


```
import numpy as np
import operator
from os import listdir
from sklearn.neighbors import KNeighborsClassifier as kNN
```

"""

函数说明:将32x32的二进制图像转换为1x1024向量。

Parameters:

filename - 文件名

Returns:

returnVect - 返回的二进制图像的1x1024向量

"""

```
def img2vector(filename):
    #创建1x1024零向量
    returnVect = np.zeros((1, 1024))
    #打开文件
    fr = open(filename)
    #按行读取
    for i in range(32):
        #读一行数据
        lineStr = fr.readline()
        #每一行的前32个元素依次添加到returnVect中
        for j in range(32):
            returnVect[0, 32*i+j] = int(lineStr[j])
    #返回转换后的1x1024向量
    return returnVect
```

"""

函数说明:手写数字分类测试

Parameters:

无

Returns:

无

```
.....
```

```
def handwritingClassTest():  
    #测试集的Labels  
    hwLabels = []  
    #返回trainingDigits目录下的文件名  
    trainingFileList = listdir('/Users/carmack/examples/knn/trainingDigits')  
    #返回文件夹下文件的个数  
    m = len(trainingFileList)  
    #初始化训练的Mat矩阵,测试集  
    trainingMat = np.zeros((m, 1024))  
    #从文件名中解析出训练集的类别  
    for i in range(m):  
        #获得文件的名字  
        fileNameStr = trainingFileList[i]  
        #获得分类的数字  
        classNumber = int(fileNameStr.split('_')[0])  
        #将获得的类别添加到hwLabels中  
        hwLabels.append(classNumber)  
        #将每一个文件的1x1024数据存储在trainingMat矩阵中  
        trainingMat[i,:] = img2vector('/Users/carmack/examples/knn/  
trainingDigits/%s' % (fileNameStr))  
    #构建kNN分类器  
    neigh = KNN(n_neighbors = 3, algorithm = 'auto')  
    #拟合模型, trainingMat为训练矩阵, hwLabels为对应的标签  
    neigh.fit(trainingMat, hwLabels)  
    #返回testDigits目录下的文件列表  
    testFileList = listdir('/Users/carmack/examples/knn/testDigits')  
    #错误检测计数  
    errorCount = 0.0  
    #测试数据的数量  
    mTest = len(testFileList)  
    #从文件中解析出测试集的类别并进行分类测试  
    for i in range(mTest):  
        #获得文件的名字  
        fileNameStr = testFileList[i]  
        #获得分类的数字  
        classNumber = int(fileNameStr.split('_')[0])  
        #获得测试集的1x1024向量,用于训练  
        vectorUnderTest = img2vector('/Users/carmack/examples/knn/testDigits/
```

```
%s' % (fileNameStr))
    #获得预测结果
    # classifierResult = classify0(vectorUnderTest, trainingMat, hwLabels,
3)

    classifierResult = neigh.predict(vectorUnderTest)
    print("分类返回结果为%d\t真实结果为%d" % (classifierResult, classNumber))
    if(classifierResult != classNumber):
        errorCount += 1.0
    print("总共错了%d个数据\n错误率为%f%%" % (errorCount, errorCount/mTest * 100))
```