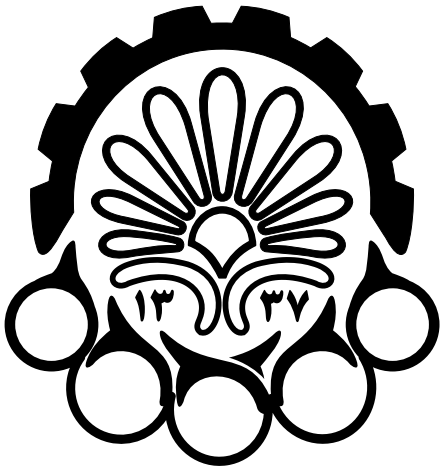


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

تمر ین اول درس شبکه های عصبی

دکتر صفابخش

غلامرضا دار ۴۰۰۱۳۱۰۱۸

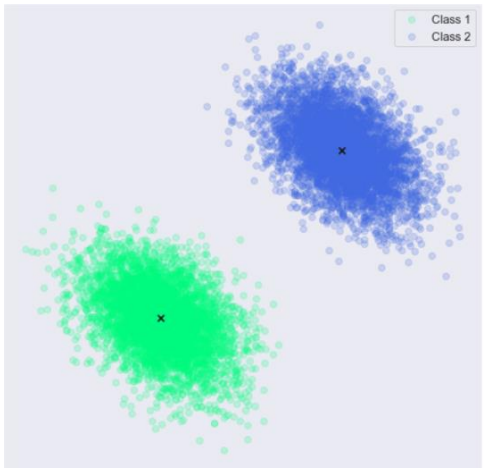
زمستان ۱۴۰۰

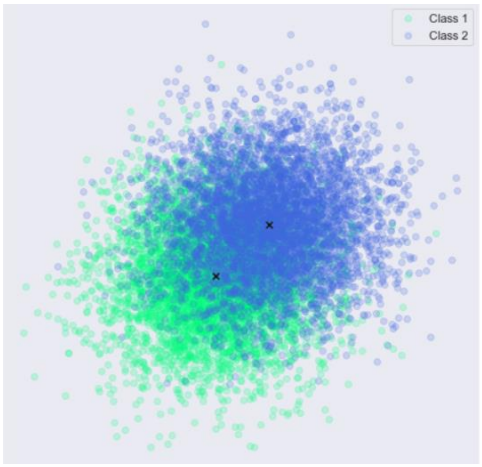
فهرست مطالب

سوال ۱)

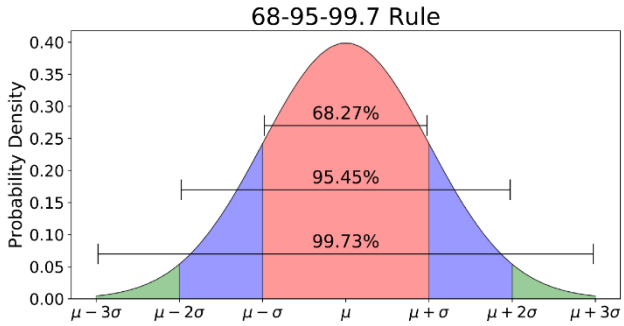
در بخش اول این سوال نیاز است داده ی مورد نیاز برای سوال را با کمک **دو** توزیع گاوسی دو متغیره تولید کنیم. برای توزیع ها قرار است از میانگین **متفاوت** و ماتریس کواریانس **یکسان** استفاده کنیم.

تحت چه شرایطی داده ها جدایی پذیر خطی خواهند بود؟ برای اینکه داده ها به صورت خطی جدایی پذیر باشند لازم است میانگین توزیع ها به اندازه کافی از هم فاصله داشته باشد به طوری که داده های دو کلاس همپوشانی نداشته باشند. به تصویر زیر دقت کنید، توزیع های سمت چپ به صورت خطی جدایی پذیر هستند و توزیع های سمت راست به صورت خطی جدایی پذیر نیستند.





اگر بخواهیم کمی تحلیلی تر به قضیه نگاه کنیم، با توجه به اینکه میدانیم حدود ۹۹/۷ درصد از داده های هر توزیع در فاصله ۳ برابر انحراف از معیار آن توزیع قرار دارد، میتوان گفت فاصله میانگین توزیع ها باید حداقل ۶ برابر انحراف از معیار در راستای خط متصل کننده میانگین ها باشد.

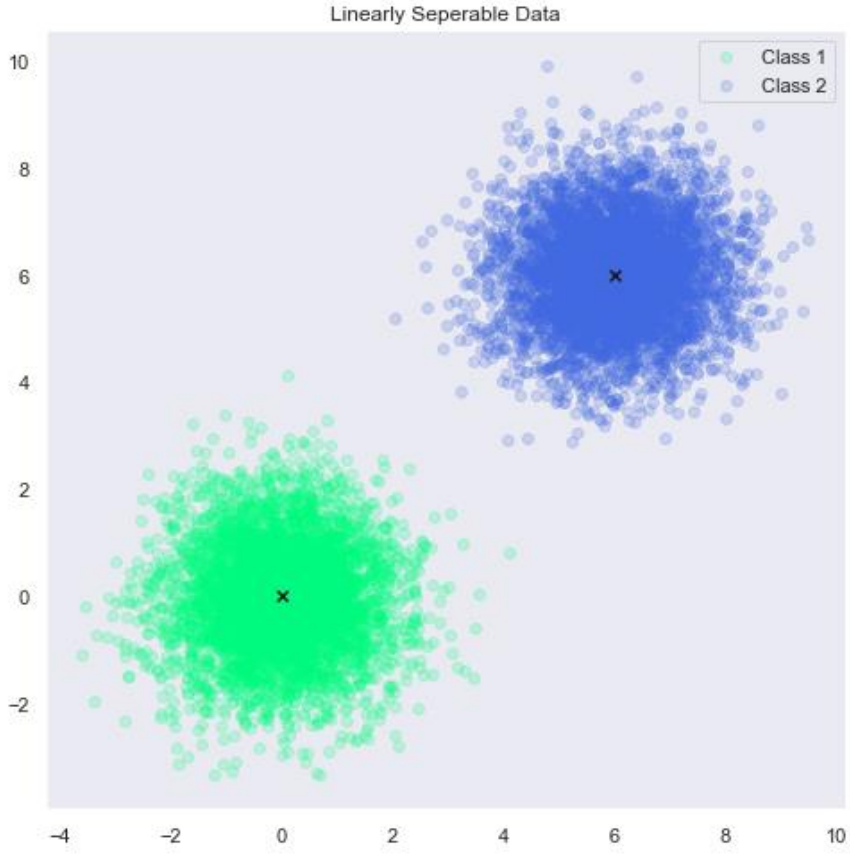


نکته مهم: برای بررسی جداپذیری خطی داده های توزیع های نرمال حتما باید داده های تولید شده را داشته باشیم و قبل از تولید شدن داده ها نمیتوان نظر قطعی داد چون طبق تعریف، داده های تولید شده توسط توزیع های نرمال میتوانند بی نهایت از مرکز فاصله داشته باشند(هرچند با احتمال بسیار کم) و تنها کاری که ما میتوانیم بکنیم این است که احتمال خطی جدایی پذیر بودن توزیع ها را فزایش دهیم.

در ادامه نمودار توزیع های دو کلاس جدایی پذیر خطی را میتوانید مشاهده کنید.

$$\mu_1 = (0,0), \qquad \mu_2 = (6,6)$$

$$cov_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad cov_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



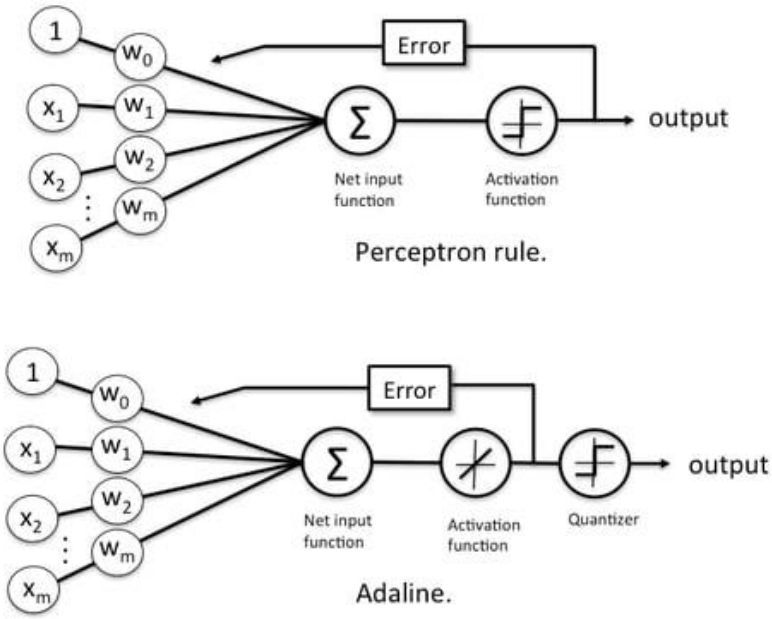
سوال ۲)

هم پرسپترون و هم آدالاین دسته بند های باینری هستند که با داده های ورودی چند بعدی کار میکنند. هر دو به صورت خطی داده ها را جدا میکنند. اگر به تصاویر زیر دقت کنید میبینید که این دو نورون بسیار به هم شبیه هستند. هر دو ورودی را به صورت آرایه چند بعدی دریافت میکنند، بع ازای هر ورودی یک وزن(ضریب) در نظر میگیرند، هر داده ورودی را در وزن متناظرش ضرب میکنند و با هم جمع میکنند. این خروجی از یک تابع فعال ساز عبور داده میشود و سپس با عمل Threshold گیری یک برچسب به داده زده میشود.

تفاوت نورون پرسپترون و آدالاین در نحوه آپدیت کردن وزن هاست. پرسپترون برای آپدیت کردن وزن ها از خروجی Threshold گرفته شده استفاده میکند و در واقع برچسب تخمین زده شده و برچسب اصلی را با هم مقایسه میکند. در صورت وجود تفاوت بین این دو برچسب، وزن ها را در جهتی که این تصمیم گیری را بهبود دهد تنظیم میکند.

اما آدالاین برای آپدیت کردن وزن ها به صورت مستقیم از خروجی ضرب داخلی X,W استفاده میکند و از میزان فاصله ی y, y_predicted ، به عنوان معیاری برای آپدیت کردن وزن ها استفاده میکند.

همچنین در حالتی که از تابع فعال سازی برای این دو نورون استفاده میکنیم، نورون پرسپترون از خروجی تابع Threshold برای آپدیت وزن ها استفاده میکند اما نورون آدالاین از خروجی تابع فعال سازی برای این کار استفاده میکند که این به این معنی است که توابع فعال سازی مختلف بر روی آدالاین تاثیر واضح تری خواهند گذاشت.

