به نام خدا

نگین اسماعیل زاده ۹۷۱۰۴۰۳۴

تمرین SVM درس هوش مصنوعی

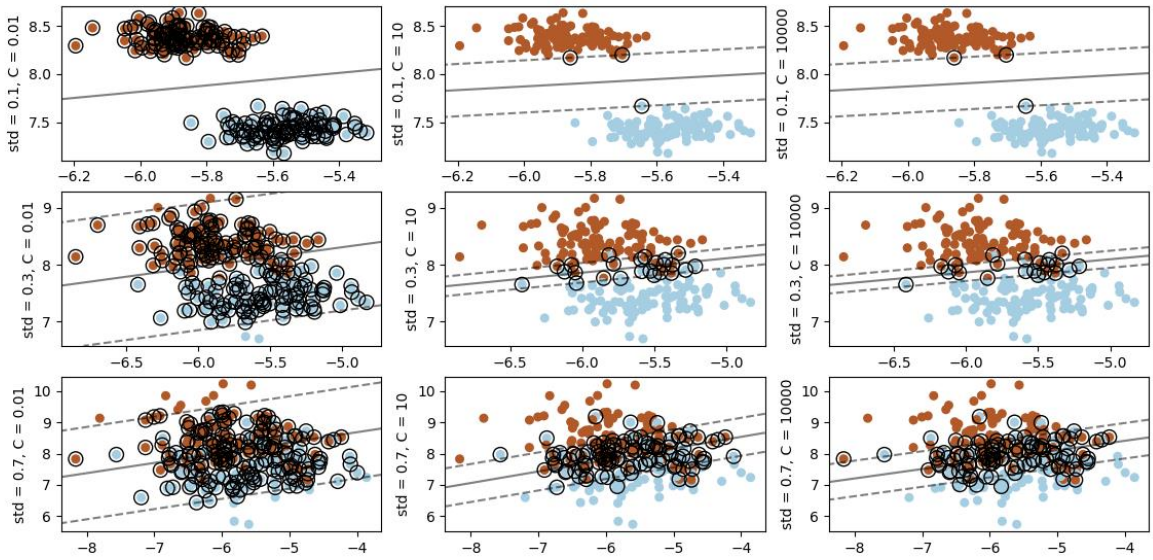
دکتر عبدی

بخش اول

در این بخش سه نمونه جمعیت داده دو کلاسه توسط سه تابع make_circles ، make_blobs و یک تابع دیگر که ربع اول وسوم را در یک دسته و ربع دوم و چهارم را در یک دسته دیگر قرار میدهد تولید کردیم. سپس با استفاده از ابزار آماده ی SVM در کتابخانه ی sklearn طبقه بندی را با تنظیم پارامتر های مختلف انجام داده و نتایج را گزارش کردیم.

نمونه ی اول :

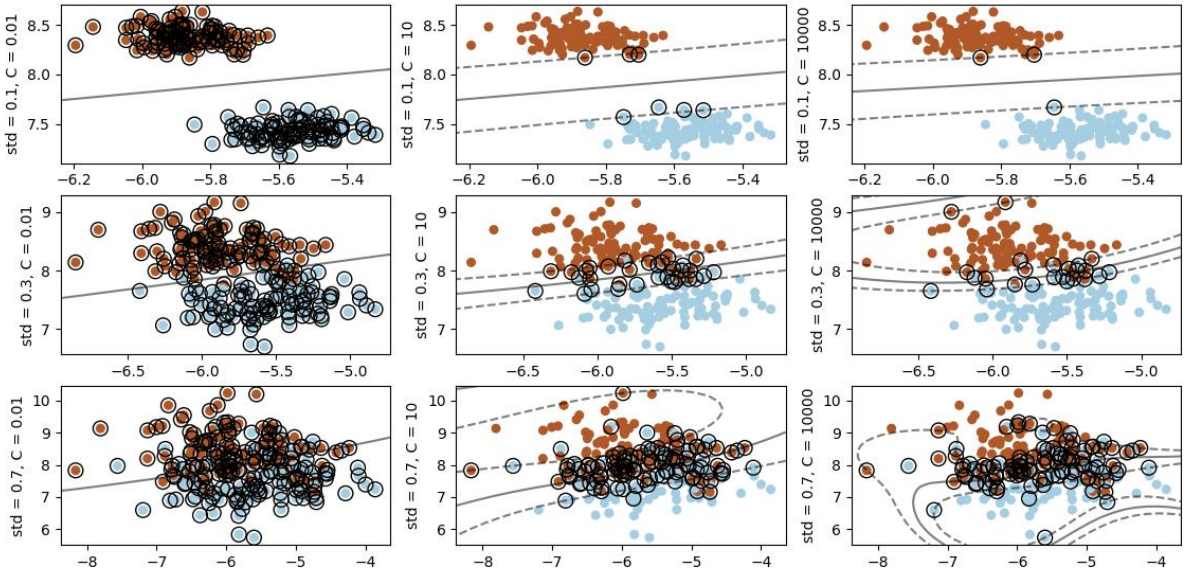
اگر از هسته ی linear برای طبقه بندی دیتای اول اسفتاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7 و C = 0.01,10,1000 به صورت زیر است :



همانطور که میبینم هر چه دیتا مخلوط تر باشد یعنی std دیتا بیشتر باشد طبقه بندی آن سخت تر است اما SVM به راحتی بهترین طبق بند خطی جدا کننده را پیدا کرد. همچنین با مقایسه متوجه میشویم که پارامتر C باید درست تعیین شود، اگر C بسیار کوچک باشد SVM درجه آزادی زیادی برای خطا قائل میشود و در دیتای کاملا تفکیک شده ی خطی نیز طبقه بندی خوب انجام نمیشود چون تقریبا همه ی نقاط میتوانند خطادار باشند و در واقع هر طبقه بندی دلخواهی جواب است، از طرفی اگر C بسیار بزرگ باشد هم طبقه بند ایده آل نیست چون دیتا ی مخلوط اجازه خطا به داده ها نمیدهد و به نوعی بیش برازش انجام میدهد. در شکل بالا و شکل های بعدی خواهیم دید که انتخاب حد متوسطی برای C متناسب با مسئله (مثلا در اینجا C = 10) پاسخ بسیار مناسبی ایجاد میکند.

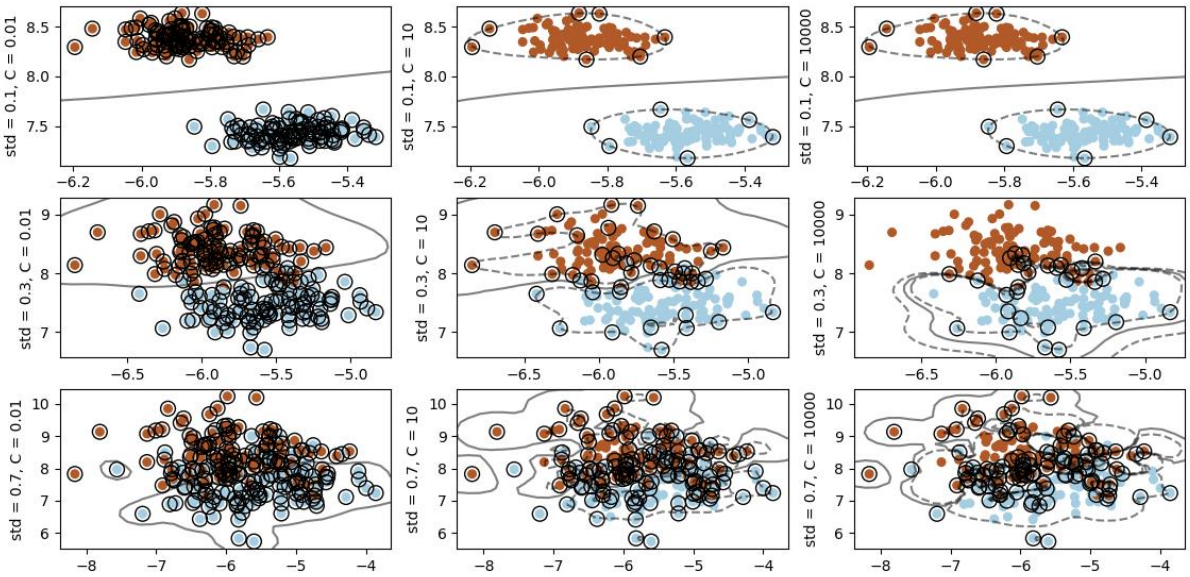
اگر از هسته ی rbf برای طبقه بندی دیتای اول اسفتاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7

