《机器学习》编程作业

题目 1.1，2.1

一、题目理解

1.1 题目 1.1

[1] 在西瓜数据集 3.0a 上分别用线性核和高斯核训练一个 SVM。
理解：在西瓜数据集 3.0a 上分别使用线性核（linear 核）和高斯核（rbf 核）训练一个支持向量机（SVM）模型，利用 svm 数学原理，采用 math 库等手写实现一个版本，并绘制出决策边界图。同时，采用 sklearn 调包实现，可以设置不同的参数，例如惩罚系数 C、核函数系数 gamma，采用不同的参数搭配进行训练，绘制出决策边界和分类性能图。

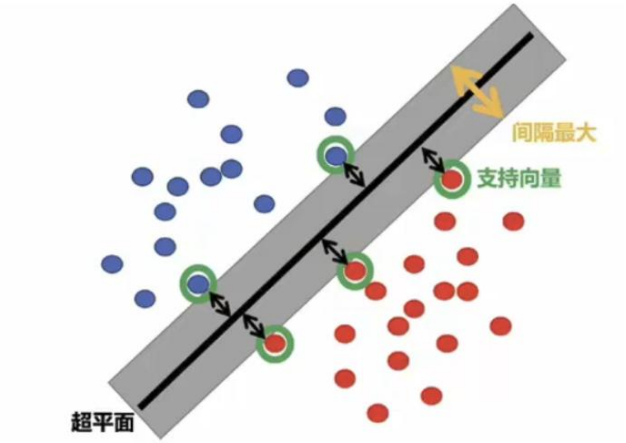
1.2 题目 2.1

[2]选择一个 UCI 数据集，分别用 linear, rbf 两个核函数训练 SVM【复用算法】。
理解：本实验中选取的 UCI 数据集仍为乳腺癌数据集，分别使用线性核（linear 核）和高斯核（rbf 核）训练一个支持向量机（SVM）模型，与题目 1.1 一致，实现手写版与 sklearn 两个版本。采用 sklearn 直接实现时，可以设置不同的参数，例如惩罚系数 C、核函数系数 gamma，采用不同的参数搭配进行训练，绘制出决策边界和分类性能图。在绘制决策边界时，由于数据集特征较多，故采用主成分分析法（PCA）进行降维，从而绘制出决策边界。

二、算法原理阐述

2.1 SVM 算法原理

支持向量机（SVM）是一种用于分类和回归的监督学习算法。SVM 的核心思想是找到一个最佳的超平面（或决策边界），将不同类别的数据点分开，并在该边界上最大化两类数据之间的间隔，以提高模型的泛化能力。



1. **基本思想：寻找最优超平面**

在 SVM 中，超平面是一个将数据点分割成不同类别的线性边界。在二维空间中，这个

边界是线，而在更高维空间中，边界是一个超平面。SVM 通过寻找一个最优超平面来将不同类别的样本分开，使得不同类别之间的“间隔”最大化。最大化间隔（margin）有助于提高分类的鲁棒性和泛化能力。

对于一个二分类问题，给定一组训练数据 (x1,y1),(x2,y2),…,(xn,yn)，其中 xi 是输入特征，yi∈{−1,1}是类别标签。SVM 的目标是找到一个超平面，满足：

$$w * x + b = 0$$

其中， w 是法向量，决定超平面的方向，b 是偏置项，决定超平面的位置。

2. 最大化间隔

在 SVM 中，间隔（margin）定义为离超平面最近的样本点到超平面的距离。SVM 通过最大化这个间隔来找到最佳的超平面。

如果我们规定所有支持向量（即最靠近超平面的样本点）满足以下条件：

$$y_i(w \cdot x_i + b) \geq 1$$

那么间隔的大小为$\frac{2}{\|w\|}$。因此，最大化间隔等价于最小化‖ w ‖，同时满足约束条件：

$$y_i(w \cdot x_i + b) \geq 1, \quad i = 1, 2, \ldots, n$$

这可以转化为一个凸优化问题：

$$\min \frac{1}{2}\|w\|^2$$

在约束条件$y_i(w \cdot x_i + b) \geq 1$ 下。

3. 软间隔与正则化参数 C

在实际应用中，数据往往是线性不可分的。为了解决这一问题，SVM 引入了软间隔（Soft Margin）概念，允许部分样本越过决策边界，从而找到一个更符合实际的数据分割方式。软间隔通过引入松弛变量ξ_i（表示样本允许误分类的程度）来实现，优化问题变为：

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

在约束条件$y_i(w \cdot x_i + b) \geq 1-\xi_i, \xi_i \geq 0$ 下。

其中，C 是正则化参数，控制误分类的惩罚力度。较大的 C 值会使模型更严格地拟合训练数据，但也可能导致过拟合；较小的 C 值则会使模型具有更好的泛化能力，但可能会放宽对训练数据的拟合要求。

4. 核函数与非线性分类

对于线性不可分的数据，SVM 使用核技巧（Kernel Trick）将数据映射到一个高维空间，使得在高维空间中数据可以线性可分。常见的核函数包括：

- 线性核：适用于数据线性可分的情况。
- 多项式核：适用于非线性关系较弱的数据。
- 高斯核（RBF 核）：适用于复杂的非线性关系，具有较强的分类能力。
- sigmoid 核：适用于神经网络相关的场景。

