西北大学

智能信息系统综合实践
实验报告

题　　目：　PythonSVM

年　　级：　2020

专　　业：　软件工程

姓　　名：　贺子璇

一、题目（原题目）

利用 Iris 数据集，交叉验证，测试 SVM 分类器性能，改变超参数，对比不同超参数对结果的影响。

对比 SVM 与逻辑回归分类器。

二、解题步骤（思路+代码）

1.SVM 超参数对结果影响

理论支持：

SVM 目标最大化 margin 且不同类别观测值分布在超平面异侧，训练数据在超平面的计算中以标量积出现，故引入核函数。处理线性不可分数据时将数据映射到更高维找到超平面进行线性分离，再回到原始维度，不再计算超平面参数，记住 SVM。

实验思路：

利用**控制变量法，可视化**不同核函数下每个超参数对 SVM 结果的影响子图和对应性能，进行观察。

具体步骤：

SVM 类定义：初始化模型，添加控制变量下的模型预测结果图

代码如下

```
def __init__(self, X_train, y_train, X_test, y_test,
             n_svm_x_axis=1, n_svm_y_axis=1, figsize=(12, 9)):
    self.X_train = X_train
    self.y_train = y_train
    self.X_test = X_test
    self.y_test = y_test
    self.n_svm_x_axis = n_svm_x_axis
    self.n_svm_y_axis = n_svm_y_axis
    self.plt = plt.figure(figsize=figsize)

    # 网格数据绘制不同类别不同颜色
    self.xx, self.yy = np.meshgrid(np.linspace(-3, 3, 200),
                                   np.linspace(-3, 3, 200))
    self.grid = np.c_[self.xx.ravel(), self.yy.ravel()]
```

```
# 可视化添加控制变量下的分类图像
def add_svm_plot(self, svm, title, k=0):
    # 可视化界面
    plt.subplot(self.n_svm_x_axis, self.n_svm_y_axis, k + 1)
    plt.title(title, size='medium')

    # 评估学习到的模型
    preds = svm.predict(self.grid).reshape(self.xx.shape)

    # 可视化不同参属下模型的分类情况 并展示评估结果
    # 不同类别不同颜色表示
    plt.pcolormesh(self.xx, self.yy, preds, cmap=ListedColormap(['lightblue', 'lightcoral']),
                   vmin=0, vmax=1)
    # 可视化支持向量
    plt.scatter(self.X_train[:, 0], self.X_train[:, 1], label="Train data",
                edgecolors='k', c=self.y_train, cmap=ListedColormap(['blue', 'red']))
    plt.scatter(self.X_test[:, 0], self.X_test[:, 1], label="Test data", edgecolors='k')
    plt.xticks(())
    plt.yticks(())
    plt.tight_layout()
```

```
# 在分类模型中展示两侧的支持向量和margin
# 决策函数将数据映射到高维空间，并找到最有分割超平面 输出：数据点到超平面的距离，
# 正值为一类 负值为一类
Z = svm.decision_function(self.grid).reshape(self.xx.shape)
plt.contour(self.xx, self.yy, Z, colors='k', levels=[-1, 0, 1],
            alpha=0.5, linestyles=['--', '-', '--'])
plt.scatter(svm.support_vectors_[:, 0], svm.support_vectors_[:, 1], s=100,
            facecolors='none', zorder=10, edgecolors='k')
```

处理数据：为方便可视化处理，选取前两类的两个特征

```
iris = load_iris()
X_2d = iris.data[:, :2]
y_2d = iris.target

if keep == "nonlinear_relationship":
    X_2d = X_2d[y_2d > 0]
    y_2d = y_2d[y_2d > 0]
    y_2d -= 1
else:
    X_2d = X_2d[y_2d < 2]
    y_2d = y_2d[y_2d < 2]

return X_2d, y_2d
```

