به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس سیستم‌های هوشمند

تمرین شماره 4

نام و نام خانوادگی : محمدرضا بختیاری

شماره دانشجویی : 810197468

دی 1400

فهرست سوالات

سوال 1 : تحلیلی 3

الف: خوشه بندی با روش کا-میانگین 3

ب: خوشه بندی سلسله مراتبی 5

سوال 2 : پیاده سازی الگوریتم خوشه بندی 6

الف : تاثیر تعداد خوشه ها 6

ب : تاثیر تکرار آزمایش 11

سوال 4 : مقدمات احتمال 12

الف : سوالات تحلیلی 12

الف-1 : 12

الف -2 : 13

الف-2-1 : 13

الف-2-2 : 13

الف -3 : 14

ب : سوالات شبیه سازی 14

ب-1 : 14

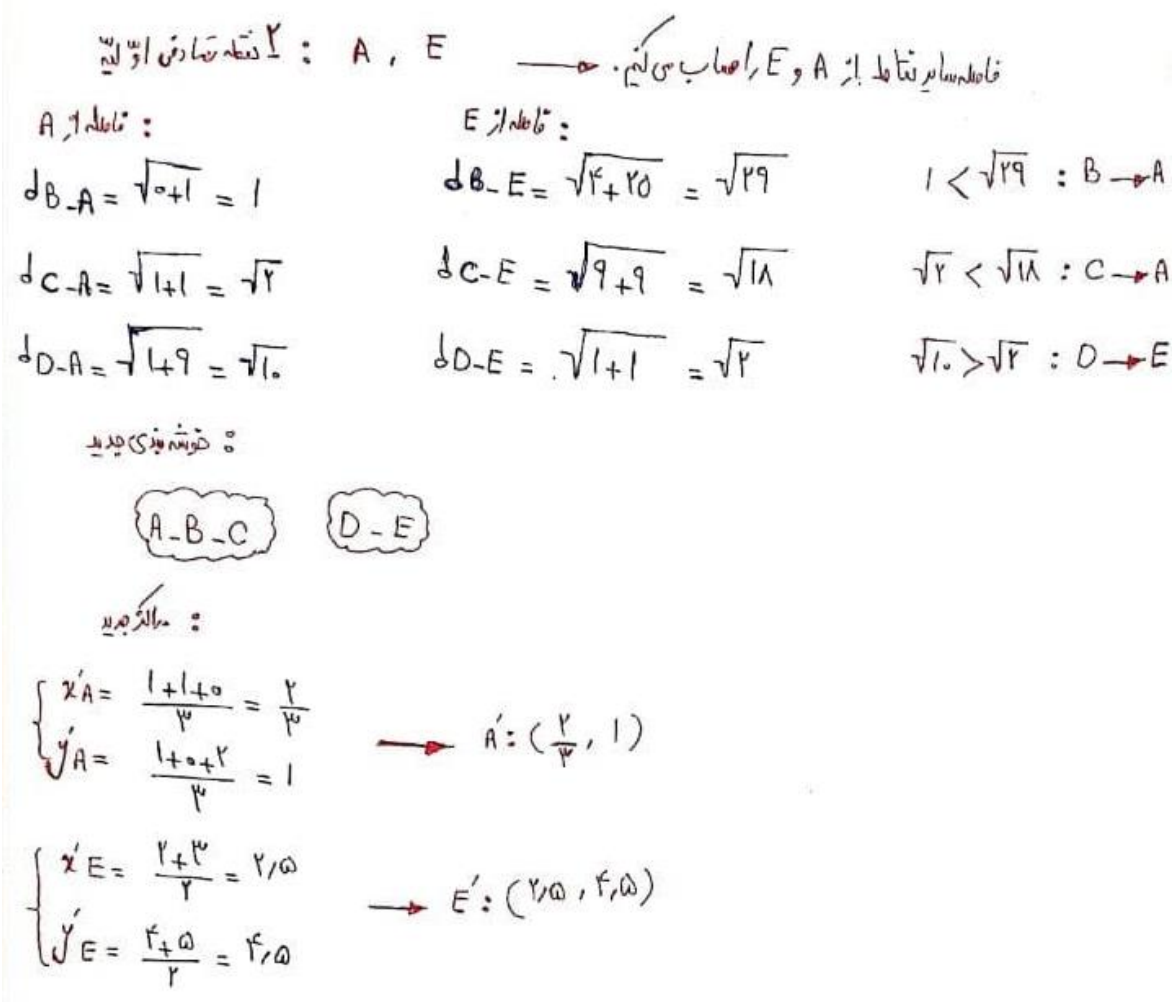
پیوست: 16

سوال 1 : تحلیلی

با استفاده از دو روش کا-میانگین و سلسله مراتبی , به صورت دستی , الگوریتم خوشه بندی را پیاده سازی می کنیم .

