实验

**实验要求：**用k-means算法对附件dataforkmeans - .txt 中的数据进行聚类，

k-means算法具体可网上查询具体算法进行学习。

**环境：python 3**

数据分布：



**验收方式**：第八周上机时验收，并交实验报告

**报告书写要求**：

一，k-means算法

二，程序实现：

三，分类结果

四：收获与体会

附：数据dataforkmeans - .txt

5.9 16.35

6.85 17.05

7.35 16.3

6.85 14.7

6.6 15.7

6.15 15.05

6.6 14.1

7.85 15.45

8.5 17.35

8.45 14.85

8.1 14.25

9.35 16.05

9.95 17.95

8.5 16.5

9.8 15.2

10.25 16.85

9.7 14.15

8.05 13.65

6.7 13.1

12.7 17.7

14.4 18.75

14 18.1

13.05 16.95

12.3 15.45

15.6 14.35

15 16.75

13.4 16.2

13.35 14.8

14.3 15.7

16.3 18.3

16.1 17.5

16.6 16.3

16.95 14.4

14.7 13.7

14.3 14.7

15.75 15.9

13.95 17.25

13.3 18.75

13.4 19.9

15.45 19.55

15.35 17.75

12.5 13.9

13.45 13.25

6.15 7.9

6.8 8.85

8.6 9.8

8.9 8.35

7.55 7.8

6.3 5.65

6.8 5.5

7.4 6.75

8.35 6.3

8.05 5.55

8.55 5.25

9.4 6.6

8.9 7.55

9.75 8.9

9.95 8.2

10.1 7.55

10.25 5.65

9.25 5.2

9.4 4.3

6.85 4.7

7.3 4.35

7.8 4.4

6.55 6.7

8.2 8.75

10.1 9.8

10.7 7.5

10.7 6.1

10.2 6.7

9.75 4.8

10.2 5

14.7 9.45

15.65 10.75

15.65 10.2

15.65 8.9

15.65 8.45

16.85 8.45

17.15 10.35

17.9 11.6

18.5 9.55

18.6 7.9

18.15 7.1

15.9 5.85

14.35 6.9

12.85 9.35

13.35 10.3

15.1 11.75

16.1 11.75

14.95 10.9

13.65 8.95

13.55 8

14.15 7.95

15.1 7.25

15.45 7.1

15.75 7.1

16.6 7.15

16.75 9.95

16.55 10.95

17.2 9.05

17.4 7.65

17.75 10.7

19.4 10.15

14.45 6.15]

**[算法分析]**

K-Means的算法可以概述为：

1、 选择初始化的个样本作为初始聚类中心；

2、 针对数据集中每个样本计算它到个聚类中心的距离:

并将其分到距离最小的聚类中心所对应的类中；

3、针对每个类别

重新计算它的聚类中心

（即属于该类的所有样本的质心）；

4、重复上面 2 3 两步操作，直到达到某个中止条件（迭代次数、最小误差变化等）。

Kmeans类设计：

设定k值，取传入data的前k个点作为初始簇心。

计算每个点到每一个簇心的距离，选取值最小的一簇加入。

针对每一簇，计算属于这一簇的所有点的几何中心，作为新的簇中心。

不断迭代这一过程，直到达到最大迭代次数，或者簇中心不再变化或仅轻微抖动。

**[程序代码]**

from sklearn.cluster import KMeans

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import math

path = "dataforkmeans.txt"

with open(path, 'r') as f:

data = f.read()

data = data.replace('[', '')

data = data.replace(']', '')

with open ("data.txt", 'w') as s:

s.write(data)

df = pd.read\_csv('data.txt', header=None)

df = df.iloc[:, 0:2]

print (df)

estimator = KMeans(n\_clusters=4)

estimator.fit(df)

label = estimator.labels\_

x0 = df[label == 0]

x1 = df[label == 1]

x2 = df[label == 2]

x3 = df[label == 3]

plt.scatter(x0.iloc[:, 0], x0.iloc[:, 1], c='red', label='0')

plt.scatter(x1.iloc[:, 0], x1.iloc[:, 1], c='green', label='0')

plt.scatter(x2.iloc[:, 0], x2.iloc[:, 1], c='blue', label='0')

plt.scatter(x3.iloc[:, 0], x3.iloc[:, 1], c='yellow', label='0')

plt.show()

class K\_Means():

def \_\_init\_\_(self, k=2) -> None:

self.k = k

self.max\_epoch = 100

self.ecount = 20

def fit(self, xdata):

count = 0

self.prev = {}

self.centers = {}

for i in range(self.k):

tmp = xdata.iloc[i]

self.centers[i] = [tmp.iloc[0], tmp.iloc[1]]

self.prev[i] = [tmp.iloc[0], tmp.iloc[1]]

for i in range(self.max\_epoch):

