课后作业：模型评估与优化

作者：欧新宇（Xinyu OU）

本文档所展示的测试结果，均运行于：Intel Core i7-7700K CPU 4.2GHz

【作业提交】

将分类结果保存到文本文档进行提交(写上每一题的题号和题目，然后再贴答案)，同时提交源代码。

1. 测试结果命名为: ex11-结果-你的学号-你的姓名.txt
2. 源代码命名为：ex1101-你的学号-你的姓名.py

结果文件，要求每小题标注题号，两题之间要求空一行

- 要求在 "鸢尾花" 数据集上完成以下任务，要求如下：
    1. 要求`训练集`、`验证集`的分割比例为：70%：30%
    2. 使用`MLP`模型进行训练和测试，MLP优化函数为lbfgs，激活函数为relu
    3. 使用`StratifiedKFold`算法进行**分层5折交叉验证**
    4. 所有算法的`随机参数` random_state=62
    5. 结果保留4位小数
    6. 输出测试集结果的时候，可以使用基于网格搜索的三种输出中的任意一种
- MLP需要搜索的参数列表为：
    - alpha: [5e-4, 1e-4, 5e-3, 1e-3, 5e-2, 1e-2],
    - hidden_layer_sizes: [[10], [20], [50], [100], [150], [10, 10],[20, 20],[50, 50]]}

> **提示**：
>
> 1. 在训练集上训练MLP模型，并使用`交叉验证`和`网格搜索`获取**最优参数**
> 2. 将最优参数应用到MLP模型，并在训练验证集上进行训练
> 3. 输出训练验证集上的评分和测试集上的评分

1. 载入MNIST数据集

```
# TODO: 1. 导入必须库 以及 定义必要的函数
# 导入机器学习数据集处理工具
from sklearn import datasets
from sklearn.model_selection import train_test_split

# TODO: 2. 创建/导入数据
iris = datasets.load_iris()

# TODO: 3. 数据预处理，包括训练集、测试集划分，数据正则化，数据清洗等
X = iris.data
y = iris.target
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y,
train_size=0.7, random_state=16)
# X_train, X_val, y_train, y_val = train_test_split(X_trainval, y_trainval,
train_size=0.7)
```

2. 使用分层*K*折交叉验证

```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier
import time

start = time.time()

SKF = StratifiedKFold(n_splits=5, random_state=16, shuffle=False)
mlp = MLPClassifier(solver='lbfgs', activation='relu', random_state=16)

params = {'alpha':[5e-4, 1e-4, 5e-3, 1e-3, 5e-2, 1e-2],
          'hidden_layer_sizes':[[10], [20], [50], [100], [150], [10, 10], [20, 20],[50, 50]]}

# params = {'alpha':[1e-6, 5e-5, 1e-5, 5e-4, 1e-4, 5e-3, 1e-3, 5e-2, 1e-2],
#           'hidden_layer_sizes':[[10], [20], [50], [100], [150], [200], [10, 10],[20, 20],[50, 50],[100, 100],[150, 150],[200, 200]]}

grid_search = GridSearchCV(mlp, params, cv=SKF, iid=False)
grid_search.fit(X_trainval, y_trainval)
print("执行时间:{:.2f}s".format(time.time()-start))
print('最优参数：{}'.format(grid_search.best_params_))
print('最佳得分（验证集）：{:.4f}'.format(grid_search.best_score_))
print('最优模型：{}'.format(grid_search.best_estimator_))
```

```
执行时间:9.97s
最优参数：{'alpha': 0.0005, 'hidden_layer_sizes': [150]}
最佳得分（验证集）：0.9723
最优模型：MLPClassifier(activation='relu', alpha=0.0005, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=[150], learning_rate='constant',
              learning_rate_init=0.001, max_iter=200, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=16, shuffle=True, solver='lbfgs', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

3. 保持参数不变，在训练验证集上重新训练，并输出结果

- 直接使用grid_search.score()进行输出

```
score_trainval = grid_search.score(X_trainval, y_trainval)
score_test = grid_search.score(X_test, y_test)

print('基于GridSearch方法的输出，训练集得分：{:.4f}，测试集得分：{:.4f}'.format(score_trainval, score_test))
```

```
基于GridSearch方法的输出，训练集得分：0.9714，测试集得分：0.9333
```

- 使用最优参数进行输出

```
best_params = grid_search.best_params_
mlp = MLPClassifier(solver='lbfgs', activation='relu', random_state=16,
**best_params)
mlp.fit(X_trainval, y_trainval)

score_trainval = mlp.score(X_trainval, y_trainval)
score_test = mlp.score(X_test, y_test)

print('基于最优参数的输出，训练集得分: {:.4f}, 测试集得分:
{:.4f}'.format(score_trainval, score_test))
```

```
基于最优参数的输出，训练集得分: 0.9714, 测试集得分: 0.9333
```

- 使用最优模型进行输出

```
model = grid_search.best_estimator_
score_trainval = model.score(X_trainval, y_trainval)
score_test = model.score(X_test, y_test)

print('基于最优模型的输出，训练集得分: {:.4f}, 测试集得分:
{:.4f}'.format(score_trainval, score_test))
```

```
基于最优模型的输出，训练集得分: 0.9714, 测试集得分: 0.9333
```