Lab x 实验题目

学 号	17363011	学 院	专 业
姓 名	陈政培	智能工程学院	智能科学与技术

1、实验目的： **sklearn 中的 SVM 应用实践**

2、实验环境：vs code、python3.7

3、实验步骤：

① 导入 iris 数据集

② 利用 SVC 估计器构造分类器

③ 模型预测

④ 利用 Grid_searchCV 寻找最佳参数

⑤ 利用获得最优参数预测

⑥ 加入 matplotlib 库，并测试不同的参数模型

4、实验结果与分析：

① 构造 SVM 分类器

② 用默认参数训练后，模型评估

```
PS C:\Users\93744\Desktop\code_and_data> & D:/Pytho
and_data/test.py
test_svc.score: 0.974
train_svc.score: 0.973
```

③ 模型预测结果

```
PS C:\Users\93744\Desktop\
and_data/test.py
test_svc.score: 0.974
train_svc.score: 0.973
[0]
```

④ Grid_searchCV 最佳参数

```
Best parameters:  {'C': 1, 'kernel': 'linear'}
```

⑤ 利用上一个问题选定的方案 C1，linear 参数预测

```
33    svc = SVC(C=1, kernel='linear')
34    svc.fit(X_train, y_train)
35    print("test_svc.score: {:.3f}".form
36    print("train_svc.score: {:.3f}".for
37
```

PROBLEMS TERMINAL ··· 2: Python

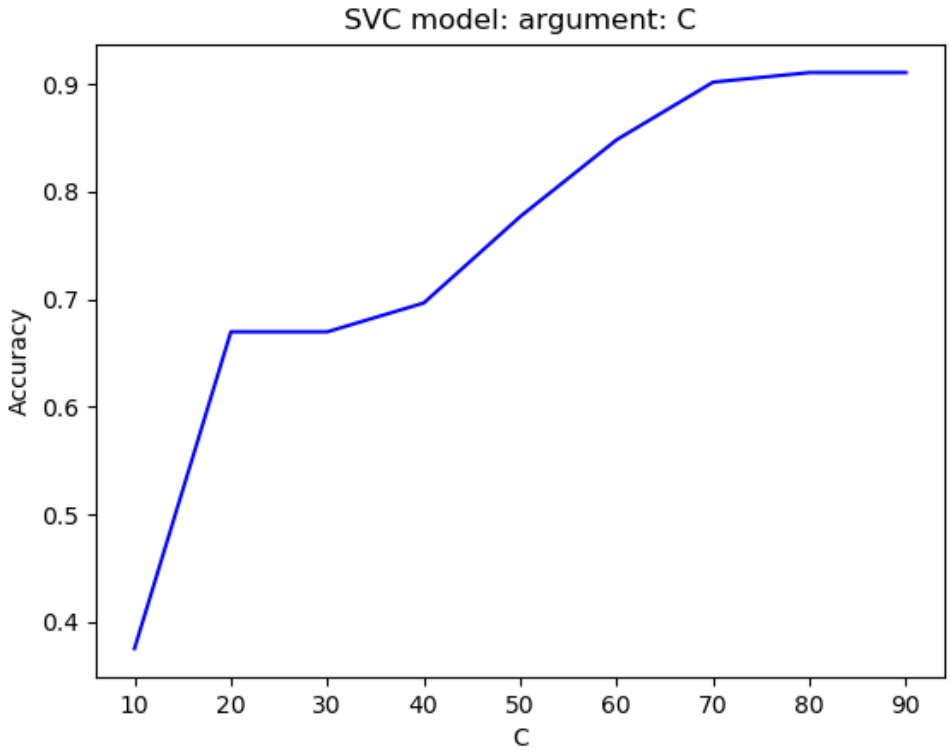
```
PS C:\Users\93744\Desktop\code_and_data> & D:/
4/Desktop/code_and_data/test.py
test_svc.score: 0.974
train_svc.score: 0.964
```