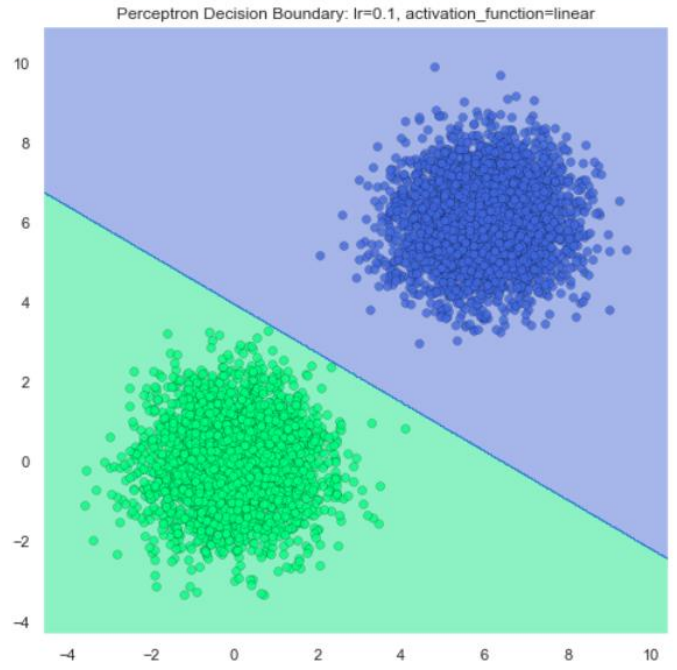


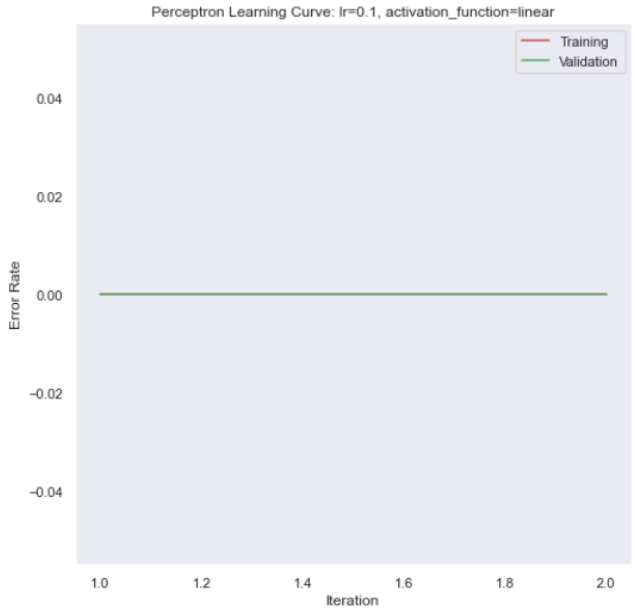
منبع تصویر : https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html

سوال ۳)

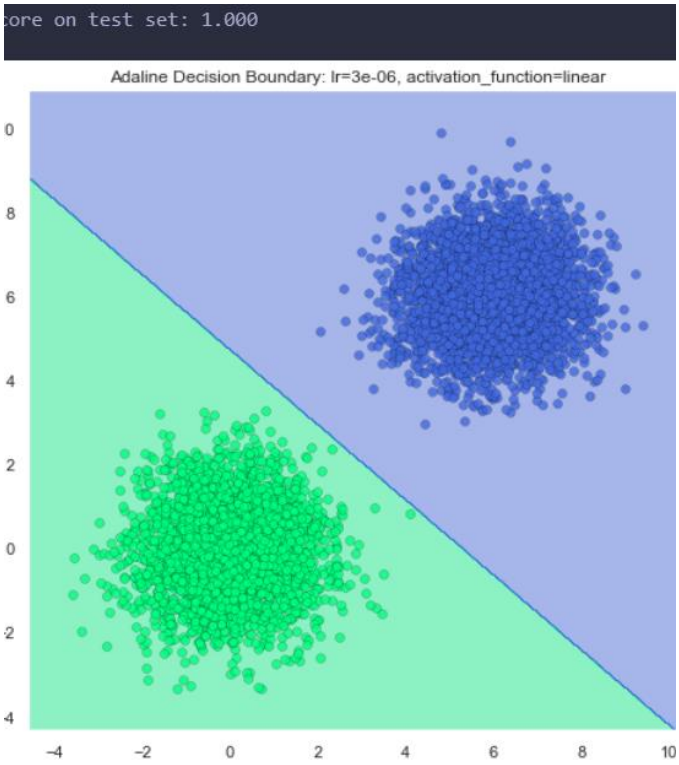
در این بخش میخواهیم داده ای که تولید کردیم را با کمک نورون های Perceptron و Adaline جداسازی کنیم. همانطور که در تصاویر زیر مشاهده میکنید پرسپترون پس از گذشت ۱ iteration به تابع جداساز بهینه میرسد. البته این تعداد بالای داده آموزش(۷۰۰۰) در این امر بی تاثیر نیست. طی آزمایش هایی که انجام شد اگر تعداد داده های آموزش را کم کنیم، تعداد iteration های مورد نیاز برای Convergence افزایش می یابد.

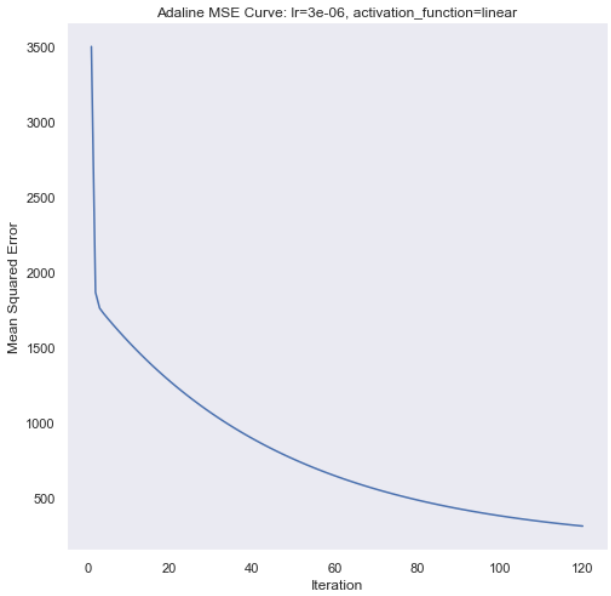
```
Converged Sooner than max_iter
Score on test set: 1.000
```

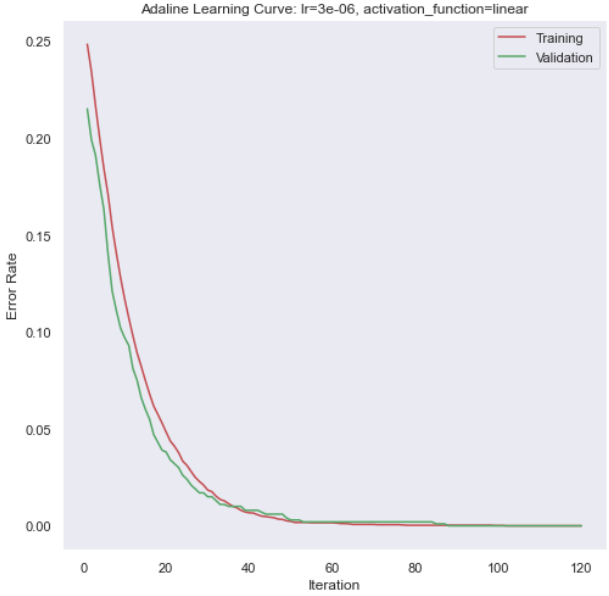




در ادامه داده ها را به کمک نورون Adaline جداسازی میکنیم. در هر دو حالت به دقت ۱۰۰ درصد بر روی داده تست میرسیم اما لازم به ذکر است که در مورد Adaline به learning rate بسیار کمتر و هم چنین تعداد iteration بالاتری نیاز داشتیم.
همچنین مقادیر متناظر با learning_rate, activation_function بر روی تصاویر نوشته شده است.

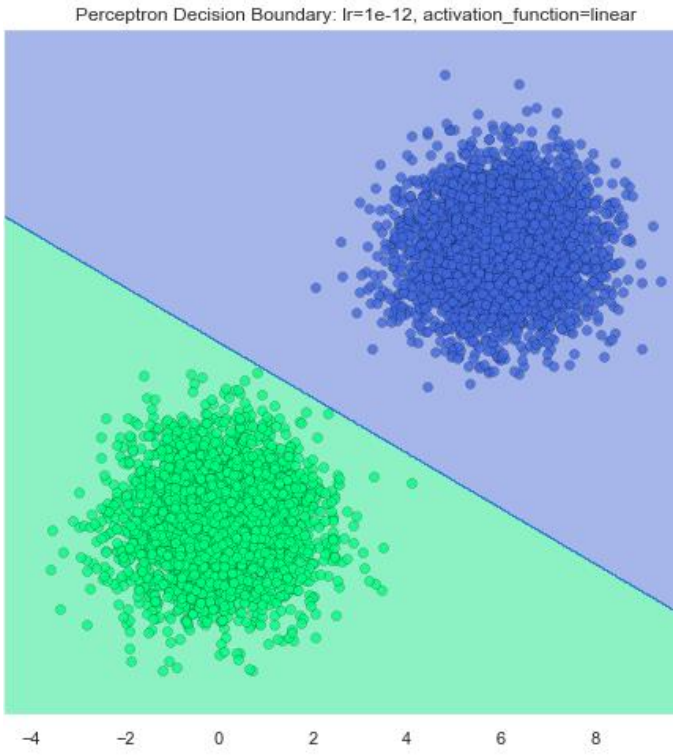


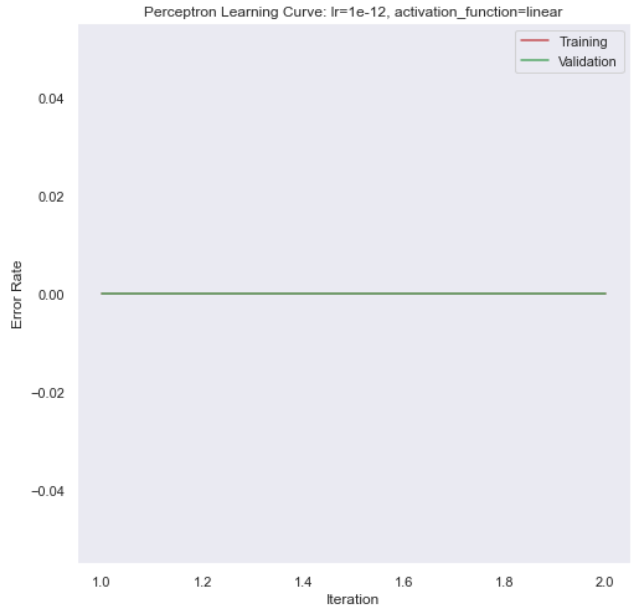


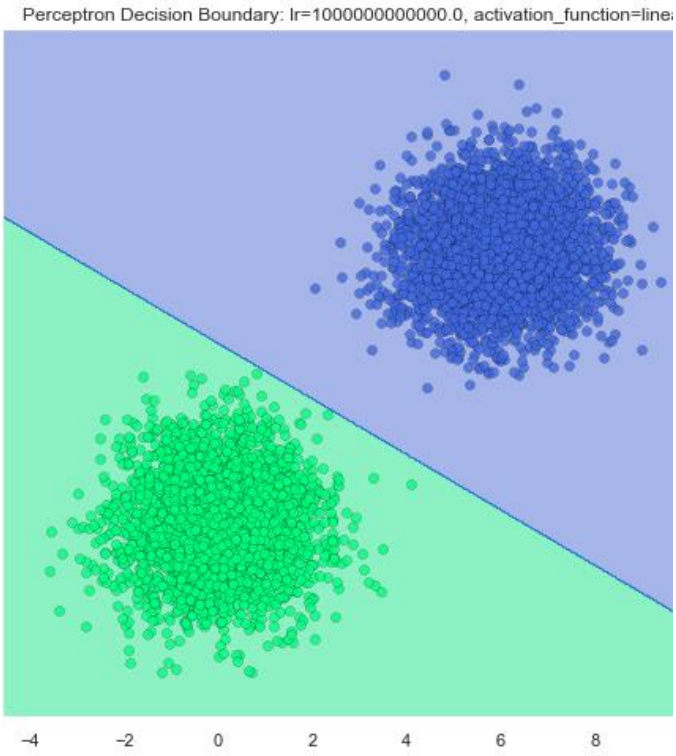


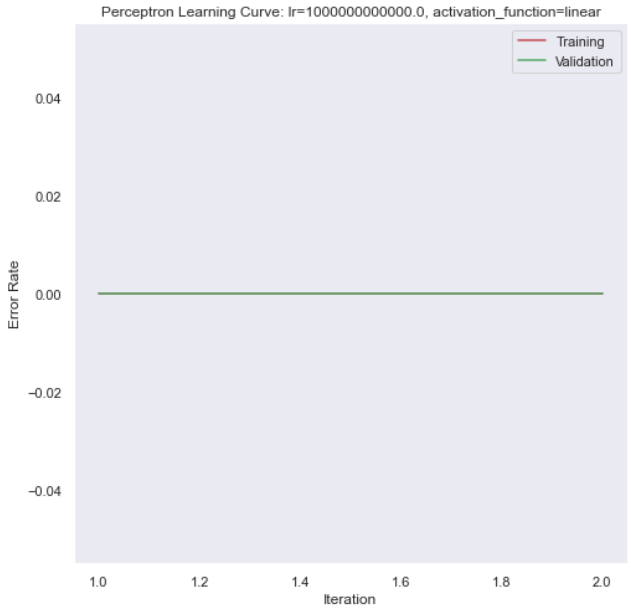
در ادامه تصمیم داریم این دو نوع نورون را با هم مقایسه کنیم. برای این منظور میخواهیم رفتار آنها در مقابل کاهش/افزایش نرخ یادگیری و تغییر تابع فعال سازی را بررسی کنیم.

