به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس سیستم‌های هوشمند

تمرین شماره 1

نام و نام خانوادگی : محمدرضا بختیاری

شماره دانشجویی : 810197468

مهر 1400

فهرست سوالات

سوال 1

در این سوال ابتدا نقاط ایستای تابع را پیدا می کنیم و در ادامه با استفاده از روش های گرادیان نزولی, جست و جوی خط و روش نیوتن مقدار بهینه (در اینجا کمینه) تابع را پیدا می کنیم و به مقایسه کلی سرعت و عملکرد روش ها می پردازیم . در نهایت نیز سعی می کنیم تابع درجه دو را یک بار به شکل ماتریسی پیاده سازی کنیم .

الف:

$$\nabla f = \begin{bmatrix} 6x_1 + 12 + 6x_2 \\ 16x_2 + 8 + 6x_1 \end{bmatrix} \xrightarrow{\nabla f = 0} \begin{cases} x_1 = -2.4 \\ x_2 = 0.4 \end{cases}$$

$$H(x_1, x_2) = \begin{bmatrix} 6 & 6 \\ 6 & 16 \end{bmatrix} \xrightarrow{\det(H - \lambda I) = 0} \begin{cases} \lambda_1 = 11 - \sqrt{61} & > 0 \\ \lambda_2 = 11 + \sqrt{61} & > 0 \end{cases}$$

$\lambda_i > 0 \longrightarrow$ Hessian matrix is Positive definite

$(-2.4 \, , \, 0.4)$: minimum

ب:

$$x^{K+1} = x^{K} + a^{K} p^{K}$$

$$p^{K} = -\nabla f(1,1) = \begin{bmatrix} -34 \\ -30 \end{bmatrix} \longrightarrow x^{2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + a_1 \begin{bmatrix} -34 \\ -30 \end{bmatrix}$$

$$\longrightarrow x^{2} = \begin{bmatrix} 1 - 34a_1 \\ 1 - 30a_1 \end{bmatrix} \quad \mathrm{I}$$

$$f(x_1, x_2) = 3x_1^2 + 12x_1 + 8x_2^2 + 8x_2 + 6x_1 x_2 \xrightarrow{\text{جاگذاری } \mathrm{I}}$$

$$f(a_1) = 13348\, a_1^2 - 1476\, a_1 + 37$$

$$\frac{df(a_1)}{da_1} = 0 \longrightarrow a_1 = \frac{41}{734} \longrightarrow x^{2} = \begin{bmatrix} -0.337 \\ -0.671 \end{bmatrix}$$

$$p^{2} = -\nabla f(x_1^2, x_2^2) = \begin{bmatrix} 5.952 \\ -4.758 \end{bmatrix}$$

$$x^{3} = x^{2} + a_2 p^{2} = \begin{bmatrix} -0.337 + 5.952\, a_2 \\ -0.671 - 4.758\, a_2 \end{bmatrix} \quad \mathrm{II}$$

$$f(x_1, x_2) = 3x_1^2 + 12x_1 + 8x_2^2 + 8x_2 + 6x_1 x_2 \xrightarrow{\text{جاگذاری } \mathrm{II}}$$

$$f(a_2) = 117.72\, a_2^2 + 58.05\, a_2 - 4.114$$

$$\frac{df(a_2)}{da_2} = 0 \longrightarrow a_2 = -0.246 \longrightarrow x^{3} = \begin{bmatrix} -1.801 \\ 0.499 \end{bmatrix}$$

با استفاده از قطعه کد زیر :

```
syms f(x1,x2) Df(x1,x2) X_k(a) X1_k(a) X2_k(a) g(a) Dg(a)
f(x1,x2) = 3*(x1*x1) + 12*x1 + 8*(x2*x2) + 8*x2 + 6*x1*x2 ;          %Define F
Df(x1,x2) = gradient(f(x1,x2), [x1, x2]) ;                           %Define F gradient
Df(x1,x2) = Df(x1,x2).' ;
X_new = [1 , 1] ;       X_old = [2 , 2] ;                            %Start point
Error = 0.2 ;           Threshold = 0.01 ;                           %Error function constraint
Temp = [ , ] ;          Temp_X = [ , ] ;
alpha = 0 ;                                                          %Step size

while Error > Threshold
   P = -Df(X_new(1),X_new(2)) ;
   X_k(a) = X_new + a.*P ;
   Temp = X_k(a);
   X1_k(a) = Temp(1) ;
   X2_k(a) = Temp(2) ;
   g(a) = f(X1_k(a),X2_k(a)) ;                                       %Define g(a)
   Dg(a) = gradient(g(a), [a]) ;                                     %Define dg(a)/da
   eqn = Dg(a) == 0 ;                                                %Find Optimal a
   alpha = solve(eqn) ;
   Temp_X = X_new ;
   X_new = X_k(alpha) ;
   X_old = Temp_X ;
   Error = abs( norm(X_new) - norm(X_old) ) ;                        %Update Loop constraint
end

disp(['Optimal X :   X_1 = ' num2str(double(X_new(1))) '   X_2 = ' num2str(double(X_new(2))) '   and alpha
is : ' num2str(double(alpha))]) ;
```

