数据科学导论第六周作业

复习题2：

数量假设：一个网页如果被许多其他网页链接到，那么这个网页比较重要，即PageRank值会相对较高。这是因为链接可以被视为一种“投票”，链接越多，表明越多网页认为该网页重要。

质量假设：一个PageRank值很高的网页链接到另一个网页，会使得被链接的网页的PageRank值相应地提高。这意味着链接的质量也很重要，一个来自权威网站的链接比来自一个不知名网站的链接更有价值。

出链和入链：PageRank算法通过网页之间的链接来决定网页的重要性，这在一定程度上消除了人为对排名的影响。同时，PageRank值的计算是离线进行的，而非在查找时计算，这样可以提升查询效率。

随机浏览者模型：PageRank算法引入了随机浏览者（random surfer）的概念，即假设一个用户在浏览器中随机打开某些页面并点击了某些链接。这个模型考虑了用户在浏览网页时的随机性，通过模拟用户的浏览行为来确定网页的重要性。

阻尼系数：为了处理那些没有外部链接的页面（也称为“死胡同”或“挂起节点”）所带来的问题，PageRank算法引入了阻尼系数（damping factor），通常取值为0.85。这意味着用户在任意时刻继续访问下一个页面的概率是0.85，而随机跳转到网络上任何一个其他页面的概率是0.15。

迭代计算：PageRank值的计算是一个迭代过程，通过不断迭代更新每个页面的PageRank值，直到这些值趋于稳定。这个迭代过程反映了网页之间相互链接的复杂关系，并最终达到一个平衡状态。

复习题3:

贝叶斯定理（Bayes' Theorem），是概率论中一个重要的定理，描述了在给定事件发生的情况下，另一个事件发生的概率。贝叶斯定理的公式为：

P(A|B) = P(B|A) \* P(A) / P(B)

其中：

P(A|B) 表示在事件 B 发生的条件下事件 A 发生的概率

P(B|A) 表示在事件 A 发生的条件下事件 B 发生的概率

P(A) 表示事件 A 发生的概率

P(B) 表示事件 B 发生的概率 贝叶斯定理也可以用条件概率的格式来表示： P(A|B) = P(B|A) \* P(A) / [P(B|A) \* P(A) + P(B|A') \* P(A')] 其中 A' 表示 A 的对立事件，即 A 不发生的情况。 片段2: 贝叶斯定理的应用非常广泛，涉及各个领域：

医学诊断：根据病人的症状和化验结果，计算疾病发生的概率。

机器学习：在分类算法中，用于根据特征预测类别的概率。

自然语言处理：在语言模型中，用于计算词、短语或句子出现的概率。

信号处理：在滤波器中，用于根据观测数据估计信号的概率。

金融风控：在信用评估中，用于计算违约发生的概率。

法律证据：在法庭审判中，用于根据证据计算嫌疑人有罪或无罪的概率。

质量控制：在缺陷检测中，用于计算产品缺陷发生的概率。

人工智能：在专家系统中，用于根据知识库推理和决策。

复习题4：

蒙特卡罗方法是一种基于随机抽样的计算方法，用于解决数学、物理、工程、金融等领域的各种问题。其基本原理可以总结如下：

随机抽样：蒙特卡罗方法的核心在于随机抽样。在处理具体问题时，通过从相关的概率分布中随机抽取样本点，来近似模拟系统的随机行为或不确定性。

统计估计：通过对这些随机样本的统计分析，可以得到问题的统计估计。例如，通过样本的平均值来估计期望值，或者通过样本的分布来估计系统的概率分布。

大数定律：蒙特卡罗方法依赖于大数定律，即随着随机样本数量的增加，样本的平均值会趋近于真实值。因此，通过增加样本数量可以提高估计的准确性。

复习题5：

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。它的主要思想可以用这样一个类比来通俗解释：

想象你正站在一座山谷的最高点，你的目标是找到山谷的最低点，那里藏有宝藏。但是，山谷被雾气笼罩，你看不见前方的路，只能感知当前位置的坡度（梯度）。梯度下降法就是教你如何一步步沿着最陡的下坡路，走向山谷的最低点。

具体步骤如下：

选择一个起点：在山谷的任意位置选择一个起点，这可以是完全随机的。

计算梯度：在当前位置，测量地面的坡度（即计算函数在该点的梯度）。梯度是一个向量，指向函数增长最快的方向。

选择下坡方向：为了下山，你需要选择与梯度相反的方向，因为那是下降最快的方向。

确定步长：决定你每步要走多远。步长太大，可能会一步越过最低点，甚至滑回到山坡上；步长太小，进展缓慢。

更新位置：根据下坡方向和步长，更新你的位置，向山谷底部更近一步。

重复这个过程：在新的位置重复上述步骤，直到你到达最低点，或者达到一定的迭代次数后停止。

践行题1：

import numpy as np

np.random.seed(0)

samples = np.random.randn(100)

print(samples)

