SVM 实验报告

林宝诚 2019080030

运行环境：WSL 2 Ubuntu 20.04 LTS, Python 3.8.5

任务 1：从 10 个类别中随机选择一个类别作为主类别，并利用二分类线性 SVM 进行分类。

源代码：svm_sklearn_q1.py

需要的 Python 库：scikit-learn, NumPy, SciPy, Matplotlib

代码思路：

- 使用 SciPy 将所有的训练和测试数据录入（根据模板），其中：
  - x_train, x_test 分别为训练和测试数据的输入
  - y_train, y_test 分别为训练和测试数据的输出

- 利用 NumPy 随机产生 1 至 10 的整数作为主类别，并将主类别的输出标签改为 0，非主类别的输出标签则改为 1
- 利用 scikit-learn 的 LinearSVC 进行二分线性分类，所有参数均为缺省值
- 计算训练和测试集的准确率，并画出 confusion matrix 观察分类的情况

实验结果

主类别为 1

```
Master Class:  1
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9979057591623036
```

主类别为 2

```
Master Class:  2
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9947643979057592
```

主类别为 3

```
Master Class:  3
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9696335078534032
```

主类别为 4

```
Master Class:  4
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9675392670157068
```

主类别为 5

```
Master Class:  5
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9727748691099476
```

主类别为 6

```
Master Class:  6
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9884816753926702
```

主类别为 7

```
Master Class:  7
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9863874345549738
```

主类别为 8

```
Master Class:  8
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9612565445026178
```

主类别为 9

```
Master Class:  9
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9947643979057592
```

主类别为 10

```
Master Class:  10
Accuracy Score on Train Set:  1.0
Accuracy Score on Test Set:  0.9968586387434555
```

**Confusion Matrix 则在’ConfusionMatrix’文件夹里给出，命名方式为 'Confusion Matrix for Linear Binary Classifier, Master Class: 【对应的主类别】'

结果分析

- 可以看到无论是哪个类别作为主类别，训练准确率皆为 100%，证明在这样的情况下测试集都是二分线性可分的
- 测试准确率并未达到 100%，证明同一类别的数据并非十分集中，会有一些数据与其他类别的数据距离十分靠近，或参杂在一起。造成上述情况的因素可能为标签错误，或者分类问题本质上并不是线性可分的。

任务 2：利用凸优化基本工具包设计 SVM 二分类器的基本代码框架

源代码：svm_cvxpy_q2.py

需要的 Python 库：CVXPY, NumPy, SciPy, scikit-learn

代码思路：

- 使用 SciPy 将属于类别 1 和类别 2 的训练和测试数据录入（根据模板），并将类别 2 的标签更改为 2，其中：
  - x_train, x_test 分别为训练和测试数据的输入
  - y_train_n, y_test_n 分别为训练和测试数据的输出

- 填写优化问题的目标函数和约束条件，并使用 CVXPY 解出 w（权重）和 b（偏置）
- 利用所计算的参数在训练和测试集进行预测，并计算准确率，再与 scikit-learn 中的 LinearSVC 进行比较（所有参数均为缺省值）

实验结果

```
Accuracy Score on Train Set(cvxpy):  1.0
Accuracy Score on Test Set(cvxpy):  0.9927007299270073
Accuracy Score on Train Set(sklearn SVM):  1.0
Accuracy Score on Test Set(sklearn SVM):  0.9927007299270073
```

结果分析

- 可见两者所计算的结果相似，因此得以验证使用 CVXPY 所设计的 SVM 二分类器。（一般来说，两者会有所差异，因为 sklearn 里的 SVM 有其他超参数，可能会影响最终结果）

任务 3：利用线性 SVM 在全部训练集上进行学习，再进行预测，并调整参数观察结果变化

源代码：svm_sklearn_q3.py

需要的 Python 库：scikit-learn, NumPy, SciPy, Matplotlib

代码思路：

- 使用 SciPy 将所有的训练和测试数据录入（根据模板），其中：
  - x_train, x_test 分别为训练和测试数据的输入
  - y_train, y_test 分别为训练和测试数据的输出