ابتدا با مقایسه دو نوع نورون در برابر تغییر learning_rate کار را شروع میکنیم. همانطور که ذکر شد در این مثال، پرسپترون طی ۱ iteration به همگرایی میرسد و به همین دلیل تغییر learning_rate هیچ گونه تاثیری ندارد. در ادامه مقادیر 10^{12} تا 10^{-12} را برای learning_rate آزمایش میکنیم و تغییری مشاهده نمیکنیم.






Perceptron Decision Boundary: lr=1000000000000.0, activation_function=linea
−4
−2
0
2
4
6
8

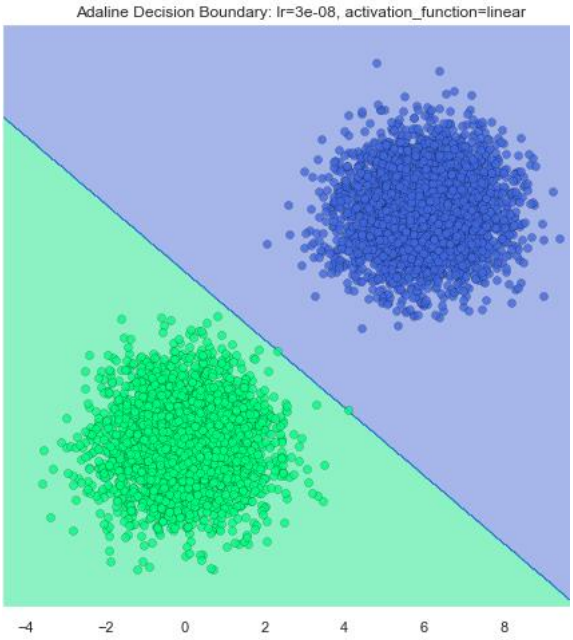

Perceptron Learning Curve: lr=1000000000000.0, activation_function=linear
Training
Validation
0.04
0.02
0.00
−0.02
−0.04
Error Rate
1.0
1.2
1.4
1.6
1.8
2.0
Iteration

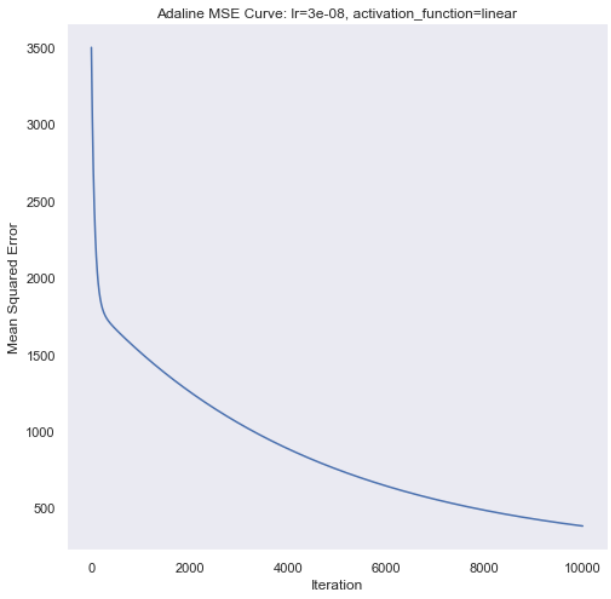
اما در مورد Adaline این داستان کمی متفاوت است. مقادیر زیاد برای Learning_rate باعث میشوند در پروسه آموزش دچار خطای زیر شویم.

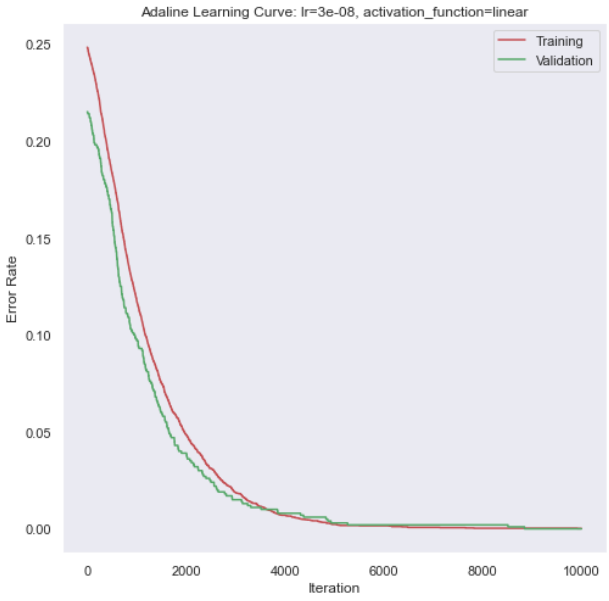
```
RuntimeWarning: overflow encountered in power
  mse = (np.power(error, 2)).sum() / 2.0
```

بنابراین از مقادیر کم تر برای learning_rate استفاده میکنیم.

مقدار حدود $3 * 10^{6}$ را در قسمت اول بخش ۳ دیدیم که بسیار مناسب بود و دقت ۱۰۰ درصد بر روی داده تست داد. مقادیر کوچک تر مانند $3 * 10^{8}$ مناسب هر چند در نهایت به تابع جداساز بهینه میرسند اما به مراتب تعداد iteration های بیشتری میطلبند. به عنوان مثال در مثال زیر پس از حدود ۹۰۰۰ iteration به تابع جداساز بهینه میرسیم.



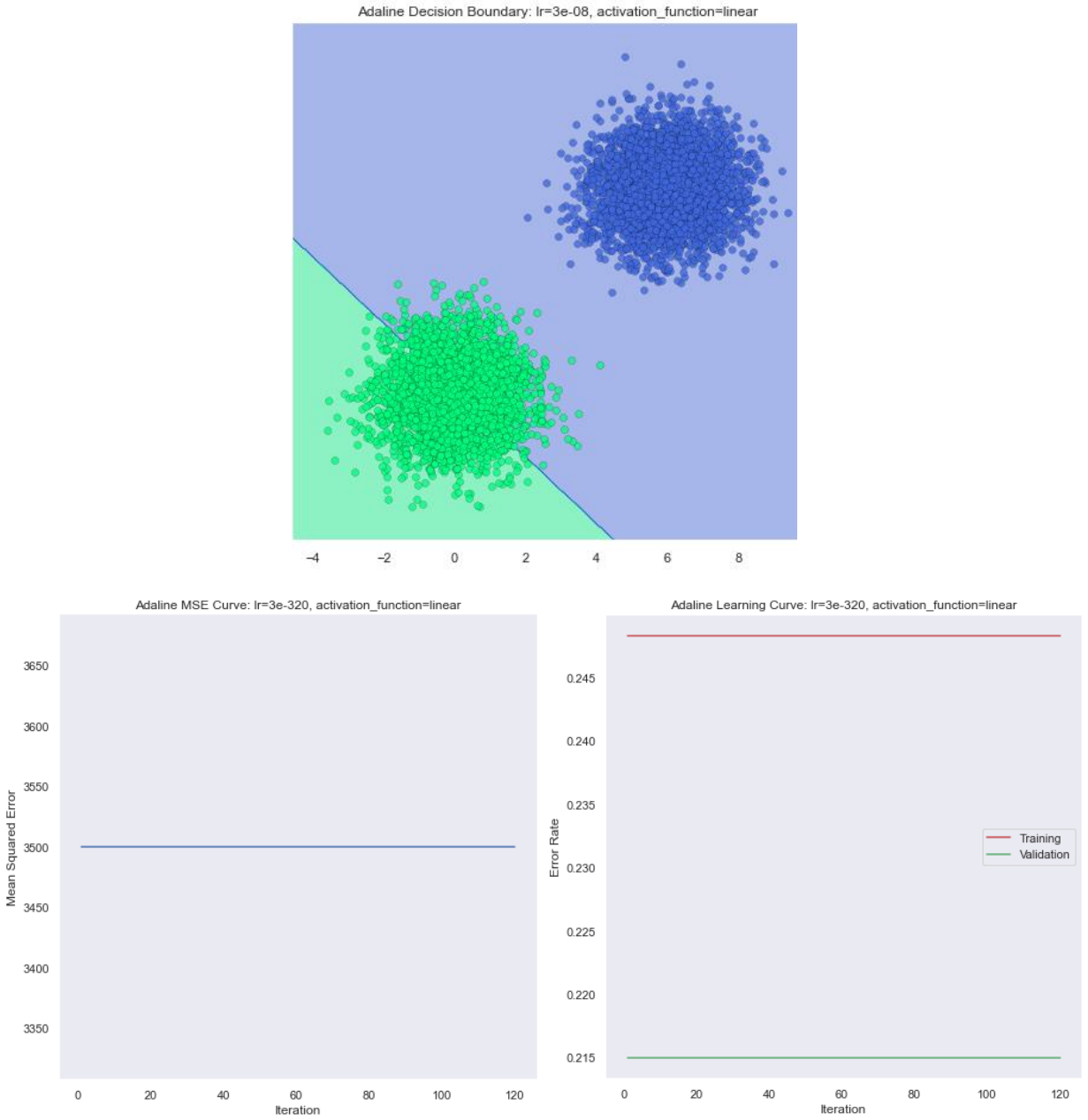




دلیل این اتفاق این است که مقادیر کوچک تر برای learning_rate یعنی وزن ها خیلی آرام تر به سمت وزن بهینه حرکت میکنند و در هر مرحله خیلی فاصله کمی را برای رسیدن به وزن بهینه طی میکنند. این اتفاق منطقی است زیرا مدل آدالاین از حاصل ضرب learning_rate در تفاضل $y, \hat{y}$ برای آپدیت کردن وزن ها استفاده میکند و وقتی learning_rate کم باشد کل عبارت مورد نیاز برای آپدیت کردن وزن ها بسیار کوچک میشود.

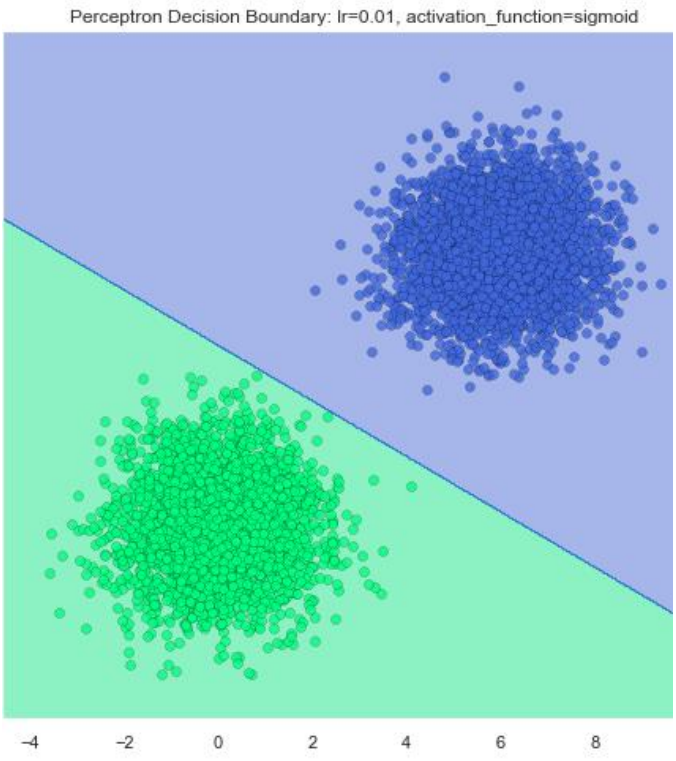
هم چنین با مقادیر learning_rate کوچک تر از $3 * 10^{-8}$ هم آنقدر تعداد iteration های مورد نیاز زیاد میشد که قابل انجام نبودند. لازم به ذکر است که تمام مقادیر اطراف $3 * 10^{-6}$ تقریبا با همان تعداد iteration به همگرایی میرسیدند.

