تمرین سری سوم درس یادگیری ماشین

فایل ضمیمه : hw3.ipynb

تمرین اول :

i	$x_1^{(i)}$	$x_2^{(i)}$	$y^{(i)}$	
1	0	-1	0	
2	1	0	0	
3	2	1	0	$N_0 = 3$
4	1	1	1	
5	-1	1	1	
6	-1	-1	1	$N_1 = 4$
7	-1	-1	1	

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$w^T x^{(i)} = w_1 x_1^{(i)} + w_2 x_2^{(i)}$$

$$\mu_0 = \frac{1}{N_0} \sum_{y^{(i)}=0} x^{(i)} = \frac{1}{N_0} \begin{bmatrix} \sum_{y^{(i)}=0} x_1^{(i)} \\ \sum_{y^{(i)}=0} x_2^{(i)} \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 0+1+2 \\ -1+0+1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\mu_1 = \frac{1}{N_1} \sum_{y^{(i)}=1} x^{(i)} = \frac{1}{N_1} \begin{bmatrix} \sum_{y^{(i)}=1} x_1^{(i)} \\ \sum_{y^{(i)}=1} x_2^{(i)} \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1-1-1-1 \\ 1+1-1-1 \end{bmatrix} = \begin{bmatrix} -1/2 \\ 0 \end{bmatrix}$$

$$\mu_0' = w^T \mu_0 = w_1 \mu_{01} + w_2 \mu_{02} = w_1$$

$$\mu_1' = w^T \mu_1 = w_1 \mu_{11} + w_2 \mu_{12} = -\frac{1}{2} w_1$$

$$s_0'^2 = \sum_{y^{(i)}=0} (w^T x^{(i)} - \mu_0')^2 = \sum_{y^{(i)}=0} (w_1 x_1^{(i)} + w_2 x_2^{(i)} - w_1)^2 =$$

$$(0 - w_2 - w_1)^2 + (w_1 + 0 - w_1)^2 + (2w_1 + w_2 - w_1)^2 = 2(w_1 + w_2)^2$$

$$s_1'^2 = \sum_{y^{(i)}=1} (w^T x^{(i)} - \mu_1')^2 = \sum_{y^{(i)}=1} (w_1 x_1^{(i)} + w_2 x_2^{(i)} + \frac{1}{2} w_1)^2 =$$

$$(w_1 + w_2 + \tfrac{1}{2} w_1)^2 + (-w_1 + w_2 + \tfrac{1}{2} w_1)^2 + (-w_1 - w_2 + \tfrac{1}{2} w_1)^2 \times 2 =$$

$$(\tfrac{3}{2} w_1 + w_2)^2 + (\tfrac{1}{2} w_1 - w_2)^2 + 2(\tfrac{1}{2} w_1 + w_2)^2 = 3w_1^2 + 4w_2^2 + 4w_1 w_2$$

$$J(w_1, w_2) = \frac{(\mu_0' - \mu_1')^2}{s_0'^2 + s_1'^2} = \frac{\frac{9}{4} w_1^2}{5w_1^2 + 6w_2^2 + 8w_1 w_2}$$

$$\frac{\partial J}{\partial w_1} = \frac{9}{4} \times \frac{2w_1(5w_1^2 + 6w_2^2 + 8w_1w_2) - w_1^2(10w_1 + 8w_2)}{(5w_1^2 + 6w_2^2 + 8w_1w_2)^2} = 0$$

$$10w_1^3 + 12w_1w_2^2 + 16w_1^2w_2 - 10w_1^3 - 8w_1^2w_2 = 0$$

$$4w_1w_2(3w_2 + 4w_1 - 2w_1) = 0$$

$$\rightarrow \begin{cases} w_1 = 0 \\ w_2 = 0 \\ 3w_2 = -2w_1 \end{cases}$$

$$\frac{\partial J}{\partial w_2} = \frac{9}{4} \times \frac{0 - w_1^2(0 + 12w_2 + 8w_1)}{(5w_1^2 + 6w_2^2 + 8w_1w_2)^2} = 0$$

