**华中科技大学计算机科学与技术学院**

**机器学习报告**



专 业： 计算机科学与技术

班 级： CS1706

学 号： U201714762

姓 名： 梁一飞

成 绩：

指导教师： 李玉华

**完成日期： 2020年 5 月 26日**

# 实验一

## 一、实验题目：KNN（K近邻）

## 二、实验要求

自己动手实现 KNN 算法。语言限定为 python。

数据集使用 MNIST 数据集。

数据集介绍：MNIST 手写数字数据集是具有 60,000 个示例的训练集和 10,000 个示例的测试集。 它是 NIST 提供的更大集合的子集。数字已经过尺寸标准化并以固定尺寸的图像为中心。对于想要在 真实数据上尝试学习技术和模式识别方法，同时在预处理和格式化方面花费最少的人来说，它是一个 很好的数据库。数据集链接：http://yann.lecun.com/exdb/mnist/ KNN

结果展示方式：

• 输入若干测试图片，输出对应每张图片 k 近邻的图片。

• 绘制 knn 算法的训练 misclassification rate 曲线，并做出分析

## 三、算法设计

KNN的原理就是当预测一个新的值x的时候，根据它距离最近的K个点是什么类别来判断x属于哪个类别

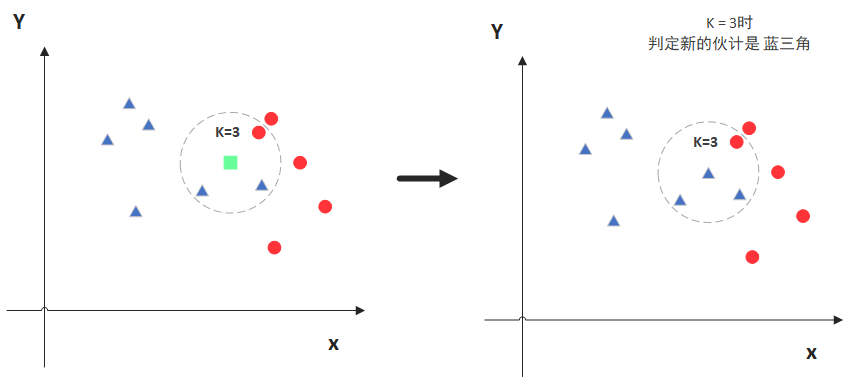


图1-1KNN算法图解（1）

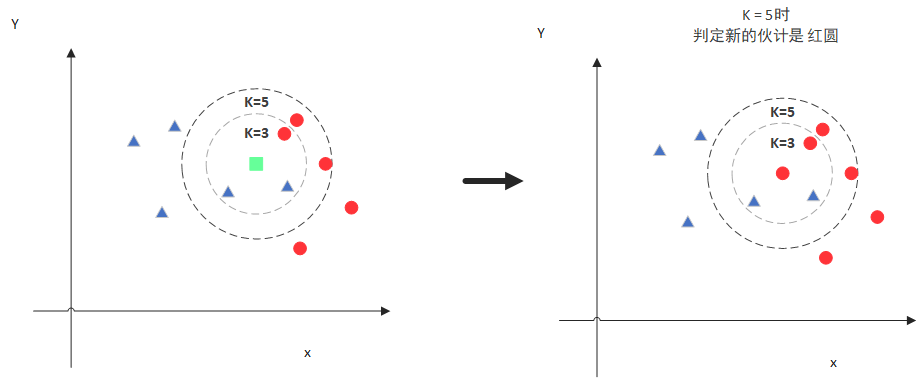
如上图1-1所示假设K=3。那么KNN算法就会找到与它距离最近的三个点（这里用圆圈把它圈起来了），看看哪种类别多一些，比如这个例子中是三角形多一些，新来的点就归类到三角了。以此类推当K=5时圆圈较多所以绿点判定为圆圈。