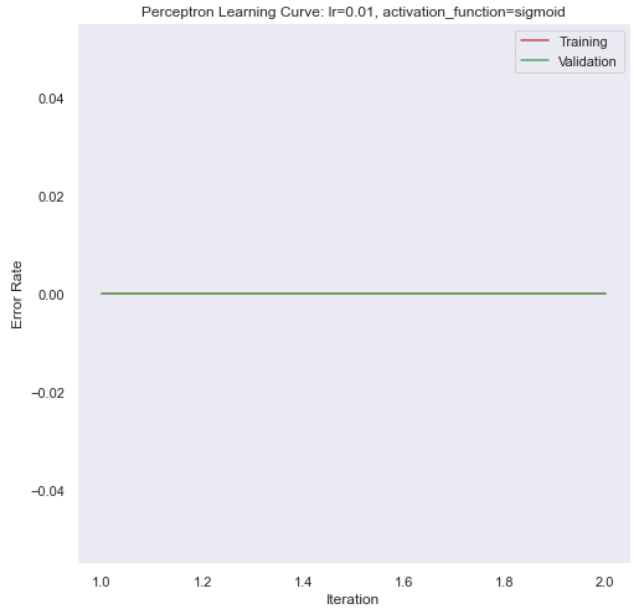
Score on test set: 0.743



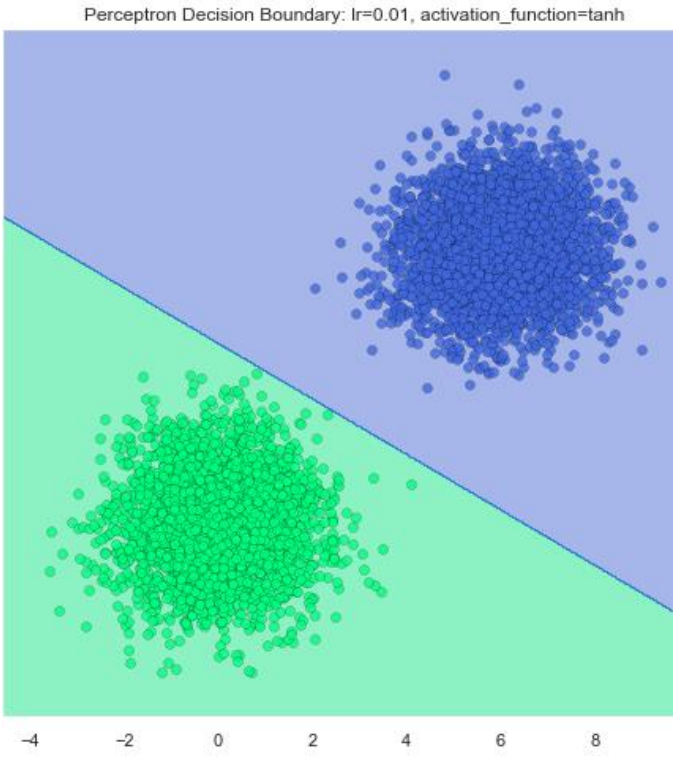
حالا نوبت به آزمایش کردن توابع مختلف فعال سازی میرسد. در هر دو نوع نورون تابع فعالسازی به خروجی *net_input* اعمال میشود و خروجی آن به تابع *Threshold* داده میشود و در نهایت برچسب ها تعیین میشوند. در این سوال از توابع فعال سازی *Linear, Sigmoid, Tanh* استفاده کردیم. خروجی نورون ها برای تابع فعال سازی *Linear* را در ابتدای این سوال مشاهده کردید.

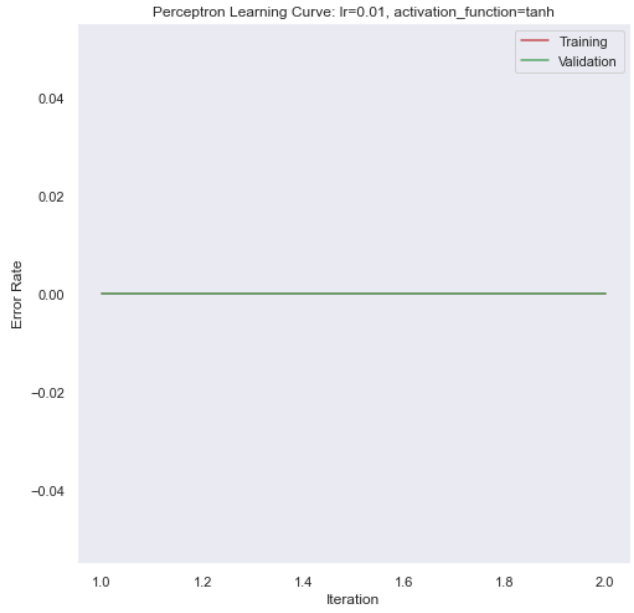
پرسپترون با فعال سازی *Sigmoid*





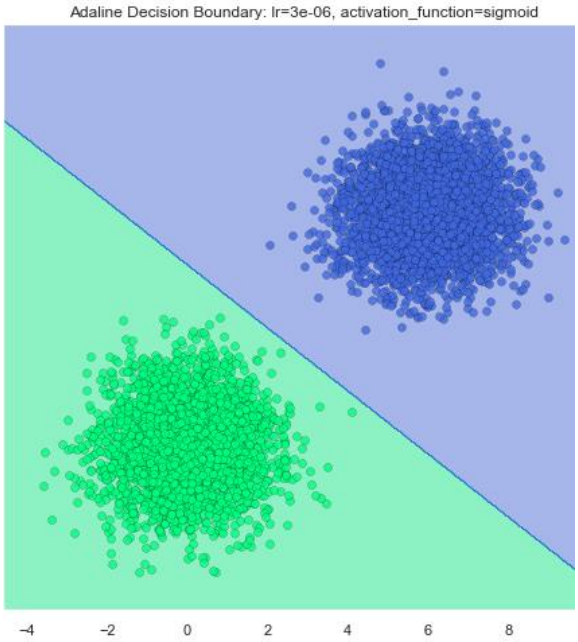
پرسپترون با فعال سازی *Tanh*

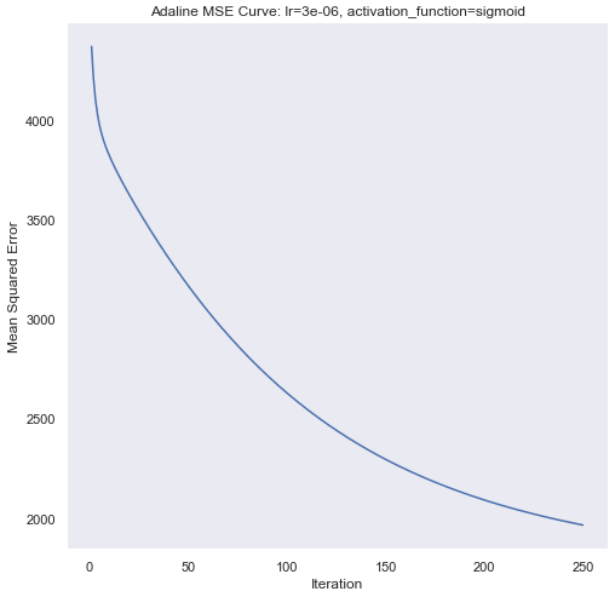


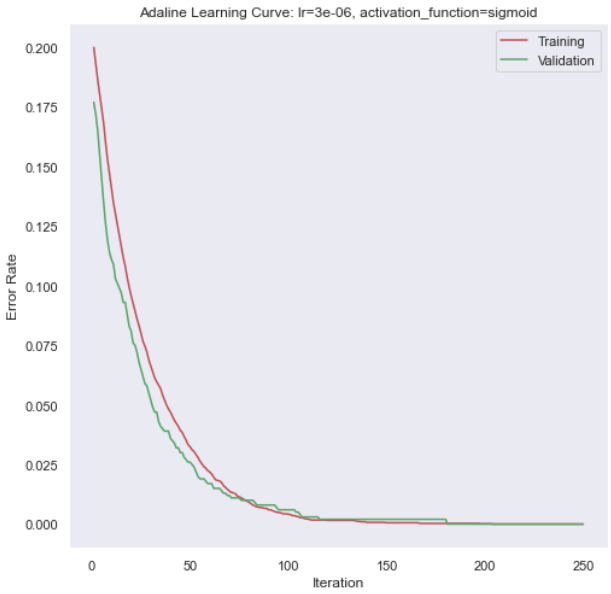


همانطورکه انتظار میرفت تغییری در نتایج مشاهده نمیکنیم. چون پرسپترون برای آموزش از برچسب داده ها استفاده میکند و توابع فعال سازی تغییری در برچسب داده ها ایجاد نمیکنند.

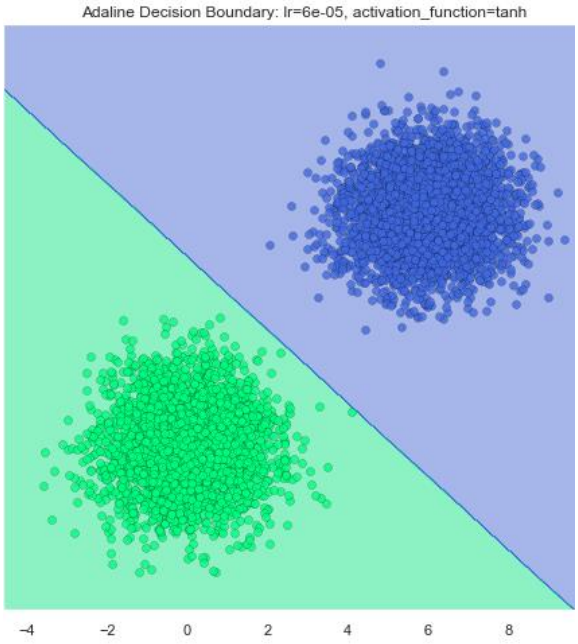
آدالاین با فعال سازی *Sigmoid*

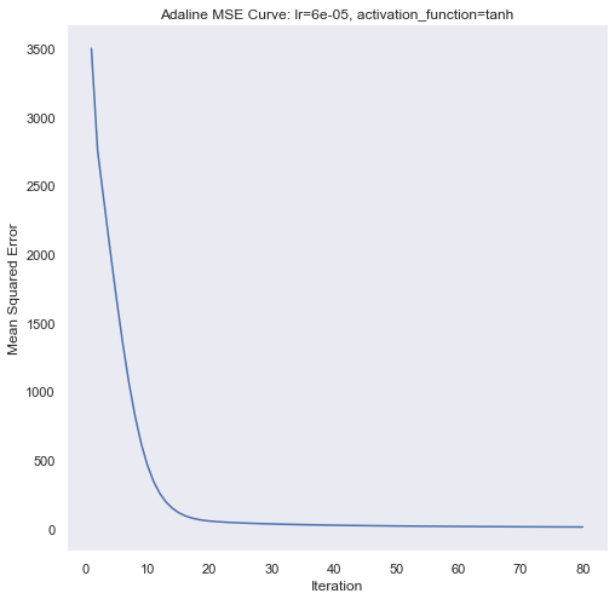


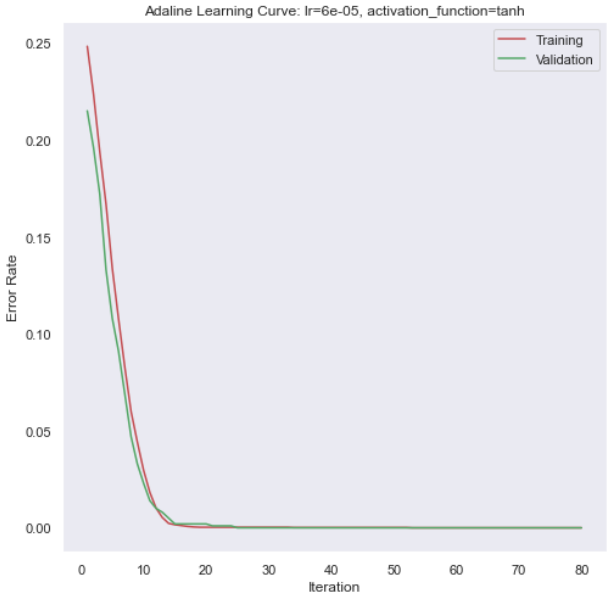




آدالاین با فعال سازی *Tanh*







با توجه به نمودار های بدست آمده مشاهده میشود که توابع فعال سازی در آدالاین موثر تر هستند، زیرا آدالاین از خروجی Treshold گرفته نشده این مقادیر برای آپدیت کردن وزن ها استفاده میکند که این باعث میشود در نحوه آموزش آن تغییر ایجاد کند. اما در مورد تغییرات این مورد مشاهده شد که با استفاده از توابع فعال سازی سیگموید و تانژانت هایپربولیک سرعت همگرایی بالا رفت و همچنین مقادیر بالاتری برای learning_rate را توانستیم استفاده کنیم.

