复习题

2. 根据网站的外部链接和内部链接的数量和质量来衡量这个网站的价值，相当于每个到该页面的链接都是对该页面的一次投票，被链接得越多就意味着被其他网站投票越多

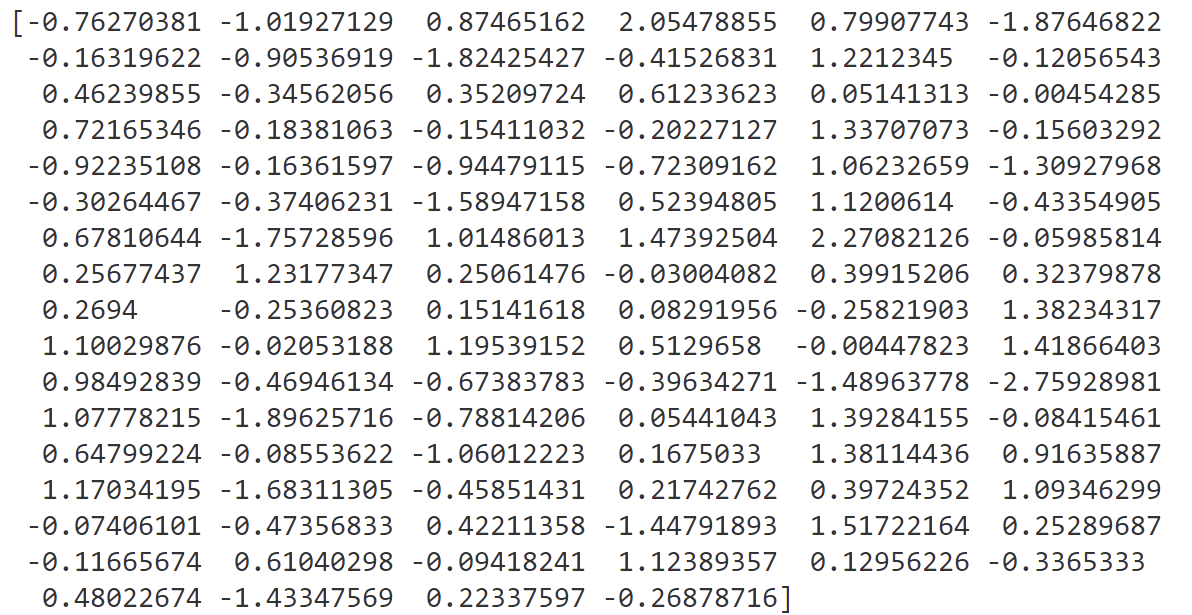
3. 贝叶斯定理描述了在一个条件概率下，两个事件之间的概率关系。这个定理提供了一种计算后验概率的方法，即在已知一些新的信息或证据后，更新对某一事件发生的信念或概率。已知A的概率是P(A)，B的概率是P(B),A发生时B发生的概率是P(B|A).那么就可以算出，B发生时A发生的概率P(A|B) =P(A)\*P(B|A)/P(B)。

