

به نام خدا



دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر



درس سیستم‌های هوشمند

تمرین شماره ۱

نام و نام خانوادگی : شایان واصف احمدزاده

شماره دانشجویی : 810197603

1400 مهر

فهرست سوالات

سوال 1 : بهینه سازی در توابع محدب 3

الف: تحلیلی(بخش های الف، ب ، ج ، د) 3

ب: شبیه سازی (بخش ب) 4

سوال 2 : بهینه سازی در توابع غیر محدب 6

الف: روش گرادیان نزولی 6

ب: طول گام در روش گرادیان نزولی 11

ب-1: قاعده‌ی Armijo 11

ب-2: روش تحلیلی 12

ج: روش فرآابتکاری 14

سوال 3 : ماشین بردار پشتیبان 17

الف: تحلیلی 17

ب: شبیه سازی 18

زیربخش 1: پیاده‌سازی بدون استفاده از کتابخانه 18

زیربخش 2: پیاده سازی با استفاده از کتابخانه 26

سوال 1

الف (تحلیلی بخش های الف ، ب ، ج ، د)

سریع ۱) از اعیان تفاضل استاد است:

$$f(x_1, x_2) = 3x_1^2 + 72x_1 + 72x_2 + 8x_2^2 + 6x_1x_2 \rightarrow \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \xrightarrow{\text{stationary point}} \nabla f = 0 \rightarrow \begin{cases} 6x_1 + 72 + 6x_2 = 0 \\ 16x_2 + 8 + 6x_1 = 0 \end{cases}$$

$$\rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -2/4 \\ 1/4 \end{pmatrix} \rightarrow H = \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix} \xrightarrow{\text{نهاد}} H = \begin{pmatrix} 6 & 6 \\ 6 & 16 \end{pmatrix} \rightarrow \begin{cases} 16 = 6 \\ 6 = 6 \end{cases} \rightarrow \begin{pmatrix} 6 & 6 \\ 6 & 16 \end{pmatrix} = 60 > 0.$$

نابراین ماتریس مسین مثبت مسین است $\left(\begin{array}{c} x_1 \\ x_2 \end{array} \right) = \begin{pmatrix} -2/4 \\ 1/4 \end{pmatrix}$ می‌شود.

ب) روش مترادول (گردابیان): به دلیل اینکه در این طبقه از روش SD استفاده می‌شود.

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) \rightarrow \alpha = \arg \min f(x^k - \alpha \nabla f(x^k)), k=1 \rightarrow \boxed{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}}$$

$$\rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^2 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^1 - \alpha^1 \begin{pmatrix} 6x_1^k + 72 + 6x_2^k \\ 16x_2^k + 8 + 6x_1^k \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \alpha^1 \begin{pmatrix} 6+72+6 \\ 16+8+6 \end{pmatrix} = \begin{pmatrix} 1-24\alpha^1 \\ 1-30\alpha^1 \end{pmatrix}$$

$$\rightarrow f(1-24\alpha^1, 1-30\alpha^1) = 3(1-24\alpha^1)^2 + 72(1-24\alpha^1) + 8(1-30\alpha^1)^2 + 8(1-30\alpha^1) + 6(1-24\alpha^1)(1-30\alpha^1)$$

$$= 37 - 1476\alpha^1 + 73248\alpha^1 \rightarrow f' \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 = -1476 + 26496\alpha^1 \rightarrow \alpha_1 = \sqrt{-557} \rightarrow \boxed{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -\sqrt{3368} \\ -\sqrt{671} \end{pmatrix}}$$

$$k=2 : x^3 = x^2 - \alpha^2 \nabla f(x^2) \rightarrow \boxed{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^3 = \begin{pmatrix} -\sqrt{3368} \\ -\sqrt{671} \end{pmatrix} - \alpha^2 \begin{pmatrix} 72+6(-\sqrt{3368}-\sqrt{671}) \\ 8+6(-\sqrt{3368})+16(-\sqrt{671}) \end{pmatrix} = \begin{pmatrix} -\sqrt{3368}-\alpha^2 513532 \\ -\sqrt{671}-\alpha^2 417568 \end{pmatrix}}$$

$$\rightarrow f(x_1^3, x_2^3) = 117/579\alpha_2^2 - 581/81\alpha_2 - 4/1111 \rightarrow f' \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = -581/8 + 235/758\alpha_2 = 0 \rightarrow \alpha_2 = \sqrt{247}$$

$$\rightarrow \boxed{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^3 = \begin{pmatrix} -118\sqrt{7} \\ 45.39 \end{pmatrix}}$$

ج) روش تحریق دم (میرن)

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})} \xrightarrow{\text{مسیر}} x^{(k+1)} = x^{(k)} - \left(\frac{\nabla f(x^{(k)})}{\nabla^2 f(x^{(k)})} \right)^{-1} \nabla f(x^{(k)}), k=1, \boxed{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}}$$

$$\nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}) = \begin{pmatrix} 6 & 6 \\ 6 & 16 \end{pmatrix}^{-1} \begin{pmatrix} 24 \\ 24 \end{pmatrix} = \begin{pmatrix} 4267 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 24 \\ 24 \end{pmatrix} = \begin{pmatrix} 3/4 \\ 1/6 \end{pmatrix} \rightarrow \boxed{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} 3/4 \\ 1/6 \end{pmatrix} = \begin{pmatrix} -2/4 \\ 1/4 \end{pmatrix}}$$

$$k=2 \rightarrow \nabla^2 f(x^{(k)}) \nabla f(x^{(k)}) = \begin{pmatrix} 1267 & -1 \\ -1 & 1 \end{pmatrix} \underbrace{\nabla f(-2/4, 1/4)}_{0} = 0 \rightarrow \boxed{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 0 = \begin{pmatrix} -2/4 \\ 1/4 \end{pmatrix}}$$

ب) نظر در این روش مبتدا می‌شود.

شکل ۱-۱: محاسبات بخش های الف تا ج

$$f(x_1, x_2) = 3x_1^2 + 72x_1 + 8x_2^2 + 8x_2 + 6x_1x_2 = \alpha(x_1, x_2)A\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \beta\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= (x_1, x_2) \begin{pmatrix} 3 & 72 \\ 8 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (72, 8) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$f_2(x_1, x_2) = \frac{1}{2}(x_1, x_2) \begin{pmatrix} 6 & 2 \\ -8 & 16 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (72, 8) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\nabla_x^T y = y, \quad \nabla_x^T (b^T x) = \nabla_x^T (x^T b) = b, \quad \nabla_x^T (x^T A x) = (A + A^T)x$$

$$\rightarrow \nabla f_1(x_1, x_2) = 2 \begin{pmatrix} 3 & 72 \\ 8 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 12 \\ 8 \end{pmatrix} = \begin{pmatrix} 6 & 6 \\ 6 & 16 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 12 \\ 8 \end{pmatrix}$$

$$\nabla f_2(x_1, x_2) = \frac{1}{2} \left[\begin{pmatrix} 6 & 2 \\ -8 & 16 \end{pmatrix} + \begin{pmatrix} 6 & -8 \\ 2 & 16 \end{pmatrix} \right] \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 12 \\ 8 \end{pmatrix} = \begin{pmatrix} 6 & 6 \\ 6 & 16 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 12 \\ 8 \end{pmatrix}$$

$$\xrightarrow{\text{stationary point}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 6 & 6 \\ 6 & 16 \end{pmatrix}^{-1} \begin{pmatrix} 12 \\ 8 \end{pmatrix} = \begin{pmatrix} -2/4 \\ 1/4 \end{pmatrix}, \quad \nabla(\nabla f_1 / \nabla f_2) = \begin{pmatrix} 6 & 6 \\ 6 & 12 \end{pmatrix} \rightarrow \text{PdV}$$

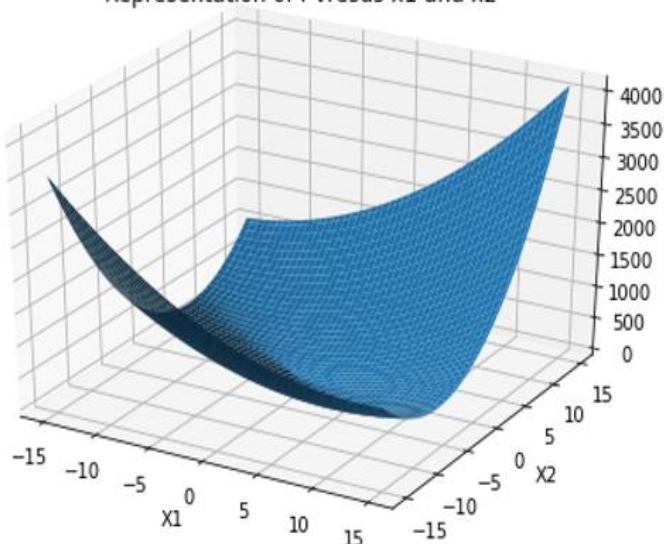
$$\rightarrow \begin{pmatrix} -2/4 \\ 1/4 \end{pmatrix} \xrightarrow{\text{is minimum}}$$

شکل ۱-۲: محاسبات بخش د

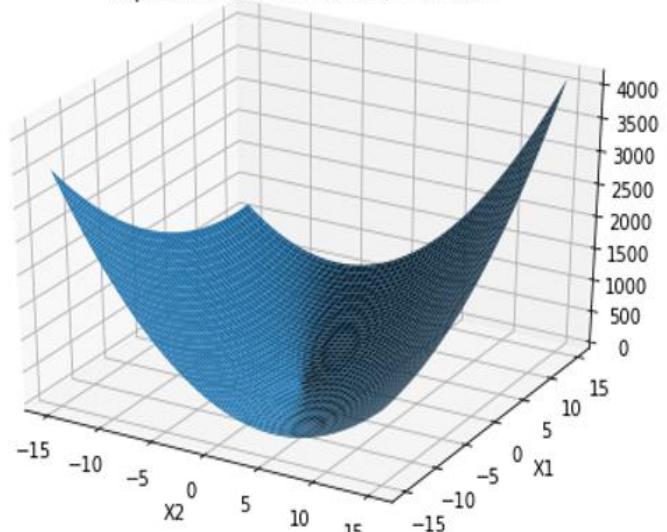
ب) (شبیه سازی بخش ب)

در این قسمت قرار است، تابع $\varphi(\alpha^k) = f(x^k - \alpha^k \nabla f(x^k))$ را در هر مرحله (k) مینیمم کنیم و α بهینه را پیدا کنیم. نمای کلی از تابع را در دو جهت مختلف به صورت زیر در قالب سه بعدی می‌بینیم.

Representation of f versus x1 and x2



Representation of f versus x1 and x2



شکل ۱-۳: نمای کلی از تابع از دو جهت مختلف

در ادامه تابع $\varphi(\alpha^k)$ و $\varphi'(\alpha^k)$ را بصورت پارامتری بدست می‌آوریم :

```
[22] phi(x_sym,y_sym,alfa)
```

$$-12a(6x + 6y + 12) - 8a(6x + 16y + 8) + 12x + 8y + (-6a(6x + 6y + 12) + 6x)(-a(6x + 16y + 8) + y) + 3(-a(6x + 6y + 12) + x)^2 + 8(-a(6x + 16y + 8) + y)^2$$

شکل 4-1 : تابع $\varphi(\alpha^k)$ بصورت پارامتری

```
[24] f_prim=sym.simplify(sym.diff(phi(x_sym,y_sym,alfa),alfa))
f_prim
```

$$1224ax^2 + 5088axy + 3840ax + 5464ay^2 + 7840ay + 3040a - 72x^2 - 264xy - 240x - 292y^2 - 400y - 208$$

شکل 4-5 : تابع $\varphi'(\alpha^k)$ بصورت پارامتری

برای مثال ، با جایگذاری مقدار (1,1) در تابع $\varphi'(\alpha^k)$ و حل معادله آن به جواب زیر می‌رسیم :

```
[26] eq1=f_prim.subs({x_sym: 1, y_sym:1})
eq1
```

$$26496a - 1476$$

```
[30] float(nsolve((eq1), (alfa), [0.5]))
```

$$0.05570652173913043$$

شکل 4-6 : تشکیل معادله پارامتری بر حسب α برای یک مقدار دلخواه (x,y) و حل معادله

در ادامه با نوشتن تابعی در جهت مخالف گرادیان حرکت کرده و در هر مرحله Learning rate را بروز رسانی می‌کنیم .
تابع خطا را بصورت $|x^{k+1} - x^k|$ تعریف می‌کنیم و در تعداد 9 epoch همگرا می‌شویم .

```
[43] IN_2,Res_2=grad_descent_with_Analytical(aggregate, np.array([1,1]),9)
```

```
Starting point at [1 1]
Optimum learning rate in epoch 1 is 0.05570652173913043 at the point [-0.33695653 -0.6711956 ]
Optimum learning rate in epoch 2 is 0.24698787128455893 at the point [-1.8068029  0.5046813]
Optimum learning rate in epoch 3 is 0.055706493771754136 at the point [-2.040061   0.21310887]
Optimum learning rate in epoch 4 is 0.24698459069089296 at the point [-2.2965019  0.41826022]
Optimum learning rate in epoch 5 is 0.05570694061838036 at the point [-2.3371985  0.3673913]
Optimum learning rate in epoch 6 is 0.2471988795182074 at the point [-2.3819802  0.40321803]
Optimum learning rate in epoch 7 is 0.05566756272401434 at the point [-2.3890736  0.3943331]
Optimum learning rate in epoch 8 is 0.25612745098039214 at the point [-2.397156   0.40076512]
Optimum learning rate in epoch 9 is 0.05464824120603015 at the point [-2.3983393  0.39916363]
Ending point after 9 epochs is [-2.3983393  0.39916363]
```

```
[46] print(f(IN_2))
```