و C = 0.01,10,1000 با در نظر گرفتن gamma = 0.1 به صورت زیر است :



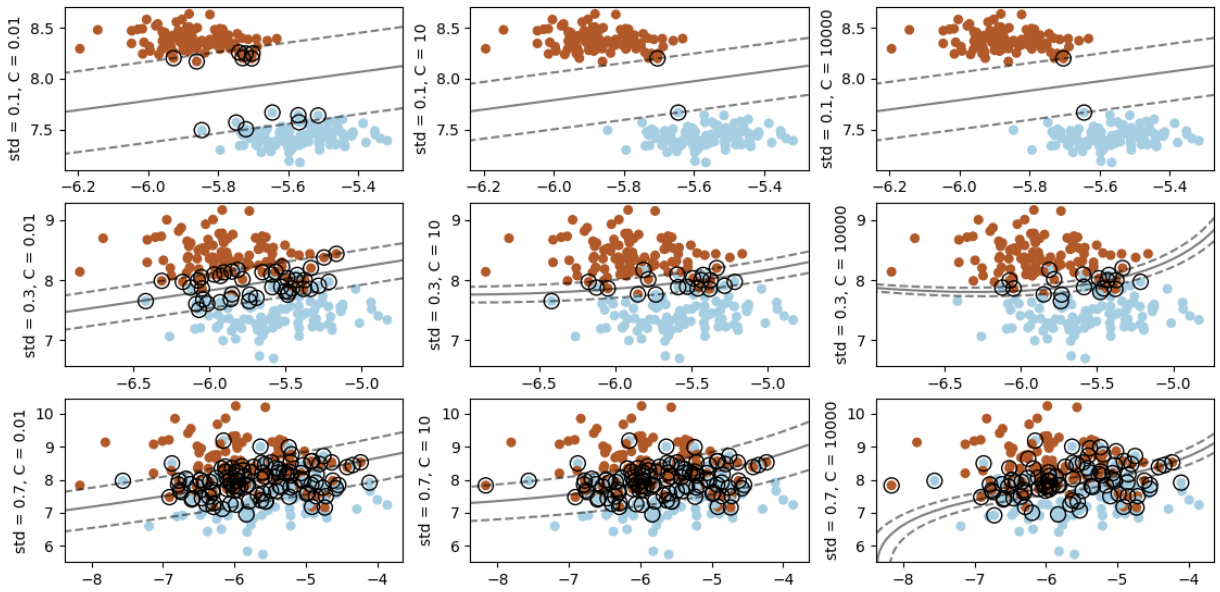
اگر از هسته ی rbf برای طبقه بندی دیتای اول اسفتاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7

و C = 0.01,10,1000 با در نظر گرفتن gamma = 10 به صورت زیر است :



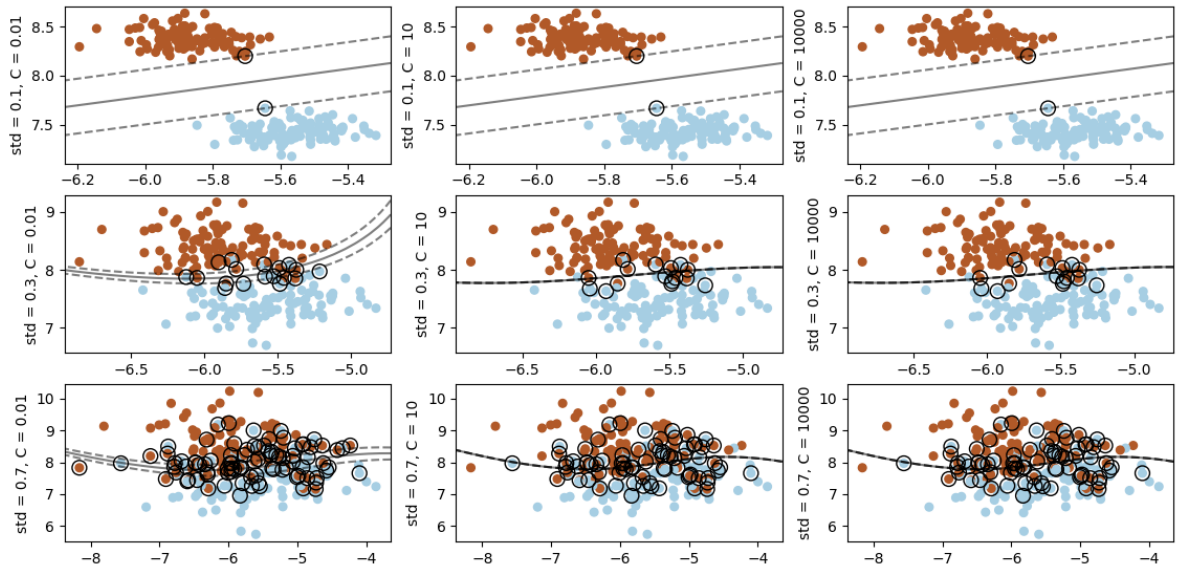
اگر از هسته ی poly برای طبقه بندی دیتای اول اسفتاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7

و C = 0.01,10,1000 با در نظر گرفتن gamma = 0.1 به صورت زیر است :



اگر از هسته ی poly برای طبقه بندی دیتای اول اسفتاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7

و C = 0.01,10,1000 با در نظر گرفتن gamma = 10 به صورت زیر است :