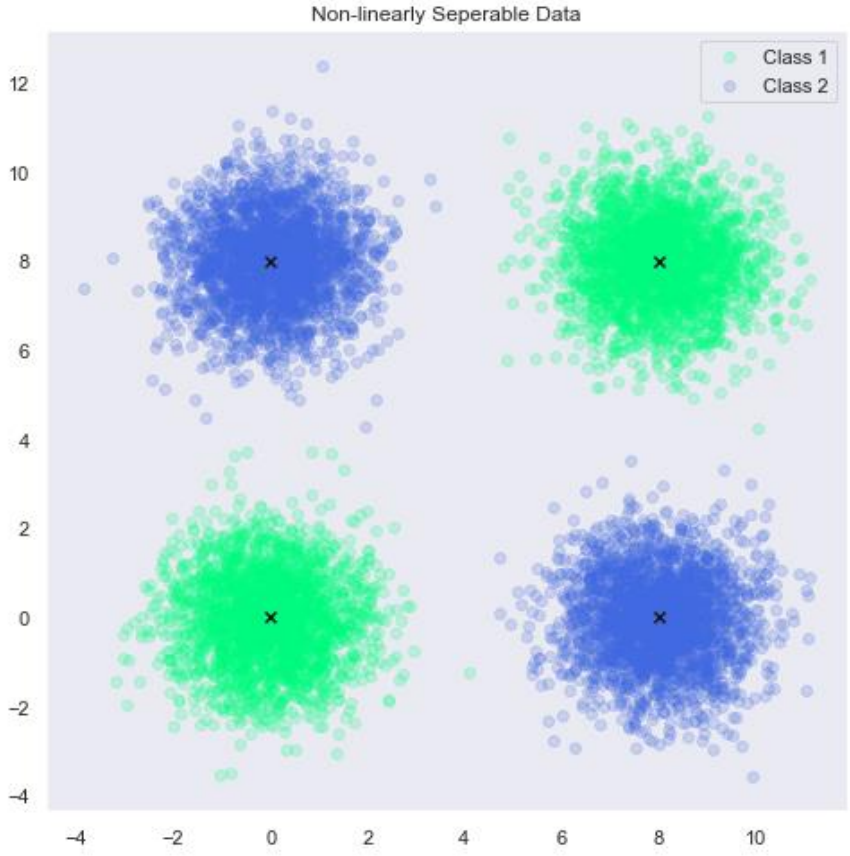
به طور خلاصه برای داده های **جدایی پذیر خطی** بهتر است از **پرسپترون** استفاده کنیم چون در تعداد iteration کمتری به تابع جدا کننده بهینه میرسیم.

سوال ۴)

در این بخش میخواهیم یک دیتاست جدایی ناپذیر خطی با کمک توزیع های گاوسی تولید کنیم. برای این کار از ۲ توزیع نرمال مختلف برای داده های هر کلاس استفاده میکنیم. به این شکل که توزیع های یک کلاس را به صورت مورب رو به روی هم قرار میدهیم. برای توزیع ها از کواریانس همانی استفاده شده و به عنوان مرکز توزیع ها نقاط زیر انتخاب شده اند.

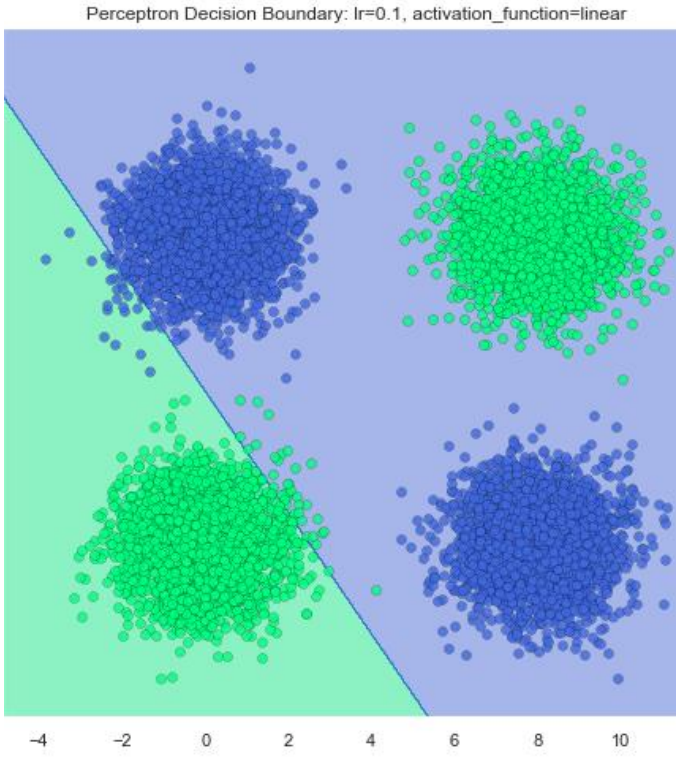
$$\mu_1 = (0, 8), \qquad \mu_2 = (8, 8)$$

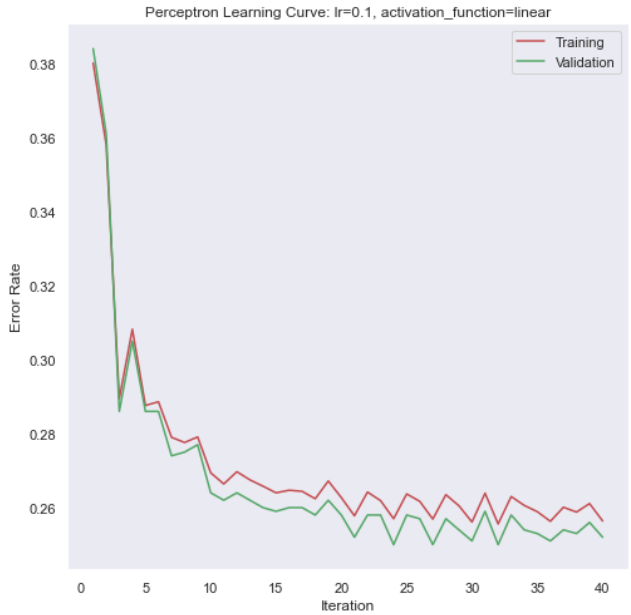
$$\mu_3 = (0, 0), \qquad \mu_4 = (8, 0)$$



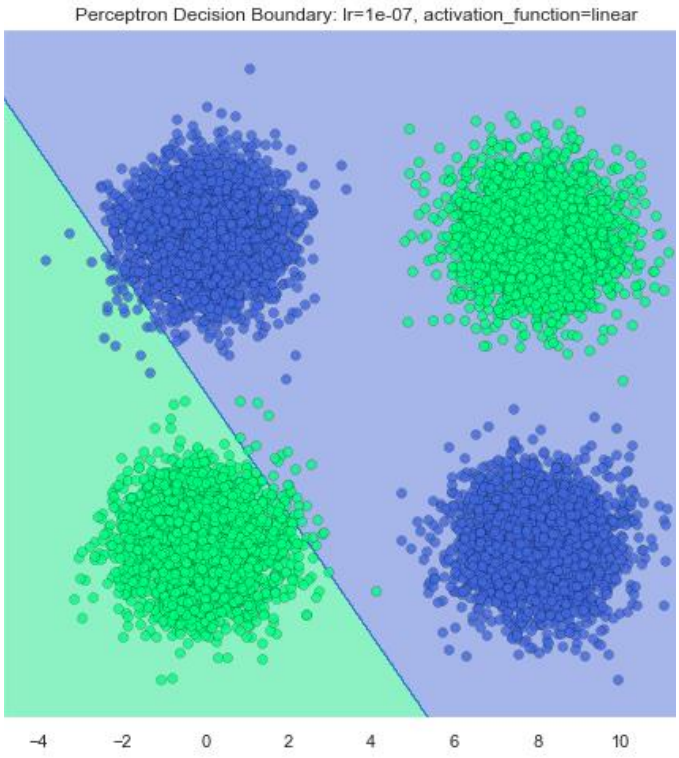
همانطورکه مشاهده میکنید این دو کلاس به صورت خطی قابل جدایی نیستند و باید از روش های غیرخطی برای جداسازی آنها استفاده کنیم.

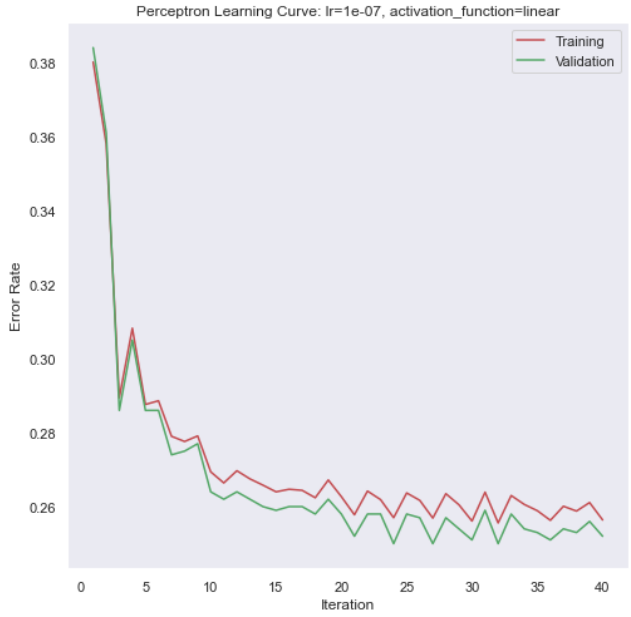
ابتدا به با استفاده از پرسپترون سعی میکنیم داده ها را جداکنیم. توقع نداریم پرسپترون خطی بتواند این داده ها را با دقت بالاتر از ۷۵ درصد جداسازی کند. در این بخش مقادیر مختلف برای learning rate را آزمایش میکنیم.



