第五次作业

复习题

2 PageRank 的设计思想是什么?

PageRank是著名的网络搜索引擎谷歌用于评测个网页“重要性”或者“影响力”的一种方法。
PageRank的设计思想主要基于以下几点：

1. **链接的重要性：** PageRank认为一个网页的重要程度可以通过它从其他网页获得的链接数量来衡量。如果一个网页得到了很多其他网页的链接，那么这个网页就可能是一个重要的网页。
2. **链接质量的影响：** 不仅仅是链接的数量重要，链接的质量同样重要。如果一个网页被许多高质量（即本身PageRank值高）的网页链接，那么这个网页的PageRank值也会更高。
3. **递归性：** PageRank的计算是一个递归的过程。每个网页的PageRank值不仅取决于指向它的网页数量和质量，还间接地依赖于这些网页本身的PageRank值。
4. **随机冲浪模型：** PageRank假设了一个“随机冲浪者”模型，即用户在浏览网页时会随机点击页面上的链接，有时也会随机跳转到互联网上的任意页面。这一模型通过引入一个“跳跃概率”（通常设定为0.15），确保了所有网页都有一定的PageRank值，即使它们没有入链。

PageRank 的算法一共可以分为四步：第一步将互联网作为一个有向图并用邻接矩阵对其进行表示；第二步将该邻接矩阵转换为超链接矩阵；第三步求解该超链接矩阵的最大特征向量(如用幂迭代法)；第四步最后求得的特征向量中的值即为对应网页的 PageRank值。

3 贝叶斯定理的内容是什么？它又有哪些重要应用?

贝叶斯定理的内容

贝叶斯定理可以用以下公式表达：

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

这里：

- $P(A|B)$ 是在已知事件B发生的条件下，事件A发生的后验概率。
- $P(B|A)$ 是在已知事件A发生的条件下，事件B发生的条件概率。
- $P(A)$ 和 $P(B)$ 分别是事件A和事件B的先验概率。

应用领域

1. **医学诊断**：利用贝叶斯定理可以帮助医生根据病人的症状和测试结果评估病人患有某种疾病的可能性。
2. **垃圾邮件过滤**：电子邮件系统使用贝叶斯分类器来识别和过滤垃圾邮件。
3. **推荐系统**：在线平台如亚马逊和Netflix使用贝叶斯方法来预测用户可能感兴趣的项目。
4. **自然语言处理**：贝叶斯方法被用于文本分类、情感分析等任务中。

5. **机器学习**：在机器学习中，贝叶斯方法被用于构建模型，特别是贝叶斯网络，用于处理不确定性问题。
6. **金融风险评估**：银行和其他金融机构使用贝叶斯统计方法来评估贷款违约的风险。
7. **法律和法庭科学**：贝叶斯定理也被用于法律决策过程中，例如评估证据对案件结果的影响。

4 试阐述蒙特卡罗方法的基本原理

蒙特卡罗方法是一种基于随机抽样的数值计算方法，广泛应用于物理、化学、工程、经济、金融等多个领域。其基本原理是通过大量随机样本的模拟来求解复杂问题的近似解。这种方法特别适用于那些传统解析方法难以解决的问题，尤其是多维积分、优化问题和概率模型的仿真等。

基本原理

1. **随机抽样**：蒙特卡罗方法的核心是使用随机数来模拟系统的状态或行为。通过对目标系统进行大量的随机抽样，可以近似地了解该系统的性质。
2. **统计估计**：通过对抽样结果进行统计分析，可以得到目标量的估计值。随着样本数量的增加，估计值的准确性通常会提高。
3. **误差分析**：蒙特卡罗方法的一个重要方面是能够估计结果的不确定性或误差范围。这通常通过计算标准误差或其他统计指标来实现。

具体步骤

1. **定义问题**：明确需要解决的问题或求解的目标量。
2. **建立模型**：根据问题的特点，建立相应的数学模型，确定需要抽样的参数及其分布。
3. **生成随机样本**：使用伪随机数生成器产生符合所需分布的随机样本。
4. **执行模拟**：根据生成的随机样本，运行模拟实验，记录每次试验的结果。
5. **统计分析**：对所有试验结果进行统计分析，计算目标量的期望值、方差等统计特征。
6. **评估误差**：通过计算标准误差等指标，评估结果的可靠性。
7. **优化与调整**：根据初步结果调整模型参数或增加样本数量，以提高精度和效率。

