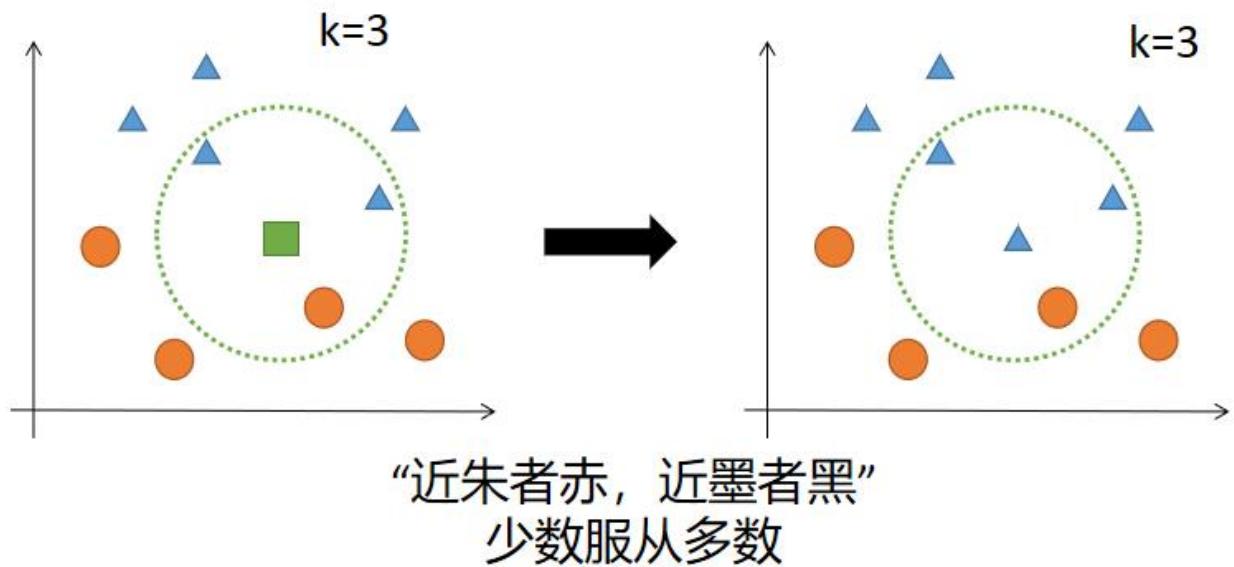


Table of Contents

- [1 什么是KNN?——>既可以分类也可以回归](#)
 - [1.1 简单定义: K Nearest Neighbors, K 个最近的邻居](#)
 - [1.2 关键:](#)
 - [1.3 k的含义? 离某点距离最近的k个点。](#)
- [2 问题:](#)
- [3 过程:](#)
- [4 最近距离, 那距离的定义?](#)
 - [4.1 闵可夫斯基 \(Minkowski\) 距离 \(\$L_p\$ 距离\)](#)
 - [4.2 曼哈顿 \(Manhattan\) 距离 \(\$L_1\$ 距离\)](#)
 - [4.3 欧式距离 \(Euclidean\) 距离 \(\$L_2\$ 距离\)](#)
 - [4.4 切比雪夫 \(Chebyshev\) 距离 \(\$L_\infty\$ 距离\)](#)
 - [4.5 在二维空间中 \$p\$ 取不同值时, 与原点的 \$L_p\$ 距离为1的点的图形:](#)
- [5 模型 model](#)
 - [5.1 抽象图解](#)
 - [5.2 具体模型:](#)
- [6 策略 strategy](#)
- [7 算法 algorithm](#)
- [8 K-NN的三个要点](#)
- [9 K-NN的回归](#)
- [10 K-NN编程小练习:](#)
- [11 与上例类似, 但是只是新增一个点判断是4类中哪一类](#)
- [12 鸢尾花数据集按照3-7拆分后用KNN进行分类, 并查看其效果](#)
- [13 新增5个点的两分类KNN](#)
- [14 玻璃分类 \(glass.data文件\)](#)
- [15 32*32手写数字识别 \(文件夹testDigits和文件夹trainingDigits\)](#)

1 什么是KNN?——>既可以分类也可以回归

- 1.1 简单定义: K Nearest Neighbors, K 个最近的邻居
- 1.2 关键:
 - (1) 近朱者赤, 近墨者黑: 根据样本近邻属性对其进行预测
 - (2) 少数服从多数
- 1.3 k的含义? 离某点距离最近的k个点。



2 问题:

- 已知 $Data = \{(x_i, y_i), i = 1, 2, \dots, N\}$, $x_i \in R^n$, $y_i \in \{c_1, c_2, \dots, c_k\}$ 为分成的类别数, 对于新的实例 $x \in R^n$, 如何确定 y ?