应用：疾病发病率计算，预测模型优化，搜索引擎

4.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。

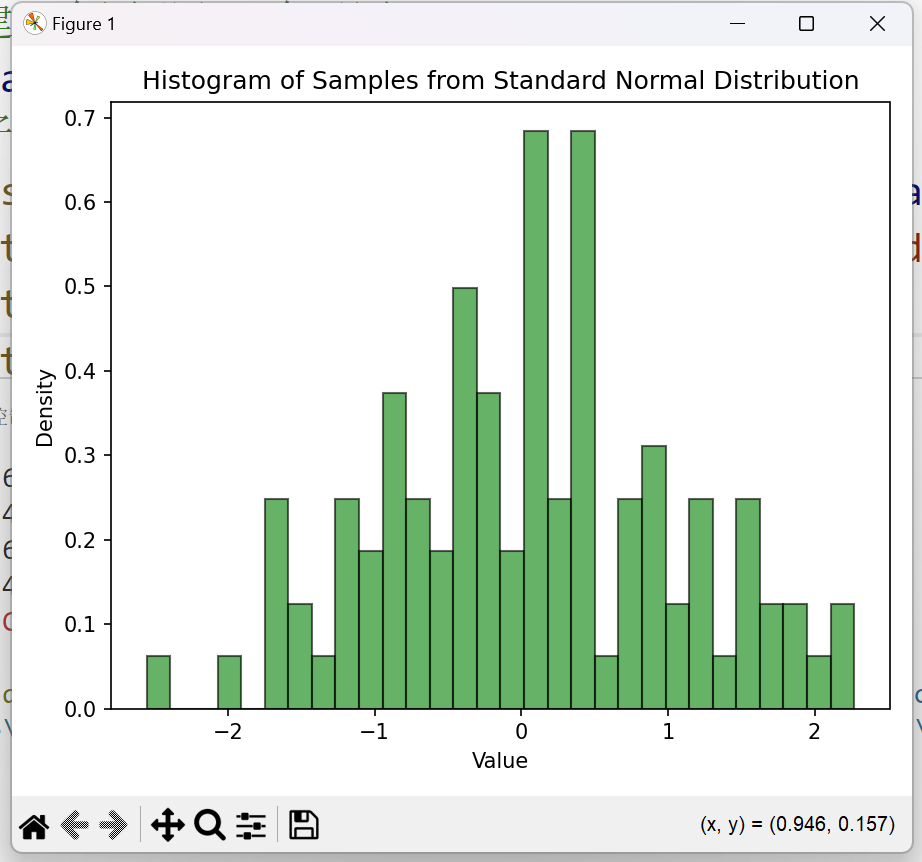
5. 梯度下降法的作用就是寻找一个极小值点，从而让函数的值尽可能地小。

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, ax = plt.subplots(figsize=(6, 5))

ax.hist(samples, bins=30, density=True, alpha=0.6, color='g', edgecolor='black')

ax.set\_title('Histogram of Samples from Standard Normal Distribution')

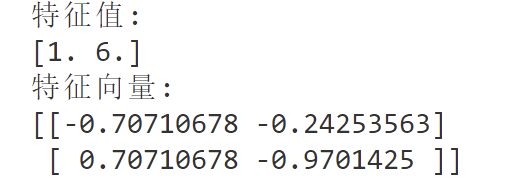
ax.set\_xlabel('Value')

ax.set\_ylabel('Density')

plt.tight\_layout()

plt.show()

3.



import numpy as np

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

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

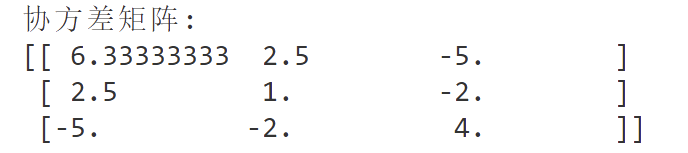
print("特征值:")

print(eigenvalues)

print("特征向量:")

print(eigenvectors)

5.



import numpy as np

data = np.array([

    [1, 2, 1],

    [-1, 1, 3],

    [4, 3, -1]