⑥ 运行 matplotlib 可视化不同参数模型的准确率变化

```
39    Cs = []
40    score = []
41    for c in range(10, 100, 10):
42        svc = SVC(C=c, gamma=0.001, kernel='rbf')
43        svc.fit(X_train, y_train)
44        accuracy = svc.score(X_train, y_train)
45        print("the accuracy of svc model with C= {0} is {1}".fo
46        Cs.append(c)
47        score.append(accuracy)
48
49    plt.title("SVC model: argument: C")
50    plt.xlabel("C")
51    plt.ylabel("Accuracy")
52    plt.plot(Cs, score, 'b')
53    plt.show()
```

PROBLEMS TERMINAL ··· 2: Python

```
4/Desktop/code_and_data/test.py
the accuracy of svc model with C= 10 is 0.375
the accuracy of svc model with C= 20 is 0.6696428571428571
the accuracy of svc model with C= 30 is 0.6696428571428571
the accuracy of svc model with C= 40 is 0.6964285714285714
the accuracy of svc model with C= 50 is 0.7767857142857143
the accuracy of svc model with C= 60 is 0.8482142857142857
the accuracy of svc model with C= 70 is 0.9017857142857143
the accuracy of svc model with C= 80 is 0.9107142857142857
the accuracy of svc model with C= 90 is 0.9107142857142857
```



5、作业：

① Decision_func.py ——可视化基于 iris 数据集的 SVM 分类器的决策边界

Decision_function()的功能主要是计算样本点到分割超平面的函数距离，将一组点集带入函数中，通过计算得到函数距离，并和标签集对照，找到分割超平面和支持向量。用法就是将点集 reshape 得到输入矩阵传入函数并计算

Matplotlib.pyplot.contour 的功能则是画出三维等高线图，方法是传入 x 和 y 作为横轴纵轴的大小，并以传入参数 z 为基准绘制等高线

可视化后的 iris 数据集的 SVM 分类器决策边界

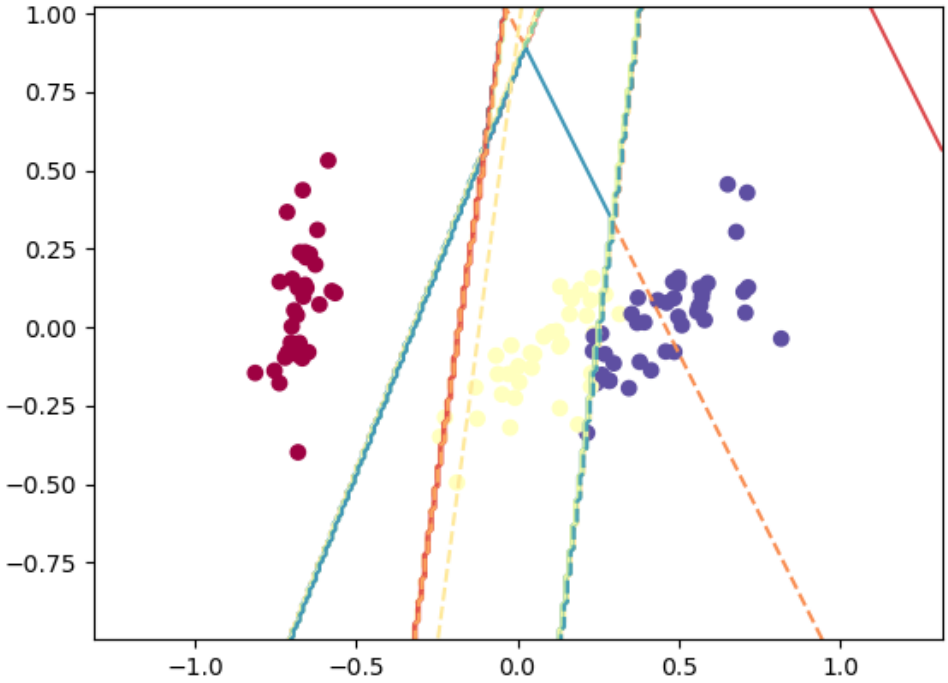
通过 PCA 提取其中的 2 个特征，并且把 3 个类别的分界线都绘制出来

```
48    print(X_train.shape)
49    x_min, x_max = X_train[:, 0].min() - .5, X_train[:, 0].max() + .5
50    y_min, y_max = X_train[:, 1].min() - .5, X_train[:, 1].max() + .5
51    h = 0.01
52    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
53    print(xx.shape)
54    print(yy.shape)
55    xy = np.vstack([xx.ravel(), yy.ravel()]).T
56    print(xy.shape)
57
58    Z = svc.decision_function(xy)
59    print(Z.shape)
60    Z1 = Z[:, 0].reshape(xx.shape)
61    Z2 = Z[:, 1].reshape(xx.shape)
62    Z3 = Z[:, 2].reshape(xx.shape)
63    # # 用预测函数预测一下
64    # Z = pred_func(np.c_[xx.ravel(), yy.ravel()])
65    # Z = Z.reshape(xx.shape)
66
67    # 然后画出图
68    plt.contour(xx, yy, Z1, cmap=plt.cm.Spectral, linestyles=['--','-','--'])
69    plt.contour(xx, yy, Z2, cmap=plt.cm.Spectral, linestyles=['--','-','--'])
70    plt.contour(xx, yy, Z3, cmap=plt.cm.Spectral, linestyles=['--','-','--'])
71    plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=plt.cm.Spectral)
72    plt.show()
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\93744\Desktop\code_and_data> & D:/Python37/python.exe c:/Users/93744/Desktop/co
(112, 2)
(203, 264)
(203, 264)
(53592, 2)
(53592, 3)
```