3 过程:

- (1) 计算距离 $d_i = d(x_i, x)$
- (2) 与近邻的 k 个点的距离排序
- (3) 多数表决: 多数类来决定结果

4 最近距离, 那距离的定义?

4.1 闵可夫斯基 (Minkowski) 距离 (L_p 距离)

- 设特征空间 χ 是 n 维实数向量空间 R^n , $x_i, x_j \in \chi$, $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)})^T$, $x_j = (x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(n)})^T$, x_i, x_j 的 L_p 距离定义为

$$L_p(x_i, x_j) = \left(\sum_{l=1}^n |x_i^{(l)} - x_j^{(l)}|^p \right)^{\frac{1}{p}}$$

4.2 曼哈顿 (Manhattan) 距离 (L_1 距离)

- 当 $p = 1$ 时的 L_1 距离定义为

$$L_1(x_i, x_j) = \sum_{l=1}^n |x_i^{(l)} - x_j^{(l)}|$$

4.3 欧式距离 (Euclidean) 距离 (L_2 距离)

- 当 $p = 2$ 时的 L_2 距离定义为

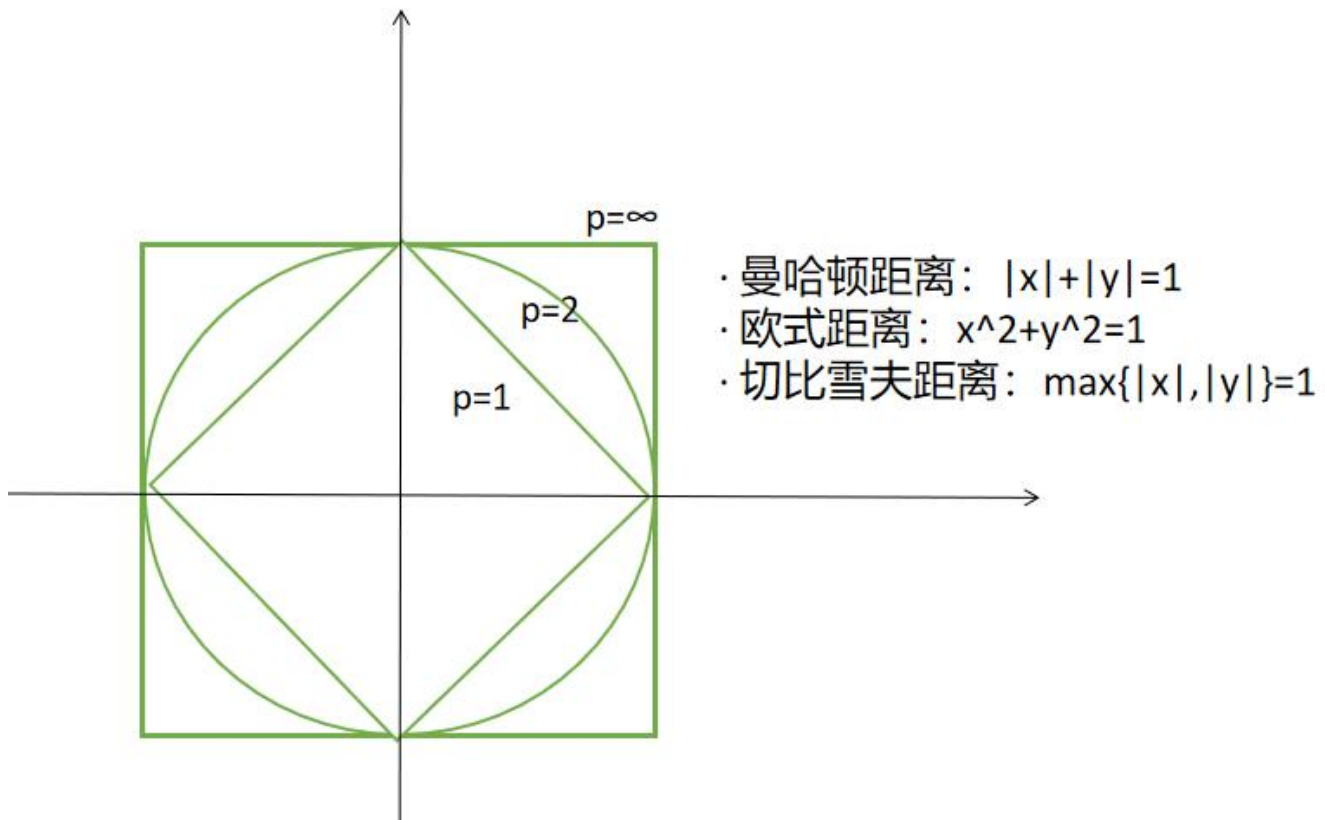
$$L_2(x_i, x_j) = \left(\sum_{l=1}^n |x_i^{(l)} - x_j^{(l)}|^2 \right)^{\frac{1}{2}}$$

