آزمایش چهارم پیاده سازی شبکه عصبی K-mean

اعضای گروه:

ملیکا صالحیان ، محدثه قاسم مهرابی، مریم حیدری

خوشه‌بندی k میانگین

روش‌ها و الگوریتم‌های متعددی برای تبدیل اشیاء به گروه‌های هم شکل یا مشابه وجود دارد. الگوریتم k-میانگین یکی از ساده‌ترین و محبوب‌ترین الگوریتم‌هایی است که در داده‌کاوی (Data Mining) به خصوص در حوزه یادگیری نظارت نشده (Unsupervised Learning) به کار می‌رود.

معمولا در حالت چند متغیره، باید از ویژگی‌های مختلف اشیا به منظور طبقه‌بندی و خوشه کردن آن‌ها استفاده کرد. به این ترتیب با داده‌های چند بعدی سروکار داریم که معمولا به هر بعد از آن، ویژگی یا خصوصیت گفته می‌شود. با توجه به این موضوع، استفاده از توابع فاصله مختلف در این جا مطرح می‌شود.

الگوریتم خوشه‌بندی-k میانگین از گروه روش‌های خوشه‌بندی تفکیکی (Partitioning Clustering) محسوب می شود.

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.

به نقاط مرکزی هر خوشه مرکز (Centroid) گفته می‌شود. ممکن است این نقطه یکی از مشاهدات یا غیر از آن‌ها باشد. در الگوریتم ، k مشاهده یا مقدار رندوم به عنوان مرکز خوشه‌ها (Centroids) در مرحله اول انتخاب شده‌اند ولی در مراحل بعدی، مقدار میانگین هر خوشه نقش مرکز را بازی می‌کند.

الگوریتم استاندارد

رایج‌ترین الگوریتم کی-میانگین با استفاده از یک تکرار شونده پالایش کار می‌کند.

ابتدا میانگین هر دسته را به صورت تصادفی مقداردهی می‌کنیم.

سپس، این دو مرحله پایین را به تناوب چندین بار اجرا می‌کنیم تا میانگین‌ها به یک ثبات کافی برسند و تغییر چندانی نکنند.

۱. از میانگین‌ها دسته می‌سازیم، دسته i ام در هر بار تکرار ، تمام داده‌هایی هستند که از لحاظ اقلیدسی کمترین فاصله را با میانگین i ام دارند.
۲. حال میانگین‌ها را بر اساس این دسته های جدید به روز می‌کنیم.
۳. در نهایت میانگین‌های مرحله آخر هر دسته به عنوان مرکز دسته جدید انتخاب می شود.

شرط توقف الگوریتم:

۱. مراکز در محاسبه جدید تغییر نکند.
۲. داده‌ها در همان خوشه قبلی باقی بمانند و تغییر خوشه ندهند.
۳. به حداکثر تعداد تکرار الگوریتم رسیده باشد.

کاربرد الگوریتم k-mean

خوشه‌بندی در حوزه‌های مختلفی مورد استفاده قرار می‌گیرد. به عنوان مثال: خوشه‌بندی مشتریان براساس سابقه خرید، براساس علایق آن‌ها، براساس فعالیت مشتریان در وب‌سایت‌ها و تقسیم‌بندی تصاویر، جداسازی صداها و...

خوشه‌بندی کا-مین نوعی از یادگیری بدون نظارت است و زمانی استفاده می‌شود که داده‌هایی بدون برچسب در اختیار داشته باشیم. هدف این خوشه‌بندی، پیداکردن بهترین گروه در داده است و k در آن

تعداد خوشه‌ها را تعیین می‌کند. داده‌ها بر اساس میزان شباهت در خوشه‌ها قرار می‌گیرد. به صورتی که در نهایت داده‌ها با بیشترین شباهت در یک گروه قرار می‌گیرند و کمترین شباهت را با سایر گروه‌ها دارند.

توضیحات کد پایتون k-mean :

در ابتدا کتابخانه‌های لازم برای استفاده در کد معرفی‌شده‌اند.

```
from numpy.linalg import norm
import numpy as np
import pandas as pd
import math as m
import matplotlib.pyplot as plt
```

در کتابخانه تعریف شده مراکز دسته ها متناسب با داده های ورودی برگردانده می شود.

```
#finding the number of class
def CNUM(x,y):
  s=0
  for i in range (k):
    A = float(x - center_x[i])
    B = float(y - center_y[i])
    #dist[i]= m.sqrt(A**2 + B**2)
    dist[i]= (A**2 + B**2)
    if dist[i] == min(dist):
      s = i
  return s
```

