基于KNN的手写数字识别

朱浩泽 1911530 计算机科学与技术

实验描述

数据集

semeion_train.csv, semeion_test.csv

实验基本要求

编程实现kNN算法；给出在不同k值（1，3，5）情况下，kNN算法对手写数字的识别精度。

实验中级要求

与 Python 机器学习包中的kNN分类结果进行对比。

具体分析

1. 导入库

```
import numpy as np
import csv
import operator
```

2. 读取实验数据

将训练集和测试集合中的数据读取，存储在list中

```
train = []
train_result = []
test = []
test_right = []
# 导入数据
with open('semeion_train.csv') as csvfile:
    reader = csv.reader(csvfile)
    rows= [row for row in reader]
for i in rows:
    ls = []
    temp = i[0].split()
    num = 0
    for m in temp[:-10]:
        m = float(m)
        ls.append(m)
    for m in temp[-10:]:
        m = int(m)
        if m == 1:
            train_result.append(num)
        num += 1
    train.append(ls)
with open('semeion_test.csv') as csvfile:
    reader = csv.reader(csvfile)
    rows= [row for row in reader]
for i in rows:
    ls = []
    temp = i[0].split()
    for m in temp[:-10]:
        m = float(m)
        ls.append(m)
    num = 0
    for m in temp[-10:]:
        m = int(m)
```

```
            if m == 1:
                test_right.append(num)
            num += 1
        test.append(ls)
```

3. KNN算法实现

我们将测试集、训练集、训练集结果、k值作为函数的输入。计算待识别的元素与训练集中各个元素的几何距离，选出几何距离最小的k个训练集数据。根据这k个数据的结果，对输入数据进行分类。

```
    def KNN(inX, train, train_result, k):
        size = len(train)
        train = np.asarray(train)
        inX = np.asarray(inX)
        result = []
        for X in inX:
            exp = np.tile(X, (size , 1))
            differ = exp - train
            square = differ ** 2
            distance = (square.sum(axis = 1)) ** 0.5
            # print(distance)
            sorted_index = distance.argsort()
            temp = [0] * 10
            for m in sorted_index[:k]:
                temp[train_result[m]] += 1
                temp = np.asarray(temp)
            result.append(temp.argsort()[-1])
        return result
```

4. 测试集结果准确率计算

```
result1 = KNN(test, train, train_result, 1)
result3 = KNN(test, train, train_result, 3)
result5 = KNN(test, train, train_result, 5)
#求相似度
def simrate(ls1, ls2):
```

```
    num = 0
    l = len(ls1)
    for i in range(l):
        if ls1[i] == ls2[i]:
            num += 1
    return format(num / l, '.2%')
print("k = 1时的准确率是: ", simrate(result1, test_right))
print("k = 3时的准确率是: ", simrate(result3, test_right))
print("k = 5时的准确率是: ", simrate(result5, test_right))
#与sklearn库中做对比
from sklearn.neighbors import KNeighborsClassifier
knn1 = KNeighborsClassifier(1)
knn1.fit(train, train_result)
knn3 = KNeighborsClassifier(3)
knn3.fit(train, train_result)
knn5 = KNeighborsClassifier(5)
knn5.fit(train, train_result)
resultsk1 = knn1.predict(test)
resultsk3 = knn3.predict(test)
resultsk5 = knn5.predict(test)
print("sklearn中k = 1时的准确率是: ", simrate(resultsk1, test_right))
print("sklearn中k = 3时的准确率是: ", simrate(resultsk3, test_right))
print("sklearn中k = 5时的准确率是: ", simrate(resultsk5, test_right))
```

实验结果

实验结果截图如下

可以看出，k = 1时准确率为85.56%，使用sklearn库中的KNN分类器k = 1时准确率为85.56%，准确率持平

k = 3时准确率为85.15%，使用sklearn库中的KNN分类器k = 1时准确率为84.10%，准确率略高

k = 1时准确率为86.82%，使用sklearn库中的KNN分类器k = 1时准确率为83.89%，准确率略高

由此我们可以得出结论：

1. 自行编写的KNN分类器在该数据模型下表现略强于sklearn中的KNN分类器
2. 准确率与k值的选取有关，k过大或过小都会影响准确率