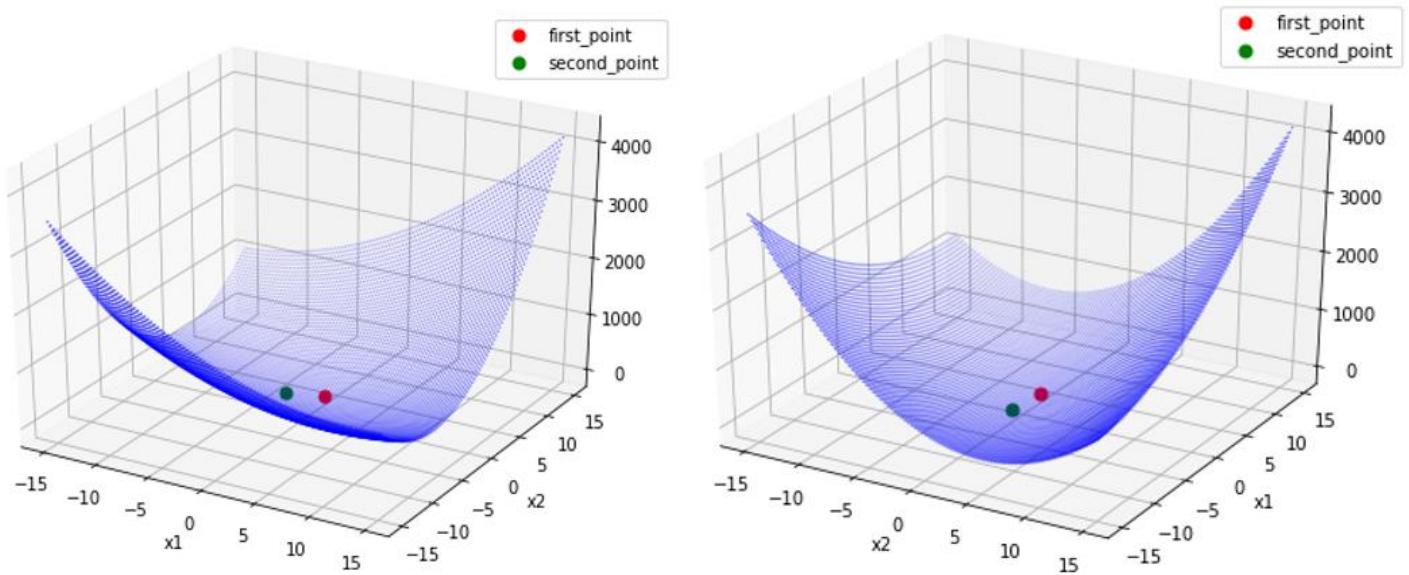
```
print(f(Res_2))
```

37

-12.799994463737782

شکل 4-7 : شروع از نقطه‌ی (1,1) و همگرایی به نقطه‌ی [-2.399,0.399] بعد از 9 تکرار

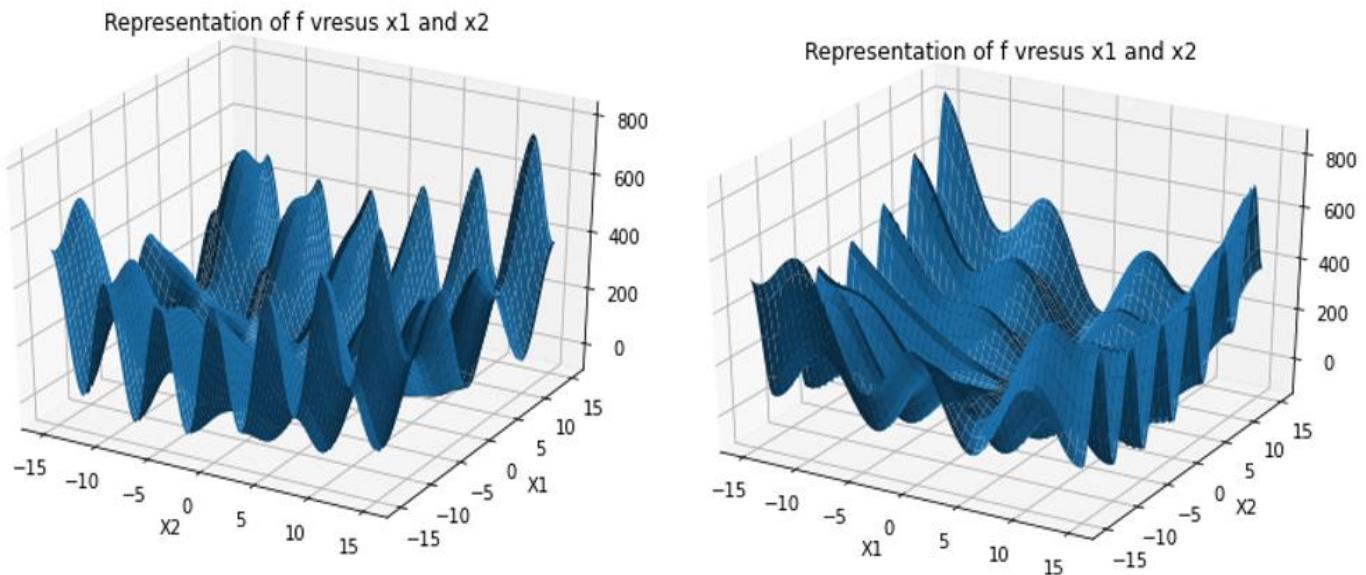
در ادامه نقاط ابتدایی و انتهایی حاصل از همگرایی را در دو حالت مختلف بصورت سه بعدی نشان می‌دهیم :



شکل 1-8 : نمایی سه بعدی از نقاط انتخابی اولیه و نقاط همگرا شده از دو زاویه

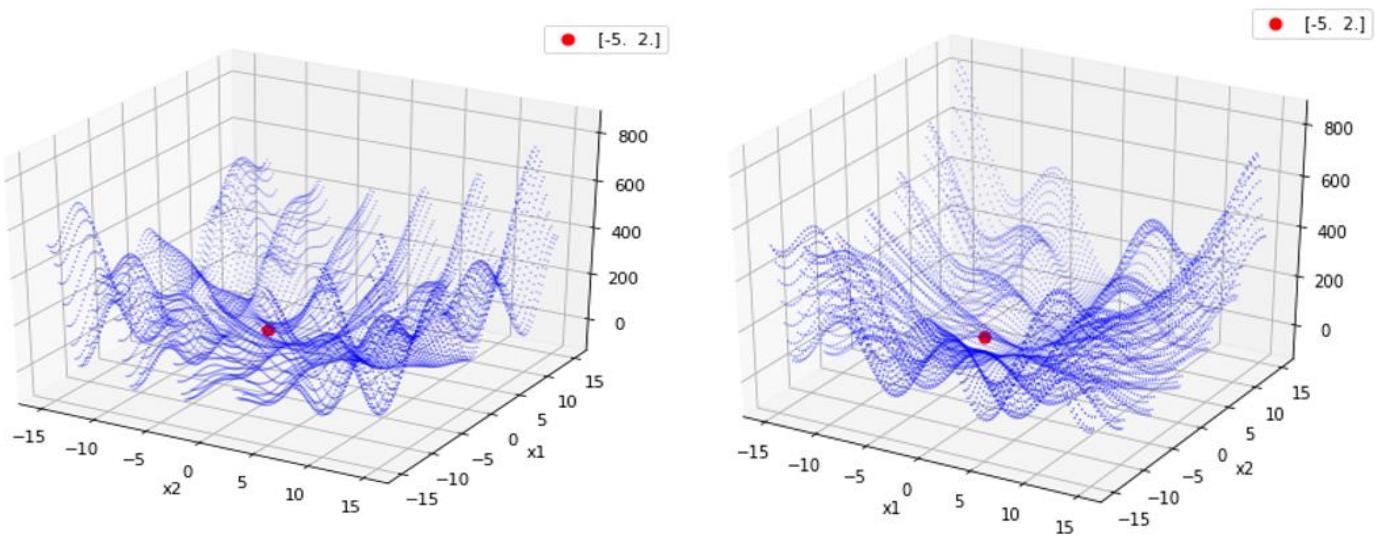
سوال 2

(الف) در ابتدا ، نمایی سه بعدی از تابع غیر محدب داده شده از دو زاویه مختلف را در زیر رسم می کنیم :



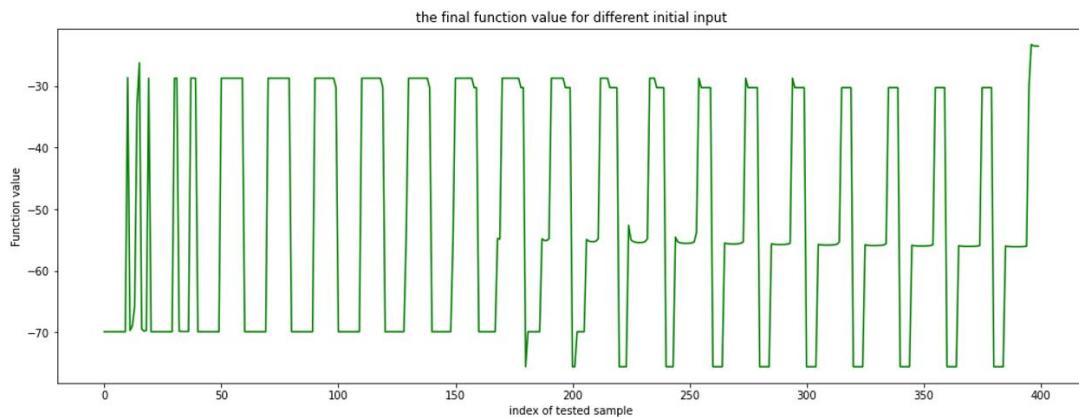
شکل 1-2 : نمایی کلی از تابع از دو جهت مختلف

برای مثال نحوهی قرارگیری نقطهی $(-5, 2)$ - $(5, 2)$ در فضای سه بعدی تابع داده شده را در زیر می بینیم :



شکل 2-1-2 : نحوه قرارگیری یک نقطه‌ی دلخواه در فضای سه بعدی تابع داده شده

از آنجا که نقاط اولیه ، نقش مهمی در گیرافتادن الگوریتم GD در مینیمم های محلی دارد ، پس انتخاب درست آنها با ارزش است . در زیر برای 400 نمونه نقاط اولیه مختلف در بازه‌ی (-5,5) مقادیر نهایی تابع حاصل از اجرای الگوریتم را حساب کرده و رسم می‌کنیم . طبق گفته‌ی سوال مینیمم اصلی تابع مقدار 75.6- دارد .



شکل 3-1-3 : مقادیر تابع موردنظر به ازای انتخاب نقاط اولیه در روش GD و تعداد تکرار 20

همانطور که از شکل بالا مشخص است ، مقدار مینیمم های محلی در حدود مقادیر 30- ، 55- و 70- دارد ، که در بعضی از نقاط اولیه در این مینیمم ها تابع گیر می‌افتد.

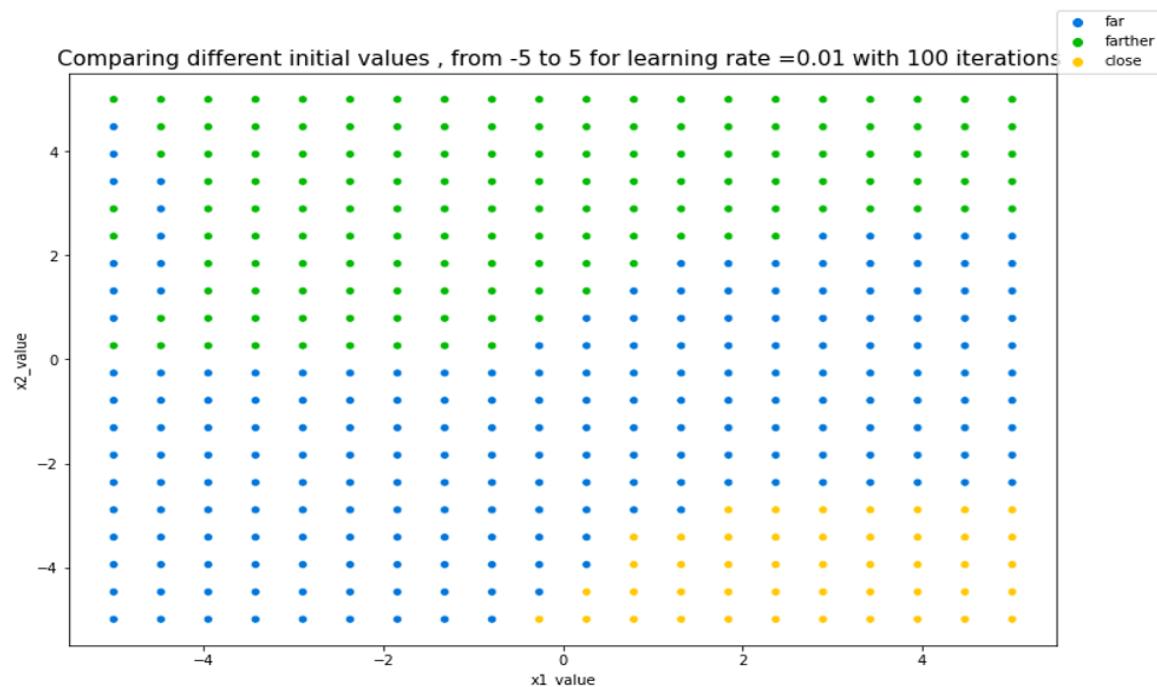
برای اینکه به درک بهتری از مقادیر این نقاط اولیه داشته باشیم ، تابعی می‌نویسیم که به ازای مقدار نهایی تابع ، سه حالت مختلف را برگرداند :

.1. **Close** : در صورتی که مقدار تابع کمتر یا مساوی 70- باشد ، این مقدار برگردانده می‌شود.