通过选择合适的核函数，SVM 可以在高维空间中找到一个超平面，将非线性可分的数据转化为线性可分。

5. 对偶问题与拉格朗日乘子法

SVM 的优化问题可以通过拉格朗日乘子法转化为对偶问题，以便使用核技巧。对偶问题形式如下：

$$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

在约束条件 $0 \leq a_i \leq C$ 和 $\sum_{i=1}^{n} a_i y_i = 0$ 下。

通过对偶问题，可以将输入数据映射到高维空间计算，而不必直接在高维空间中求解，避免了高维空间计算的复杂性。

6. 支持向量与决策边界

SVM 仅依赖于支持向量来确定分类边界。支持向量是指那些位于或接近决策边界的样本点。对于这些样本点，拉格朗日乘子α_i不为零，而对于非支持向量点，α_i等于 0。最终的决策函数为：

$$f(x) = \text{sign}\left(\sum_{i \in \text{SV}} \alpha_i y_i K(x_i, x) + b \right)$$

其中，SV 表示支持向量的集合。SVM 的分类决策由支持向量决定，因此称为“支持向量机”。

2.2 PCA 主成分分析

主成分分析（PCA）是一种常用的降维方法，旨在保留数据集中尽可能多的信息，同时减少特征的数量。PCA 的核心思想是通过寻找数据的“主成分”，即数据方差最大的方向，将数据从高维空间映射到低维空间。PCA 常用于数据可视化、特征提取和数据压缩。

PCA 的基本步骤:

（1）数据标准化

为了避免特征尺度不同带来的影响，首先对数据进行标准化，使得每个特征的均值为 0，方差为 1。标准化后的数据有利于计算协方差矩阵，确保 PCA 的效果不受原始特征数值范围的影响。

（2）计算协方差矩阵

在标准化数据的基础上，计算协方差矩阵。协方差矩阵表示每对特征之间的线性关系，矩阵中的元素反映了特征之间的相关性。协方差矩阵的大小为 d×d（其中 d 是原始数据的特征数），它描述了数据在各特征维度上的分布。

（3）特征值分解

对协方差矩阵进行特征值分解，得到特征值和特征向量。特征值表示数据沿着特征向量方向的方差大小。方差越大的方向说明数据在该方向的分布越广，因此保留更多的信息。特征向量则定义了新的坐标轴方向。

（4）选择主成分

按特征值从大到小排序，选择前 k 个特征向量作为主成分，以此形成新的 k 维空间（其中 k 通常远小于原始维度 d）。这些主成分构成新的坐标轴，每个主成分都代表数据方差最大的方向。

（5）数据投影

将原始数据投影到选定的主成分上，得到降维后的数据。在新的低维空间中，每个数据点由 k 个主成分表示。这些主成分保留了原始数据中最多的信息，同时大幅减少了数据的维度。

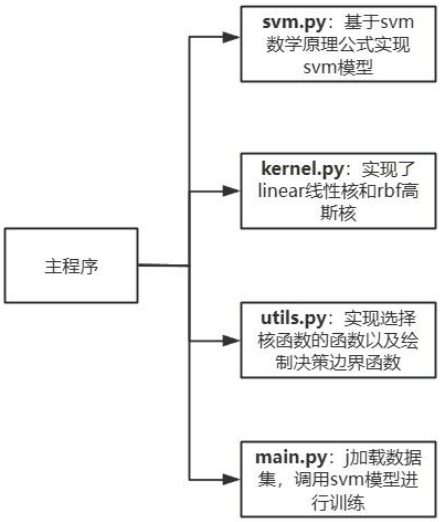
三、设计思路、代码结构及核心部分介绍

3.1 实验设计思路

本次实验实现了两个版本的 svm 模型，分别为调用 math 库等的手写实现版与采用 sklearn 中 SVM 模块的直接实现版。其中，sklearn 实现版设置一系列超参数（如 C 值、gamma 值、核函数等）来对 svm 模型进行调试。在西瓜数据集 3.0a 与乳腺癌数据集下，分别使用 linear 核函数与 rbf 核函数，同时设置不同的 C 值、gamma 值训练多次，分析比较不同参数训练下，模型得到的分类结果。

3.2 手写版代码结构

3.2.1 总体代码框架



3.2.2 具体实现代码结构

1. svm.py

所定义函数如下：

（1）__init__(self, epoch, c, epsilon, kernel)　//初始化模型的超参数:最大迭代次数,正则化参数，精度阈值,核函数类型

（2）fit(self, x, y)　//用于模型训练，使用简化 SMO 算法迭代优化 alpha 值以找到支持向量，最终确定超平面 w 和偏置 b。

（3）compute_w(self, alpha, x, y)　//根据最终的 alpha 值计算权重向量 w，用于定义决策超平面。

（4）compute_b(self, x, y)　//计算偏置 b

（5）compute_h(self, x)　//计算给定样本 x 的预测值

（6）compute_error(self, x, y)　//计算预测误差，即预测值与真实标签的差。

（7）compute_L_H(self, a_j_old, a_i_old, y_j, y_i)　//计算 alpha[j] 的取值范围 L 和 H，确保更新值满足 SVM 的约束条件

（8）predict(self, x)　//对新样本 x 进行预测

（9）accuracy(self, y, y_hat)　//计算预测的准确率，通过比较预测结果和实际标签 y，统

计正确分类的样本数量。

2. kernel.py

实现了线性核以及高斯核

```
import numpy as np
import numpy.linalg as la

class Kernel:
    @classmethod
    def linear(cls, x1, x2):
        return np.inner(x1, x2)

    @classmethod
    def gaussian(cls, sigma):
        return lambda x, y: \
            np.exp(-np.sqrt(la.norm(x-y) ** 2 / (2 * sigma ** 2)))
```