clf = {}

for i in range(self.k):

clf[i] = []

for i in range(xdata.shape[0]):

dist = []

for j in range(self.k):

x = pow(self.centers[j][0]-xdata.iloc[i, 0], 2)

y = pow(self.centers[j][1]-xdata.iloc[i, 1], 2)

distance = math.sqrt(x+y)

dist.append(distance)

classfication = dist.index(min(dist))

clf[classfication].append([xdata.iloc[i, 0], xdata.iloc[i, 1]])

for c in self.centers:

self.centers[c] = np.average(clf[c], axis=0)

for j in range(self.k):

if self.centers[j][0] == self.prev[j][0] and self.centers[j][1] == self.prev[j][1]:

count = count + 1

else:

count = 0

if count > self.ecount:

break

for c in self.centers:

self.prev[c] = self.centers[c]

def predict(self, xdata):

label = []

for i in range(xdata.shape[0]):

dist = []

for j in range(self.k):

x = pow(self.centers[j][0]-xdata.iloc[i, 0], 2)

y = pow(self.centers[j][1]-xdata.iloc[i, 1], 2)

distance = math.sqrt(x+y)

dist.append(distance)

classfication = dist.index(min(dist))

label.append(classfication)

return np.array(label)

df = pd.read\_csv('data.txt', header=None)

df = df.iloc[:, 0:2]

optimizer = K\_Means(k = 4)

optimizer.fit(df)

label = optimizer.predict(df)

x0 = df[label == 0]

x1 = df[label == 1]

x2 = df[label == 2]

x3 = df[label == 3]

plt.scatter(x0.iloc[:, 0], x0.iloc[:, 1], c='red', label='0')

plt.scatter(x1.iloc[:, 0], x1.iloc[:, 1], c='green', label='0')

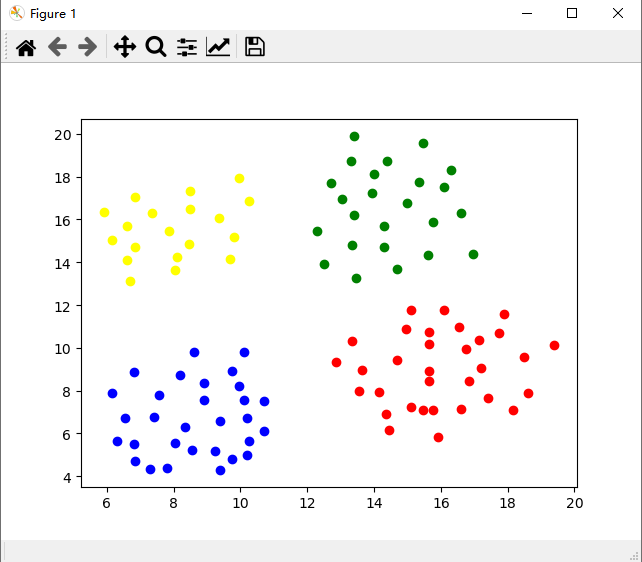
plt.scatter(x2.iloc[:, 0], x2.iloc[:, 1], c='blue', label='0')

plt.scatter(x3.iloc[:, 0], x3.iloc[:, 1], c='yellow', label='0')

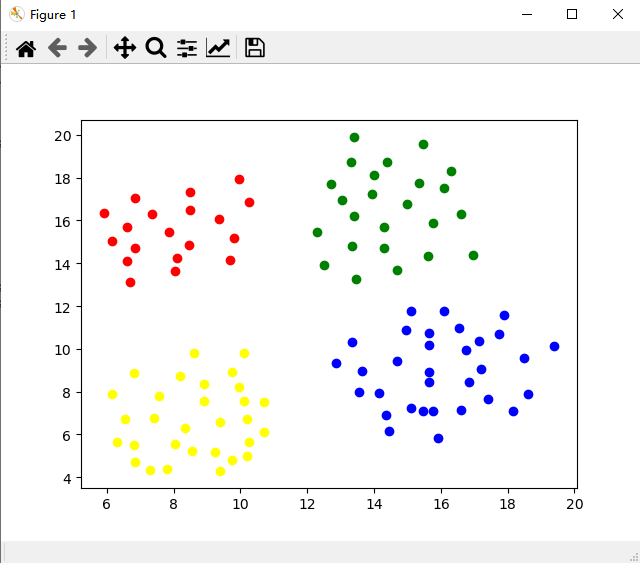
plt.show()

**[分类结果]**

Sklearn.cluster提供的KMeans算法：



自行实现的K-Means类：



**[收获与体会]**

在本次实验，我了解了kmeans算法的思想与具体实现流程，了解了机器学习的基本思想，掌握了迭代法的简单编程实现。在使用算法时，深入了解其具体实现流程有助于我们加深对算法的理解，有助于对方法的进一步应用。