Dataset1.csv به منظور استخراج داده ها در کد پایتون بارگذاری و در متغیر data ذخیره می شوند. سایر متغیرهای لازم نیز متناسب با کد تعریف شده اند.

```
# Data
k= 10
data_number = input('Enter the number of data')
data_number = int(data_number)
center_y_new = np.empty(k)
center_x_new = np.empty(k)
center_x = np.empty(k)
```

```
center_y = np.empty(k)
dist = np.empty(k)
CLASS = [[]for _ in range (k)]
E22= [0 for _ in range (k)]
Error=[[]for _ in range (k)]
ErrorT=[[]for _ in range (k)]
A = [[]for _ in range (k)]
data = pd.read_csv("Dataset1.csv")
flag = 1
error_num= 10
iteration = 1000
```

در این قسمت به تعداد دفعات مطلوب تعداد کلاس ها از کاربر دریافت شده و داده ها دسته بندی می شوند و در نهایت میزان خطا محاسبه می شود.

ابتدا مراکز دسته ها مقدار دهی رندوم اولیه می شوند.سپس تابع **classify** فراخوانی می شود که با فراخوانی تابع **CNUM** فاصله هر داده تا مرکز دسته را محاسبه می کند و شماره دسته با کمترین فاصله را برگردانده و در نهایت در دسته متناظر در متغیر **A** ذخیره می کند.

در مرحله بعد میانگین هریک از کلاس ها محاسبه می شود و در صورتی که خطای مقادیر جدید با مرکز دسته های سابق بیشتر از میزان مشخصی باشد جایگزین مرکز دسته های قبلی می شود و سپس داده ها مجددا طبقه بندی می شوند. در صورتی که مجموع مربعات خطاهای مراکز دسته های جدید از قبلی از مقدار مشخصی کمتر باشد الگوریتم به مقدار نهایی خود همگرا شده و از حلقه خارج می شود.

```
for s in range(error_num):
 k = input('Enter the number of class')
 k = int(k)

# Center initializing
 for i in range (k):
```

```
    center_x[i] = np.random.uniform(size=(1,1))
    center_y[i] = np.random.uniform(size=(1,1))

#input data & classification
def classify(x):
      A = [[]for _ in range (k)]
    for j in range (data_number):
      X = data.iloc[[j],[0]].values
      Y = data.iloc[[j],[1]].values
      X = float(X)
      Y = float(Y)
      index = CNUM(X,Y)
      A[index].append([float(X),Y])
    return A

CLASS= classify(data_number)

#Finding new centers
for o in range (k):
    if CLASS[o] != []:
      m =np.mean(CLASS[o], axis=0)
      #print(m)
      center_x_new[o] = float(m[0])
      center_y_new[o] = float(m[1])
      #print(center_y_new[o])

 #Reclassification by new centers
for j in range (iteration):
   while flag:
     for i in range (k):
       C = center_x_new[i]- center_x[i]
       D = center_y_new[i]-center_y[i]
       C = C.astype('float')
       D = D.astype('float')
```

```
    E2 = C**2+D**2
    if E2 > (0.001)**2:
      center_x[i] = center_x_new[i]
      center_y[i] = center_y_new[i]
  CLASS= classify(data_number)

  #finding distance between the new center & old one
  for m in range (k):
    C2 = center_x_new[m]- center_x[m]
    D2 = center_y_new[m]-center_y[m]
    C2 = C2.astype('float')
    D2 = D2.astype('float')
    E22[m] = C**2+D**2
  if (np.sum(E22) < (0.001)**2):
    flag = 0
```

محاسبه و رسم خطا به ازای تعداد دسته های از یک تا ده:

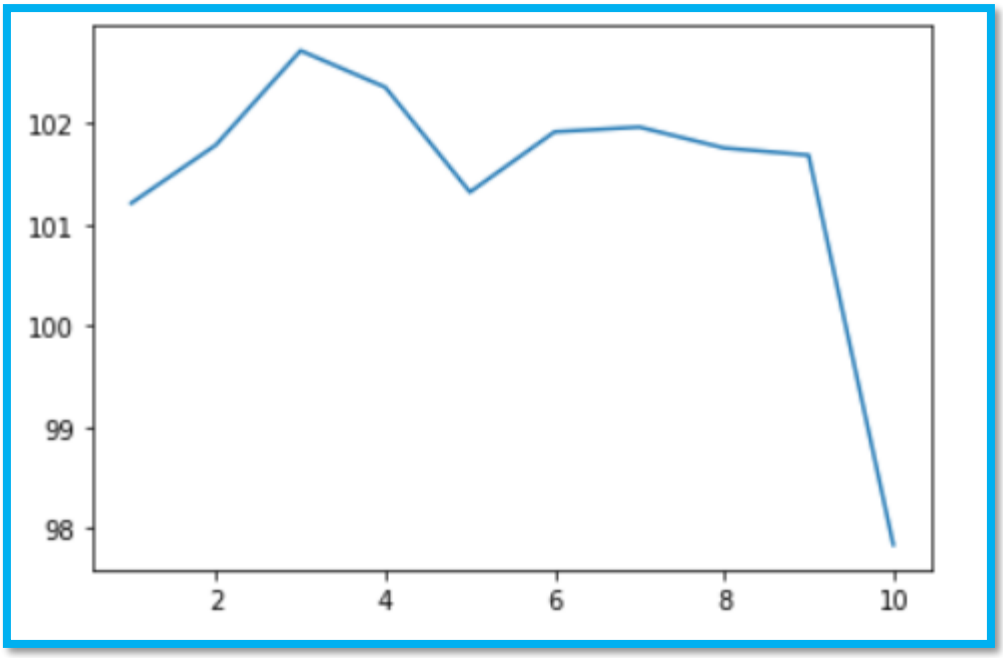
```
 E =0
  for a in range (k):
    for b in range (len(CLASS[a])):
      C3 = CLASS[a][b][0]- center_x[a]
      D3 = CLASS[a][b][1]-center_y[a]
      C3 = C3.astype('float')
      D3 = D3.astype('float')
      E =E + C3**2+D3**2
  E = E/(data_number)
#Error = np.mean(Error)

  ErrorT[s]= E

print(ErrorT)
y =[1,2,3,4,5,6,7,8,9,10]
```

```
plt.plot(y,ErrorT)
plt.show()
```