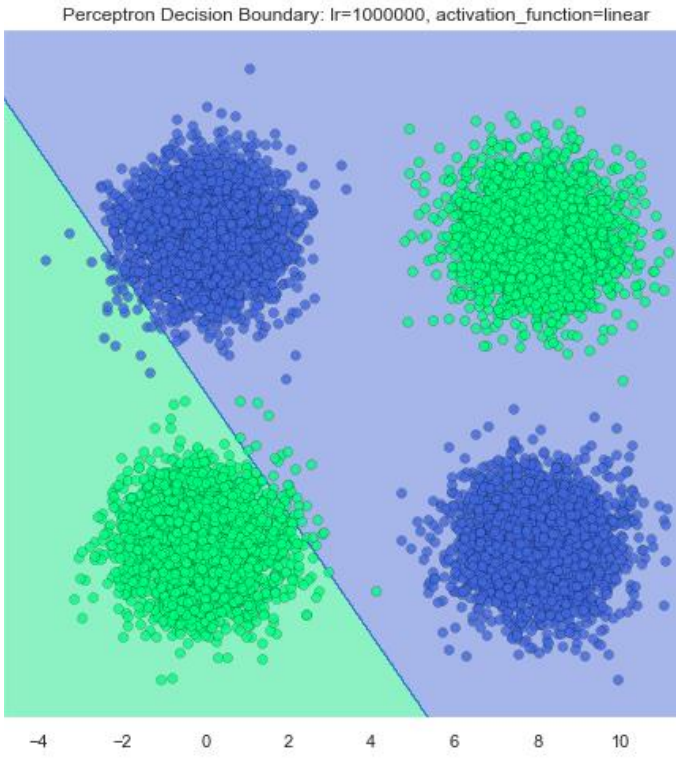


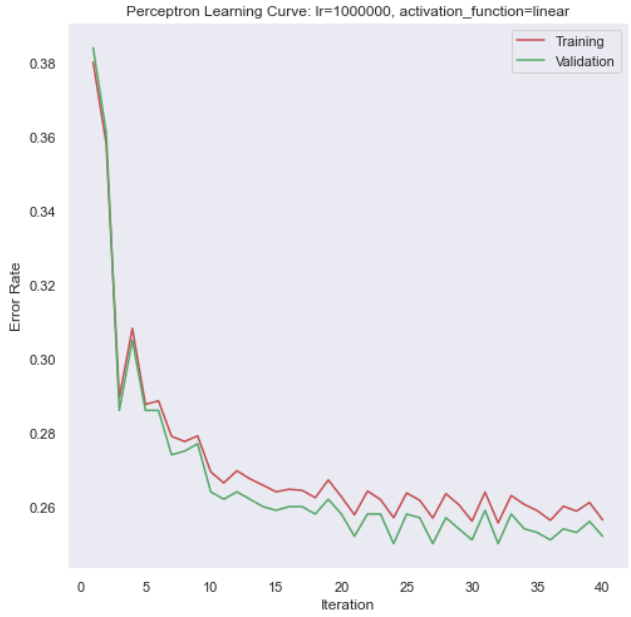
Score on test set: 0.74





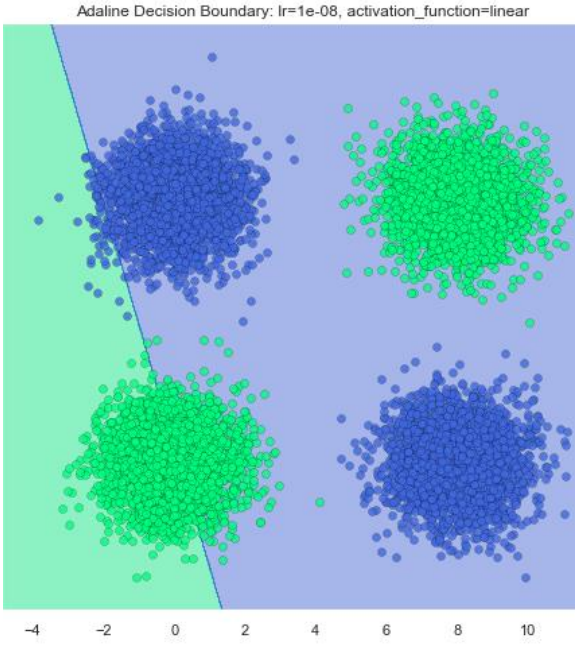
Score on test set: 0.74

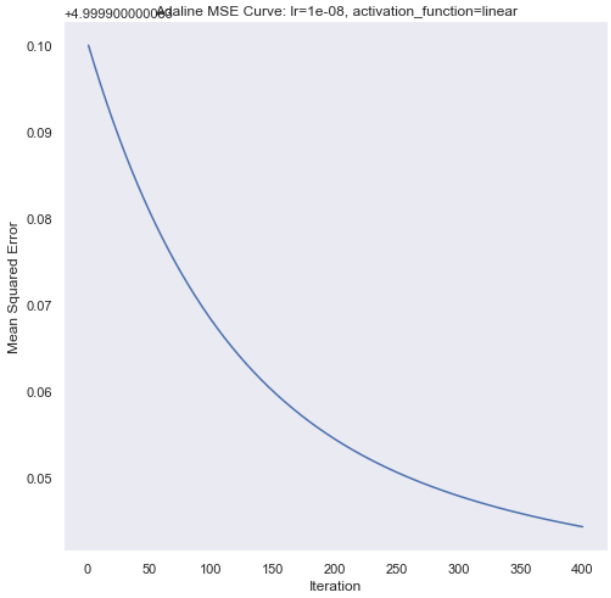




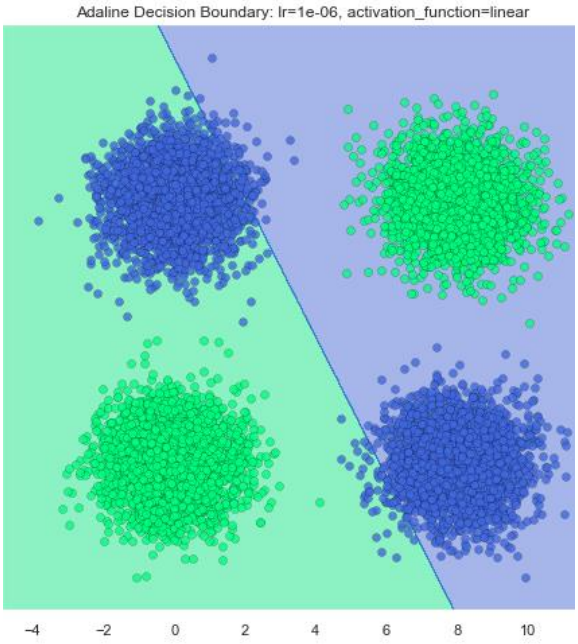
همانطورکه دیدم در مورد پرسپترون، learning rate تاثیر زیادی نداشت و با رنج وسیعی از نرخ های آزمایش به نتیجه مشابهی رسیدیم. در ادامه نورون Adaline را بررسی میکنیم.

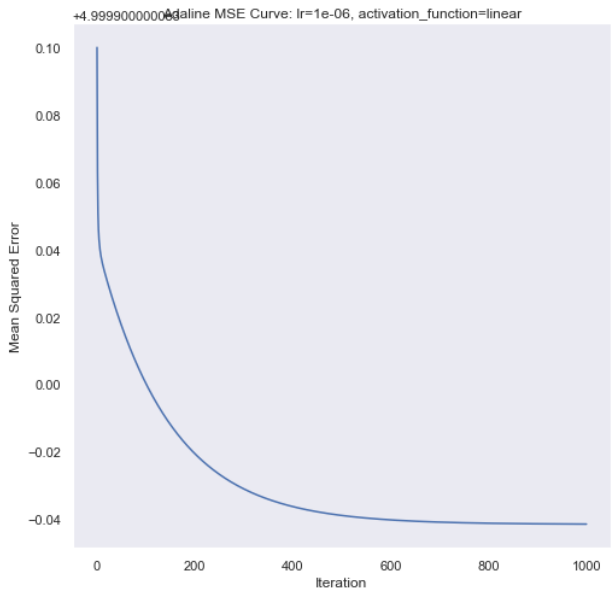
Score on test set: 0.64

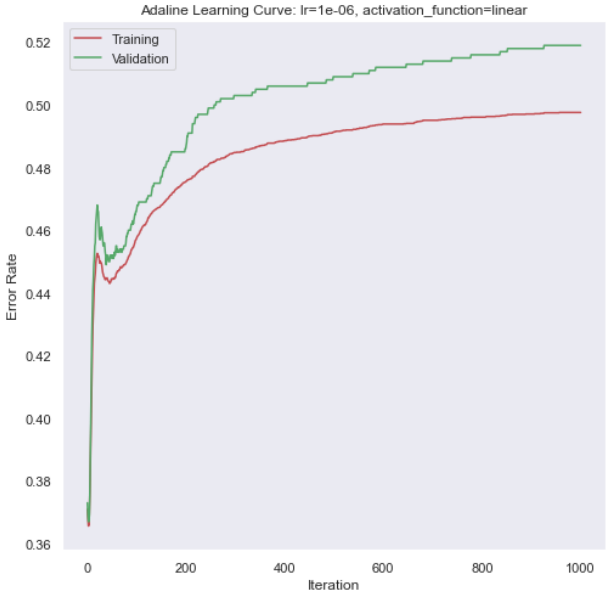




Score on test set: 0.52

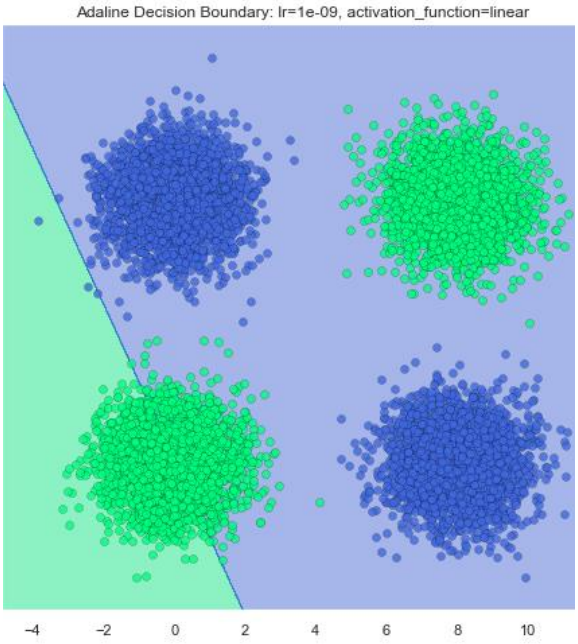


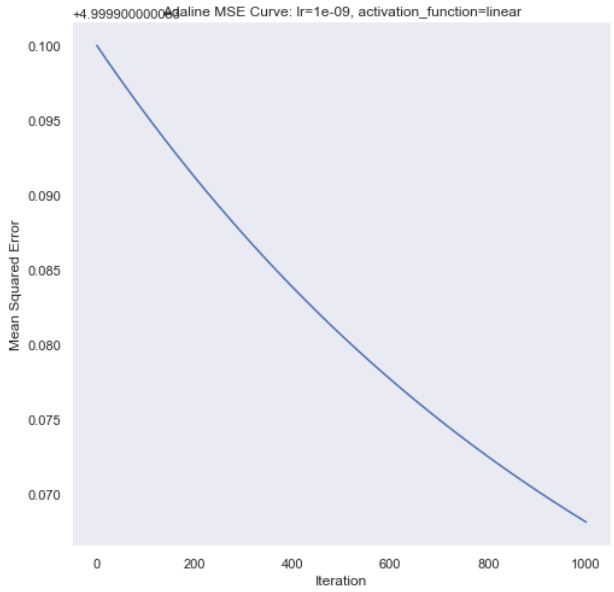


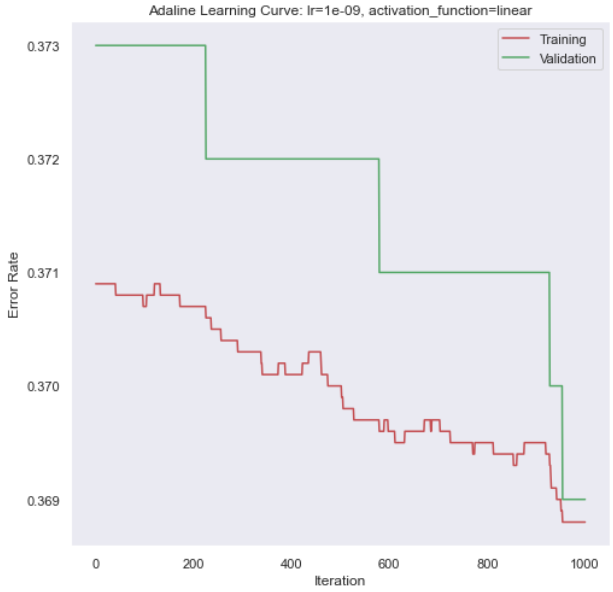


همانطورکه مشاهده میشود مقادیر بسیار بالای لرنینگ ریت باعث میشوند در طی ایتریشن های زیاد شبکه بتواند از یک سری مینیمم های محلی خارج شود و به نتایج حتی بدتری برسد. بنابراین در این مثال افزایش لرنینگ ریت انتخاب خوبی نیست

