课后作业：神经网络(Neural Networks)

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: ex08-结果-你的学号-你的姓名.txt
2. 输出图片命名为: ex08-性能对比图-你的学号-你的姓名.png (.jpg)
3. 源代码命名为:

- ex08-01Baseline-你的学号-你的姓名.py
- ex08-02Single-你的学号-你的姓名.py
- ex08-03Multi-你的学号-你的姓名.py
- ex08-04Logistic-你的学号-你的姓名.py
- ex08-05Tanh-你的学号-你的姓名.py
- ex08-06ReLU-你的学号-你的姓名.py
- ex08-07All-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. 要求训练集和测试集的分割比例为: 1:9
2. 先构建一个基于默认参数的Baseline模型（ex08-01Baseline），分别在Baseline的基础上设置单隐层模型(增加/减少神经元)（ex08-02Single）、多隐层模型（ex08-03Multi），并输出评分结果。【注意：该题可能需要多次运行，并选择一个出一个较好的结果，提供给第3题使用。】
3. 选择一个较好的模型，在此基础上测试三种不同的激活函数 {'logistic', 'tanh', 'relu'}，并输出评分结果。分别命名为：ex08-04Logistic,ex08-05Tanh,ex08-06ReLU。
4. 对以上六个模型 {'Baseline', 'Single', 'Multi', 'Logistic', 'Tanh', 'ReLU'}, 绘制测试集性能曲线图.（ex08-07All, ex08-性能对比图）
5. 所有模型性能评分，都写入文件（ex08-结果），格式为：

```
01Baseline: 训练集准确率: 1.0000, 测试集准确率: 0.9704.
02Single: 训练集准确率: 1.0000, 测试集准确率: 0.9630.
...
06ReLU: 训练集准确率: 1.0000, 测试集准确率: 0.9778.
```

数据集载入方法

```
from sklearn import datasets
iris = datasets.load_iris()

统一设置：random_state=10

```

Baseline

```python
# TODO: 1. 导入必须库 以及 定义必要的函数
# 导入数据集工具包
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 导入MLP神经网络包
from sklearn.neural_network import MLPClassifier

# 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_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.1, random_state=10)

# TODO: 4. 构建模型，并进行模型训练（或称为拟合数据）
mlp_Baseline = MLPClassifier(solver='lbfgs', random_state=10)
mlp_Baseline.fit(X_train, y_train)

# TODO: 5. 输出预测结果
score_train = mlp_Baseline.score(X_train, y_train)
score_test = mlp_Baseline.score(X_test, y_test)
print("训练集准确率：{0:.4f}，测试集准确率：{1:.4f}.".format(score_train, score_test))

```

```
训练集准确率：1.0000，测试集准确率：0.9704.
```

Single

```python
# TODO: 1. 导入必须库 以及 定义必要的函数
# 导入数据集工具包
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 导入MLP神经网络包
from sklearn.neural_network import MLPClassifier

# 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_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.1, random_state=10)

# TODO: 4. 构建模型，并进行模型训练（或称为拟合数据）
mlp_Single = MLPClassifier(solver='lbfgs', random_state=10,
                             hidden_layer_sizes=[10])
mlp_Single.fit(X_train, y_train)

# TODO: 5. 输出预测结果
score_train = mlp_Single.score(X_train, y_train)
score_test = mlp_Single.score(X_test, y_test)
```

```
print("训练集准确率：{0:.4f}，测试集准确率：{1:.4f}.".format(score_train, score_test))

```

```
训练集准确率：1.0000，测试集准确率：0.9630.
```

Multi

```
# TODO: 1. 导入必须库 以及 定义必要的函数
# 导入数据集工具包
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 导入MLP神经网络包
from sklearn.neural_network import MLPClassifier

# 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_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.1, random_state=10)

# TODO: 4. 构建模型，并进行模型训练（或称为拟合数据）
mlp_Multi = MLPClassifier(solver='lbfgs', random_state=10,
                              hidden_layer_sizes=[32, 128, 128] )
mlp_Multi.fit(X_train, y_train)

# TODO: 5. 输出预测结果
score_train = mlp_Multi.score(X_train, y_train)
score_test = mlp_Multi.score(X_test, y_test)
print("训练集准确率：{0:.4f}，测试集准确率：{1:.4f}.".format(score_train, score_test))