$$w_1^2(12w_2 + 8w_1) = 0$$

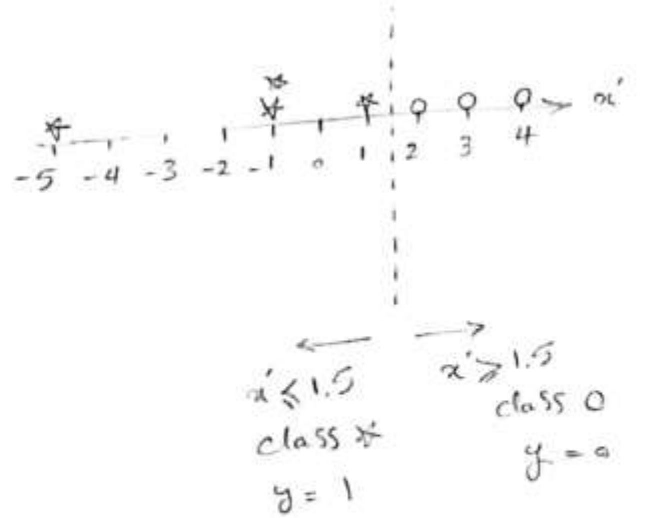
$$\rightarrow \begin{cases} w_1 = 0 \\ 3w_2 = -2w_1 \end{cases}$$

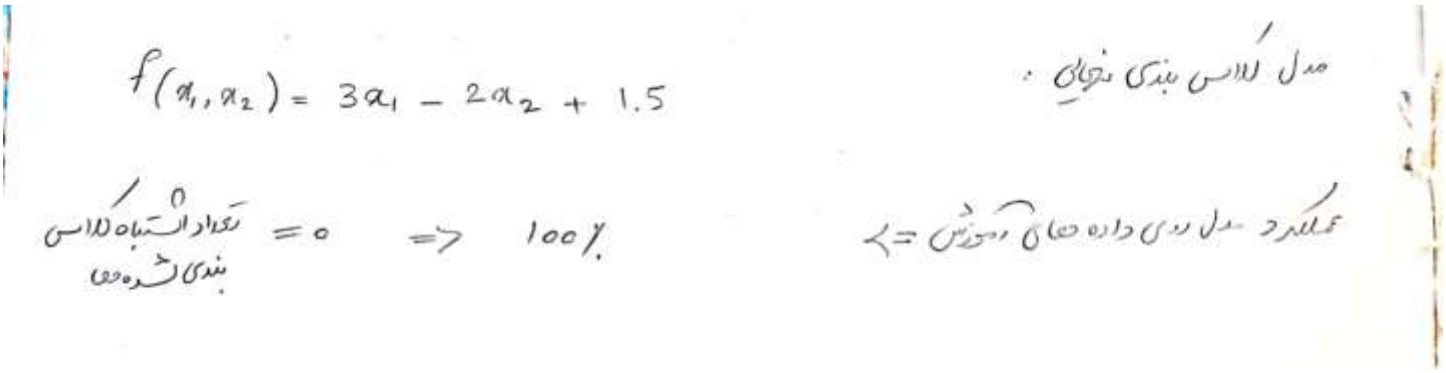
$$3w_2 = -2w_1, \quad w_2^2 + w_1^2 = 1 \rightsquigarrow \frac{4}{9}w_1^2 + w_1^2 = 1 \rightsquigarrow w_1 = \frac{3}{\sqrt{13}}, \quad w_2 = -\frac{2}{\sqrt{13}}$$

$$\uparrow \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$

که بردار بهینه را نشان می‌دهد. راستای خواسته شده $-2x = 3y$ روی صفحه‌ی xy است.
کل داده‌ها را روی این راستا تصویر می‌کنیم:

	x_1	x_2	$x' = w^T x$
1	0	-1	2
2	1	0	3
3	2	1	4
4	1	1	1
5	-1	1	-5
6	-1	-1	-1
7	-1	-1	-1

