模式识别第二次课程作业报告

57119101王晨阳

# 问题描述

试用K均值法对如下模式分布进行聚类分析。编程实现，编程语言不限。

# 算法介绍

设待分类的模式特征矢量集为 ，类的数目 事先取定。 为第 个聚类集，聚类中心为 ，包含的样本数为

* 任选 个模式特征向量 作为初始聚类中心
* 若 ，则
* 若 ，则结束
* 否则

由于只要求实现 k-means，所以我们先把所有点画出来看一下

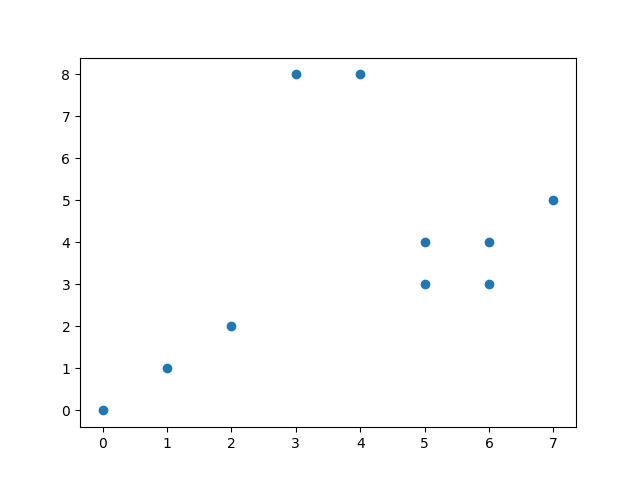


图1 数据点

经过简单观察，分为3类较为合适。

我们直接模拟以上过程。程序中几乎每条语句我都做了详细的注释，因此不再赘述：

import numpy as np

from matplotlib import pyplot

class KMeans(object):

    # initializations

    def \_\_init\_\_(self, k=1):  # k is the number of clusters

        self.K = k

    # fit the data

    def Fit(self, data):

        self.Centers = {}  # Centers is the dictionary of centers of clusters

        # choose the first K points to initialize centers

        for i in range(self.K):

            self.Centers[i] = data[i]

        while True:

            self.Clusters = {}  # Clusters is the dictionary of clusters of points

            # initialize the lists of centers

            for i in range(self.K):

                self.Clusters[i] = []

            # calculate the distance between each point and each center

            for X in data:

                Distances = []  # Distances is the list of distances of each point to each center

                for Center in self.Centers:

                    # calculate the 2-norm distance

                    Distances.append(np.linalg.norm(

                        X - self.Centers[Center], ord=2))

                # find out the index of the closest center

                Category = Distances.index(min(Distances))

                # add the point to the list of points belonging to the closest center

                self.Clusters[Category].append(X)

            # record the previous centers

            PreCenters = dict(self.Centers)

            # calculate the new centers

            for Cluster in self.Clusters:

                # calculate the mean of the points in the cluster

                self.Centers[Cluster] = np.average(

                    self.Clusters[Cluster], axis=0)

            # check if the new centers are the same as the previous centers

            Flag = True  # Flag is the flag of weather the centers are the same

            for Center in self.Centers:

                if self.Centers[Center].any() != PreCenters[Center].any():

                    # if the centers are not the same

                    Flag = False

            if Flag:  # satisfy the stopping condition

                break

    '''

    # predict the cluster for input data point

    def Predict(self, data):

        for X in data:

            Distances = []

            for Center in self.Centers:

                Distances.append(np.linalg.norm(

                    X - self.Centers[Center], ord=2))

            Category = Distances.index(min(Distances))

        return Category

    '''

if \_\_name\_\_ == '\_\_main\_\_':

    # the data set

    x = np.array([[0, 0], [3, 8], [2, 2], [1, 1], [5, 3],

                  [4, 8], [6, 3], [5, 4], [6, 4], [7, 5]])

    # plot the data

    '''

    pyplot.scatter(x[:, 0], x[:, 1])

    pyplot.show()

    '''

    Kmeans = KMeans(k=3)  # k=3 clusters

    Kmeans.Fit(x)  # fit the data

    # plot the data

    print(Kmeans.Centers)

    # plot the centers

    for Center in Kmeans.Centers:

        # red for the first center, blue for the second center, and green for the third center

        pyplot.scatter(Kmeans.Centers[Center][0], Kmeans.Centers[Center][1], marker='\*', s=150, c=(

            'r' if Center == 0 else ('b' if Center == 1 else ('g' if Center == 2 else 'y'))))

    # plot the data

    for X in Kmeans.Clusters:  # iterate through the clusters

        # iterate through the points in the cluster

        for point in Kmeans.Clusters[X]:

            # red for the first center, blue for the second center, and green for the third center

            pyplot.scatter(point[0], point[1], c=(

                'r' if X == 0 else ('b' if X == 1 else ('g' if X == 2 else 'y'))))

    # prediction

    '''

    predict = np.array([[0, 9], [3, 9]])

    for X in predict:

        Category = Kmeans.Predict(X)

        pyplot.scatter(X[0], X[1], c=('r' if Category == 0 else (

            'b' if Category == 1 else ('g' if Category == 2 else 'y'))), marker='x')

    '''

    pyplot.show()

# 结果分析

我们在 Python 3.9.6 64bit 环境下测试程序。

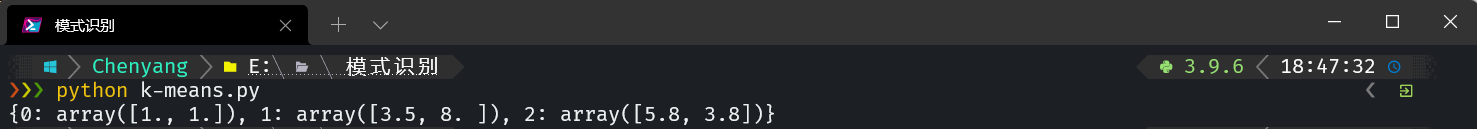


图2 Windows terminal 截图

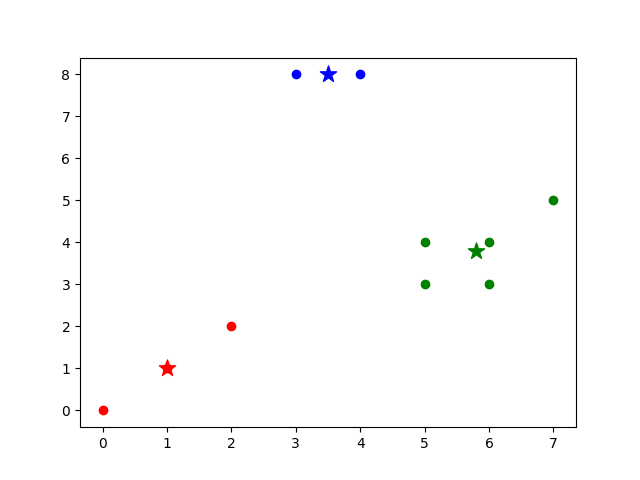


图3 分类结果

得到三个聚类中心分别为

经验证，结果无误。