نتیجه زیر حاصل خواهد شد :

```
>> IS_HW1_Q1_Part_B
Optimal X :   X_1 = -2.389   X_2 = 0.39431   and alpha is : 0.055707
>>
```

شکل 1-1 : نتیجه محاسبات شبیه سازی

در الگوریتم به کارگرفته شده تابع خطا را اختلاف اندازه های نقاط متوالی در نظر می گیریم که اگر این مقدار از مقدار آستانه ی تحمل تعریف شده (در اینجا 0.01) کمتر شود الگوریتم متوقف می شود و نقطه بهینه به همراه طول پله در خروجی نمایش داده می شوند .

ج:

$$p^1 = -\nabla^2 f(x)^{-1} \nabla f(x) = \frac{-1}{60}\begin{bmatrix} 12 & -6 \\ -6 & 6 \end{bmatrix}\begin{bmatrix} 24 \\ 30 \end{bmatrix} = \begin{bmatrix} -3.4 \\ -0.6 \end{bmatrix}$$

$$x^2 = x^1 + p^1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -3.4 \\ -0.6 \end{bmatrix} = \begin{bmatrix} -2.4 \\ 0.4 \end{bmatrix}$$

$$p^2 = -\nabla^2 f(x^2)^{-1} \nabla f(x^2) = \frac{-1}{60}\begin{bmatrix} 12 & -6 \\ -6 & 6 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x^3 = x^2 + p^2 = \begin{bmatrix} -2.4 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -2.4 \\ 0.4 \end{bmatrix}$$

مشاهده می کنیم که با استفاده از روش نیوتن در یک مرحله به نقطه بهینه دست پیدا می کنیم در نتیجه روش نیوتن در این سوال کارایی دارد .

د:

$$f(x_1,x_2) = \frac{1}{2}\begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} 6 & 20 \\ -8 & 16 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 12 & 8 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\frac{df(x_1,x_2)}{dx_1} = \frac{1}{2}\begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 6 & 20 \\ -8 & 16 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} 6 & 20 \\ -8 & 16 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 12 & 8 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$= 6x_1 + 12 + 6x_2$$

$$\frac{df(x_1,x_2)}{dx_2} = \frac{1}{2}\begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} 6 & 20 \\ -8 & 16 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} 6 & 20 \\ -8 & 16 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 12 & 8 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$= 16x_2 + 8 + 6x_1$$

$$\nabla f = \begin{bmatrix} \frac{df(x_1,x_2)}{dx_1} \\ \frac{df(x_1,x_2)}{dx_2} \end{bmatrix} = \begin{bmatrix} 6x_1 + 12 + 6x_2 \\ 16x_2 + 8 + 6x_1 \end{bmatrix} \longrightarrow \nabla^2 f(x_1,x_2) = \begin{bmatrix} 6 & 6 \\ 6 & 16 \end{bmatrix}$$

$$\det(\nabla^2 f - \lambda I) = 0 \longrightarrow \begin{cases} \lambda_1 = 11 - \sqrt{61} & > 0 \\ \lambda_2 = 11 + \sqrt{61} & > 0 \end{cases}$$

$\lambda_i > 0 \longrightarrow \nabla^2 f(x_1,x_2)$ is positive definitte

$$\nabla f = 0 \longrightarrow \begin{cases} x_1 = -2.4 \\ x_2 = 0.4 \end{cases} : \text{minimum}$$

تابع قسمت الف :

$$f(x_1,x_2) = 3x_1^2 + 12x_1 + 8x_2^2 + 8x_2 + 6x_1x_2$$

$$\longrightarrow f(x_1,x_2) = \begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} 3 & 6 \\ 0 & 8 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 12 & 8 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

همانگونه که مشاهده کردیم هر دو تابع درجه دو یکسان بوده و فقط نمایش آن ها متفاوت بوده است . پس طبق انتظار باید برای هر دو به نقاط ایستای یکسان دست پیدا کنیم که در نتایج محاسبات دستی نیز به همین نتیجه رسیدیم .

توجه شود که تابع درجه دوم موجود در قسمت الف را با بی نهایت نمایش ماتریسی همانند قسمت د می توان نمایش داد . به این صورت که مجموع عناصر قطر فرعی ماتریس عددی اول همواره برابر ضریب عبارت x1*x2 که در اینجا 6 است باشد .