4.4 切比雪夫 (Chebyshev) 距离 (L_∞ 距离)

- 当 $p = \infty$ 时的 L_∞ 距离定义为 ——> 各坐标距离的最大值

$$L_\infty(x_i, x_j) = \max_l |x_i^{(l)} - x_j^{(l)}|$$

4.5 在二维空间中 p 取不同值时, 与原点的 L_P 距离为1的点的图形:



5 模型 model

· 5.1 抽象图解

以x为中心, 包含k个样本的领域

index	X	Y	d	N _k (x)	C1	C2	C3	...	C _k
1	x1	y1	d1=dis(x1,x)	✓	×	✓	×	×	×
2	x2	y2	d2=dis(x2,x)	×	×	×	×	×	×
...
...
n	xn	yn	dn=dis(xn,x)	✓	✓	×	×	×	×

点x与其他各点的距离

按列求和, 多数表决

5.2 具体模型:

给定一个训练集: $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, 其中 x_i 为实例的特征向量, $y_i \in y = \{c_1, c_2, \dots, c_k\}$, 输入实例特征向量 x , 输出该实例特征向量的类别 y

$$y = \arg \max_{c_j} \sum_{x_i \in N_k(x)} I(y_i = c_j), i = 1, 2, \dots, N, j = 1, 2, \dots, K$$

其中 I 为指示函数, 即当 $y_i = c_j$ 时 I 为 1, 否则 I 为 0

6 策略 strategy

$$y = \arg \max_{c_j} \sum_{x_i \in N_k(x)} I(y_i = c_j), i = 1, 2, \dots, N, j = 1, 2, \dots, K$$

- KNN中的决策规则通常就是“投票选举”——少数服从多数的方式
- 如果损失函数是0-1损失函数, 那么分类函数就是:

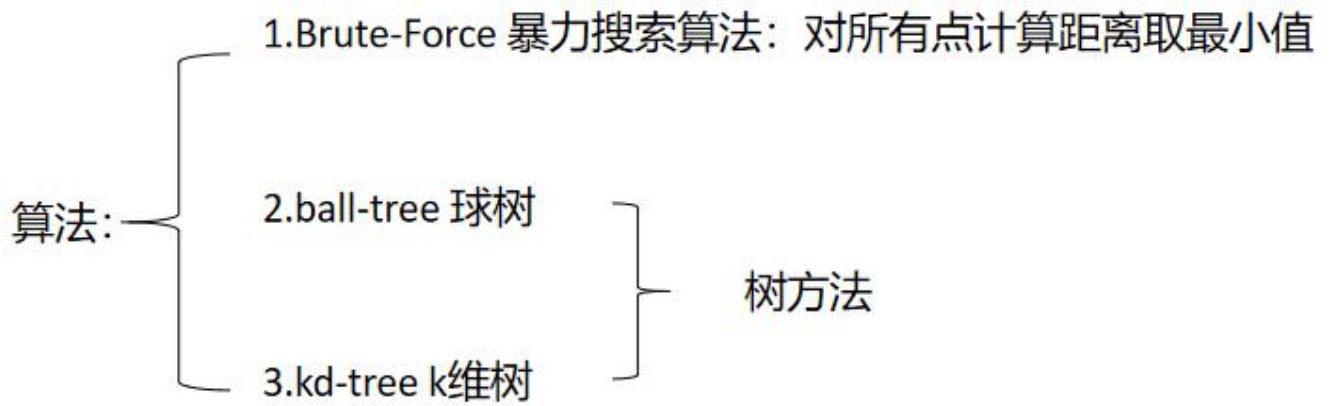
$$f: R^n \rightarrow c_1, c_2, \dots, c_k$$

- 对于相邻k个训练实例点构成集合N, 误分类率是:

$$\frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i \neq c_j) = 1 - \frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i = c_j)$$

- 要使误分类率最小, 那么就是要求正确的概率最大, 所以少数服从多数的规则正好可以满足经验风险最小化。

7 算法 algorithm



=====

输入: 给定一个训练集: $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, 其中 x_i 为实例的特征向量, $y_i \in y = \{c_1, c_2, \dots, c_k\}$ 为实例的类别, $i = 1, 2, \dots, N$