در نهایت تابع خطا بر اساس مجموع مربعات خطاهای هر بار تکرار به صورت زیر رسم می شود:



مشاهده می شود که با افزایش تعداد کلاس ها خطا کاهش می یابد و در تعداد ۱۰ به مقدار مینیمم خود می رسد.

تمرین- کد متلب پیاده سازی شبکه عصبی RBF

```
%%computational Lab 4 K-mean code%%
clear all
clc
close all
%% Reading data from Dataset1 %%
TX = readtable('Dataset1.csv','ReadVariableNames',true);
X  = TX.X;
Y  = TX.Y;
%% initializing %%
k = 50; %limit of the number of classes that can be chosen by user
g = 1;
error_num= 10; %the number of trial
iteration = 10; %the limit of updationg
E = zeros(error_num,k);
F = zeros(1,error_num);
center_x =zeros(1,k);
center_y =zeros(1,k);
dist = zeros(1,k);
%% initializing centers of classes by random numbers
for i = 1:k
    center_x(1,i) = rand();
    center_y(1,i) = rand();
end
center_x
%% main %%
for s = 1:error_num
  k = input('Enter the number of class');
  center_y_new = zeros(1,k);
  center_x_new =zeros(1,k);
%input data & classification
  for j = 1:length(X)
      index = CNUM(k,X(j),Y(j),center_x(1,:),center_y(1,:));
```

```
    CLASS_X(index,j)= X(j);
    CLASS_Y(index,j) = Y(j);
 end
%Finding new centers
 flag = 1;

    while flag
        for l = 1 : iteration %to limit the number of updating
        m_1 = 0;
        m_2 = 0;
        sum_1 = 0
        sum_2 = 0
        for o = 1:k
            for j = 1:length(X)
                if CLASS_X(o,j) ~= 0
                    m_1 = m_1+1;
                    sum_1 = sum_1 +CLASS_X(o,j);
                end
            end
            center_x_new(1,o) = sum_1 / m_1
            for j = 1:length(X)
                if CLASS_Y(o,j) ~= 0
                    m_2=m_2+1;
                    sum_2 = sum_2 +CLASS_Y(o,j);
                end
            end
            center_y_new(1,o) = sum_2 / m_2
        end
% input data & classification
        for j = 1:length(X)
            index = CNUM(k,X(j),Y(j),center_x_new(1,:),center_y_new(1,:));
            CLASS_X(index,j)= X(j);
            CLASS_Y(index,j) = Y(j);
        end
        for i = 1:k
            T=center_x(1,i) - center_x_new(1,i)
```

```
            D=center_y(1,i) - center_y_new(1,i)
            if (sqrt(T^2+D^2) < 10)
                flag = 0
            end
        end
    end
 end
 error_total = zeros(1,k);
 for a = 1:k
    n = 0;
    for b=1:length(X)
        if CLASS_X(a,b) = 0 && CLASS_Y(a,b) =0
            error_sum_1= center_x_new(1,a)-CLASS_X(a,b)
            error_sum_2= center_y_new(1,a)-CLASS_Y(a,b)
            error_total(1,a) = sqrt(error_sum_1^2+error_sum_2^2) +error_total(1,a)
            n=n+1;
        end
        E(s,a)= error_total(1,a)/n
        error_total = zeros(1,k);
    end
    F(1,g)= sum(E(s))/k
 end
 g=g+1;
end
F

plot(1:10,F)
```

نتیجه کد متلب به ازای تعداد دسته از ۱ تا ۱۰: