دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی برق

گزارش آزمایش چهارم
مقدمه‌ای بر هوش محاسباتی
دکتر عبداللهی

نام و نام خانوادگی
کیان بهزاد

شماره دانشجویی
۹۵۲۳۰۱۷

پیاده سازی الگوریتم Kmean

قصد داریم به پیاده سازی این الگوریتم در محیط **پایتون** بپردازیم.
پس در ابتدا می‌بایست توابعی برای سهولت در پیاده سازی الگوریتم نهایی تعریف کنیم.

۱ - تابع فاصله اقلیدسی (مرتبه ۲)
این تابع با دریافت دو ورودی، فاصله اقلیدسی مرتبه ۲ آنها را محاسبه می‌کند.

```
def distance(center, input):
    sum = 0
    for i in range(len(input)):
        sum += (center[i]-input[i])**2
    return np.sqrt(sum)
```

۲ - تابع مرکز
این تابع با گرفتن مجموعه‌ای از ورودی‌ها (input ها) مرکز آنها را محاسبه می‌کند.

```
def make_center(input: np.ndarray):
    sum = np.zeros((1, len(input[0])))
    for i in range(len(input)):
        sum += input[i]
    return sum/len(input)
```

۳ - تابع ماسک
این تابع با گرفتن تمامی داده ها و همینطور مرکز کلاس ها، مشخص می‌کند هر داده به کدام مرکز نزدیکتر است تا آن کلاس برای آن داده در نظر گرفته شود.

```
def make_mask(data, centers):
    mask = np.zeros((len(data), 1))
    for i in range(len(data)):
        min_dist = 10000000
        index = -1
        for j in range(len(centers)):
            if distance(centers[j], data[i]) < min_dist:
                min_dist = distance(centers[j], data[i])
                index = j
        mask[i] = index
    return mask
```

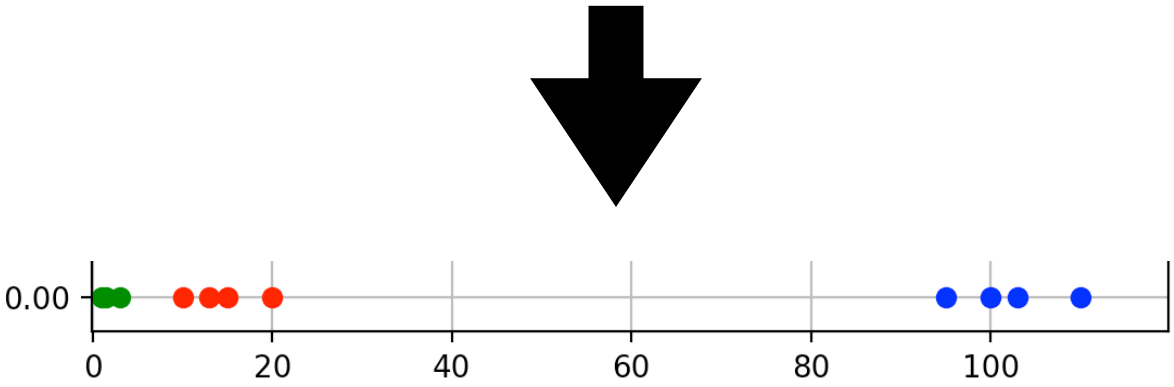
حال مطابق با الگوریتم داده شده و بصورت زیر عمل می‌کنیم.

```
def k_Mean(data, k):
    centers = np.zeros((k, len(data[0])))
    last_centers = np.zeros((k, len(data[0])))

    for i in range(k):
        centers[i] = np.random.randint(0, 50, len(data[0]))

    while True:
        mask = make_mask(data, centers)

        for i in range(k):
            my_list = []
            for j in range(len(mask)):
                if mask[j] == i:
                    my_list.append(data[j])
            #print(np.array(my_list))
            if len(my_list) == 0:
                return k_Mean(data, k)
            centers[i] = make_center(np.array(my_list))
        error = 0
        for i in range(len(centers)):
            error += distance(centers[i], last_centers[i])
        if error < 1:
            return mask, centers
        last_centers = centers
```

همانطور که مشاهده می‌شود، این الگوریتم بر ازای ورودی با تمام ابعاد کار می‌کند، در اینجا به منظور آزمایش کارکرد تابع را با ورودی های ۱ بعدی و ۲ بعدی خواهیم دید.