输出: 实例 x 所属的类别 y

(1) 给定距离度量, 在训练集 T 中找出与 x 最近邻的 k 个点, 涵盖这 k 个点的 x 的邻域记作 $N_k(x)$

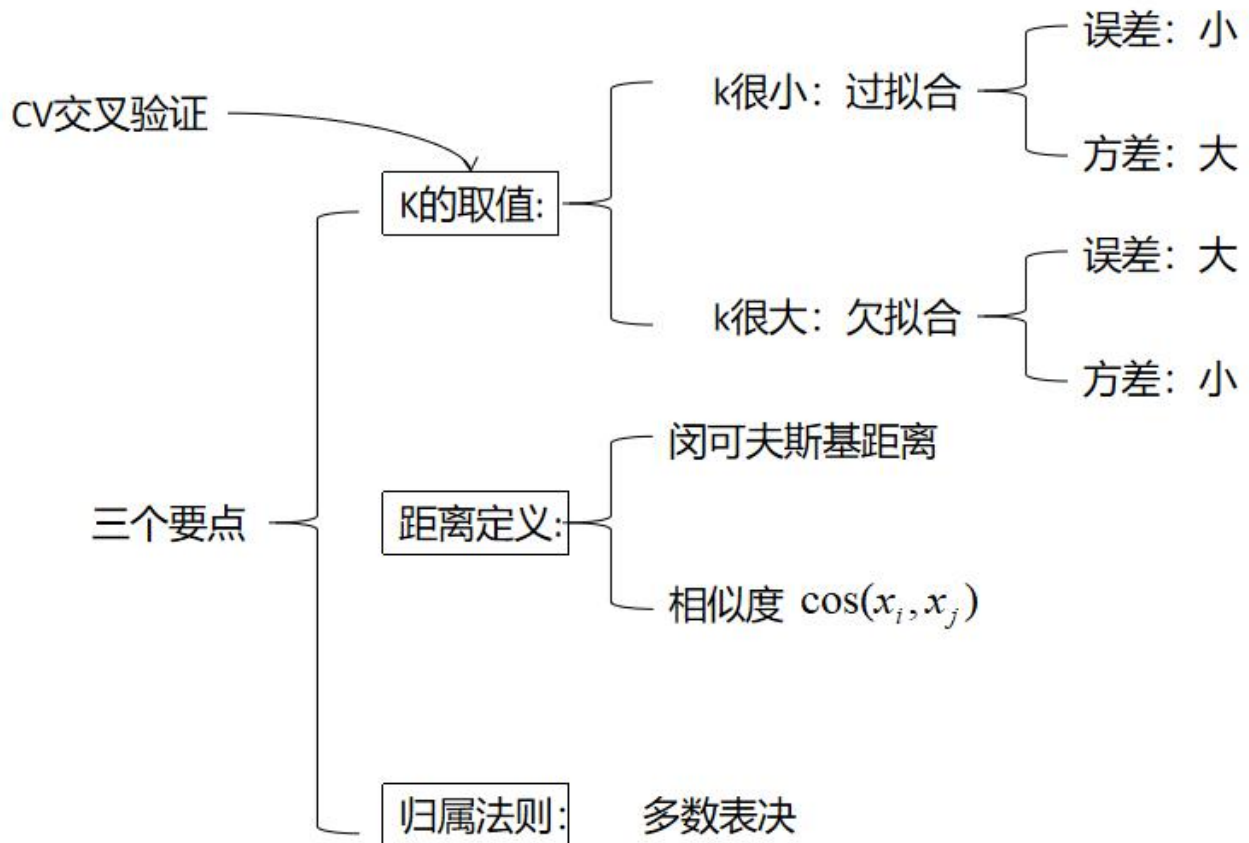
(2) 在 $N_k(x)$ 中根据分类决策规则 (eg. 多数表决) 决定 x 的类别 y

$$y = \arg \max_{c_j} \sum_{x_i \in N_k(x)} I(y_i = c_j), i = 1, 2, \dots, N, j = 1, 2, \dots, K$$