3. utils.py

有两个功能，一是实现选择核函数的函数，在 fit 函数中被调用。二是实现画图函数，画出决策边界图。

```
import numpy as np
import matplotlib.pyplot as plt
from svm_manual.kernel import Kernel



def choose_kernel(name):
    k = Kernel()
    if name == 'linear':
        return k.linear
    if name == 'gaussian':
        return k.gaussian(sigma=1.0)

def plot_hyperplane(clf, x, y, h=0.02):
    x_min, x_max = x[:, 0].min() - 1, x[:, 0].max() + 1
    y_min, y_max = x[:, 1].min() - 1, x[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])  # SVM 的分割超平面
    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap='hot', alpha=0.5)
    plt.scatter(x[:, 0], x[:, 1], c=y)
```

```
    plt.show()
```

4. main.py

```
import numpy as np
import pandas as pd
from 手动实现版.svm import SVM
from 手动实现版 import utils

def load_data(filename):
    data = pd.read_csv(filename, delimiter=' ')
    x = data[['密度', '含糖率']]
    y = data['好瓜']
    return np.array(x), np.array(y)

# 读取数据
x, y = load_data(r'E:\Python_file\MachineLearning4\3.0a.txt')
model = SVM(c=1.0, kernel='linear')
support_vectors, iterations = model.fit(x, y)
y_hat = model.predict(x)
score = model.accuracy(y, y_hat)
print(score)
utils.plot_hyperplane(model, x, y)

model1 = SVM(c=1.0, kernel='gaussian')
support_vectors, iterations = model1.fit(x, y)
y_hat = model1.predict(x)
score = model1.accuracy(y, y_hat)
print(score)
utils.plot_hyperplane(model1, x, y)
```

3.3 手写版核心部分介绍

主要介绍一下 svm.py 中 SVM 类的实现。

fit()函数是核心函数，其主要步骤如下：

（1）初始化：

- 初始化拉格朗日乘子α为 0。
- 设置最大迭代次数和容忍误差（epsilon）。
- 选择合适的核函数（线性核或高斯核）。

（2）主循环：

- 对每对数据点，随机选择两个不同的样本，计算其对应的拉格朗日乘子α。
- 计算更新后的拉格朗日乘子，并通过选择合适的范围(L, H)来限制它们。
- 更新分类器的权重（w）和偏置（b）。

（3）收敛判断：

• 如果所有拉格朗日乘子的变化小于设定的误差容忍度（epsilon），则停止训练。

• 支持向量的选择：

• 最终得到的支持向量是对应于拉格朗日乘子α大于 0 的样本。

具体介绍见下方详细注释代码：

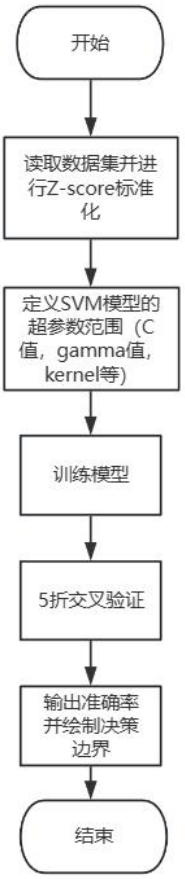
```
def fit(self, x, y):
    m = x.shape[0]  # 获取样本数量（行数）
    alpha = np.zeros(m)  # 初始化拉格朗日乘子α为 0，α是一个与样本数量相同的向量
    count = 0  # 初始化迭代次数为 0

    # 循环开始，最多迭代 epoch 次
    while count <= self.epoch:
        count += 1  # 每进行一次循环，计数器增加
        # 保存当前的α值，以便在迭代结束时检查是否收敛
        alpha_prev = np.copy(alpha)
        # 遍历每个样本点（内层循环）
        for j in range(0, m):
            # 随机选择第二个样本点
            i = random.randint(j, m-1)
            # 获取两个样本点的特征向量和标签
            x_i, x_j, y_i, y_j = x[i, :], x[j, :], y[i], y[j]
            # 计算η（eta），是计算拉格朗日乘子更新的关键值
            # η的计算公式：η = K(x_i, x_i) + K(x_j, x_j) - 2*K(x_i, x_j)
            # 这里的 K 表示核函数，通常是高斯核或线性核。
            eta = self.kernel(x_i, x_i) + self.kernel(x_j, x_j) - 2*self.kernel(x_i, x_j)
            # 如果 η 为 0，则跳过这次计算，避免除以 0
            if eta == 0:
                continue
            # 获取当前α值
            alpha_prime_j, alpha_prime_i = alpha[j], alpha[i]
            # 计算当前拉格朗日乘子的边界值 L 和 H
            # 依据α的更新规则，L 和 H 的计算方式不同，具体取决于 y_i 和 y_j 的值
            L, H = self.compute_L_H(alpha_prime_j, alpha_prime_i, y_j, y_i)
            # 计算权重 w，这里 w 是通过α的加权和得到的，属于 SVM 的优化目标之一
            self.w = self.compute_w(alpha, x, y)
            # 计算偏置 b，这里 b 是通过平均值的方式计算的，属于 SVM 的优化目标之
一
            self.b = self.compute_b(x, y)
            # 计算样本点 i 和样本点 j 的预测误差
            error_i = self.compute_error(x_i, y_i)
            error_j = self.compute_error(x_j, y_j)
            # 更新α[j]的值
            # α[j]的新值 = α[j] + y_j * (error_i - error_j) / η
            # 通过这一公式来调整α的值，使得 SVM 优化目标更好地逼近最优解
```

```
                alpha[j] = alpha_prime_j + y_j * (error_i - error_j) / eta
                # 将更新后的α值限制在 L 和 H 之间
                alpha[j] = max(alpha[j], L)
                alpha[j] = min(alpha[j], H)
                # 更新α[i]的值
                # α[i]的新值 = α[i] + y_i * y_j * (α[j] - α'[j])
                # 这一公式通过对α[i]的调整，保证α的更新能够满足 KKT 条件
                alpha[i] = alpha_prime_i + y_i * y_j * (alpha_prime_j - alpha[j])
            # 如果α的变化非常小，表明已经收敛，跳出循环
            # 通过比较当前α与上一次迭代的α差异，判断是否收敛
            if np.linalg.norm(alpha - alpha_prev) < self.epsilon:
                break
    # 迭代结束后，计算最终的权重 w 和偏置 b
    self.w = self.compute_w(alpha, x, y)
    self.b = self.compute_b(x, y)
    # 找到所有支持向量：即α > 0 的样本点
    alpha_idx = np.where(alpha > 0)[0]  # 获取所有支持向量的索引
    support_vectors = x[alpha_idx, :]  # 获取支持向量对应的样本数据
    # 返回支持向量和总迭代次数
    return support_vectors, count
```