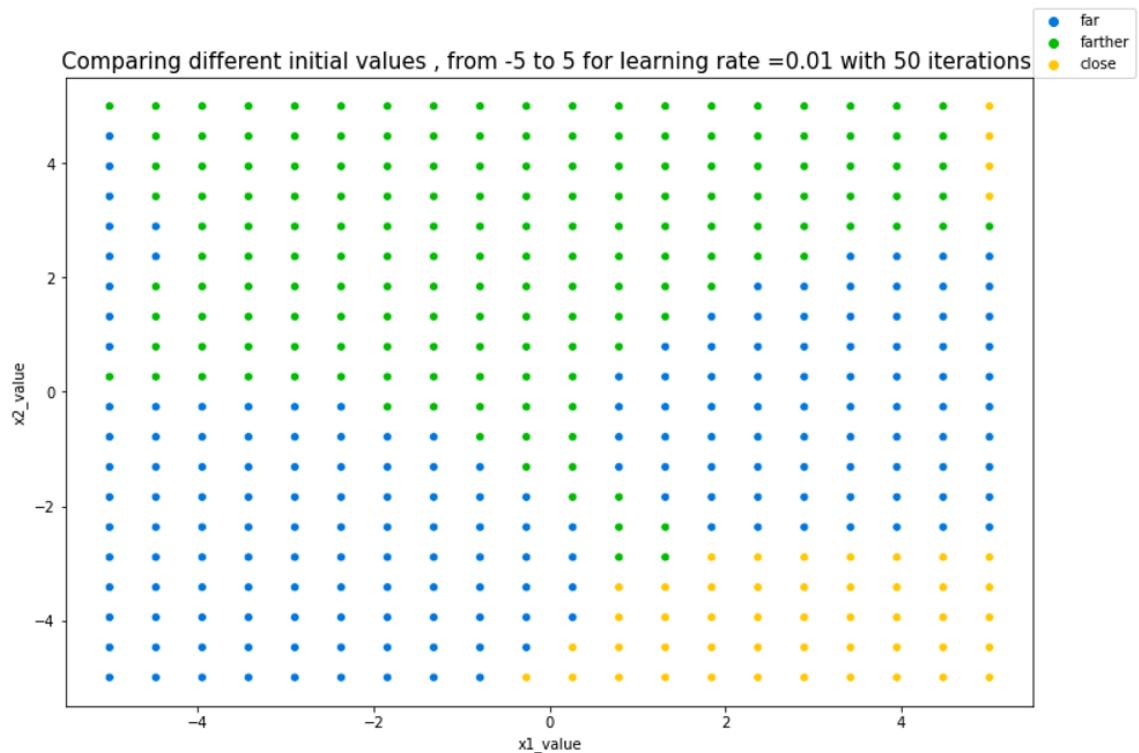
.2. **far** : در صورتی که مقدار تابع بین 70- تا 50- باشد ، این مقدار برگردانده شود.

.3. **farther** : در صورتی که مقدار بیشتر از 50- باشد ، این مقدار برگردانده می‌شود.

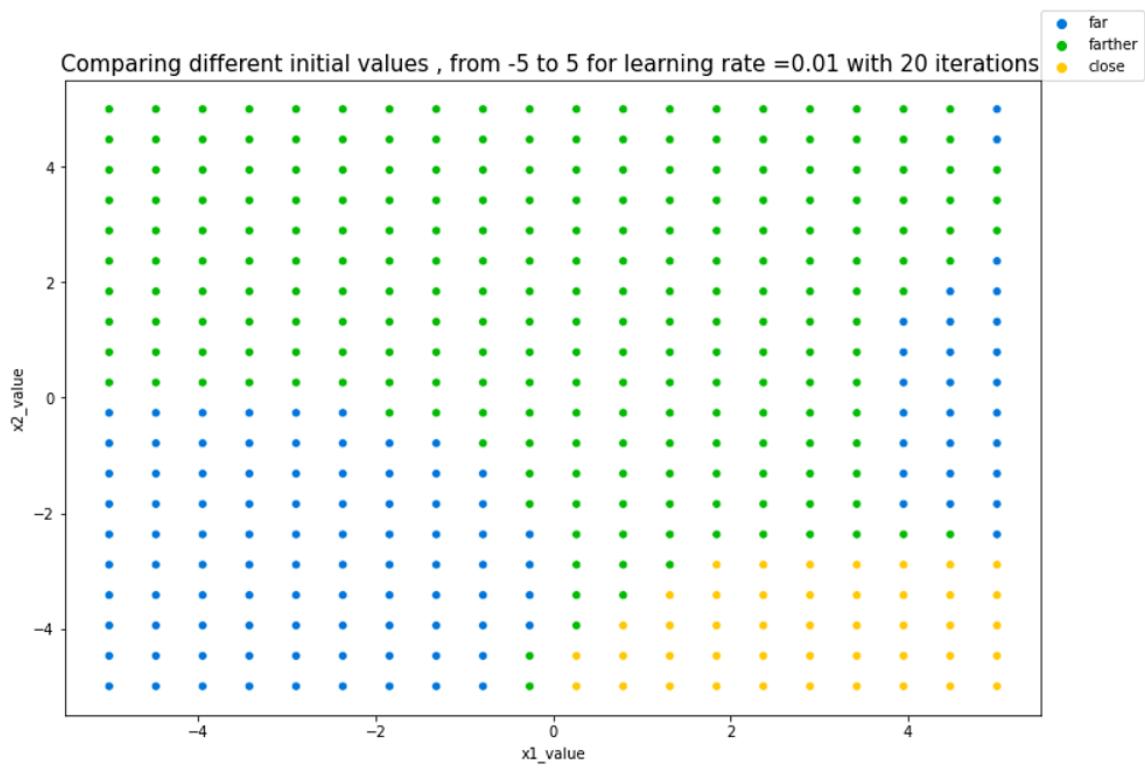
در سه شکلی که در ادامه آمده است، تاثیر مقادیر اولیه به همراه سه مختلف Iteration (20, 50, 100) در مقدار نهایی بدست آمده درتابع آورده شده است.



شکل 2-1-4 : تقسیم بندی نقاط اولیه در بازه‌ی [-5,5] به سه قسمت مختلف برای 100 تکرار



شکل 2-1-5 : تقسیم بندی نقاط اولیه در بازه‌ی [-5,5] به سه قسمت مختلف برای 50 تکرار

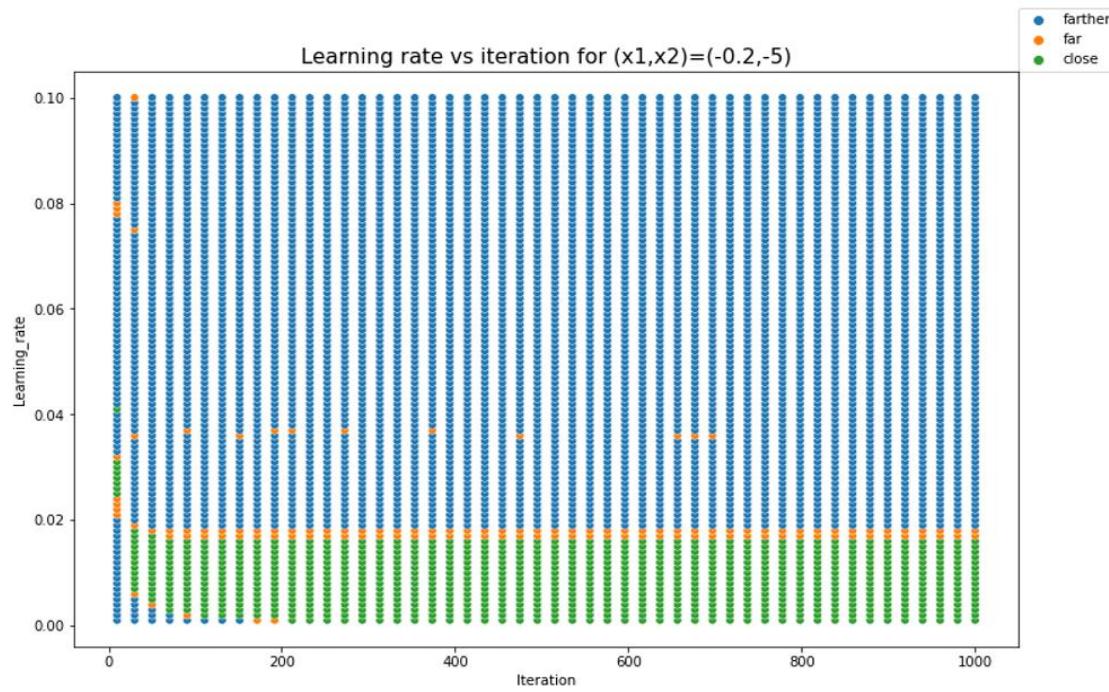


شکل ۱-۶-۲: تقسیم بندی نقاط اولیه در بازه‌ی [-۵,۵] به سه قسمت مختلف برای ۲۰ تکرار

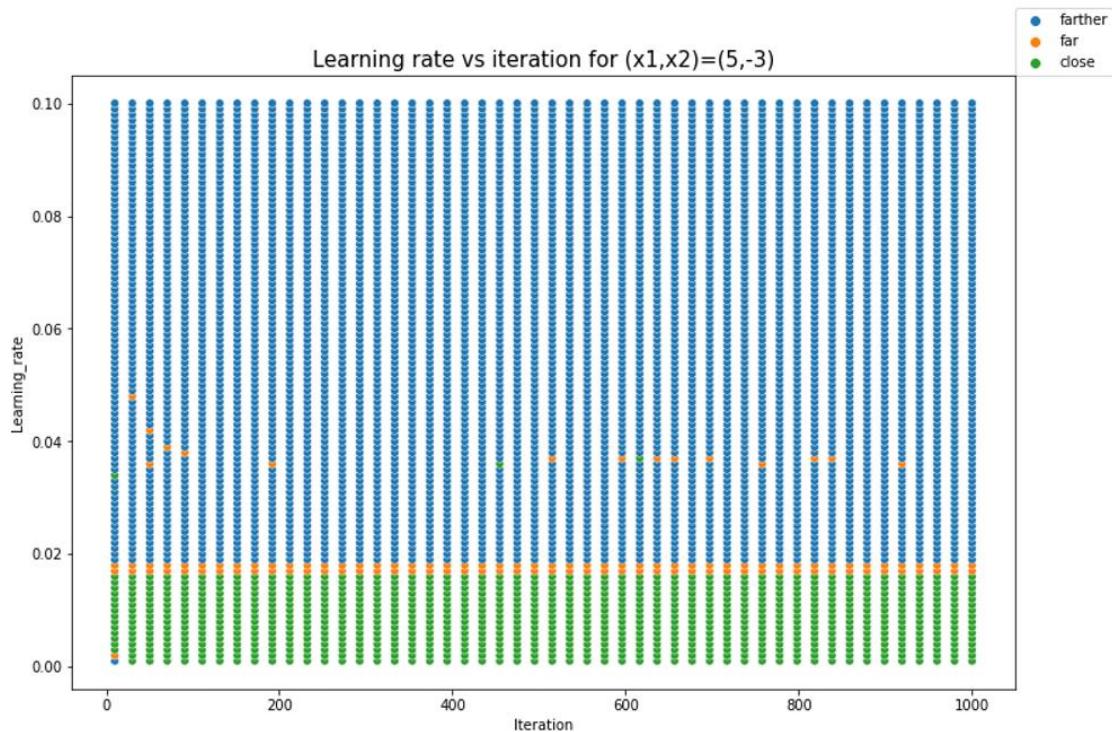
از خروجی‌های بدست آمده در بالا دو نتیجه‌ی مهم می‌توان گرفت:

- ⌚ با کاهش تعداد تکرارها (Iterations)، تعداد نقاط سبز (farther) (نقاطی که مینیمم محلی بزرگتری دارند) بیشتر شده و تعداد نقاط آبی (far) (نقاطی که مینیمم محلی به نسبت کمتری دارند) کاهش می‌یابد ولی تعداد نقاط نارنجی (close) (نقاطی با مینیممی بسیار نزدیک به مینیمم اصلی تابع) ثابت می‌ماند. این به این معنا است که در صورت انتخاب این نقاط در تعداد تکرار بسیار کمتری همگرا می‌شویم.
- ⌚ میتوان گفت با انتخاب نقاط اولیه مطلوب ما (نقاط نارنجی رنگ) برای x_1 در بازه‌ی [2,5] و برای x_2 در بازه‌ی [-5,3] همیشه به حدود مینیمم اصلی تابع 75.6- همگرا می‌شویم.

همچنین در نمودار بالا می‌توان جای نقاط اولیه و تکرارها (Iterations) را عرض کرد و تحلیل دیگری داشت. ما برای دو نقطه‌ی (-0.2, -5) و (5, -3) که جزو نقاط آبی رنگ می‌باشد، نمودارهای زیر را رسم می‌کنیم:



شکل 2-1-7 : رابطه‌ی بین تعداد تکرارها(Learning rate) و نرخ یادگیری(Iterations) به ازای مقدار اولیه $(-0.2, -5)$



شکل 2-1-8 : رابطه‌ی بین تعداد تکرارها(Learning rate) و نرخ یادگیری(Iterations) به ازای مقدار اولیه $(5, -3)$

*طبق نتایج بالا ، برای نقطه‌ی (-5, -0.2) ، در 200 تکرار (Iteration) اول ، برای اینکه به مینیمم اصلی همگرا شویم (نقطه سبز) ، با افزایش تکرارها ، بازه‌ی قابل قبول برای نرخ یادگیری کاهش می‌یابد . همچنانی با افزایش تکرارها تعداد نقاط نارنجی (far) کاهش می‌یابد تا جایی که تقریباً برای حدود تکرار 200 به بعد و برای Learning rate های کمتر از 0.02 به مینیمم اصلی همگرا می‌شویم .

*در شکل دوم برای نقطه‌ی (5, -3) ، تقریباً به جز چند Iteration اول ، برای تمامی Iteration های دیگر و نرخ‌های یادگیری کمتر از 0.02 به مینیمم اصلی تابع همگرا می‌شویم .

ب) در این قسمت می‌خواهیم برای یک نقطه اولیه مشخص (در اینجا (10,7)) ، دو الگوریتم تحلیلی و قاعده‌ی Armijo را اجرا کنیم و نتایج را تحلیل کنیم .