الف: خوشه بندی با روش کا-میانگین

ابتدا به صورت تصادفی دو نقطه اولیه A , E را به عنوان مرکز دو خوشه در نظر می گیریم . سپس با محاسبه ی فاصله ی سایر نقاط از این دو نقطه , نقاط با فاصله کمتر را به خوشه مورد نظر اختصاص داده و در انتها مرکز خوشه را به روز رسانی می کنیم و این مراحل را آنقدر ادامه می دهیم تا تغییر در مرکز خوشه , تغییری در خوشه ها ایجاد نکند .



فاصله از A' :

$d_{A-A'} = \sqrt{\frac{1}{9} + 0} = \frac{1}{3}$

$d_{B-A'} = \sqrt{\frac{1}{9} + 1} = \frac{\sqrt{10}}{3}$

$d_{C-A'} = \sqrt{\frac{4}{9} + 1} = \frac{\sqrt{13}}{3}$

$d_{D-A'} = \sqrt{\frac{16}{9} + 9} = \frac{\sqrt{97}}{3}$

$d_{E-A'} = \sqrt{\frac{49}{9} + 16} = \frac{\sqrt{193}}{3}$

فاصله از E' :

$d_{A-E'} = \sqrt{\frac{9}{4} + \frac{49}{4}} = \frac{\sqrt{58}}{2}$ $\quad \frac{1}{3} < \frac{\sqrt{58}}{2} : A \to A'$

$d_{B-E'} = \sqrt{\frac{9}{4} + \frac{81}{4}} = \frac{\sqrt{90}}{2}$ $\quad \frac{\sqrt{10}}{3} < \frac{\sqrt{90}}{2} : B \to A'$

$d_{C-E'} = \sqrt{\frac{25}{4} + \frac{25}{4}} = \frac{\sqrt{50}}{2}$ $\quad \frac{\sqrt{13}}{3} < \frac{\sqrt{50}}{2} : C \to A'$

$d_{D-E'} = \sqrt{\frac{1}{4} + \frac{1}{4}} = \frac{\sqrt{2}}{2}$ $\quad \frac{\sqrt{97}}{3} > \frac{\sqrt{2}}{2} : D \to E'$

$d_{E-E'} = \sqrt{\frac{1}{4} + \frac{1}{4}} = \frac{\sqrt{2}}{2}$ $\quad \frac{\sqrt{193}}{3} > \frac{\sqrt{2}}{2} : E \to E'$

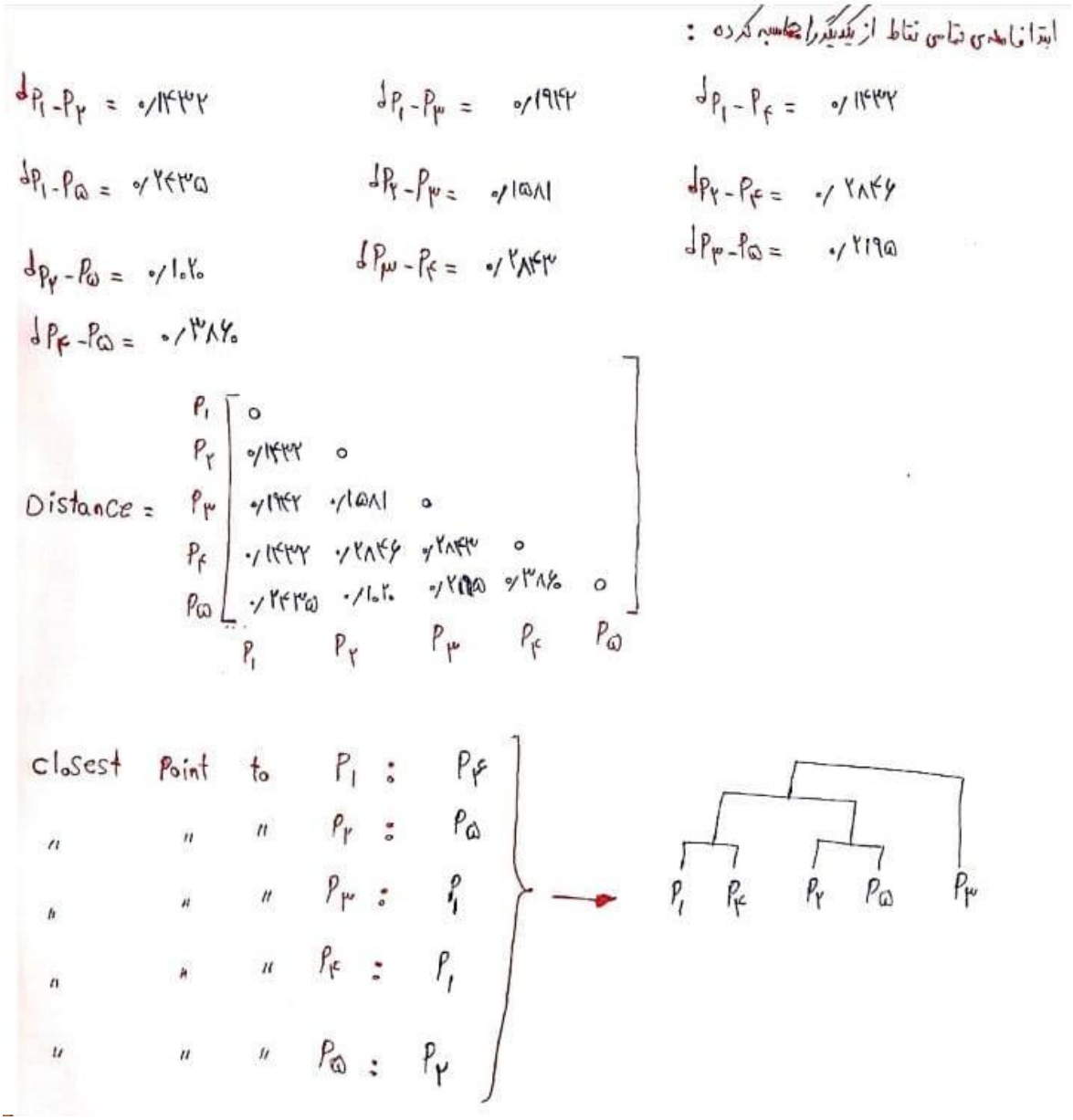
خوشه‌بندی جدید :