```

```
训练集准确率：1.0000，测试集准确率：0.9778.
```

Logistic

```
# TODO: 1. 导入必须库 以及 定义必要的函数
# 导入数据集工具包
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 导入MLP神经网络包
from sklearn.neural_network import MLPClassifier

# 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_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.1, random_state=10)

# TODO: 4. 构建模型，并进行模型训练（或称为拟合数据）
mlp_Logistic = MLPClassifier(solver='lbfgs', random_state=10, activation='logistic',
                             hidden_layer_sizes=[32, 128, 128] )
mlp_Logistic.fit(X_train, y_train)

# TODO: 5. 输出预测结果
score_train = mlp_Logistic.score(X_train, y_train)
score_test = mlp_Logistic.score(X_test, y_test)
print("训练集准确率: {0:.4f}, 测试集准确率: {1:.4f}.".format(score_train, score_test))

```

```
训练集准确率: 1.0000, 测试集准确率: 0.9556.
```

Tanh

```
# TODO: 1. 导入必须库 以及 定义必要的函数
# 导入数据集工具包
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 导入MLP神经网络包
from sklearn.neural_network import MLPClassifier

# 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_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.1, random_state=10)

# TODO: 4. 构建模型，并进行模型训练（或称为拟合数据）
mlp_Tanh = MLPClassifier(solver='lbfgs', random_state=10, activation='tanh',
                             hidden_layer_sizes=[32, 128, 128] )
mlp_Tanh.fit(X_train, y_train)

# TODO: 5. 输出预测结果
score_train = mlp_Tanh.score(X_train, y_train)
score_test = mlp_Tanh.score(X_test, y_test)
print("训练集准确率: {0:.4f}, 测试集准确率: {1:.4f}.".format(score_train, score_test))

```

```
训练集准确率: 1.0000, 测试集准确率: 0.9556.
```

ReLU

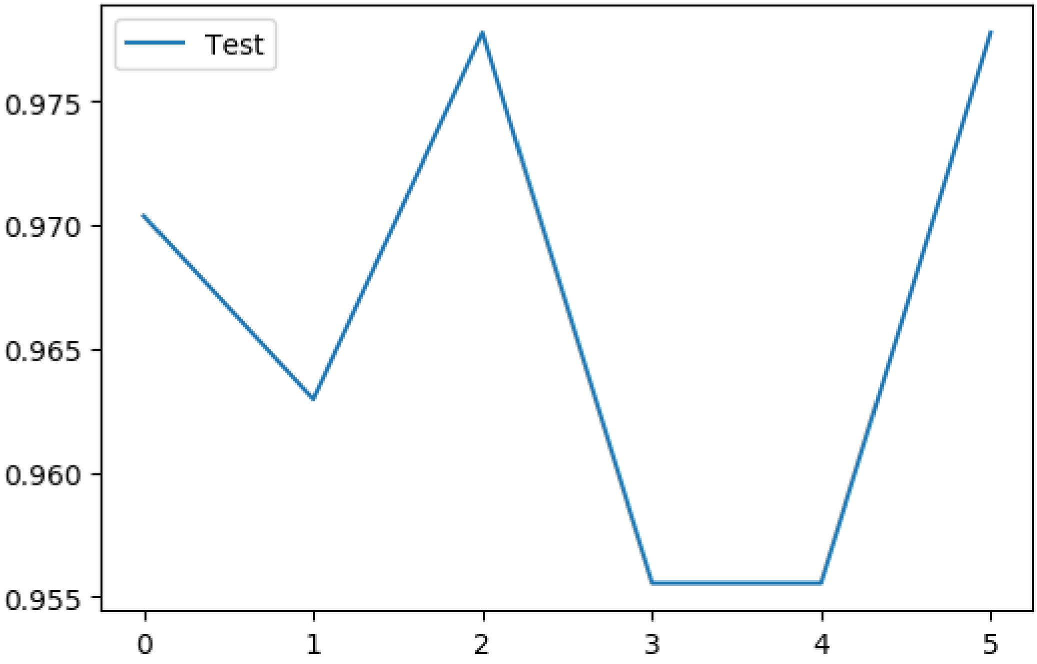
```
# TODO: 1. 导入必须库 以及 定义必要的函数
# 导入数据集工具包
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 导入MLP神经网络包
from sklearn.neural_network import MLPClassifier

# 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_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.1, random_state=10)

# TODO: 4. 构建模型，并进行模型训练（或称为拟合数据）
mlp_ReLU = MLPClassifier(solver='lbfgs', random_state=10, activation='relu',
                             hidden_layer_sizes=[32, 128, 128] )
mlp_ReLU.fit(X_train, y_train)

# TODO: 5. 输出预测结果
score_train = mlp_ReLU.score(X_train, y_train)
score_test = mlp_ReLU.score(X_test, y_test)
print("训练集准确率：{0:.4f}，测试集准确率：{1:.4f}.".format(score_train, score_test))

