华东师范大学计算机科学技术系实验报告

课程名称： 数据分析实践	**年级：** 2016 级	**实践作业成绩：**
指导教师： 兰曼	**姓名：** 英嘉豪	**作业提交日期：**
实验编号： 3	**学号：** 10175102247	**实践作业编号：** 3

1 实验名称

手写数字识别

2 实验目的

运用本学期的多种算法和策略进行手写体的数字识别。

实验要求

在前面数据分析的基础之上，对手写体数字图像进行预处理和构建识别系统。本次实验通过手写体数字图像实例来进行数字图像的实践分析和识别类别，也适用于其他字符的多种图像类型数据。此外，本实验比较了基于多种不同分类算法的识别系统的性能。

3 实验内容

3.1 环境准备

3.1.1 基本环境

- Windows 10
- Python 3.6.7

3.1.2　依赖

- pandas 0.25.3
- scikit-learn 0.21.3
- scipy 1.4.1
- numpy 1.17.1
- matplotlib 3.1.2

3.2　实验内容

3.2.1　模型选择以及工具函数

本次实验要求对手写体数字图像进行预处理和构建识别系统，基于不同分类算法构建训练模型，再使用模型去预测验证集数据，统计识别错误的样本个数并计算预测准确率和错误率。使用的分类算法为：

- 决策树（Decision Tree，DT）
- 支持向量机（Support Vector Machine，SVM）
- K-最近邻（k-Nearest Neighbor，kNN）
- 朴素贝叶斯（Naïve Bayes，NB）
- 随机森林（Random Forest，RF）

```
# 加载数据
def loadData(folder_name):
    hwLabels = []
    trainingFieldList = listdir(folder_name)
    m = len(trainingFieldList)
    trainingMat = np.zeros((m,1024))
    for i in range(m):
```

```
        fileName= trainingFieldList[i]
        fileStr = fileName.split('.')[0]
        classNumStr = int(fileStr.split('_')[0])
        hwLabels.append(classNumStr)
        trainingMat[i,:]=img2vec(folder_name+fileName)
    return trainingMat,hwLabels
```

函数接受一个字符串作为输入，表示需要打开的数据文件文件名。函数先打开图片文件夹。将图片的二分灰度图片转化为1*1024的特征向量，返回了特征矩阵和标签列表。

```
# 计算误差
def cal_accuracy(classifierResult,goldLabels):
    mTest = len(classifierResult)
    errorCount = 0.0 # 统计识别错误的样本个数
    for i in range(mTest):
        if classifierResult[i] != goldLabels[i]:
            errorCount += 1.0
    print("\t 测试样本个数为:  %d " % mTest)
    print("\t 预测错误个数为:  %d " % errorCount)
    print("\t 预测错误率为:  %2.2f%% " % (errorCount/float(mTest)*100))
    print("\t 预测准确率为:  %2.2f%%" % ((1-errorCount/float(mTest))*100))
    return (1-errorCount/float(mTest))
```

我们以统计正确的标签为基准，计算模型的误差和正确率。

3.2.2 模型调参工具

在选择不同模型中存在很多可选的参数，有些参数的存在是对模型效率和准确率有着至关重要的影响，通常算法不够好，需要调试参数时必不可少。比如 SVM 的惩罚因子 C，核函数 kernel，gamma 参数等，对于不同的数据使用不同的参数，结果效果可能差 1-5 个点，而盲目的循环列举参数也会导致模型结果保存结果过多，效果不好。Sklrean提供了网格搜索函数GridSearchCV它存在的意义就是自动调参，只要把参数输进去，就能给出最优化的结果和参数。

```
class sklearn.model_selection.GridSearchCV(estimator, param_grid, scoring=None,
fit_params=None, n_jobs=1, iid=True, refit=True, cv=None, verbose=0,
pre_dispatch='2*n_jobs', error_score='raise',
return_train_score='warn')
```

