### **1.前言**

sklearn神经网络，进行多分类，数字识别。

### **2.python代码**

（1）数据集用的sklearn自带，数字0~9分类   
（2）采用MLPClassifier   
（3）执行代码如下multi\_class\_nn.py:

from sklearn.neural\_network import MLPClassifier

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# 测试集，画图对预测值和实际值进行比较

def test\_validate(x\_test, y\_test, y\_predict, classifier):

x = range(len(y\_test))

plt.plot(x, y\_test, "ro", markersize=5, zorder=3, label=u"true\_v")

plt.plot(x, y\_predict, "go", markersize=8, zorder=2, label=u"predict\_v,$R$=%.3f" % classifier.score(x\_test, y\_test))

plt.legend(loc="upper left")

plt.xlabel("number")

plt.ylabel("true?")

plt.show()

# 神经网络数字分类

def multi\_class\_nn():

digits = datasets.load\_digits()

x = digits['data']

y = digits['target']

# 对数据的训练集进行标准化

ss = StandardScaler()

x\_regular = ss.fit\_transform(x)

# 划分训练集与测试集

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_regular, y, test\_size=0.1)

clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden\_layer\_sizes=(5,), random\_state=1)

clf.fit(x\_train, y\_train)

# 模型效果获取

r = clf.score(x\_train, y\_train)

print("R值(准确率):", r)

# 预测

y\_predict = clf.predict(x\_test) # 预测

print(y\_predict)

print(y\_test)

# 绘制测试集结果验证

test\_validate(x\_test=x\_test, y\_test=y\_test, y\_predict=y\_predict, classifier=clf)

multi\_class\_nn()