其中 I 为指示函数, 即当 $y_i = c_j$ 时 I 为 1, 否则 I 为 0

=====

8 K-NN的三个要点



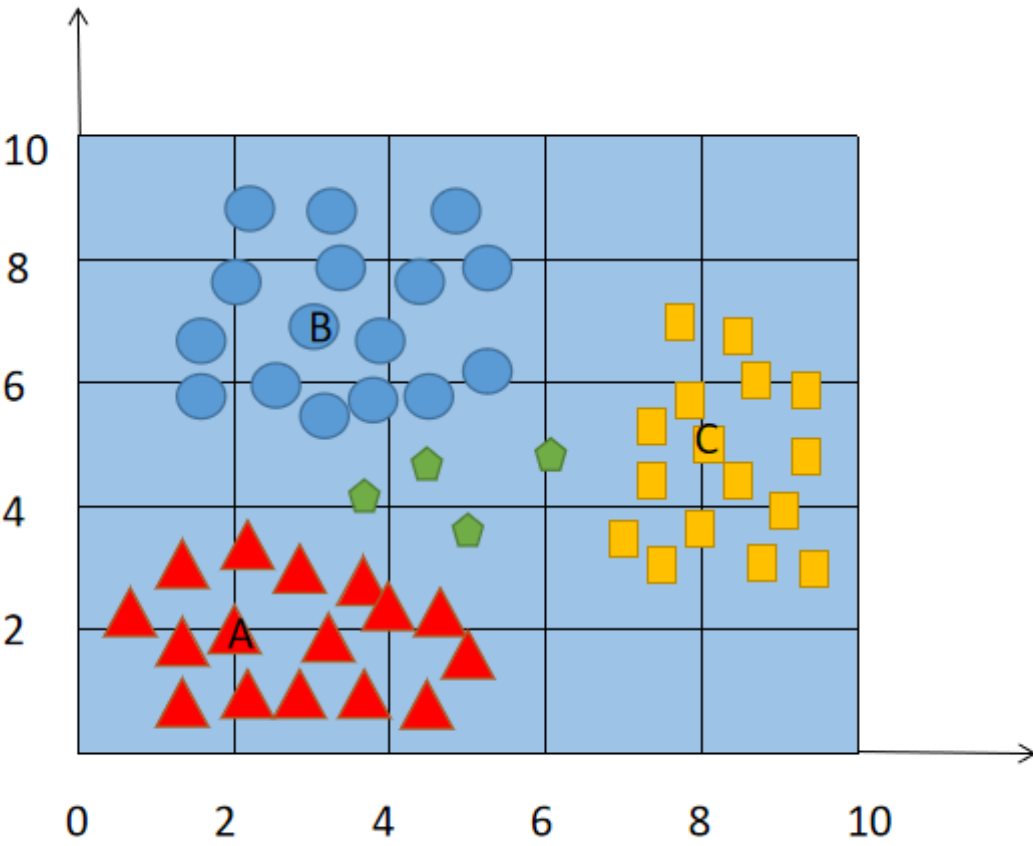
9 K-NN的回归

- 已知 $Data = \{(x_i, y_i), i = 1, 2, \dots, N\}$, $x_i \in R^n$, $y_i \in \{c_1, c_2, \dots, c_k\}$

$$y = \frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i = c_j), i = 1, 2, \dots, N, j = 1, 2, \dots, K$$

10 K-NN编程小练习:

- 在平面上产生三个点A(2,2),B(3,7),C(8,5)在上述三个点的附近用正态随机数各生成15个点作为A类、B类、C类, 再在 $x_1 \in [0, 10]$, $x_2 \in [0, 10]$ 上均匀生成若干点, 用KNN进行分类



In [1]:

```
1 import numpy as np
2 Y = np.repeat(np.array(['A', 'B', 'C']), 15)
3 X = np.zeros(shape=(len(Y), 2))
4 cc = np.array([[2, 2], [3, 7], [8, 5]])
5 for i in range(len(Y)):
6     if Y[i] == 'A':
7         X[i, 0] = np.random.normal(cc[0][0], 1.5)
8         X[i, 1] = np.random.normal(cc[0][1], 1.5)
9     elif Y[i] == 'B':
10        X[i, 0] = np.random.normal(cc[1][0], 1.5)
11        X[i, 1] = np.random.normal(cc[1][1], 1.5)
12    else:
13        X[i, 0] = np.random.normal(cc[2][0], 1.5)
14        X[i, 1] = np.random.normal(cc[2][1], 1.5)
15
16 testset = np.random.uniform(low=0, high=10, size=(3, 2))
17 abc = np.array(['A', 'B', 'C'])
18
19
20 def dis(x, y):
21     return np.sqrt(np.sum((x-y)**2))
22
23
24 def k_nn(feature, labels, x, k):
25     distance = np.zeros(shape=np.shape(feature)[0])
26     distance = np.array([dis(feature[i, :], x) for i in range(len(distance))])
27     thred = np.sort(distance)[0:k-1][-1]
28     Index = np.where(distance <= thred)
29     temp = np.array([np.sum(labels[Index] == abc[i]) for i in range(len(abc))])
30     for i in range(len(abc)):
31         if temp[i] == max(temp):
32             result = abc[i]
33
34     return result
35
36 k_nn(X, Y, testset, 3)
```

Out[1]:

'B'

11 与上例类似，但是只是新增一个点判断是4类中哪一类

In [2]:

```
1 import numpy as np
2 import operator
3
4 def kNN_classify0(inX, dataSet, labels, k):
5     dataSetSize = dataSet.shape[0]
6
7     # 计算欧式距离
8     diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet
9     sqDiffMat = diffMat ** 2
10    sqDistances = sqDiffMat.sum(axis=1)
11    distances = sqDistances ** 0.5
12    sortedDistIndicies = distances.argsort()
13    classCount = {}
14
15    # 选择距离最小的k个点
16    for i in range(k):
17        voteIlabel = labels[sortedDistIndicies[i]]
18        classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1
19
20    # 排序
21    sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1), reverse=True)
22    return sortedClassCount[0][0]
23
24 def testDataSet():
25     data = np.array(
26         [[0.1, 0.1], [0.31, 0.15], [0.66, 0.93], [0.54, 0.89], [1.0, 0.3], [0.86, 0.21], [0.4
27         labels = ['M', 'M', 'N', 'N', 'P', 'P', 'Q', 'Q']
28     return data, labels
29
30
31 data, labels = testDataSet()
32 predict = kNN_classify0([0.1, 0.5], data, labels, 4)
33 print("kNN predicts the [0.1,0.5] belongs to ", predict)
```