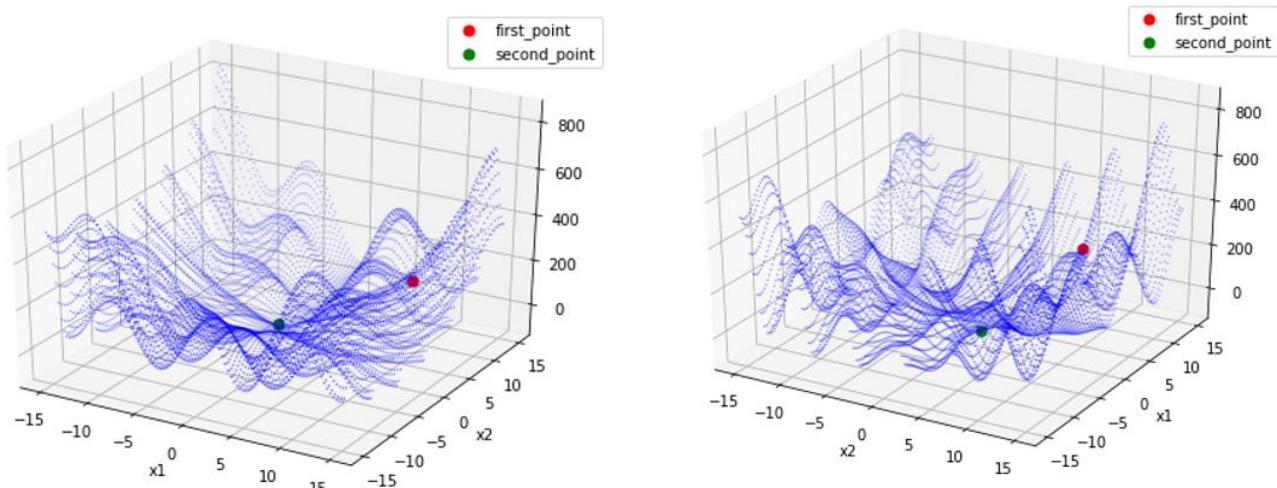
ب-1) در زیر ابتدا به کمک قاعده‌ی Armijo و $\beta = 0.5$ ، $\sigma = 1e - 4$ و 5 تکرار، الگوریتم را اجرا می‌کنیم . نتایج در زیر آورده شده است :

```
Starting point at [10 7]
Optimum learning rate in epoch 1 is 0.125 at the point [10. 7.]
Optimum learning rate in epoch 2 is 0.125 at the point [ 5.9830933 -7.349364 ]
Optimum learning rate in epoch 3 is 0.25 at the point [-0.69719076 -3.8783956 ]
Optimum learning rate in epoch 4 is 0.03125 at the point [-2.330353  3.5675075]
Optimum learning rate in epoch 5 is 0.03125 at the point [-1.5947603  2.0410316]
Ending point after 5 epochs is [-1.5503504  2.5940385]
```

```
[237] print(f(IN_1))
print(f(Res_1))
200.3525491562421
-28.53638193727287
```

شکل 2-2-1 : روشن Armijo برای تعیین نرخ یادگیری بهینه

همانطور که از نتایج پیداست ، بعد از 5 تکرار ، به نقطه‌ی (-1.55, 2.59) می‌رسیم که مقدار تابع در آن برابر -28.54 می‌باشد. بنابراین در جهت مناسبی حرکت کرده‌ایم و روش با تعداد تکرار بیشتر به شرط گریز از مینیمم محلی همگرایی دارد. در ادامه نقاط ابتدایی و انتهایی حاصل از همگرایی را در دو حالت مختلف بصورت سه بعدی نشان می‌دهیم :



شکل 2-2 : نمایی سه بعدی از نقاط انتخابی اولیه و نقاط همگرا شده از دو زاویه

ب-2 در این قسمت قرار است تابع $\varphi(\alpha)$ را بصورت پارامتری تشکیل داده و سپس با حل مشتق آن برابر صفر ، مقدار بهینه‌ی α را در هر epoch پیدا کنیم .ابتدا مشتق های جزئی تابع f را نسبت به x_1 و x_2 به ترتیب در زیر حساب می‌کنیم :

```
[20] sym.diff(f_sym(x_sym,y_sym),x_sym)
```

$$2x + 2.0\pi y \sin(0.2\pi x) - 15 \cos(0.4\pi y)$$

```
[21] sym.diff(f_sym(x_sym,y_sym),y_sym)
```

$$6.0\pi x \sin(0.4\pi y) + 2y - 10 \cos(0.2\pi x)$$

شکل 3-2-3: مشتق جزئی تابع f نسبت به دو متغیر موجود

در ادامه ، تابع $\varphi(\alpha)$ را بصورت پارامتری تشکیل می‌دهیم :

```
def phi(x_sym,y_sym,alfa):
    return f_sym(x_sym-alfa*sym.diff(f_sym(x_sym,y_sym),x_sym),y_sym-alfa*sym.diff(f_sym(x_sym,y_sym),y_sym))

phi(x_sym,y_sym,alfa)

- (-15a (2x + 2.0\pi y \sin(0.2\pi x) - 15 \cos(0.4\pi y)) + 15x) \cos(\pi (-0.4a (6.0\pi x \sin(0.4\pi y) + 2y - 10 \cos(0.2\pi x)) + 0.4y)) +
(-a (2x + 2.0\pi y \sin(0.2\pi x) - 15 \cos(0.4\pi y)) + x)^2 -
(-10a (6.0\pi x \sin(0.4\pi y) + 2y - 10 \cos(0.2\pi x)) + 10y) \cos(\pi (-0.2a (2x + 2.0\pi y \sin(0.2\pi x) - 15 \cos(0.4\pi y)) + 0.2x)) +
(-a (6.0\pi x \sin(0.4\pi y) + 2y - 10 \cos(0.2\pi x)) + y)^2
```

شکل 4-2-4 : تابع $\varphi(\alpha)$ بصورت پارامتری

در نهایت ، از تابع $\varphi(\alpha)$ بر حسب α مشتق می‌گیریم و $\varphi'_{(\alpha)}$ را تشکیل می‌دهیم :

```
[26] f_prim=sym.simplify(sym.diff(phi(x_sym,y_sym,alfa),alfa))
f_prim
```

```
(a (2x + 2.0\pi y \sin(0.2\pi x) - 15 \cos(0.4\pi y)) - x) (4x + 4.0\pi y \sin(0.2\pi x) - 30 \cos(0.4\pi y)) -
15\pi (a (2x + 2.0\pi y \sin(0.2\pi x) - 15 \cos(0.4\pi y)) - x) (2.4\pi x \sin(0.4\pi y) + 0.8y - 4.0 \cos(0.2\pi x)) \sin(\pi (0.4a (6.0\pi x \sin(0.4\pi y) + 2y - 10 \cos(0.2\pi x)) - 0.4y)) -
10\pi (a (6.0\pi x \sin(0.4\pi y) + 2y - 10 \cos(0.2\pi x)) - y) (0.4x + 0.4\pi y \sin(0.2\pi x) - 3.0 \cos(0.4\pi y)) \sin(\pi (0.2a (2x + 2.0\pi y \sin(0.2\pi x) - 15 \cos(0.4\pi y)) - 0.2x)) +
(a (6.0\pi x \sin(0.4\pi y) + 2y - 10 \cos(0.2\pi x)) - y) (12.0\pi x \sin(0.4\pi y) + 4y - 20 \cos(0.2\pi x)) +
(30x + 30.0\pi y \sin(0.2\pi x) - 225 \cos(0.4\pi y)) \cos(\pi (0.4a (6.0\pi x \sin(0.4\pi y) + 2y - 10 \cos(0.2\pi x)) - 0.4y)) +
(60.0\pi x \sin(0.4\pi y) + 20y - 100 \cos(0.2\pi x)) \cos(\pi (0.2a (2x + 2.0\pi y \sin(0.2\pi x) - 15 \cos(0.4\pi y)) - 0.2x))
```

شکل 5-2-5 : تابع $\varphi'_{(\alpha)}$ بصورت پارامتری

حال کافی است ، با حل معادله‌ی پارامتری بالا برای هر x_1 و x_2 ، α را پیدا کنیم . ولی باید توجه کرد که طبق معادله‌ی بدست آمده در بالا ، ترم‌های کسینوسی و سینوسی وجود دارد که برای یک جفت (x_1, x_2) ، بیشمار جواب وجود دارد و در واقع با استفاده از nsolve در sympy و دادن نقاط اولیه برای شروع حل معادله ، به جوابهای متفاوتی مرسیم . بنابراین انتخاب نقاط اولیه برای حل معادله بسیار مهم بوده و امکان دارد با انتخاب یک نقطه‌ی اولیه اصلاً جواب نداشته باشیم ! و یا همچنین به مقدار α منفی به عنوان جواب برسیم ، که این هم قابل قبول نیست .

برای این منظور تابعی نوشتم (Sel_start_a) ، که با گرفتن نقطه‌ی اولیه و بررسی وجود داشتن جواب و همچنین مثبت بودن α مقدار کوچکی در 1e-3 order به مقدار اولیه اضافه می‌کند تا معادله جواب قابل داشته باشد .

برای مثال برای مقدار اولیه $\alpha = 0.05$ برای α جواب نخواهیم داشت ولی با تغییر α اولیه به 3×10^{-3} دارای جواب قابل قبول خواهیم بود.

در نهایت با اجرای الگوریتم GD با کمک روش تحلیلی به نتایج زیر می‌رسیم :

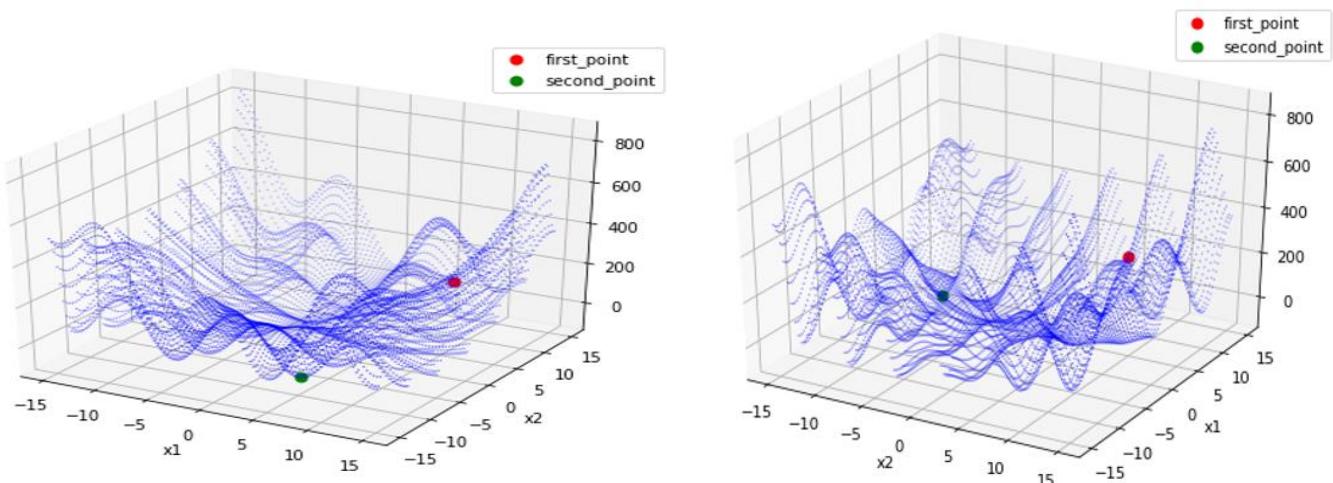
```
Starting point at [10 7]
Optimum learning rate in epoch 1 is 0.14750244724195194 at the point [ 5.259971 -9.93253 ]
Optimum learning rate in epoch 2 is 0.021612263736707354 at the point [ 5.136294 -9.897908 ]
Optimum learning rate in epoch 3 is 0.008496833059534935 at the point [ 5.130218 -9.919612 ]
Optimum learning rate in epoch 4 is 0.02163245996749606 at the point [ 5.120901 -9.917005 ]
Optimum learning rate in epoch 5 is 0.008514065295673143 at the point [ 5.120457 -9.91859 ]
Ending point after 5 epochs is [ 5.120457 -9.91859 ]
```

```
print(f(IN_2))
print(f(Res_2))

200.3525491562421
-50.70972630246388
```

شکل 2-2-6 : GD با روش تحلیلی برای تعیین نرخ یادگیری بهینه

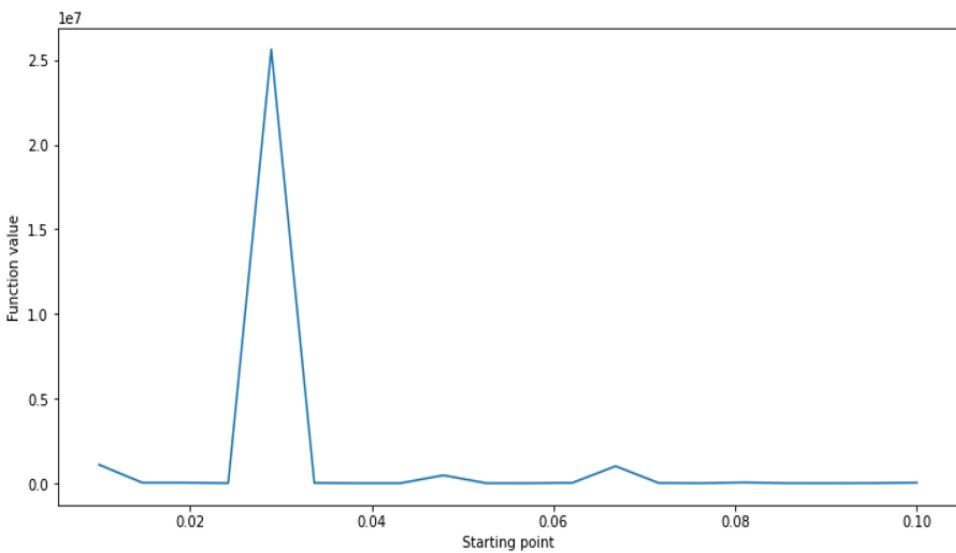
همانطور که از نتایج پیداست ، بعد از 5 تکرار ، به نقطه‌ی (5.12,-9.92) می‌رسیم که مقدار تابع در آن برابر -50.71 می‌باشد. بنابراین در جهت مناسبی حرکت کرده‌ایم و روش با تعداد تکرار بیشتر به شرط گریز از مینیمم محلی همگرایی دارد. در ادامه نقاط ابتدایی و انتهایی حاصل از همگرایی را در دو حالت مختلف بصورت سه بعدی نشان می‌دهیم :



شکل 2-2-7 : نمایی سه بعدی از نقاط انتخابی اولیه و نقاط همگرا شده از دو زاویه

طبق نتایج ، با نقطه‌ی شروع یکسان ، روش تحلیلی در تعداد تکرار یکسان به مقدار کوچکتری نسبت به روش Armijo رسید . در واقع روش تحلیل دقیق تر خواهد بود ولی حجم محاسباتی بیشتری دارد . علاوه بر این ما در روش تحلیلی برای حل معادله نیاز به نقطه‌ی اولیه داریم که این هم برای ما مشکل ساز است ، چون جواب یکتا نداریم . الگوریتم Armijo ساده تر است و با کمی تکرار بیشتر میتواند مثل روش تحلیل دقیق باشد .

