复习题

2.pagerank的主要设计思想涉及链接分析、随机游走模型和迭代计算。

其中链接分析又分为重要性假设和数量与质量并重；

随机游走模型包括马尔可夫链和平稳分布；

迭代计算需要初始化、迭代更新、阻尼因子和收敛

3.贝叶斯定理的内容是P(A|B)=P(B|A)\*P(A)/P(B)

应用：在机器学习中，贝叶斯定理被广泛应用于分类问题；在决策分析中，贝叶斯原理是一种重要的决策原则，用于计算在给定条件下采取某种行动的最佳选择。

4.蒙特卡洛方法的基本原理是通过数字模拟试验，得到所要求解的出现某种事件的概率，作为问题的近似解。

5.梯度下降法的主要思想是通过不断迭代沿着函数的梯度的反方向更新参数，以逐渐接近函数的最小值。

践习题

1.

```

import numpy as np

# 生成100个服从标准正态分布的样本

samples = np.random.randn(100)

print(samples)

```

2.

```

import numpy as np

import matplotlib.pyplot as plt

samples = np.random.randn(100) #100个样本

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

from scipy.stats import norm

xmin, xmax = plt.xlim()

x = np.linspace(xmin, xmax, 100)

p = norm.pdf(x, 0, 1)

plt.plot(x, p, 'k', linewidth=2)

title = "Fit results: mean = %.2f, std = %.2f" % (np.mean(samples), np.std(samples))

plt.title(title)

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("特征向量:")

print(eigenvectors)

```

5.

```

import numpy as np

X = np.array([1, -1, 4])

Y = np.array([2, 1, 3])

Z = np.array([1, 3, -1])

data\_matrix = np.column\_stack((X, Y, Z))

covariance\_matrix = np.cov(data\_matrix, 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 0.25 \* 2 \* (x - 0.5)

def gradient\_descent(initial\_x, learning\_rate, max\_iterations):

x = initial\_x

x\_values = [x]

f\_values = [f(x)]

for i in range(max\_iterations):

grad = grad\_f(x)

x = x - learning\_rate \* grad

x\_values.append(x)

f\_values.append(f(x))

return x\_values, f\_values

initial\_x = 2.0

learning\_rate = 0.1

max\_iterations = 50

x\_values, f\_values = gradient\_descent(initial\_x, learning\_rate, max\_iterations)

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

f\_range = f(x\_range)

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

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

plt.scatter(x\_values[-1], f\_values[-1], color='r', zorder=5)

plt.xlabel('x')

plt.ylabel('f(x)')

plt.legend()

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

plt.grid(True)

plt.show()

```