درس یادگیری ماشین

تمرین سری چهارم

فرهاد دلیرانی

۹۶۱۳۱۱۲۵

dalirani@aut.ac.ir

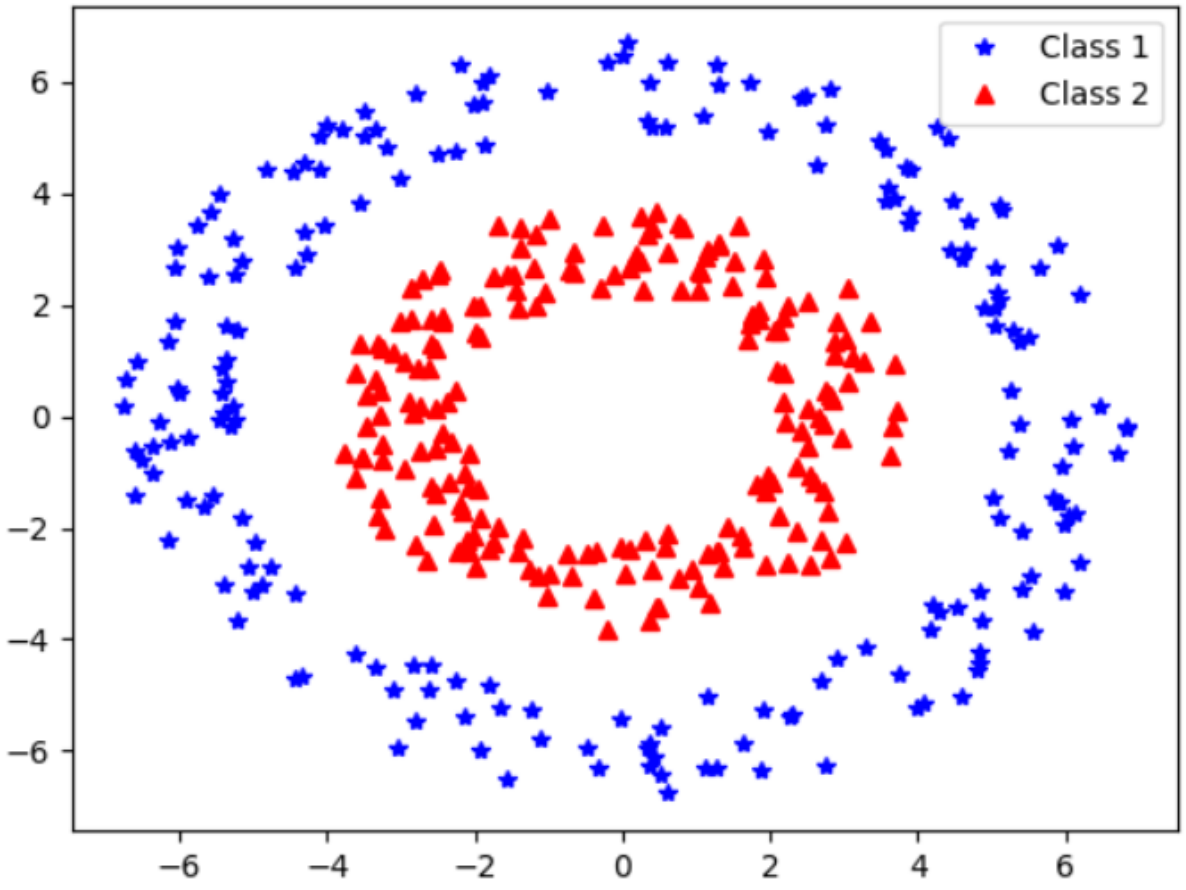
dalirani.1373@gmail.com

سوال ۱ :

کدهای این سوال در فایل problem1.py قرار دارد.

قسمت الف)

در این قسمت داده‌ها را از فایل خوانده‌ام و آن‌ها را رسم کرده‌ام:

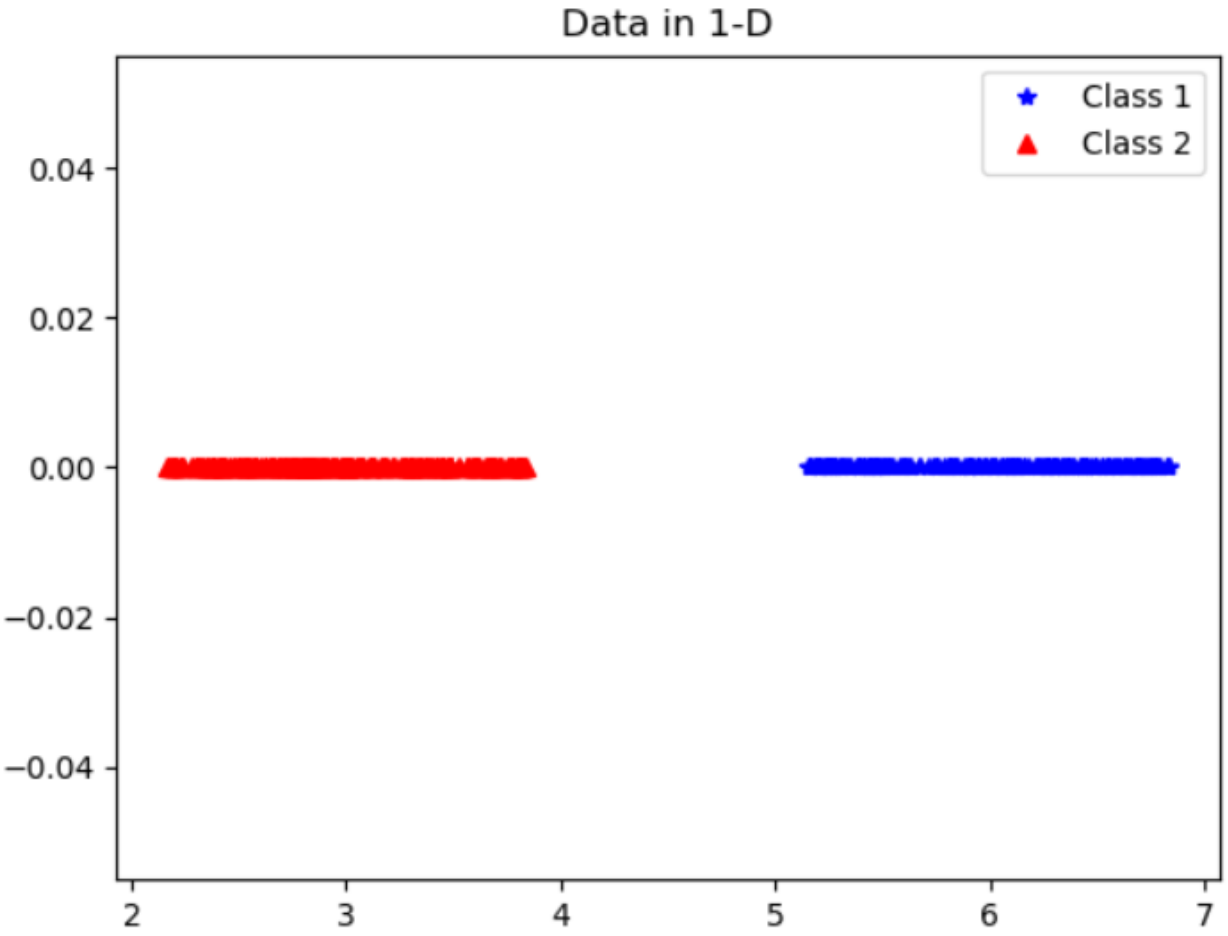


قسمت ب)

در این قسمت باید داده‌ها را از دو بعد به یک بعد طوری نگاشت کنیم که داده‌ها به صورت خطی جدا پذیر باشند به همین منظور از نگاشت زیر برای بردن داده‌ها از ۲ بعد به یک بعد استفاده کرده‌ام:

$$[x1\ x2] \rightarrow [\sqrt{x_1^2 + \chi_2^2}]$$

در زیر تصویر نقطه‌های نگاشت شده به یک بعد را مشاهده می‌کنید:

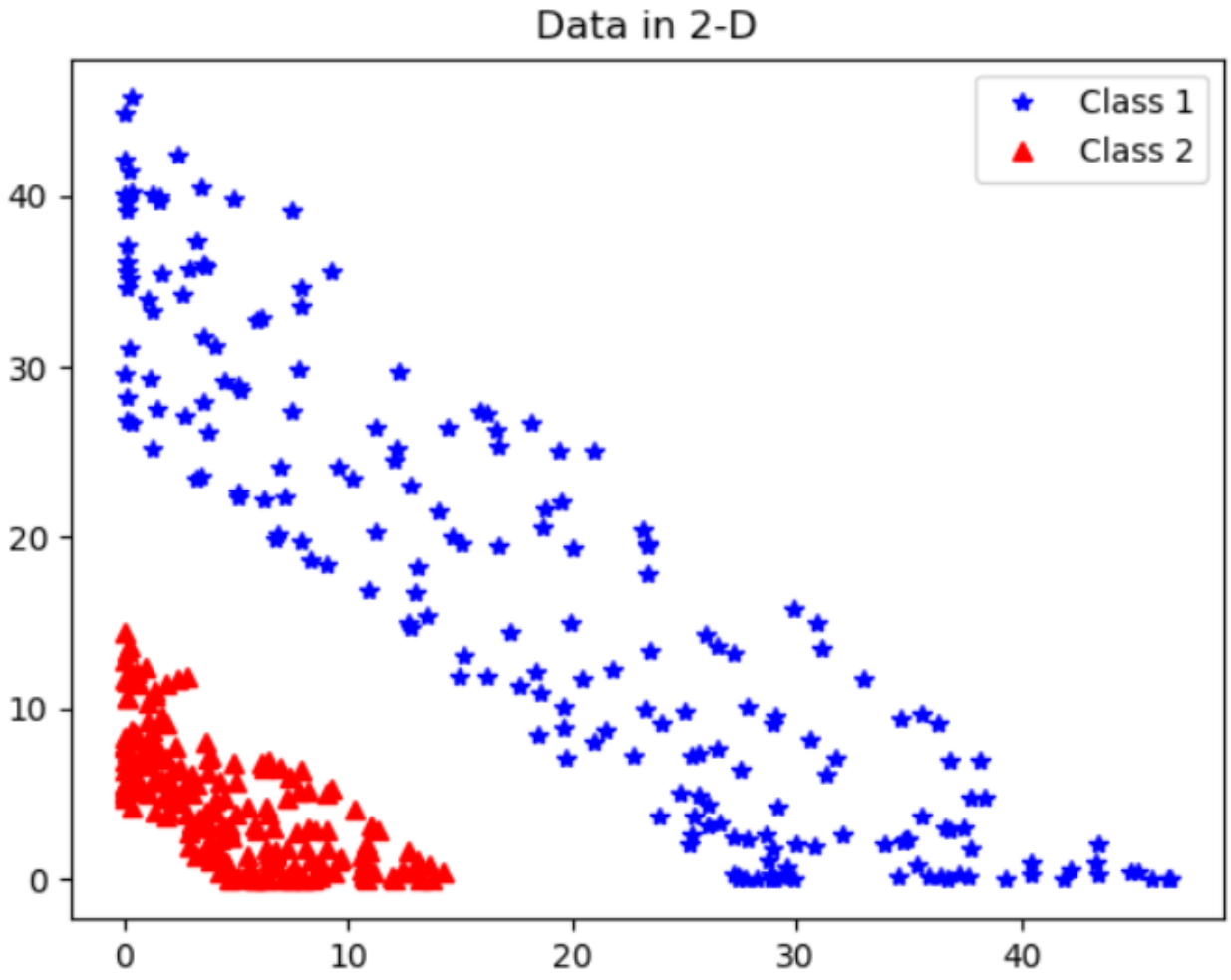


بخش ج)

در این بخش خواسته شده است که داده‌ها را از دو بعد بع مختصاتی جدیدی در دو بعد ببریم به طوری که داده‌ها خطی جداپذیر باشند. به این منظور از نگاشت زیر استفاده کرده‌ام:

$$[x1\ x2] \rightarrow [x_1^2\ x_2^2]$$

در تصویر زیر داده های نگاشت شده را مشاهده می‌کنید:

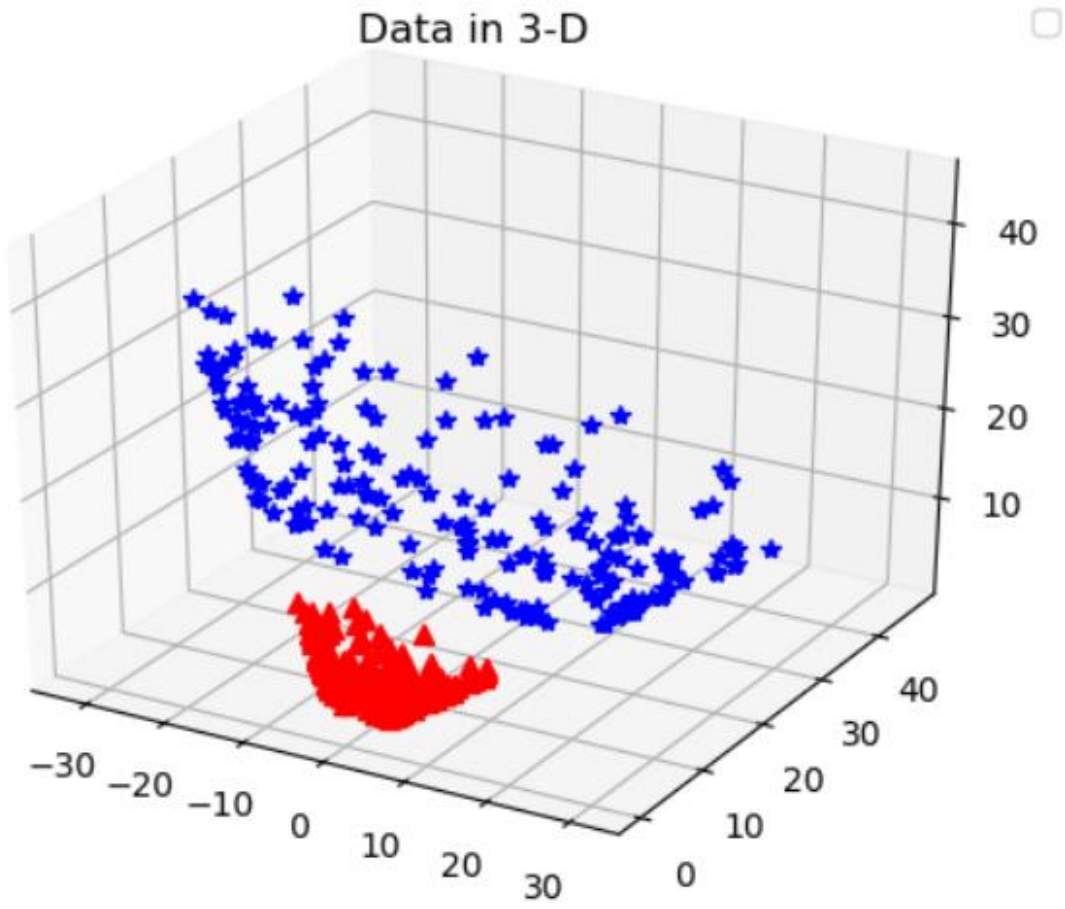


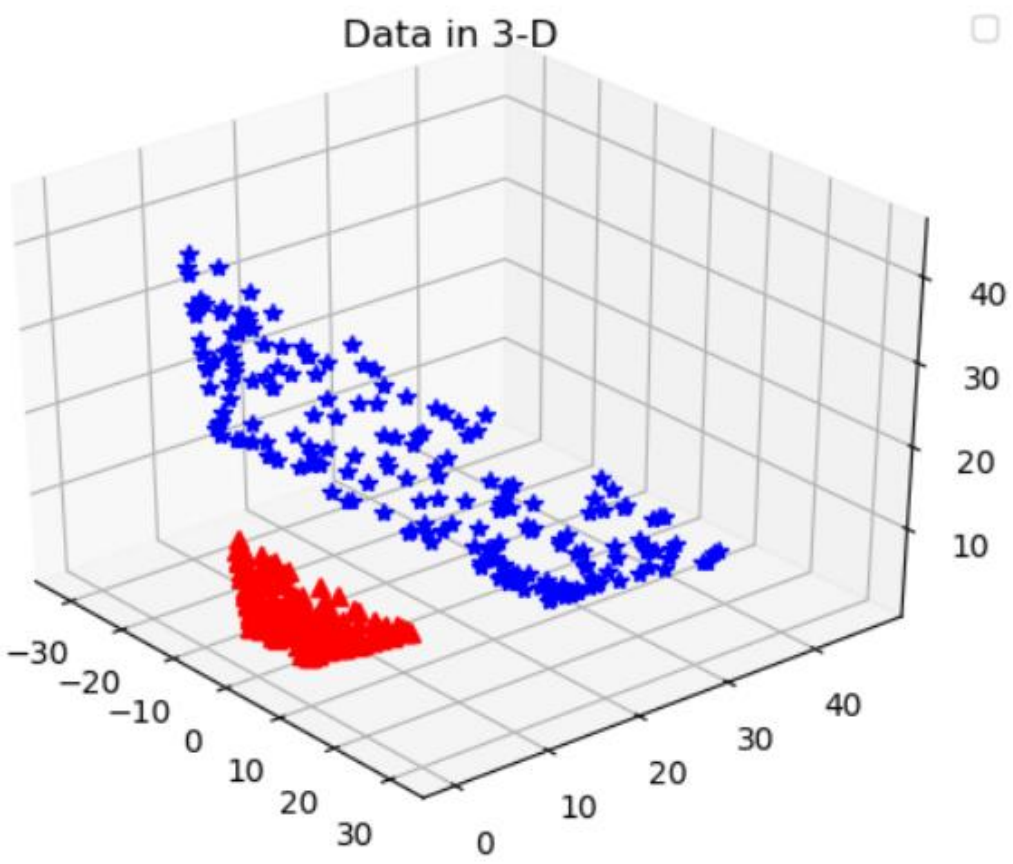
بخش ج)

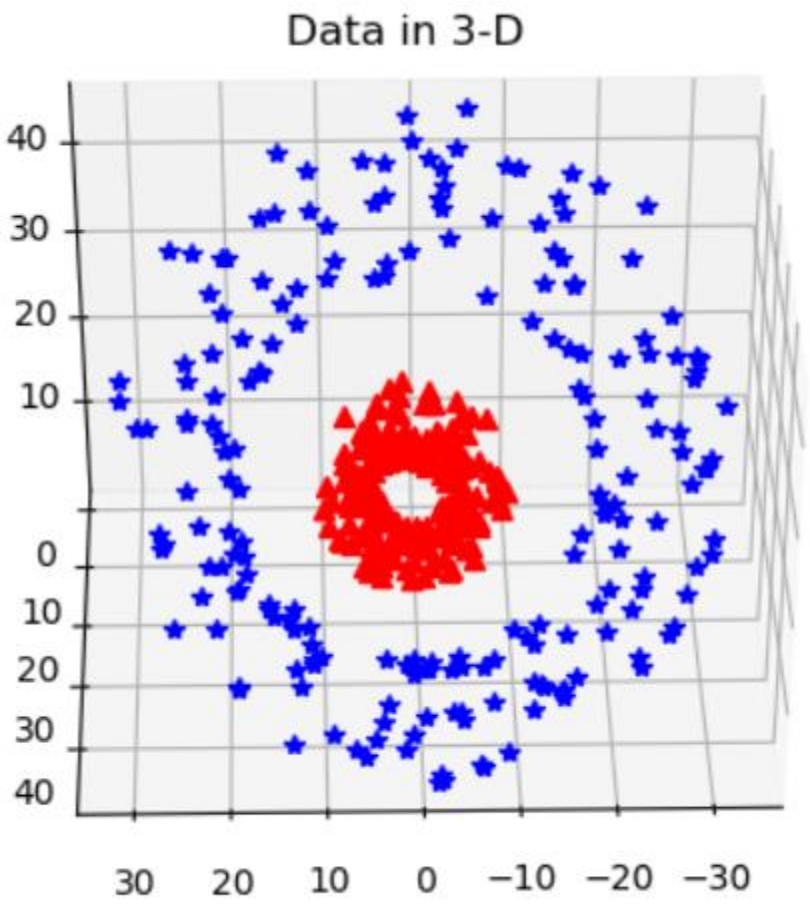
در این بخش داده‌ها باید از دو بعد به سه بعد بروند به طوری که در مختصات جدید جداپذیر باشند، به همین منظور از نگاشت دو به سه بعد زیر استفاده کرده ام:

$$[x1\ x2] \rightarrow [\sqrt{2}x_1x_2\ \ x_1^2\ \ x_2^2]$$

در شکل‌های زیر تصویر داده‌های نگاشت شده را از چند زاویه‌ی مختلف مشاهده می‌کنید.


Data in 3-D
40
30
20
10
40
30
20
10
0
−30
−20
−10
0
10
20
30


Data in 3-D
40
30
20
10
−30
−20
−10
0
10
20
30
0
10
20
30
40



بخش ه)

برای دو بعد به یک بعد:

$$\Phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \left[\sqrt{x_1^2 + x_2^2}\right]$$

$$\Phi\left(\begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}\right) = \left[\sqrt{x'^2_1 + x'^2_2}\right]$$

$$K(x\ \ x') = \Phi(x) \cdot \Phi(x') = \sqrt{(x_1^2 + x_2^2) * (x'^2_1 + x'^2_2)}$$