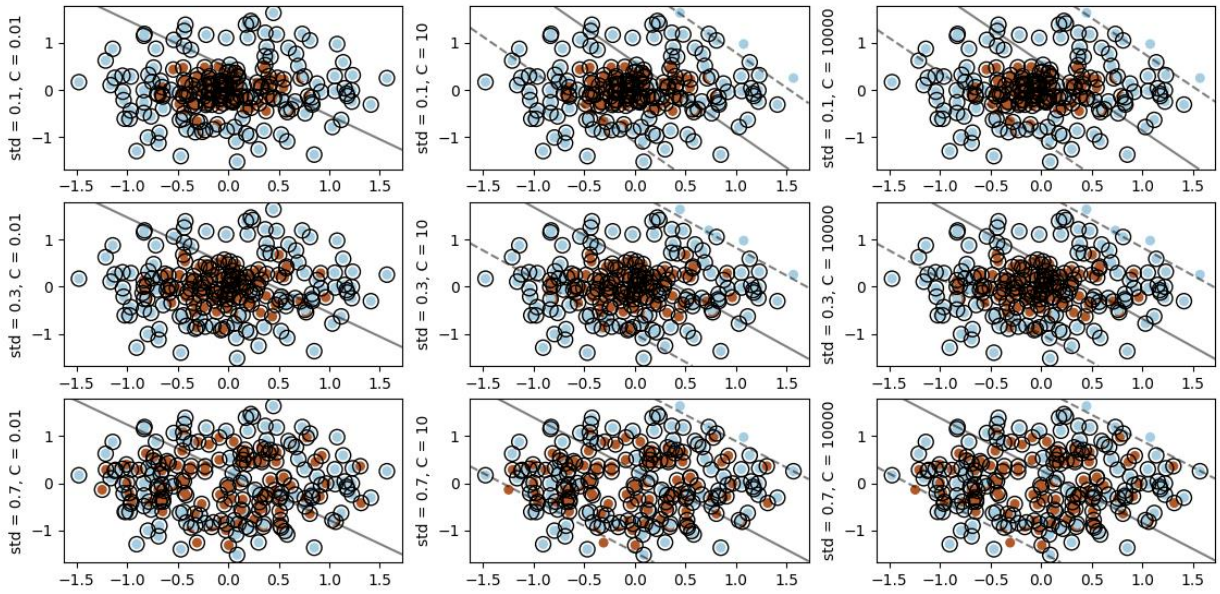


همانطور که مشاهده میکنیم با انتخاب مناسب پارامتر های gamma و C و هسته ی مناسب میتوانیم طبقه بندی بسیار خوبی انجام دهیم. برای مثال برای این نوع دیتا هسته ی linear با C = 10 و هسته ی poly با C = 10 و gamma = 10 بسیار تفکیک خوبی برای هر std از دیتا ایجاد کرده اند.

نمونه ی دوم :

اگر از هسته ی linear برای طبقه بندی دیتای دوم استفاده کنیم نتیجه برای ۹ حالت

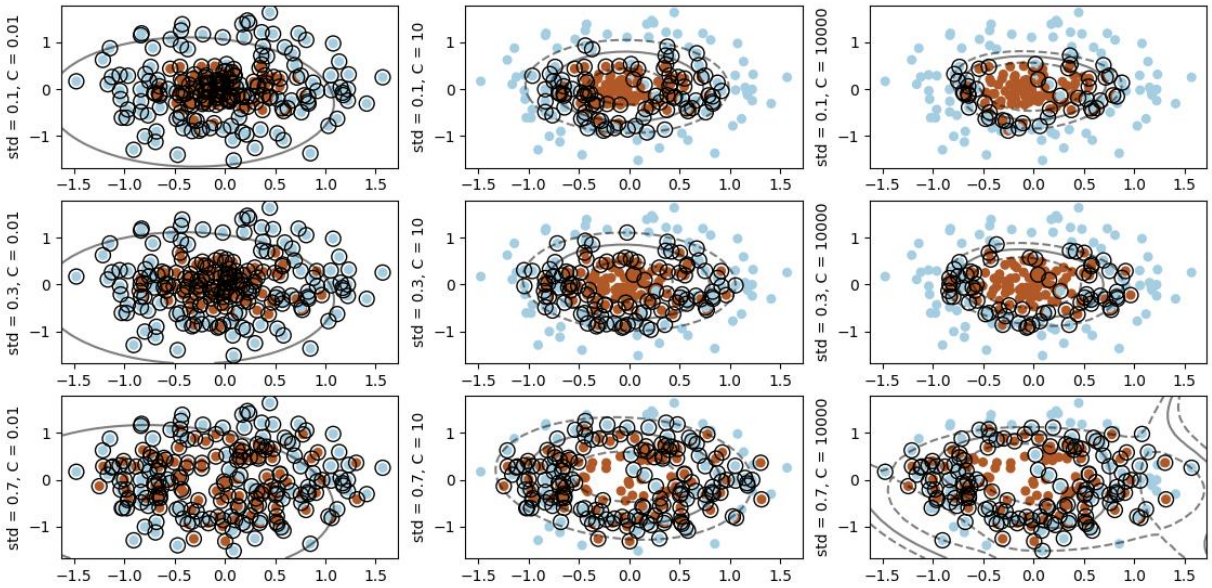
factor = 0.1,0.2,0.7 و C = 0.01,10,1000 به صورت زیر است :



همانطور که میبینم هر چه دیتا مخلوط تر باشد یعنی factor دیتا بیشتر باشد طبقه بندی آن سخت تر است و SVM در حالت هسته خطی نمیتواند این دیتا را به درستی طبقه بندی کند. اما با تغییر هسته خواهیم دید این کار به آسانی ممکن است!

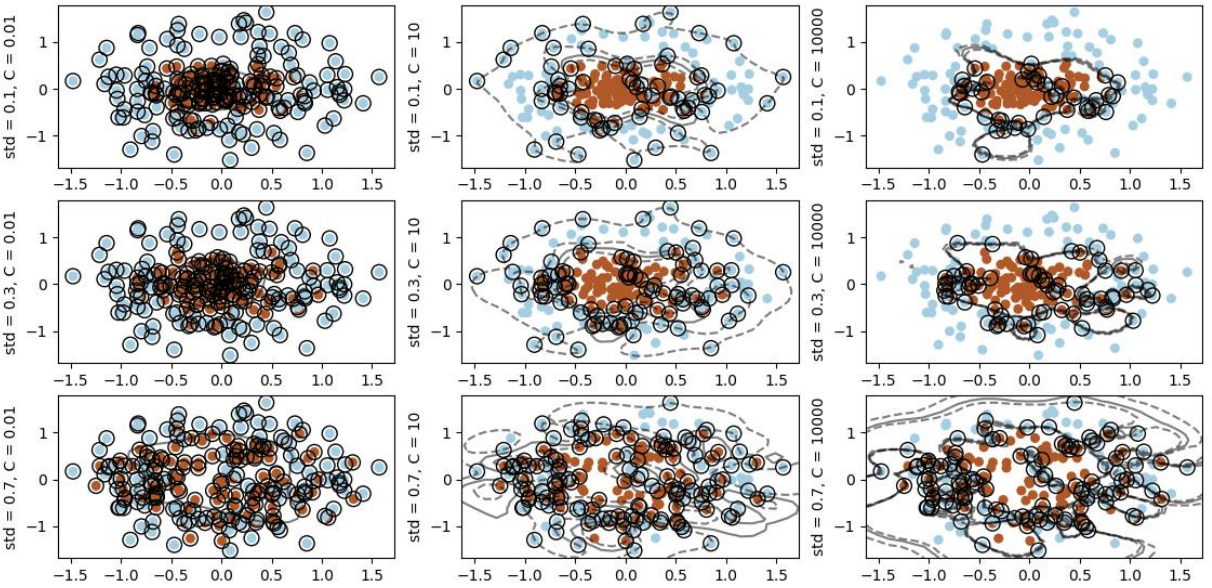
اگر از هسته ی rbf برای طبقه بندی دیتای دوم استفاده کنیم نتیجه برای ۹ حالت factor = 0.1,0.2,0.7

و C = 0.01,10,1000 با در نظر گرفتن gamma = 0.1 به صورت زیر است :