3.4 西瓜数据集 sklearn 版整体代码结构

由于西瓜数据集 3.0a 的数据量较小，使用留出法划分训练集、测试集的话可能会导致模型过拟合，svm 模型训练出来的结果也具有偶然性。使用交叉验证可以更好地利用数据、更稳定地评估模型表现，并减少评估结果的波动性。因此，这里采用 5 折交叉验证法评估 svm 模型在西瓜数据集 3.0a 上的表现。

整体代码结构如下：



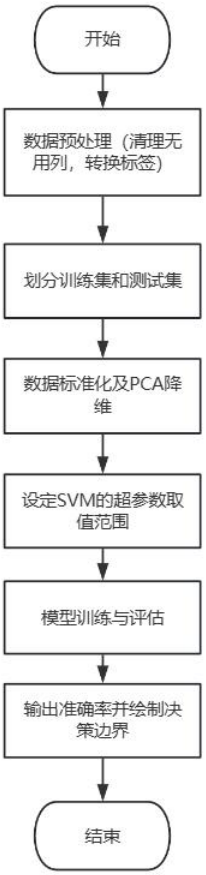
代码结构总结

- **导入库和设置字体**：导入所需的库，并设置字体显示。
- **读取并预处理数据**：读取数据集并进行 Z-score 标准化。
- **定义参数网格**：设定 SVM 模型的超参数范围。
- **遍历参数组合并训练模型**：使用参数网格中的每组参数训练模型。
- **计算准确率并输出**：对当前参数组合下的模型进行评估，打印准确率。
- **生成网格并预测**：生成用于绘制决策边界的网格点，并进行预测。
- **绘制决策边界和数据分布**：绘制每种参数组合下的决策边界和数据分布。

3.5 乳腺癌数据集 sklearn 版整体代码结构

乳腺癌数据集相较于西瓜数据集 3.0a 具有更多的数据，采用交叉验证会消耗较大的计算资源，故这里仍然采用留出法进行评估。

整体代码结构如下：



代码结构总结

• 导入库和设置字体：导入所需的库，并设置中文显示。
• 加载和清洗数据：加载数据集，清理无用列，转换标签。
• 划分训练集和测试集：将数据分为训练集和测试集。
• 标准化和降维：使用 Z-score 标准化特征数据，并用 PCA 将数据降至二维，便于可视化。
• 定义参数网格：设定 SVM 的超参数取值范围。
• 模型训练和评估：遍历每种参数组合，训练模型并在测试集上计算准确率。
• 绘制决策边界和数据点分布：生成网格用于绘制决策边界区域，同时绘制数据点分布，显示各类样本和分类边界。

3.6 sklearn 版核心部分介绍

（1）西瓜数据集 3.0a

标准化数据：

使用 `StandardScaler` 对特征数据进行 Z-score 标准化，使得数据的均值为 0、方差为 1。标准化后，模型训练的收敛速度更快，且模型性能更稳定。

```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

定义三个超参数的取值范围：

C：SVM 的正则化参数，控制对误分类样本的惩罚力度。较大的 C 值倾向于更严格地拟合训练数据，但可能导致过拟合。
gamma：RBF 核的参数，决定高斯分布的宽度（影响模型的非线性特征）。较大的 gamma 值让模型更加关注于局部特征，适合处理复杂分布。
kernel：核函数类型，选择线性核（linear）和 RBF 核（rbf）来适应不同的数据分布。

```
param_grid = {
    'C': [1, 10, 100],  # 正则化参数的取值范围
    'gamma': [0.1, 1],  # RBF 核的 gamma 参数取值范围
    'kernel': ['linear', 'rbf']  # 核函数类型：线性核和 RBF 核
}
```

评估模型：

计算准确率：使用训练好的模型在整个数据集上进行预测，并计算模型的准确率。
输出结果：打印出每种参数组合对应的准确率。通过观察这些结果，可以分析不同参数对模型分类效果的影响。

```
accuracy = accuracy_score(y, model.predict(X_scaled))
print(f"当前参数组合: C={params['C']}, gamma={params['gamma']}, kernel={params['kernel']}, 准确率={accuracy:.2f}")
```

决策边界的绘制：

• 网格生成：在二维平面上生成网格点，覆盖标准化后的特征空间。
• 网格范围设置：将 X_scaled 的最小值和最大值扩大 1 个单位，使得图像边缘不会截断数据。