ماتریس کرنل برابر با ماتریسی است که در آن مقدار کرنل به ازای هر دو نقطه به دست آمده است، از آنجایی که این ماتریس بزرگ است بخشی از آن را در تصویر زیر مشاهده می‌کنید:

```
Kernel Matrix of Mapping To One dimensional data:
Size: (400L, 400L)

[5.9511161522089999, 8.3849587610875798, 14.502112733892515, 6.6221682941253457, 7.2799776910871374, 6.3985703924053299, 13.975251579104805, 16.462536469032479, 16.21431712206
[8.3849587610875798, 11.814175967484999, 20.433077445009225, 9.3304527478617167, 10.257288071708174, 9.0154094623610437, 19.69074458799145, 23.195260496553967, 22.845526272791
[14.502112733892515, 20.433077445009225, 35.339803218, 16.137381406774754, 17.740379195452217, 15.592501774976299, 34.055909631247538, 40.117106380306005, 39.512227419726393,
[6.6221682941253457, 9.3304527478617167, 16.137381406774754, 7.3688887586979988, 8.1008732168607143, 7.1200777965974487, 15.551111008851777, 18.318863933724892, 18.04265519448
[7.2799776910871374, 10.257288071708174, 17.740379195452217, 8.1008732168607143, 8.9055689432400005, 7.8273467565022834, 17.095871954280121, 20.138558073476482, 19.83491229305
[6.3985703924053299, 9.0154094623610437, 15.592501774976299, 7.1200777965974487, 7.8273467565022834, 6.8796679512580017, 15.02602683150174, 17.700326416170547, 17.433443881248
[13.975251579104805, 19.69074458799145, 34.055909631247538, 15.551111008851777, 17.095871954280121, 15.02602683150174, 32.818659845309007, 38.65965356759461, 38.07674983459221
[16.462536469032479, 23.195260496553967, 40.117106380306005, 18.318863933724892, 20.138558073476482, 17.700326416170547, 38.65965356759461, 45.540214652612015, 44.8535669806
[16.214317122066298, 22.845526272791094, 39.512227419726393, 18.042655194485647, 19.834912293056071, 17.43344388124801, 38.076749834592214, 44.853566980606253, 44.177272466333
[8.4220351065584182, 11.866415517148202, 20.523427780648237, 9.3717098487415047, 10.302643423711269, 9.0552735148044228, 19.777812499679989, 23.297824565854743, 22.94654389836
[12.696017944209244, 17.888339627299509, 30.938580056129709, 14.127630068282397, 15.530990327785371, 13.650609808566726, 29.814582724501648, 35.12091733253321, 34.591370067351
[15.536144664828504, 21.889999957826873, 37.859607444525146, 17.288011522709525, 19.005306512707072, 16.704280797409396, 36.484169474620607, 42.977542630883825, 42.32953449519
[15.348003626794062, 21.624914416755484, 37.401131677347543, 17.078655565771424, 18.775154298460624, 16.501993756663815, 36.042350113069539, 42.457089220003255, 41.81692839301
[6.2700151532473134, 8.8342786708772305, 15.279229016871914, 6.9770265761167467, 7.6700856227575924, 6.7414468636457663, 14.724135259720137, 17.344704838840308, 17.08318430605
[5.7993598848763428, 8.1711383598761902, 14.132302022650387, 6.4532998808291104, 7.0943348280665059, 6.235403831010581, 13.618876075680538, 16.042734028369182, 15.800844391765
[12.861754762634591, 18.12185823996758, 31.342459591243284, 14.312055489678036, 15.733735545635039, 13.82880809476792, 30.203789332837612, 35.579394086776354, 35.0429339864579
[6.2713825314827272, 8.8362052691524973, 15.282561143621361, 6.97854814090645, 7.6717583313377133, 6.7429170527109035, 14.727346330442831, 17.348487408948458, 17.0869098432069
[16.180143037788891, 22.797375929219392, 39.428949525278888, 18.004627615860986, 19.793107266101504, 17.396700306050978, 37.996497422599717, 44.759031418818616, 44.08416229579
[13.430436422110262, 18.923115036328436, 32.72826442591316, 14.944862102506875, 16.429401650686032, 14.440247955116515, 31.539247935278983, 37.152534707618216, 36.592355058472
[6.1762311403090502, 8.7021395795162366, 15.050689312137253, 6.8726674103583916, 7.5553599974280061, 6.6406113912537545, 14.503898392989896, 17.085270693426214, 16.82766186490
```

برای دو بعد به دو بعد:

$$\Phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix}$$

$$\Phi\left(\begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}\right) = \begin{bmatrix} x'^2_1 \\ x'^2_2 \end{bmatrix}$$

$$K(x\ \ x') = \Phi(x) \cdot \Phi(x') = \chi_1^2 \chi_1'^2 + \chi_2^2 \chi_2'^2$$

ماتریس کرنل برابر با ماتریسی است که در آن مقدار کرنل به ازای هر دو نقطه به دست آمده است، از آنجایی که این ماتریس بزرگ است بخشی از آن را در تصویر زیر مشاهده می‌کنید:

```
Kernel Matrix of Mapping To Two dimensional data:
Size: (400L, 400L)

[798.98033220375339, 17.068358799745372, 70.827253288320463, 178.79920333760364, 119.68851826827377, 90.163975073415614, 51.8038355
[17.068358799745372, 2014.6242613713928, 1563.4969916189957, 1407.4488653019357, 252.26129838725711, 138.42022645677022, 167.726195
[70.827253288320463, 1563.4969916189957, 1217.5394601246965, 1104.3126388901965, 204.24659588950041, 113.83888919418411, 133.799544
[178.79920333760364, 1407.4488653019357, 1104.3126388901965, 1018.1261860076786, 200.78999544789298, 115.2926633131496, 127.7010711
[119.68851826827377, 252.26129838725711, 204.24659588950041, 200.78999544789298, 48.884974267847909, 30.427651534559036, 28.4158433
[90.163975073415614, 138.42022645677022, 113.83888919418411, 115.2926633131496, 30.427651534559036, 19.424267729295316, 17.1367893
[51.803835592401789, 167.72619545723595, 133.79954487583089, 127.70107110453547, 28.415843365674384, 17.136789363912175, 17.141592
[1048.7313604705548, 23.116730070658981, 93.519867596961262, 235.18641222159982, 157.1900095674425, 118.39639702477861, 68.0560327
[378.24586067072812, 11.270223448041484, 36.004015073383563, 86.868316362514648, 57.057272610431639, 42.900751724024097, 24.7883582
[716.49769000440597, 556.24068246028833, 482.9900323369211, 537.28896789827093, 174.39122639941019, 117.51172874473943, 91.2019558
[112.6972747521654, 166.66063503035883, 137.36250747655998, 139.6790408635826, 37.24441733044754, 23.848184176322732, 20.894017845
[582.31205757760608, 1048.6397082652172, 855.15503911941209, 852.38429856825223, 215.68726129881929, 135.93016704650717, 123.470065
[201.15329641903244, 69.558386591156008, 68.439307469573976, 90.491967585940046, 38.223406740380462, 27.122263128508031, 18.4406704
[1268.6307269121778, 18.753735750677816, 105.98711226849767, 278.08255568594831, 189.00805095786242, 142.59779373992751, 81.5642482
[175.74647307762996, 139.96425201378008, 121.20510353866706, 134.24660170589789, 43.212807712787153, 29.062888693130688, 22.6622262
[142.93773818526452, 50.831152227345207, 49.720788310862034, 65.280866632704956, 27.335211764346276, 19.367950518559869, 13.219880
[19.814146162106912, 583.52824065287746, 453.93281777819249, 410.76858555079355, 75.254746355493083, 41.749467007191633, 49.5191317
[0.84008298479771271, 316.98215967231442, 245.86850822739103, 221.06331475580311, 39.419374356726074, 21.573532675863703, 26.273743
[55.735098578924557, 1697.5353832076946, 1320.3934219115588, 1194.5640699664054, 218.64225314267782, 121.24056902939553, 143.935331
[4.9760886766489483, 1764.5198468634851, 1368.6786341219465, 1230.6384988058105, 219.47685102031497, 120.12508342740209, 146.274898
[10.664055335107403, 1690.382195182365, 1311.5976190343513, 1180.1641589949768, 211.1231321097591, 115.73487346271988, 140.5009751
```

برای دو بعد به سه بعد:

$$\Phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} \sqrt{2}x_1x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

$$\Phi\left(\begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}\right) = \begin{bmatrix} \sqrt{2}x'_1x'_2 \\ x'^2_1 \\ x'^2_2 \end{bmatrix}$$

$$K(x\ \ x') = \Phi(x) \cdot \Phi(x') = 2x_1x_1x'_1x'_2 + x_1^2x_1'^2 + x_2^2x_2'^2$$