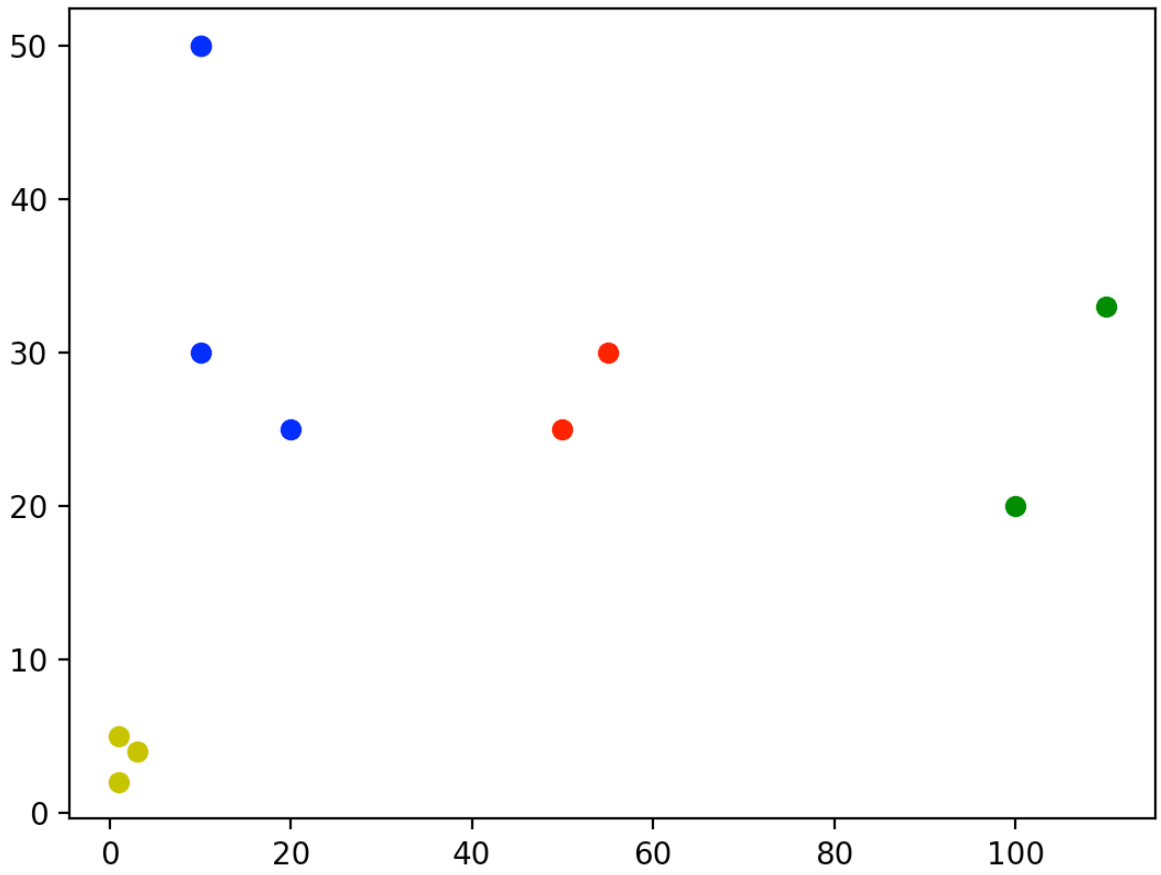
آزمایش اول - یک بعدی

```
data = np.array([[1], [3], [1.5], [10], [13], [100], [110], [103], [95], [20], [15]])
k = 3
mask, centers = k_Mean(data, k)
fig, ax = plt.subplots()
template = ['ro', 'bo', 'go', 'yo', 'r*', 'b*', 'g*', 'y*']
for i in range(len(data)):
    ax.plot(data[i][0], 0, template[int(mask[i])])
plt.grid()
plt.show()
```