```
x_min, x_max = X_scaled[:, 0].min() - 1, X_scaled[:, 0].max() + 1
y_min, y_max = X_scaled[:, 1].min() - 1, X_scaled[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))
```

决策边界和数据分布的可视化：

• 绘制决策边界：使用 contourf 将网格点预测结果填充成区域，不同的颜色表示不同的分类区域。
•绘制数据点：使用散点图绘制数据点，c=y 表示使用 好瓜 列的值作为颜色标签，区分“好瓜”和“坏瓜”。
• 设置标签和标题：设置 x 轴和 y 轴标签为“密度”和“含糖率”，标题包含当前参数组合和模型的准确率，以便更直观地观察参数对模型的影响。

```
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.3)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=y, cmap=plt.cm.coolwarm, edgecolors='k', s=50)
plt.xlabel('密度 (标准化)')
plt.ylabel('含糖率 (标准化)')
plt.title(f'SVM 决策边界 (C={params["C"]}, gamma={params["gamma"]}, kernel={params["kernel"]})\n 准确率: {accuracy:.2f}')
```

```
plt.show()
```

（2）**乳腺癌数据集**

PCA 降维：

使用主成分分析 (PCA) 将数据降维至二维。通过降维，可以在二维平面上可视化数据，便于后续的决策边界绘制。这样可以简化数据结构，同时保留尽可能多的信息（即数据方差）。

```
pca = PCA(n_components=2)
train_X_2d = pca.fit_transform(train_X)
test_X_2d = pca.transform(test_X)
```

训练模型并预测：

首先，使用三重循环遍历所有参数组合。对于每个参数组合，创建对应的 SVM 模型。
线性核不需要设置 gamma 参数，因为线性核不涉及非线性映射，而 RBF 核需要设置 gamma 参数，用于控制高斯核的影响范围。

```
for C in param_grid['C']:
    for gamma in param_grid['gamma']:
        for kernel in param_grid['kernel']:
            if kernel == 'linear':
                model = svm.SVC(C=C, kernel=kernel)
            else:
                model = svm.SVC(C=C, gamma=gamma, kernel=kernel)
            model.fit(train_X_2d, train_y)
            test_pred = model.predict(test_X_2d)
            accuracy = metrics.accuracy_score(test_y, test_pred)
            print(f"参数组合：C={C}, gamma={gamma}, kernel={kernel}, 测试集准确率={accuracy:.4f}")
```

可视化训练数据分布：

用不同颜色区分类别（良性和恶性），绘制出训练集数据点。