其中参数

- estimator指的是选择使用的分类器
- param_grid需要最优化的参数的取值，值为字典或者列表
- cv 交叉验证参数，默认 None
- scoring 模型评价标准，可以选择预定义的也可以选择自己定义

3.2.3 learningcurve 学习曲线

在我采用的模型中有不少树类模型，这是就离不开learning_curve学习曲线来对模型的拟合程度进行判断。

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.datasets import load_digits
from sklearn.model_selection import learning_curve
from sklearn.model_selection import ShuffleSplit


def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
```

```
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - Qtest_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")
    plt.legend(loc="best")
    return plt
```

4 实验结果及分析

4.1 决策树（Decision Tree，DT）

sklearn.DecisionTreeClassifier()关键参数如下：

- Criteriton: gini 或者 entropy，前者为基尼系数，后者为信息熵
- Max_depth: 限制决策树中的最大深度，深度越大越容易过拟合
- Max_features: 特征值的数量
- Max_samples_split 节点的最小样本数量，当样本数量小于此值时，节点将不会划分

```python
# 设置参数矩阵：
tuned_parameters =
                  [
                  {'criterion': ['entropy','gini'],
                  'max_depth': [5,10,15,20,30],
                  'min_samples_split': [5,10,15]}
                  ]
X_train,y_train = X, y
score = 'precision'
print("# Tuning hyper-parameters for %s" % score)
print()
clf = GridSearchCV(DecisionTreeClassifier(), tuned_parameters, cv=10,
                   scoring='%s_macro' % score)

clf.fit(X_train, y_train)
```

由于类标中只有 0 到 9 的标称数据，因而评分函数我选择为score = 'precision'

```
Detailed classification report:

             precision    recall  f1-score   support

          0       1.00      0.85      0.92        39
          1       0.70      0.88      0.78        48
          2       0.78      0.69      0.73        45
          3       0.89      0.96      0.92        49
          4       0.78      0.81      0.79        36
          5       0.82      0.84      0.83        37
          6       0.91      0.91      0.91        45
          7       0.91      0.84      0.88        51
          8       0.63      0.63      0.63        30
          9       0.88      0.83      0.86        54

avg / total       0.84      0.83      0.83       434


使用测试集验证平均准确率：
DecisionTreeClassifier {'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 5}
        测试样本个数为：  434
        预测错误个数为：  73
        预测错误率为：  16.82%
        预测准确率为：  83.18%


0.8317972350230415
```

图 1: 决策树调参结果

采用十折交叉验证的结果，并采用准确率作为评分依据，可得到当 criterion 为信息熵，最大深度为 10，节点最小样本量为 5 时，准确率最高，因此采用这种参数组合进行验证集验证。DecisionTreeClassifier()算法平均预测准确率为 0.84，在预测“0”时，准确率最高为 100%，预测“8”时，准确率最低为 63%。

4.2 支持向量机（Support Vector Machine，SVM）

支持向量机是一类按监督学习方式对数据进行二院分类的广义线性分类器，其决策边界是对学习样本求解最大边距超平面。SVC() 关键参数如下：

- gamma: 默认为 auto 时，其值为样本特征值的倒数
- C: 惩罚参数，也是 SVM 主要的调整参数，C>0 时，称为软间隔 SVM，C=0 时，称为硬间隔 SVM
- Kernel: 核函数，包括多项式核（poly），径向基数函数核（rbf）也称高斯核，Sigmoid 核（sigmoid）

```
from sklearn.svm import SVC
X_train,y_train = X, y
tuned_parameters = {'kernel': ['rbf','poly','linear','sigmoid'],
                    'gamma': ['auto',1e-3, 1e-4],
                    'C': [1, 10, 100, 1000]
                   }
score = 'precision'

clf = GridSearchCV(SVC(), tuned_parameters, cv=10,scoring='%s_macro' % score)
clf.fit(X_train, y_train)
print("SCV",clf.best_params_)
cal_accuracy(y_pred,y_true)
```