- 利用 scikit-learn 的 SVM 进行分类
- 计算训练和测试集的准确率，并画出 confusion matrix 观察分类的情况

参数设定：

- C 值：0.01，1，100
- 核函数:Linear, RBF, Poly
- Degree（仅用于 Poly 核函数）：2，3
- 其他参数为缺省值

分类器类别

本质上，SVM 是个二分类器。若要将 SVM 运用在许多类别的分类上，可采取如下的方式：

1. One Vs One
   - 所有类别中，对任意两个类别训练一个分类器
   - 分类器总数为 $\frac{n(n-1)}{2}$，其中 n 为类别数目

2. One Vs Rest
   - 与任务 1 的想法相似，即选取一个主类别，将其他类别视为非主类别，并进行分类
   - 总共训练 n 个分类器

两种方式的比较

One Vs One	One Vs Rest
分类时使用投票形式,投票最多者为未知样本的类别	分类时将未知样本分类为具有最大分类函数值的那类。
当类别数量多时,需要的分类器数量大幅增加,计算所损耗的资源多	由于负集的数据量往往比正集数据量大得多,容易出现 biased

实验结果与分析

1. Linear,
   - C=0.01

```
Paramters:
C= 0.01
Kernel= linear
Accuracy Score on Train Set (One Vs One):  1.0
Accuracy Score on Test Set (One Vs One):  0.9151832460732985
Accuracy Score on Train Set (One Vs Rest):  1.0
Accuracy Score on Test Set (One Vs Rest):  0.9350785340314136
```

   - C=1

```
Paramters:
C= 1
Kernel= linear
Accuracy Score on Train Set (One Vs One):  1.0
Accuracy Score on Test Set (One Vs One):  0.9151832460732985
Accuracy Score on Train Set (One Vs Rest):  1.0
Accuracy Score on Test Set (One Vs Rest):  0.9350785340314136
```

   - C=100

```
Paramters:
C= 100
Kernel= linear
Accuracy Score on Train Set (One Vs One):  1.0
Accuracy Score on Test Set (One Vs One):  0.9151832460732985
Accuracy Score on Train Set (One Vs Rest):  1.0
Accuracy Score on Test Set (One Vs Rest):  0.9350785340314136
```

分析：若使用线性分类器，通过调整 C 值并不会增加测试集的准确率，这是因为在训练集上他们都是线性可分的，而 C 值则是与错误分类点的惩罚有关，在训练集上并无错误分类点，因此 C 值不会对结果产生影响。

2. RBF
   - C=0.01

```
Paramters:
C= 0.01
Kernel= rbf
Accuracy Score on Train Set (One Vs One):  0.9333333333333333
Accuracy Score on Test Set (One Vs One):  0.8345549738219895
Accuracy Score on Train Set (One Vs Rest):  0.99
Accuracy Score on Test Set (One Vs Rest):  0.9151832460732985
```

   - C=1

```
Paramters:
C= 1
Kernel= rbf
Accuracy Score on Train Set (One Vs One):  0.9966666666666667
Accuracy Score on Test Set (One Vs One):  0.8984293193717278
Accuracy Score on Train Set (One Vs Rest):  0.9966666666666667
Accuracy Score on Test Set (One Vs Rest):  0.9319371727748691
```

   - C=100

```
Paramters:
C= 100
Kernel= rbf
Accuracy Score on Train Set (One Vs One):  1.0
Accuracy Score on Test Set (One Vs One):  0.9172774869109948
Accuracy Score on Train Set (One Vs Rest):  1.0
Accuracy Score on Test Set (One Vs Rest):  0.9403141361256544
```

分析：随着 C 的增加，分类器在训练集和测试集上的准确率都相对提高，在 C=100 时，准确率超越了线性的 SVM 分类器。当 C 很小时，分类器对离群点的容忍程度很高，于是尽可能将最大分类间隔变大。通过观察实验结果，对于此数据集，C 过小会导致欠拟合，因此准确率较低。随着 C 的增加，数据的拟合逐渐好转，因此准确率增加。另外，RBF 为分类器引入了非线性的性质，因此不同数据类别之间的分隔能够为曲线，相比线性分类器伸缩性较大，准确率也会因而变好（当然也有可能会变成过拟合，届时测试集的准确率就会降低）。

3. Poly
   - C=0.01, degree=2,3

```
Paramters:
C= 0.01
Kernel= poly
degree= 2
Accuracy Score on Train Set (One Vs One):  0.7566666666666667
Accuracy Score on Test Set (One Vs One):  0.7193717277486911
Accuracy Score on Train Set (One Vs Rest):  0.9666666666666667
Accuracy Score on Test Set (One Vs Rest):  0.8858638743455497
degree= 3
Accuracy Score on Train Set (One Vs One):  0.8733333333333333
Accuracy Score on Test Set (One Vs One):  0.8094240837696335
Accuracy Score on Train Set (One Vs Rest):  0.98
Accuracy Score on Test Set (One Vs Rest):  0.9162303664921466
```

