第五次作业

姓名

2025 年 1 月 2 日

复习题

第二题

PageRank 算法的设计思想基于对网页链接结构的深入分析。它将互联网视为一个有向图，其中网页是节点，超链接是有向边。通过构建随机游走模型（一阶马尔可夫链），算法能够量化网页的重要性。随机游走模型描述了用户如何沿着链接浏览网页，最终收敛到一个平稳分布。在这个分布中，每个网页被访问的概率即为其 PageRank 值，代表其重要性。PageRank 是递归定义的，一个网页的权重取决于链接到它的其他网页的权重，因此需要通过迭代计算来不断更新权重值。为了模拟用户随机跳转的行为，算法引入了阻尼因子（通常为 0.85），表示用户有一定概率随机跳转到任意网页，而非按链接浏览。这使得算法更符合实际用户行为，并提高了鲁棒性。最终，PageRank 通过量化网页的重要性，为搜索引擎提供了一种评估网页在搜索结果中排名的方法，从而提升搜索结果的质量和相关性。

第三题

贝叶斯定理是概率论中的重要定理，用于描述随机事件 A 和 B 的条件概率关系。其数学表达式为：

$$P(A|B) = \frac{P(A) \times P(B|A)}{P(B)}$$

其中，$P(A|B)$ 是在 B 发生的情况下 A 发生的概率，称为后验概率；$P(A)$ 是 A 的先验概率，即不考虑 B 时 A 发生的概率；$P(B|A)$ 是已知 A 发生后 B 的条件概率；$P(B)$ 是 B 的先验概率。贝叶斯定理在多个领域有广泛应用。在医疗诊断中，它可以帮助医生根据病人的症状和检测结果估计疾病的概率。在机器学习中，它是贝叶斯分类器和贝叶斯网络的基础，用于处理不确定性和噪声数据。在经济学中，贝叶斯方法可用于分析市场行为和预测经济变量。此外，贝叶斯定理还被用于信号处理、搜索引擎优化、财务分析和法律证据评估等领域。通过不断更新后验概率，贝叶斯定理使得模型能够适应新的观测数据，从而提高预测和决策的准确性。

第四题

蒙特卡罗方法是一种基于随机抽样的数值模拟方法，其核心思想是通过大量随机样本的采样来估计问题的解。该方法特别适用于无法用精确数学公式求解或求解非常困难的问题。蒙特卡罗方法的基本原理包括随机抽样、模拟计算和统计分析。首先，在定义的概率空间中生成随机样本，样本数量决定了结果的精度。然后，将这些样本代入目标函数，统计满足条件的样本数目，从而得到目标函数的估计值。最后，利用大数定律和中心极限定理，计算期望值、方差等统计量，并进行进一

步分析。蒙特卡罗方法在物理学、金融工程、计算数学、机器学习和生物医学等领域有广泛应用。例如，在物理学中，它可以模拟物质在不同条件下的行为；在金融工程中，它可以模拟股票价格的随机波动；在机器学习中，它可以用于训练神经网络。尽管蒙特卡罗方法简单易懂且适用性广，但其计算结果可能存在误差，因此在实际应用中需要合理选择样本数量和计算精度。

第五题

梯度下降法是一种用于寻找函数最小值（或局部最小值）的优化算法。其基本原理是通过迭代更新参数，使得目标函数的值逐渐减小。梯度下降法的核心思想是利用梯度信息确定移动方向。梯度是一个矢量，指向函数值增长最快的方向，因此梯度下降法沿着梯度的反方向（即下坡方向）移动。每次移动的步长由学习率决定，学习率过大会导致在最低点附近振荡，过小则会导致收敛速度过慢。算法从一个初始点开始，计算该点的梯度，并根据梯度和学习率更新位置。重复这一过程，直到满足终止条件，如梯度值小于某个阈值或达到预设的迭代次数。梯度下降法在机器学习和数值优化中广泛应用，但其可能找到局部最小值而非全局最小值，且初始点和学习率的选择对结果有重要影响。尽管如此，梯度下降法仍然是一种简单而有效的优化算法。

践习题

第一题

```
import numpy as np
import matplotlib.pyplot as plt

# 生成 100 个符合标准正态分布的样本
samples = np.random.normal(0, 1, 100)
print(samples)  # 打印这些样本

# 绘制直方图
plt.hist(samples, bins=20, density=True, alpha=0.6, color='g')

# 设置图表标题和坐标轴标签
plt.title('Standard Normal Distribution Samples')
plt.xlabel('Value')
plt.ylabel('Density')

# 显示图形
plt.show()
```

第二题

```
import numpy as np
import matplotlib.pyplot as plt

# 生成 100 个符合标准正态分布的样本
```

```
samples = np.random.normal(0, 1, 100)
print(samples)  # 打印这些样本

# 绘制直方图
plt.hist(samples, bins=20, density=True, alpha=0.6, color='g')

# 设置图表标题和坐标轴标签
plt.title('Standard Normal Distribution Samples')
plt.xlabel('Value')
plt.ylabel('Density')

# 显示图形
plt.show()
```

第三题

```
import numpy as np

# 输入四个整数代表二阶矩阵
a11, a12, a21, a22 = map(int, input("请输入四个整数，代表二阶矩阵（按顺序为 11、12、21、22 位置

matrix = np.array([[a11, a12], [a21, a22]])

eigenvalues, eigenvectors = np.linalg.eig(matrix)

print("特征值为：", eigenvalues)
print("特征向量为：")
for i in range(len(eigenvalues)):
    print(eigenvectors[:, i])
```

第五题

```
import numpy as np

data = np.array([[1, 2, 3],
                 [1, -1, 4],
                 [2, 1, 3],
                 [1, 3, -1]])

# 去掉第一行的标识行
data = data[1:]

covariance_matrix = np.cov(data)
```

```
print("协方差矩阵为：")
print(covariance_matrix)
```

补充题

```
import numpy as np
import matplotlib.pyplot as plt

def function(x):
    return 0.25 * (x - 0.5)**2 + 1

def derivative(x):
    return 0.5 * (x - 0.5)

x = np.linspace(-2, 2, 400)
y = function(x)

# 梯度下降参数
learning_rate = 0.1
num_iterations = 100
current_x = 1.5
iterations = []
values = []

for i in range(num_iterations):
    gradient = derivative(current_x)
    current_x = current_x - learning_rate * gradient
    iterations.append(i)
    values.append(current_x)
    print(f"Iteration {i}: x = {current_x}")

print(f"Final result: x = {current_x}")

plt.plot(x, y, label='Function')
plt.scatter(values, function(np.array(values)), c='r', label='Iteration Points')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Gradient Descent for Function 0.25*(x - 0.5)^2 + 1')
plt.legend()
plt.show()
```