图1-2 KNN算法图解（2）

流程图如下图1-3所示

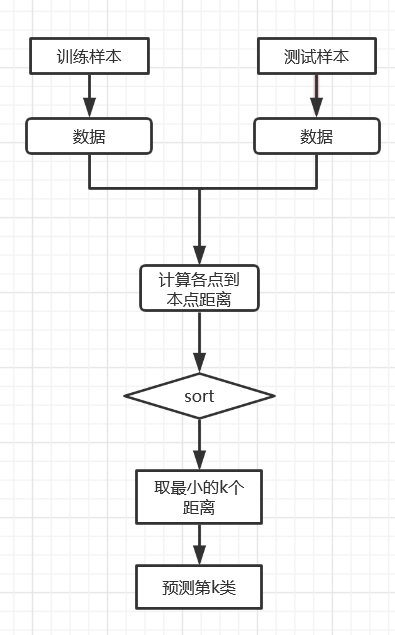


图1-3 KNN流程图

1. 计算给定测试对象与训练集中每个对象的欧几里得距离

欧几里得距离公式为：

按照公式计算相应的欧几里得距离

2）按照欧几里得距离大小进行排序。

3）按照K的数值，取K得距离最近的训练对象，作为测试对象的K邻近。

4）按照标签选出出现频率最高的k邻近作为标签。

5）将选取的最高频率出现的邻近标签，录入预测集中，与测试集的lable做比较，若正确，则count（正确个数）增加

6）比较测试总数与其中正确数量的比值，得到，算法的精确度accuracy。

## 四、实验环境与平台

1）实验环境条件：W

2）pycharm平台编译运行

## 五.代码实现

（1）欧几里得距离

def EuclidianDis(self):#欧几里得距离

return distances = np.sqrt(np.sum(np.square(self.Xtr - X[i,:]), axis = 1))

（2）最高频k邻近的获取

sortedDistances = distances.argsort()#进行排序

classCount = {}

for i in range(self.k): #选取最小k个

Label = labels[sortedDistanceIndicies[i]]

classCount[Label] = classCount.get(Label, 0) +

sortedCount = sorted(classCount.items()),#选取最高频

key=operator.itemgetter(1)

return sortedCount[0]

（3）数据的读取处理

def test\_img(self):#读取测试集图片

fr = open(self.test\_img\_file, "rb")

if fr is "":

return

self.skipheader(fr)

for i in range(0, self.TEST\_NUM):

self.test\_data\_set[i] = fromstring(fr.read(self.HEIGHT \* self.WIDTH), uint8)

def train\_img(self):#读取训练集图片

fr = open(self.train\_img\_file, "rb")

if fr is "":

return

self.skip\_header(fr)

for i in range(0, self.TRAIN\_NUM):

self.train\_data\_set[i] = string(fr.read(self.HEIGHT\*self.WIDTH), uint8)

def test\_label(self):#读取测试集标签

fr = open(self.test\_label\_file, "rb")

if fr is "":

return

self.skip\_header(fr, 8)

self.test\_label\_set = fromstring(fr.read(self.TEST\_NUM), uint8)

def train\_label(self):#读取训练集标签

fr = open(self.train\_label\_file, "rb")

if fr is "":

return

（4）KNN实现

for ind,k in enumerate(Kvalue):

knn = kNNHandWriting.KnnHandWriting(k)

knn.create\_data\_set("train-images.idx3-ubyte",

"train-labels.idx1-ubyte",

"t10k-images.idx3-ubyte",

"t10k-labels.idx1-ubyte")#文件读取

count = 0

total = 10000#选取测试集10000个

start = 0

for i in range(start,start+total):

result = knn.classfy(knn.imgReader.get\_test\_set()[i],

knn.imgReader.get\_train\_set(),

f knn.imgReader.get\_train\_label())#调用函数（2）功能函数classfy

if knn.imgReader.get\_test\_label()[i] == result[0] :

count += 1#检验正确，则加一

accurancy[t] =((float(count)/total)\*100.0)

print("for k=",Kvalue[t],"accurancy=",accurancy[t])

t=t+1#输出结果

(5)曲线的绘制与存储

plt.plot(Kvalue, accuracy)

plt.ylabel('Accuracy')

plt.xlabel('k values')

plt.savefig('KNN.png')