(A_B_C) (D_E)

متاسفانه می‌گیم خوشه بندی جدید تغییری نکرده ← به پایداری رسیده‌ایم .

ب: خوشه بندی سلسله مراتبی

ابتدا فاصله تمامی نقاط از یکدیگر را محاسبه کرده و برای مقایسه ی راحت تر در ماتریسی مربعی ذخیره می کنیم . سپس نزدیک ترین دو نقطه به یکدیگر را در یک شاخه قرار داده و نمودار درختی آن را رسم می کنیم .



سوال 2 : پیاده سازی الگوریتم خوشه بندی

در این سوال قصد پیاده سازی الگوریتم خوشه بندی با استفاده از کا-میانگین[1] و بررسی تاثیر فراپارامتر[2] تعداد خوشه ها و همچنین تاثیر تکرار آزمایش بر نتیجه خروجی را داریم .

الف : تاثیر تعداد خوشه ها

پیاده سازی را به این صورت انجام می دهیم که ابتدا به تعداد خوشه ها , به صورت کاملا تصادفی از بین دادگان موجود , مرکز خوشه انتخاب می کنیم .

```
Number_Of_Clusters = 5 ;                                    % # Number Of Clusters
iteration = 150 ;                                           % # Number Of iteration
Index_Of_Center_Of_Clusters = randi([1 150],1,Number_Of_Clusters) ;    % Randomly choose center
Center_Of_Clusters = [] ;
Index_Of_Data = [] ;

for Index = 1:Number_Of_Clusters
   Center_Of_Clusters(Index , 1) = iris(Index_Of_Center_Of_Clusters(Index) , 1) ;    % Index 1 Clusters
   Center_Of_Clusters(Index , 2) = iris(Index_Of_Center_Of_Clusters(Index) , 2) ;    % Index 2 Clusters
   Center_Of_Clusters(Index , 3) = iris(Index_Of_Center_Of_Clusters(Index) , 3) ;    % Index 3 Clusters
   Center_Of_Clusters(Index , 4) = iris(Index_Of_Center_Of_Clusters(Index) , 4) ;    % Index 4 Clusters
   Center_Of_Clusters(Index , 5) = iris(Index_Of_Center_Of_Clusters(Index) , 5) ;    % Index 5 Clusters
end
```

در ادامه فاصله هر داده را از این مراکز با استفاده از نرم اقلیدسی[3] محاسبه کرده و داده را به خوشه ای اختصاص می دهیم که کمترین فاصله را با آن دارد .

```
   Index_Of_Data = [] ;
   Min_Distance = [] ;
   Min_Distance_Index = 0 ;
   Min_Value = 0 ;

   Data = 1 ;
   for Data = 1:150

      Min_Distance = [] ;
      Min_Distance_Index = 0 ;
      Min_Value = 0 ;

      Counter = 1 ;
      for Counter = 1:Number_Of_Clusters
         Min_Distance(end+1) = Distance(iris(Data , :) , Center_Of_Clusters(Counter , :)) ;
      end

      [Min_Value,Min_Distance_Index] = min(Min_Distance) ;
      Index_Of_Data(Min_Distance_Index , end+1) = Data ;
```

