人工智能及其应用实验报告

花国栋

2023 年 1 月 13 日

目录

1 实验简介

1.1 实验目的

1. 掌握 python 的基础数学操作与科学计算库。
2. 学习数据预处理的各种基本方法。
3. 了解 SVM 与 PCA 原理，掌握实际的代码应用以及如何优化超参数。

1.2 实验要求

选取合适的算法对给定的数据集训练与测试，进行参数优化，计算算法评价的指标。

1.3 研究方法

首先读取数据，然后统计缺失值，本实验中无缺失值，故接下来拆分特征列与目标列，然后进行数据集划分，其中训练集占比 80%，测试集占比 20%，接着对特征进行 PCA 降维，然后进行数据的归一化，最后构建 SVM 分类模型，通过网格搜索与 10 折交叉验证寻找最佳分类器，接着再用最佳分类器在测试集上测试，使用混淆矩阵与分类报告进行算法模型的分类效果的评估。

1.4 数据集介绍

该数据集一共有 11 个特征值和一个目标值，共 1599 组，分别为：

1-固定酸度 2-挥发性酸度 3-柠檬酸 4-残留糖 5-氯化物 6-游离二氧化硫 7-总二氧化硫 8-密度 9-pH 10-硫酸盐 11-醇 12-质量

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

图 1: 数据前五行

1.5 技术工具

- Python 版本 3.9.15
- 代码编辑器 Visual Studio Code
- 虚拟环境配置工具 Anaconda
- 辅助工具 Jupyter Notebook

2 实验步骤及算法简述

2.1 导入实验相关库

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
```

2.2 读取数据

使用 pandas 库的 read_csv() 方法读取数据，并进行是否有缺失值的判断。

```
data=pd.read_csv('winequality-red.csv',sep=';') #从文件中读取data
#print(data.isnull().sum())  #统计有几个缺失值，为0所以不做后续处理
```

2.3 拆分特征列与目标列

从之前数据集的组成可以看到，label=quality 的那一列为目标列，剩余的则为特征列，由于 quality 分布区间为 0-10，分布太广，且过于细致，所以将质量分为两个等级：劣质，优质。将多分类问题转化为二分类问题，也有利于缓解因为数据集过少训练出的模型不准确的问题。

```
X = data.drop(labels='quality',axis=1).copy() #X是特征列
y = data['quality'].copy()                    #y是目标列
for i in range(1599):
    y[i]=np.where(y[i]<=5,-1,1)#将Quality大于5为优质，定为1，反之则为劣质，定为-1
```

2.4 划分训练集与测试集

用 sklearn 库的 train_test_split 的方法按照 8:2 的比例划分训练集与测试集，随机种子 random_state 设为固定值 0，这是为了让后面的 PCA 降维时对同一划分的数据集进行手动调参。

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

2.5 PCA 降维

主成分分析 (Principal Component Analysis,PCA) 算法能够将高维问题简化成低维问题，具有简单、快速，且主成分之间相互正交，可消除原始数据成分间的影响。观察原数据集结合常识直觉，可以发现 pH 与固定酸度、挥发性酸度等都有相关性，故可以借助 PCA 降维，去除噪声和不重要的特征。在 random_state 不变的情况下，对维度 n_components 进行手动选择，

可以发现维度在 7 时效果最好，这也符合数学直觉，当维度过低时，特征损失较大，当维度过高时，PCA 效果不明显。

```
pca=PCA(n_components=7)
pca.fit(X_train)
X_train_pca=pca.transform(X_train)
X_test_pca=pca.transform(X_test)
```

2.6 归一化

使用了 sklearn 库的 MinMaxScaler 方法对原始数据进行线性变换，变换到 [0,1] 区间。计算公式为：

$$X_{scaled} = \frac{X - \min}{\max - \min}(new\_\max - new\_\min) + new\_\min \tag{1}$$

```
X_scaler = MinMaxScaler()
X_train_scaled=X_scaler.fit_transform(X_train_pca)
X_test_scaled=X_scaler.transform(X_test_pca)
```

2.7 网格搜索 & 交叉验证来最优化 SVM 模型

分类问题选用 SVM 模型，sklearn 库中的 SVC 方法。支持向量机 (Support Vector Machine,SVM)，通俗来讲，它是一种二类分类模型，其基本模型定义为特征空间上的间隔最大的线性分类器，其学习策略便是间隔最大化，最终可转化为一个凸二次规划问题的求解。

在使用 SVM 对数据进行训练的时候，为了使得在模型准确度达到一定的要求的条件下，模型的泛化能力尽量更好，防止在训练过程中模型的过拟合。因此我们需要不断的调整 SVC() 方法中 gamma 和 C 的值，并对数据不断地进行交叉验证，以找到合适的 gamma 和 C 的值。

参数 C 的作用：C 是惩罚系数，理解为调节优化方向中两个指标（间隔大小，分类准确度）偏好的权重，即对误差的宽容度。C 越高，说明越不能容忍出现误差，容易过拟合；C 越小，容易欠拟合。C 过大或者是过小，泛化能力都会变差。

参数 gamma 的作用：gamma 是选择 RBF 函数作为 kernel 后，该函数自带的一个参数。隐含地决定了数据映射到新的特征空间后的分布，gamma 越大，支持向量越少；gamma 越小，支持向量越多。支持向量的个数影响训练与预测的速度。

网格搜索 (Grid Search) 就是在 C,gamma 组成的二维参数矩阵中，依次实验每一对参数的效果，可以得到全局最优。

k 折交叉验证 (K-fold Cross-Validation)，将训练集分为 k 份，其中 k-1 份为训练集，剩下一份为验证集，用训练集训练模型，再在验证集上进行测试，获得其在验证集上的 Accuracy,k 次之后取平均。

使用 sklearn 库的 GridSearchCV 方法，进行网格搜索结合 k 折交叉验证获得最好的超参数 gamma 和 C 后，代入模型，将训练集上训练模型。