```
                precision    recall  f1-score   support

             0       1.00      0.97      0.99        39
             1       0.94      0.92      0.93        48
             2       1.00      1.00      1.00        45
             3       0.96      1.00      0.98        49
             4       0.92      1.00      0.96        36
             5       0.92      0.95      0.93        37
             6       1.00      0.96      0.98        45
             7       0.98      1.00      0.99        51
             8       0.96      0.87      0.91        30
             9       0.98      0.98      0.98        54

    accuracy                           0.97       434
   macro avg       0.97      0.96      0.96       434
weighted avg       0.97      0.97      0.97       434


使用测试集验证平均准确率：
SCV {'C': 100, 'gamma': 0.001, 'kernel': 'poly'}
        测试样本个数为：  434
        预测错误个数为：  14
        预测错误率为：  3.23%
        预测准确率为：  96.77%


0.967741935483871
```

图 2: SVM 调参结果

采用十折交叉验证的结果，并采用准确率作为评分依据，可得到当 C 为 100，gamma 为 0.001，核函数为 poly 时，准确率最高，因此采用这种参数组合进行验证集的验证。SVC() 算法平均预测准确率为 96.7%，，预测 “4”、“5” 时，准确率最低且为 92%。

4.3 K-最近邻（k-Nearest Neighbor，kNN）

其基本做法是给定测试实例，基于某种距离度量找出训练集中与其最靠近的 k 个实例点，然后基于这 k 个最近邻的信息来进行预测

```
X_train,y_train = X, y

tuned_parameters = [{'algorithm': ['kd_tree','ball_tree','brute'],
                     'n_neighbors': [3,4,5,6,7,10,13,15,20]}
                    ]

score = 'precision'
```

```
print("# Tuning hyper-parameters for %s" % score)
print()

clf = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=10,
                   scoring='%s_macro' % score)
clf.fit(X_train, y_train)
```

采用十折交叉验证的结果，并采用准确率作为评分依据，可得到当 n_neighbors 为 4，algorithm 为 ball_tree 时，准确率最高。

```
Detailed classification report:

             precision    recall  f1-score   support

          0       0.97      0.97      0.97        39
          1       0.89      0.98      0.93        48
          2       1.00      0.98      0.99        45
          3       0.92      1.00      0.96        49
          4       0.95      0.97      0.96        36
          5       0.97      0.95      0.96        37
          6       1.00      0.96      0.98        45
          7       0.96      1.00      0.98        51
          8       0.96      0.73      0.83        30
          9       0.94      0.93      0.93        54

avg / total       0.96      0.95      0.95       434


使用测试集验证平均准确率：
knn {'algorithm': 'ball_tree', 'n_neighbors': 4}
        测试样本个数为:  434
        预测错误个数为:  20
        预测错误率为:  4.61%
        预测准确率为:  95.39%


