**《人工智能基础A》实验报告四**

**机器学习之无监督学习**

学号 姓名 联系方式 .

## 实验目的

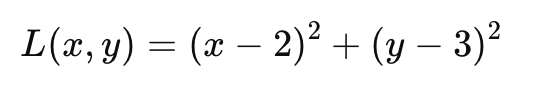
通过实验，理解梯度下降，学会应用无监督学习技术。具体目标要求如下：

1. 实现梯度下降算法。
2. 学会应用聚类算法对数据集进行聚类，探索数据的潜在分类结构。
3. 学会运用降维技术，便于数据可视化和理解。
4. 学会利用大模型生成代码。

## 实验内容及要求

### 梯度下降

梯度下降是一种经典而广泛应用的优化算法，主要用于最小化损失函数，通过逐步调整模型参数来提高预测性能。我们可以通过模拟梯度下降的过程来深入理解梯度下降。

1. **损失函数定义：**我们选择一个二次函数作为损失函数，其形式为：

该函数的最小值位于点 (2, 3)，这是我们希望通过梯度下降算法找到的目标。

def loss\_function(x, y):

return (x - 2)\*\*2 + (y - 3)\*\*2

1. **梯度计算：**损失函数的梯度可以通过对函数求偏导得到，用于指导梯度下降过程中的更新方向。

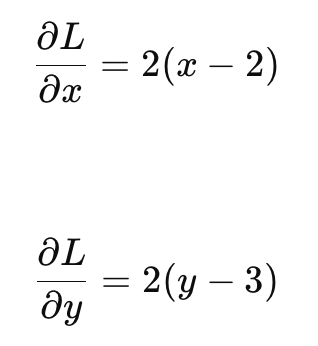
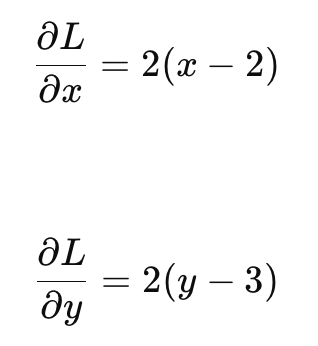
# 定义损失函数的梯度

def gradient(x, y):

grad\_x = 2 \* (x - 2)

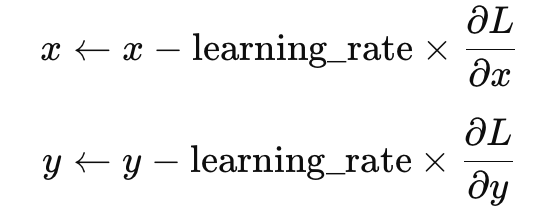
grad\_y = 2 \* (y - 3)

return grad\_x, grad\_y



1. **梯度下降算法定义**

我们实现了梯度下降算法，步骤如下：

1. 初始化起始点和学习率。
2. ****在每次迭代中，根据当前点的梯度更新坐标：

def gradient\_descent(initial\_point, learning\_rate, num\_iterations):

x, y = initial\_point

trajectory = [(x, y)] # 记录梯度下降过程中的点

for \_ in range(num\_iterations):

grad\_x, grad\_y = gradient(x, y)

x -= learning\_rate \* grad\_x

y -= learning\_rate \* grad\_y

trajectory.append((x, y))

return trajectory

1. 记录每一步的坐标，以便后续可视化。
2. **梯度下降算法实现**

initial\_point = (0, 0) # 起始点

learning\_rate = 0.1 # 学习率

num\_iterations = 50 # 迭代次数

# 执行梯度下降

trajectory = gradient\_descent(initial\_point, learning\_rate, num\_iterations)

# 转换为numpy数组以便绘图

import numpy as np

trajectory = np.array(trajectory)

# 创建网格以绘制损失函数的曲面

x\_vals = np.linspace(-1, 5, 400)

y\_vals = np.linspace(-1, 5, 400)

X, Y = np.meshgrid(x\_vals, y\_vals)

Z = loss\_function(X, Y)

# 绘制损失函数的曲面

fig = plt.figure(figsize=(12, 6))

ax1 = fig.add\_subplot(121, projection='3d')

ax1.plot\_surface(X, Y, Z, cmap='viridis')

ax1.set\_title('Loss Function Surface')

ax1.set\_xlabel('x')

ax1.set\_ylabel('y')

ax1.set\_zlabel('Loss')

# 绘制梯度下降的路径

ax2 = fig.add\_subplot(122)

ax2.contour(X, Y, Z, levels=20, cmap='viridis')

ax2.plot(trajectory[:, 0], trajectory[:, 1], 'r-', linewidth=2, label='Gradient Descent Path')

ax2.scatter(trajectory[0, 0], trajectory[0, 1], color='blue', label='Start Point')

ax2.scatter(trajectory[-1, 0], trajectory[-1, 1], color='green', label='End Point')

ax2.set\_title('Gradient Descent Path on Loss Function Contour')

ax2.set\_xlabel('x')

ax2.set\_ylabel('y')

ax2.legend()

# 显示图形

plt.tight\_layout()

plt.show()

1. **可视化结果**

# 绘制损失函数的曲面

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(12, 6))

ax1 = fig.add\_subplot(121, projection='3d')

ax1.plot\_surface(X, Y, Z, cmap='viridis')

ax1.set\_title('Loss Function Surface')

ax1.set\_xlabel('x')

ax1.set\_ylabel('y')

ax1.set\_zlabel('Loss')

