量和安全性？"

实践练习

练习 4.4.1：掷骰子统计

模拟掷两个骰子1000次，统计各种点数和（2-12）的出现频率。

练习 4.4.2：随机颜色生成器

创建一个随机颜色生成器，能够生成RGB颜色值或十六进制颜色代码。

练习 4.4.3：简单抽奖程序

设计一个抽奖程序，设置不同奖项的中奖概率，模拟抽奖过程。