```
params = {
    'gamma':[0.01, 0.1, 0.5, 1, 2, 10, 100],
    'C': [0.01, 0.1, 0.5, 1, 2, 10, 100]
    }
```

```
clf=svm.SVC()#选用SVM模型
grid=GridSearchCV(clf,params,cv=10,n_jobs=-1)
grid.fit(X_train_scaled,y_train)
```

2.8 结果展示

```
y_pred=grid.predict(X_test_scaled)
print('最佳分类器：',grid.best_estimator_)
print("混淆矩阵\n",confusion_matrix(y_test,y_pred))
print("分类报告\n",classification_report(y_test,y_pred))
```

```
最佳分类器:  SVC(C=2, gamma=100)
混淆矩阵
 [[105  43]
 [ 29 143]]
分类报告
              precision    recall  f1-score   support

          -1       0.78      0.71      0.74       148
           1       0.77      0.83      0.80       172

    accuracy                           0.78       320
   macro avg       0.78      0.77      0.77       320
weighted avg       0.78      0.78      0.77       320
```

图 2: 分类器在测试集上的混淆矩阵与分类报告

2.9 PR 曲线绘制

PR 曲线中的 P 代表的是 Precision（查准率），R 代表的是 Recall（查全率），其代表的是精准率与召回率的关系，一般情况下，将 Recall 设置为横坐标，Precision 设置为纵坐标。使用的是 matplotlib 库中的绘图方法。

```
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
plt.figure()
plt.plot(recall, precision)
plt.title('Precision Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.show()
```

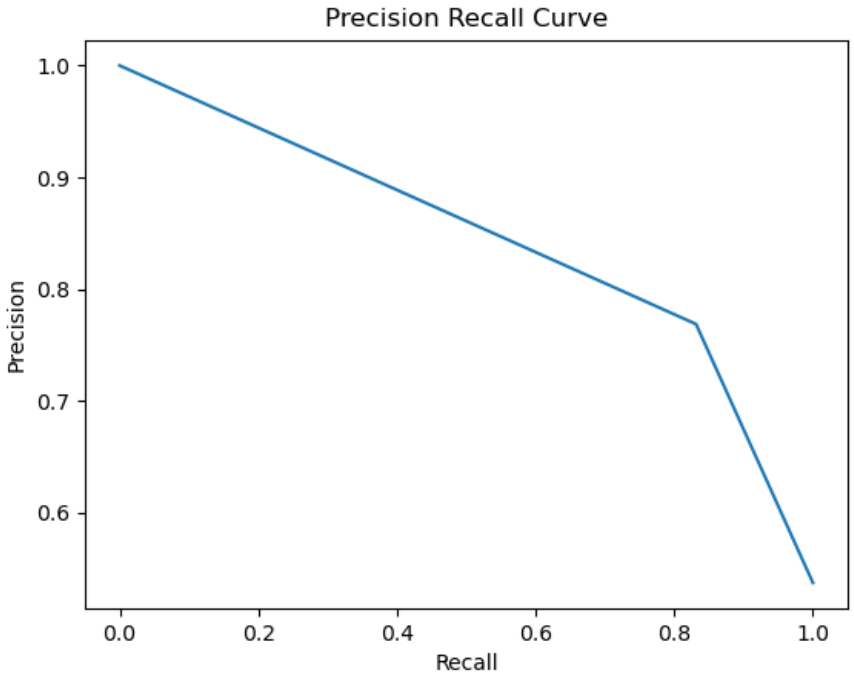

图 3: PR 曲线

3 结果说明

表 1: 混淆矩阵

混淆矩阵		预测结果	
		正例	反例
真实情况	正例	TP(真正例)	FN(假反例)
	反例	FP(假正例)	TN(真反例)

查准率:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

查全率:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

准确度:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + +TN + FN} \tag{4}$$

F1 score:

$$\text{F1} = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

4 实验源码

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt

#读取数据
data=pd.read_csv('winequality-red.csv',sep=';') #从文件中读取data
#print(data.isnull().sum())  #统计有几个缺失值，为0所以不做后续处理

#拆分特征列与目标列
X = data.drop(labels='quality',axis=1).copy() #X是特征列
y = data['quality'].copy()                    #y是目标列
for i in range(1599):
    y[i]=np.where(y[i]<=5,-1,1)#将Quality大于5为优质，定为1，反之则为劣质，定为-1

#8:2划分训练集与测试集
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)

#PCA降维
pca=PCA(n_components=7)
pca.fit(X_train)
X_train_pca=pca.transform(X_train)
X_test_pca=pca.transform(X_test)

#归一化
X_scaler = MinMaxScaler()
X_train_scaled=X_scaler.fit_transform(X_train_pca)
X_test_scaled=X_scaler.transform(X_test_pca)

#网格搜索交叉验证来调整SVC函数中的超参数
params = {
    'gamma':[0.01, 0.1, 0.5, 1, 2, 10, 100],
    'C': [0.01, 0.1, 0.5, 1, 2, 10, 100]
    }
clf=svm.SVC()#选用SVM模型
grid=GridSearchCV(clf,params,cv=10,n_jobs=-1)
grid.fit(X_train_scaled,y_train)

#结果展示
y_pred=grid.predict(X_test_scaled)
print('最佳分类器: ',grid.best_estimator_)
print("混淆矩阵\n",confusion_matrix(y_test,y_pred))
print("分类报告\n",classification_report(y_test,y_pred))

#PR曲线
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
plt.figure()
```

```
plt.plot(recall, precision)
plt.title('Precision Recall Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.show()
```