1. 一个网页的重要性可以通过链接到它的其他网页的数量和质量来衡量。如果一个网页被很多其他网页链接，特别是被一些高质量的网页链接，那么这个网页通常被认为是重要的。
2. 对于两个事件A和B，若事件A和B满足条件概率的定义，即事件B非空，并且P(A) > 0，则有：P(A|B) = P(A) × P(B|A) / P(B)

应用：医疗诊断、机器学习、经济学

1. 蒙特卡罗方法的核心思想是利用随机数和概率统计方法来模拟问题，通过大量随机样本的采样，得到问题的概率分布或期望值。这种方法特别适用于那些无法用精确数学公式求解的问题，或者公式求解非常困难的问题。
2. 下山时在每个点都向向下最陡峭的方向走

践习题：

1、

import numpy as np

samples = np.random.randn(100)

print(samples)

2、

import numpy as np

import matplotlib.pyplot as plt

samples = np.random.randn(100)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

ax1.hist(samples, bins=30, edgecolor='black', alpha=0.7, density=True)

ax1.set\_title('Histogram of Samples')

ax1.set\_xlabel('Value')

ax1.set\_ylabel('Density')

ax2.plot(np.sort(samples), np.linspace(0, 1, len(samples)), marker='.', linestyle='none')

ax2.plot(np.linspace(-4, 4, 100), np.exp(-0.5 \* np.linspace(-4, 4, 100)\*\*2) / np.sqrt(2 \* np.pi) \* len(samples) \* 0.03, 'r-', lw=2, label='Standard Normal PDF')

kde = ax2.kdeplot(samples, shade=True, color='blue')

ax2.set\_title('Kernel Density Estimation (KDE) of Samples')

ax2.set\_xlabel('Value')

ax2.set\_ylabel('Density')

ax2.legend()

plt.tight\_layout()

plt.show()

3、

import numpy as np

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

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

print("特征值:")

print(eigenvalues)

print("\n特征向量:")

print(eigenvectors)

5、

import numpy as np

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

covariance\_matrix = np.cov(data, rowvar=False)

print("协方差矩阵:")

print(covariance\_matrix)

\*\*、

import numpy as np

import matplotlib.pyplot as plt

def f(x):

return 0.25 \* (x - 0.5)\*\*2 + 1

def grad\_f(x):

return 0.5 \* (x - 0.5)

alpha = 0.1

x\_init = 2.0

num\_iterations = 50

x = x\_init

x\_values = [x]

f\_values = [f(x)]

for i in range(num\_iterations):

grad = grad\_f(x)

x = x - alpha \* grad

x\_values.append(x)

f\_values.append(f(x))

x\_range = np.linspace(-1, 2, 400)

y\_range = f(x\_range)

plt.plot(x\_range, y\_range, label='f(x) = 0.25\*(x-0.5)^2 + 1')

plt.scatter(x\_values, f\_values, color='red', label='Gradient Descent Iterations')

plt.plot(x\_values, f\_values, 'ro-', markersize=2)

plt.xlabel('x')

plt.ylabel('f(x)')

plt.title('Gradient Descent for Local Minimum')

plt.legend()

plt.grid(True)

plt.show()