   - C=1, degree=2

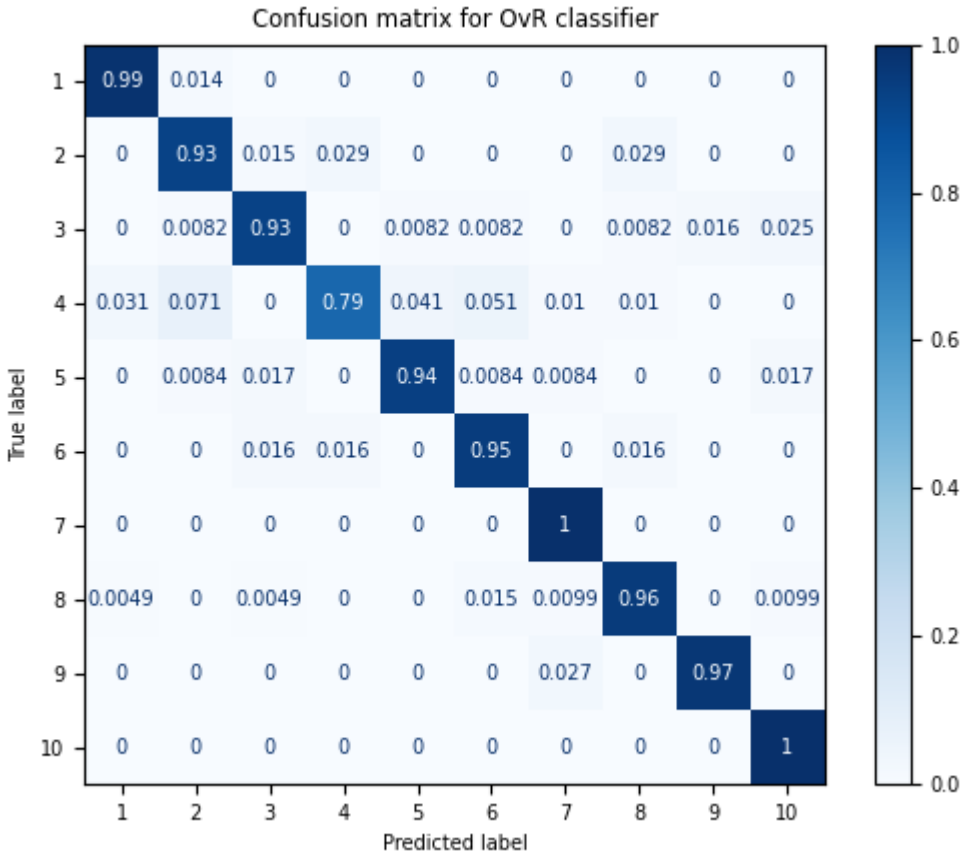
```
Paramters:
C= 1
Kernel= poly
degree= 2
Accuracy Score on Train Set (One Vs One):  1.0
Accuracy Score on Test Set (One Vs One):  0.9026178010471204
Accuracy Score on Train Set (One Vs Rest):  1.0
Accuracy Score on Test Set (One Vs Rest):  0.93717277486911
degree= 3
Accuracy Score on Train Set (One Vs One):  1.0
Accuracy Score on Test Set (One Vs One):  0.875392670157068
Accuracy Score on Train Set (One Vs Rest):  1.0
Accuracy Score on Test Set (One Vs Rest):  0.9089005235602095
```

   - C=100, degree=2,3

```
Paramters:
C= 100
Kernel= poly
degree= 2
Accuracy Score on Train Set (One Vs One):  1.0
Accuracy Score on Test Set (One Vs One):  0.9036649214659686
Accuracy Score on Train Set (One Vs Rest):  1.0
Accuracy Score on Test Set (One Vs Rest):  0.9267015706806283
degree= 3
Accuracy Score on Train Set (One Vs One):  1.0
Accuracy Score on Test Set (One Vs One):  0.875392670157068
Accuracy Score on Train Set (One Vs Rest):  1.0
Accuracy Score on Test Set (One Vs Rest):  0.9089005235602095
```

分析：基本与 RBF 的分析类似。在 C 过小时，数据欠拟合，因此准确率低。通过实验我们可以发现，在给定的参数下，RBF 的表现要优于 Poly。（核函数的选择并没有任何硬性规定，基本上都是 trial-and-error）

综合分析：通过观察 Confusion Matrix，不难发现标签为 4 的数据类别分类准确率都较低，这有可能是因为此测试数据中有较多离群点，或者数据本质上较不集中（与其他类别有些重叠）但训练数据少并没充分反映出来。解决的方案包括检查并更正错误的标签和增加训练的数据。（也可能需要测试多几个超参数或者调整除 C 值和核函数以外的其他超参数才能提高测试准确率。）



例： Confusion Matrix with kernel=RBF, C=100