结果：

1.76405235 0.40015721 0.97873798 2.2408932 1.86755799 -0.97727788

0.95008842 -0.15135721 -0.10321885 0.4105985 0.14404357 1.45427351

0.76103773 0.12167502 0.44386323 0.33367433 1.49407907 -0.20515826

0.3130677 -0.85409574 -2.55298982 0.6536186 0.8644362 -0.74216502

2.26975462 -1.45436567 0.04575852 -0.18718385 1.53277921 1.46935877

0.15494743 0.37816252 -0.88778575 -1.98079647 -0.34791215 0.15634897

1.23029068 1.20237985 -0.38732682 -0.30230275 -1.04855297 -1.42001794

-1.70627019 1.9507754 -0.50965218 -0.4380743 -1.25279536 0.77749036

-1.61389785 -0.21274028 -0.89546656 0.3869025 -0.51080514 -1.18063218

-0.02818223 0.42833187 0.06651722 0.3024719 -0.63432209 -0.36274117

-0.67246045 -0.35955316 -0.81314628 -1.7262826 0.17742614 -0.40178094

-1.63019835 0.46278226 -0.90729836 0.0519454 0.72909056 0.12898291

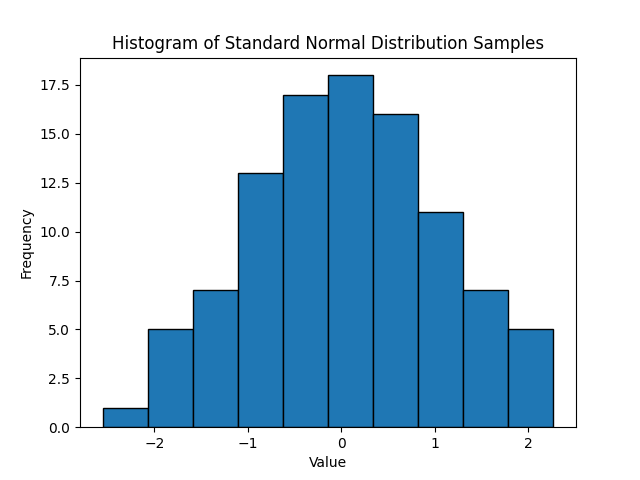
1.13940068 -1.23482582 0.40234164 -0.68481009 -0.87079715 -0.57884966

-0.31155253 0.05616534 -1.16514984 0.90082649 0.46566244 -1.53624369

1.48825219 1.89588918 1.17877957 -0.17992484 -1.07075262 1.05445173

-0.40317695 1.22244507 0.20827498 0.97663904 0.3563664 0.70657317

0.01050002 1.78587049 0.12691209 0.40198936

践行题2：

践行题3：

import numpy as np

# 定义一个2x2矩阵

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

# 计算特征值

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

print("特征值:")

print(eigenvalues)

print("特征向量:")

print(eigenvectors)

特征值:

[1. 6.]

特征向量:

[[-0.70710678 -0.24253563]

 [ 0.70710678 -0.9701425 ]]

践行题5：

import numpy as np

# 数据矩阵

data = np.array([

    [1, -1, 4],

    [2, 1, 3],

    [1, 3, -1]

])

# 计算协方差矩阵

cov\_matrix = np.cov(data, rowvar=True)

print("协方差矩阵 C:")

print(cov\_matrix)

协方差矩阵 C:

[[ 6.33333333  2.5        -5.        ]

 [ 2.5         1.         -2.        ]

 [-5.         -2.          4.        ]]

额外：

import numpy as np

import matplotlib.pyplot as plt

# 定义函数及其导数

def f(x):

    return 0.25 \* (x - 0.5)\*\*2 + 1

def f\_prime(x):

    return 0.5 \* (x - 0.5)

# 选择初始点和学习率

x = 0

alpha = 0.1

max\_iter = 30

tolerance = 1e-6

# 存储迭代过程中的x值

x\_values = [x]

# 梯度下降

for i in range(max\_iter):

    grad = f\_prime(x)

    x\_new = x - alpha \* grad

    x\_values.append(x\_new)

    x = x\_new

    if abs(x\_new - x) < tolerance:

        break

# 打印结果

print(f"局部极小值点: {x}")

print(f"函数的局部极小值: {f(x)}")

# 绘制迭代过程

plt.plot(x\_values, [f(x\_val) for x\_val in x\_values], marker='o')

plt.title('Gradient Descent Iteration Process')

plt.xlabel('x')

plt.ylabel('f(x)')

plt.show()

局部极小值点: 0.025

函数的局部极小值: 1.05640625