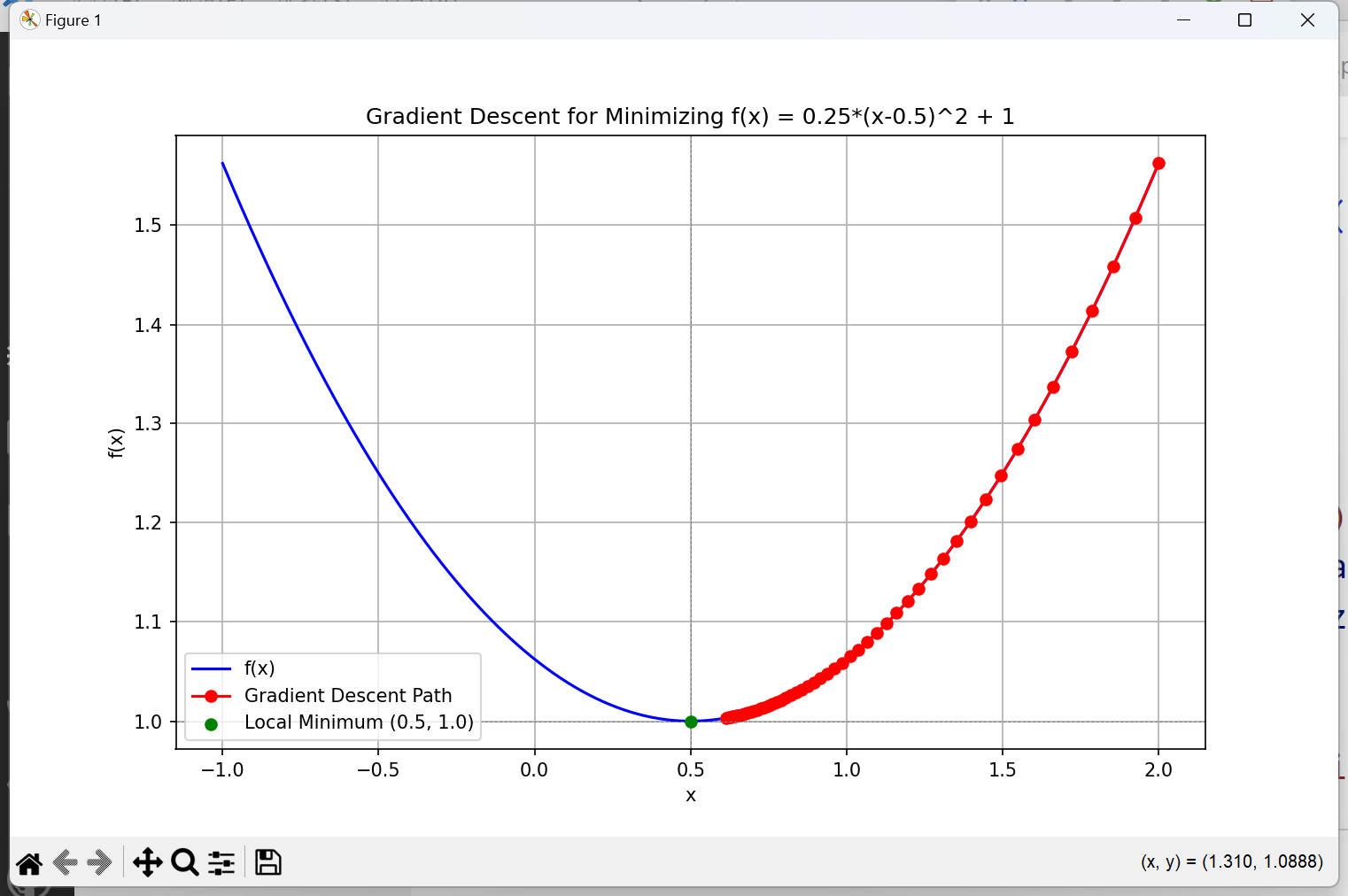
])

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

print("协方差矩阵:")

print(covariance\_matrix)

6.



import numpy as np

import matplotlib.pyplot as plt

def f(x):

    return 0.25 \* (x - 0.5)\*\*2 + 1

def df(x):

    return 0.5 \* (x - 0.5)

def gradient\_descent(start\_x, learning\_rate, num\_iterations):

    x = start\_x

    x\_values = [x]

    f\_values = [f(x)]

    for \_ in range(num\_iterations):

        grad = df(x)

        x -= learning\_rate \* grad

        x\_values.append(x)

        f\_values.append(f(x))

    return x\_values, f\_values

start\_x = 2.0

learning\_rate = 0.1

num\_iterations = 50

x\_values, f\_values = gradient\_descent(start\_x, learning\_rate, num\_iterations)

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

f\_range = f(x\_range)

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

plt.plot(x\_range, f\_range, label='f(x)', color='blue')

plt.plot(x\_values, f\_values, 'ro-', label='Gradient Descent Path')

plt.scatter(0.5, 1.0, color='green', zorder=5, label='Local Minimum (0.5, 1.0)')

plt.xlabel('x')

plt.ylabel('f(x)')

plt.title('Gradient Descent for Minimizing f(x) = 0.25\*(x-0.5)^2 + 1')

plt.legend()

plt.grid(True)

plt.axhline(1, color='grey', linestyle='--', linewidth=0.5)

plt.axvline(0.5, color='grey', linestyle='--', linewidth=0.5)

plt.show()