گزارشکار آزمایش جلسه چهارم

Amirkabir University of Technology
(Tehran Polytechnic)

الگوریتم clustering و پیاده سازی الگوریتم Kmean

اعضای گزارش

عباس بدیعی

محمد حسین طیب زاده

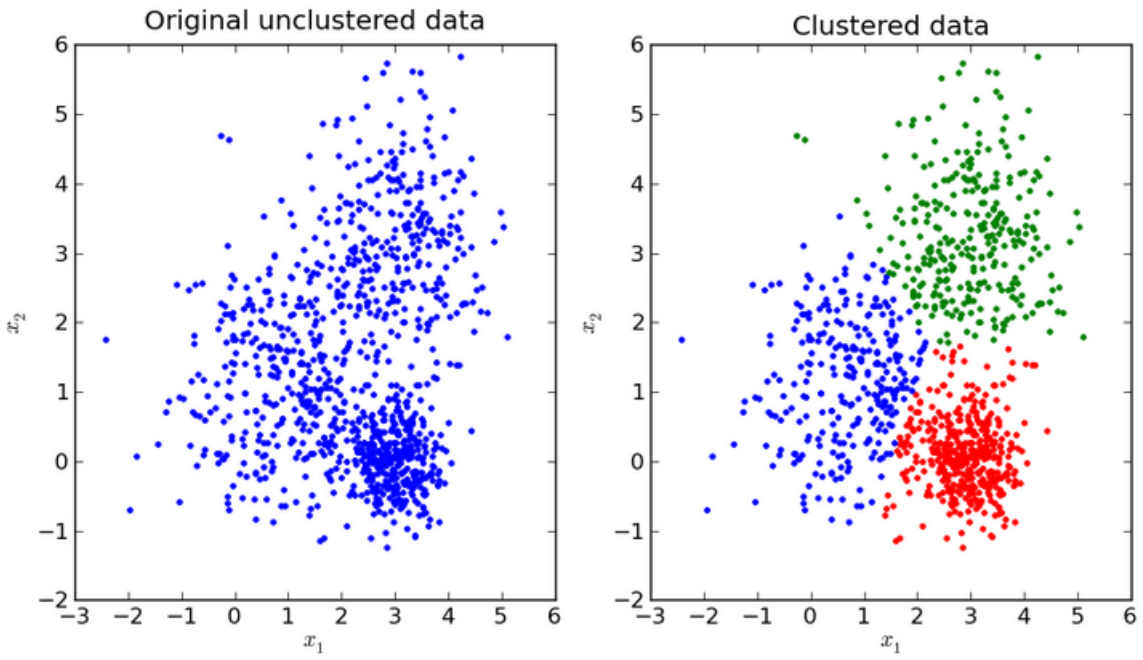
نویسنده گزارش : عباس بدیعی

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.

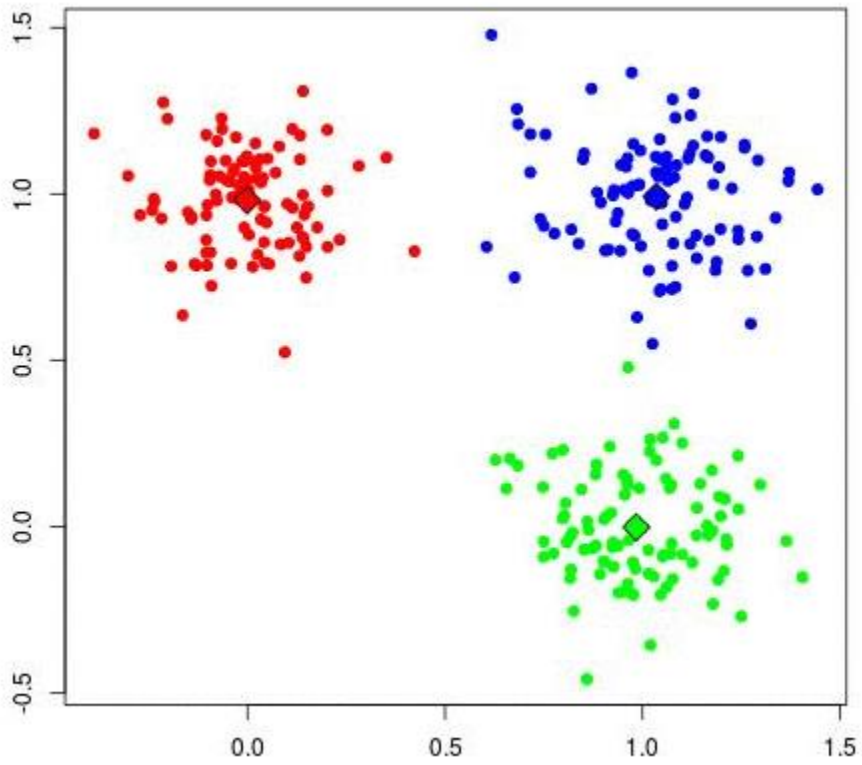
K-Means یک الگوریتم بسیار ساده است که داده‌ها را در K خوشه، گروه‌بندی می‌کند. نتیجه ای از خوشه‌بندی K-Means به شکل زیر است.



الگوریتمK-Means ، با فرض داشتن ورودی‌های x_n ، ...، x_3، x_2، x_1به 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 x_i 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

K مرکز خوشه (مرکزوار) به طور تصادفی انتخاب می‌شوند. فرض می‌شود که این مرکزوارها c_k ، ...، c_3، c_2، c_1 هستند. می‌توان گفت که:

$C = c_1, c_2, \ldots, c_k$

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.

$$\arg\min_{c_i \in C} dist(c_i, x)^2$$

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.

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$$

S_iمجموعه‌ای از همه نقاط تخصیص داده شده به خوشه i^{th} 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 :