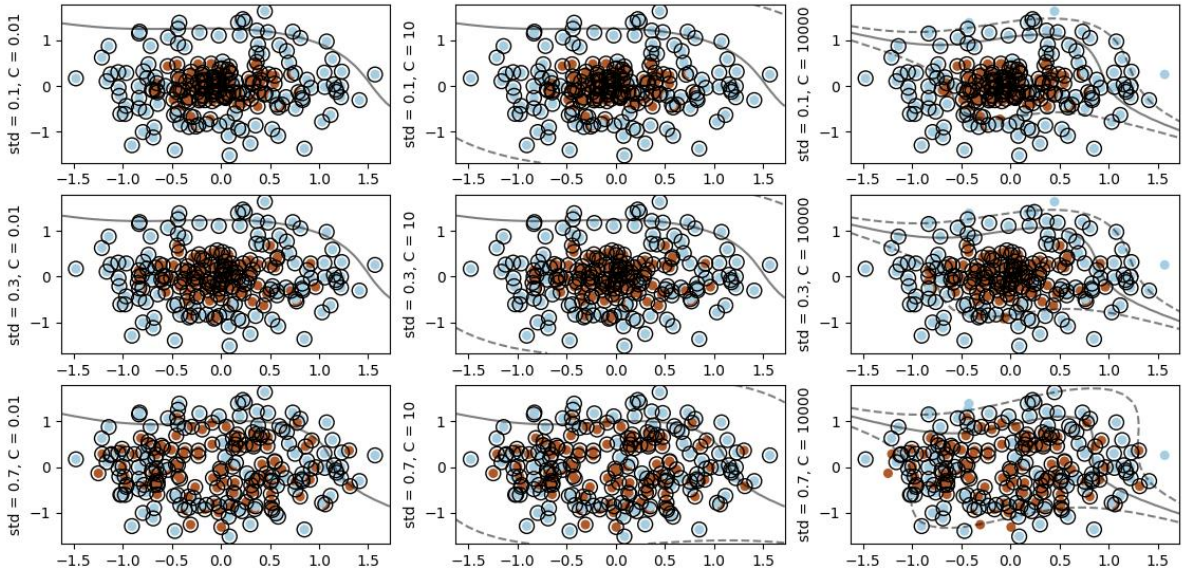


اگر از هسته ی rbf برای طبقه بندی دیتای دوم استفاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7

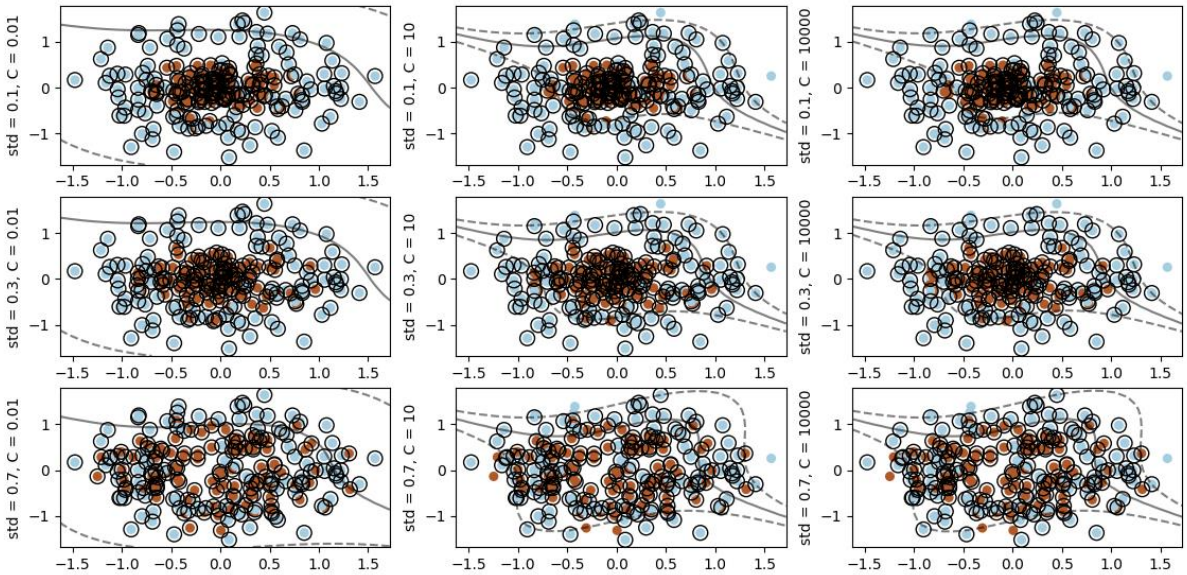
و C = 0.01,10,1000 با در نظر گرفتن gamma = 10 به صورت زیر است :



اگر از هسته ی poly برای طبقه بندی دیتای دوم استفاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7 و C = 0.01,10,1000 با در نظر گرفتن gamma = 0.1 به صورت زیر است :



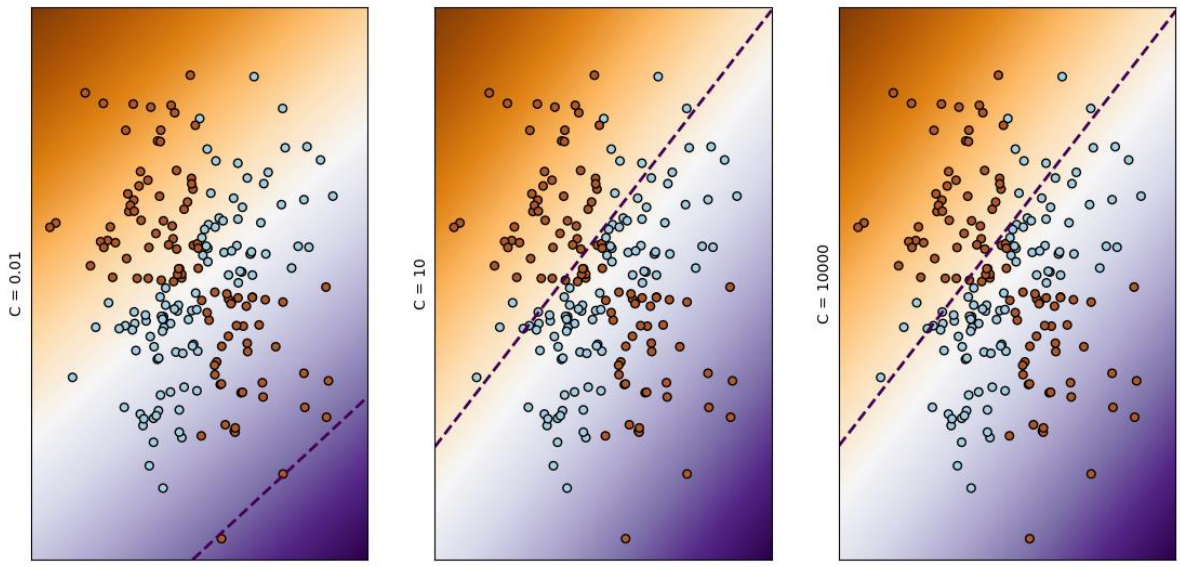
اگر از هسته ی poly برای طبقه بندی دیتای دوم استفاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7 و C = 0.01,10,1000 با در نظر گرفتن gamma = 1 به صورت زیر است :



همانطور که مشاهده میکنیم با انتخاب مناسب پارامتر های gamma و C و هسته ی مناسب میتوانیم طبقه بندی بسیار خوبی انجام دهیم. برای این مثال دیدیم که باقی هسته ها غیر از rbf نمیتوانند طبقه بندی را انجام دهند اما حتی اینچنین دیتای پیچیده ای را هم میبینیم که توسط هسته ی rbf با C = 10 و gamma = 0.1 بسیار تفکیک خوبی برای هر std از دیتا میتوانیم ایجاد کنیم (که در شکل ها مشخص است).

