Handwritten Digit Recognition Based on K Nearest Neighbourhood

Experiment Description

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。

Experiment Process

Import Libraries

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

Read Data

Read train data and test data respectively and store in lists.

```
train = []
train_result = []
test = []
test_right = []
# import data
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)
```

Define KNN algorithm

Use train data, test data, train reult and k as the input of function. Calculate the geometry distance of the element to be identified, and choose the nearest k elements, which is used as the basis of classification.

```
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
```

Calculate the Accuracy

```
   result1 = KNN(test, train, train_result, 1)
   result3 = KNN(test, train, train_result, 3)
   result5 = KNN(test, train, train_result, 5)
   # calculate similarity
   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 similarity: ", simrate(result1, test_right))
    print("k = 3 similarity: ", simrate(result3, test_right))
    print("k = 5 similarity: ", simrate(result5, test_right))
    # compare with sklearn
    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 similarity: ", simrate(resultsk1, test_right))
    print("sklearn中k = 3 similarity: ", simrate(resultsk3, test_right))
    print("sklearn中k = 5 similarity: ", simrate(resultsk5, test_right))
```

```
knn中k = 1时的准确率是:  85.56%
knn中k = 3时的准确率是:  83.89%
knn中k = 5时的准确率是:  83.26%
sklearn中k = 1时的准确率是:  85.56%
sklearn中k = 3时的准确率是:  84.10%
sklearn中k = 5时的准确率是:  83.89%
```


It can be seen that:

When k = 1, the accuracy of the model from sklearn library and built from scratch are close.

When k = 3, the accuracy of the model from sklearn library is slightly higher than the model built from scratch.

When k = 5, the accuracy of the model from sklearn library is slightly higher than the model built from scratch.

Conclusion:

1. The accuracy of knn is slightly lower than the accuracy of the model from sklearn library.
2. The accuracy of both models drops slightly as k increases.