[1] K-means
[2] Hyperparameter
[3] Euclidean norm

پس از اختصاص یافتن هر داده به خوشه مورد نظر , مراکز خوشه ها را به روز رسانی می کنیم . به این صورت که مرکز هر خوشه را برابر با میانگین داده های درون خوشه در نظر می گیریم .

```
  for item = 1:Number_Of_Clusters

    s1 = [] ;      Len1 = 0 ;
    s1 = size(Index_Of_Data) ;
    Len1 = s1(1) ;

    if item <= Len1

      List = [] ;    s = [] ;    Len = 0 ;
      y1 = 0 ;   y2 = 0 ;   y3 = 0 ;   y4 = 0 ;   y5 = 0 ;
      List = Index_Of_Data(item , :) ;
      List(List==0) = [];
      s = size(List) ;
      Len = s(2) ;
      i = 1 ;

      for i = 1:Len
        y1 = y1 + iris(i,1) ;
        y2 = y2 + iris(i,2) ;
        y3 = y3 + iris(i,3) ;
        y4 = y4 + iris(i,4) ;
        y5 = y5 + iris(i,5) ;
      end
                                                  % Update Center of Clusters
      if Len ~= 0
        Center_Of_Clusters(item , 1) = y1 / Len ;
        Center_Of_Clusters(item , 2) = y2 / Len ;
        Center_Of_Clusters(item , 3) = y3 / Len ;
        Center_Of_Clusters(item , 4) = y4 / Len ;
        Center_Of_Clusters(item , 5) = y5 / Len ;
      end

    end

  end
```

حال به مقایسه ی نتایج به دست آمده به ازای تعداد خوشه های مختلف می پردازیم . معیار مقایسه را نسبت شباهت درونی به شباهت بیرونی در نظر می گیریم . به این صورت که شباهت درونی برابر است با مجموع فواصل داده های درون یک خوشه از مرکز همان خوشه و شباهت بیرونی برابر است با مجموع فواصل هر داده از مرکز خوشه های دیگر .

```
% Internal similarity
function Distance = Internal_similarity(Data , Center)
  for j = 1:size(Data)
    Distance = Distance + norm(Data(j) - Center) ;
  end
end
```

```
% External similarity
function Distance = External similarity(Data , Centers)
   for k = 1:size(Centers)
      Distance = Distance + norm(Data - Centers(k)) ;
   end
end
```

$$\text{if } \mu = \frac{\text{Internal similarity}}{\text{External similarity}} \rightarrow \quad \mu_1 = 0.0838 \ , \ \mu_2 = 0.0363 \ , \ \mu_3 = 0.0091$$

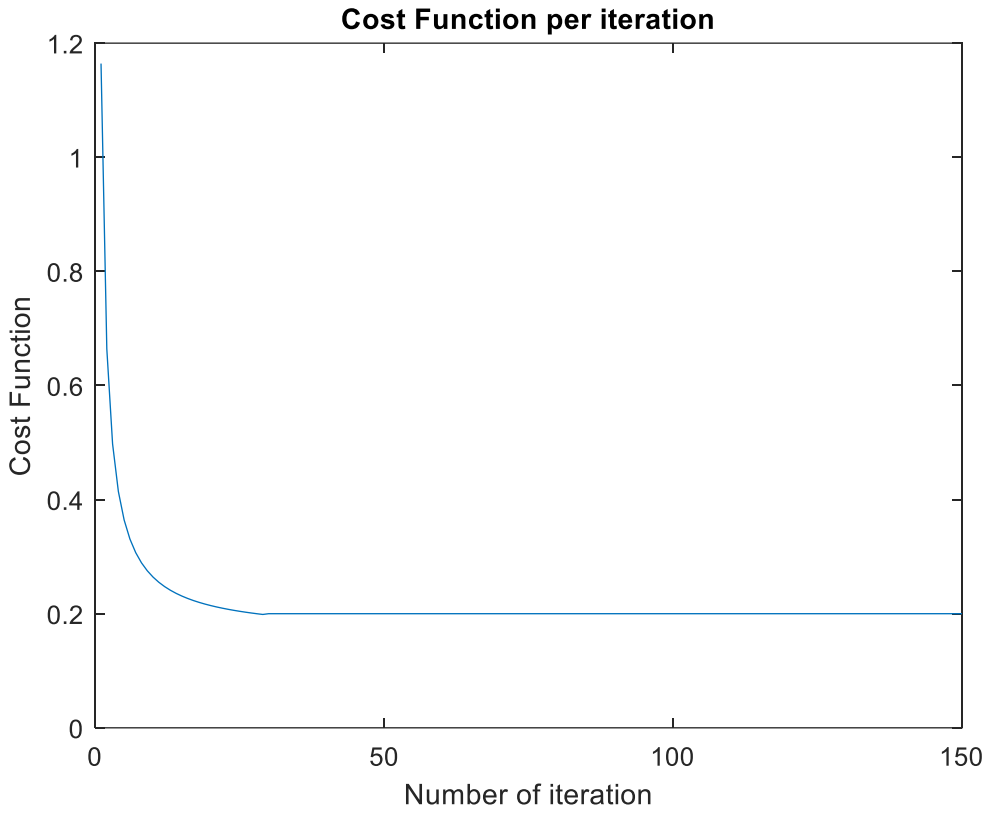
که به ترتیب μ_1 , μ_2 و μ_3 برابر است با نسبت شباهت درونی به شباهت بیرونی برای 5 , 10 و 20 خوشه . این نسبت هر چه کمتر باشد نشان دهنده ی نتیجه بهتری می باشد , زیرا نشان می دهد داده های درون یک خوشه به هم نزدیک تر و داده های یک خوشه با خوشه ی دیگر دور تر از هم می باشد . که در اینجا هنگامی که از 20 خوشه استفاده می کنیم نسبت کمتری می گیریم و در نتیجه نتیجه بهتری خواهیم داشت .