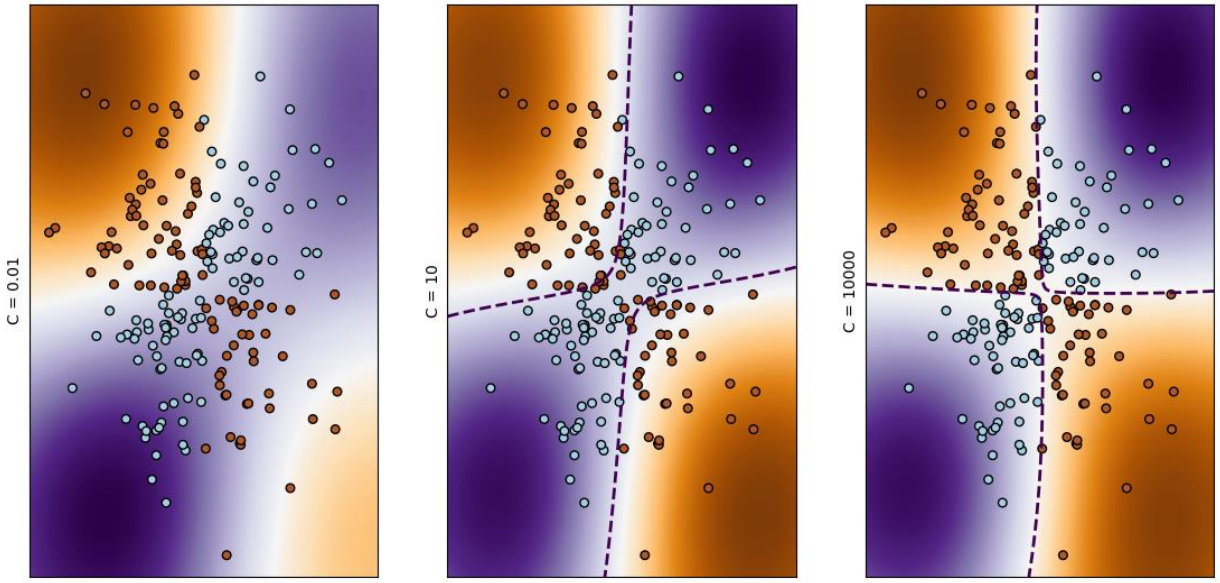
نمونه ی سوم :

اگر از هسته ی linear برای طبقه بندی دیتای سوم استفاده کنیم نتیجه برای ۳ حالت C = 0.01,10,1000 به صورت زیر است :

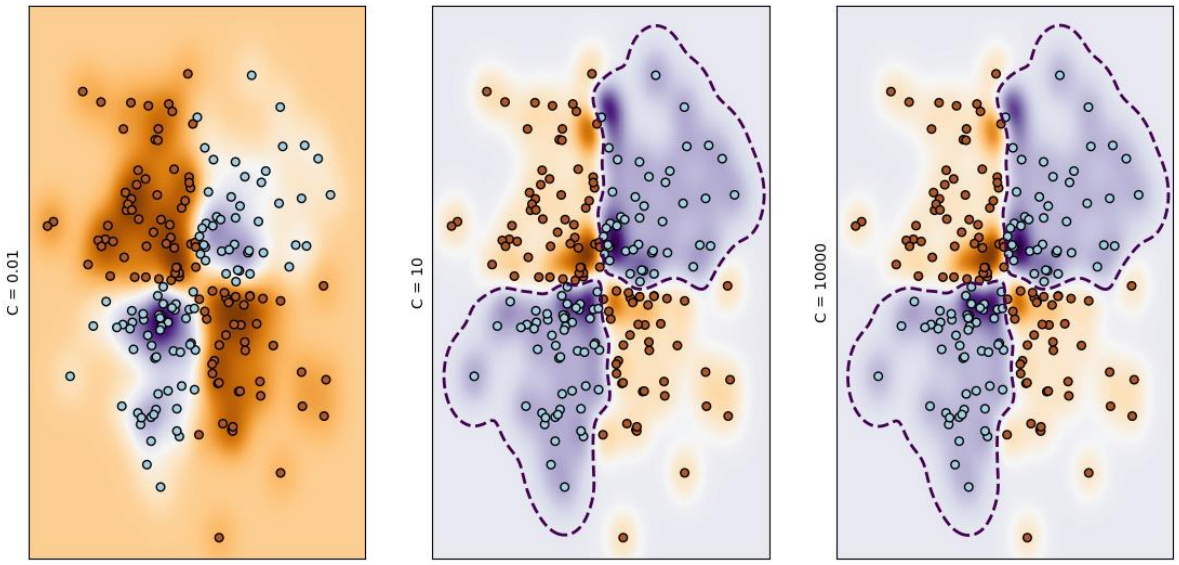


همانطور که میبینم دیتا ی سوم یک نوع در هم بودن از جنس xor دارد و با یک خط تفکیک پذیر نیست. بنابراین بهتر از هسته های دیگری کمک بگیریم.

اگر از هسته ی rbf برای طبقه بندی دیتای سوم استفاده کنیم نتیجه برای ۳ حالت C = 0.01,10,1000 با gamma = 0.1 به صورت زیر است :

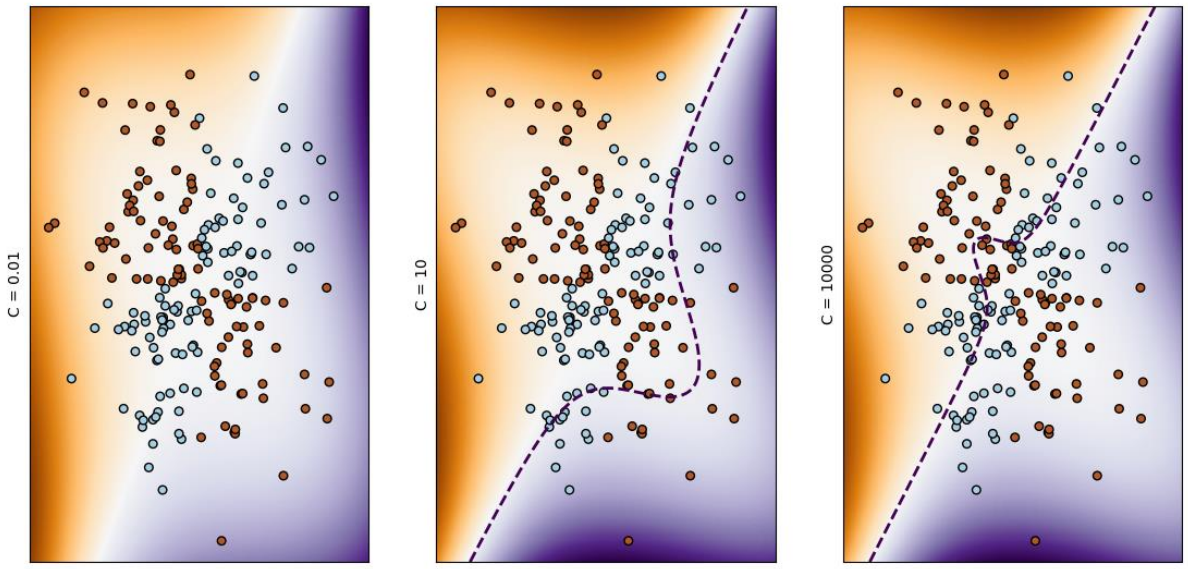


اگر از هسته ی rbf برای طبقه بندی دیتای دوم استفاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7 و C = 0.01,10,1000 با در نظر گرفتن gamma = 10 به صورت زیر است :



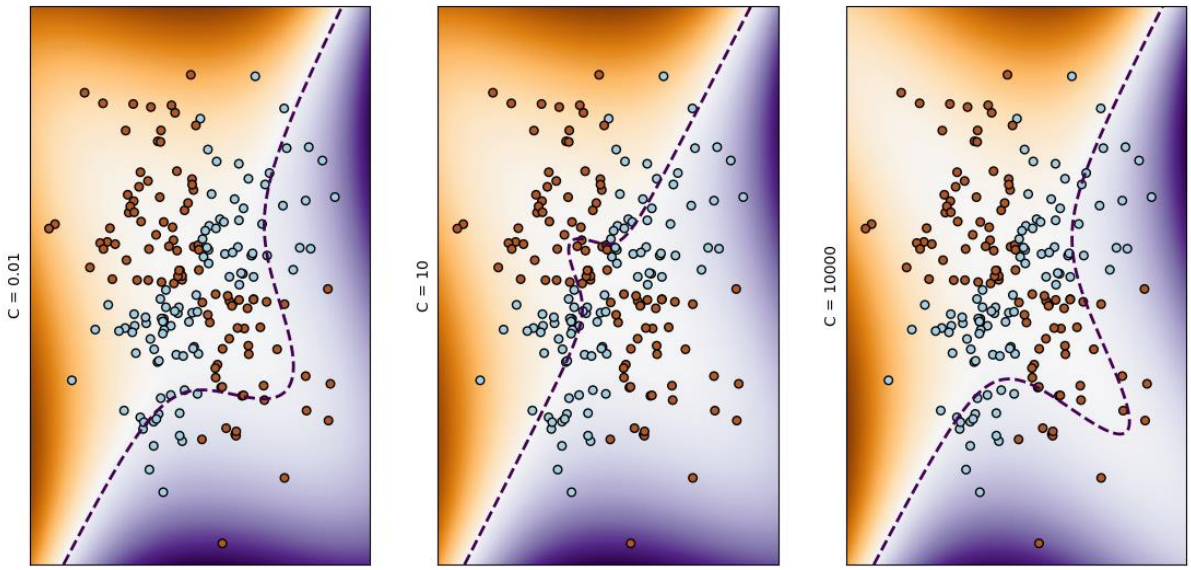
اگر از هسته ی poly برای طبقه بندی دیتای دوم استفاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7