برای اینکه یک تخمینی از نقطه‌ی اولیه برای حل معادله‌ی $\varphi'(\alpha)$ مساوی صفر داشته باشیم ، نمودار زیر را برای $\alpha \in [0.001, 0.1]$ و خروجی نهایی حاصل از همگرایی رسم می‌کنیم :



شکل 8-2-8 : خروجی نهایی حاصل از همگرایی برای نقاط اولیه مختلف برای حل معادله $\varphi'_{\alpha=0}$

همانطور که در شکل دیده می‌شود ، برای $\alpha \in [0.02, 0.04]$ تابع پیک‌های عجیبی می‌خورد که در واقع برای زمانی است که مقدار α بهینه منفی در آمده است و تابع در هر epoch در سمت مخالف گرادیان ، بزرگ و بزرگتر شده است .

ج) در این قسمت ، ما از الگوریتم Simulated annealing به عنوان روش فرالبتکاری استفاده می‌کنیم . این روش سه پارامتر مهم دارد(با فرض ثابت بودن Temperature اولیه) که در همگرایی به مینیمم اصلی تاثیر گذار هستند :

- ✓ ضریب A
- ✓ تعداد Iteration داخلی (Inner_c)
- ✓ دمای نهایی (T-final)

مزیت این روش نسبت به روش‌های دیگر این است که با تنظیم این Hyper parameter ها ، می‌توانیم از دام مینیمم های محلی تا حدی خوبی فرار کنیم .

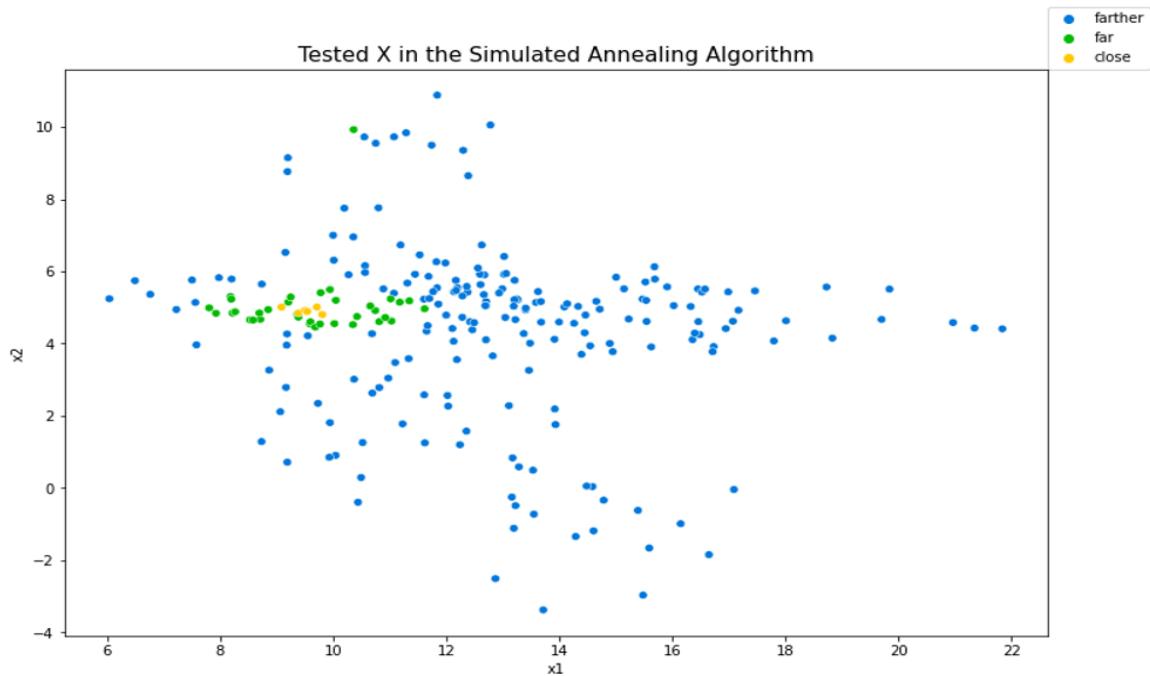
در ابتدا با بازی با این پارامترها ، به مقدار مینیمم همگرا می‌شویم و سپس ترکیب مناسبی از این سه پارامتر ارائه می‌دهیم که همواره به مینیمم اصلی همگرا شویم . با اجرای الگوریتم برای $A=1.8$ ، $Inner_c=5$ ، $T_final=1$ و برای $x_0=[10, 7]$ تکرار خواهیم داشت :

```
Res,X=SA(f,np.array([10,7]),1000,1,1.8,5,100)
print("xopt= ",Res)
print("ALL tested x : \n{}".format(X))
xopt= [9.71344831 5.01303304] [221] f(Res)
-75.52060464297278
```

شکل 9-2-2: نقطه‌ی همگرایی و مقدار نهایی حاصل از همگرایی در روش SA

*همانطور که می‌بینید به مینیمم اصلی تابع همگرا شده‌ایم .

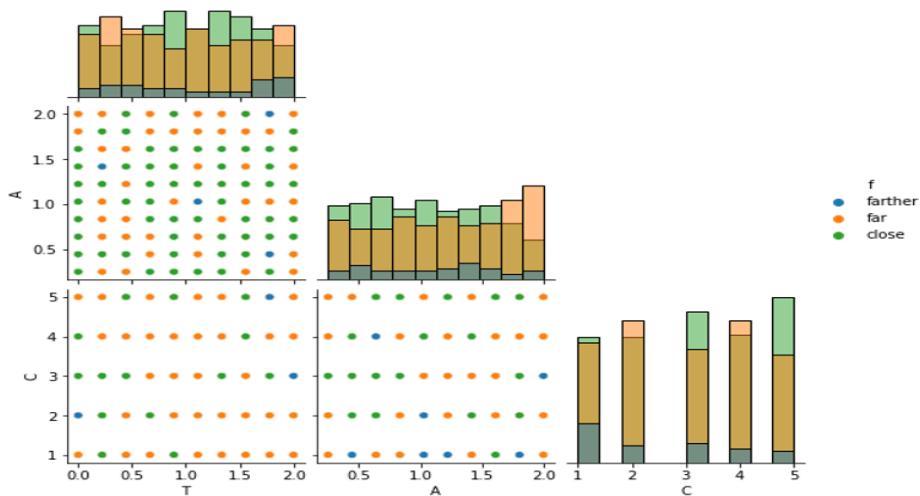
در زیر، تمامی زوج (x_1, x_2) هایی که در روش SA تا رسیدن به همگرایی تست شده‌اند را می‌بینیم:



شکل 10-2-10 : نقاط تست شده در فرآیند همگرایی Simulated Annealing

همانطور که در شکل بالا معلوم است ، نقاط با وضعیت بهتر (مینیمم محلی کمتر) که با رنگ های سبز (far) و نارنجی (close) مشخص سده‌اند ، در یک همسایگی یا ابر چگالی مرکز قرار گرفته‌اند.

برای اینکه تخمینی از ترکیب سه Hyper parameter مطرح شده در بالا داشته باشیم ، با تغییر پارامتر T-final از 0 تا 2 ، پارامتر A از 0.25 تا 10 و پارامتر c Inner- c از 1 تا 5 ، مقادیر نهایی همگرا شده را مانند قبل به سه قسمت تقسیم می‌کنیم :



شکل 10-2-11 : ترکیب سه پارامتر روش SA در مقدار نهایی همگرایی

طبق نتایج بدست آمده در بالا ، به سادگی می توانیم با انتخاب ترکیب مناسبی از سه پارامتر A,C,T برای مثال (0.6,5,0.6) به مینیمم اصلیتابع همگرا شویم :

```
[259] Res,X=SA(f,np.array([10,7]),1000,0.6,0.6,5,100)
```

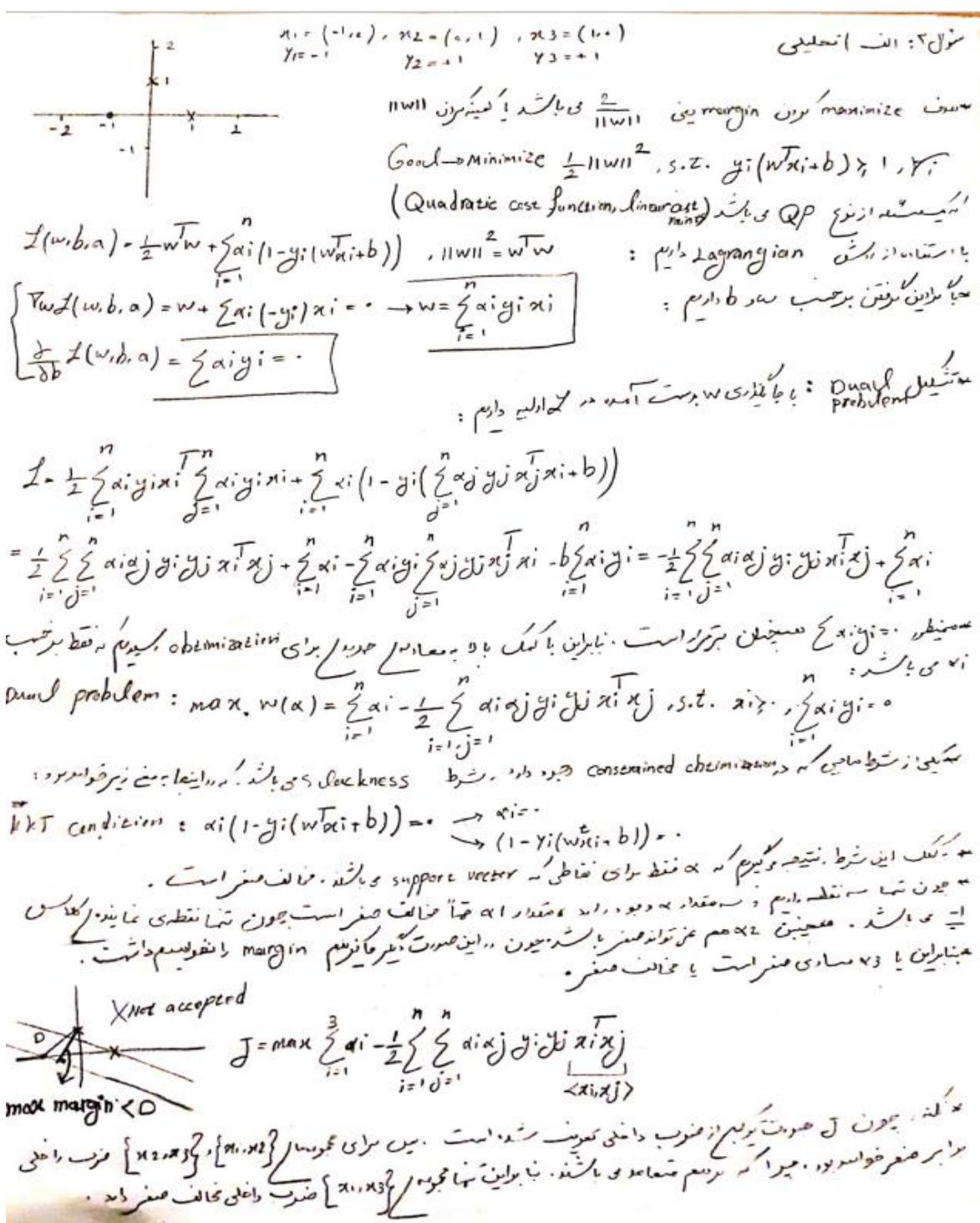
▶ f(Res)

↳ -75.44936893837409

شكل 2-12 : انتخاب مناسب Hyper parameter ها برای همگرایی مطلوب

سوال 3

الف) محاسبات تحليلي



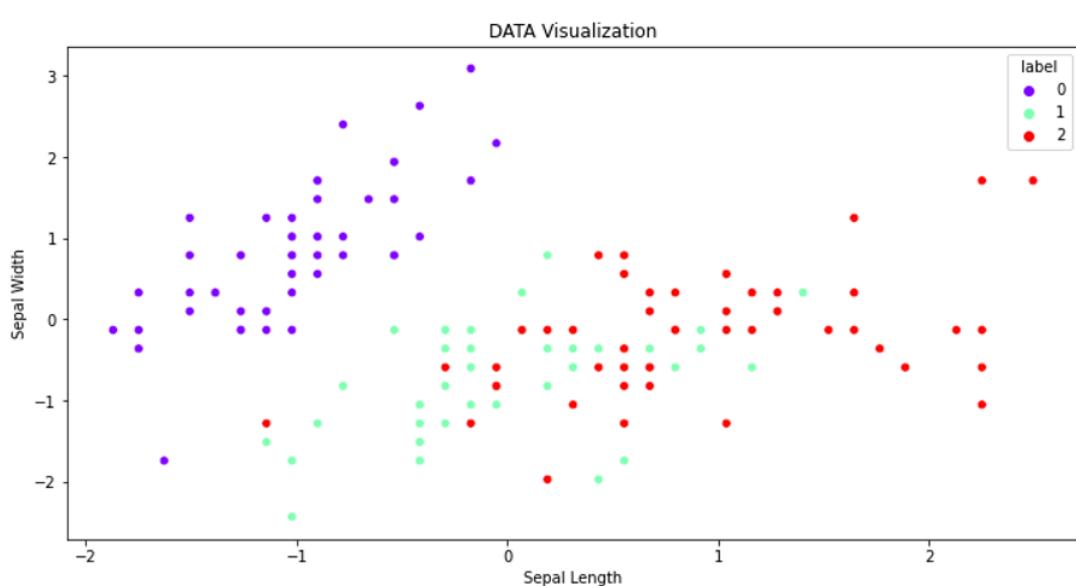
شکل ۱-۱-۳: محاسبات مربوط به برداشتیان قسمت الف