سوال 2

در این سوال ابتدا با روش گرادیان نزولی , تابع غیر محدب داده شده را بهینه سازی می کنیم , سپس با شروع از دو نقطه مختلف , طول گام را یک بار با روش تحلیلی و یک بار با روش آرمیجو به دست می آوریم و به مقایسه آن ها می پردازیم . در نهایت با روش های فراابتکاری (در اینجا تبرید شبیه سازی) , تابع را بهینه سازی می کنیم و کمینه محلی را به دست می آوریم .

الف:

از دو نقطه ی (7 , 0.5-) و (7 , 1) به عنوان نقاط شروع استفاده می کنیم . با استفاده از قطعه کد زیر نتیجه را برای هر دو نقطه ی آغازین بررسی می کنیم :

```
syms f(x1,x2) Df(x1,x2) X_k(a) X1_k(a) X2_k(a) g(a) Dg(a)
f(x1,x2) = (x1*x1) - (10*x2*cos(0.2*pi*x1)) + (x2*x2) - (15*x1*cos(0.4*pi*x2)) ; %Define F
Df(x1,x2) = gradient(f(x1,x2), [x1, x2]) ;                                        %Define F gradient
Df(x1,x2) = Df(x1,x2).' ;
X_new = [7 , -0.5] ;     X_old = [2 , 2] ;    Start_Point = X_new ;                %Start point
Error = 0.2 ;            Threshold = 0.01 ;                                        %Error function constraint
Temp = [ , ] ;           Temp_X = [ , ] ;
alpha = 0 ;                                                                        %Step size

while Error > Threshold
   P = -Df(X_new(1),X_new(2)) ;
   X_k(a) = X_new + a.*P ;
   Temp = X_k(a);
   X1_k(a) = Temp(1) ;
   X2_k(a) = Temp(2) ;
   g(a) = f(X1_k(a),X2_k(a)) ;                                                     %Define g(a)
   Dg(a) = gradient(g(a), [a]) ;                                                   %Define dg(a)/da
   eqn = Dg(a) == 0 ;                                                              %Find Optimal a
   alpha = vpasolve(eqn,a) ;
   Temp_X = X_new ;
   X_new = X_k(alpha) ;
   X_old = Temp_X ;
   Error = abs( norm(X_new) - norm(X_old) ) ;                                      %Update Loop constraint
end

disp(['Optimal X is :  ( X_1 = ' num2str(double(X_new(1))) '   X_2 = ' num2str(double(X_new(2))) ' )   And
Optimal f is : ( ' num2str(double(f(X_new(1),X_new(2)))) ' )   and the Start point is : ( x_1 = '
num2str(double(Start_Point(1))) '   x_2 = ' num2str(double(Start_Point(2))) ' )'  ]) ;
```

توجه شود که طول پله ثابت نیست و مانند قسمت ب در سوال اول از طول پله متغییر استفاده می کنیم .

نتایج خروجی برای حالتی که از نقطه شروع (7 , 0.5-) استفاده می کنیم به شرح زیر است :

```
>> IS_HW1_Q2_Part_A
Optimal X is :  ( X_1 = 7.4618   X_2 = -0.0013688 )   And Optimal f is : ( -56.2487 )   and the Start point is : ( x_1 = 7   x_2 = -0.5 )
>>
```

شکل 1-2 : نقطه شروع (7 , 0.5-) که به 56.2- همگرا می شود .

نتایج خروجی برای حالتی که از نقطه شروع (1 , 7) استفاده می کنیم به شرح زیر است :

```
>> IS_HW1_Q2_Part_A
Optimal X is :  ( X_1 = 5.2308   X_2 = -4.9989 )   And Optimal f is : ( -75.576 )   and the Start point is : ( x_1 = 7   x_2 = 1 )
fx >>
```

شکل 2-2 : نقطه شروع (1 , 7) که به 75.5- همگرا می شود .

مشاهده می کنیم که با استفاده از دو نقطه ی اولیه متفاوت به دو نتیجه متفاوت دست پیدا کردیم و همانطور که در صورت سوال نیز ذکر شده است , به ازای شروع از نقطه (0.5- , 7) الگوریتم در دام کمینه های محلی افتاده است ولی به ازای نقطه شروع (1 , 7) به مقدار بهینه تابع (حدود 75.5-) دست پیدا می کنیم .

در این سوال نیز همانند قسمت ب در سوال اول , در الگوریتم به کار گرفته شده تابع خطا را اختلاف اندازه های نقاط متوالی در نظر می گیریم که اگر این مقدار از مقدار آستانه ی تحمل تعریف شده (در اینجا 0.01) کمتر شود الگوریتم متوقف می شود .

