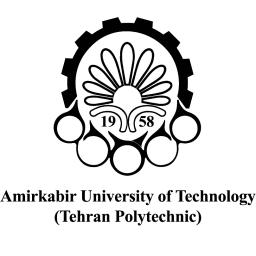
**گزارشکار آزمایش جلسه چهارم**



**الگوریتم clustering و پیاده سازی الگوریتم Kmean**

**اعضای گزارش**

**عباس بدیعی**

**محمد حسین طیب زاده**

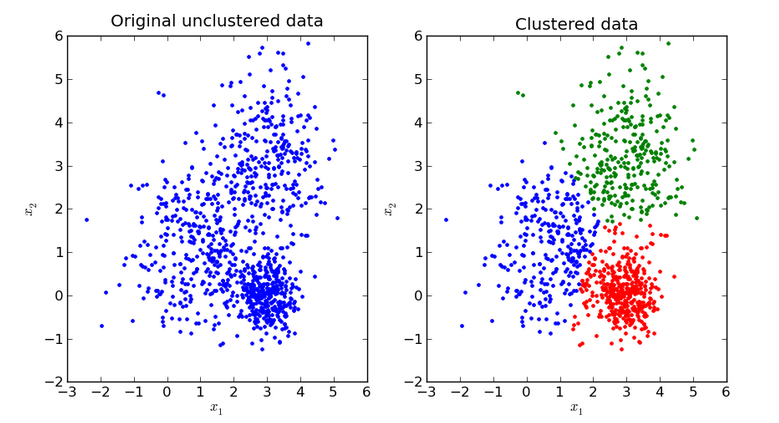
**نویسنده گزارش : عباس بدیعی**

بسم الله الرحمن الرحیم

در این آزمایش به الگوریتم clustering و پیاده سازی الگوریتم Kmean پرداخته ایم .

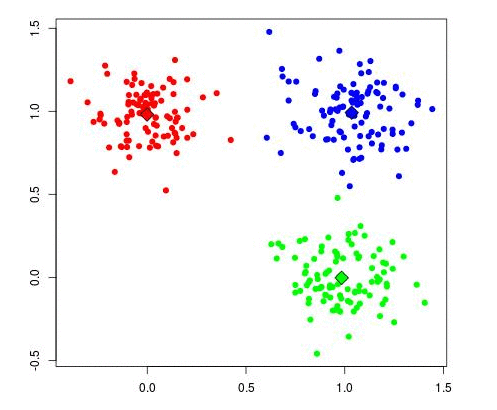
الگوریتم های Clustering برای خوشه بندی داده ها مورد استفاده قرار می گیرند ، این الگوریتم ها را می‌توان بر اساس مدل خوشه ای طبقه‌بندی کرد.  
یکی از الگوریتم های Clustering که در مبحث خوشه بندی مطرح می شود الگوریتم k-means می باشد ، این الگوریتم داده ها را به k ناحیه تقسیم می کند و براساس یک معیار شباهت آنها را طبقه بندی می کند.

K-Means یک الگوریتم بسیار ساده است که داده‌ها را در K خوشه، گروه‌بندی می‌کند. نتیجه ای از خوشه‌بندی K-Means به شکل زیر است.



الگوریتم K-Means، با فرض داشتن ورودی‌های xn ،… ،x3 ،x2 ،x1 به شکل زیر کار می‌کند.

* گام اول: انتخاب K‌ نقطه تصادفی به عنوان مرکز خوشه‌ها که به آن )مرکز دسته ها( گفته می‌شود.
* گام دوم: هر xi به نزدیک‌ترین خوشه با محاسبه فاصله آن از هر مرکز تخصیص داده می‌شود.
* گام سوم: پیدا کردن مرکز خوشه‌های جدید با محاسبه میانگین نقاط تخصیص داده شده به یک خوشه
* گام ۴: تکرار گام ۲ و ۳ تا هنگامی که هیچ یک از نقاط تخصیص داده شده به خوشه‌ها تغییر نکنند. ( ارور از یک حد معین کاهش یابد ).



**گام اول**

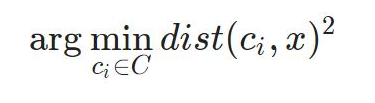
Kمرکز خوشه (مرکزوار) به طور تصادفی انتخاب می‌شوند. فرض می‌شود که این مرکزوارها ck،… ،c3،c2،c1 هستند. می‌توان گفت که:

C = c1, c2, …, ck

C مجموعه‌ای از همه مراکز خوشه هاست.

**گام دوم**

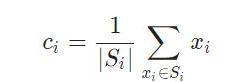
در این گام، هر مقدار ورودی به نزدیک‌ترین مرکز تخصیص داده می‌شود. این کار با محاسبه فاصله اقلیدسی (L2) بین نقطه و هر مرکزوار یا همان مرکز دسته انجام می‌شود.



