西北大学

智能信息系统综合实践
实验报告

题　　目：　SVM

年　　级：　2020

专　　业：　软件工程

姓　　名：　朱泽昊

一、题目（原题目）

- 1. 利用Iris Dataset（选前两类），交叉验证，测试SVM分类器性能，改变SVM超参数，对比不同超参数对结果的影响。
- 2. 对比SVM和第一章中的逻辑回归分类器，自定衡量指标。

二、解题步骤（思路+代码）

问题一：利用 Iris 数据集的前两类，使用交叉验证方法，测试 SVM 分类器性能，并且改变超参数，对比不同超参数对结果的影响。

解题思路：

（1）将包导入，先将数据准备好，所以先定义 X，把 iris 数据集的前两类数据传给 X，再定义 y，将 iris 数据集的前两类标签传给 y

（2）将拿出的数据划分为训练集和测试集，尽管是使用交叉验证的方法，但是还有第二题要与逻辑回归比较，所以还是要将数据集给划分，方便计算第二题的指标。【交叉验证里面划分为训练集和验证集】

（3）定义 svm 分类器，题目要求改变超参数，所以我使用网格搜索的思想，用一个列表将超参数给全部定义（因为有些超参数不兼容，所以就使用几个常见的变量进行组合），将数据和变量列表全部送到 SVM 分类器之中，之后会在训练之后会有一个变量存储最佳的参数。

（4）Prarm_grid 是前面定义的超参数列表，之后使用

```
GridSearchCV(svm, param_grid, cv=5)
```

算法将为每个超参数组合构建模型，然后使用交叉验证对每个模型进行评分以确定最佳超参数组合，并且返回一个带有最佳超参数组合的训练好的模型，以及与最佳超参数组合相应的模型评分。

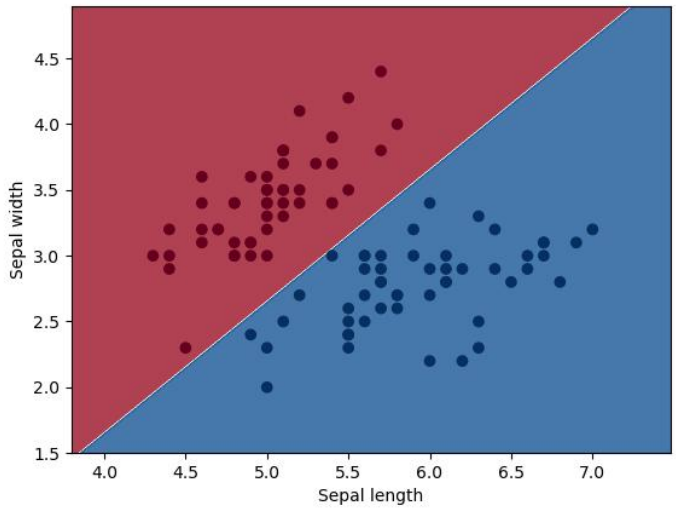
（5）之后使用该模型的 fit 方法进行训练，之后可以画出最佳的超参数组合的图像。因为这只有一种超参数训练出来的图像，之后会有使用别的超参数生成的图像与之对比。

代码如下：

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score, GridSearchCV, train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
import numpy as np
import matplotlib.pyplot as plt

# 导入数据集
iris = load_iris()

# 提取前两类，即前100个样本
X = iris.data[:100, :2]
y = iris.target[:100]
# 将数据集分成训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 定义SVM分类器
svm = SVC()

# 定义超参数网格
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf', 'poly'], 'gamma': ['scale', 'auto']}

# 使用网格搜索和交叉验证进行模型选择和调参
svm_cv = GridSearchCV(svm, param_grid, cv=5)
svm_cv.fit(X_train, y_train)
# 在测试集上进行预测
y_pred = svm_cv.predict(X_test)
```

```
# 绘制决策边界和样本点的散点图
x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))
Z = svm_cv.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.RdBu, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdBu)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```

