**西安电子科技大学人工智能学院**

**模式识别 课程实践报告**

成 绩

**实践课题名称 分析K近邻算法的错误率**

班级 2020039 姓名 刘焕宇 学号 20009200770

实践日期 2022 年 11 月

|  |
| --- |
| 指导教师评语：  指导教师：  年 月 日 |

一、实验内容

K近邻算法是一个经典的分类算法，也是最简单的机器学习算法之一,在1968年由和提出，此后不断发展，其理论不断丰富，如今K近邻算法体系已经较为成熟且完备，广泛应用于图像识别、文本分类、字符处理等各个领域。

该算法的思路是：给定测试样本，基于某种距离度量找出在训练集中与其最靠近的k个“邻居”的信息来进行预测。通常，在分类任务中可使用“投票法”,即选择这k个样本中出现最多的类别标记作为预测结果；在回归任务中可使用“平均法”，即将这k个样本的实值输出标记的平均值作为预测结果；还可基于距离远近进行加权平均或者加权投票，距离越近的样本权重越大。

有趣的是，不同衡量距离的尺度、不同的k值选择，会使KNN算法得到不同的结果。距离、k值的不同会随着样本特征、维度的不同呈现出多样性的变化。这就要求我们充分分析数据，提取出数据中的关键特征，并运用合适的距离公式。

本实验旨在分析不同距离衡量尺度、不同k值选择对分类错误率的影响。

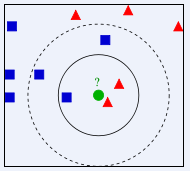


图1 K近邻算法示例

二、实验原理

首先介绍本实验选用的几个距离衡量尺度，设特征向量,

，则二者间的距离可以用以下方法计算：

1. 欧几里得距离(Euclidean distance)
2. 曼哈顿距离(Manhattan Distance)
3. 切比雪夫距离(Chebyshev distance)
4. 闵可夫斯基距离(Minkowski Distance)

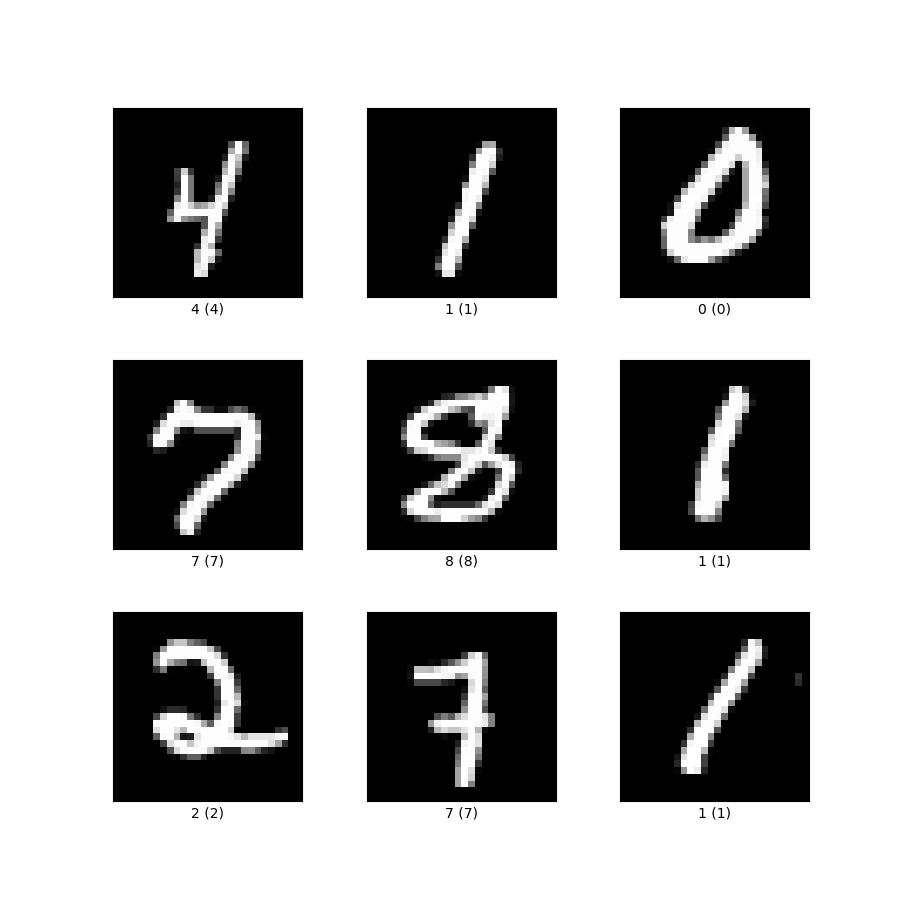
不难发现，欧氏距离、曼哈顿距离、切比雪夫距离分别为闵可夫斯基距离，，以及趋于无穷的特殊情况。故又可称为范数，范数与无穷范数

1. 余弦距离(Cosine Dsitance)

即效仿余弦公式，向量的内积除以向量模的乘积得到余弦相似度，再取反加上得到余弦距离

其次，由于算力的限制，本实验采用等样本采样的方式，从60000张有标签图片中等概率选取0~9每类数字各500，再以划分9:1的比例划分为标记点与待测点。多次采样取平均值作为某种距离，某个k值下的正确率。