و C = 0.01,10,1000 با در نظر گرفتن gamma = 0.1 به صورت زیر است :



اگر از هسته ی poly برای طبقه بندی دیتای دوم استفاده کنیم نتیجه برای ۹ حالت std = 0.1,0.2,0.7

و C = 0.01,10,1000 با در نظر گرفتن gamma = 1 به صورت زیر است :



همانطور که مشاهده میکنیم با انتخاب مناسب پارامتر های gamma و C و هسته ی مناسب میتوانیم طبقه بندی بسیار خوبی در یان دیتاست نیز انجام دهیم. برای این مثال دیدیم که باقی هسته ها غیر از rbf نمیتوانند طبقه بندی ایده آل را انجام دهند اما حتی اینچنین دیتای پیچیده ای را هم میبینیم که توسط هسته ی rbf با با C = 1000 و gamma = 0.1 بسیار تفکیک خوبی برای هر std از دیتا میتوانیم ایجاد کنیم (که در شکل ها مشخص است).

تفاسیر :

۱) همانطور که دیدیماکثر دیتا ها با یک هسته ی rbf و با انتخاب پارامتر های مناسب قابل تفکیک به صورت ایده آل!! هستند. زیرا حتی یک تفکیک کنند ی خطی نیزمیتواند حالت خاص شعاعی باشد (شعاع بسیار بزرگ) بنابراین هسته ی rbf در SVM بسیار ابزار کاربردی در مسائل طبقه بندی می باشد. البته برای یک طبقه بندی خطی یا یک طبقه بندی که بیشتر جنس poly دارد طبعا استفاده از خود این دو کرنل بسیار دقت بالاتری نسبت به حالت تقریبی با استفاده از rbf ایجاد میکند.

۲) هر چه در هم تنیده بودن دیتا بیشتر میشود پارامتر های هسته های مختلف ناچار به تغییر اند. به طور کلی همانطور که دیدم و صحبت شد در دیتا های ساده تر نسبت به دیتا های پیچیده تر C کوچکتری لازم است چون نیاز نداریم به خطا بهای بسیاری دهیم. اما انتخاب C های خیلی کوچک موجب غلط شدن طبقه بندی و در نظر نگرفتن خطا های ناشی از در هم تنیده بودن می باشد بنابر این به طور کلی C بزرگتر عموما نتیجه بهتری میدهد چون در حالت مینمم خطای داده ها طبقه بندی را انجام میدهد. همچنین پارامتر gamma که برای دو هسته ی poly و rbf کاربرد داشت میزان پراکندگی اعضای داخل یک دسته را نشان میدهد بنابر این اگر واریانس داخل دسته هایمان زیاد باشد (داده های دارای پراکندگی بیشتر) بهتر از gamma ی بزرگی در نظر بگیریمدر حالی که این موضوع در داده های منسجم میتواند بر عکس باشد.

۳) بطور کلی در مورد این سه دیتاستی که استفاده شد دیدیم که در مورد دیتا ست اول با هر میزان از پیچیدگی(در هم تنیدگی) SVM با هسته های poly و linear بسیار قابلیت تفکیک مناسبی دارد، دیتا ست دوم و سوم هم با هسته ی rbf کاملا با کمترین خطا قابل تفکیک و طبقه بندی درست هستند.

بخش دوم

در این بخش مشابه پروژه شبکه عصبی دیتا لود و استفاده شد. برای طبقه بند SVM از چهار هسته ی poly, rbf, linear, sigmoid استفاده شد با پارامتر های مختلف. برای مقایسه ی آزمایش ها از معیار عددی accuracy استفاده شده است که برای اعتبار بیشتر با روش 5 fold cross validation روی داده های تست اندازه گیری شده است. نسبت تعداد داده های آموزش به تست در تمامی آزمایش ها ثابت و برابر با ۷۰ به ۳۰ در نظر گرفته شده است.

آزمایش اول : پیدا کردن جواب بهینه برای هسته ی rbf

```
Kernel = RBF, gammma = 1e-08
Accuracy =  0.9410893730546911 with std =  0.011320166999413411
Kernel = RBF, gammma = 1e-07
Accuracy =  0.9534118867644878 with std =  0.009288248039087104
Kernel = RBF, gammma = 1e-06
Accuracy =  0.9634207796057506 with std =  0.008937083022400307
Kernel = RBF, gammma = 1e-05
Accuracy =  0.3192129835482437 with std =  0.004665285820947464
Kernel = RBF, gammma = 0.0001
Accuracy =  0.2144775455758115 with std =  0.0025461333328603854
Kernel = RBF, gammma = 0.001
Accuracy =  0.17520231213872833 with std =  0.00016518716955185087
```

```
Kernel = RBF, gammma = 1e-06, C = 0.1
Accuracy =  0.8348147324736919 with std =  0.012116780044802258
Kernel = RBF, gammma = 1e-06, C = 1
Accuracy =  0.9614932562620423 with std =  0.00834740340192655
Kernel = RBF, gammma = 1e-06, C = 10
Accuracy =  0.9634207796057506 with std =  0.008937083022400307
Kernel = RBF, gammma = 1e-06, C = 100
Accuracy =  0.9634207796057506 with std =  0.008937083022400307
Kernel = RBF, gammma = 1e-06, C = 1000
Accuracy =  0.9634207796057506 with std =  0.008937083022400307
Kernel = RBF, gammma = 1e-06, C = 5000
Accuracy =  0.9634207796057506 with std =  0.008937083022400307
Kernel = RBF, gammma = 1e-06, C = 10000
Accuracy =  0.9634207796057506 with std =  0.008937083022400307
```

نتایج نشان میدهد که بیشترین دقت برای هسته ی rbf حدودا با gamma = 1e-6 و C >= 10 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.9634

آزمایش دوم : پیدا کردن جواب بهینه برای هسته ی poly

```
Kernel = Poly, gammma = 1e-08
Accuracy =  0.6830954498295538 with std =  0.016516437836057586
Kernel = Poly, gammma = 1e-07
Accuracy =  0.9387853860975248 with std =  0.010790154699017918
Kernel = Poly, gammma = 1e-06
Accuracy =  0.9337787164665776 with std =  0.010022587398438804
Kernel = Poly, gammma = 1e-05
Accuracy =  0.9337787164665776 with std =  0.010022587398438804
Kernel = Poly, gammma = 0.0001
Accuracy =  0.9337787164665776 with std =  0.010022587398438804
Kernel = Poly, gammma = 0.001
Accuracy =  0.9337787164665776 with std =  0.010022587398438804
```

```
Kernel = Poly, gammma = 1e-07, C = 0.1
Accuracy =  0.25760337927967986 with std =  0.006758133149708355
Kernel = Poly, gammma = 1e-07, C = 1
Accuracy =  0.6830954498295538 with std =  0.016516437836057586
Kernel = Poly, gammma = 1e-07, C = 10
Accuracy =  0.8991136801541426 with std =  0.009944133476947743
Kernel = Poly, gammma = 1e-07, C = 100
Accuracy =  0.9422447013487476 with std =  0.005801608731928485
Kernel = Poly, gammma = 1e-07, C = 1000
Accuracy =  0.9387853860975248 with std =  0.010790154699017918
Kernel = Poly, gammma = 1e-07, C = 5000
Accuracy =  0.9337787164665776 with std =  0.010022587398438804
Kernel = Poly, gammma = 1e-07, C = 10000
Accuracy =  0.9337787164665776 with std =  0.010022587398438804
```