ماتریس کرنل برابر با ماتریسی است که در آن مقدار کرنل به ازای هر دو نقطه به دست آمده است، از آنجایی که این ماتریس بزرگ است بخشی از آن را در تصویر زیر مشاهده می‌کنید:

```
Kernel Matrix of Mapping To Three dimensional data:
Size: (400L, 400L)

[147.05324270048297, 317.56394834109449, 292.82619855552502, 33.371190419202151, 13.613804692491223, 102.7480036
[317.56394834109449, 845.04648841636526, 471.86400598990292, 358.11883846711896, 1.6923787241674759, 189.4217256
[292.82619855552502, 471.86400598990292, 796.15022665363472, 18.029559506267784, 102.07308464798844, 244.9156885
[33.371190419202151, 358.11883846711896, 18.029559506267784, 1642.7649006014203, 173.263554344109, 0.5293335839
[13.613804692491223, 1.6923787241674759, 102.07308464798844, 173.263554344109, 49.216657168771277, 22.1268725624
[102.74800308720663, 189.42172563367626, 244.91568858818258, 0.52933358398829888, 22.126872562489282, 79.5341834
[450.12014129587544, 1262.0740171684317, 613.67958755005202, 668.16568953876788, 0.056667065821386586, 256.81212
[1.0421050860550096, 148.09921888260715, 126.24979159829201, 1529.959477234745, 223.36632672998573, 11.156205241
[2.6600355203903643, 55.895332597886579, 13.940184394106412, 367.3879342530588, 45.791456171719048, 0.4008285445
[191.51346952585283, 861.52454737888024, 77.744998472149732, 1539.1136131733351, 75.34104888822894, 64.39328652
[8.3002385111696348, 80.029810000072104, 2.3506677956474773, 339.76869900913994, 34.354819450902859, 0.30933879
[0.10194659401150297, 18.452559452762408, 37.35400426124599, 292.37170763882352, 47.752515337808134, 4.57794578
[308.52876276412979, 778.77713920434644, 497.21034113053855, 254.39939729940807, 5.5215485578398251, 192.0796795
[62.591568247811423, 218.33124767377234, 55.081688408546263, 230.21559722686388, 4.3964082917968703, 28.89215634
[420.79399027486164, 787.77155522662895, 986.92369236246793, 5.0634029192168164, 84.863934262213348, 322.6833387
[151.17757855740729, 418.35632579582938, 210.72833991714771, 210.04709752395817, 0.10918204947960675, 87.2385094
[0.59859124051149948, 15.788700201334976, 51.31752269825683, 328.82978820802595, 56.893059001332475, 7.0504633335
[513.63779326119334, 1284.1411760917335, 839.56272127611089, 398.26441982667666, 10.745095820100936, 322.1975327
[13.298278817988798, 75.714932551046701, 1.3797136842230984, 191.50950398995681, 13.467867196450239, 3.0085508095
[62.668540218318128, 121.56516117349659, 141.4426724607132, 2.4686320866681779, 10.742200292379582, 47.005016126
[138.82507500672818, 252.11695403375853, 336.10441737092947, 0.17007299930348108, 31.796730699675638, 108.437796
```

بخش و)

در این بخش باید یک SVM بر روی داده‌های اصلی آموزش دهیم به همین دلیل از امکانات libsvm بر روی داده‌های اصلی بدون استفاده از نگاشت و تکنیک کرنل بدین صورت استفاده کرده‌ام:

```
# train linear svm for original data without mapping and kernel trick
five_cv_accuracy = svm.svm_train(label, dataset, '-t 0 -v 5')
```

دقت الگوریتم را که با استفاده از ۵-cross-validation به دست آمده را در زیر مشاهده می‌کنید:

```
Cross Validation Accuracy = 52%
Accuracy of model without kernel and data mapping:  52.0  %
```

دقت الگوریتم بسیار کم است و دلیل آن این است که داده‌ها خطی جداپذیر نیستند.

بخش ز)

در این بخش باید برای داده‌های اصلی نگاشت شده به بعدهای یک، دو و سه SVM آموزش دهیم و دقت هر کدام از آن‌ها را با استفاده از ۵-fold-cross-validation به دست بیاوریم.

میزان خطا برای SVM آموزش داده شده با استفاده از داده‌های نگاشت شده به ۱ بعد:

```
*
optimization finished, #iter = 5
nu = 0.007058
obj = -1.145266, rho = 6.699049
nSV = 4, nBSV = 2
Total nSV = 4
Cross Validation Accuracy = 100%
Accuracy of model with data mapped to 1-D:  100.0  %
```

میزان خطا برای SVM آموزش داده شده با استفاده از داده‌های نگاشت شده به ۲ بعد:

```
*
optimization finished, #iter = 111
nu = 0.000173
obj = -0.027757, rho = 3.462541
nSV = 3, nBSV = 0
Total nSV = 3
Cross Validation Accuracy = 100%
Accuracy of model with data mapped to 2-D:  100.0  %
```

میزان خطا برای SVM آموزش داده شده با استفاده از داده‌های نگاشت شده به ۳ بعد:

```
.*
optimization finished, #iter = 345
nu = 0.000173
obj = -0.027715, rho = 3.465974
nSV = 4, nBSV = 0
Total nSV = 4
Cross Validation Accuracy = 100%
Accuracy of model with data mapped to 3-D:  100.0  %
```

در هر سه مورد دقت الگوریتم بر روی داده‌های نگاشت داده شده که با استفاده از ۵-fold-cross-validation به دست آمده است برابر است زیر دیتاها را از فضای دو بعدی جدا ناپذیر به فضای جداپذیر در یک، دو و سه بعد برده‌ایم.

بخش ح)

در این قسمت باید از ماتریس‌های کرنلی که برای کرنل‌های نگاشت به یک, دو و سه بعد نوشته‌ایم استفاده کنیم و بدون انتقال مستقیم داده‌ها به فضای دیگر به صورت ضمنی(غیر مستقیم) داده‌ها را نگاشت کنیم و SVM ها را آموزش و ارزیابی کنیم:

کرنل دو بعد به یک بعد: همین طور که در تصویر زیر مشاهده می‌شود دقت SVM که با ۵-fold-cv به دست آمده است برابر با ۱۰۰ است.

```
*
optimization finished, #iter = 4
nu = 0.007110
obj = -1.147031, rho = 6.750150
nSV = 4, nBSV = 2
Total nSV = 4
Cross Validation Accuracy = 100%
Accuracy of model with Kernel to 1-D:  100.0  %
```

کرنل دو بعد به دو بعد: همین طور که در تصویر زیر مشاهده می‌شود دقت SVM که با ۵-fold-cv به دست آمده است برابر با ۱۰۰ است.

```
..*
optimization finished, #iter = 727
nu = 0.000172
obj = -0.027448, rho = 3.427121
nSV = 3, nBSV = 0
Total nSV = 3
Cross Validation Accuracy = 100%
Accuracy of model with Kernel to 2-D:  100.0  %
```

کرنل دو بعد به ۳ بعد: همین طور که در تصویر زیر مشاهده می‌شود دقت SVM که با ۵-fold-cv به دست آمده است برابر با ۱۰۰ است.

```
*
optimization finished, #iter = 236
nu = 0.000174
obj = -0.027774, rho = 3.461135
nSV = 4, nBSV = 0
Total nSV = 4
Cross Validation Accuracy = 100%
Accuracy of model with Kernel to 3-D:  100.0  %
```

نتایج SVM های این بخش با نتایج SVM های بخش قبل(آموزش SVM برو روی داده‌های نگاشت شده) برابر است و این نشان می‌دهد با اینکه تکنیک کرنل داده‌ها را مستقیم به فضای جدید نمی‌برد و آن کار را ضمنی انجام می دهند ولی عملکردی مشابه دارد.

سوال دو)

بخش الف) کدهای این بخش در problem2-a.pyاست. در این بخش باید یک SVM همراه با تکنیک کرنل **RBF** برای داده‌های پارکینسون ایجاد کنید و همین‌طور باید مقدارهای مناسبی برای پارامترهای C و Gamma انتخاب کنیم.

الف-۱) در این بخش یک جدول ایجاد کرده‌ام که سطرهای آن C با مقدارهای

۱۰^-۱۰، 10^-۹، 10^-8 ، .... ،۱۰ ^10

است و ستون‌های آن برابر با Gamma با مقدارهای زیر است

۱۰^-۱۰، 10^-۹، 10^-8 ، .... ،۱۰ ^10

است.

سپس برابر هر مقدارهای مختلف C وGamma یک SVM ایجاد کرده‌ام و آن را با داده‌های آموزش، آموزش داده‌ام و بعد آن را با مجموعه داده‌ی Validation آزموده ام و دقت مدل بر داده‌های validation را به دست آورده‌ام. در هر خانه‌ی جدول ۳ مقدار را قرار داده‌ام:

C, Gamma, accuracy of SVM with parameters C and Gamma on validation set

در آخر C وGamma که بزرگ‌ترین دقت را بر روی مجموعه‌ی Validation ایجاد می‌کند را انتخاب می‌کنیم.