در اینجا تابع هزینه[1] را برابر با مجموع فاصله هر داده از مرکز خوشه ای که در آن قرار دارد در نظر می گیریم :

$$\mathrm{Cost}(C) = \sum_{k=1}^{K} \sum_{x \in C_i} L_2(x - m_i)^2$$

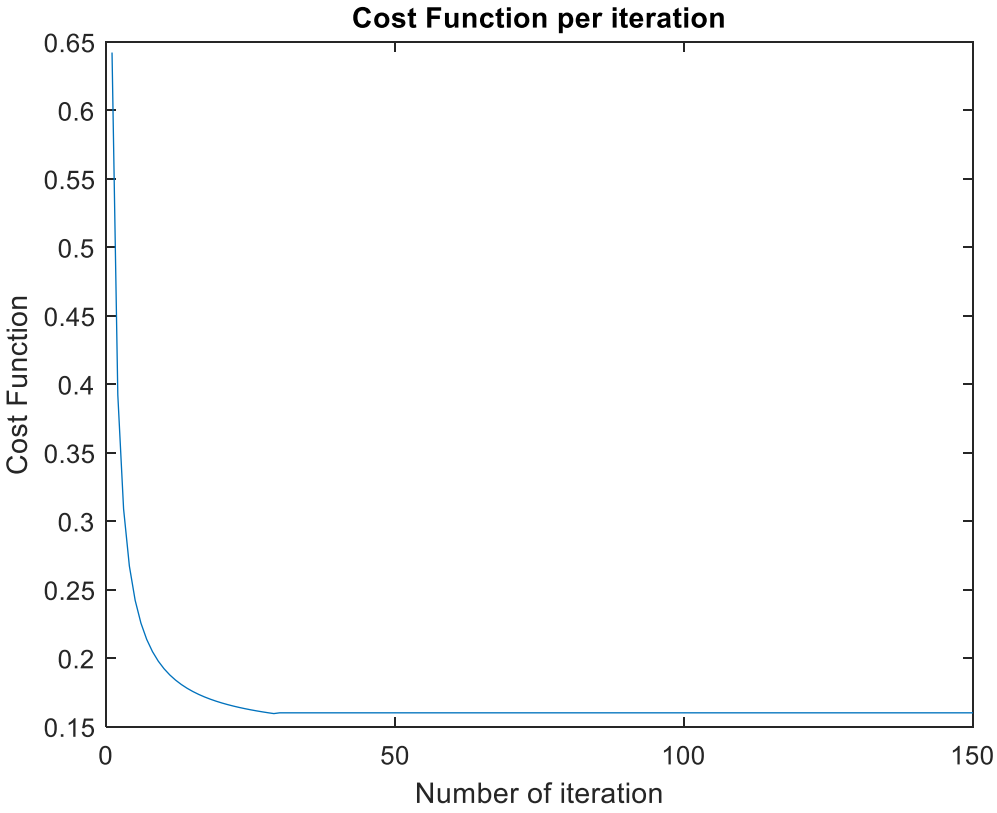
شکل 1-1 : رابطه تابع هزینه استفاده شده در این سوال

به ازای مقادیر مختلف از تعداد خوشه ها , نمودار تابع هزینه بر حسب تعداد تکرار را رسم می کنیم .