kNN predicts the [0.1,0.5] belongs to M

12 鸢尾花数据集按照3-7拆分后用KNN进行分类，并查看其效果

In [3]:

```

1 import numpy as np
2 from sklearn.datasets import load_iris
3 from sklearn.model_selection import train_test_split
4 from collections import Counter
5 #from sklearn.neighbors import KNeighborsClassifier, KDTree
6
7 class KNN(object):
8     def __init__(self, n_neighbors=3, p=2):
9         """
10         parameter: n_neighbors 临近点个数
11         parameter: p 距离度量
12         """
13         self.n = n_neighbors
14         self.p = p
15
16
17     def fit(self, X_train, y_train):
18         self.X_train = X_train
19         self.y_train = y_train
20
21     def predict(self, X):
22         # 取出n个点
23         knn_list = []
24         for i in range(self.n):
25             dist = np.linalg.norm(X - self.X_train[i], ord=self.p)
26             knn_list.append((dist, self.y_train[i]))
27
28         for i in range(self.n, len(self.X_train)):
29             max_index = knn_list.index(max(knn_list, key=lambda x: x[0]))
30             dist = np.linalg.norm(X - self.X_train[i], ord=self.p)
31             if knn_list[max_index][0] > dist:
32                 knn_list[max_index] = (dist, self.y_train[i])
33
34         # 统计
35         knn = [k[-1] for k in knn_list]
36         return Counter(knn).most_common()[0][0]
37
38     # 统计准确度
39     def score(self, X_test, y_test):
40         right_count = 0
41         for X, y in zip(X_test, y_test):
42             label = self.predict(X)
43             if label == y:
44                 right_count += 1
45         return right_count / len(X_test)
46
47 #####
48
49 iris = load_iris()
50 X = iris.data[:100, [0, 2]]
51 y = iris.target[:100]
52 y = np.where(y == 1, 1, -1)
53
54 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
55 knn = KNN()
56 knn.fit(X_train, y_train)
57 score = knn.score(X_test, y_test)
58 print("score = %s"%score)

```

socre = 1.0

13 新增5个点的两分类KNN

In [4]:

```

1  # 两分类的分离器
2  import numpy as np
3
4  def eu_dis(a, b):
5      dis = np.sqrt(np.dot(a-b, a-b))
6      return dis
7
8
9  class Knn_algorithm(object):
10     def __init__(self, X_train, X_test, Y_train, k): # Y_test,k):
11         self.X_train = X_train
12         self.X_test = X_test
13         self.Y_train = Y_train
14         # self.Y_test=Y_test
15         self.k = k
16
17     def classifiy(self):
18         Y_res = np.zeros(shape=np.shape(self.X_test)[0])
19         ndist = np.zeros(shape=np.shape(self.X_train)[0])
20         for ii in range(np.shape(self.X_test)[0]):
21             for jj in range(np.shape(self.Y_train)[0]):
22                 ndist[jj] = eu_dis(self.X_test[ii], self.X_train[jj])
23             temp = np.sort(ndist)[self.k-1]
24             index = np.where(ndist <= temp)
25             # count=np.sum(self.Y_train[index]==1)
26             if np.sum(self.Y_train[index] == 1) > self.k/2:
27                 Y_res[ii] = 1
28             else:
29                 Y_res[ii] = -1
30         return Y_res
31
32
33 X_1 = np.array([[3, 3], [4, 3], [1, 1], [4.5, 2.5], [0.5, 1],
34                [0.5, 1.5], [2, 3], [2.5, 4], [2, 0.5], [3, 0.1],
35                [0.7, 0.8], [1, 1.2], [2, 0.9], [3, 4], [4, 1]])
36 Y_1 = np.array([1, 1, -1, 1, -1, -1, 1, 1, -1, -1, -1, -1, -1, 1, 1])
37 X_2 = np.array([[1.3, 4], [2.2, 2], [0.8, 1.2], [3.4, 4.3], [1.7, 3.6]])
38
39
40 KNN = Knn_algorithm(X_1, X_2, Y_1, 5)
41 print(KNN.classifiy())

```

[1. -1. -1. 1. 1.]