نتایج نشان میدهد که بیشترین دقت برای هسته ی poly حدودا با gamma = 1e-7 و C = 100 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.9422

آزمایش سوم : پیدا کردن جواب بهینه برای هسته ی linear

```
Kernel = Linear, C = 0.1
Accuracy =  0.9372372906476952 with std =  0.011778024350305902
Kernel = Linear, C = 1
Accuracy =  0.9372372906476952 with std =  0.011778024350305902
Kernel = Linear, C = 10
Accuracy =  0.9372372906476952 with std =  0.011778024350305902
Kernel = Linear, C = 100
Accuracy =  0.9372372906476952 with std =  0.011778024350305902
Kernel = Linear, C = 1000
Accuracy =  0.9372372906476952 with std =  0.011778024350305902
Kernel = Linear, C = 5000
Accuracy =  0.9372372906476952 with std =  0.011778024350305902
Kernel = Linear, C = 10000
Accuracy =  0.9372372906476952 with std =  0.011778024350305902
```

نتایج نشان میدهد که بیشترین دقت برای هسته ی poly حدودا با C >= 0.1 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.93724

آزمایش سوم : پیدا کردن جواب بهینه برای هسته ی sigmoid

```
Kernel = Sigmoid, gammma = 1e-11
Accuracy =  0.7269994071439159 with std =  0.01881454195612725
Kernel = Sigmoid, gammma = 1e-10
Accuracy =  0.9233748332592263 with std =  0.006807914485133774
Kernel = Sigmoid, gammma = 1e-09
Accuracy =  0.9410886319845858 with std =  0.010220364320366949
Kernel = Sigmoid, gammma = 1e-08
Accuracy =  0.9376226471024159 with std =  0.01164959343909961
Kernel = Sigmoid, gammma = 1e-06
Accuracy =  0.5360034089224841 with std =  0.012584173318099946
Kernel = Sigmoid, gammma = 1e-05
Accuracy =  0.22909293019119606 with std =  0.025971961693986148
Kernel = Sigmoid, gammma = 0.0001
Accuracy =  0.1744323402993923 with std =  0.0009839960000248037
Kernel = Sigmoid, gammma = 0.001
Accuracy =  0.1744323402993923 with std =  0.0009839960000248037
Kernel = Sigmoid, gammma = 1
Accuracy =  0.1744323402993923 with std =  0.0009839960000248037
```

```
Kernel = Sigmoid, gammma = 1e-09, C =  0.1
Accuracy =  0.1744323402993923 with std =  0.0009839960000248037
Kernel = Sigmoid, gammma = 1e-09, C =  1
Accuracy =  0.3253757225433526 with std =  0.001748827473623122
Kernel = Sigmoid, gammma = 1e-09, C =  10
Accuracy =  0.7269994071439159 with std =  0.01881454195612725
Kernel = Sigmoid, gammma = 1e-09, C =  100
Accuracy =  0.9233748332592263 with std =  0.006807914485133774
Kernel = Sigmoid, gammma = 1e-09, C =  1000
Accuracy =  0.9410886319845858 with std =  0.010220364320366949
Kernel = Sigmoid, gammma = 1e-09, C =  5000
Accuracy =  0.9383941010819624 with std =  0.009799725241901307
Kernel = Sigmoid, gammma = 1e-09, C =  10000
Accuracy =  0.9376226471024159 with std =  0.011649593439099 61
```

نتایج نشان میدهد که بیشترین دقت برای هسته ی poly حدودا با gamma = 1e-9 و C = 1000 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.9411

نتایج :

۱) اگر این چهار آمایش را با هم مقایسه کنیم به این نتیجه میرسیم که با پارامتر های بهینه ی ذکر شده در هر حالت، در مجموع SVM با هسته های rbf و poly دقت بالاتری در طبقه بندی این دیتاست به نسبت هسته های linear و sigmoid داشتند. همچنین بهترین طبقه بندی با دقت تست بسییار خوب! توسط هسته ی rbf ایجاد شد.

۲) اگر این نتیجه را با نتیجه ی پروژه ی شبکه عصبی که روی همین دیتا ست انجام شد مقایسه کنیم به این نتیجه میرسم که اولا SVM توانست در حد بهترین شبکه های ارائه شده در شبکه عصبی طبقه بندی انجام دهد و ثانیا این طبقه بندی را با کمترین میزان پیچیدگی هم از نظر حجم محاسبات (زمان) هم از نظر تعیین پارامتر های بهینه داشت به این دلیل که در SVM ما لازم بود تا نهایتا ۲ یا ۳ پارامر را تنظیم کنیم تا ب بهترین جواب برسیم. این در حالیست که در شبکه عصبی تعداد زیادی پارامتر برای تنظیم وجود داشت و عملا رسیدن به جواب بهینه نیاز به میزان خوبی explore داشت.

بخش سوم

در این بخش همانند بخش قبل دیتا را لود میکنیم. مسئله با بخش قبل هیچ تفاوتی ندارد چون مجددا باید طبقه بندی بین حروف/ اعداد درست و حروف / اعدادی که تصویر نمایش میدهند انجام دهیم. در این بخش مجددا تمامی آزمایش های بخش قبل برای یافتن یک طبقه بند بهینه با SVM انجام شد و در هر آزمایش نسبت داده های تست به آموزش ۲۰ به ۸۰ در نظر گرفته شده است. تمامی دقت ها نیز مطابق بخش قبل به روش 5 fold cross validation محاسبه شده اند.

آزمایش اول : پیدا کردن جواب بهینه برای هسته ی rbf

```
kernel = RBF, Gamma = 1e-07
Accuracy =  0.920327868852459 with std =  0.03262327118129944
kernel = RBF, Gamma = 1e-06
Accuracy =  0.9370491803278689 with std =  0.019517149068619875
kernel = RBF, Gamma = 1e-05
Accuracy =  0.8642076502732241 with std =  0.012606262476387606
kernel = RBF, Gamma = 0.0001
Accuracy =  0.23513661202185793 with std =  0.00762603498335279
kernel = RBF, Gamma = 0.001
Accuracy =  0.2318032786885246 with std =  0.001873926579178387
kernel = RBF, Gamma = 0.01
Accuracy =  0.2318032786885246 with std =  0.001873926579178387
kernel = RBF, Gamma = 0.1
Accuracy =  0.2318032786885246 with std =  0.001873926579178387
kernel = RBF, Gamma = 1
Accuracy =  0.2318032786885246 with std =  0.001873926579178387
kernel = RBF, Gamma = 10
Accuracy =  0.2318032786885246 with std =  0.001873926579178387
```

```
kernel = RBF , Gamma = 1e-05, C = 1e-06
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = RBF , Gamma = 1e-05, C = 1e-06
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = RBF , Gamma = 1e-05, C = 1e-05
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = RBF , Gamma = 1e-05, C = 0.0001
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = RBF , Gamma = 1e-05, C = 0.001
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = RBF , Gamma = 1e-05, C = 0.01
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = RBF , Gamma = 1e-05, C = 0.1
Accuracy =  0.5265027322404372 with std =  0.0062829649625375985
kernel = RBF , Gamma = 1e-05, C = 1
Accuracy =  0.9038797814207651 with std =  0.026879691993759117
kernel = RBF , Gamma = 1e-05, C = 10
Accuracy =  0.9337158469945355 with std =  0.02799051980298706
kernel = RBF , Gamma = 1e-05, C = 100
Accuracy =  0.9370491803278689 with std =  0.019517149068619875
kernel = RBF , Gamma = 1e-05, C = 10000
Accuracy =  0.9370491803278689 with std =  0.019517149068619875
kernel = RBF , Gamma = 1e-05, C = 10000
Accuracy =  0.9370491803278689 with std =  0.019517149068619875
kernel = RBF , Gamma = 1e-05, C = 100000
Accuracy =  0.9370491803278689 with std =  0.019517149068619875
kernel = RBF , Gamma = 1e-05, C = 1000000
Accuracy =  0.9370491803278689 with std =  0.019517149068619875
```