绘制数据分布散点图，判断两类数据是线性可分的，如图 1.1

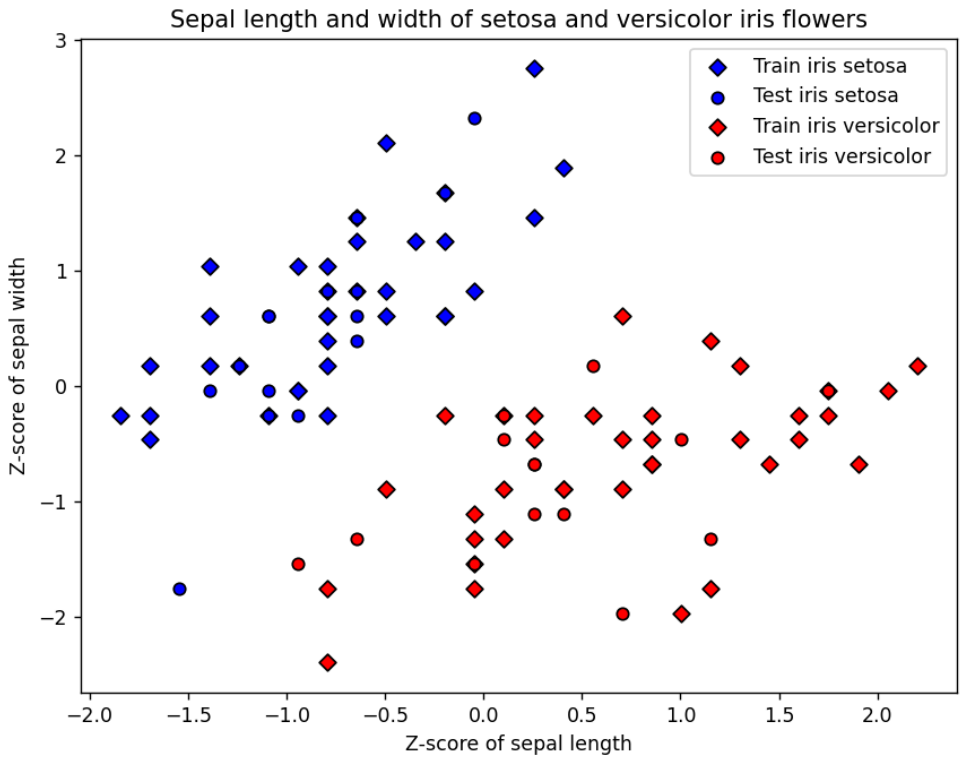

图 1.1

交叉验证法，8：2 拆分数据集

```
# 交叉验证法-拆分数据 8：2
X_2d, y_2d = load_iris_2d(keep="linear_relationship")
X_train, X_test, y_train, y_test = train_test_split(X_2d, y_2d, test_size=0.3,
                                                    random_state=8)
```

数据归一化处理：

```
# 数据归一化处理
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

通过控制变量法验证不同超参数对 SVM 结果的影响，并通过可视化结果及数学原理分析更好理解实验结果。

控制变量法：

线性核函数：不映射到高维，观察超参数 C 的影响

```
# 变量：C
C_range = [5e-3, 1e-2, 5e-2, 1e-1, 0.5, 1, 3, 5, 10]
visualizer = SvmVisualizer(X_train, y_train, X_test, y_test, 3, 3)
k = 0

for C in C_range:
    clf = svm.SVC(C=C, kernel="linear", random_state=8)
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    title = f"R2 score={score:.2f}. C={C}"
    visualizer.add_svm_plot(clf, title, k)
    k += 1
```

由图 1.2，其中有白色边框的点是特征向量，虚线为边缘，无论纵向或横向观察，可得，随着 C 增大，margin 越来越小，定义超平面所需的支持向量越来越少，得分越来越高。

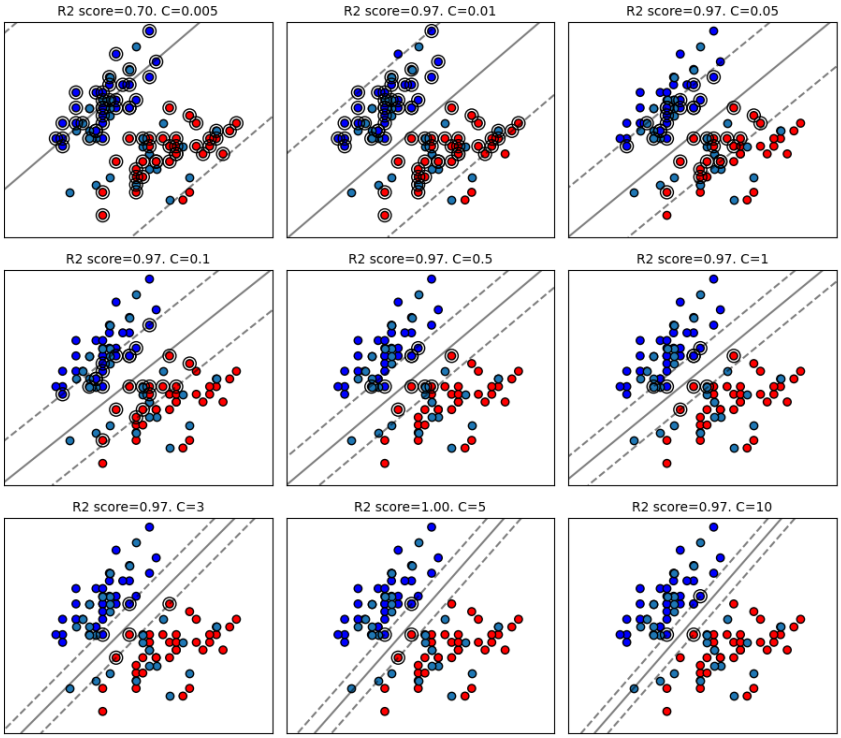

图 1.2

RBF 核函数

新超参数，将数据映射到更高维，探究 C 和超参数 γ 的影响