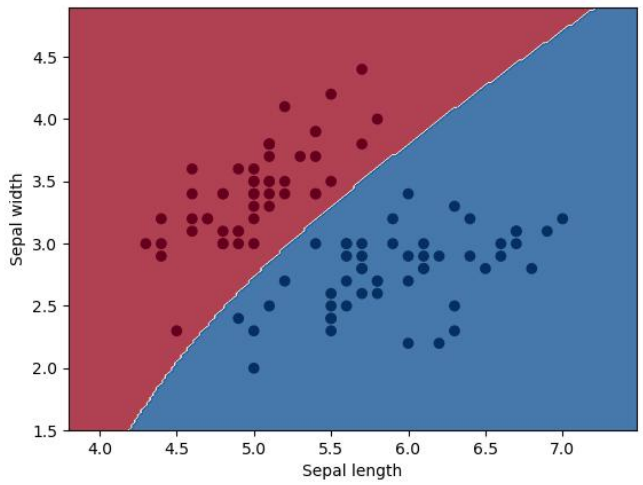
结果：



接下来使用另外一组超参数与之对比：将超参数列表限定到

```
param_grid = {'C': [0.1, 1, 10], 'kernel': ['poly'], 'gamma': ['scale']}
```

结果：



下面是对上述一些概念和超参数的解释：

【网格搜索：网格搜索是一种穷举搜索的方法，它通过将超参数的所有可能组合形成一个网格，然后在网格中进行搜索。这种方法虽然计算量较大，但可以找到最优的超参数组合。

惩罚参数 C（C parameter）：C 是一个正则化参数，控制分类器在训练过程中对误分类样本的惩罚。较小的 C 值允许更多的误分类样本，从而使决策边界更具弹性，但可能会导致过拟合。较大的 C 值会强制分类器对误分类样本进行更强的惩罚，从而使决策边界更严格，但也可能会导致欠拟合。

核函数（Kernel）：SVM 分类器使用核函数将数据映射到高维空间，以便找到非线性决策边界。常见的核函数包括线性核、多项式核和径向基函数（RBF）核。不同的核函数适用于不同类型的数据集，因此需要选择合适的核函数以获得最佳性能。

核函数的超参数（Kernel coefficient gamma）：当使用 RBF 核时，需要选择 gamma 值。gamma 值定义了每个样本对决策边界的影响程度，即 gamma 值越大，每个样本的影响程度越小，决策边界越平滑。但是，如果 gamma 值过大，可能会导致过拟合。

多项式核函数的次数（Degree）：当使用多项式核函数时，需要选择多项式的次数。多项式的次数越高，决策边界越复杂，但也可能会导致过拟合。】

问题一：对比 svm 和逻辑回归分类器，自定义指标

解题思路：

（1）加载 iris 数据集，并选择前两类的数据。
（2）将数据集分为训练集和测试集。
（3）创建逻辑回归分类器和 SVM 分类器，并使用训练集训练模型。
（4）之后定义一个 def plot_boundary(clf, X, y):函数将两个图像都画出来
（5）对测试集进行预测，并计算准确率、精准率、召回率和 F1 分数。
代码如下：

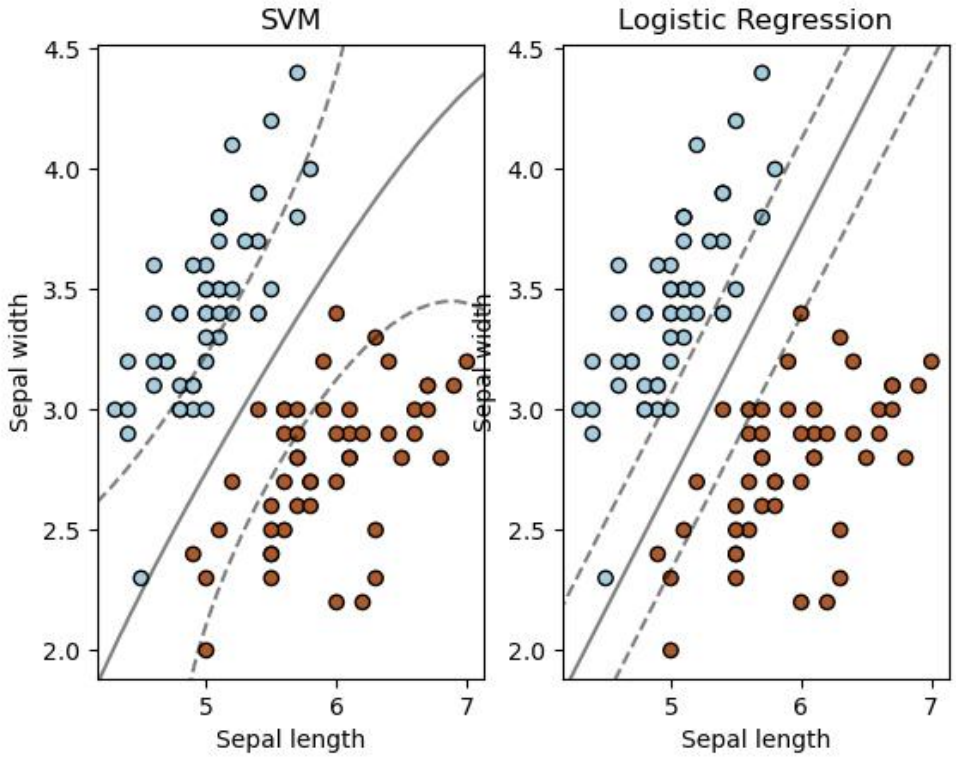
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# 加载iris数据集
iris = load_iris()
X = iris.data[0:100, 0:2]  # 选择前两类的数据
y = iris.target[0:100]