Score on test set: 0.63

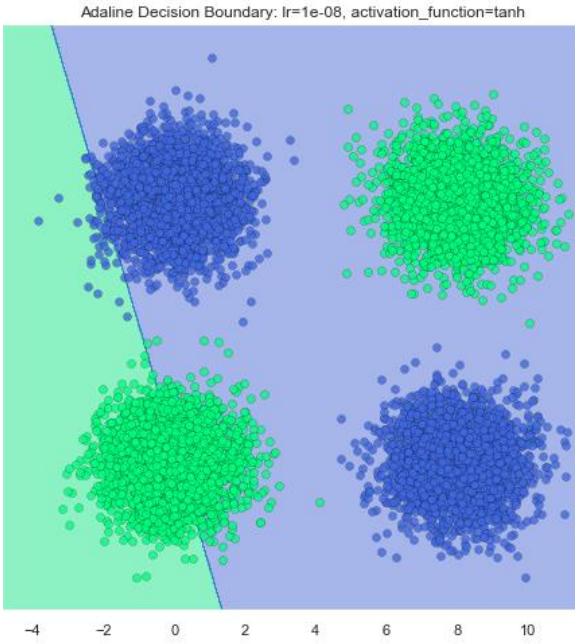


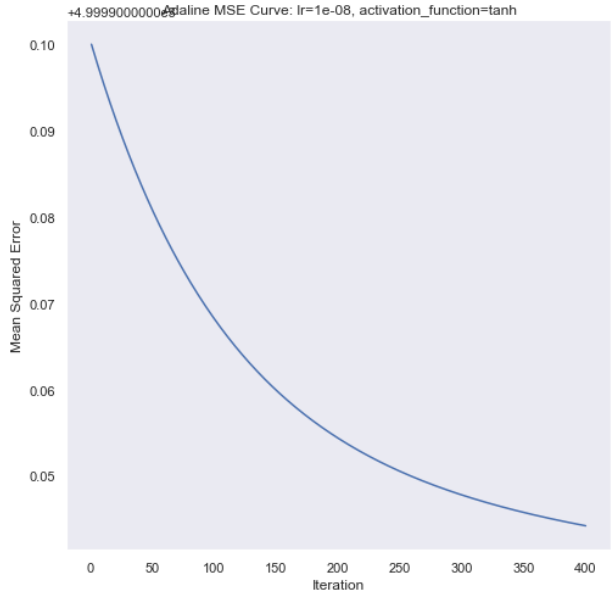


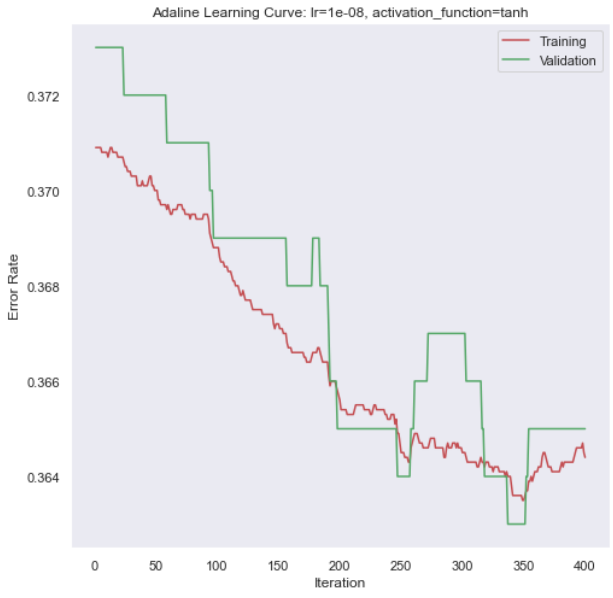


مقادیر کمتر learning rate باعث میشوند روند آموزش کند تر پیش برود و همانطورکه مشاهده میکنید برای رسیدن به نتیجه اولیه به تعداد ایتریشن بیشتری نیاز داریم.

Score on test set: 0.64

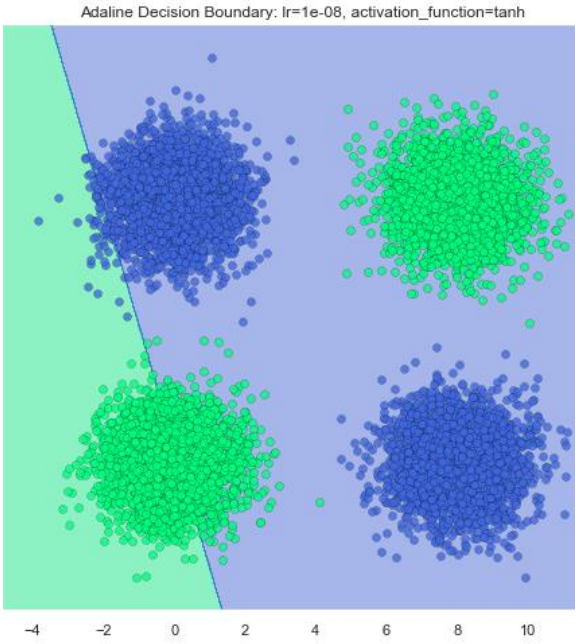


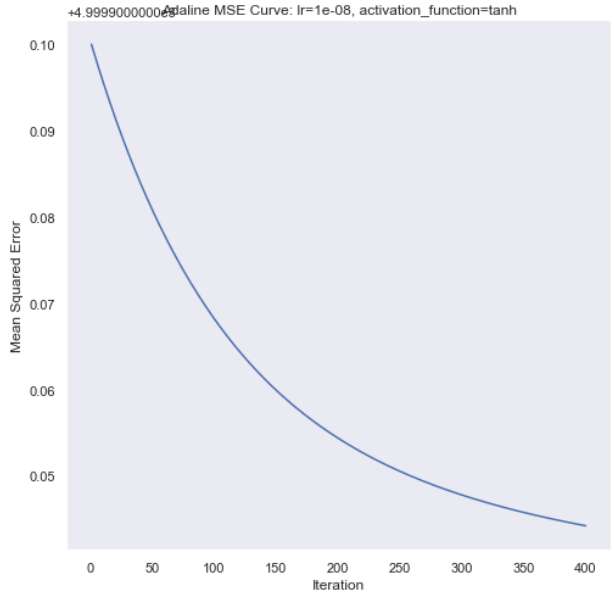




تابع فعال سازی Tanh تاثیری در یادگیری Adaline ندارد.

Score on test set: 0.64

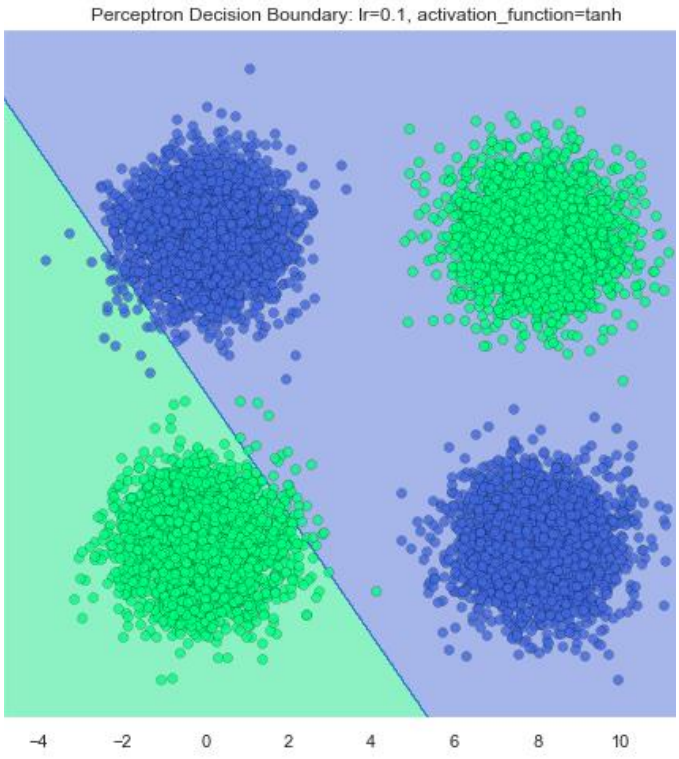


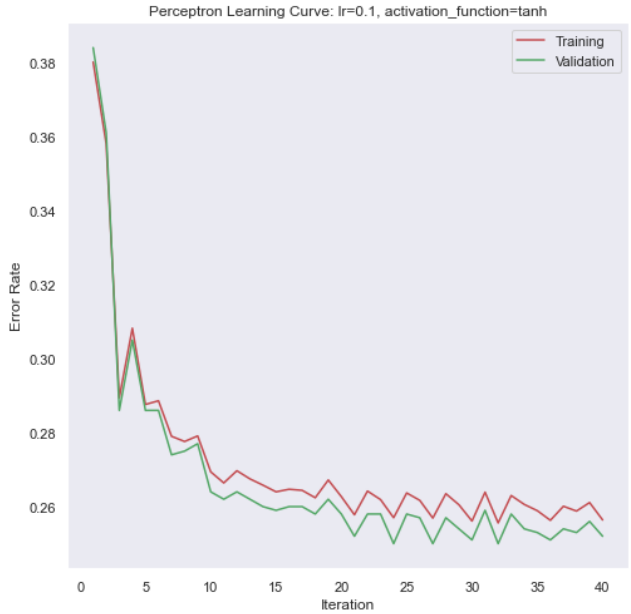


اما تابع فعال سازی Sigmoid بخاطر رنج کمتری که دارد باعث میشود یادگیری کند تر انجام بگیرد (خروجی اعداد کمتری از نظر اندازه هستند) و به تعداد ایتریشن بیشتر یا نرخ یادگیری بزرگتر نیاز داشته باشیم.

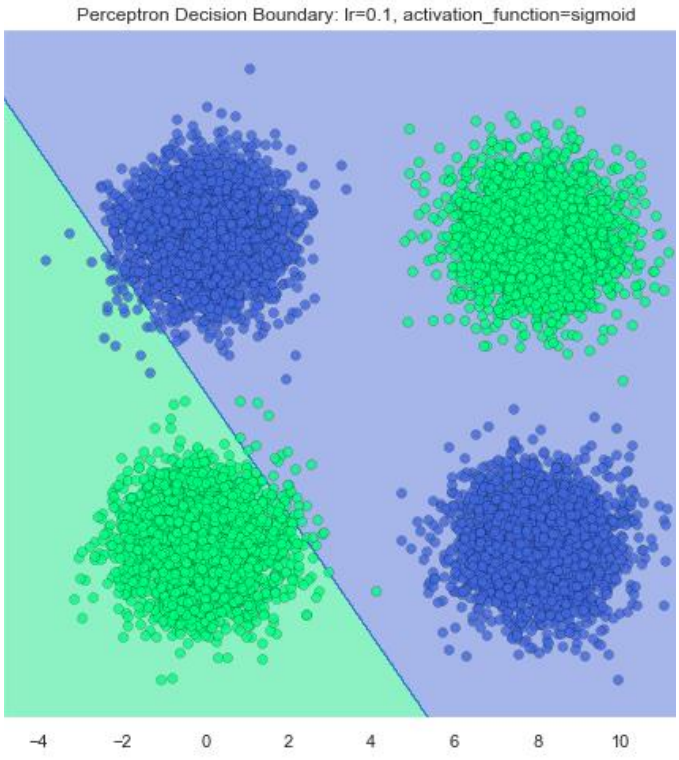
و همانطور که در سوال ۳ توضیح داده شد، توابع فعال سازی در آموزش پرسپترون نقشی ندارند زیرا در نهایت برچسب داده ها (خروجی تابع Threshold) است که برای آپدیت وزن ها استفاده میشود. و تابع فعال سازی در این تصمیم گیری نقشی ندارد و صرفا اعداد قبل از تابع Threshold را کمی تغییر میدهد که این تغییر به اندازه ای نیست که تصمیم گیری در مورد برچسب داده ها عوض بشود.(اعداد مثبت همچنان مثبت میمانند ...)