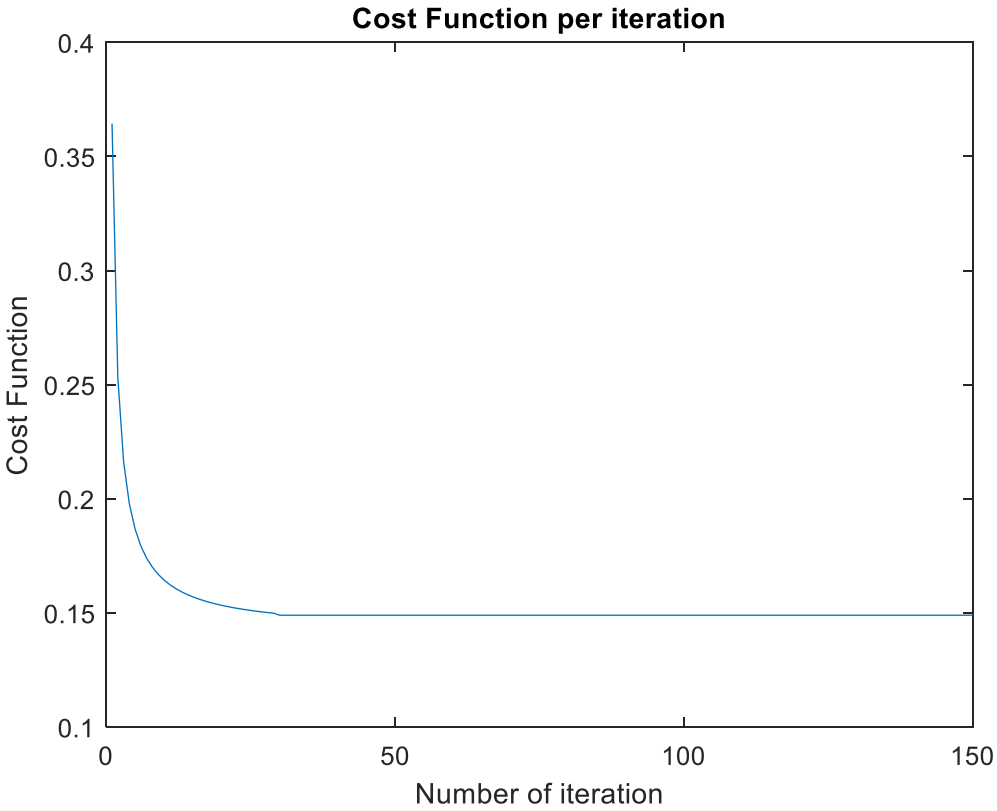


شکل 2-1 : تابع هزینه برحسب تعداد تکرار به ازای 5 خوشه

[1] Cost function



شکل 3-1 : تابع هزینه برحسب تعداد تکرار به ازای 10 خوشه



شکل 4-1 : تابع هزینه برحسب تعداد تکرار به ازای 20 خوشه

ب : تاثیر تکرار آزمایش

در این قسمت با توجه به این که در مرحله اول از پیاده سازی الگوریتم مراکز خوشه ها به صورت کاملا تصادفی انتخاب می شوند و با درنظر گرفتن این که مقدار دهی اولیه مراکز خوشه ها تاثیر مستقیمی بر روی خروجی به دست آمده دارد , برای هر تعداد خوشه الگوریتم را چند بار تکرار کرده و سپس میانگین و واریانس مربوط به تابع هزینه قسمت قبل را به ازای تعداد خوشه های مختلف گزارش می کنیم .

$Mean_1 = 0.3352$ $Var_1 = 0.0029$

$Mean_2 = 0.1966$ $Var_2 = 0.0081$

$Mean_3 = 0.1503$ $Var_3 = 0.0115$

مشاهده می کنیم که با افزایش تعداد خوشه , میانگین تابع هزینه کاهش یافته و در مقابل واریانس تابع هزینه افزایش پیدا می کند .

در نهایت می توان نتیجه گرفت تعداد خوشه 20 , عملکرد مطلوبی داشته و نسبت به دو مورد دیگر از کارایی و دقت بالاتری برخوردار است .

سوال 4 : مقدمات احتمال

الف : سوالات تحلیلی

الف-1 :

با استفاده از مدل فون نویمان[1] الگوریتم را پیاده سازی می کنیم :

احتمال شیر آمدن سکه برابر است با :

$$P(H) = p \in (0,1)$$

در نتیجه احتمال خط آمدن سکه برابر خواهد بود با :

$$P(T) = q = 1-p \in (0,1)$$

که در آن هدف ما پیاده سازی یک الگوریتم عادلانه (انتخاب فرد به صورت تصادفی با احتمال یکنواخت) می باشد . که در آن $P(X=0)$ و $P(X=1)$ به ترتیب احتمال انتخاب شدن فرد a و b است و خواهیم داشت :

$$P(X=0) = P(X=1) = 0.5$$

در گام نخست , ابتدا سکه را دوبار پرتاب می کنیم . اگر در پرتاب نخست سکه شیر آمد و در پرتاب بعدی خط آمد , $X=0$ و اگر در پرتاب نخست سکه خط آمد و در پرتاب بعدی شیر آمد $X=1$ را قرار می دهیم .

اگر نتیجه به دست آمده چیزی غیر از موارد بالا شد (هر دو شیر یا هر دو خط) به گام نخست بازگشته و دوباره , سکه را دوبار پرتاب می کنیم .

احتمال پرتاب شدن هر دو سکه به صورت شیر **یا** هر دو سکه به صورت خط برابر خواهد بود با :

$$P(HH) + P(TT) = p^2 + q^2$$

در نتیجه احتمال پرتاب شدن یک شیر و سپس یک خط پشت سر هم (یا به عبارتی $X=0$) برابر خواهد بود با :

$$P(HT) + (P(HH) + P(TT))*P(HT) + (P(HH) + P(TT))^2*P(HT) + \dots$$
$$= \frac{pq}{1-p^2-q^2} = \frac{1}{2}$$

به همین شیوه احتمال $X=1$ نیز برابر با $\frac{1}{2}$ خواهد بود و در نتیجه به یک توزیع عادلانه (رسیدن به احتمال یکنواخت برای انتخاب تصادفی یک فرد) دست پیدا کردیم .