$$\begin{aligned}
 & \rightarrow J = \alpha_1 + \alpha_2 + \alpha_3 - 2 \times \frac{1}{2} (\alpha_1 \underbrace{\alpha_3 y_i}_{-1} \underbrace{y_i x_i^T}_{-1} \alpha_3) - \frac{1}{2} \sum_{i=1}^3 \alpha_i^2 \underbrace{\|x_i\|^2}_{\alpha_1 = \sqrt{2 + \alpha_3}} \rightarrow -\alpha_1 + \alpha_2 + \alpha_3 = \\
 & \rightarrow J = 2(\alpha_2 + \alpha_3) - (\alpha_2 + \alpha_3)\alpha_3 - \frac{1}{2} (\alpha_2^2 + \alpha_3^2 + (\alpha_2 + \alpha_3)^2) = 2\alpha_2 + 2\alpha_3 - \alpha_2\alpha_3 - \frac{1}{2}\alpha_2^2 - \frac{1}{2}\alpha_3^2 - \frac{1}{2}(\alpha_2 + \alpha_3)^2 \\
 & \rightarrow J = 2\alpha_2 + 2\alpha_3 - 2\alpha_2\alpha_3 - \frac{1}{2}\alpha_2^2 - \frac{1}{2}\alpha_3^2 \rightarrow \left\{ \begin{array}{l} \frac{\partial J}{\partial \alpha_2} = 2 - 2\alpha_3 - 2\alpha_2 = 0 \\ \frac{\partial J}{\partial \alpha_3} = 2 - 2\alpha_2 - 4\alpha_3 = 0 \end{array} \right. \\
 & \rightarrow \left\{ \begin{array}{l} 2\alpha_3 + 2\alpha_2 = 2 \\ 2\alpha_2 + 4\alpha_3 = 2 \end{array} \right. \rightarrow \alpha_3 = 0 \quad \left\{ \begin{array}{l} \alpha_1 = 1 \\ \alpha_2 = 1 \end{array} \right. \rightarrow w = \underbrace{\alpha_1 \alpha_1 + \alpha_2 \alpha_2 + \alpha_3 \alpha_3}_{(1, 1, 0)} = (\alpha_1 + \alpha_2, \alpha_2) = (1, 1) \\
 & \rightarrow \alpha_2 = 1 \rightarrow \alpha_1 = 1 \rightarrow w = \underbrace{\alpha_1 \alpha_1 + \alpha_2 \alpha_2 + \alpha_3 \alpha_3}_{(1, 1, 0)} = (\alpha_1 + \alpha_2, \alpha_2) = (1, 1) \\
 & \rightarrow \left\{ \begin{array}{l} \alpha_1 : \alpha_1(1 - y_i(w^T x_i + b)) = 0 \rightarrow 1(1 + 1(-1 + b)) = 0 \rightarrow 1 - 1 + b = 0 \rightarrow b = 0 \\ \alpha_2 : \alpha_2(1 - y_i(w^T x_i + b)) = 0 \rightarrow 1(1 - 1(1 + b)) = 0 \rightarrow 1 - 1 - b = 0 \rightarrow b = 0 \end{array} \right. \\
 & \text{Diagram: A 2D plot showing three classes of data points (Sepal Length vs Sepal Width). Three parallel lines represent the decision boundaries for } \alpha_1 = 1, \alpha_2 = 1, \text{ and } \alpha_3 = 0. \text{ The lines are: } \\
 & \quad \text{Line 1: } w^T x + b = 1 \rightarrow x_1 + x_2 = 1 \\
 & \quad \text{Line 2: } w^T x + b = 0 \rightarrow x_1 + x_2 = 0 \\
 & \quad \text{Line 3: } w^T x + b = -1 \rightarrow x_1 + x_2 = -1
 \end{aligned}$$

شکل 2-3: محاسبات مربوط به بردار پشتیبان قسمت الف

ب) (پیاده سازی با کتابخانه پایه – استفاده از SGD و SD برای پیاده سازی)

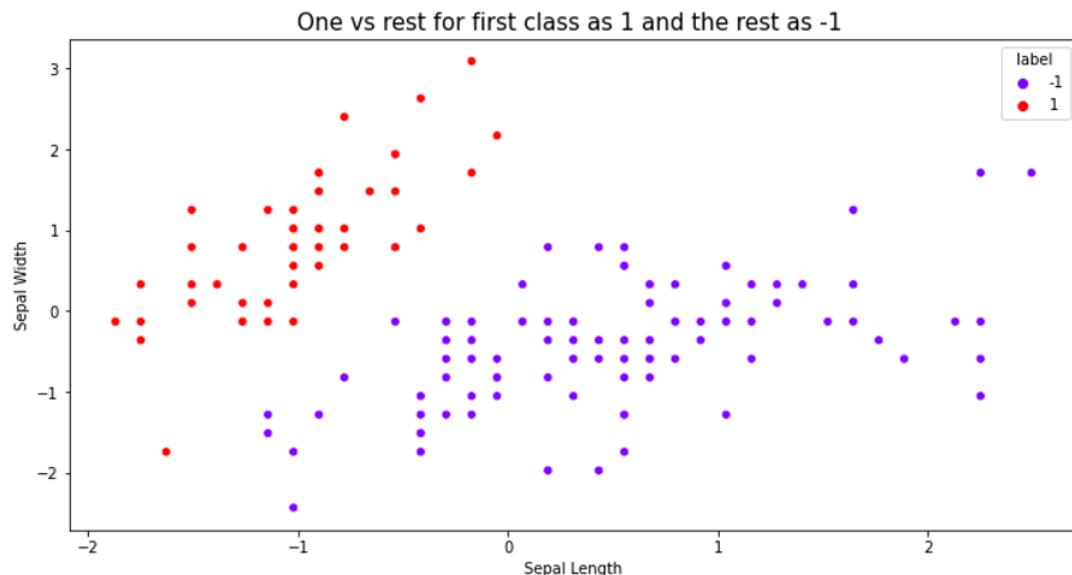
در ابتدا نمایی از داده ها در سه کلاس بر حسب دو ویژگی رسم می کنیم :



شکل 1-3-2: نمایی از سه کلاس و دو ویژگی داده شده دیتاست Iris

طبق خواسته‌ی سوال ، باید از روش one-vs-all یا one-vs-rest برای تفکیک کلاس‌ها استفاده کنیم . به این منظور در هر بار یکی از کلاس‌ها را برابر لیبل 1 و مابقی کلاس‌ها را لیبل -1 می‌دهیم :

لیبل 1 برای کلاس اول ①



شکل 2-3 : نمایی از دو کلاس و دو ویژگی حاصل از one-vs-rest

برای راحتی کار با داده‌ها ، در هر مرحله Dataset از داده‌ها به همراه لیبل‌های جدید assign شده به هر داده تشکیل می‌دهیم .

همچنین یک ستون از عناصر 1 به دلیل ترم بایاس به Dataset اضافه می‌کنیم . نمایی از Dataset برای حالت 1 را در زیر می‌بینیم

	Sepal Length	Sepal Width	bias	label
0	-0.900681	1.019004	1.0	1
1	-1.143017	-0.131979	1.0	1
2	-1.385353	0.328414	1.0	1
3	-1.506521	0.098217	1.0	1
4	-1.021849	1.249201	1.0	1
...
145	1.038005	-0.131979	1.0	-1
146	0.553333	-1.282963	1.0	-1
147	0.795669	-0.131979	1.0	-1
148	0.432165	0.788808	1.0	-1
149	0.068662	-0.131979	1.0	-1

150 rows × 4 columns

شکل 3-3 : نمایی از Dataset تشکیل شده در حالت اول

در ادامه دو تابع به نام lossGradient و loss_all می‌نویسیم ، که در این دو ، تابع خطا و مشتق تابع خطا مربوط به Svm را محاسبه می‌کنیم :

```
[ ] def loss_all(W, x, y, C):
    return 1/2 * np.sum(W**2) + C * np.sum([np.max([0, 1 - y[i] * (W @ x[i])]) for i in range(x.shape[0])])
```

شکل 3-2-4 : تابع هدف در Svm

```
[ ] def lossGradient(W, x, y, C):
    lossGrad = np.zeros(W.shape[0])
    distance = [np.max([0, 1 - y[i] * (W @ x[i])]) for i in range(x.shape[0])]
    #print(distance)
    for i,d in enumerate(distance):
        if d==0:
            lossGrad+=W
        else:
            lossGrad+=W - C * y[i] * x[i]

    return lossGrad/x.shape[0]
```

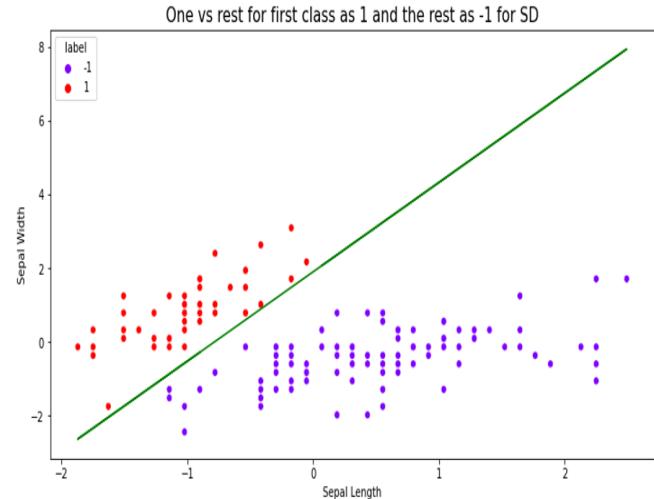
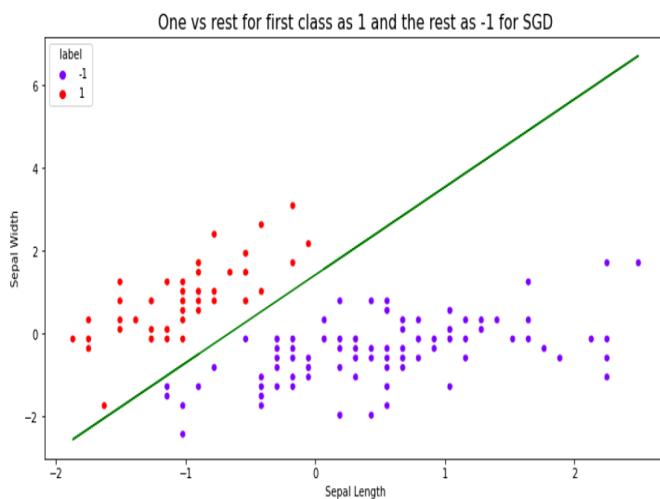
شکل 3-2-5 : مشتق تابع هدف در Svm

همانطور که می‌بینید ، هر دوی این توابع دارای دو ترم اصلی W (ماتریس وزن) و C (ضریبی برای تنظیم میزان Margin بودن Hard margin می‌باشد . در صورتی که C خیلی بزرگ انتخاب شود در هستیم و بر عکس . در تمامی حالات $C=1e5$ در نظر گرفته شده است .)

با اجرای الگوریتم SGD و SD با روش Armijo برای حالت اول ، خطوط جدا کننده به شکل زیر خواهد بود :

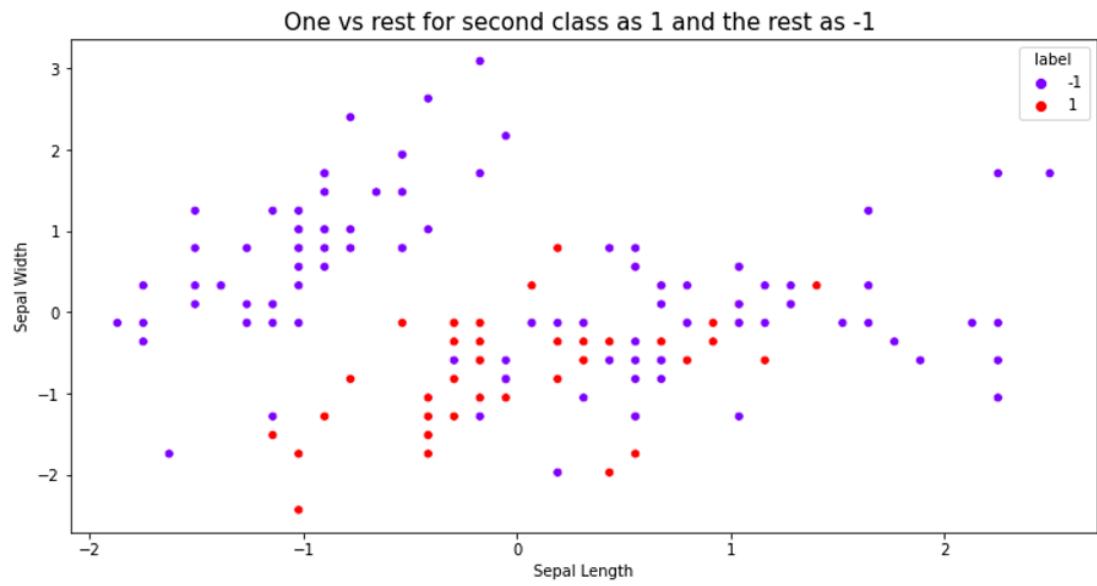
The Slope of the line is : 2.125537891718019
The Intercept of the line is : 1.416033634893979

The Slope of the line is : 2.4200860471769543
The Intercept of the line is : 1.9006386215594753