ب:

طول گام بهینه را یک بار با شروع از نقطه (0,0) و یک بار با شروع از نقطه (10,7) به ازای روش تحلیلی و روش آرمیجو به دست می آوریم و با هم مقایسه می کنیم .

روش تحلیلی با شروع از (0,0) :

```
>> IS_HW1_Q2_Part_B_Analytical_method
alpha is : ( -0.069855 )   and number of iteration is : ( 4 )   and the Start point is : ( x_1 = 0   x_2 = 0 )
fx >>
```

شکل 3-2 : طول گام بهینه به همراه تعداد تکرار به ازای شروع از (0,0) با روش تحلیلی

روش تحلیلی با شروع از (10,7) :

```
>> IS_HW1_Q2_Part_B_Analytical_method
alpha is : ( -0.038317 )   and number of iteration is : ( 6 )   and the Start point is : ( x_1 = 10   x_2 = 7 )
fx >>
```

شکل 4-2 : طول گام بهینه به همراه تعداد تکرار به ازای شروع از (10,7) با روش تحلیلی

در روش تحلیلی از قطعه کدی که برای قسمت الف به کار بردیم , استفاده می کنیم و فقط نقاط شروع را تغییر می دهیم .

در روش آرمیجو از قطعه کد زیر استفاده می کنیم :

```
syms f(x1,x2) Df(x1,x2)
f(x1,x2) = (x1*x1) - (10*x2*cos(0.2*pi*x1)) + (x2*x2) - (15*x1*cos(0.4*pi*x2)) ; %Define F
Df(x1,x2) = gradient(f(x1,x2), [x1, x2]) ;                                   %Define F gradient
Df(x1,x2) = Df(x1,x2).' ;
alpha = 10 ;          beta = 0.5 ;    c = 0.001 ;                              %Default Value
X_old = [0 , 0] ;       F_new = 0 ;      Status = 1 ;                          %Start Point
while Status
   X_new = X_old - (alpha.*Df(X_old(1),X_old(2))) ;
   F_new = f(X_new(1),X_new(2)) ;
   if F_new<=f(X_old(1),X_old(2)) -
((c*alpha).*(Df(X_old(1),X_old(2))*transpose(Df(X_old(1),X_old(2)))))          %Check armijo condition
      Status = 0 ;
   end
   alpha = alpha * beta ;
end
disp(['alpha is :  ' num2str(double(alpha)) '  and Optimal f is : ' num2str(double(f(X_new(1),X_new(2))))  '
and the Start point is : ( x_1 = ' num2str(double(X_old(1))) '   x_2 = ' num2str(double(X_old(2))) ' )' ]) ;
```

حال نتایج به دست آمده از روش آرمیجو را بررسی می کنیم .

روش آرمیجو با شروع از (0,0) :

```
>> IS_HW1_Q2_Part_B_Armijo_Rule
alpha is :  0.039063  and Optimal f is : -13.5709 and the Start point is : ( x_1 = 0   x_2 = 0 )
>>
```

شکل 2-5 : طول گام بهینه به ازای شروع از (0,0) با روش آرمیجو

روش آرمیجو با شروع از (10,7) :

```
>> IS_HW1_Q2_Part_B_Armijo_Rule
alpha is :  0.078125  and Optimal f is : 6.3943 and the Start point is : ( x_1 = 10   x_2 = 7 )
>>
```

شکل 2-6 : طول گام بهینه به ازای شروع از (10,7) با روش آرمیجو

مشاهده می کنیم طبق Iteration های به دست آمده در حالت تحلیلی , سرعت این روش بیشتر بوده اما به عنوان مثال وقتی از نقطه (10,7) شروع به حرکت می کنیم نقطه بهینه نهایی 245 می شود که نشان می دهد روش تحلیلی لزوما به ازای نقاط شروع مختلف همگرا نخواهد شد .

ج:

با استفاده از روش تبرید شبیه سازی , به این صورت عمل می کنیم که اگر در هر مرحله مقدار تابع کاهش پیدا کرد , مقدار جدید را جایگزین کرده و اگر مقدار تابع کاهش پیدا نکرد با یک احتمالی مقدار جدید را جایگزین می کنیم .

در ابتدا temperature مقدار زیادی دارد و ممکن است الگوریتم در هر مرحله نقاطی را که لزوما منجر به کاهش نمی شوند را انتخاب کند ,ولی در ادامه با کاهش دما , احتمال انتخاب نقاطی که منجر به کاهش نمی شوند , کاهش پیدا کرده و به مقدار بهینه نزدیک تر می شویم .