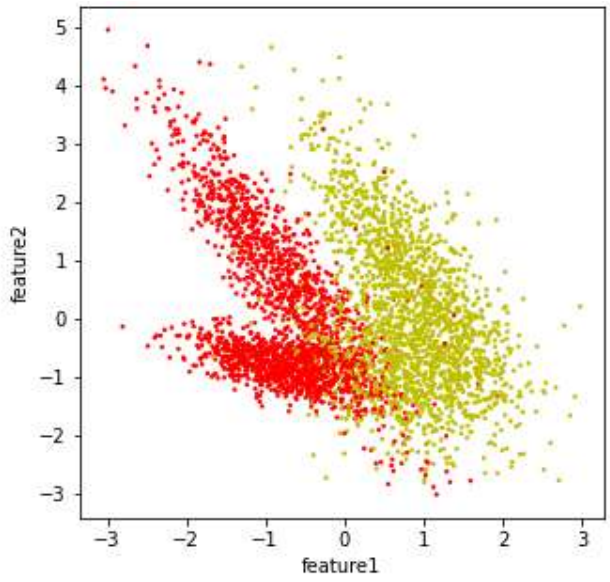




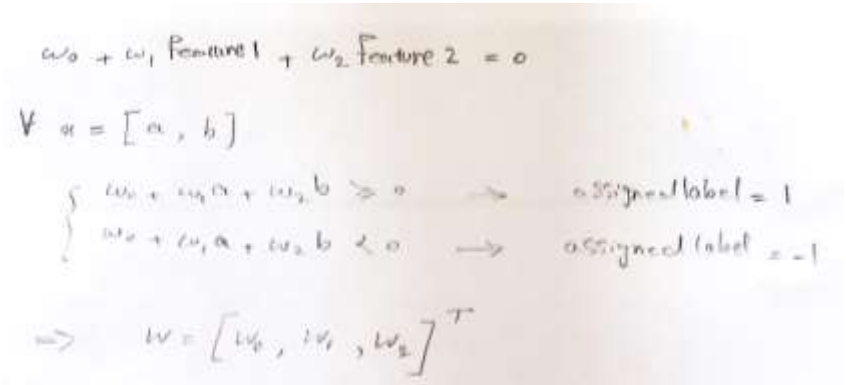
تمرین دوم :

روی داده های فایل train.csv مدل هر مرحله ساخته شده و با داده های test.csv پارامترهای مدل ارزیابی شده. این مجموعه دادگان سه ستون دارد که دو ستون اول feature1 و feature2 به عنوان ویژگی های ورودی استفاده شده و labelرا برچسب هر داده گرفتم که مقداری برابر ۰ یا ۱ دارد که برای راحتی کار -۱ و ۱ در نظر گرفتم. یک ستون ۱ هم اضافه کردم تا محاسبات بایاس مدل راحت تر انجام شود. این تغیرات روی هر دو دسته ی تست و اموزش قبل از کار با دیتاست ها انجام می‌گیرد. دسته بند هر دو مرحله یک پرسپترون است که نرخ یادگیری آن را برابر ۰,۰۰۱ گرفتم وشرط توقف الگوریتم تکرار یادگیری و آپدیت وزن ها، یا صفر شدن تعداد داده های اشتباه کلاس بندی شده اشت و یا حداکثر ۱۰۰ بار تکرار فرایند. که با توجه به درهم رفتگی داده های این مساله شرط اول اتفاق نمیوفتد چون داده ها با مدل خطی جداشدنی نیستند.

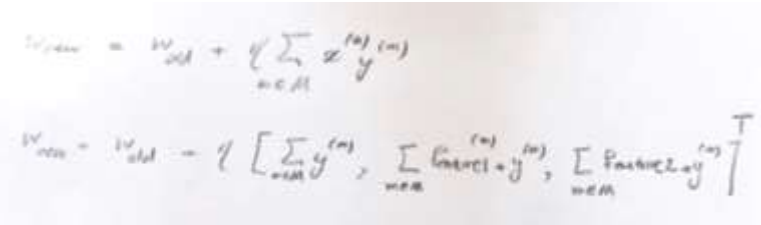
نمودار دو ویژگی روی شکل زیر داده های قرمز نظیر برچسب -۱ و زرد نظیر برچسب ۱ هستند.