آزمایش دوم - دو بعدی

```
data = np.array([[1, 2], [3, 4], [1, 5], [10, 30], [10, 50], [100, 20], [20, 25], [10, 50], [55, 30], [50, 25], [110, 33]])
k = 4
mask, centers = k_Mean(data, k)
fig, ax = plt.subplots()
template = ['ro', 'bo', 'go', 'yo', 'r*', 'b*', 'g*', 'y*']
for i in range(len(data)):
    ax.plot(data[i][0], data[i][1], template[int(mask[i])])
plt.show()
```



در ادامه می‌خواهیم از این الگوریتم استفاده کرده و با دادن یک عکس رنگی به تابع، آن را بصورت سیاه و سفید و همچنین با حجم کمتر درآوریم.

```
lena = cv2.imread('lena.jpg', 0)
lena = np.reshape(lena, (1, -1))
lena = np.transpose(lena)

k = 5
mask, centers = k_Mean(lena,k)
for i in range(len(lena)):
  lena[i] = centers[int(mask[i])]

lena = np.transpose(lena)
lena = np.reshape(lena, (446, 651))
cv2.imwrite("mylena.png", lena)
cv2.imshow(lena)
```

تمرین

۱. با توجه به نمودار خطا بر حسب تعداد کلاس ها که در قسمت ارزیابی الگوریتم رسم کرده اید حالت بهینه الگوریتم در چه تعداد کلاس رخ می دهد؟

جواب قطعی برای تعیین تعداد بهینه خوشه ها در الگوریتم های خوشه بندی وجود ندارد! تعیین تعداد بهینه خوشه ها از سویی تابع معیار در نظر گرفته شده به عنوان معیار شباهت بین داده ها در الگوریتم خوشه بندی است و از سوی دیگر تابع پارامترهای تنظیم شده در الگوریتم موردنظر است. یک روش ساده و رایج بین کاربران، بررسی دندروگرام حاصل از یک الگوریتم خوشه بندی سلسله مراتبی مانند Agglomerative است تا از این طریق، در صورت حصول یک مقدار خاص برای خوشه ها از آن تعداد به عنوان مقدار k در الگوریتم های خوشه بندی مبتنی بر تقسیم استفاده شود. اما این روش، فاقد پشتوانه‌ی ریاضی و اثباتی قوی است.

۲. به نظر شما چرا الگوریتم توانایی دسته بندی صحیح داده ها را در قسمت محدودیت های $k-mean$ نداشت؟ برای بهبود الگوریتم چه پیشنهادی دارید؟

در الگوریتم k mean در صورتیکه مقادیر اولیه مرکز کلاس‌ها به خوبی انتخاب نشوند ممکن است هیچ data یی به آن محول نشود و درراقع یکی از کلاس‌ها از بین برود.

۳. الگوریتم $k-mean$ پیاده سازی شده را در محیط متلب مجددا پیاده سازی کنید.

```
function y=kMeansCluster(m,k,isRand)
    % initial value of centroid
    if isRand
        p = randperm(size(m,1));      % random initialization
        for i=1:k
            c(i,:)=m(p(i),:);
        end
    else
        for i=1:k
           c(i,:)=m(i,:)        % sequential initialization
        end
    end

    temp=zeros(maxRow,1);   % initialize as zero vector

    while 1
        d=DistMatrix(m,c);  % calculate objcets-centroid distances
        [~,g]=min(d,[],2);  % find group matrix g
        if g==temp
            break;          % stop the iteration
        else
            temp=g;         % copy group matrix to temporary variable
        end
        for i=1:k
            f=find(g==i);
            if f            % only compute centroid if f is not empty
                c(i,:)=mean(m(g==i,:),1);
            end
        end
    end

    y=[m,g];

end
```

گزارش و کد را می‌توانید در repository زیر ببینید(Az_jalase3)

https://github.com/kianbehzad/computational-intelligence