14 玻璃分类 (glass.data文件)

In [5]:

```
1 import os
2 import numpy as np
3 import pandas as pd
4 import operator
5
6 #os.chdir("D:\\陈雪东文件\\2021秋季学期授课\\数据挖掘与机器学习\\code_myself\\ML_myself_code\\
7
8 glass=pd.read_csv('glass.data', sep=',', header=None)
9 glass.columns=['Id', 'RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Be', 'Ke', 'Type']
10
11 data_x=glass.loc[:, glass.columns[1:10]]
12
13 def testDataSet():
14     data = np.array(data_x)
15     labels = [str(x) for x in glass['Type'].tolist()]
16     return data, labels
17
18 def kNN_classify0(inX, dataSet, labels, k):
19     dataSetSize = dataSet.shape[0]
20
21     # 计算欧式距离
22     diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet
23     sqDiffMat = diffMat ** 2
24     sqDistances = sqDiffMat.sum(axis=1)
25     distances = sqDistances ** 0.5
26     sortedDistIndicies = distances.argsort()
27     classCount = {}
28
29     # 选择距离最小的k个点
30     for i in range(k):
31         voteIlabel = labels[sortedDistIndicies[i]]
32         classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1
33
34     # 排序
35     sortedClassCount = sorted(classCount.items(), key=operator.itemgetter(1), reverse=True)
36     return sortedClassCount[0][0]
37
38
39 data, labels = testDataSet()
40 predict = kNN_classify0(np.array(glass.loc[23, glass.columns[1:10]]), data, labels, 5)
41 print(predict)
42 print(labels[23])
```

1
1

In [6]:

```

1 import os
2 import numpy as np
3 import pandas as pd
4 import operator
5
6 #os.chdir("D:\\陈雪东文件\\2021秋季学期授课\\数据挖掘与机器学习\\code_myself\\ML_myself_code\\
7
8 glass=pd.read_csv('glass.data', sep=',', header=None)
9 glass.columns=['Id', 'RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Be', 'Ke', 'Type']
10
11 data_x=glass.loc[:, glass.columns[1:10]]
12
13 def testDataSet():
14     data = np.array(data_x)
15     labels = [str(x) for x in glass['Type'].tolist()]
16     return data, labels
17
18 #####
19 #####
20
21 def eu_dis(a,b):
22     dis=np.sqrt(np.dot(a-b,a-b))
23     return dis
24
25 class Knn_algorithm(object):
26     def __init__(self, X_train, X_test, Y_train, k):#Y_test, k):
27         self.X_train=X_train
28         self.X_test=X_test
29         self.Y_train=Y_train
30         #self.Y_test=Y_test
31         self.k=k
32
33     def classifiy(self):
34         Y_res=np.zeros(shape=np.shape(self.X_test)[0])
35         ndist=np.zeros(shape=np.shape(self.X_train)[0])
36         for ii in range(np.shape(self.X_test)[0]):
37             for jj in range(np.shape(self.Y_train)[0]):
38                 ndist[jj]=eu_dis(self.X_test[ii], self.X_train[jj])
39                 temp=np.sort(ndist)[self.k-1]
40                 index=np.where(ndist<=temp)
41                 #count=np.sum(self.Y_train[index]==1)
42                 if np.sum(self.Y_train[index]==1)>self.k/2:
43                     Y_res[ii]=1
44                 else:
45                     Y_res[ii]=-1
46         return Y_res
47
48 #####
49 #####
50
51 KNN = Knn_algorithm(X_1, X_2, Y_1, 5)
52 print(KNN.classifiy())

```

[1. -1. -1. 1. 1.]

In [7]:

```

1  import os
2  import numpy as np
3  import pandas as pd
4  #import operator
5
6  #os.chdir("D:\\陈雪东文件\\2021秋季学期授课\\数据挖掘与机器学习\\code_myself\\ML_myself_code\\
7
8  glass=pd.read_csv('glass.data', sep=',', header=None)
9  glass.columns=['Id', 'RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Be', 'Ke', 'Type']
10
11  data_x=glass.loc[:, glass.columns[1:10]]
12
13  def DataSet():
14      data = np.array(data_x)
15      labels = np.array(glass['Type'])
16      return data, labels
17
18  #####
19  #####
20
21  def eu_dis(a,b):
22      dis=np.sqrt(np.dot(a-b,a-b))
23      return dis
24
25  class Knn_algorithm(object):
26      def __init__(self, X_train, X_test, Y_train, k):
27          self.X_train=X_train
28          self.X_test=X_test
29          self.Y_train=Y_train
30          self.k=k
31
32      def classifiy(self):
33          Y_res=np.zeros(shape=np.shape(self.X_test)[0])
34          ndist=np.zeros(shape=np.shape(self.X_train)[0])
35          uniY_train=np.unique(self.Y_train)
36          count=np.zeros(shape=np.shape(uniY_train)[0])
37          for ii in range(np.shape(self.X_test)[0]):
38              for jj in range(np.shape(self.Y_train)[0]):
39                  ndist[jj]=eu_dis(self.X_test[ii], self.X_train[jj])
40                  temp=np.sort(ndist)[self.k-1]
41                  index=np.where(ndist<=temp)
42                  count=[np.sum(np.array(self.Y_train[index])==uniY_train[i]) for i in range(len(
43                      #count=np.sum(self.Y_train[index]==1)
44                      bb=np.where(count==max(count))
45                      Y_res[ii]=uniY_train[bb[0][0])
46                      #if np.sum(self.Y_train[index]==1)>self.k/2:
47                      #    Y_res[ii]=1
48                      #else:
49                      #    Y_res[ii]=-1
50                  return Y_res
51
52  #####
53  #####
54
55  X_1,Y_1=DataSet()
56  X_2=np.array(data_x.iloc[103:205,:])
57
58  KNN = Knn_algorithm(X_1, X_1, Y_1, 3)
59  print(KNN.classifiy())