```
# 变量 C gama
C_range = [1e-1, 1, 1e1]
gamma_range = [1e-1, 1, 1e1]
visualizer = SvmVisualizer(X_train, y_train, X_test, y_test, 3, 3)
k = 0

for C in C_range:
    for gamma in gamma_range:
        clf = svm.SVC(C=C, gamma=gamma, kernel="rbf", random_state=8)
        clf.fit(X_train, y_train)
        score = clf.score(X_test, y_test)
        title = f"R2 score={score:.2f}. C={C}, gamma={gamma}"
        visualizer.add_svm_plot(clf, title, k)
        k += 1
```

由图 1.3，横向对比 γ，纵向对比 C，γ 越大，超平面越复杂，可以表示非线性分离关系，充当支持向量的数据点越多。

当 γ 足够高，训练集中的每个数据点都可以成为支持向量，会

导致过拟合和计算成本的增加。

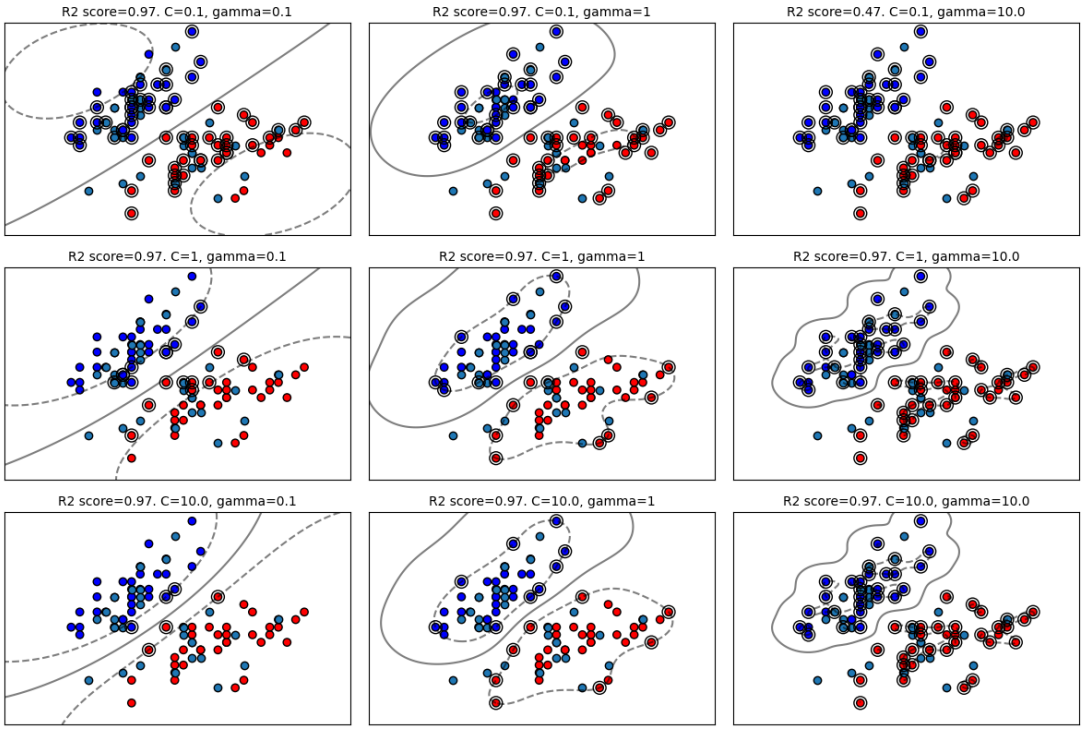

图 1.3

Sigmod 核函数

两个新超参数 r，γ’

C 在前两个核函数实验验证影响效果，在该核函数下 C 不变，控制 r，γ’ 的变化，水平方向 γ’ 不变，对比 r，同理，纵向对比 γ’ 。

由图 1.4，γ’ 增大使超平面更复杂，r 的影响效果从图中较难总结出规律，根据 sigmoid 核函数表达式 1.1 得常数水平平移了函数图像。

$$K(\vec{x_i}, \vec{x_j}) = tanh(\gamma \vec{x_i} \cdot \vec{x_j} + r) \quad (1.1)$$