[1] Von Neumann

تعداد پرتاب های مورد نیاز برای تولید X نیز برابر خواهد بود با :

$$\sum_{n=1}^{\infty} 2n(1-p^2-q^2)(p^2+q^2)^{n-1} = \frac{1}{pq}$$

-برگرفته از مرجع مذکور .

الف -2 :

با توجه به این که مدت زمان انتظار از توزیع نمایی پیروی می کند خواهیم داشت :

$f_X(x) = \lambda_1 * e^{-\lambda_1 x} * u(x)$ $\quad F_X(x) = \left(1 - e^{-\lambda_1 x}\right) * u(x)$ $\quad S.t: \lambda_1 > 0$

$f_Y(y) = \lambda_2 * e^{-\lambda_2 y} * u(y)$ $\quad F_Y(y) = \left(1 - e^{-\lambda_2 y}\right) * u(y)$ $\quad S.t: \lambda_2 > 0$

الف-2-1 :

Z = min(X,Y) → $f_Z(z) = f_X(z) + f_Y(z) - f_{XY}(z,z)$

G = max(X,Y) → $f_G(g) = f_{XY}(g,g)$

با توجه به رابطه ی کوواریانس خواهیم داشت :

$Cov_{ZG} = E\{\tilde{Z} \quad \tilde{G} \quad\} = E\{ZG\} - mean_Z * mean_G$

الف-2-2 :

با در نظر گرفتن این که نمرات از توزیع نرمال پیروی می کنند خواهیم داشت :

$f_U(u) = \frac{e^{-\frac{(u-\eta)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$ $\quad F_U(u) = G\left(\frac{u-\eta}{\sigma}\right)$ $\quad S.t: N(\eta, \sigma^2)$

$f_V(v) = \frac{e^{-\frac{(v-\eta)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$ $\quad F_V(v) = G\left(\frac{v-\eta}{\sigma}\right)$ $\quad S.t: N(\eta, \sigma^2)$

همچنین می دانیم :

Z = max(U,V) → $f_Z(z) = f_{UV}(z,z)$

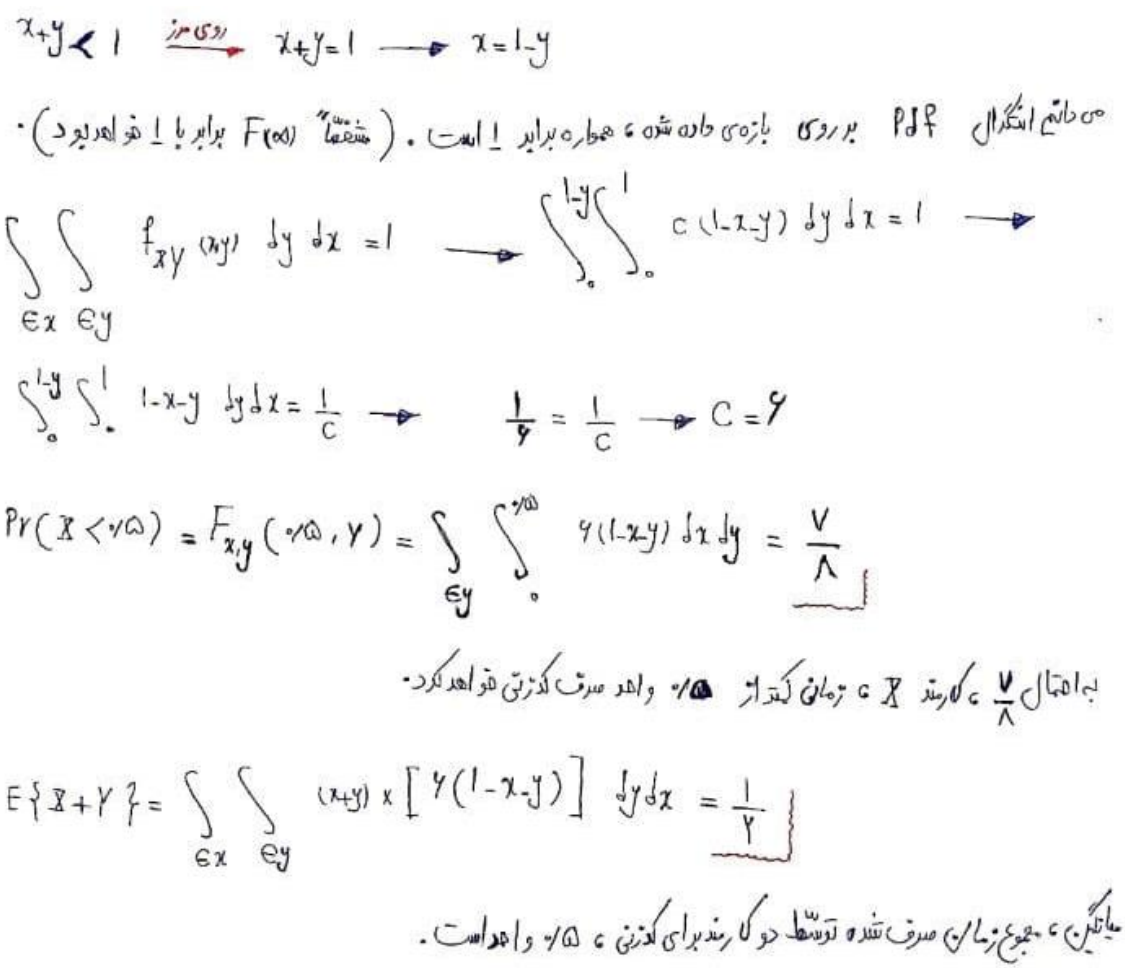
Cov(V , Z) = $E\{VZ\} - mean_V * mean_Z$