5 梯度下降法的主要思想是什么？你能用通俗的语言解释出来吗？

梯度下降法是一种常用的优化算法，主要用于寻找函数的最小值。它的主要思想非常直观，可以用一个简单的比喻来解释：想象你站在一座山的山顶，想要找到山脚下的最低点。梯度下降法就像是你在山顶上闭着眼睛，通过感知脚下坡度的方向，一步步向下走，直到走到最低点。

1. **定义目标函数**：首先，你需要有一个目标函数，这个函数描述了你要优化的目标。比如，在机器学习中，这个函数可能是损失函数，表示模型预测值与真实值之间的差距。
2. **初始位置**：选择一个起点，即初始参数值。这相当于你在山顶的位置。
3. **计算梯度**：梯度是函数在某一点上的导数（对于多维函数，是偏导数组成的向量）。梯度的方向指出了函数值增加最快的方向。因此，负梯度方向就是函数值减少最快的方向。这相当于你用脚感受地面的坡度，确定哪个方向最陡峭。
4. **更新参数**：沿着负梯度方向，以一个小步长（学习率）移动参数值。这相当于你沿着最陡峭的方向迈出一小步。步长的选择非常重要，太大会导致错过最低点，太小则收敛速度慢。
5. **重复迭代**：不断重复计算梯度和更新参数的过程，直到达到一个停止条件，比如参数变化非常小或者达到最大迭代次数。这相当于你不断地走，直到你觉得已经到达最低点。