با استفاده از قطعه کد زیر داریم :

```
syms f(x1,x2)
f(x1,x2) = (x1*x1) - (10*x2*cos(0.2*pi*x1)) + (x2*x2) - (15*x1*cos(0.4*pi*x2)) ;    %Define F
X_min = [0,0] ;     F_min = f(X_min(1),X_min(2)) ;
X_new = [ , ] ;     Counter = 0 ;
X_old = [1,1] ;     X_Random = [ , ] ;     Start_Point = X_old ;                     %Set start point
Temperature = 1000 ;  Temperature_rate = 1 ;   T = [Temperature , Temperature] ;     %Set the temperature

while Counter < 1000
   X_Random  = normrnd(X_old , T)/100;
   X_new = X_old + X_Random ;
   if rand() <= min(1 , exp((f(X_old(1),X_old(2)) - f(X_new(1),X_new(2)))/Temperature))
      X_old = X_new ;
   end
   if rand() > min(1 , exp((f(X_old(1),X_old(2)) - f(X_new(1),X_new(2)))/Temperature))
      X_old = X_old ;                                                                %do nothing
   End
   if f(X_old(1),X_old(2)) < F_min
      X_min = X_old ;
      F_min = f(X_min(1),X_min(2)) ;
   end
   Temperature = Temperature - Temperature_rate ;                                    %update temperature
   T = [Temperature , Temperature] ;
   Counter = Counter + 1 ;
end

disp(['Optimal F is : ( ' num2str(double(F_min)) ' ) and Optimal X is : ( ' num2str(double(X_min)) ' ) and
Start point is : ( ' num2str(double(Start_Point)) ' )']) ;
```

برای مثال با شروع از نقطه (1,1) به نتیجه زیر خواهیم رسید که همان طور که مشاهده می کنیم به مقدار بهینه دست پیدا کردیم .

```
>> IS_HW1_Q2_Part_C
Optimal F is : ( -74.9127 ) and Optimal X is : ( 5.1104      -4.909 ) and Start point is : ( 1   1 )
>>
```

شکل 2-6 : رسیدن به کمینه محلی با روش تبرید شبیه سازی (با شروع از نقطه (1,1))

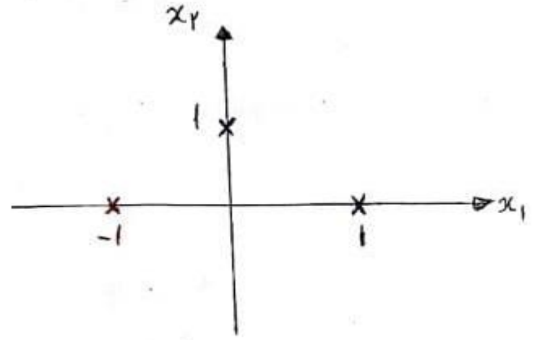
سوال 3

در این سوال با استفاده از الگوریتم ماشین بردار پشتیبان داده ها را طبقه بندی می کنیم و در نهایت دقت داده آموزش , ماتریس آشفتگی و ماتریس اطمینان را گزارش می کنیم .

الف:

نقاط آبی رنگ را با برچسب 1- و نقطه ی قرمز رنگ را با برچسب 1+ در نظر می گیریم , داریم :

$S=\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$



$$\begin{cases} s_1=\begin{pmatrix} 0 \\ 1 \end{pmatrix} \longrightarrow \tilde{s}_1=\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \\ s_2=\begin{pmatrix} 1 \\ 0 \end{pmatrix} \longrightarrow \tilde{s}_2=\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \\ s_3=\begin{pmatrix} -1 \\ 0 \end{pmatrix} \longrightarrow \tilde{s}_3=\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \end{cases}$$

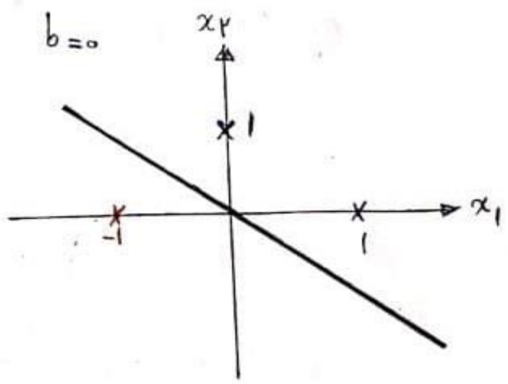
$$\begin{cases} a_1 \tilde{s}_1 \cdot \tilde{s}_1 + a_2 \tilde{s}_2 \cdot \tilde{s}_1 + a_3 \tilde{s}_3 \cdot \tilde{s}_1 = -1 \longrightarrow 2a_1 + a_2 + a_3 = -1 \\ a_1 \tilde{s}_1 \cdot \tilde{s}_2 + a_2 \tilde{s}_2 \cdot \tilde{s}_2 + a_3 \tilde{s}_3 \cdot \tilde{s}_2 = -1 \longrightarrow a_1 + 2a_2 = -1 \\ a_1 \tilde{s}_1 \cdot \tilde{s}_3 + a_2 \tilde{s}_2 \cdot \tilde{s}_3 + a_3 \tilde{s}_3 \cdot \tilde{s}_3 = +1 \longrightarrow a_1 + 2a_3 = +1 \end{cases}$$