و

G = min(U,V) → $f_G(g) = f_U(g) + f_V(g) - f_{UV}(g,g)$

Cov(V , G) = $E\{VG\} - mean_V * mean_G$

الف -3 :



ب : سوالات شبیه سازی

ب-1 :

با استفاده از قطعه کد زیر مقادیر n را از یک تا 100 تغییر داده و به ازای هر مقدار 10000 بار آزمایش را انجام داده و احتمال این که حداقل یک زوج تولد یکسانی داشته باشند را حساب کرده و در انتها نمودار تعداد جمعیت بر حسب احتمال به دست آمده را رسم می کنیم .

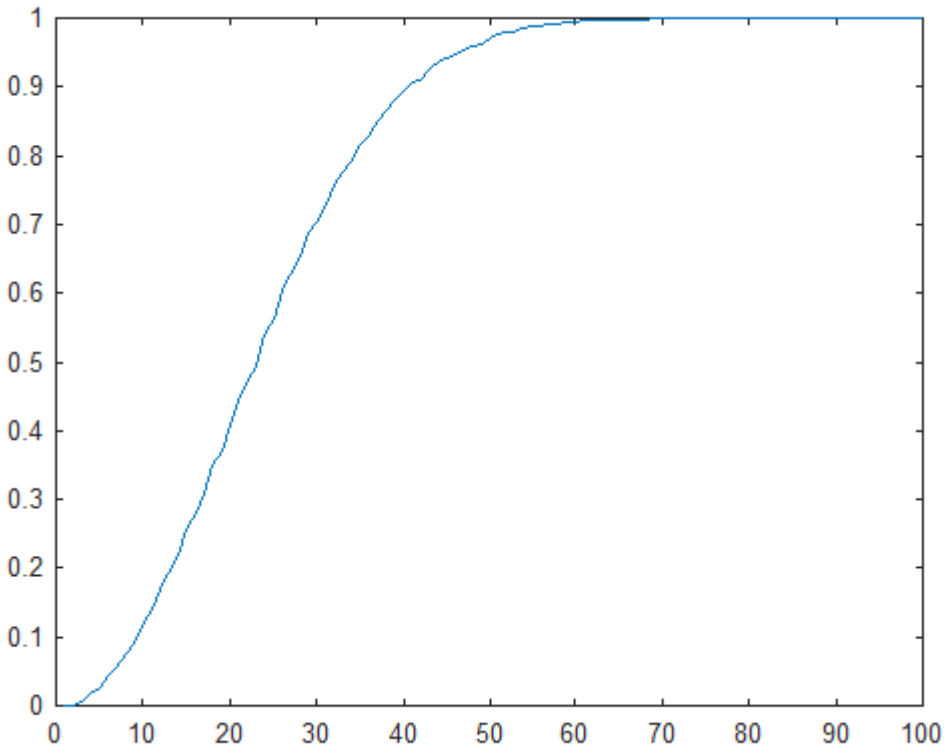
با استفاده از قطعه کد زیر خواهیم داشت :

```
%% Part A
people = 100 ;
Birthday = [] ;
Same_Birthday = 0 ;
Len = 0 ;
Prob = [] ;
for n = 1:people
   Same_Birthday = 0 ;
   for Number_Of_Tests = 1:10000
```

```
        Birthday = randi([1 365],1,n) ;
        [au,ia,ic] = unique(Birthday,'stable') ;
        Len = size(au) ;
        if Len(2) ~= size(Birthday)
           Same_Birthday = Same_Birthday + 1 ;
        end
     end
     Prob(end+1) = Same_Birthday / 10000 ;
end

number = 1:100 ;
plot(number , Prob) ;
```

در نتیجه نمودار احتمال مذکور بر حسب جمعیت برابر خواهد شد با :



شکل 1-4 : نمودار احتمال وجود حداقل یک زوج با تولد یکسان بر حسب تعداد افراد

مشاهده می کنیم وقتی تعداد افراد از یک حدی بیشتر می شود احتمال فوق به یک می رسد و این نشان می دهد وقتی تعداد افراد از یک حدی که بیشتر می شود , قطعا حداقل یک زوج وجود دارد که تولد یکسانی دارند .

پیوست:

تمامی کد ها در نرم افزار متلب R2020b اجرا و تست شده اند .