践习题

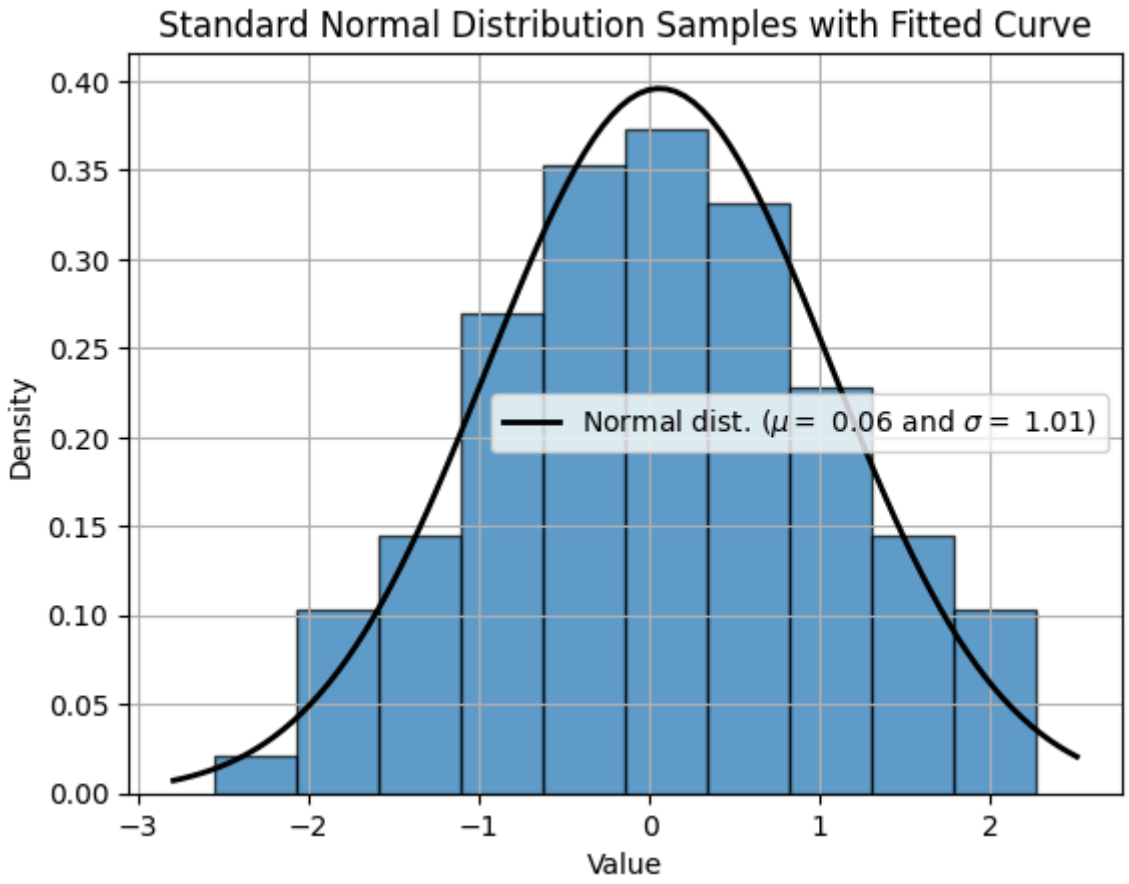
1 使用 numpy 生成服从标准正态分布的 100 个样本

2 通过 Python 程序为抽样出的样本绘图展示

代码

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# 设置随机种子以确保结果可复现（可选）
np.random.seed(0)

# 生成100个服从标准正态分布的样本
samples = np.random.normal(loc=0.0, scale=1.0, size=100)

# 绘制直方图
plt.hist(samples, bins=10, density=True, edgecolor='black', alpha=0.7)

# 计算并绘制正态分布曲线
mu, std = norm.fit(samples)  # 计算样本的均值和标准差
xmin, xmax = plt.xlim()  # 获取直方图的x轴范围
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)  # 计算正态分布的概率密度函数值
plt.plot(x, p, 'k', linewidth=2)

# 添加图例
plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$ {:.2f})'.format(mu, std)],
loc='best')

# 设置图表标题和标签
plt.title('Standard Normal Distribution Samples with Fitted Curve')
plt.xlabel('Value')
plt.ylabel('Density')

# 显示网格
plt.grid(True)

# 显示图形
plt.show()
```

运行结果：



3 通过python程序计算矩阵$A=\begin{pmatrix}2 & 1\\4 & 5\end{pmatrix}$的特征值和特征向量。

代码

```python
import numpy as np

# 定义矩阵 A
A = np.array([[2, 1], [4, 5]])

# 计算特征值和特征向量
eigenvalues, eigenvectors = np.linalg.eig(A)

# 输出特征值和对应的特征向量
for i in range(len(eigenvalues)):
    print(f"特征值 {i+1}: {eigenvalues[i]:.2f}")
    print("对应的特征向量：")
    print(eigenvectors[:, i])
    print("\n")
```

运行结果

```
特征值 1: 1.00
对应的特征向量:
[-0.70710678  0.70710678]


特征值 2: 6.00
对应的特征向量:
[-0.24253563 -0.9701425 ]
```

5 通过python计算给定矩阵的协方差矩阵

Data	1	2	3
X	1	-1	4
Y	2	1	3
Z	1	3	-1

代码