شکل 3-2-6 : خطوط جدا کننده برای حالت SD در راست و SGD در چپ

لیبل 1 برای کلاس دوم ②

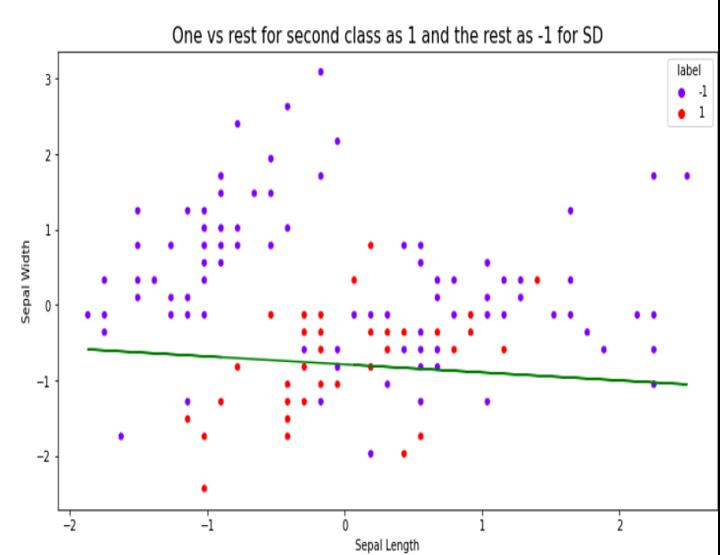
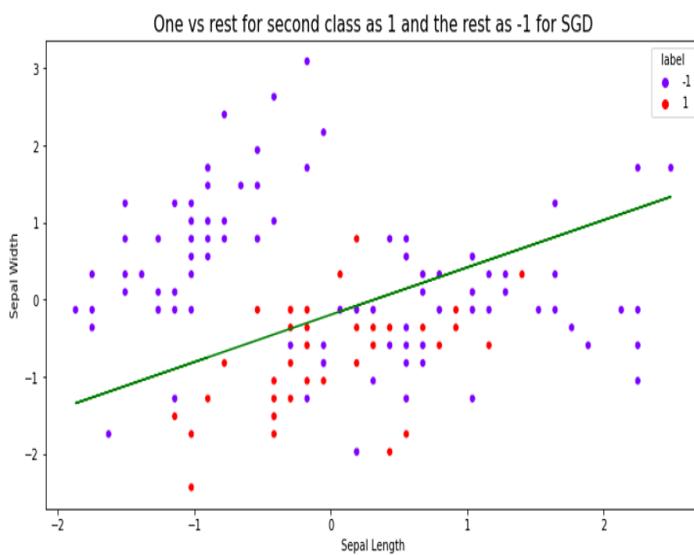


شکل 7-3 : نمایی از دو کلاس و دو ویژگی حاصل از one-vs-rest

با اجرای الگوریتم SGD و SD با روش Armijo برای حالت دوم ، خطوط جدا کننده به شکل زیر خواهد بود :

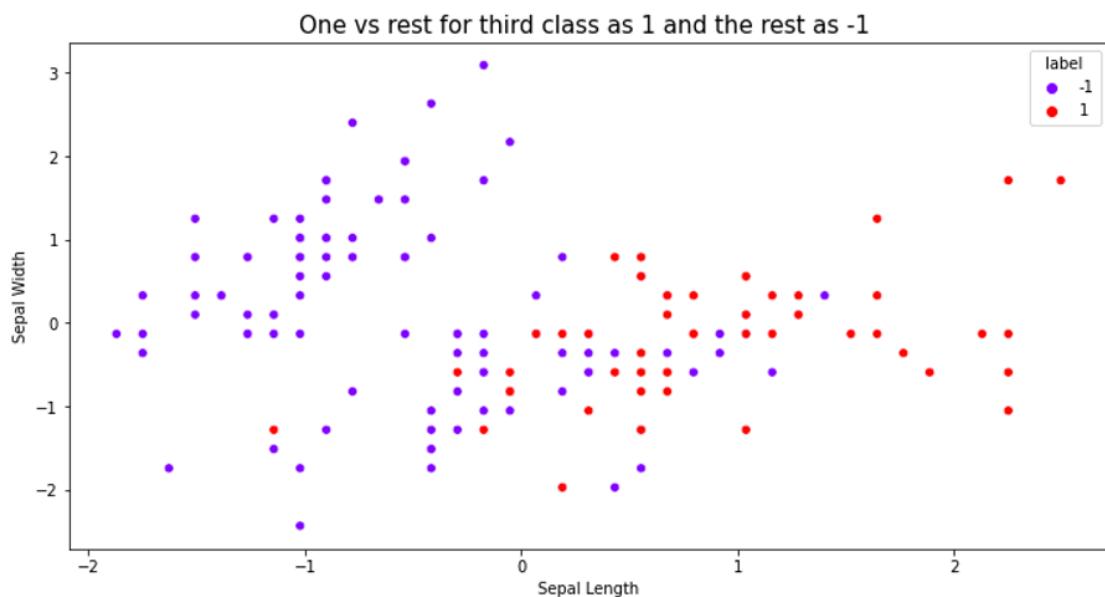
The Slope of the line is : 0.6138036387437594
The Intercept of the line is : -0.19605519530773322

The Slope of the line is : -0.10624435350961324
The Intercept of the line is : -0.7887831666629819



شکل 8-3 : خطوط جدا کننده برای حالت SD در راست و SGD در چپ

لیبل 1 برای کلاس سوم ③

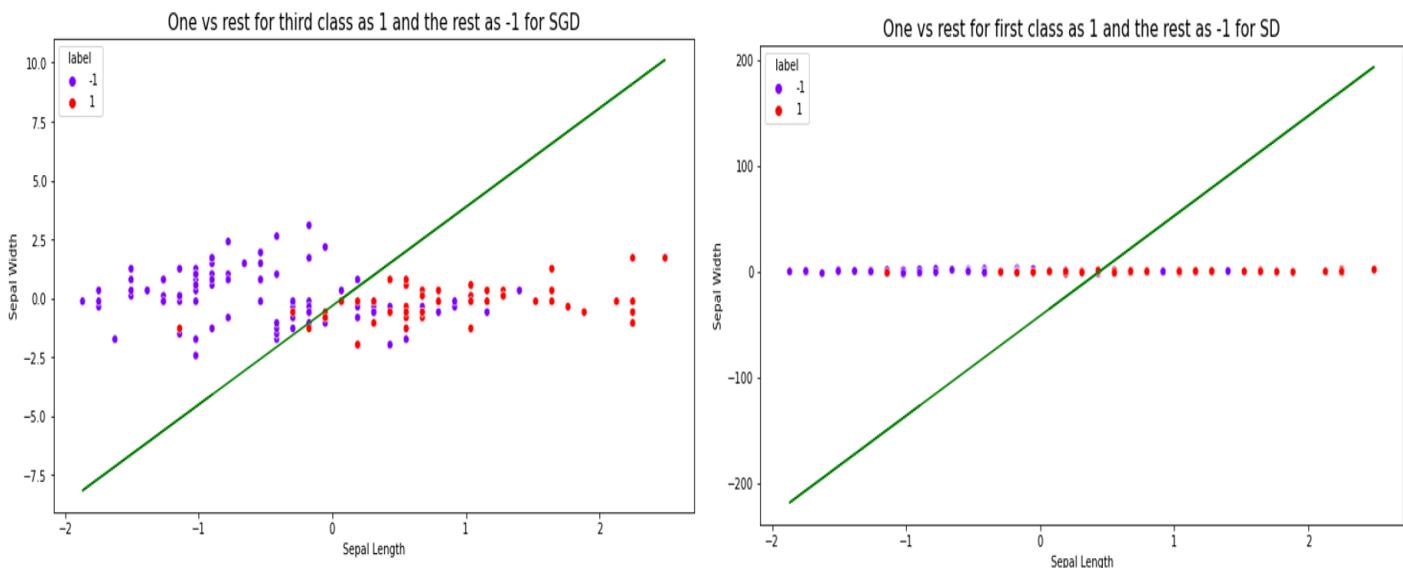


شکل 9-2-3 : نمایی از دو کلاس و دو ویژگی حاصل از one-vs-rest

با اجرای الگوریتم SGD و SD با روش Armijo برای حالت سوم ، خطوط جدا کننده به شکل زیر خواهد بود :

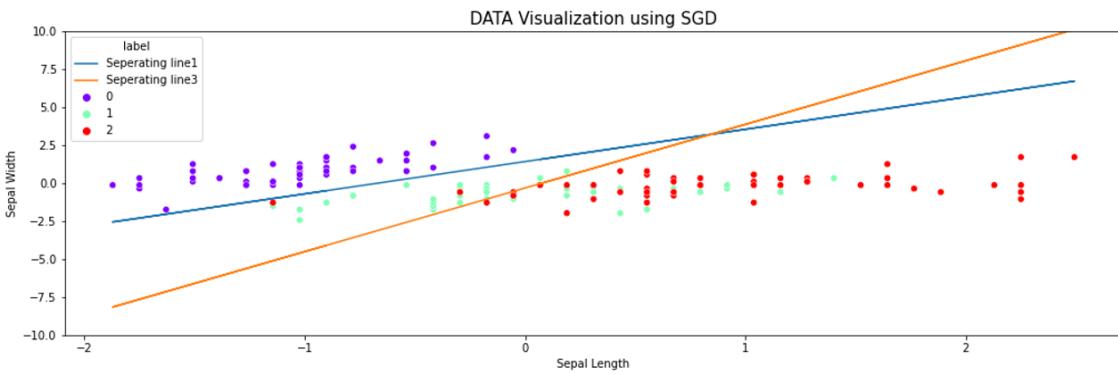
The Slope of the line is : 4.190651794660897
The Intercept of the line is : -0.32400590934754975

The Slope of the line is : 94.37641595523125
The Intercept of the line is : -41.8985844335967

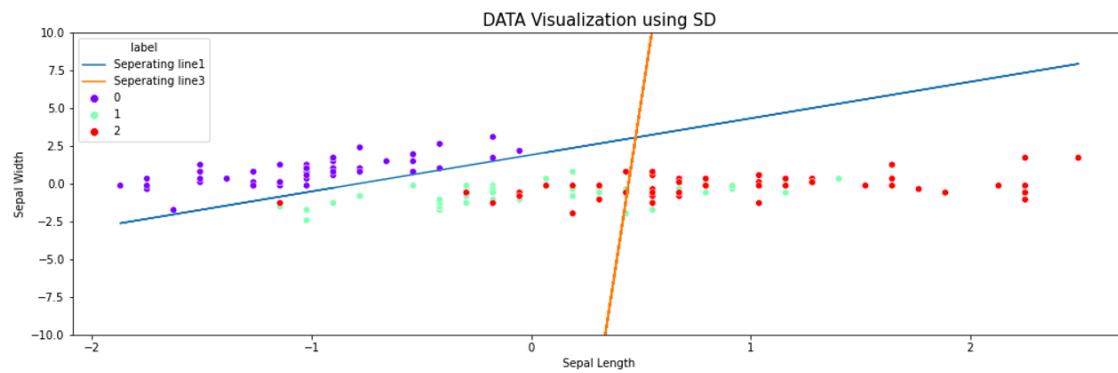


شکل 10-2-3 : خطوط جدا کننده برای حالت SD در راست و SGD در چپ

در نهایت ، بعد از بررسی هر سه حالت و رسم خطوط جدا کننده ، خطوط جدا کننده را برای سه کلاس اولیه برای هر دو روش SGD و SD رسم می کنیم :



شکل 11-2-3 : خطوط جداکننده برای سه کلاس اولیه و تقسیم فضا به سه قسمت برای روش SGD



شکل 12-2-3 : خطوط جداکننده برای سه کلاس اولیه و تقسیم فضا به سه قسمت برای روش SD

در نهایت برای هر دو روش SGD و SD و برای هر سه حالت بحث شده در بالا ، Accuracy را محاسبه می کیم :

```

acc1=np.mean(np.array(list(map(convert,X_new @ W1_SGD.reshape(3,1)>=0))==new_label1))
acc2=np.mean(np.array(list(map(convert,X_new @ W2_SGD.reshape(3,1)>=0))==new_label2))
acc3=np.mean(np.array(list(map(convert,X_new @ W3_SGD.reshape(3,1)>=0))==new_label3))
total_acc=(acc1+acc2+acc3)/3
print("Accuracy for class 1 and the rest is {}".format(acc1))
print("Accuracy for class 2 and the rest is {}".format(acc2))
print("Accuracy for class 3 and the rest is {}".format(acc3))
print("The total accuracy will be {}".format(total_acc))

```

```

Accuracy for class 1 and the rest is 1.0
Accuracy for class 2 and the rest is 0.6266666666666667
Accuracy for class 3 and the rest is 0.7333333333333333
The total accuracy will be 0.7866666666666666

```

```

[91] acc1=np.mean(np.array(list(map(convert,X_new @ W1_SD.reshape(3,1) >=0))==new_label1))
acc2=np.mean(np.array(list(map(convert,X_new @ W2_SD.reshape(3,1) >=0))==new_label2))
acc3=np.mean(np.array(list(map(convert,X_new @ W3_SD.reshape(3,1) >=0))==new_label3))
total_acc=(acc1+acc2+acc3)/3
print("Accuracy for class 1 and the rest is {}".format(acc1))
print("Accuracy for class 2 and the rest is {}".format(acc2))
print("Accuracy for class 3 and the rest is {}".format(acc3))
print("The total accuracy will be {}".format(total_acc))