نتایج نشان میدهد که بیشترین دقت برای هسته ی rbf حدودا با gamma = 1e-5 و C >= 10 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.9337

آزمایش دوم : پیدا کردن جواب بهینه برای هسته ی poly

```
kernel = Poly, Gamma = 1e-07
Accuracy =  0.9369945355191257 with std =  0.019664236424364483
kernel = Poly, Gamma = 1e-06
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly, Gamma = 1e-05
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly, Gamma = 0.0001
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly, Gamma = 0.001
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly, Gamma = 0.01
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly, Gamma = 0.1
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly, Gamma = 1
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly, Gamma = 10
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
```

```
kernel = Poly , Gamma = 0.0001, C = 1e-06
Accuracy =  0.4337704918032787 with std =  0.005211636728990765
kernel = Poly , Gamma = 0.0001, C = 1e-06
Accuracy =  0.4337704918032787 with std =  0.005211636728990765
kernel = Poly , Gamma = 0.0001, C = 1e-05
Accuracy =  0.8144808743169399 with std =  0.030886162435787536
kernel = Poly , Gamma = 0.0001, C = 0.0001
Accuracy =  0.9271584699453552 with std =  0.032626474624561946
kernel = Poly , Gamma = 0.0001, C = 0.001
Accuracy =  0.9369945355191257 with std =  0.019664236424364483
kernel = Poly , Gamma = 0.0001, C = 0.01
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly , Gamma = 0.0001, C = 0.1
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly , Gamma = 0.0001, C = 1
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly , Gamma = 0.0001, C = 10
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly , Gamma = 0.0001, C = 100
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly , Gamma = 0.0001, C = 10000
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly , Gamma = 0.0001, C = 10000
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly , Gamma = 0.0001, C = 100000
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
kernel = Poly , Gamma = 0.0001, C = 1000000
Accuracy =  0.9336612021857922 with std =  0.021343257791857747
```

```
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 1
Accuracy =  0.9371584699453553 with std =  0.019124902404052305
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 2
Accuracy =  0.9404371584699455 with std =  0.013123858406363337
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 3
Accuracy =  0.9238797814207651 with std =  0.01293547775881342
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 4
Accuracy =  0.9205464480874317 with std =  0.012196202272678126
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 5
Accuracy =  0.9271584699453552 with std =  0.008030370254676013
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 6
Accuracy =  0.933879781420765 with std =  0.017764186031282058
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 7
Accuracy =  0.9074863387978143 with std =  0.036717733367179414
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 8
Accuracy =  0.894207650273224 with std =  0.04366301452293619
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 9
Accuracy =  0.8908743169398908 with std =  0.050676571543476726
kernel = Poly , Gamma = 0.0001, C = 1 , degree = 10
Accuracy =  0.8841530054644808 with std =  0.04056949251609546
```

نتایج نشان میدهد که بیشترین دقت برای هسته ی poly حدودا با gamma = 0.0001 و C = 1 و degree = 2 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.9404

آزمایش سوم : پیدا کردن جواب بهینه برای هسته ی sigmoid

```
kernel = Sigmoid, Gamma = 1e-07
Accuracy =  0.9204371584699456 with std =  0.02881932192148546
kernel = Sigmoid, Gamma = 1e-06
Accuracy =  0.516775956284153 with std =  0.0897688978158893
kernel = Sigmoid, Gamma = 1e-05
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid, Gamma = 0.0001
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid, Gamma = 0.001
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid, Gamma = 0.01
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid, Gamma = 0.1
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid, Gamma = 1
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid, Gamma = 10
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
```

```
kernel = Sigmoid , Gamma = 1e-07, C = 1e-06
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid , Gamma = 1e-07, C = 1e-06
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid , Gamma = 1e-07, C = 1e-05
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid , Gamma = 1e-07, C = 0.0001
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid , Gamma = 1e-07, C = 0.001
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid , Gamma = 1e-07, C = 0.01
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid , Gamma = 1e-07, C = 0.1
Accuracy =  0.2251912568306011 with std =  0.008599224910159686
kernel = Sigmoid , Gamma = 1e-07, C = 1
Accuracy =  0.46038251366120225 with std =  0.030172149827617904
kernel = Sigmoid , Gamma = 1e-07, C = 10
Accuracy =  0.8674863387978142 with std =  0.028134382735238047
kernel = Sigmoid , Gamma = 1e-07, C = 100
Accuracy =  0.9337158469945355 with std =  0.035041598754706575
kernel = Sigmoid , Gamma = 1e-07, C = 10000
Accuracy =  0.9038797814207651 with std =  0.034160829704850564
kernel = Sigmoid , Gamma = 1e-07, C = 10000
Accuracy =  0.9038797814207651 with std =  0.034160829704850564
kernel = Sigmoid , Gamma = 1e-07, C = 100000
Accuracy =  0.893879781420765 with std =  0.025388145157515987
kernel = Sigmoid , Gamma = 1e-07, C = 1000000
Accuracy =  0.893879781420765 with std =  0.025388145157515987
```

نتایج نشان میدهد که بیشترین دقت برای هسته ی sigmoid حدودا با gamma = 1e-7 و C = 100 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.9337

آزمایش چهارم : پیدا کردن جواب بهینه برای هسته ی linear

```
kernel = Linear , C = 1e-06
Accuracy =  0.8872131147540984 with std =  0.032828025191271135
kernel = Linear , C = 1e-06
Accuracy =  0.8872131147540984 with std =  0.032828025191271135
kernel = Linear , C = 1e-05
Accuracy =  0.9270491803278688 with std =  0.03901157154389074
kernel = Linear , C = 0.0001
Accuracy =  0.920327868852459 with std =  0.0373846379667095
kernel = Linear , C = 0.001
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
kernel = Linear , C = 0.01
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
kernel = Linear , C = 0.1
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
kernel = Linear , C = 1
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
kernel = Linear , C = 10
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
kernel = Linear , C = 100
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
kernel = Linear , C = 10000
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
kernel = Linear , C = 10000
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
kernel = Linear , C = 100000
Accuracy =  0.9070491803278689 with std =  0.034696828780723755
```

نتایج نشان میدهد که بیشترین دقت برای هسته ی linear حدودا با gamma = 1e-7 و C = 1e-5 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.9270

نتایج نشان میدهد که بیشترین دقت در طبقه بندی برای هسته ی poly و با حدودا gamma = 0.0001 و C = 1 و degree = 2 ایجاد میشود و این دقت تقریبا برابر است با : accuracy = 0.9404

همانطور که مبینیم دقت طبقه بندی بسیار قابل قبول است. این درحالیست که تعداد زیادی از عکس ها را با چشم خود هم نمیتوانیم تشخیص دهیم.