بعد از انتخاب پارامترهای مناسب یک SVM با آن‌ها آموزش می‌دهیم و اینبار دقت مدل نهایی را با مجموعه‌ی test می‌سنجیم.

در زیر بهترین دقت روی مجموعه‌ی Validation و پرامترهای C و Gamma آن را مشاهده می‌کنید و در شکل بعد دقت SVM آموزش داده شده با آن پارامترها را بر روی مجموعه‌ی تست مشاهده می‌کنید:

```
Part A:
> Best Validation Accuracy is 92.3076923077 Which belongs to C:1000000, Gamma:1e-06
Accuracy = 87.1795% (34/39) (classification)
> Accuracy of SVM (with best validation accuracy) on test set is 87.1794871795 ,
Which belongs to C:1000000, Gamma:1e-06
Number Of SV: 38
```

بهترین C و Gamma و دقت SVM متناظر با آن‌ها بر روی مجموعه‌ی تست و validation در تصویر بالا نمایش داده شده است. همین طور تعداد ساپورت وکتورها را مشاهده می‌کنید.

بخش الف-۲) بر خلاف بخش قبل که به صورت exhaustive تعداد فضای بزرگی از پارامترها را جست و جو می‌کرد، در این بخش به صورت تصادفی بیست جفت عدد CوGamma انتخاب می‌کنیم و آن جفتی را انتخاب می‌کنیم که بهترین دقت را بر روی مجموعه‌ی Validation ایجاد کند. سپس آن را با مجموعه داده‌های تست مورد ارزیابی قرار می‌دهیم. در تصویر زیر بهترین C و Gamma پیدا شده و دقت آن بر روی مجموعه‌ی validation و test را مشاهده می‌کنید:

```
Part B:
> Best Validation Accuracy is 87.1794871795 Which belongs to C:1000, Gamma:0.01
Accuracy = 79.4872% (31/39) (classification)
> Accuracy of SVM (with best validation accuracy) on test set is 79.4871794872 ,
Which belongs to C:1000, Gamma:0.01
Number Of SV: 93
```

بهترین C و Gamma پیدا شده به وسیله‌ی روش جست و جوی تصادفی و دقت SVM متناظر با آن‌ها بر روی مجموعه‌ی تست و validation در تصویر بالا نمایش داده شده است. همین طور تعداد ساپورت وکتورها را مشاهده می‌کنید.

بخش الف-۳) روش اول که exhaustive است بسیار هزینه بر است و زمان و محاسبات زیادی را احتیاج دارد. روش دوم تصادفی است و اگر رنج پارامتر C و Gamma بزرگ باشد مناسب نخواهد بود. می‌توان از روش‌های آپتیمایزیشن مختلف استفاده کرد مانند الگوریتم‌های ژنتیک، گرادیان نزولی و ...

بخش ب) کدهای این بخش در problem2-b.pyاست.در این قسمت به جای استفاده از یک کرنل RBF از یک کرنل Polynomial استفاده می‌کنیم، همه چیز مانند قسمت قبل است ولی به جای Gamma و C دو پارامتر Gamma و Degree را ست می‌کنیم. به دلیل شباهت با بخش قبل فقط نتایج را نمایش می‌دهم:

بخش ب-۱)

در تصویر زیر بهترین degree چند جمله‌ای و gamma را که توسط روش exhaustive پیدا کرده‌ایم را مشاهده می‌کنید. دقت متناظر با بهترین پارامترهای پیدا شده بر روی مجموعه‌ی validationو test و تعداد ساپورت وکتورها هم مشخص شده است.

```
Part A:
> Best Validation Accuracy is 87.1794871795 Which belongs to Degree:1, Gamma:1000000
Accuracy = 71.7949% (28/39) (classification)
> Accuracy of SVM (with best validation accuracy) on test set is 71.7948717949 ,
Which belongs to Degree:1, Gamma:1000000
Number Of SV: 117
```

بخش ب-۲)

در تصویر زیر بهترین degree چند جمله‌ای و gamma را که توسط روش جست و جوی تصادفی پیدا کرده‌ایم را مشاهده می‌کنید. دقت متناظر با بهترین پارامترهای پیدا شده بر روی مجموعه‌ی validationو test و تعداد ساپورت وکتورها هم مشخص شده است.

```
Part B:
> Best Validation Accuracy is 84.6153846154 Which belongs to C:5, Gamma:0.0001
Accuracy = 76.9231% (30/39) (classification)
> Accuracy of SVM (with best validation accuracy) on test set is 76.9230769231 ,
Which belongs to Degree:5, Gamma:0.0001
Number Of SV: 46
```

بخش ج) کدهای این بخش در problem2-c.pyاست. در این قسمت به جای استفاده از یک کرنل RBF از یک کرنل sigmoid استفاده می‌کنیم، همه چیز مانند قسمت قبل است. به دلیل شباهت با بخش قبل فقط نتایج را نمایش می‌دهم:

بخش ج-۱)

در تصویر زیر بهترین C و gamma را که توسط روش exhaustive پیدا کرده‌ایم را مشاهده می‌کنید. دقت متناظر با بهترین پارامترهای پیدا شده بر روی مجموعه‌ی validationو test و تعداد ساپورت وکتورها هم مشخص شده است.

```
Part A:
> Best Validation Accuracy is 92.3076923077 Which belongs to C:100000000, Gamma:1e-08
Accuracy = 84.6154% (33/39) (classification)
> Accuracy of SVM (with best validation accuracy) on test set is 84.6153846154 ,
Which belongs to C:100000000, Gamma:1e-08
Number Of SV: 34
```

بخش ج-۲)

در تصویر زیر بهترین C و gamma را که توسط روش جست و جوی تصادفی پیدا کرده‌ایم را مشاهده می‌کنید. دقت متناظر با بهترین پارامترهای پیدا شده بر روی مجموعه‌ی validationو test و تعداد ساپورت وکتورها هم مشخص شده است.

```
Part B:
> Best Validation Accuracy is 89.7435897436 Which belongs to C:10000000, Gamma:1e-07
Accuracy = 79.4872% (31/39) (classification)
> Accuracy of SVM (with best validation accuracy) on test set is 79.4871794872 ,
Which belongs to C:10000000, Gamma:1e-07
Number Of SV: 34
```

بخش د) کدهای این بخش در problem2-d.py است. پارامتر C وزن مجموع Slack variable ها است که مشخص می‌کند چه میزان ساپورت وکتورها می‌توانند margin ها را زیر در نظر نگیرند و وارد margin شوند. هر چه مقدار C بیشتر باشد وزن خطا بیشتر می‌شود در نتیجه SVM میزان کمترین به نمونه‌ها اجازه می‌دهد تا وارد margin شوند و اگر بسیار بزرگ باشد soft margin SVM تبدیل به Hard margin SVM می‌شود. ولی اگر C کم باشد وزن خطا هم کم می‌شود و SVM میزان بیشتری به نمونه‌ها اجازه می‌دهد تا وارد Margin شوند در نتیجه ئهنای مارجین بیشتر خواهد بود.

