Ex5：手写字体的检测

学号：17343124
姓名：伍斌
完成时间：2020.12.30

任务 1：

针对 MINIST 数据集(手写体数字)完成如下任务：

1. 用 Adaboost 或者 SVM 训练一个手写体数字的分类器(用一半数据训练，一半数据测试)；
2. 针对 SVM/Adaboost 给出不同训练参数的训练性能和错误率等；
3. 针对测试数据分析为什么这些数据分类错误以及改进思路；
4. 用 A4 纸写若干数字，分别采用上面训练好的分类器进行识别测试，分析测试性能和改进思路.
5. 完整的实验报告.

参考资料：

MINIST 数据集: https://www.cnblogs.com/xianhan/p/9145966.html

编程语言：

PYTHON 或者 C++。

图像操作相关 C++库：CIMG

任务 2：

每个同学按照例图 1，用干净 A4 纸写 10 张，并拍照作为 Final Project 的测试用图，用 Excel 表格标注各行的信息。

数据相关要求:

1．光线充足，且 A4 纸拍照均匀；

2．书写规范，不要连写；

3．学号和身份证号码可以不对(但必须位数相同，例如身份号码必须 18 位，学号 8 位，手机号 11 位)，且书写三次都要相同；

4．严格按照例子表格进行标注；

5．数据完整性和规范性占本次成绩 50%；

6．上交作业要上传所有拍照数据和标注数据；

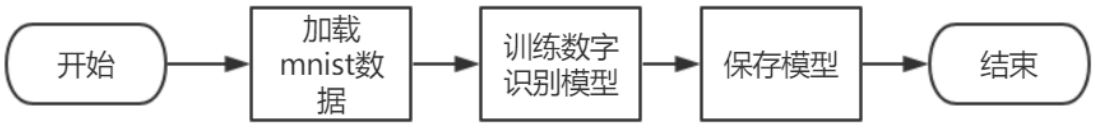
实现过程：

【任务一】

实验环境：

Windows 10 64 位系统、Visual Studio Code、Anaconda（已配置好 trensorflow）

主要步骤：



具体流程：

1. 导入 mnist 数据集：

```
import tensorflow.examples.tutorials.mnist.input_data as input_data
data_dir = '../MNIST_data/'
mnist = input_data.read_data_sets(data_dir,one_hot=False)
batch_size = 30000
test_x = mnist.test.images[:5000]
test_y = mnist.test.labels[:5000]
```

一共 60000 个示例的训练集以及 10000 个示例的测试集，取 30000 用于训练，5000 用于测试训练出的模型。

2. 调用 Adaboost 分类器进行训练：

```
batch_x,batch_y = mnist.train.next_batch(batch_size)
clf_rf = AdaBoostClassifier(n_estimators = 60)
clf_rf.fit(batch_x,batch_y)
```

3. 评估预测的效果：

```
y_pred_rf = clf_rf.predict(test_x)
acc_rf = accuracy_score(test_y,y_pred_rf)
print("%s n_estimators = 60, accuracy:%f" % (datetime.now(), acc_rf))
```

4. 选取较好的参数(弱分类器数量)：

重复 2、3 步骤，通过调节弱分类器数量（修改 n_estimators 的值）来获得一个训练效果比较不错的数量参数:

```
(C:\Users\42405\AppData\Local\conda\conda\envs\tensorflow) C:\Users\42405\Desktop\work2\adaboost>python ab_train.py
Extracting ../MNIST_data/train-images-idx3-ubyte.gz
Extracting ../MNIST_data/train-labels-idx1-ubyte.gz
Extracting ../MNIST_data/t10k-images-idx3-ubyte.gz
Extracting ../MNIST_data/t10k-labels-idx1-ubyte.gz
2020-12-30 12:05:43.920813 n_estimators = 10, accuracy:0.553600
2020-12-30 12:05:55.163168 n_estimators = 20, accuracy:0.596800
2020-12-30 12:06:12.012011 n_estimators = 30, accuracy:0.654800
2020-12-30 12:06:34.478699 n_estimators = 40, accuracy:0.678000
2020-12-30 12:07:02.844648 n_estimators = 50, accuracy:0.684800
2020-12-30 12:07:36.581879 n_estimators = 60, accuracy:0.686400
2020-12-30 12:08:16.026884 n_estimators = 70, accuracy:0.693200
2020-12-30 12:09:01.319266 n_estimators = 80, accuracy:0.710800
2020-12-30 12:09:51.983785 n_estimators = 90, accuracy:0.698600
2020-12-30 12:10:48.577197 n_estimators = 100, accuracy:0.696200
2020-12-30 12:11:50.956515 n_estimators = 110, accuracy:0.699200
2020-12-30 12:12:58.655168 n_estimators = 120, accuracy:0.697000
Total time 440.44 s
```

