Amirkabir University of Technology
(Tehran Polytechnic)

Electrical Engineering Department

گزارشکار آزمایشگاه مقدمه‌ای بر هوش محاسباتی

آزمایش شماره‌های 5

K-Mean Algorithm

نام استاد: محمدحسین امینی

نام دانشجو: محمدعرشیا ثمودی - 9723021

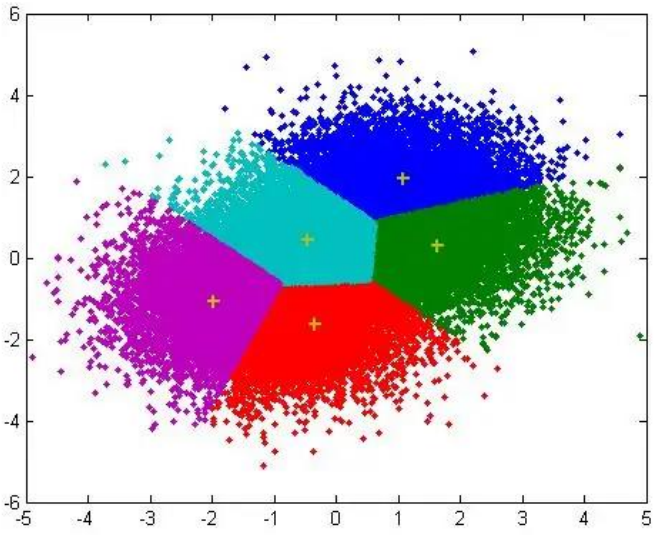
آزمایش پنجم

هدف آزمایش: پیاده‌سازی الگوریتم K-mean در پایتون

شرح آزمایش:

1. مقدمه

روش‌ها و الگوریتم‌های متعددی برای تبدیل اشیاء به گروه‌های همشکل یا مشابه وجود دارد. الگوریتم k-میانگین یکی از ساده‌ترین و محبوب‌ترین الگوریتم‌هایی است که در «داده‌کاوی» (Data Mining) بخصوص در حوزه «یادگیری نظارت نشده» (Unsupervised Learning) به کار می‌رود.
در خوشه‌بندی k-میانگین از بهینه‌سازی یک تابع هدف (Object Function) استفاده می‌شود. پاسخ‌های حاصل از خوشه‌بندی در این روش، ممکن است به کمک کمینه‌سازی (Minimization) یا بیشینه‌سازی (Maximization) تابع هدف صورت گیرد. به این معنی که اگر ملاک «میزان فاصله» (Distance Measure) بین اشیاء باشد، تابع هدف براساس کمینه‌سازی خواهد بود پاسخ عملیات خوشه‌بندی، پیدا کردن خوشه‌هایی است که فاصله بین اشیاء هر خوشه کمینه باشد. در مقابل، اگر از تابع مشابهت (Dissimilarity Function) برای اندازه‌گیری مشابهت اشیاء استفاده شود، تابع هدف را طوری انتخاب می‌کنند که پاسخ خوشه‌بندی مقدار آن را در هر خوشه بیشینه کند.



شکل 1. مدل همسایگی

مدل همسایگی در پایتون

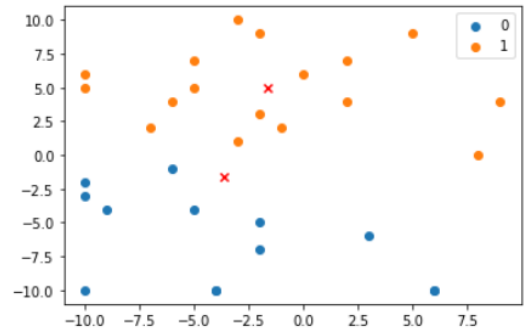
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
```

```
[34] class KMeans:
        def __init__(self,k):
          self.k = k

        def fit(self,x):

          #Random Centroids
          centroids = x[np.random.choice(x.shape[0], self.k,replace= False)]

          #Clusters's Distance From 2D-points
          distance = cdist(x, centroids ,'euclidean')

          #Label Assignment to K Clusters
          label = np.array([np.argmin(i) for i in distance])

          iterations = 100
          for i in range(iterations):
            centroids = []
            for icluster in range(self.k):
              #Updating Centroids by taking mean of Cluster it belongs to
              centroids.append(np.array(x)[label == icluster].mean(axis=0))

            distances = cdist(x, centroids ,'euclidean')
            label = np.array([np.argmin(i) for i in distances])

          return centroids,label
```

طبق مدل بالا، برای داده‌های تصادفی x تعداد همسایگی k = 2 را قرار دادیم. طبق شکل پایین، نوع داده‌ها و همچنین ساختار مرکزی هر خوشه مشخص گردیده است.

```python
# x = np.array([[1,1],[2,0.5],[3,0.4],[10,15],[11.5,14],[9.4,8],[4.5,2.3],[1.3,2.3],[-4.6,6]])
x = np.random.randint(-10, 11, size=(30, 2))


km = KMeans(k = 2)
centroids,label = km.fit(x)

print(centroids)
print(label)

# Plotting
K = len(centroids)
K_label = np.arange(K)

for i in K_label:
  plt.scatter(x[label == i, 0] , x[label == i, 1] , label = i)


plt.scatter([centroids[0][0], centroids[1][0]], [centroids[1][0], centroids[1][1]], c = 'r', marker = 'x')
plt.legend()
plt.show()
```

```
[array([-3.61538462, -6.30769231]), array([-1.64705882,  4.94117647])]
[0 1 1 0 0 1 0 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1 1 1]
```

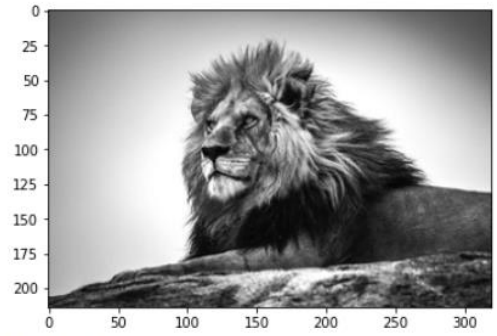


کم کردن حجم عکس با استفاده از خوشه‌بندی

با استفاده از کتابخانه PIL یک عکس دلخواه در Colab بارگذاری می‌شود. به منظور سادگی کار با آرایه خوانده شده از عکس، آن به فرمت عددی Float درآورده می‌شود.

سپس تمام آرایه عکس به یک آرایه سطری تبدیل می‌شود تا الگوریتم خوشه میانگین بر روی آن پیاده شود. این الگوریتم با K = 20 پیاده‌سازی می‌شود تا عکس ساده‌تر گردد یعنی توریع رنگ در آن به تعداد K تقسیم می‌شود.

```
from PIL import Image

img = Image.open("/content/lion.jpg").convert('L')
img_arr=np.array(img)
plt.figure()
plt.imshow(img_arr,cmap='gray')
plt.show()
imgshape=img_arr.shape
img_arr =np.reshape(img_arr,(1,-1))
img_arr = np.transpose(img_arr)
print(img_arr.shape)
one_dim=np.reshape(img_arr,(img_arr.shape[0]*img_arr.shape[1],1))
one_dim = one_dim.astype('float64')


km = KMeans(k = 20)
center,lab = km.fit(one_dim)
remake=np.reshape(lab,imgshape)

plt.figure()
plt.imshow(remake,cmap='gray')
plt.show()
```




```
(68480, 1)
```