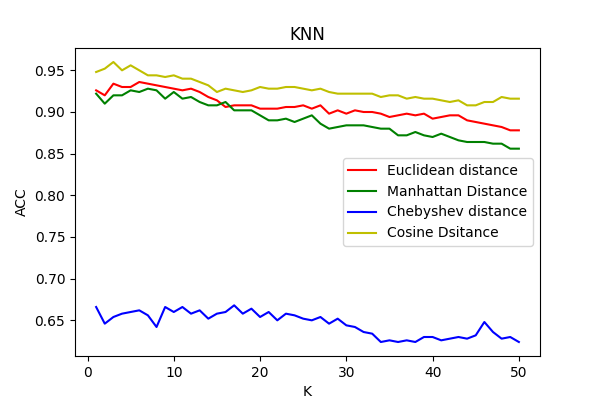
三、实验结果与分析

 是一个手写体数字的图片数据集，该数据集来由美国国家标准与技术研究所（National Institute of Standards and Technology (NIST)）发起整理，由60000个训练样本和10000个测试样本组成，每个样本都是一张28 \* 28像素的灰度手写数字图片。一共统计了来自250个不同的人手写数字图片，其中50%是高中生，50%来自人口普查局的工作人员。该数据集的收集目的是希望通过算法，实现对手写数字的识别。

原始的Special Database 3 数据集和Special Database 1数据集均是二值图像，MNIST从这两个数据集中取出图像后，通过图像处理方法使得每张图像都变成28×28大小的灰度图像，且手写数字在图像中居中显示。

图2 MNIST数据集

选择上文提到的四种距离尺度（闵可夫斯基距离除外），并让k从1到50取值，以k为横坐标，识别准确率为纵坐标，可作出如下实验结果图：

图3 KNN实验结果图

可以看出：

1. 切比雪夫距离准确率最差，余弦距离结果最好，有30%左右的差距不同的距离选择对分类的影响非常之大。
2. 准确率并不随着K值的增加而递增，呈现波动且下降的趋势，这可能与距离特征未能很好提取有关，也可能与算力不足、未能采用全部样本作为数据集有关。待算力增强后进一步实验

四、源程序代码

import **pandas** **as** **pd**

import **numpy** **as** **np**

import **matplotlib.pyplot** **as** **plt**

**def** read(path, label, siz=50):

df = pd.read\_csv(path)

data = df.values

np.random.shuffle(data)

**if** label:

np.random.shuffle(data)

n = len(data)

**if** label:

x = data[:, 1:].reshape(n, 28 \* 28)

x = x.astype("float32")

y = data[:, 0].reshape(n)

retx = np.zeros((siz \* 10, 28 \* 28))

rety = np.zeros(siz \* 10)

cnt = np.zeros(10)

top = 0

**for** i, vec **in** enumerate(x):

**if** cnt[y[i]] == siz:

**continue**

retx[top] = vec

rety[top] = y[i]

top += 1

**if** top == siz \* 10:

**break**

rety = rety.astype("int32")

**return** retx, rety

**else**:

x = data.reshape(n, 28 \* 28)

x = x.astype("float32")

**return** x

**def** getdis(x, y, typ=0):

**if** typ == 0: *# 欧氏距离*

**return** np.linalg.norm(x - y, ord=2)

**elif** typ == 1: *# 曼哈顿距离*

**return** np.linalg.norm(x - y, ord=1)

**elif** typ == 2: *# 切比雪夫距离*

**return** np.linalg.norm(x - y, ord=np.inf)

**elif** typ == 3: *# 余弦距离*

a\_norm = np.linalg.norm(x)

b\_norm = np.linalg.norm(y)

similiarity = np.dot(x, y.T) / (a\_norm \* b\_norm)

dist = 1. - similiarity

**return** dist

**def** knn(typ=0): *# 默认欧氏距离*

dis = np.zeros((len(x\_test), len(x\_train)))

idx = np.zeros((len(x\_test), len(x\_train)), dtype='int32')

**for** i, u **in** enumerate(x\_test):

**for** j, v **in** enumerate(x\_train):

dis[i, j] = getdis(u, v, typ)

idx[i] = np.argsort(dis[i]) *# 从小到大的下标*

ACC = list()

**for** k **in** range(1, kmax):

acc = 0

**for** i **in** range(len(x\_test)):

cnt = np.zeros(10)

**for** j **in** range(k): *# 前k小的投票*

cnt[y\_train[idx[i][j]]] += 1

ans = np.argmax(cnt)

**if** ans == y\_test[i]:

acc += 1

acc /= len(x\_test)

ACC.append(acc)

print(ACC)

**return** ACC

**if** \_\_name\_\_ == "\_\_main\_\_":

siz = 500

valid = int(siz \* 9)

x, y = read("./train.csv", label=**True**, siz=siz) *# 每类照片500张*

x\_train, y\_train = x[:valid], y[: valid]

x\_test, y\_test = x[valid:], y[valid:]

kmax = 51

ACC1 = knn(typ=0)

ACC2 = knn(typ=1)

ACC3 = knn(typ=2)

ACC4 = knn(typ=3)

px = range(1, kmax)

col = ["r", "g", "b", "y"]

plt.figure(figsize=(6, 4))

plt.plot(px, ACC1, c=col[0])

plt.plot(px, ACC2, c=col[1])

plt.plot(px, ACC3, c=col[2])

plt.plot(px, ACC4, c=col[3])

plt.title('KNN')

plt.xlabel('K')

plt.ylabel('ACC')

labels = ['Euclidean distance', 'Manhattan Distance', 'Chebyshev distance', 'Cosine Dsitance']

plt.legend(labels, loc='best', fancybox=**True**)

plt.show()