```
sns.scatterplot(x=train_X_2d[:, 0], y=train_X_2d[:, 1], hue=train_y,
                palette=["blue", "red"], edgecolor='k')
plt.title(f"SVM 决策边界 (C={C}, gamma={gamma}, kernel={kernel})\n 测试集准确率：{accuracy:.4f}")
plt.xlabel('主成分 1')
plt.ylabel('主成分 2')
plt.legend(['Benign', 'Malignant'])
plt.show()
```

四、题目 1.1 实验流程、测试结果及分析

4.1 流程与测试结果

4.1.1 手动实现版实验流程及实验结果

设定 C=1，首先选择线性核进行训练，然后选用高斯核进行训练，得出两种核函数的训练结果。

得到两种线性核的正确率与决策边界图分别如下：

核函数	正确率
linear	0.4117647058823529
rbf	0.47058823529411764

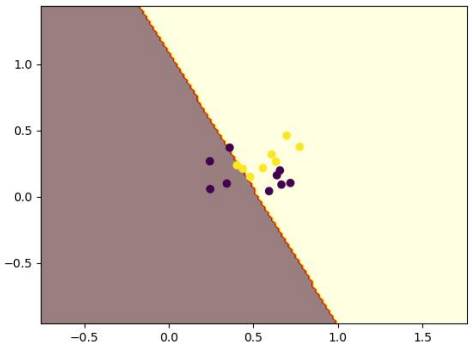


图 1 手写版 linear 核决策边界

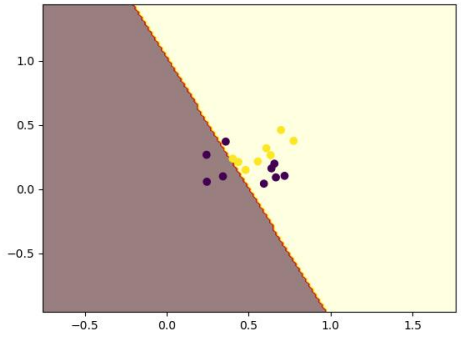


图 2 手写版 rbf 核决策边界

由于手写实现较简易，所以结果相对于 sklearn 版来说并不是特别好。

4.1.3 sklearn 版实验流程

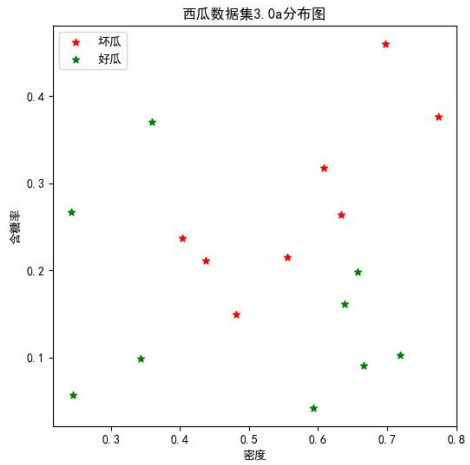
（1）首先进行数据可视化与数据预处理，将数据划分为 X 与 Y。由于西瓜数据集 3.0a 较小，故采用交叉验证法进行评估，不用划分训练集与测试集。

（2）设置不同的参数，惩罚系数 C，核函数系数 gamma,并选择不同的核函数，线性核函数‘linear’与高斯核函数‘rbf’。

（3）依次取不同值的参数，组成不同的参数搭配，在每一种参数搭配的条件下训练数据，通过五折交叉验证法评估训练结果，并画出该参数搭配下，svm 训练得到的边界。

4.1.4 sklearn 版测试结果

首先给出西瓜数据集 3.0a 的可视化图像如下：



取不同的参数进行训练，得到交叉验证法准确率见下表：

表 1 西瓜数据集不同参数准确率

C	gamma	kernel	准确率
1	0.1	linear	0.82

C	gamma	kernel	准确率
1	0.1	rbf	0.76
1	1	linear	0.82
1	1	rbf	1.00
10	0.1	linear	0.82
10	0.1	rbf	0.88
10	1	linear	0.82
10	1	rbf	1.00
100	0.1	linear	0.82
100	0.1	rbf	1.00
100	1	linear	0.82
100	1	rbf	1.00

绘制折线图如下：

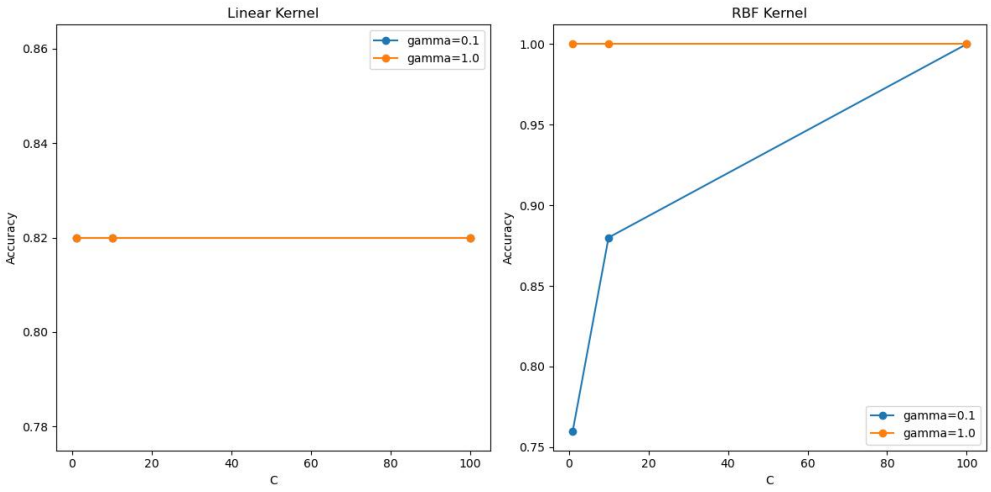
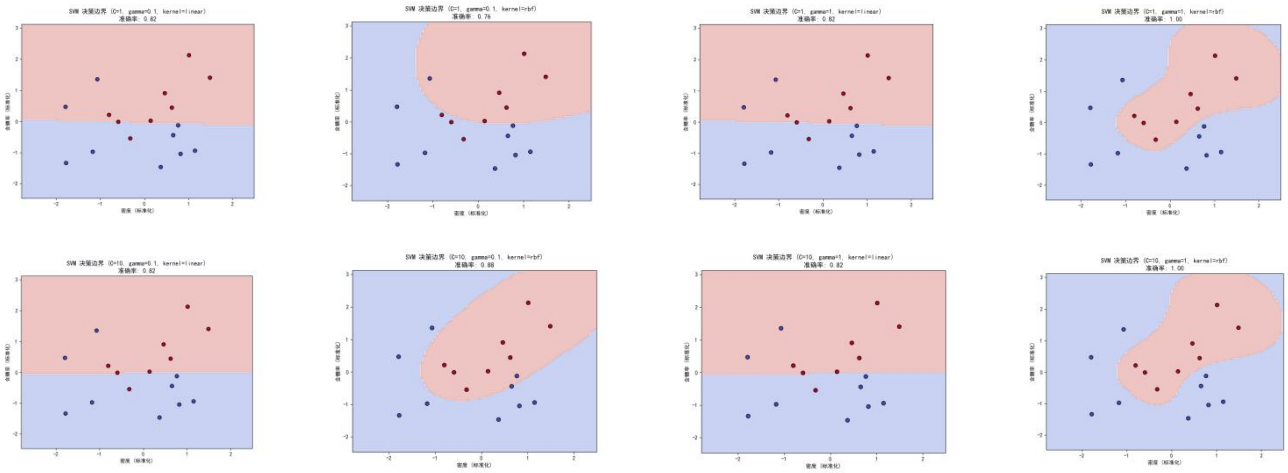


图 3 西瓜数据集参数评估图

同时，给出不同参数下 svm 的决策边界图：



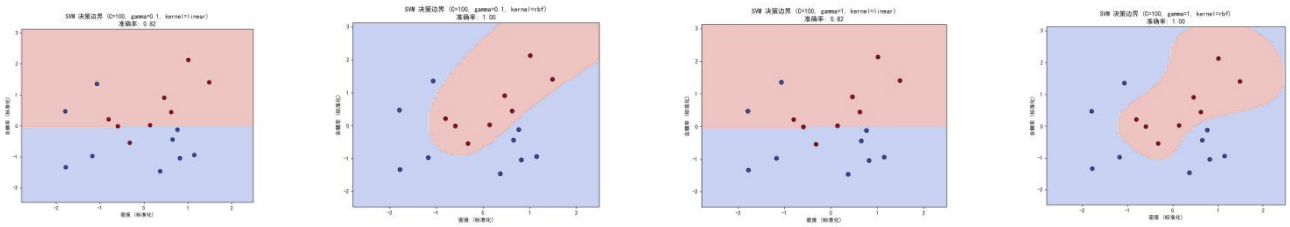


图 4 西瓜数据集-svm 边界图

