**一、简述支持向量机**的思路、关键操作（不超过500字，逻辑要通，**要有小标题**，要排版清晰，请不要网络大幅摘抄）

支持向量机（SVM）是一种强大的监督学习算法，用于分类和回归分析。其思路主要基于找到一个最佳的超平面，将不同类别的数据点有效地分开，并以最大化边界距离来提高分类的鲁棒性。

**1.思路：**

SVM的关键思路在于通过寻找能够有效划分不同类别数据的超平面来实现分类。它不仅仅寻找一个能区分不同类别的超平面，而且还希望使得这个超平面与最近的数据点之间的距离最大化，从而增加分类的稳定性。

**2.关键操作：**

2.1核函数选择：SVM可以使用不同的核函数，如线性核函数、多项式核函数和高斯核函数，来处理线性不可分和非线性问题，根据具体情况选择合适的核函数。

2.2超平面优化：通过对数据点进行分类，确定最佳的超平面，使得不同类别的数据点到超平面的距离最大化。

2.3正则化参数选择：正则化参数C的选择对于控制模型的复杂度和容忍错误的程度非常重要，需要根据具体问题进行合理设定。

2.4边界距离最大化：SVM的目标是找到一个最大化边界距离的超平面，以保证分类的鲁棒性和泛化能力。

通过以上关键操作，SVM能够有效地处理线性和非线性分类问题，并且在特征空间维度较高时依然表现出色，成为了机器学习中重要的算法之一。

**二、利用SVM解决问题（回归或分类）。**

附：Sklearn常用数据集

|  |  |  |
| --- | --- | --- |
| 序号 | 数据集 | 学号尾号 |
| 1 | 鸢尾花数据集：load\_iris（） | 0 1 |
| 2 | 手写数字数据集：load\_digits（） | 2 3 |
| 3 | 乳腺癌数据集：load-breast-cancer（） | 4 5 8 |
| 4 | 糖尿病数据集：load-diabetes（） | 6 7 9 |

利用SVM对以上数据集（学号尾号不同采用不同数据集）进行分析，找出最佳参数下的SVM更适合本数据集（准确率更高，不同回归、不同参数下的不同准确率要列出）。

要求：

1）代码要列出，重点代码加注释说明，特别是自己调试过程中的自我理解；

2）运行结果要截图，结果要文字解释；

3）注意排版

from sklearn import datasets  
from sklearn.model\_selection import train\_test\_split  
from sklearn.svm import SVC  
from sklearn.metrics import accuracy\_score  
from sklearn.model\_selection import GridSearchCV  
iris = datasets.load\_iris()  
X = iris.data  
y = iris.target  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
# 定义参数网格  
param\_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kernel': ['rbf', 'linear', 'poly', 'sigmoid']}  
# 创建SVM模型  
svm = SVC()  
# 使用GridSearchCV进行交叉验证  
grid\_search = GridSearchCV(svm, param\_grid, cv=5)  
grid\_search.fit(X\_train, y\_train)  
# 输出最佳参数  
best\_params = grid\_search.best\_params\_  
print("最佳参数：", best\_params)  
# 使用最佳参数重新训练模型  
best\_svm = SVC(\*\*best\_params)  
best\_svm.fit(X\_train, y\_train)  
# 在测试集上评估模型  
y\_pred = best\_svm.predict(X\_test)  
accuracy = accuracy\_score(y\_test, y\_pred)  
print("最佳参数下的准确率：", accuracy)  
# 输出不同参数设置下的准确率  
print("不同参数设置下的准确率：")  
means = grid\_search.cv\_results\_['mean\_test\_score']  
params = grid\_search.cv\_results\_['params']  
for mean, param in zip(means, params):  
 print(param, mean)

import numpy as np  
import matplotlib.pyplot as plt  
  
# 选取 sepal length 和 petal length 作为特征  
X = iris.data[:, :2]  
y = iris.target  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
# 定义网格范围  
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, 0.01),  
 np.arange(y\_min, y\_max, 0.01))  
# 训练最佳参数下的 SVM 模型  
best\_svm.fit(X\_train, y\_train)  
Z = best\_svm.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)   
plt.figure()  
plt.contourf(xx, yy, Z, alpha=0.4, cmap=plt.cm.coolwarm)  
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm, s=20, edgecolors='k')  
plt.xlabel('Sepal length')  
plt.ylabel('Petal length')  
plt.show()

**结果：**

文本

描述已自动生成

文本

描述已自动生成

图表, 散点图

描述已自动生成

**三、[附加题].**利用支持向量机解决**自己遇到的问题**。

[解题要求同题2]

**SVM分析红酒数据集**

from sklearn import datasets  
from sklearn.model\_selection import train\_test\_split  
from sklearn.svm import SVC  
from sklearn.metrics import accuracy\_score  
from sklearn.model\_selection import GridSearchCV  
import numpy as np  
import matplotlib.pyplot as plt  
wine = datasets.load\_wine()  
X = wine.data[:, :2] # 只选取两个特征  
y = wine.target  
# 划分训练集和测试集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
# 定义参数网格  
param\_grid = {'C': [0.1, 1, 10], 'kernel': ['rbf', 'linear']}  
# 创建SVM模型  
svm = SVC()  
# 使用GridSearchCV进行交叉验证  
grid\_search = GridSearchCV(svm, param\_grid, cv=5)  
grid\_search.fit(X\_train, y\_train)  
# 输出最佳参数  
best\_params = grid\_search.best\_params\_  
print("最佳参数：", best\_params)  
# 使用最佳参数重新训练模型  
best\_svm = SVC(\*\*best\_params)  
best\_svm.fit(X\_train, y\_train)  
# 在测试集上评估模型  
y\_pred = best\_svm.predict(X\_test)  
accuracy = accuracy\_score(y\_test, y\_pred)  
print("最佳参数下的准确率：", accuracy)  
# 输出不同参数设置下的准确率  
print("不同参数设置下的准确率：")  
means = grid\_search.cv\_results\_['mean\_test\_score']  
params = grid\_search.cv\_results\_['params']  
for mean, param in zip(means, params):  
 print(param, mean)  
# 绘制决策边界  
h = .02 # 步长  
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))  
Z = best\_svm.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)  
  
plt.figure()  
plt.contourf(xx, yy, Z, alpha=0.4, cmap=plt.cm.coolwarm)  
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm, s=20, edgecolors='k')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.show()

**结果：**

文本

描述已自动生成

图表, 散点图

描述已自动生成