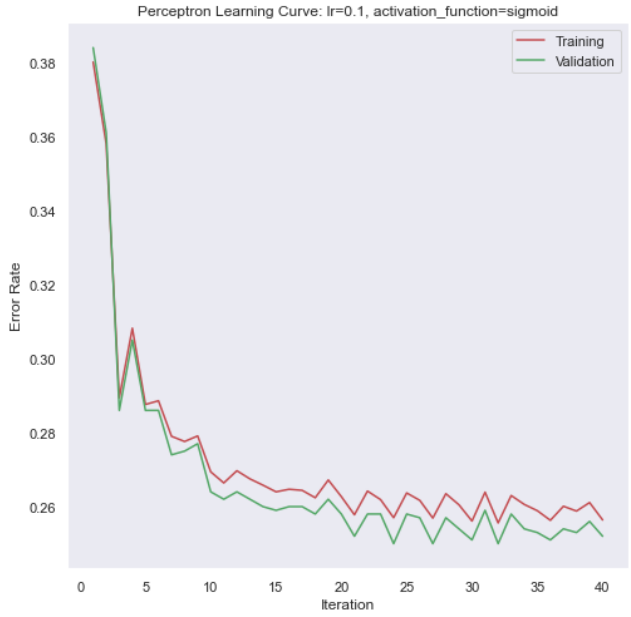
Score on test set: 0.74





Score on test set: 0.74



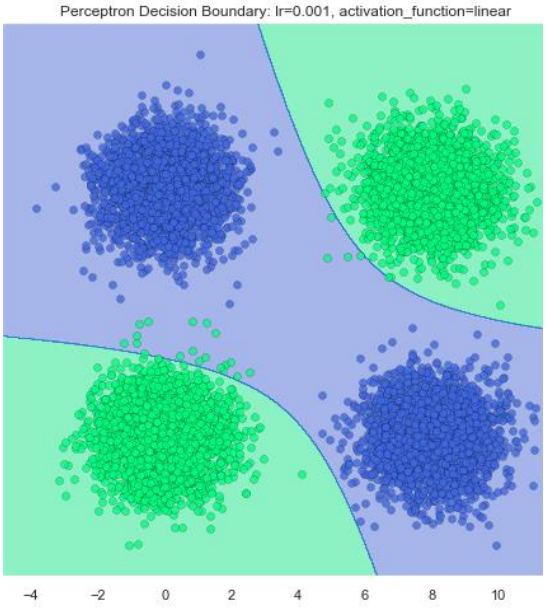


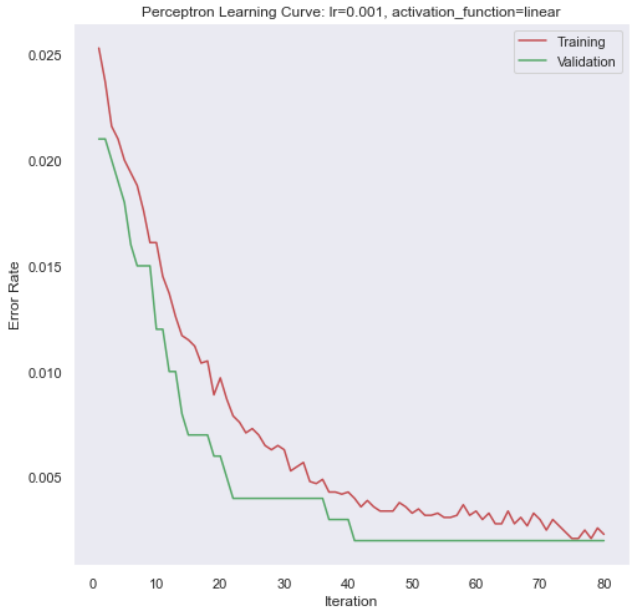
در نهایت به این نتیجه میرسیم که برای داده های غیر خطی نورون پرسپترون بهتر عمل میکند. هر چند هیچ کدام به نتیجه بهینه ۱۰۰ درصد نخواهند رسید.

سوال ۵)

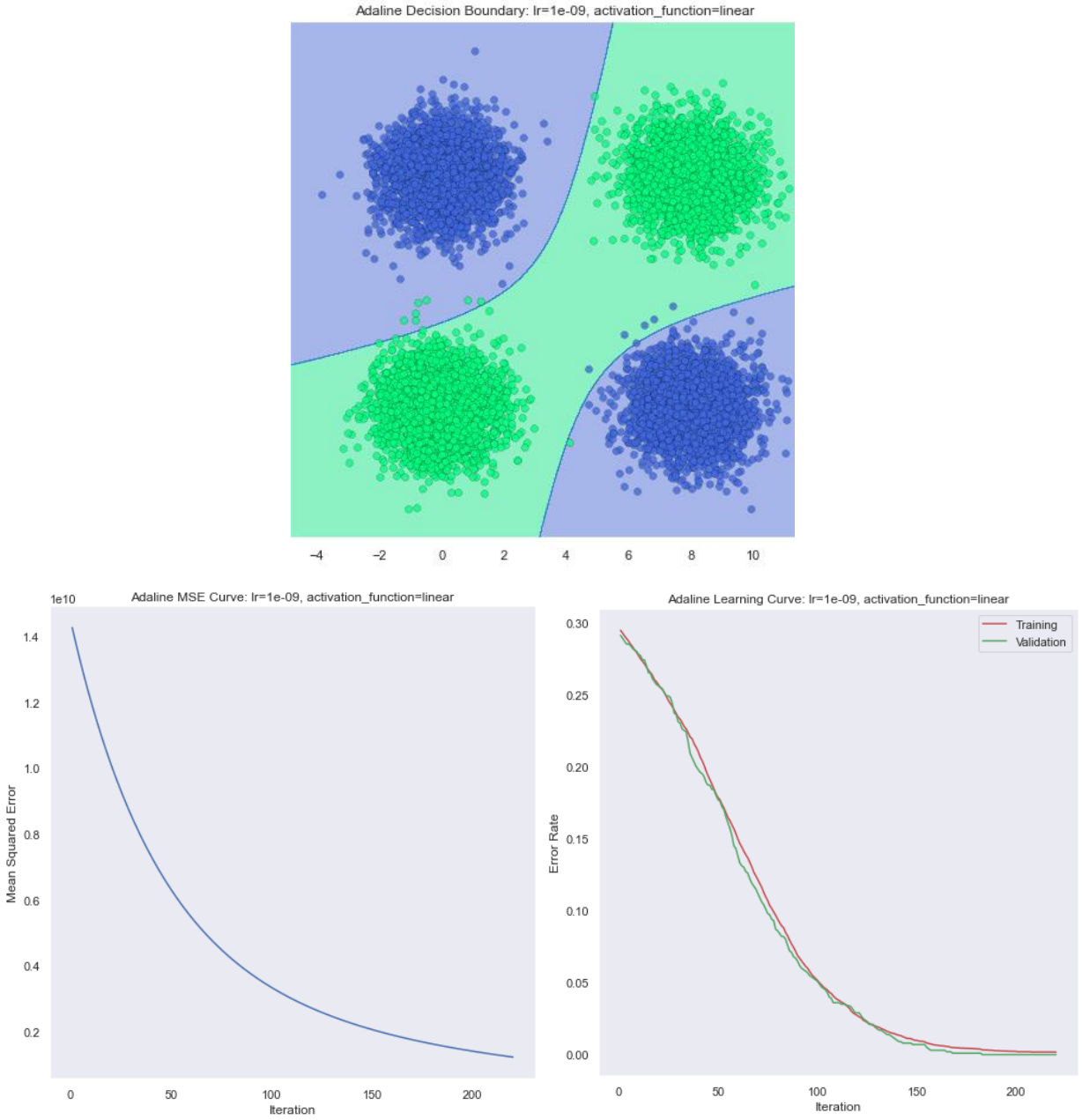
در بخش نهایی این سوال میخواهیم نورون های مان قابلیت جداسازی درجه ۲ را داشته باشند. به همین دلیل علاوه بر x_1, x_2 به آنها $x_1.x_2, x_1^2, x_2^2$ را نیز به عنوان ورودی میدهیم. به این ترتیب مدل ما میتواند تابع جداساز درجه ۲ تولید کند و شکل تابع جداسازمان که قبلا محدود به خط بود الان میتواند بیضی، دایره، هذلولی، سهمی و خط باشد.

لازم است قبل از دیدن نتایج مدل یک نکته مهم را بیان بکنیم; طبق آزمایش های اولیه، هم مدل پرسپترون و هم مدل آدالاین نتایج غیرقابل قبولی دادند. این مشکل را توانستیم با تغییر مقدار اولیه وزن ها از ۰ حل بکنیم. برای این کار وزن اولیه را به طور رندوم تولید کردیم و پس از انجام چندین آزمایش، به مقدار مناسبی برای شروع رسیدیم. پس از آن، هر دو مدل توانستند به خوبی جدا سازی داده ها را انجام دهند. Perceptron) Score on test set : 0.997



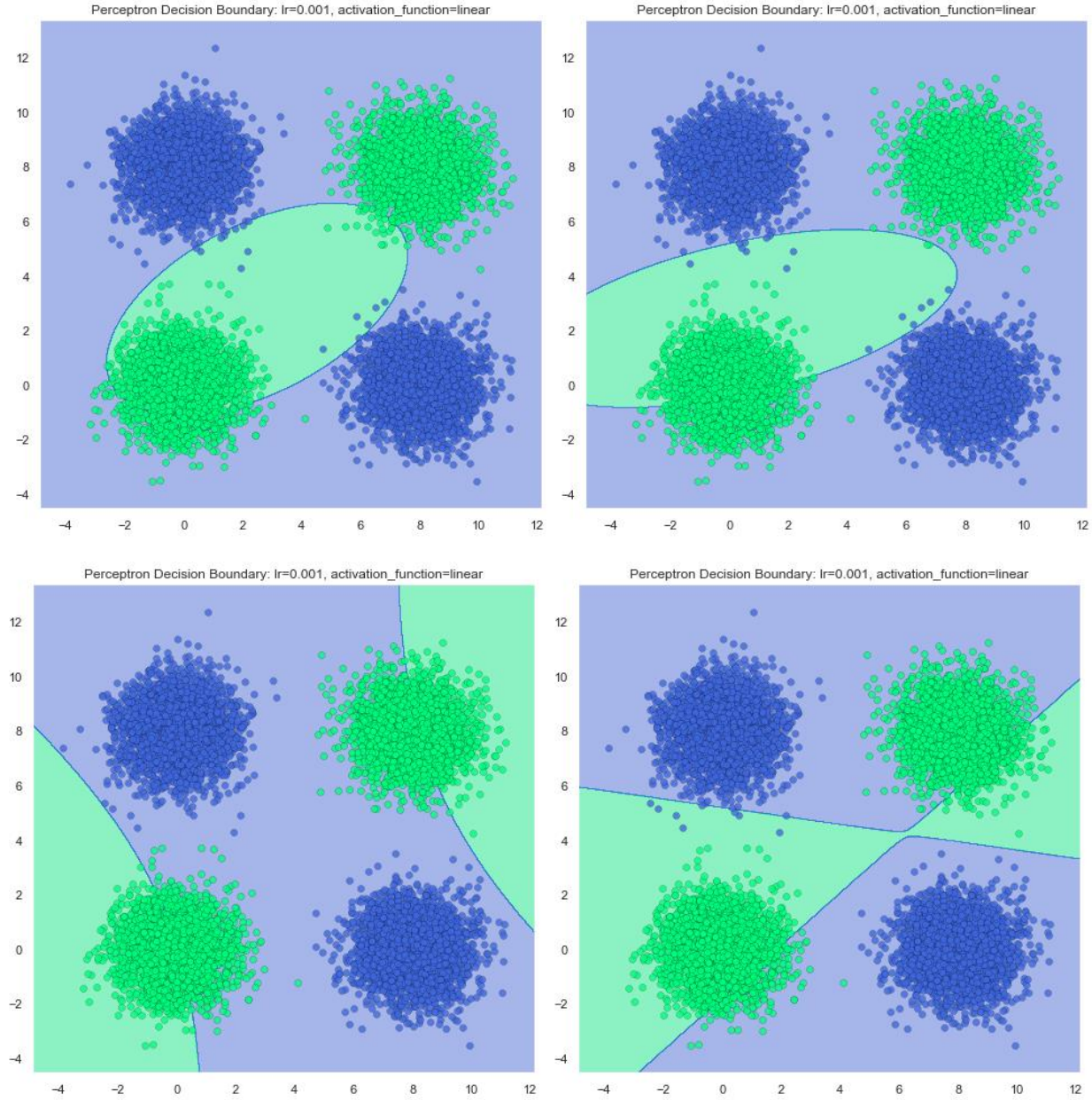


Adaline) Score on test set : 0.997



بنابراین در این حالت هر دو نورون توانستند نتایج خوبی بدست بیاورند اما چون پرسپترون این کار را در تعداد ایتریشن های کمتری انجام میدهد، روش بهتر **perceptron** خواهد بود.

در ادامه تعدادی از نتایج با مقادیر وزن ابتدایی نامناسب را مشاهده میکنیم.



پایان