$$\longrightarrow \begin{cases} a_1 = -1 \\ a_2 = 0 \\ a_3 = 1 \end{cases}$$

$$\tilde{w} = \sum_{i=1}^{3} a_i \tilde{s}_i = a_1 \tilde{s}_1 + a_2 \tilde{s}_2 + a_3 \tilde{s}_3 = \begin{pmatrix} -1 \\ -1 \\ 0 \end{pmatrix}$$

$y = wx + b \quad , \quad w = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \quad b = 0$

$\longrightarrow y = \begin{pmatrix} -1 \\ -1 \end{pmatrix} X$

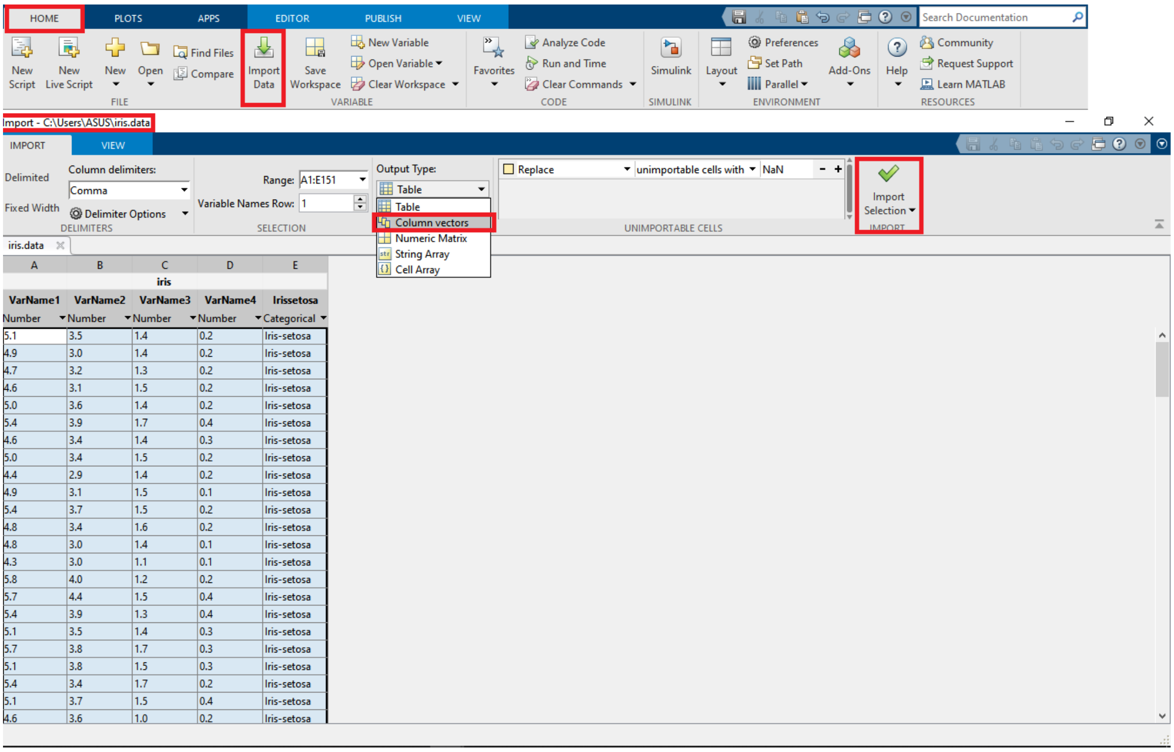


ب:

ابتدا با استفاده از قطعه کد زیر داده ها را طبقه بندی می کنیم و نمای کلی دسته بندی داده ها را مشاهده می کنیم .

توجه : دقت شود که با توجه به این که پیاده سازی را با نرم افزار متلب انجام می دهیم ، برای استخراج دادگان از کد ذکر شده در صورت سوال نمی توان استفاده کرد و **باید حتما در هنگام اولین اجرا یک بار در قسمت Home با استفاده از گزینه Import Data داده ها را استخراج کنیم .**

نحوه استخراج داده ها :

