实验 4：基于 k 近邻的车牌号识别

陈泓宇 2022/7/22

实验任务

本次实验的任务是：尝试用 K-NN 的方法对分割好的常用车牌号字符图像进行自动识别和转化。

数据分析

在本次实验给出的数据中，共有 65 个不同分类的字符，辨别放入编号为 0~64 的文件夹中。

由于实验数据已经区分好 train 与 test 数据集，因此不再需要自行分割训练集与测试集

读取数据

```
def readFileContent(lables,flag):
    X = []
    Y = []
    for lableName in lables:#取出每一个路径进行读取
        filename = lableName
        if(flag == 1):#判断是训练集文件夹还是测试集文件夹
            pict = readPict('./data/test/' + filename)
        else:
            pict = readPict('./data/train/' + filename)
        X.append(pict)
        Y.append(lables[filename])
    return X,Y
```

其中 X 为图片数据，Y 为标签数据

图片数据的读取方式如下

```
def readPict(path):
    returanVec=np.zeros(399)
    img = Image.open(path)
    pixel = np.array(img)
    for i in range(19):
        for j in range(19):
            returanVec[i*19+j]=pixel[i][j]
    return returanVec
```

其中需要把二维向量转换成一维，方便后续拟合

标签的读取方式如下：

```python
import os
def readFileIndex(filepath):
    lables = dict()
    filetype = '.jpg'
    for i in range(0,65):#因为有0-64个类别
        str1 = str(i)
        filepath1 = os.path.join(filepath,str1)#读取文件夹
        for root,dirs,files in os.walk(filepath1):
            for j in files:
                if filetype+' 'in j+' ':
                    str1 = str(i)+'/'+j
                    lables[str1] = i
    return lables
```
0.4s

构建 K-NN 模型

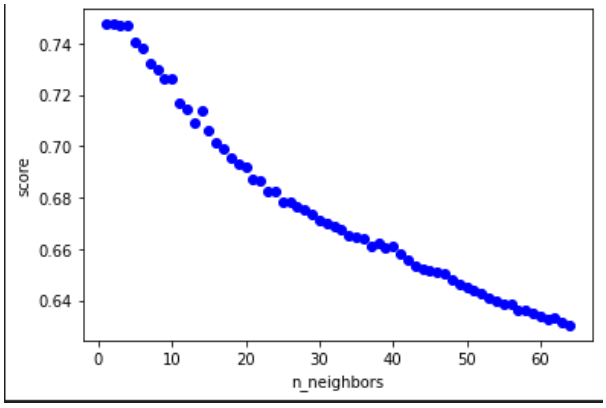
使用 sklearn 中的 KNeighborsClassifierg 构建模型，算法参数选用 kd_tree.

```python
neigh = KNeighborsClassifier(weights = 'distance',algorithm="kd_tree", n_neighbors = i)
```

实验结果

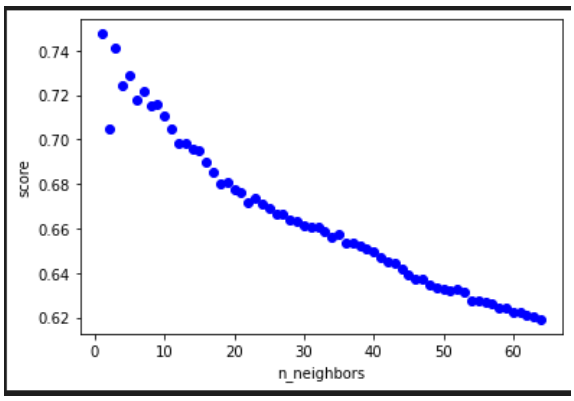
```python
for i in range(1,65):
    start = datetime.now()
    neigh = KNeighborsClassifier(weights = 'distance',algorithm="kd_tree", n_neighbors = i)
    neigh.fit(X_train,Y_train)
    end = datetime.now()
    print(end- start)
    Neighbor.append(i)
    Score.append(neigh.score(X_test,Y_test))
```

当选择权重“distance”时，n_neighbors 从 1 到 64 的结果如下:



可以看到随着近邻数增加，得分会降低

当我们将权重改为“uniform”的时候，得出如下结果:



可以看到除了当 n 的数值较小时，分数有较大的浮动外，结果大致与上次实验相近。

在实验过程中，尝试将 algorithm="kd_tree"去掉，让这个函数采用默认算法，发现运行速度大幅提高。经查询，默认算法会根据传递给 fit 方法的值来决定最合适的算法。不一定采用课程所讲 kd_treed 算法。