```

```
训练集准确率：1.0000，测试集准确率：0.9778.
```

所有方法对比

```
# TODO: 1. 导入必须库 以及 定义必要的函数
import numpy as np
import matplotlib.pyplot as plt
# 导入数据集工具包
from sklearn import datasets
from sklearn.model_selection import train_test_split
# 导入MLP神经网络包
from sklearn.neural_network import MLPClassifier

# 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_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.1, random_state=10)

xticks = ['Baseline', 'Single', 'Multi', 'Logistic', 'Tanh', 'ReLU']
num = len(xticks)
scores_test = np.zeros([1, num])

# TODO: 4. 构建模型，并进行模型训练（或称为拟合数据）
mlp_Baseline = MLPClassifier(solver='lbfgs', random_state=10)
```

```python
mlp_Baseline.fit(X_train, y_train)
scores_test[0, 0] = mlp_Baseline.score(X_test, y_test)

mlp_Single = MLPClassifier(solver='lbfgs', random_state=10, hidden_layer_sizes=[10])
mlp_Single.fit(X_train, y_train)
scores_test[0, 1] = mlp_Single.score(X_test, y_test)

mlp_Multi = MLPClassifier(solver='lbfgs', random_state=10, hidden_layer_sizes=[32, 128, 128] )
mlp_Multi.fit(X_train, y_train)
scores_test[0, 2] = mlp_Multi.score(X_test, y_test)


mlp_Logistic = MLPClassifier(solver='lbfgs', random_state=10, activation='logistic', hidden_layer_sizes=[32, 128, 128] )
mlp_Logistic.fit(X_train, y_train)
scores_test[0, 3] = mlp_Logistic.score(X_test, y_test)


mlp_Tanh = MLPClassifier(solver='lbfgs', random_state=10, activation='tanh', hidden_layer_sizes=[32, 128, 128] )
mlp_Tanh.fit(X_train, y_train)
scores_test[0, 4] = mlp_Tanh.score(X_test, y_test)


mlp_ReLU = MLPClassifier(solver='lbfgs', random_state=10, activation='relu', hidden_layer_sizes=[32, 128, 128] )
mlp_ReLU.fit(X_train, y_train)
scores_test[0, 5] = mlp_ReLU.score(X_test, y_test)

plt.figure(dpi=100)
plt.plot(scores_test[0, :], label='Test')
plt.legend(loc='best')
plt.savefig('results/Ch08HW01NN.png', dpi=150)
plt.show()
```


Test