که در آن dist() فاصله اقلیدسی است.

**گام سوم**

در این گام، مرکزخوشه های جدید با محاسبه میانگین کلیه نقاط اختصاص داده شده به هر خوشه محاسبه می‌شود.



Si مجموعه‌ای از همه نقاط تخصیص داده شده به خوشه ith است.

### گام چهارم

در این گام، مراحل ۲ و ۳ تکرار می‌شوند تا هیچ یک از نقاط تخصیص داده شده به خوشه‌ها تغییر نکنند. این یعنی تا هنگامی که خوشه‌ها پایدار شوند، الگوریتم تکرار می‌شود.

در کد مقدار k=10 می باشد. و کد مطابق تعریف های بالا به صورت زیر می باشد :

import numpy as np

import cv2

class Kmean:

    def \_\_init\_\_(self , cluster\_num , input\_dim):

        self.cluster\_num = cluster\_num

        self.C =[]# 300\*np.random.random([cluster\_num , input\_dim])

        # for i in range(cluster\_num):

        #   self.C.append(np.array([np.random.randint(255), np.random.randint(255), np.random.randint(255)]))

        # self.C = np.array(self.C)

        self.input\_dim = input\_dim

        self.clusters = []

    def setRand(self):

        for i in range(self.cluster\_num):

            self.C.append(self.input\_data[np.random.randint(9000)])

    def setClusterNum(self,cluster\_num):

        self.cluster\_num = cluster\_num;

        self.\_\_init\_\_(self.cluster\_num, self.input\_dim)

    def setInputDimension(self , input\_dim):

        self.input\_dim = input\_dim

        self.\_\_init\_\_(self.cluster\_num , input\_dim)

    def dist(self,p1,p2):

        return np.sqrt(np.sum((p2-p1)\*\*2))

    def centerOfMass(self , data\_arr):

        s = 0;

        n = 0;

        # print(data\_arr)

        for i in data\_arr:

            s = s + np.float64(i)

            n = n+1;

        # print(s)

        if (n==0):

            return np.array([-1000,-1000,-1000])

        return (1.0/n)\*s;

    def train(self, input\_data):

        self.input\_data = input\_data

        self.setRand()

        for i in range(10):

            self.train\_(input\_data)

    def train\_(self , input\_data):

        self.lastClusters = self.clusters.copy

        self.clusters = [[] for i in range(self.cluster\_num)];

        for data in input\_data:

            d , last\_d = 0 , 100000000;

            nearest\_ci = 0;

            for ci in range(self.cluster\_num):

                d = self.dist(data, self.C[ci])

                if d < last\_d:

                    last\_d = d ;

                    nearest\_ci = ci ;

            self.clusters[nearest\_ci].append(data)

        self.clusters = np.array(self.clusters)

        for ci in range(self.cluster\_num):

            if len(self.clusters[ci]) > 0:

                self.C[ci] = self.centerOfMass(self.clusters[ci])

    def getModifiedData(self):

        output\_data = [];

        for data in self.input\_data:

            d , last\_d = 0 , 100000000;

            nearest\_ci = 0;

            for ci in range(self.cluster\_num):

                d = self.dist(data, self.C[ci])

                if d < last\_d:

                    last\_d = d ;

                    nearest\_ci = ci ;

            output\_data.append(self.C[nearest\_ci])

        return output\_data

    def getCenterOfCluster(self):

        return self.C

img = cv2.imread('test2.jpg')

# print(img.size)

kmean = Kmean(5,3);

# inp\_data = np.array([[0,0,0],[1,1,1.1],[1,1.1,1.2],[1.2,0.9,1],[1.1,1.1,0.9],[-0.1,0.1,-0.1],[-0.2,-0.1,0.1]]);

img\_vec = img.reshape(img.shape[0]\*img.shape[1] , 3);

print(img\_vec.shape)

# kmean.train(inp\_data)

# print(kmean.getCenterOfCluster())

# kmean.train(inp\_data)

# print(kmean.getCenterOfCluster())

kmean.train(img\_vec)

# print(np.array(kmean.getCenterOfCluster()))

# print(np.array(np.uint8(kmean.getCenterOfCluster())))

# print(kmean.getModifiedData())

out\_img = np.uint8(kmean.getModifiedData()).reshape(img.shape)

print(out\_img.shape)

cv2.imshow("img", img)

cv2.imshow("out\_img", out\_img)

cv2.waitKey(0)

در انتهای کد نیز با استفاده از این الگوریتم یک عکس رنگی به عکسی با حجم کمتر تبدیل کنیم :