0.9539170506912442
```

图 3: KNN 调参结果

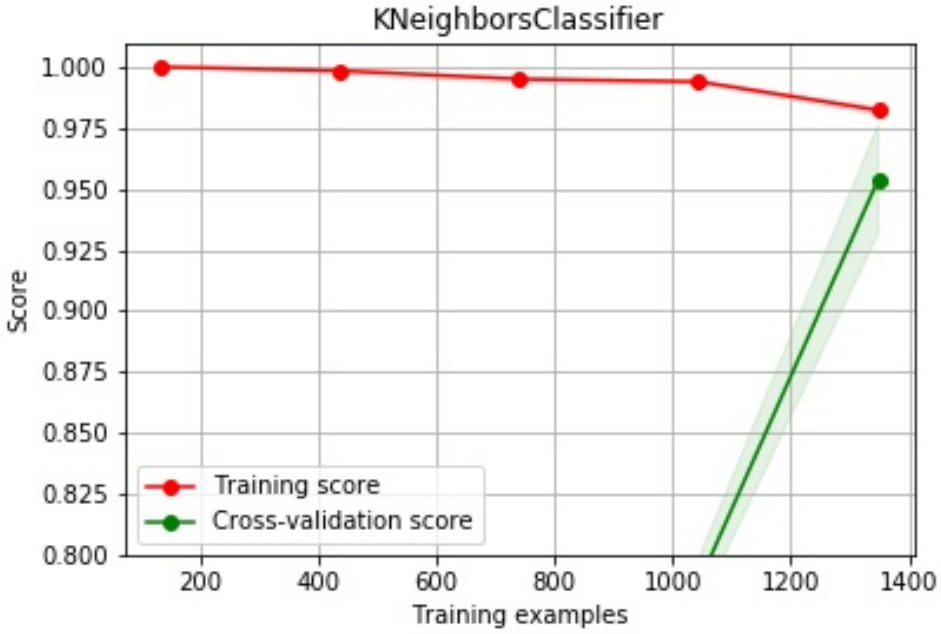

图 4

可以看出在现有数据的基础上KNN模型略有些过拟合，在图中可能看不出收敛形式，我重新利用训练数据将数据量扩大后结果如下

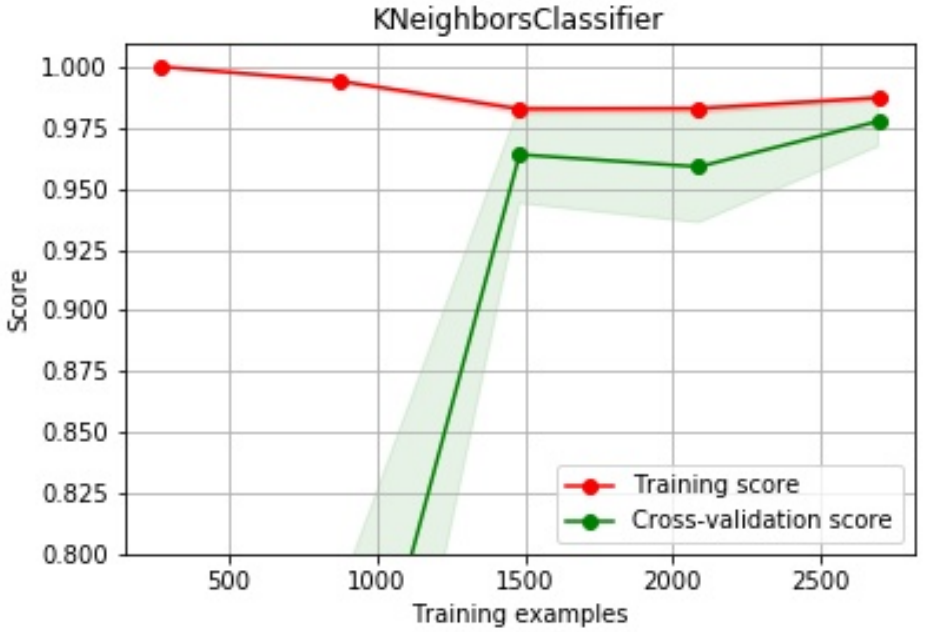

图 5

是略有些过拟合，不过总体来说拟合效果还是可以的。

4.4 随机森林（Random Forest，RF）

随机森林是一种集成学习方法，由许多棵决策树构成的森林共同来进行预测。随机主要体现在以下两个方面：每棵树的训练集是随机且有放回抽样产生的；训练样本的特征是随机选取的。RandomForestClassifier() 关键参数如下：

- max_features：限制每棵树能使用的特征数
- n_estimators：决策树的个数，默认为 10
- min_samples_leaf：叶子节点所需最少的样本数，默认为 1