در فایل problem2-d.py کدی است که از یک کرنل RBF استفاده می‌کند. مقدار gamma ثابت و برابر ۰,۰۲ فرض شده است و مقدار c از ۱۰ به توان ۲۰- تا ۱۰ به توان ۲۰ تغییر می‌کند. میزان خطای مدل را به ازای C های مختلف مشاهده می‌کنید:

```
C:  10.0                    Accuracy:  71.7948717949
C:  100.0                   Accuracy:  71.7948717949
C:  1000.0                  Accuracy:  71.7948717949
C:  10000.0                     Accuracy:  71.7948717949
C:  100000.0                    Accuracy:  71.7948717949
C:  1000000.0                   Accuracy:  71.7948717949
C:  10000000.0                  Accuracy:  71.7948717949
C:  100000000.0                     Accuracy:  71.7948717949
C:  1000000000.0                    Accuracy:  71.7948717949
C:  10000000000.0                   Accuracy:  71.7948717949
C:  1e+11                   Accuracy:  71.7948717949
C:  1e+12                   Accuracy:  71.7948717949
C:  1e+13                   Accuracy:  71.7948717949
C:  1e+14                   Accuracy:  71.7948717949
C:  1e+15                   Accuracy:  71.7948717949
C:  1e+16                   Accuracy:  71.7948717949
C:  1e+17                   Accuracy:  71.7948717949
C:  1e+18                   Accuracy:  71.7948717949
C:  1e+19                   Accuracy:  71.7948717949
C:  0.01                    Accuracy:  74.358974359
C:  1e-20                   Accuracy:  74.358974359
C:  1e-19                   Accuracy:  74.358974359
C:  1e-18                   Accuracy:  74.358974359
C:  1e-17                   Accuracy:  74.358974359
C:  1e-16                   Accuracy:  74.358974359
C:  1e-15                   Accuracy:  74.358974359
C:  1e-14                   Accuracy:  74.358974359
C:  1e-13                   Accuracy:  74.358974359
C:  1e-12                   Accuracy:  74.358974359
C:  1e-11                   Accuracy:  74.358974359
C:  1e-10                   Accuracy:  74.358974359
C:  1e-09                   Accuracy:  74.358974359
C:  1e-08                   Accuracy:  74.358974359
C:  1e-07                   Accuracy:  74.358974359
C:  1e-06                   Accuracy:  74.358974359
C:  1e-05                   Accuracy:  74.358974359
C:  0.0001                  Accuracy:  74.358974359
C:  0.001                   Accuracy:  74.358974359
C:  0.1                     Accuracy:  74.358974359
```

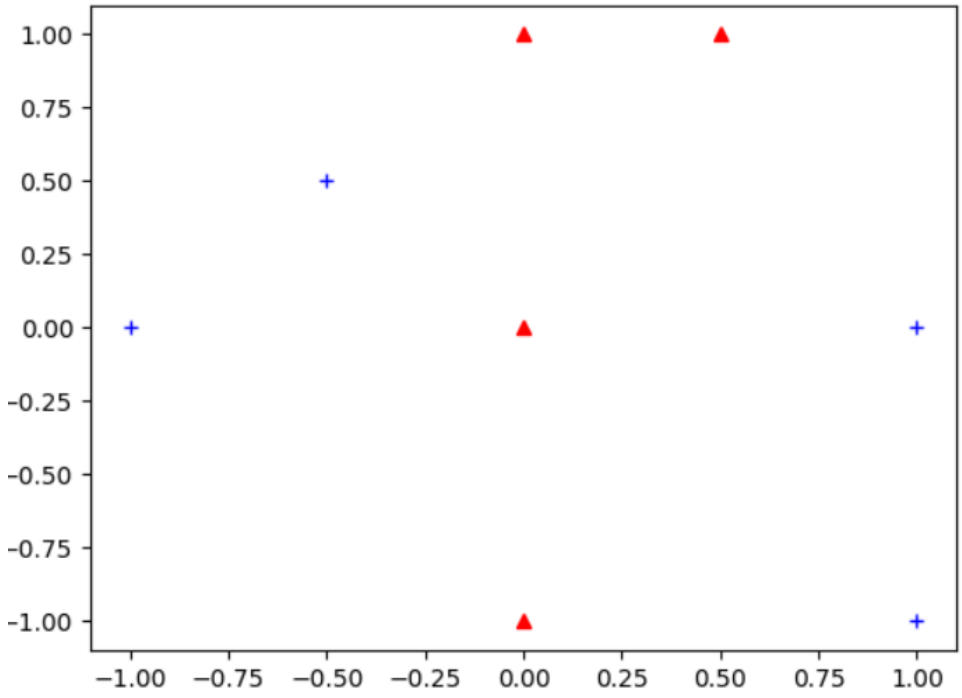
وقتی C برابر با ۱۰ به‌توان ۱۹ است دقت مدل برابر با ۷۱,۷۹ است و زمانی که میزان C برابر با ۱۰ به توان ۲۰- است دقت الگوریتم برابر است با ۷۴,۳۵ . هر چه C بیشتر باشد نمونه‌ها به میزان کم‌تری می‌توانند از margin عبور کنند و آن را زیر پا بگذارند در نتیجه عرض مرز تصمیم کوچک‌تر می‌شود و مدل بیشتر Over fit می‌شود.

بخش ه) در قسمت‌های قبل علاوه بر پارامترها و دقت مدل بر روی مجموعه‌ی تست و آموزش تعداد ساپورت وکتورها را هم گزارش کرده‌ام.

سوال ۳:

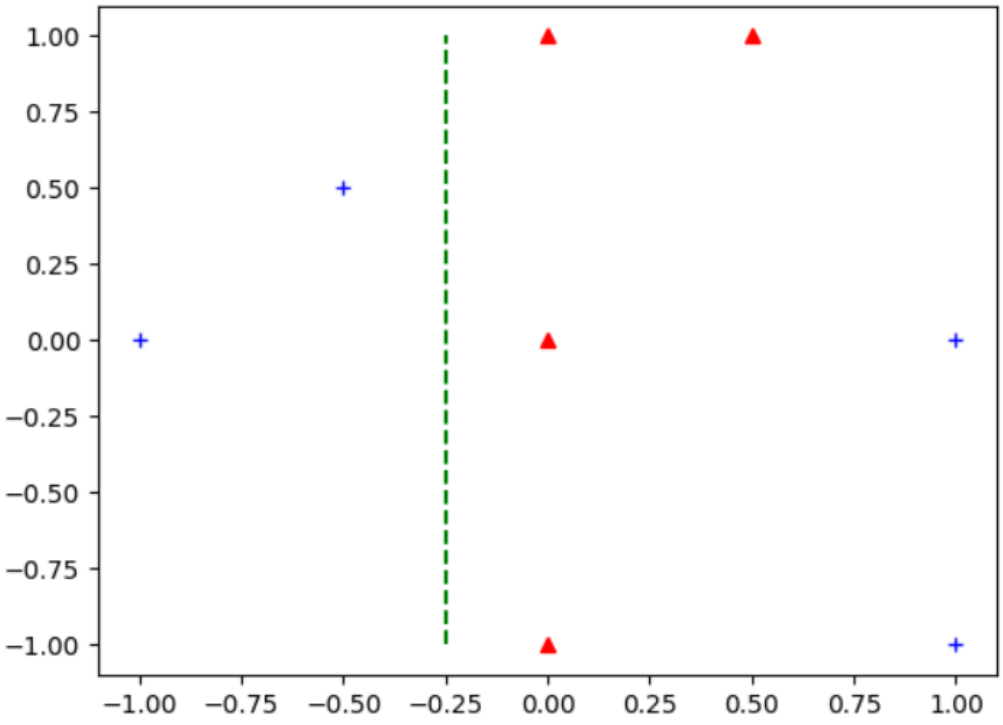
بخش الف) الگوریتم آدابوست در برابر نویز بسیار حساس است. فرض کنید که یک داده‌ی مثبت کاملا در قسمت منفی قرار گرفته باشد. در هر بار دسته‌بندی وزن آن مرتب افزایش پیدا می‌کند تا زمانی که درست دسته‌بندی شود. همین باعث می‌شود این الگوریتم به نویز بسیار حساس باشد.