# 绘制梯度下降的路径

ax2 = fig.add\_subplot(122)

ax2.contour(X, Y, Z, levels=20, cmap='viridis')

ax2.plot(trajectory[:, 0], trajectory[:, 1], 'r-', linewidth=2, label='Gradient Descent Path')

ax2.scatter(trajectory[0, 0], trajectory[0, 1], color='blue', label='Start Point')

ax2.scatter(trajectory[-1, 0], trajectory[-1, 1], color='green', label='End Point')

ax2.set\_title('Gradient Descent Path on Loss Function Contour')

ax2.set\_xlabel('x')

ax2.set\_ylabel('y')

ax2.legend()

# 显示图形

plt.tight\_layout()

plt.show()

### 聚类 K-means

**参考资料：**[https://scikit-learn.org/stable/modules/clustering.html#k-means](https://scikit-learn.org/stable/modules/clustering.html" \l "k-means)

在鸢尾花例子中，聚类就像是在没有任何先验知识的情况下，仅仅根据花的特征（花萼和花瓣的长度和宽度）来将它们分类。

1. **数据加载**：X包含特征数据（萼片和花瓣的长度和宽度），y包含真实的类别标签。

from sklearn import datasets

# 加载数据集

iris = datasets.load\_iris()

X = iris.data

y = iris.target

1. **数据标准化：**标准化的目的是将所有特征调整到相同的尺度，这对K-means算法很重要，因为它基于距离计算。

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

1. **模型创建与训练：**指定簇的数量为3（因为鸢尾花有3个品种）。random\_state=42确保结果可重复。

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=3, random\_state=42)

kmeans.fit(X\_scaled)

1. **模型预测：**predict方法将每个数据点分配到最近的簇。y\_kmeans包含每个数据点的簇标签。

y\_kmeans = kmeans.predict(X\_scaled)

y\_kmeans = kmeans.predict(X\_scaled)

1. **可视化**：创建两个子图：左图显示原始数据的真实类别，右图显示K-means聚类的结果。我们只使用前两个特征（萼片长度和宽度）来创建2D散点图。

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 5))

# 原始数据的真实类别

plt.subplot(121)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')

plt.title('True Labels')

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Sepal Width (cm)')

# K-means聚类的结果

plt.subplot(122)

plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, cmap='viridis')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1],

s=300, c='red', marker='\*', label='Centroids')

plt.title('K-means Clustering Results')

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Sepal Width (cm)')

plt.legend()

plt.show()

1. 模型评估

计算聚类的准确率。需要注意的是，K-means是一种无监督学习算法，通常不会用准确率来评估。但在这个例子中，由于我们知道真实的类别标签，所以可以用它来了解聚类效果。

from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(y, y\_kmeans)

print(f"Clustering Accuracy: {accuracy:.2f}")

1. 分析簇的组成

for i in range(3):

cluster = np.where(y\_kmeans == i)[0]

print(f"\nComposition of Cluster {i}:")

for iris\_type in range(3):

count = np.sum(y[cluster] == iris\_type)

print(f" {iris.target\_names[iris\_type]}: {count}")

### 降维

**参考资料：**[https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html" \l "sklearn.decomposition.PCA)

鸢尾花数据集有四个特征：花萼长度、花萼宽度、花瓣长度和花瓣宽度。我们可以想象三维空间，但四维空间超出了我们的直觉。主成分分析（Principal Component Analysis，PCA）要做的就是试图找到数据变化最大的方向（我们称之为主成分），然后用这些方向来表示数据。

1. **数据加载**

from sklearn import datasets

# 加载数据集

iris = datasets.load\_iris()

X = iris.data

y = iris.target

1. **数据标准化**

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

1. **应用PCA：**这里我们创建了一个PCA对象，并使用fit\_transform方法对标准化后的数据进行转换。X\_pca就是转换后的数据，其中每一列对应一个主成分。

from sklearn.decomposition import PCA

pca = PCA()

X\_pca = pca.fit\_transform(X\_scaled)

1. **分析主成分：**explained\_variance\_ratio\_告诉我们每个主成分解释了多少比例的方差。通过查看 explained\_variance\_ratio\_，可以决定选择多少个主成分以保留足够的方差（通常选择累积方差达到 90% 或 95%）。

explained\_variance\_ratio = pca.explained\_variance\_ratio\_

print(f"Explained Variance Ratio: {explained\_variance\_ratio}")

1. **分析主成分的组成：**打印出每个主成分的特征向量,表示每个原始特征对主成分的贡献。然后计算并打印原始特征与主成分的相关性。这告诉我们每个主成分与原始特征的关系强度和方向。

print("主成分的特征向量:")

for i, component in enumerate(pca.components\_):

print(f"PC{i+1}: {component}")

print("\n原始特征与主成分的相关性:")

for i, component in enumerate(pca.components\_):

correlations = component \* np.sqrt(pca.explained\_variance\_[i])

print(f"PC{i+1}:")

for j, corr in enumerate(correlations):

print(f" {iris.feature\_names[j]}: {corr:.3f}")

### 4.利用大模型生成代码

利用ChatGPT、DeepSeek等大模型生成下面几个问题的python求解代码：

1. 装箱子

2. 用动态规划实现斐波那契数列求解

3. 八数码

4. 用回溯法实现n皇后问题求解

生成的代码可能会出错，最终提交的代码要确保代码能够运行（如果只有一个函数或class，则可要求大模型生成样例）。

## 提交内容：

**仅提交ipynb文件，命名为lab4-姓名-学号.ipynb。**

**ipynb文件共8个cell：一个实现梯度下降、一个对鸢尾花数据进行聚类、一个实现鸢尾花数据的降维操作、四个用来填问题求解的代码以及最后一个描述心得与思路。要用注释注明每个Cell内的代码功能。所有代码都必须能够运行。**