```
X_train,y_train = X, y


# Set the parameters by cross-validation
tuned_parameters = [{'n_estimators': [5,10,20,50,100,200,500],
                     'max_features': ["sqrt","log2",0.2],
                     'min_samples_leaf': [1,5,10,50,100,150,200]}
                    ]

score = 'precision'

print("# Tuning hyper-parameters for %s" % score)
print()

clf = GridSearchCV(RandomForestClassifier(), tuned_parameters, cv=5,
                    scoring='%s_macro' % score)
clf.fit(X_train, y_train)
```

Detailed classification report:

	precision	recall	f1-score	support
0	1.00	0.97	0.99	39
1	0.96	0.96	0.96	48
2	0.96	1.00	0.98	45
3	0.96	1.00	0.98	49
4	0.97	0.97	0.97	36
5	0.97	0.97	0.97	37
6	1.00	0.96	0.98	45
7	0.98	1.00	0.99	51
8	0.97	0.93	0.95	30
9	1.00	0.98	0.99	54
avg / total	0.98	0.98	0.98	434

图 6: 随机森林模型结果

采用十折交叉验证的结果，并采用准确率作为评分依据，可得到当 n_estimators 为 100，max_features 为 sqrt，min_samples_leaf 为 1 时，准确率最高，因此采用这种参数组合进行验证集的验证。RandomForestClassifier () 算法平均预测准确率最高且为 97.93%。其拟合程度曲线如下：

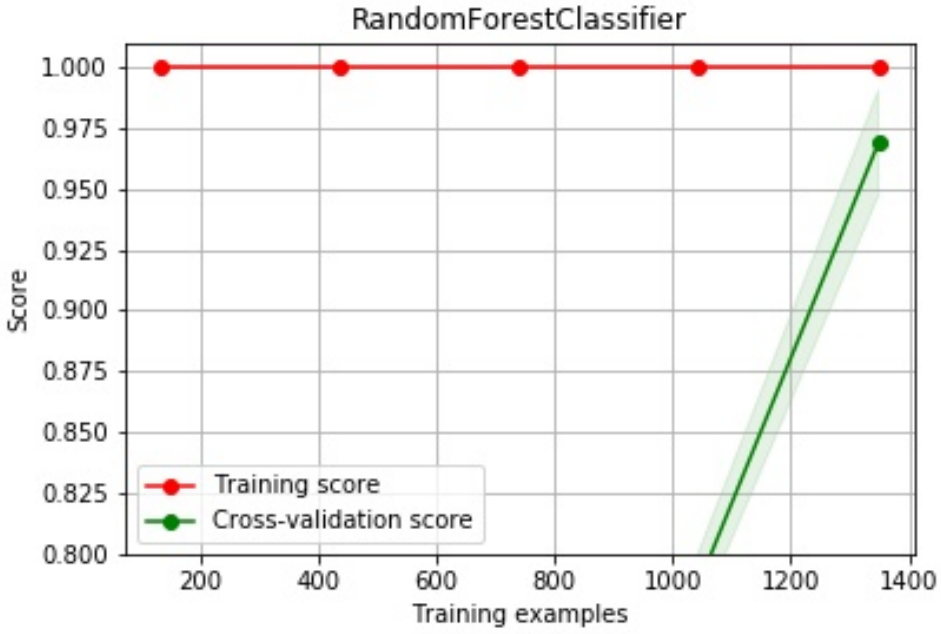

图 7: 随机森林模型结果

同样可以看出其模型也是略有过拟合现象。

4.5 朴素贝叶斯

scikit-learn 中，一共有 3 个朴素贝叶斯的分类算法类。分别是 GaussianNB，MultinomialNB 和 BernoulliNB。BernoulliNB () 算法平均预测准确率最高且为 93.32%，GaussianNB () 算法平均预测准确率最低且为 72.35%

5 问题讨论

- 本次实验我们使用的数据是二分的灰度点图，这样可能效果不如手动将手写图片做了预处理的灰图图像效果好。

- 在本次实验中，使用了十折的交叉验证，在一定程度上防止了过拟合问题。

- 实验中重复使用训练数据是否是一种合适的方法。

6 结论

- 当样本少数量但是样本特征非常多的时候，决策树很容易过拟合，一般来说，样本数比特征数多一些会比较容易建立健壮的模型
- 做了交叉验证可以降低过拟合的程度。