1. **PageRank的设计思想是什么?**

**PageRank的设计思想是基于网页之间的链接关系来衡量网页的重要性。如果一个网页有很多其他网页链接到它，那这个网页可能更重要。而且，来自重要网页的链接会比来自普通网页的链接更有“权重”。**

**2、贝叶斯定理的内容是什么?它又有哪些重要应用?**

**贝叶斯定理是用来计算在已知条件下某个事件发生的概率，核心思想是通过已有的信息更新对事件发生概率的估计。**

**P(A∣B)=P(B∣A)⋅P(A)P(B)P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}P(A∣B)=P(B)P(B∣A)⋅P(A)**

**在医疗诊断中，可以计算检测结果呈阳性时，患者实际患病的概率，这对医生的判断非常有帮助。**

**在金融和经济学中，贝叶斯定理常用于处理不确定性，帮助评估市场走势和投资风险。**

**3、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技术，广泛应用于解决复杂的数学问题和进行概率模拟。**

**这种方法的核心在于使用随机数生成器，随机生成大量数据点以代表整个样本空间。这种方式的优势在于能够在无法精确计算的情况下，利用“通过随机生成的数据，近似估计所需的量”。**

1. **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?**

**梯度下降法是一种优化算法，常用于机器学习和深度学习中，主要目标是最小化损失函数。想象自己站在一个山谷的边缘，周围的地形崎岖不平。我们希望**

**找到山谷的最低点，也就是最优解。首先，从一个随机选择的起始点开始，然后计算这个点的梯度。梯度指向了函数上升最快的方向，就像站在山顶时，指向最高处的箭头。接下来，我们需要沿着梯度的相反方向移动，也就是往下坡的方向走。这一步的大小由一个叫学习率的参数决定，学习率就像是你每次**

**下坡的步伐大小。步伐太大可能会让你跌倒，步伐太小则会让你走得很慢。通过不断重复这个过程，计算新的点的梯度并调整位置，我们会逐渐接近山谷的底部，找到函数的最低点，也就是损失函数的最小值。最终，我们的目标是找到最佳参数，使得模型的预测更加准确。**

1. **使用 numpy 生成服从标准正态分布的 100个样本。**

import numpy as np

# 设置随机种子以便复现结果

np.random.seed(42)

# 生成100个服从标准正态分布的样本

samples = np.random.normal(loc=0, scale=1, size=100)

# 打印生成的样本

print(samples)

**2、通过 Python 程序为抽样出的样本绘图展示**

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# 设置随机种子以便复现结果

np.random.seed(42)

# 生成100个服从标准正态分布的样本

samples = np.random.normal(loc=0, scale=1, size=100)

# 设置绘图风格

sns.set(style="whitegrid")

# 绘制直方图和密度图

plt.figure(figsize=(10, 6))

sns.histplot(samples, bins=15, kde=True, stat='density', color='blue',

edgecolor='black')

# 添加标题和标签

plt.title('Histogram and Density Plot of Standard Normal Distribution Samples',

fontsize=16)

plt.xlabel('Sample Value', fontsize=14)

plt.ylabel('Density', fontsize=14)

plt.xlim(-4, 4) # 设置x轴范围

# 显示图形

plt.show()

**3、通过 Python 程序计算矩阵 (2145)(2415) 的特征值和特征向量。**

import numpy as np

# 定义矩阵

matrix = np.array([[2, 1], [4, 5]])

# 计算特征值和特征向量

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

print("特征值:", eigenvalues)

print("特征向量:\n", eigenvectors)

import numpy as np

# 定义数据矩阵

data\_matrix = np.array([[1, 2, 3], # 第一行：data

[1, -1, 4], # 第二行：X

[2, 1, 3], # 第三行：Y

[1, 3, -1]]) # 第四行：Z

# 计算协方差矩阵

cov\_matrix = np.cov(data\_matrix, rowvar=False) # rowvar=False 表示每列代表一个变量

# 打印协方差矩阵

print("协方差矩阵:\n", cov\_matrix)