4.2 sklearn 版结果分析

1. 线性核的表现

对于所有 C 值（1、10、100），线性核的准确率保持一致，均为 0.82。这说明在西瓜数据集 3.0a 上，线性核的分类效果不受 C 值的变化影响。这可能是因为西瓜数据集中的类别分布没有很强的线性分隔特性，导致线性核无法很好地适应数据的分布，准确率较低。线性核的决策边界是一条简单的直线，表现出较低的灵活性，无法很好地捕捉数据的复杂分布特性。

2. RBF 核的表现

在 RBF 核的情况下，不同的 C 和 gamma 参数组合对分类结果有显著影响。

当 C=1 且 gamma=0.1 时，准确率为 0.76，表现较差。这是因为小的 gamma 值会导致模型的决策边界比较平滑，无法适应数据的复杂分布。

当 gamma=1 时，模型的准确率提升明显，在 C=1,10,100 时均达到 1.00。这说明高 gamma 值使得模型的决策边界更加灵活，可以更好地适应数据的非线性分布特性。

决策边界图显示，当 gamma 较大时，RBF 核的决策边界能够弯曲，更好地包围数据点，从而实现更高的分类准确率。

结论

西瓜数据集 3.0a 在特征空间中存在较明显的非线性分布，因此 RBF 核表现优于线性核。使用高 gamma 和高 C 值的 RBF 核可以生成更加灵活的决策边界，更好地拟合数据分布。

五、题目 2.1 实验流程、测试结果及分析

5.1 流程与测试结果

5.1.1 实验流程

（1）首先进行数据预处理，删除 id 列，这一列对分类无贡献，并处理标签列，将 diagnosis 列的文本值映射为数值（恶性 "M" 设为 1，良性 "B" 设为 0），方便模型处理。

（2）继续对数据进行处理，首先利用 Z-Score 标准化方法对数据进行标准化，并且能够提升模型的稳定性以及准确性。然后利用主成分分析法将数据的多个特征降为二维，最后将数据划分为训练集与测试集，用于模型评估。

（2）设置不同的参数，惩罚系数 C，核函数系数 gamma,并选择不同的核函数，线性核函数‘linear’与高斯核函数‘rbf’。

（3）依次取不同值的参数，组成不同的参数搭配，在每一种参数搭配的条件下训练数据，通过留一法评估训练结果，并画出该参数搭配下，svm 训练得到的边界。

5.1.2 测试结果

取不同的参数进行训练，得到交叉验证法准确率见下表：

表 2 乳腺癌数据集不同参数准确率

C	gamma	kernel	测试集准确率
1	0.1	linear	0.9708
1	0.1	rbf	0.9474
1	1	linear	0.9708
1	1	rbf	0.9532
10	0.1	linear	0.9708
10	0.1	rbf	0.9474
10	1	linear	0.9708
10	1	rbf	0.9532
100	0.1	linear	0.9708
100	0.1	rbf	0.9532
100	1	linear	0.9708
100	1	rbf	0.9591

绘制折线图如下：

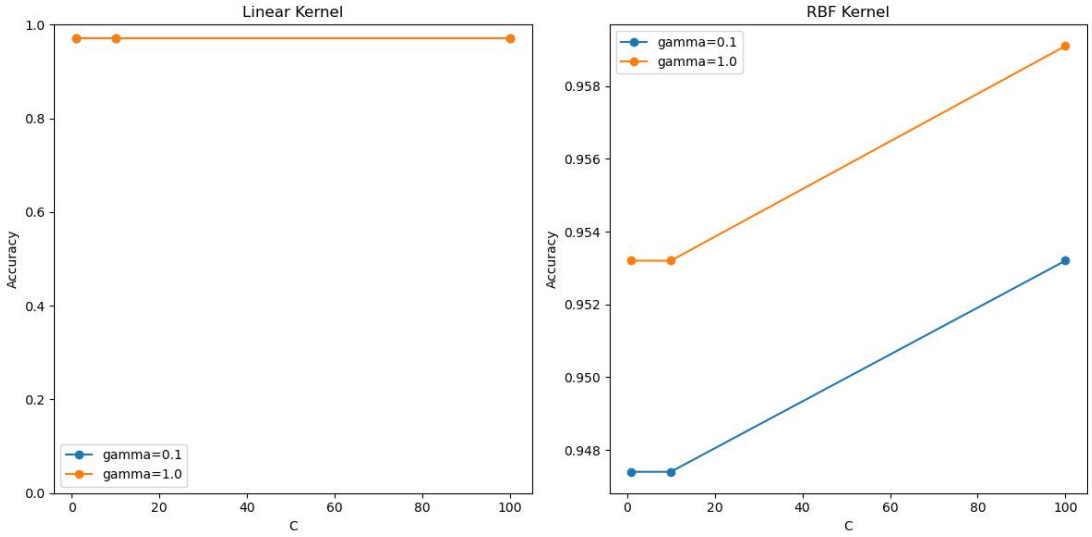
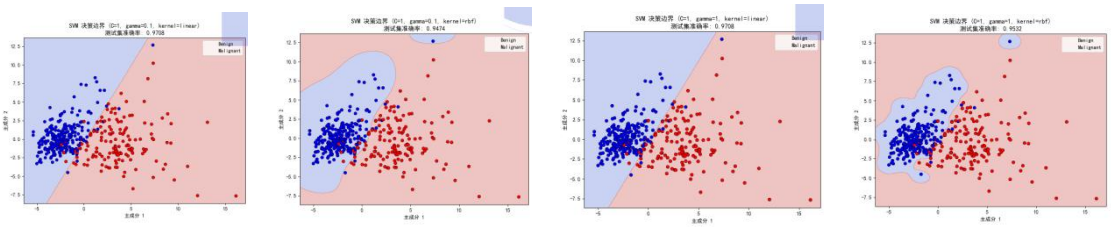
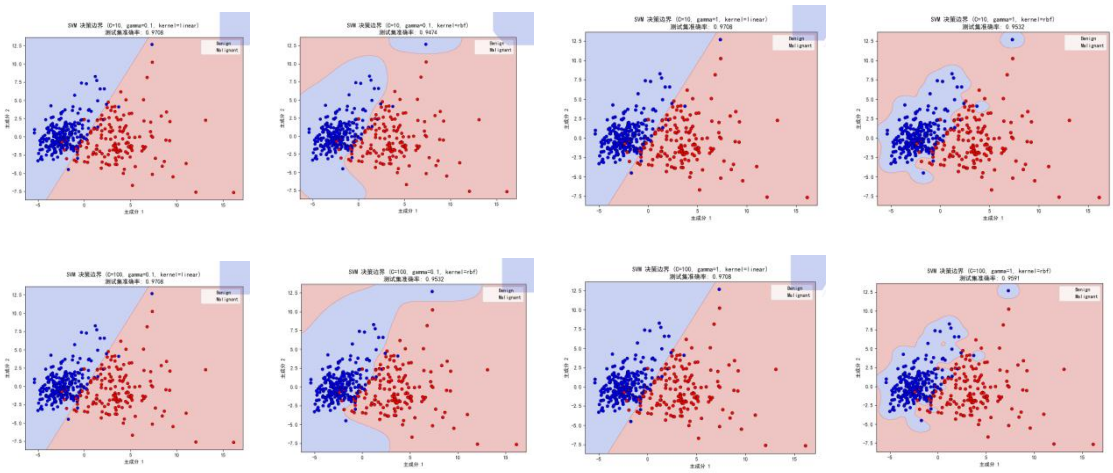


图 5 乳腺癌数据集参数评估图

对原始数据的所有特征进行 PCA 主成分分析降维后，给出不同参数下 svm 的决策边界图：





