2024.10.14作业

复习题

- T2

  PageRank 的设计思想基于网页之间的链接结构，通过模拟随机游走者在网络中浏览网页来评估网页的重要性。它认为一个网页的价值不仅取决于自身的链接数量，还受指向它的网页质量影响。算法通过迭代计算每个网页的排名，直到结果收敛，并引入阻尼因子（通常为 0.85）以模拟用户随机跳转的行为。这种方法有效地反映了网页在网络中的重要性，广泛应用于搜索引擎和网络分析中。

- T3

  贝叶斯定理描述了条件概率之间的关系，它提供了如何基于已知信息（事件 B）更新事件 A 的概率（后验概率）的公式。

  应用：机器学习——作为朴素贝叶斯分类器的基础，广泛应用于文本分类、垃圾邮件过滤和情感分析等任务。

  贝叶斯定理在不确定性条件下的推理和决策中具有重要意义。

- T4

  蒙特卡罗方法是一种基于随机采样的计算技术，通过生成大量随机样本来估计复杂问题的数值解或概率分布。它依赖于统计分析，根据样本分布评估目标函数，从而得到结果的估计，并随着样本数量的增加，估计结果会趋于真实值。该方法广泛应用于金融、物理、工程和运筹学等领域，特别适合于高维和复杂问题的求解。

- T5

  梯度下降法是一种优化算法，旨在寻找函数的最小值。可以想象你在山谷中，目标是找到最低点。在当前位置，你查看周围的坡度（梯度），找到向下走的最快方向，然后沿着这个方向走一小步。接着，你重复这个过程，重新检查坡度，继续前进，直到找不到更低的点为止。这个方法逐步逼近目标函数的最小值，广泛应用于机器学习和数据分析中优化模型参数。

践习题

- T1

```
import numpy as np
samples = np.random.normal(loc=0, scale=1, size=100)
print(samples)
```

- T2

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

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

plt.figure(figsize=(10, 6))
sns.histplot(samples, bins=10, kde=True, color='blue', stat='density',
linewidth=0)

xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
p = np.exp(-0.5 * x**2) / np.sqrt(2 * np.pi)  # 标准正态分布的 PDF
plt.plot(x, p, 'k', linewidth=2)

plt.title('Samples from Standard Normal Distribution')
plt.xlabel('Value')
plt.ylabel('Density')
plt.show()
```

- T3

```
import numpy as np

matrix = np.array([[2, 1],
                   [4, 5]])
eigenvalues, eigenvectors = np.linalg.eig(matrix)

print("特征值：")
print(eigenvalues)

print("特征向量：")
print(eigenvectors)
```

- T5

```
import numpy as np

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

cov_matrix = np.cov(data, rowvar=False)

print("协方差矩阵 C：")
print(cov_matrix)
```

- 补充

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

# 设置中文字体
rcParams['font.sans-serif'] = ['SimHei']
rcParams['axes.unicode_minus'] = False

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

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

def gradient_descent(starting_point, learning_rate, num_iterations):
```

```
    x_values = [starting_point]
    for _ in range(num_iterations):
        grad = gradient(x_values[-1])
        new_x = x_values[-1] - learning_rate * grad
        x_values.append(new_x)
    return x_values

starting_point = 0.0
learning_rate = 0.1
num_iterations = 20

x_values = gradient_descent(starting_point, learning_rate, num_iterations)

x = np.linspace(-1, 2, 100)
y = f(x)
plt.plot(x, y, label='f(x)', color='blue')
plt.scatter(x_values, f(np.array(x_values)), color='red', label='迭代过程',
zorder=5)
plt.plot(x_values, f(np.array(x_values)), color='red', linestyle='--',
linewidth=1, zorder=4)
plt.title('梯度下降法迭代过程')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.axhline(1, color='gray', linestyle='--', linewidth=0.7)
plt.axvline(0.5, color='gray', linestyle='--', linewidth=0.7)
plt.legend()
plt.grid()
plt.show()
```