```

```

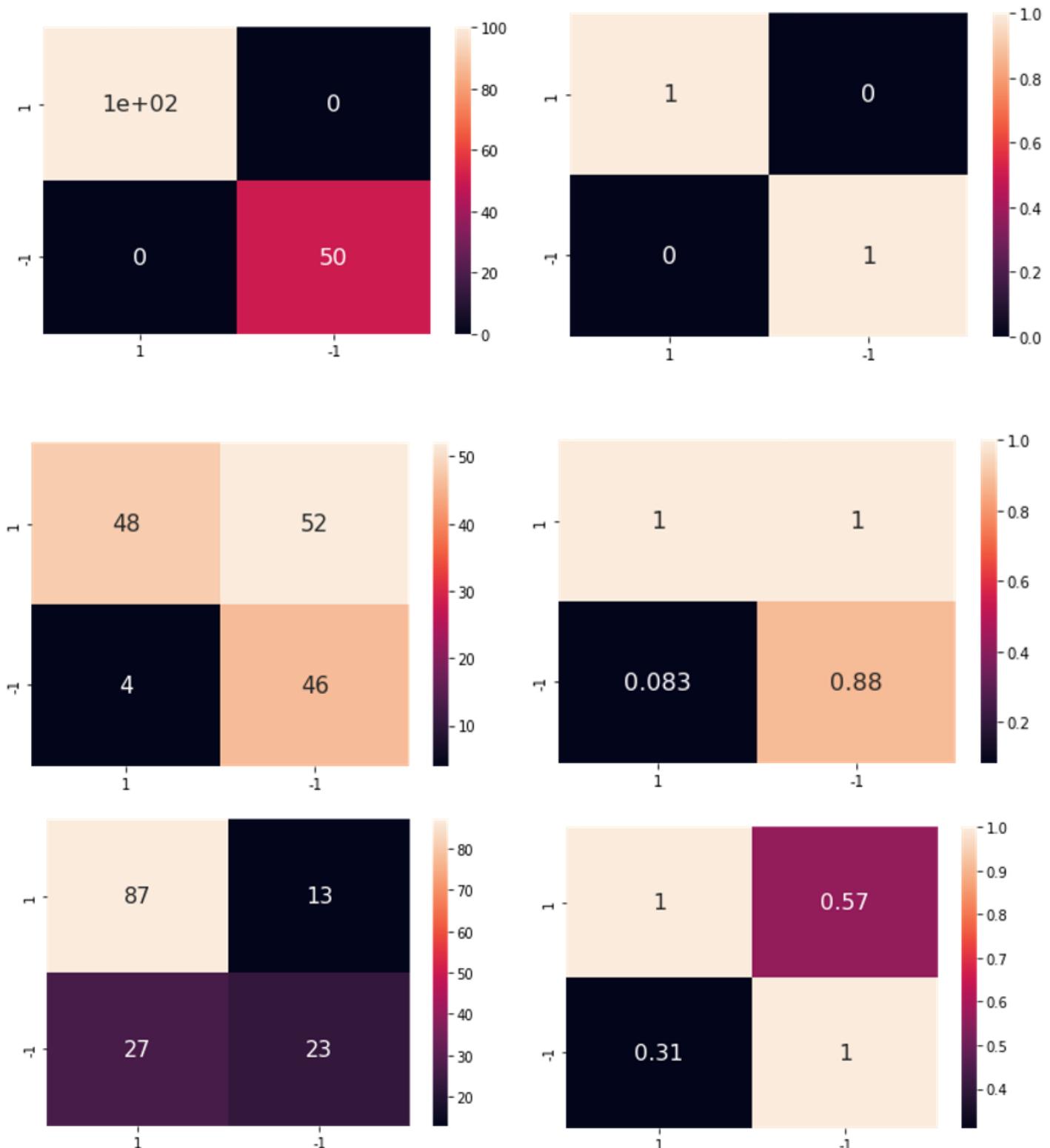
Accuracy for class 1 and the rest is 1.0
Accuracy for class 2 and the rest is 0.74
Accuracy for class 3 and the rest is 0.8133333333333334
The total accuracy will be 0.8511111111111111

```

شکل 13-2-3 : دقیقیت هر کلاس برای دو روش SGD و SD

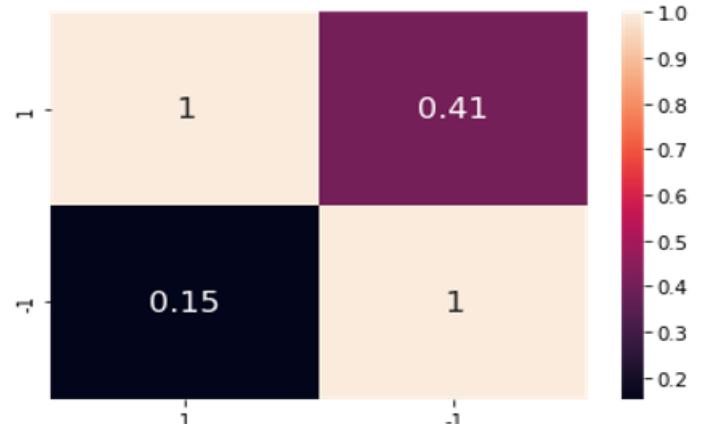
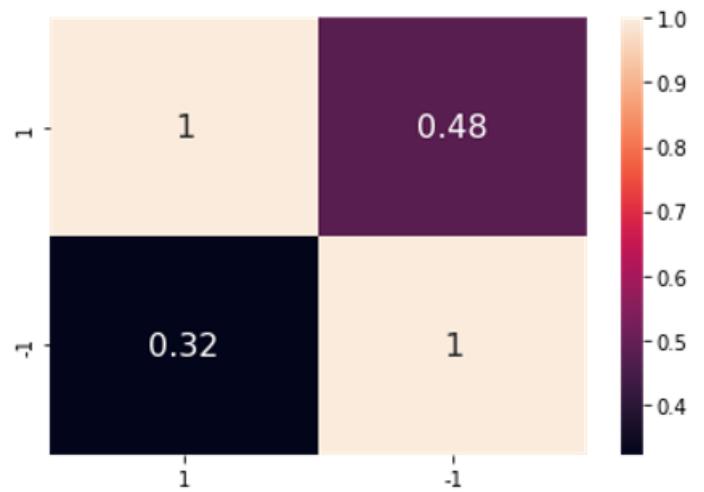
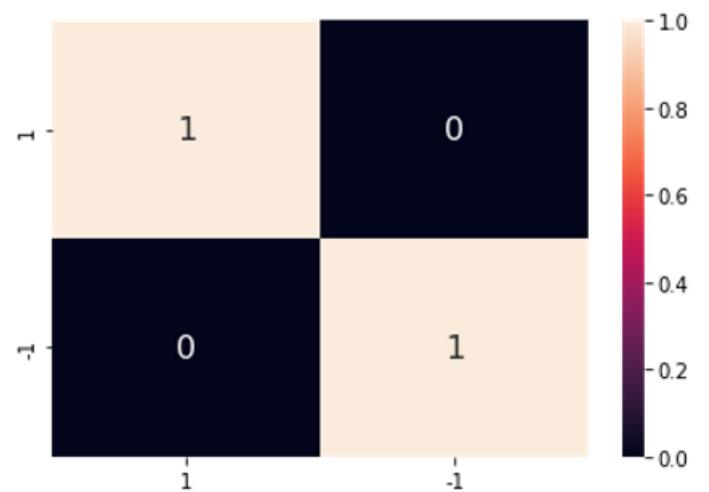
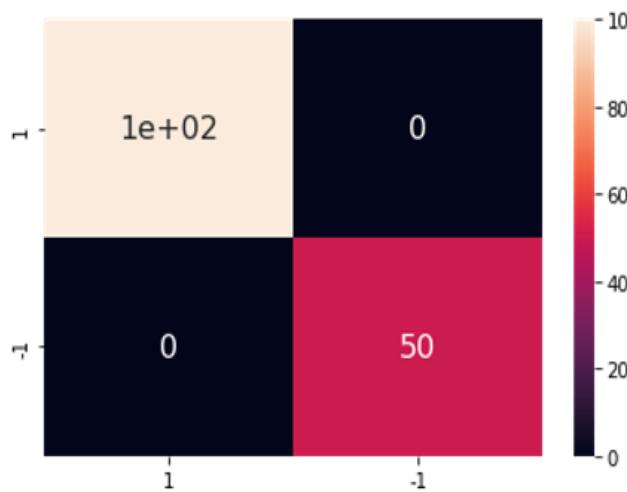
در نهایت ، ماتریس آشتفتگی (Confusion matrix) و ماتریس اطمینان (Confidence matrix) را برای هر یک از سه حالت و دو روش SGD و SD رسم می کنیم (ماتریس آشتفتگی و اطمینان مربوط به هر حالت ، در یک سطر می باشد) :

:SGD ★



شکل 4-2-14 : ماتریس آشفتگی و ماتریس اطمینان برای هر سه حالت در روش SGD

SD ★



شکل 3-15 : ماتریس آشفتگی و ماتریس اطمینان برای هر سه حالت در روش SGD

ب) (پیاده سازی با کتابخانه Sklearn

در این قسمت از کتابخانه آماده Sklearn ، برای fit کردن داده ها به مدل SVC استفاده می کنیم :

```
model = SVC(kernel='linear', C=1000)
model.fit(X, Y)

SVC(C=1000, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
     decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
     max_iter=-1, probability=False, random_state=None, shrinking=True,
     tol=0.001, verbose=False)
```

شکل 3-2-16 : آموزش و fit کردن داده های آموزش به مدل SVC

ضرایب بدست آمده برای کدام از 3 حالت بصورت زیر می باشد :

```
[28] model.coef_
```

```
array([[-6.31777572e+00,  5.26481134e+00,  7.10542736e-15],
       [-8.56758652e+00,  7.14059760e+00,  7.10542736e-15],
       [-1.60655373e+00, -7.96047821e-01,  0.00000000e+00]])
```

شکل 3-2-17 : ضرایب بدست آمده از خروجی مدل SVC

حال Classification report و Accuracy-score را برای مدل آموزش دیده بدست می آوریم :

```
[29] model.predict(X_new)
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2,
       2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2])
```

```
[37] accuracy_score(label,model.predict(X_new))
```

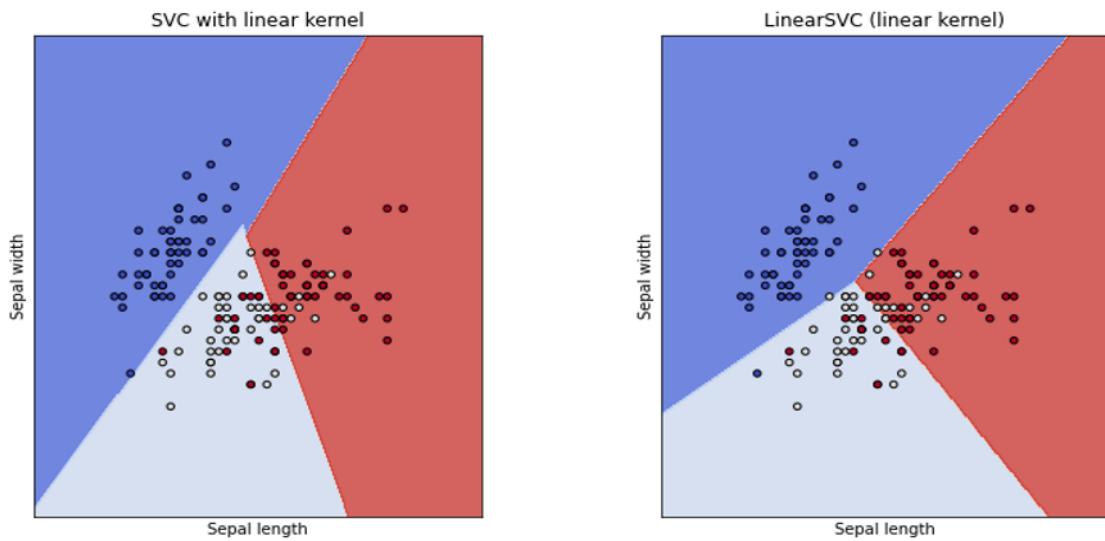
0.82

```
[38] print(classification_report(label,model.predict(X_new)))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	50
1	0.72	0.76	0.74	50
2	0.74	0.70	0.72	50
accuracy			0.82	150
macro avg	0.82	0.82	0.82	150
weighted avg	0.82	0.82	0.82	150

شکل 3-2-18 : گزارش طبقه بند و دقت مدل SVC بر روی دادگان Iris

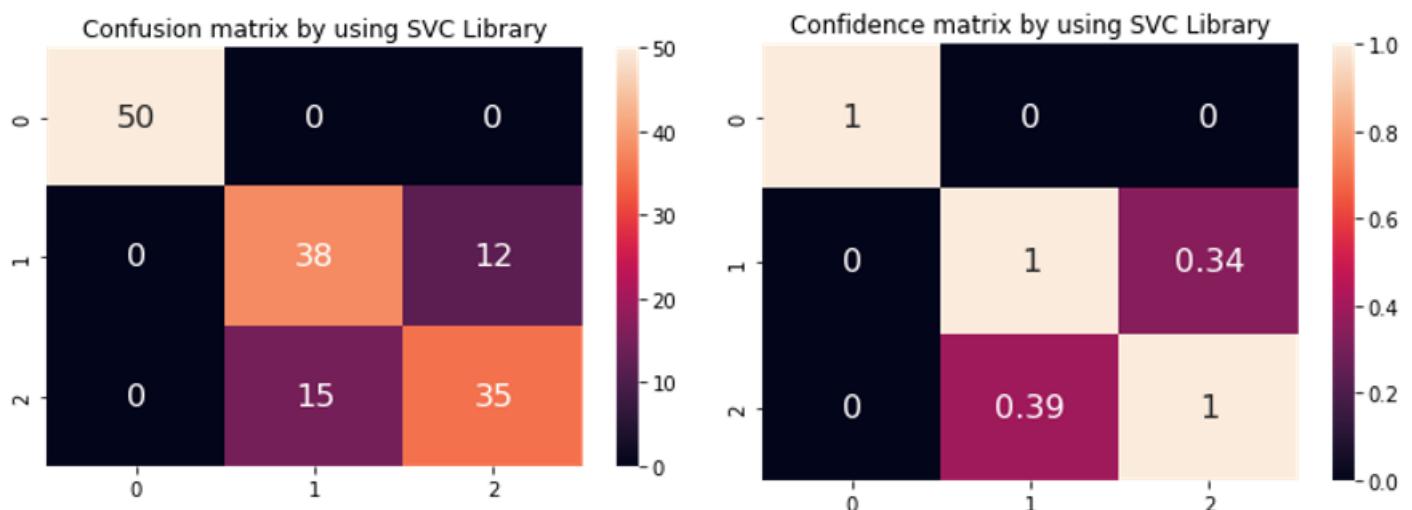
در نهایت خطوط جداساز و در واقع ناحیه جدا کننده سه کلاس را برای دو حالت one-vs-rest و one_vs_one رسم می کنیم :



شکل 19-3 : خطوط و نواحی جداکننده سه کلاس با استفاده از دو روش one-vs-rest و one-vs-one

* در شکل سمت چپ (SVC) از روش one-vs-one و در شکل سمت راست (LinearSVC) از روش one-vs-all استفاده شده است.

همچنین ماتریس آشфтگی و ماتریس اطمینان حاصل از این طبقه بند در زیر آمده است :



شکل 20-3 : ماتریس آشфтگی و ماتریس اطمینان حاصل از طبقه بندی سه کلاس

پایان ...