```python
import numpy as np

# 数据转换为 NumPy 数组
data = np.array([[1, -1, 4],
                 [2, 1, 3],
                 [1, 3, -1]])

# 计算协方差矩阵
cov_matrix = np.cov(data)

# 输出协方差矩阵
print(cov_matrix)
```

运行结果

```
[[ 6.33333333  2.5        -5.        ]
 [ 2.5         1.         -2.        ]
 [-5.         -2.          4.        ]]
```

使用梯度下降法求解函数 $f(x) = 0.25 \cdot (x - 0.5)^2 + 1$ 的局部极小值

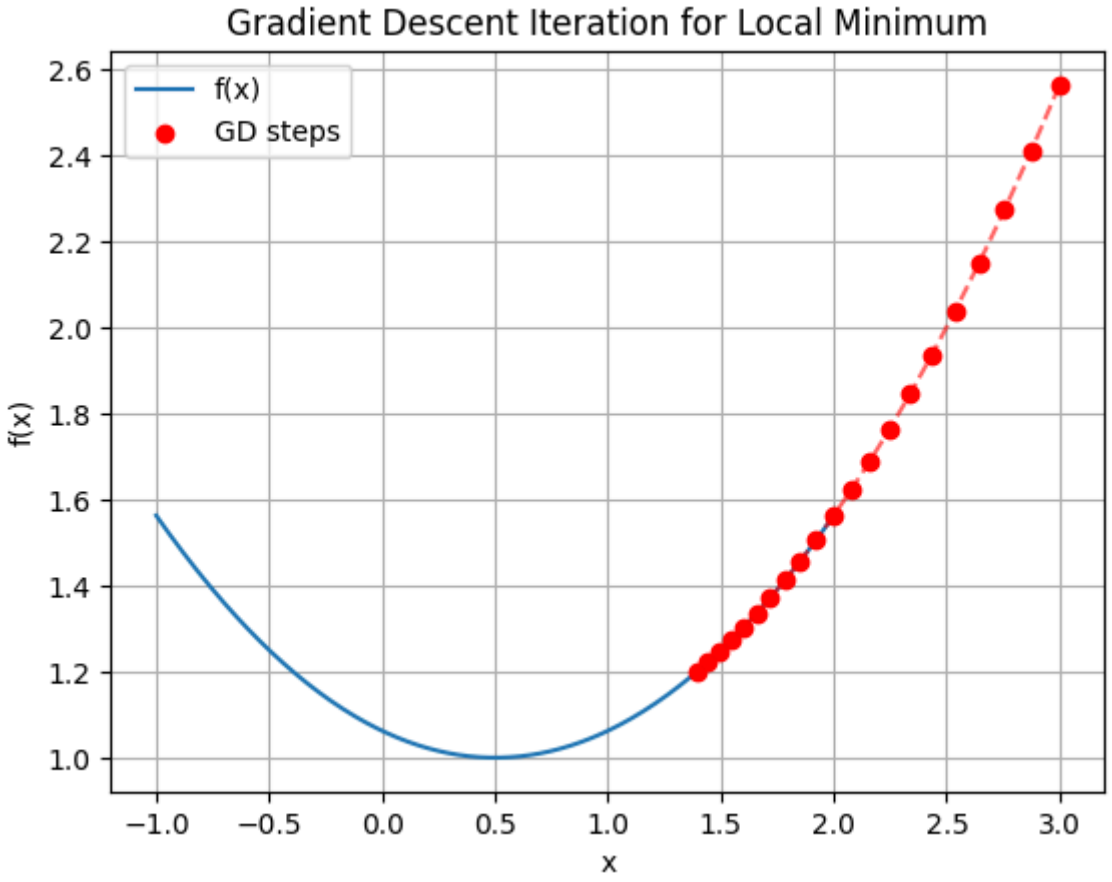
导数:

$$f'(x) = 0.25 \cdot 2 \cdot (x - 0.5) = 0.5 \cdot (x - 0.5)$$

代码

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

# 定义目标函数
def f(x):
    return 0.25 * (x - 0.5)**2 + 1

# 定义函数的导数
def f_prime(x):
    return 0.5 * (x - 0.5)

# 梯度下降算法
def gradient_descent(starting_point, learning_rate, num_iterations):
    x_values = [starting_point]
    x = starting_point
    for _ in range(num_iterations):
        grad = f_prime(x)
        x = x - learning_rate * grad
        x_values.append(x)
    return x_values

# 初始化参数
starting_point = 3  # 初始x值
learning_rate = 0.1  # 学习率
num_iterations = 20  # 迭代次数

# 运行梯度下降算法
x_values = gradient_descent(starting_point, learning_rate, num_iterations)

# 绘图展示迭代过程
x_range = np.linspace(-1, 2, 400)
y_range = f(x_range)

plt.plot(x_range, y_range, label="f(x)")
plt.scatter(x_values, f(np.array(x_values)), color='red', label="GD steps",
zorder=5)
plt.plot(x_values, f(np.array(x_values)), color='red', linestyle='dashed',
alpha=0.6)
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Gradient Descent Iteration for Local Minimum")
plt.legend()
plt.grid(True)
plt.show()
```

运行结果


Gradient Descent Iteration for Local Minimum
f(x)
GD steps
f(x)
x
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0