```
# 变量  gama r d

C_range = [1]
gamma_range = [1e-1, 1, 1e1]
coef0_range = [-50, -1, 0, 1, 50]
visualizer = SvmVisualizer(X_train, y_train, X_test, y_test, 3, 5, figsize=(16, 10))
k = 0

for C in C_range:
    for gamma in gamma_range:
        for coef0 in coef0_range:
            clf = svm.SVC(C=C, gamma=gamma, coef0=coef0, kernel="sigmoid", random_state=8)
            clf.fit(X_train, y_train)
            score = clf.score(X_test, y_test)
            title = f"R2 score={score:.2f}. C={C},\n gamma={gamma}, r={coef0}"
            visualizer.add_svm_plot(clf, title, k)
            k += 1
plt.show()
```

结果

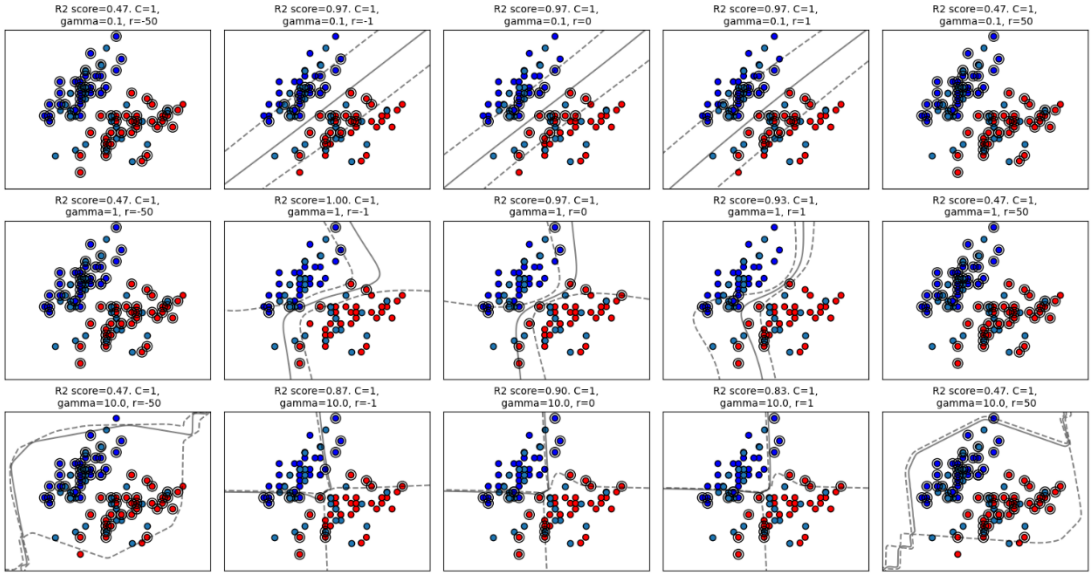

图 1.4

多核式核函数

三个新超参数：影响标量积γ’’，常量 r，多项式阶数 d

```
C_range = [1]
gamma_range = [0.1, 1]
degree_range = [2, 3]
coef0_range = [-50, -1, 0, 1, 50]
visualizer = SvmVisualizer(X_train, y_train, X_test, y_test, 4, 5, figsize=(20, 16))
k = 0

for C in C_range:
    for degree in degree_range:
        for gamma in gamma_range:
            for coef0 in coef0_range:
                clf = svm.SVC(C=C, gamma=gamma, degree=degree, coef0=coef0, kernel="poly", random_state=8)
                clf.fit(X_train, y_train)
                score = clf.score(X_test, y_test)
                title = f"R2 score={score:.2f}. C={C}, gamma={gamma},\n d={degree}, r={coef0}"
                visualizer.add_svm_plot(clf, title, k)
                k += 1