# 将数据集分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 创建逻辑回归分类器和SVM分类器
lr = LogisticRegression(random_state=42)
svc = SVC(random_state=42)

# 训练模型
lr.fit(X_train, y_train)
svc.fit(X_train, y_train)

# 对测试集进行预测
lr_pred = lr.predict(X_test)
svc_pred = svc.predict(X_test)
```

```python
def plot_boundary(clf, X, y):
    # 绘制数据点
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired, edgecolors='k')
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')

    # 绘制分类边界
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
    xx = np.linspace(xlim[0], xlim[1], 30)
    yy = np.linspace(ylim[0], ylim[1], 30)
    XX, YY = np.meshgrid(xx, yy)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)
    ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])
# 绘制SVM分类器的分类结果
plt.subplot(1, 2, 1)
plot_boundary(svm, X, y)
plt.title('SVM')

# 绘制逻辑回归分类器的分类结果
plt.subplot(1, 2, 2)
plot_boundary(lr, X, y)
plt.title('Logistic Regression')

plt.show()
```

```python
# 计算评价指标
print('Logistic Regression:')
print(f'Accuracy: {accuracy_score(y_test, lr_pred):.2f}')
print(f'Precision: {precision_score(y_test, lr_pred):.2f}')
print(f'Recall: {recall_score(y_test, lr_pred):.2f}')
print(f'F1 score: {f1_score(y_test, lr_pred):.2f}')
print()

print('SVM:')
print(f'Accuracy: {accuracy_score(y_test, svc_pred):.2f}')
print(f'Precision: {precision_score(y_test, svc_pred):.2f}')
print(f'Recall: {recall_score(y_test, svc_pred):.2f}')
print(f'F1 score: {f1_score(y_test, svc_pred):.2f}')
```

结果：

```
Logistic Regression:
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 score: 1.00

SVM:
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 score: 1.00
```



通过图像可以看出二者的区别：
逻辑回归是一种线性分类算法，它的分类边界是一条直线或平面，这条直线或平面可以将不同的数据点分成两类。逻辑回归使用 sigmoid 函数将线性函数的输出值压缩到 0 到 1 之间，表示数据点属于某一类的概率。然后将这个概率与一个阈值比较，以确定数据点所属的类别。

SVM 则是一种非线性分类算法，它的分类边界是一个曲面，可以更好地适应数据点的分布。SVM 通过找到支持向量，即距离分类边界最近的数据点，来确定分类边界。SVM 的目标是找到能够最大化支持向量和分类边界之间的间隔的分类边界，这个间隔被称为“最大边界(margin)”。

评价指标：
准确率（Accuracy）：分类器正确分类的样本数占总样本数的比例。这是最基本的评价指标之一，但不适合在类别分布不均的情况下使用。
精准率（Precision）：指分类器预测为正例的样本中，真正为正例的样本所占的比例。适合在正例样本较少、不希望将负例样本误分类为正例时使用。
召回率（Recall）：指真正为正例的样本中，被分类器预测为正例的样本所占的比例。适合在正例样本较多、不希望将正例样本误分类为负例时使用。
F1 分数（F1 score）：综合考虑精准率和召回率的指标，其计算公式为：2 * (精准率 * 召回率) / (精准率 + 召回率)。适合在对分类器综合性能进行评价时使用。

三、总结（心得体会）

SVM 是一种非常强大的分类算法。它在处理线性可分和线性不可分的数据时都表现出色，而且对于高维数据和大规模数据也能很好地处理。它的基本思想是找到一个能够最大化数据间距离的超平面，从而将不同类别的数据分开。并且它有很多参数需要调节。比如，需要选择核函数的类型、核函数的参数、惩罚因子的大小等等。在实际应用中，需要根据数据的特点来调节这些参数，以达到最好的分类效果。