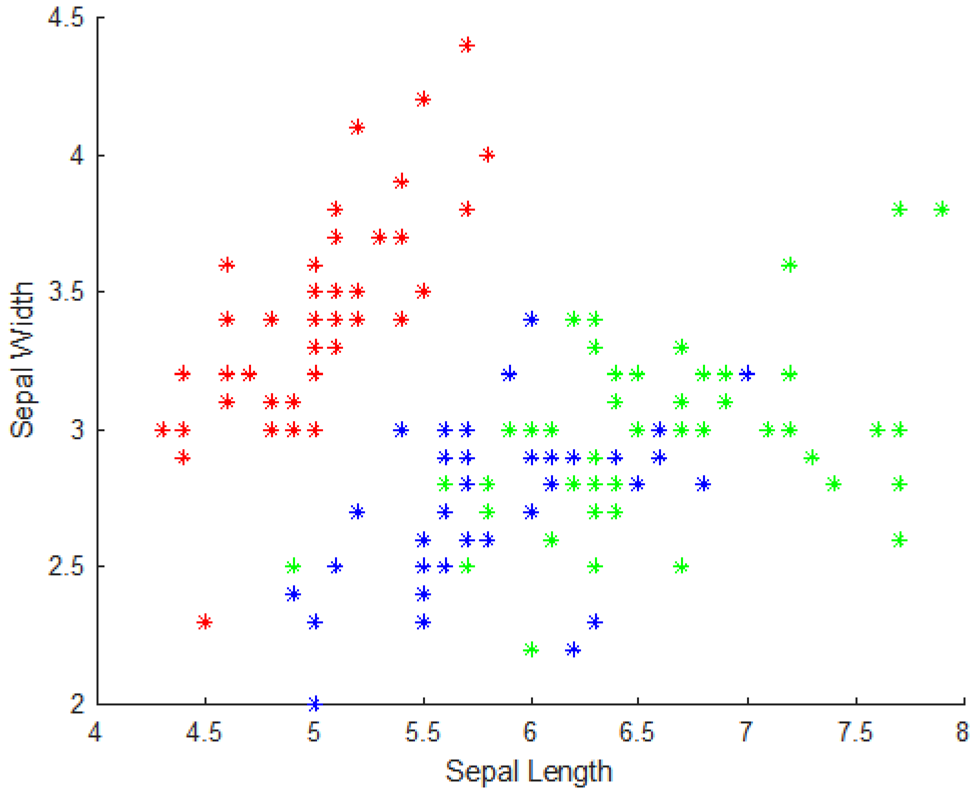


شکل 1-3 : نحوه Import کردن داده ها قبل از اولین اجرای برنامه

با استفاده از قطعه کد زیر داریم :

```
Iris_setosa_Length = VarName1(1:50) ;
Iris_setosa_width =  VarName2(1:50) ;
Iris_versicolor_Length = VarName1(51:100) ;
Iris_versicolor_width =  VarName2(51:100) ;
Iris_virginica_Length =  VarName1(101:150) ;
Iris_virginica_width =   VarName2(101:150) ;
hold on                                              %Plot
plot(Iris_setosa_Length,Iris_setosa_width,'r*');
plot(Iris_versicolor_Length,Iris_versicolor_width,'b*');
plot(Iris_virginica_Length,Iris_virginica_width,'g*');
hold off
xlabel('Sepal Length') ;
ylabel('Sepal Width') ;
```

که نتیجه زیر حاصل خواهد شد :



شکل 1-3 : دسته بندی داده ها

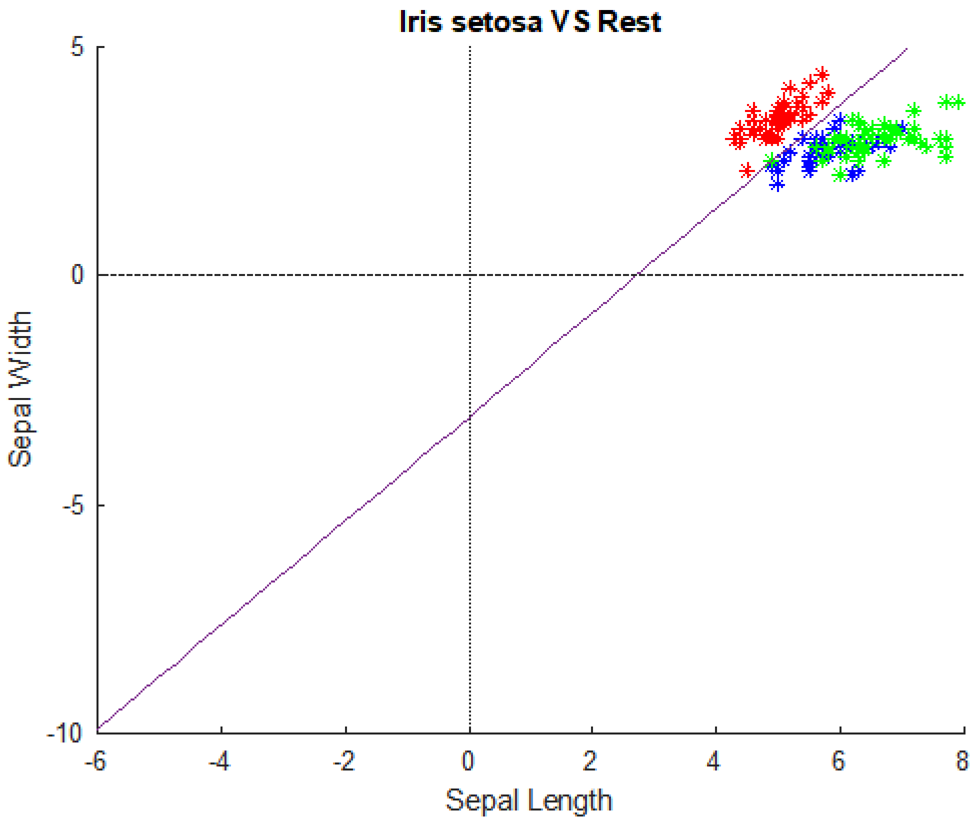
حال باتوجه به الگوریتم یک طبقه در قیاس با بقیه , هر دسته را یک بخش مجزا در نظر گرفته و دو دسته ی دیگر را نیز با هم در نظر می گیریم . به عنوان مثال یک بار دسته ی setosa را جدا در نظر می گیریم و با دو دسته دیگر قیاس می کنیم , که نتیجه آن را نیز در ادامه خواهیم دید .