由上图实验结果可得，参数为 50 之后，正确率基本稳定在 68% 以上浮动，峰值在参数为 80 处，故之后选取弱分类器参数为 80 进行训练。

分析：针对测试数据分析为什么这些数据分类错误以及改进思路：

（参考博客：https://www.cnblogs.com/kevincong/p/7840382.html ）

AdaBoost 算法

- 优点：泛化错误率低，易编码，可以应用在大部分分类器上，无参数调整。
- 缺点：对离群点敏感。
- 适用数据类型：数值型和标称型数据。

可见 AdaBoost 算法适合的数据集类型为数值型和标称型数据。

- 标称型：一般在有限的数据中取，而且只存在'是'和'否'两种不同的结果（一般用于分类）
- 数值型：可以在无限的数据中取，而且数值比较具体化，例如 4.02,6.23 这种值（一般用于回归分析）

而本次实验所用 MNIST 数据集显然**不属于**这两种类型，而且**数据集本身有错误样本。**

并且发现在参数上升至 80 后，测试错误率在达到了一个最小值之后又开始上升了（正确率下降）。这类现象称之为过拟合（overfitting ,也称过学习）。有文献声称，对于表现好的数据集，AdaBoost 的测试错误率就会达到一个稳定值，并不会随着分类器的增多而上升。或许在本例子中的数据集也称不上"表现好"。该数据集一开始有 30% 的缺失值，对于 Logistic 回归而言，这些缺失值的假设就是有效的，而对于决策树却可能并不合适。

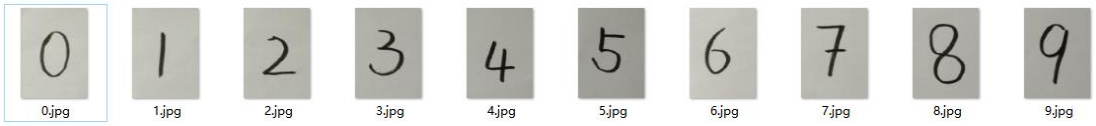
改进思路：之后换用多层感知器（MLP）训练数据集。

5. 在对自己的手写图片读入前先进行几步处理：

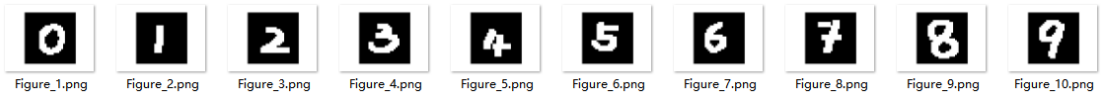
- 将图片转为二值图
- resize 为 mnist 训练集要求的(28*28)尺寸
- 将图像进行膨胀处理

```
    img = cv2.resize(img, (28, 28), interpolation=cv2.INTER_CUBIC)
    GrayImage = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    ret,thresh2=cv2.threshold(GrayImage,127,255,cv2.THRESH_BINARY_INV)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3, 3))
    img = cv2.dilate(thresh2,kernel)
```