5.2 结果分析

1、线性核的表现：

在不同的 C 值下（1、10、100），线性核的准确率保持一致，均为 0.9708。这表明，在线性核的情况下，惩罚系数 C 对模型的分类性能影响较小，可能是因为乳腺癌数据集在 PCA 降维后的数据结构已经接近线性可分，因此线性核可以较好地分离数据。决策边界图显示，线性核在二维平面上生成了一条清晰的直线边界，适合于此类接近线性可分的数据。

2、RBF 核的表现：

在 RBF 核的情况下，随着 C 值的增加，模型的准确率也有所提高。例如，当 gamma 且 C 从 1 增加到 100 时，测试集准确率从 0.9532 提高到了 0.9591。这表明较高的 C 值能够增强模型对复杂数据的拟合能力，使决策边界更加灵活。

同时，较大的 gamma 值（如 1）相比较小的 gamma 值（如 0.1），在相同 C 值下准确率更高。这是因为较大的 gamma 值可以使模型在局部区域产生更复杂的边界，更好地捕捉到数据的非线性特征。

从决策边界图可以看出，RBF 核的边界相较于线性核更加弯曲，能够适应数据的复杂分布。

结论

从实验结果来看，对于乳腺癌数据集，线性核已经能够很好地分类，而 RBF 核适用于对非线性边界的进一步优化。对于乳腺癌数据集，线性核表现稳定且具有良好的泛化性，是一个更简单有效的选择。而 RBF 核在调参后可以获得稍高的准确率，但需要更高的模型复杂度和较高的 C、gamma 值。

六、问题与收获

6.1 遇到的问题

一开始，我选用的西瓜数据集训练验证评估方法为留一法，但训练准确度一直不够高。后续查阅资料得知，小数据集不适合采用留一法，故将评估方法换为 5 折交叉验证方法，训练准确率确实得到了一定的提升。

在处理乳腺癌数据集时，一开始我觉得数据特征太多，便采用相关性分析提取出几个特征，仅仅使用几个特征进行训练，但结果并不好。后面利用所有特征进行训练，效果更差了，于是我切换成了使用 PCA 主成分分析，将数据降到二维，然后进行训练，训练准确率确实得到了提升，并且利用 PCA 可以实现 svm 可视化边界。

6.2 收获

通过本次实验，我深刻认识到了数据处理对训练结果的影响，例如，仅仅对乳腺癌数据集进行数据标准化及主成分分析 PCA，便使得准确率大幅提升。

通过这次实验，我对 SVM 的线性核和 RBF 核有了更直观的理解。在西瓜数据集 3.0a 上，通过不同参数的组合，观察到线性核适合数据呈现线性分布的情况，而 RBF 核在处理非线性数据时更有效，尤其是在分布复杂的情况下。针对乳腺癌数据集，由于特征较多，采用 PCA 降维方法，将数据映射到二维平面，有助于清晰地展示不同核函数下的决策边界。此过程中，线性核的决策边界较简单，而 RBF 核可以产生更灵活的边界，对非线性模式的适应能力更强。

实验进一步加深了我对 SVM 核函数选择的理解，也让我更熟悉调参的流程和技巧。通过复用 SVM 算法在不同数据集上的表现，我理解了在实际应用中如何根据数据分布和特征来灵活选择合适的核函数，并结合降维手段提升可视化效果。这次实验对我在分类模型的选择和优化上起到了很大帮助，使我对 SVM 在不同数据场景下的应用更有信心。

七、参考资料

[1]python 机器学习 | SVM 算法介绍及实现_svm 算法实现-CSDN 博客
[2]机器学习 SVM——CSDN 博客
[3]《西瓜书》
[4]SVM 支持向量机算法-原理篇 - 码农充电站 - 博客园