可视化决策边界

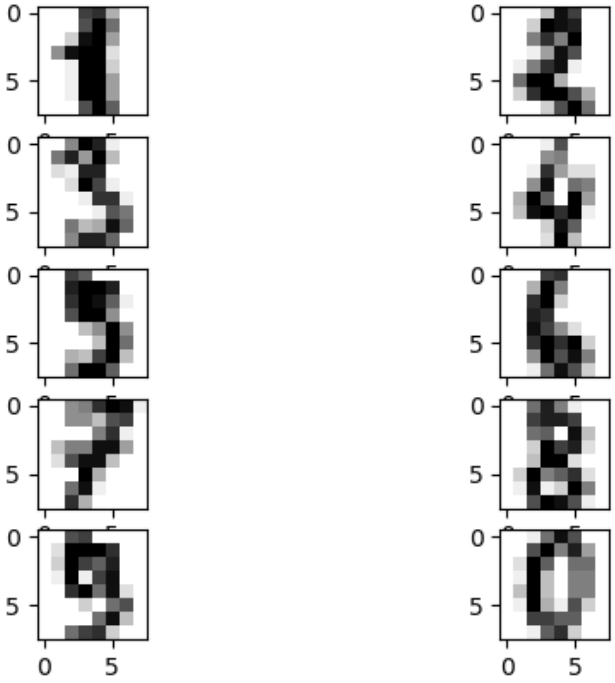


② Digits.py ——对手写数字体识别，利用 matplotlib 可视化测试集中前 10 个数字的灰度图像

运行结果（显示前 10 个和最后几个测试集）

```
PS C:\Users\93744\Desktop\code_and_data> &
/code_and_data/digits.py
实际数字:  [0 1 2 3 4 5 6 7 8 9]
预测数字:  [0 1 2 3 4 5 6 7 8 9]
实际数字:  [4 9 0 8 9 8]
预测数字:  [4 9 0 8 9 8]
```