导入的手写数字：



处理后的效果：



测试效果：

```
the digit is: 0
the result is: 0
the digit is: 1
the result is: 6
the digit is: 2
the result is: 2
the digit is: 3
the result is: 2
the digit is: 4
the result is: 6
the digit is: 5
the result is: 6
the digit is: 6
the result is: 6
the digit is: 7
the result is: 6
the digit is: 8
the result is: 6
the digit is: 9
the result is: 6
```

分析：test 集的准确率为 68%左右，自己手写数字的准确率为 30%，这是一个相当不期望的结果。

改进思路：考虑用一个自己搭建的 MLP 网络对模型进行测试。

【神经网络的多层感知器 MLPClassifier 进行手写字体的检测】

1. 训练数字识别模型

具体流程如下：首先加载 mnist 数据集，然后使用神经网络的多层感知器 MLPClassifier 进行训练，再把训练得到的模型保存下来，以用于后续数字识别工作。

（1）加载 mnist 数据集:

```
def load_mnist(path, kind='train'):
    # 加载mnist 数据集
    labels_path = os.path.join(path, '%s-labels.idx1-ubyte' % kind)
    images_path = os.path.join(path, '%s-images.idx3-ubyte' % kind)
    with open(labels_path, 'rb') as lbpath:
        magic, n = struct.unpack('>II',
                                 lbpath.read(8))
        labels = np.fromfile(lbpath, dtype=np.uint8)

    with open(images_path, 'rb') as imgpath:
        magic, num, rows, cols = struct.unpack('>IIII',
                                               imgpath.read(16))
        images = np.fromfile(imgpath, dtype=np.uint8).reshape(len(label
s), 784)
    return images, labels
```

（2）加载训练数据集

```
def train_model(mlp, save_dir):
    # 加载训练数据集
    [train_images, train_labels] = load_mnist('./mlp/data/')
    train_labels = train_labels.astype('int32')
    train_images = train_images.astype('int32')
    # 开始训练
    mlp.fit(train_images, train_labels)
    # 保存模型
    joblib.dump(mlp, save_dir)
```

```
    mlp = joblib.load(save_dir)
    # 加载测试数据集
    [test_images, test_labels] = load_mnist('./mlp/data/', kind = 't10k
')
    test_labels = test_labels.astype('int32')
    test_images = test_images.astype('int32')
    # 预测测试数据集
    y_pred = list(mlp.predict(test_images))
    print(accuracy_score(test_labels, y_pred)) # 测试集的得分

mlp = MLPClassifier(hidden_layer_sizes=(400, 200), activation='logistic
',
                    solver='sgd', learning_rate_init=0.001, max_iter=400, v
erbose = True)
save_dir = './mlp/model/classify.m'
if(os.path.isfile(save_dir)):
    # 加载模型
    mlp = joblib.load(save_dir)
else:
    train_model(mlp, save_dir)
```

2. 进行数字识别预测

具体流程如下：首先加载上一步得到的模型，再用存储的 model 对自己的手写字体进行测试。

```
Iteration 267, loss = 0.02713053
Iteration 268, loss = 0.02702990
Iteration 269, loss = 0.02692866
Iteration 270, loss = 0.02682837
Iteration 271, loss = 0.02673689
Iteration 272, loss = 0.02664042
Iteration 273, loss = 0.02654579
Training loss did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
0.9603
```

模型测试结果：采用两层隐藏层神经网络，第一个隐藏层有 400 个神经元，第二个隐藏层有 200 个神经元，最大迭代次数设为 400 。模型在测试集上的预测准确率为 0.9603。

自己手写数字测试结果：

```
the digit is 0:
recognize result:
0
the digit is 1:
recognize result:
1
the digit is 2:
recognize result:
2
```

```
the digit is 3:
recognize result:
3
the digit is 4:
recognize result:
4
the digit is 5:
recognize result:
5
the digit is 6:
recognize result:
6
the digit is 7:
recognize result:
7
the digit is 8:
recognize result:
8
the digit is 9:
recognize result:
9
```

最终效果 10 张图全部都预测准确。

【任务二】

见文件夹中任务二文件夹。