بخش ب)



Data	Feature 0	Feature 1	Class
X1	-۱	۰	۱
X2	-۰.۵	۰.۵	۱
X3	۰	۱	-۱
X4	۰.۵	۱	-۱
X5	۱	۰	۱
X6	۱	-۱	۱
X7	۰	-۱	-۱
X8	۰	۰	-۱

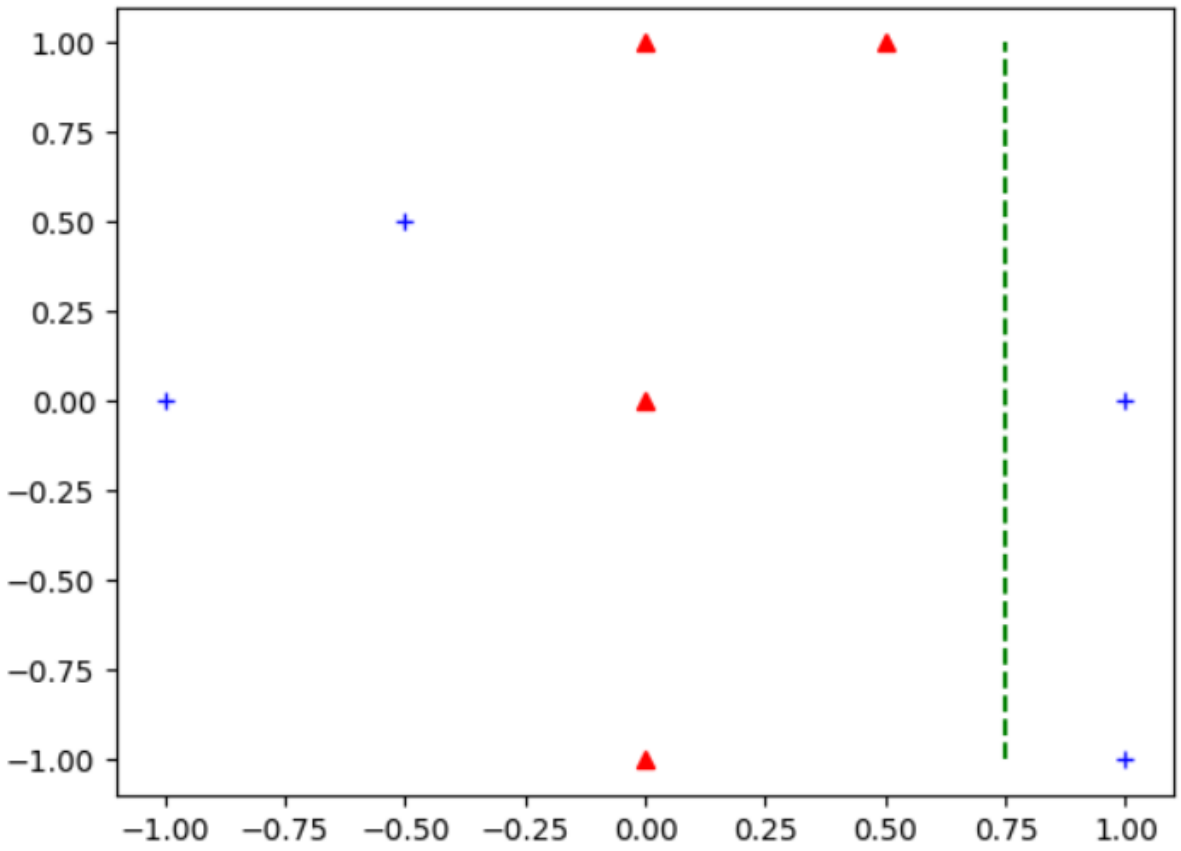
مرحله‌ی یک:



مرز تصمیم برابر است با feature 0 = -0.25

Data	W0	Classification	W1
X1	۰.۱۲۵	Correct	۰,۰۸۳۳
X2	۰.۱۲۵	Correct	۰,۰۸۳۳
X3	۰.۱۲۵	Correct	۰,۰۸۳۳
X4	۰.۱۲۵	Correct	۰,۰۸۳۳
X5	۰.۱۲۵	Misclassification	۰.۲۵
X6	۰.۱۲۵	Misclassification	۰.۲۵
X7	۰.۱۲۵	Correct	۰,۰۸۳۳
X8	۰.۱۲۵	Correct	۰,۰۸۳۳
Error	**Weight of misclassified= 0.25**		
Alpha1	**۰.۵*Ln((1-e)/e) = 0.5493**		

مرحله‌ی دو:

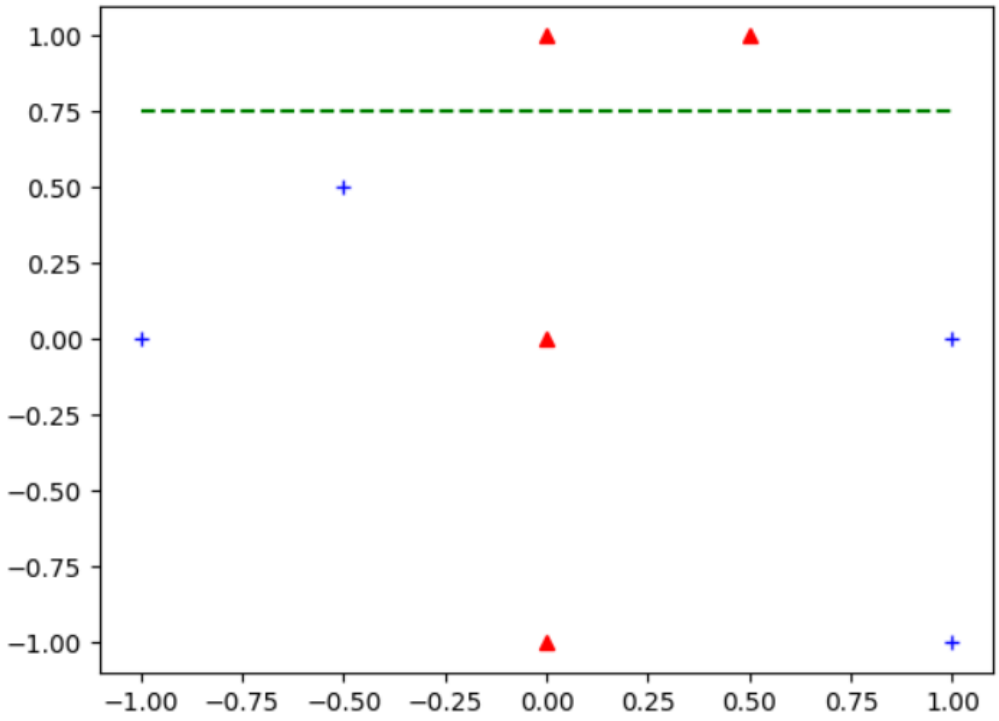


مرز تصمیم برابر است با feature 0 = 0.75

Data	W1	Classification	W2
X1	۰,۰۸۳۳	Misclassification	۰.۲۵
X2	۰,۰۸۳۳	Misclassification	۰.۲۵
X3	۰,۰۸۳۳	Correct	۰.۰۴۹۹
X4	۰,۰۸۳۳	Correct	۰.۰۴۹۹
X5	۰.۲۵	Correct	۰.۱۴۹۹
X6	۰.۲۵	Correct	۰.۱۴۹۹
X7	۰,۰۸۳۳	Correct	۰.۰۴۹۹
X8	۰,۰۸۳۳	Correct	۰.۰۴۹۹
Error	**Weight of misclassified= 0.1666**		

Alpha2	**۰.۵*Ln((1-e)/e) = 0.8049**

مرحله‌ی سه:



مرز تصمیم برابر است با feature 1 = 0.75

Data	W2	Classification	W3
X1	۰.۲۵	Correct	۰.۱۳۸۸
X2	۰.۲۵	Correct	۰.۱۳۸۸
X3	۰.۰۴۹۹	Correct	۰.۰۲۷۷
X4	۰.۰۴۹۹	Correct	۰.۰۲۷۷
X5	۰.۱۴۹۹	Correct	۰.۰۸۳۲
X6	۰.۱۴۹۹	Correct	۰.۰۸۳۲
X7	۰.۰۴۹۹	Misclassification	۰.۲۵
X8	۰.۰۴۹۹	Misclassification	۰.۲۵
Error	**Weight of misclassified= 0.0998**		
Alpha3	**۰.۵*Ln((1-e)/e) = 1.0997**		