显示前 10 个灰度图像



③ Horse.py——利用 SVR 预测马的死亡率，并通过网格搜索和随机搜索寻找最佳组合参数

导入训练和测试的数据集，进行格式的转换之后，就可以放到 LogisticRegression 回归分类器中进行分类。回归分类器和之前用到的 svc 参数有所不同，其中正则化系数 c 是一样的，但是优化函数参数叫做 solver，其中有五种可选参数

liblinear：内部使用了坐标轴下降法来迭代优化损失函数。

lbfgs：拟牛顿法的一种，利用损失函数二阶导数矩阵即海森矩阵来迭代优化损失函数。

newton-cg：也是牛顿法家族的一种，利用损失函数二阶导数矩阵即海森矩阵来迭代优化损失函数。

sag：即随机平均梯度下降，是梯度下降法的变种，和普通梯度下降法的区别是每次迭代仅仅用一部分的样本来计算梯度，适合于样本数据多的时候。

saga：线性收敛的随机优化算法的的变重

最优正确率可以达到 73.134328%（使用不同的参数组合）

```
PS C:\Users\93744\Desktop\code_and_data> & D:/P
正确率:73.134328%
Best parameters:  {'C': 10, 'solver': 'sag'}
PS C:\Users\93744\Desktop\code_and_data> & D:/P
正确率:73.134328%
Best parameters:  {'C': 1, 'solver': 'lbfgs'}
PS C:\Users\93744\Desktop\code_and_data> & D:/P
正确率:71.641791%
Best parameters:  {'C': 1, 'solver': 'lbfgs'}
PS C:\Users\93744\Desktop\code_and_data> & D:/P
正确率:71.641791%
Best parameters:  {'C': 1, 'solver': 'lbfgs'}
PS C:\Users\93744\Desktop\code_and_data> & D:/P
正确率:71.641791%
```

利用网格搜索和随机搜索寻找最佳组合参数

```
45      tuned_parameters = [{'solver': ['sag'], 'C': [1, 10, 100, 1000]},
46                          {'solver': ['lbfgs'], 'C': [1, 10, 100, 1000]},
47                          {'solver': ['newton-cg'], 'C': [1, 10, 100, 1000]},
48                          {'solver': ['saga'], 'C': [1, 10, 100, 1000]},
49                          {'solver': ['liblinear'], 'C': [1, 10, 100, 1000]}]
50      clf = GridSearchCV(classifier, tuned_parameters)
51      clf.fit(trainingSet, trainingLabels)
52      print("Best parameters: ",clf.best_params_)
53      tuned_parameters = [{'solver': ['sag'], 'C': [1, 10, 100, 1000]},
54                          {'solver': ['lbfgs'], 'C': [1, 10, 100, 1000]},
55                          {'solver': ['newton-cg'], 'C': [1, 10, 100, 1000]},
56                          {'solver': ['saga'], 'C': [1, 10, 100, 1000]},
57                          {'solver': ['liblinear'], 'C': [1, 10, 100, 1000]}]
58      clf = RandomizedSearchCV(lr, tuned_parameters)
59      clf.fit(trainingSet, trainingLabels)
60      print("Best parameters: ",clf.best_params_)
```

运行后结果:

```
PS C:\Users\93744\Desktop\code_and_data> & D:/Pyth
正确率:73.134328%
Best parameters:  {'C': 1, 'solver': 'lbfgs'}
Best parameters:  {'solver': 'saga', 'C': 10}
```

其中网格搜索得到的参数组合正确率只有 71.641791%，而随机搜索得到的参数组合就能够得到最优的正确率 73.134328%

6、实验总结：

① 在可视化分类器决策变量时，由于 iris 数据集本身具有 centers=4，所以不能够直接只用 PPT 中的代码，需要通过 PCA 提取两个主成分进行边界计算。并且通过打印 dicision_function()函数生成的边界矩阵参数发现生成了 3 个不同的类别，有 3 个分界线，所以需要分别选取 3 个分界线进行 reshape，然后共同绘制到图像中

② 关于通过网格搜索和随机搜索得到最佳组合参数的时候，放入 SearchCV 函数中的分类器参数也会影响到输出的结果，并不是每一次都会得到一样的结果。liblinear 适用于小数据集，而 sag 和 saga 适用于大数据集因为速度更快等因素

参考文献:

1. https://blog.csdn.net/steve_d/article/details/85798641
2. https://blog.csdn.net/c406495762/article/details/77851973
3. https://blog.csdn.net/qq_33039859/article/details/69810788