با استفاده از قطعه کد زیر داریم :

```
Iris_setosa_Length = VarName1(1:50) ;                    %Setosa Length
Iris_setosa_width =  VarName2(1:50) ;                    %Setosa Width
Iris_versicolor_Length = VarName1(51:100) ;              %Versicolor Length
Iris_versicolor_width =  VarName2(51:100) ;              %Versicolor Width
Iris_virginica_Length =  VarName1(101:150) ;             %Virginica Length
Iris_virginica_width =   VarName2(101:150) ;             %Virginica Width
S = [Iris_setosa_Length(1) , Iris_setosa_width(1) , 1] ;
Temp = [ , ] ;     Summ = [ ] ;      Temp_Sum = [ ];
Counter_1 = 2 ;    Counter_2 = 1 ;    Counter_3 = 1 ;   Counter_4 = 1 ;
                                                         %Create S Vectors
while Counter_1 < 151
   Temp = [VarName1(Counter_1) , VarName2(Counter_1) , 1] ;
   S = [S ; Temp] ;
   Counter_1 = Counter_1 + 1 ;
end
                                                         %Create matrix Summ
while Counter_2 < 151
   Counter_3 = 1 ;
   while Counter_3 < 151
      Temp_Sum(end+1) = sum(S(Counter_2,:).*S(Counter_3,:)) ;
      Counter_3 = Counter_3 + 1 ;
   end
   Summ = [Summ ; Temp_Sum] ;
   Temp_Sum = [ ];
   Counter_2 = Counter_2 + 1 ;
end

Y_positive = 1.+zeros(1,50) ;                            % (+1) Label
Y_negative = -1.+zeros(1,50) ;                           % (-1) Label
Y_1 = [Y_negative Y_positive Y_positive] ;
Y_2 = [Y_positive Y_negative Y_positive] ;
Y_3 = [Y_positive Y_positive Y_negative] ;
A_1 = Y_1*inv(Summ) ;
A_2 = Y_2*inv(Summ) ;
A_3 = Y_3*inv(Summ) ;
W = [0,0,0] ;
                                                         %Create matrix W
while Counter_4 < 151
   W = W + (A_1(Counter_4).*S(Counter_4,:)) ;
   Counter_4 = Counter_4 + 1 ;
end
                                                         % Y = WX + b
f = @(x) ((W(1)/W(2))*x)+(-W(3)/W(2)-3);
hold on
 plot(Iris_setosa_Length,Iris_setosa_width,'r*');
 plot(Iris_versicolor_Length,Iris_versicolor_width,'b*');
 plot(Iris_virginica_Length,Iris_virginica_width,'g*');
 fplot( f ) ;
 xline(0) ;
 yline(0) ;
 xlabel('Sepal Length') ;
 ylabel('Sepal Width') ;
 title('Iris setosa VS Rest') ;
 hold off
```

مشاهده می کنیم دسته ی مربوطه را با دقت خوبی طبقه بندی کرده ایم :



شکل 2-3 : مقیاس دسته ی setosa با دو دسته ی دیگر

پیوست:

تمامی شبیه سازی ها با استفاده از نرم افزار متلب R2020b انجام شده و در پوشه Codes موجود می باشد .

لازم به ذکر است همان طور که در سوال 3 نیز اشاره شده لازم است قبل از اجرای کد مربوط به این سوال , داده های مذکور با روش ذکر شده Import شوند .