plt.show()
```

由图 1.5，d 的增大会增大 margin,其余指标与前分析相近。

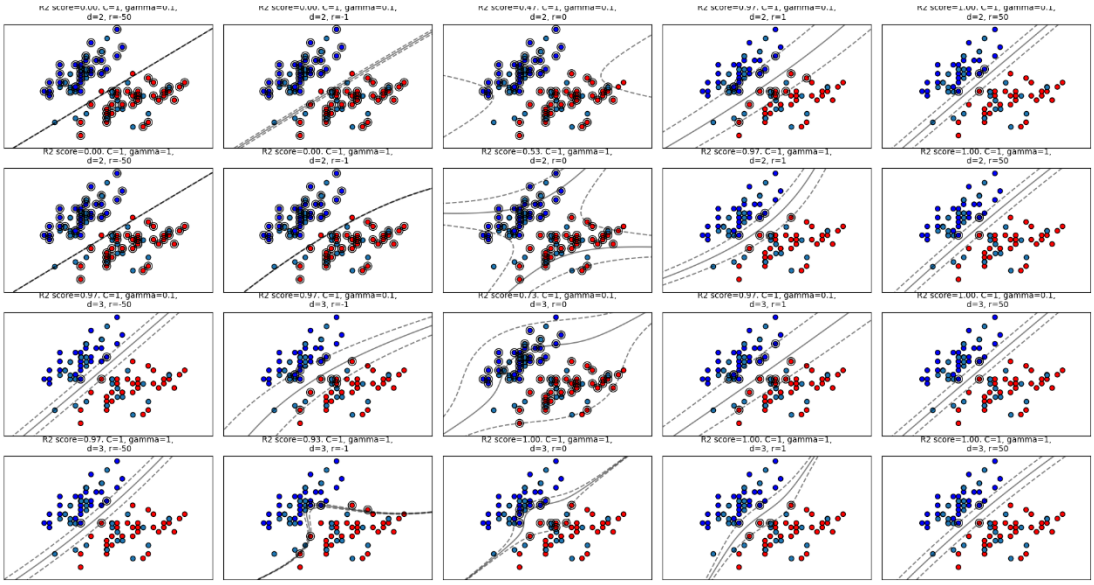

图 1.5

总结：由图像可以看出，超参数主要影响两个方面：margin/边缘，是否线性分离。

2.SVM vs 逻辑回归模型

1.指标选取

准确率一般用来评估模型的全局准确程度，不能包含太多信息，无法全面评价一个模型性能。如果一个分类器的性能比较好，那

么它应该有如下的表现：在 Recall 值增长的同时，Precision 的值保持在一个很高的水平。而性能比较差的分类器可能会损失很多 Precision 值才能换来 Recall 值的提高。故通常使用 P-R 曲线。平均精度 AP，MAP。AP 就是 Precision-recall 曲线下面的面积，通常来说一个越好的分类器，AP 值越高。

mAP 是多个类别 AP 的平均值。这个 mean 的意思是对每个类的 AP 再求平均，得到的就是 mAP 的值，mAP 的大小一定在[0,1]区间，越大越好。该指标是目标检测算法中最重要的一个。

加载数据并处理

```
# 导入数据并放缩

iris = datasets.load_iris()

# 取前两类
iris_X = iris.data
iris_X = iris_X[:100]
iris_y = iris.target
iris_y=iris_y[:100]

print(iris_y)
# 放缩
from  sklearn.preprocessing import StandardScaler

# min-max归一化
scaler = StandardScaler()
iris_X = scaler.fit_transform(iris_X)

# 选取两个特征简单观测实现结果
X=iris_X[:,:2]
```

```
# 默认参数下的逻辑回归模型
LogR = linear_model.LogisticRegression()
# 折数：CV=5

print("\t使用逻辑回归\t",cross_val_score(LogR,X,iris_y,cv=5,scoring='accuracy'

# 默认参数下SVM使用交叉验证
model = svm.SVC()
print("\t 使用SVM\t\t\t",cross_val_score(model,X,iris_y,cv=5,scoring='accurac

C=1
print("SVM-最佳score")
liner_svc = svm.SVC(kernel='linear',C=C).fit(X,iris_y)
rbf_svc = svm.SVC(kernel='rbf',gamma=0.7,C=C).fit(X,iris_y)
sig_svc = svm.SVC(kernel='sigmoid',C=C).fit(X,iris_y)
poly_svc = svm.SVC(kernel='poly',C=C).fit(X,iris_y)
```

```
交叉验证-平均score
    使用逻辑回归    0.8133333333333332
     使用SVM          0.8066666666666666
SVM-最佳score
    使用线性核函数的SVM     0.8133333333333334
    使用RBF的SVM   0.82
    使用sigmoid函数的SVM    0.7866666666666666
    使用多核式核函数的SVM    0.7466666666666667
```

图 2.1

根据图 2.1，在线性可分的情况下，逻辑回归的得分情况与 SVM 非常接近，并略优于默认参数下的 SVM，故拿到线性可分数据时，先尝试逻辑回归分类模型，其次可以使用 RBF 核函数下的 SVM 模型。

三、心得体会

1. 实践对比了 SVM 各核函数下超参数对模型结果的影响。
2. 对比了线性可分数据下各核函数 SVM 与线性逻辑回归模型各指标的差异。
3. 不足：没有针对各指标进行理论分析。