## 六、实验结果

（1）输出对应每张图片 k 近邻的图片如图1-4。

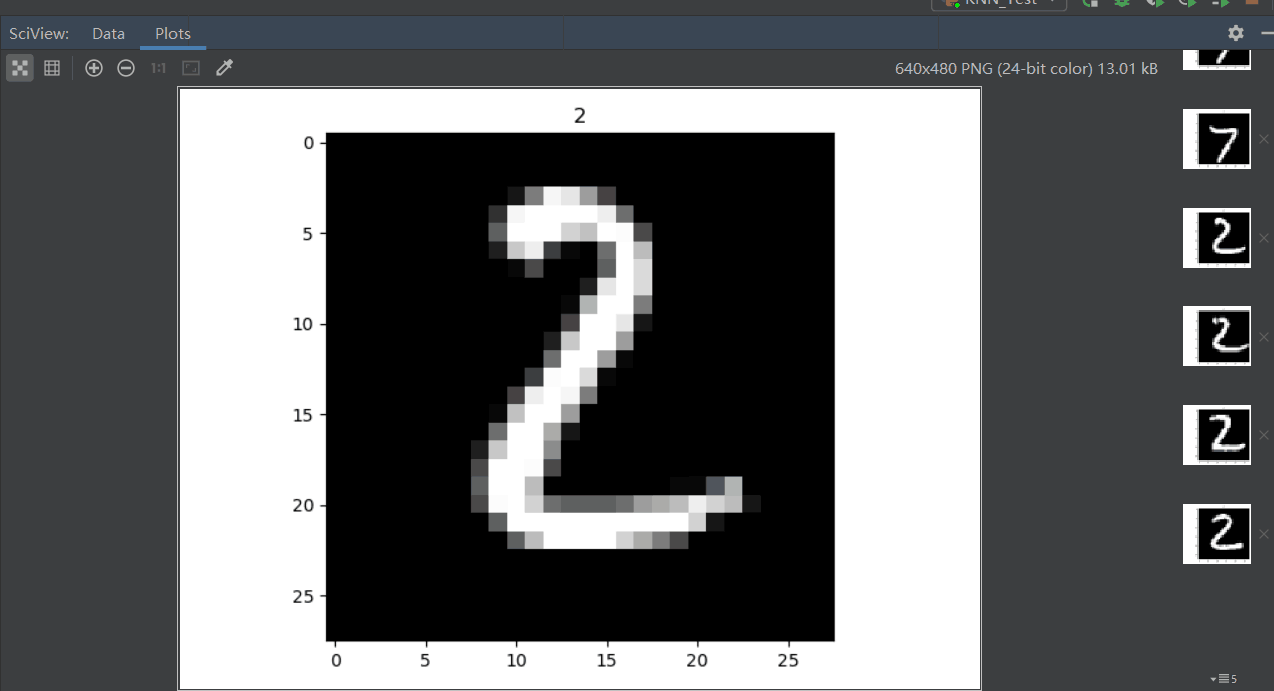


图1-4 k近邻部分图片

因为输出图片众多，选取了部分作为输出结果展示

（2）初步功能调试，选择K=15的k近邻进行程序调试如下图1-5

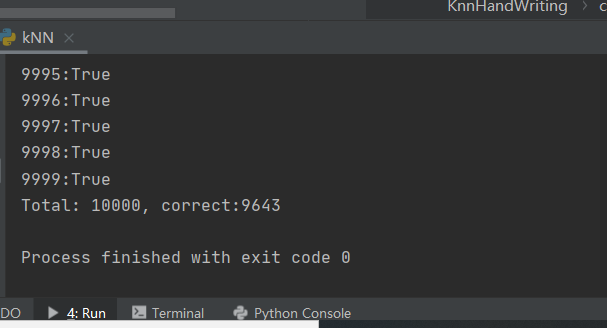


图1-5 K=15 k近邻运行

（3）程序运行正确后选取K=1，3，5，10，15，30，50，75，90，100进行测试如下图1-6所示

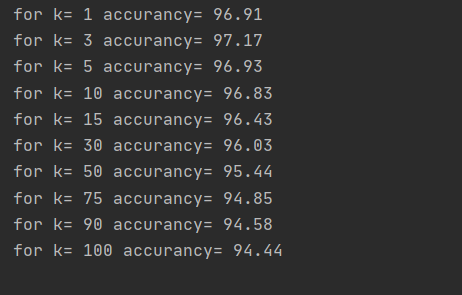


图1-6 不同k的运行结果

（4）将结果绘制成折线图如下图1-7所示

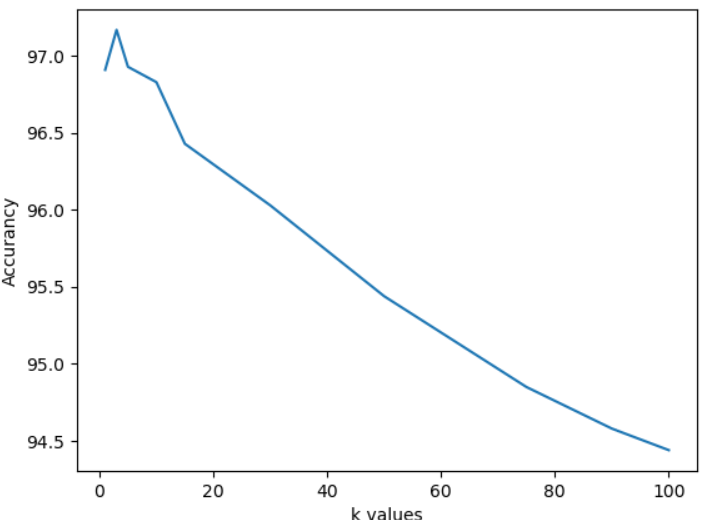


图1-7 k的折线图

## 七、结果分析

随着k的提高，准确度先上升在k=3-5时取得最大值，大于5时随这k的上升准确度下降，但都保持在百分之95左右。由此可见k值不是越大越好，本实验k值取3-5比较合适。