```

```
[1. 1. 2. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 3. 1. 1. 1. 1. 1.
1. 1. 3. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 3. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2.
2. 2. 2. 2. 2. 2. 1. 2. 2. 2. 1. 2. 2. 2. 2. 2. 2. 2. 2. 1. 2. 2. 1. 2.
2. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 6. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
2. 2. 2. 2. 2. 1. 1. 2. 2. 2. 2. 5. 2. 2. 1. 2. 1. 2. 2. 2. 2. 2. 2. 2.
1. 1. 1. 3. 3. 1. 1. 1. 3. 3. 1. 1. 1. 1. 3. 3. 3. 3. 3. 7. 5. 5. 5. 5.
5. 5. 5. 5. 5. 5. 5. 5. 6. 6. 6. 6. 7. 6. 2. 2. 6. 7. 7. 1. 2. 7. 6. 7.
7. 7. 7. 7. 7. 7. 7. 7. 7. 7. 1. 7. 7. 7. 7. 7. 7. 7. 7. 7. 7. 7.]
```

15 32*32手写数字识别（文件夹testDigits和文件夹trainingDigits）

In [8]:

```

1 import os
2 import numpy as np
3 import operator
4 from numpy import *
5
6 def kNN_classify0(inX, dataSet, labels, k):
7     dataSetSize = dataSet.shape[0]
8
9     # 计算欧式距离
10    diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet
11    sqDiffMat = diffMat ** 2
12    sqDistances = sqDiffMat.sum(axis=1)
13    distances = sqDistances ** 0.5
14    sortedDistIndicies = distances.argsort()
15    classCount = {}
16
17    # 选择距离最小的k个点
18    for i in range(k):
19        voteIlabel = labels[sortedDistIndicies[i]]
20        classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1
21
22    # 排序
23    sortedClassCount = sorted(
24        classCount.items(), key=operator.itemgetter(1), reverse=True)
25    return sortedClassCount[0][0]
26
27
28 def img2vector(filename):
29     """
30     格式化输入的数据图像，转换为向量
31     :param filename: 文本文件名称
32     :return: 1*1024的数字向量
33     """
34    dataVect = np.zeros((1, 1024)) # 预先分配一部分数据
35    fr = open(filename)
36    for i in range(32):
37        lineText = fr.readline()
38        for j in range(32):
39            dataVect[0, 32 * i + j] = int(lineText[j]) # 利用偏移量使整个手写数字数据分布在类
40    return dataVect
41
42 def handwritingClassificationTest():
43    hwLabels = []
44    trainingFileList = os.listdir('trainingDigits') # 加载训练数据
45    m = len(trainingFileList)
46    trainingMat = np.zeros((m, 1024))
47    for i in range(m):
48        fileNameStr = trainingFileList[i]
49        fileStr = fileNameStr.split('.')[0] # 切分数据
50        classNumStr = int(fileStr.split('_')[0])
51        hwLabels.append(classNumStr)
52        trainingMat[i, :] = img2vector('trainingDigits/%s' % fileNameStr)
53    testFileList = os.listdir('testDigits') # 遍历测试数据集
54    errorCount = 0.0
55    mTest = len(testFileList)
56    for i in range(mTest):
57        fileNameStr = testFileList[i]
58        fileStr = fileNameStr.split('.')[0] # 切分数据
59        classNumStr = int(fileStr.split('_')[0])

```



```
60     vectorUnderTest = img2vector('testDigits/%s' % fileNameStr)
61     classifierResult = kNN_classify0(
62         vectorUnderTest, trainingMat, hwLabels, 3)
63     print("the classifier came back with: %d, the real answer is: %d" %
64         (classifierResult, classNumStr))
65     if (classifierResult != classNumStr):
66         errorCount += 1.0
67     # 打印测试数据
68     print("\nthe total number of errors is: %d" % errorCount)
69     print("\nthe total accuracy is: %f" % (1 - errorCount / float(mTest)))
70
71 handwritingClassificationTest()
```

```
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
the classifier came back with: 0, the real answer is: 0
```

In []:

1