مدل پرسپترون این مرحله یک بردار سه تایی به صورت زیر است.



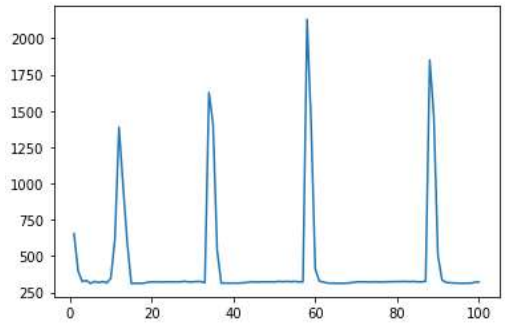
این مقادیر با به روز کردن هرباره ی وزن ها با تمام داده های آموزش (batch mode) از رابطه ی زیر به دست می‌آید.



این مدل در تابع train_perceptron پیاده می‌شد. هربار با پارامترهای به دست آمده مرحله قبل با تابع classify برچسب تمام مجموعه ی آموزش را با رابطه ی XW به دست می‌آورد و در ستون 'assigned_label' قرار می‌دهد. سپس تابع find_missed با مقایسه ی مقدار براورد شده با label داده ها ، در ستون 'missclass' تعین می کند که داده ی مورد نظر درست یا اشتباه دسته بندی شده. و سپس اپدیت وزن ها از رابطه بالا محاسبه می شود.

حاصل آموزش داده ها برداری به صورت زیر است. همچنین تعداد خطاهای طبقه بندی روی داده های آموزش در هر مرحله در طول تکرارها در زیر آمده.

W0	W1	W2
0.096305	1.383040	0.372819



الگوریتم اگر در یکی از قله های نمودار متوقف می شد مدل خطای بالا داشت و قابل استفاده نبود. بهتر از شرط توقف ترکیبی از خطا و تعداد تکرار باشد. اینجا مدل من دچار چنین مشکلی نشده پس نتایج را با همین وضعیت می‌شود تحلیل کرد.

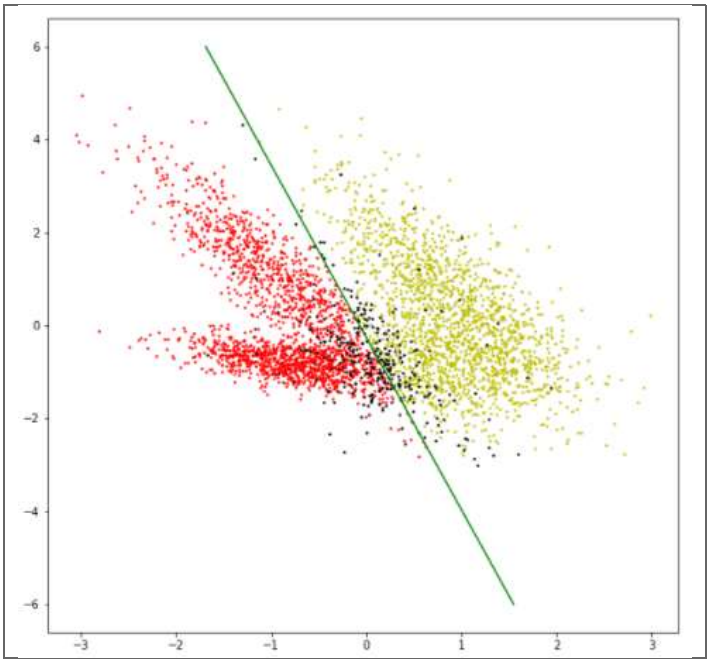
داده های تست و آموزش را با مدل برازش می‌دهم و با شمارش تعداد اشتباه کلاس بندی شده ها معیارهای دقت و خطای زیر به دست می آید و همینطور روی نمودارهای هر دو دسته داده های تست و آموزش خط سبز رنگ مدل پرسپترون است و داده های رنگی داده هایی هستند که درست دسته بندی شده اند و نقاط سیاه رنگ نقاطی هستند که مدل نتیجه ی خطا برای نام کلاس آن ها براورد کرده.

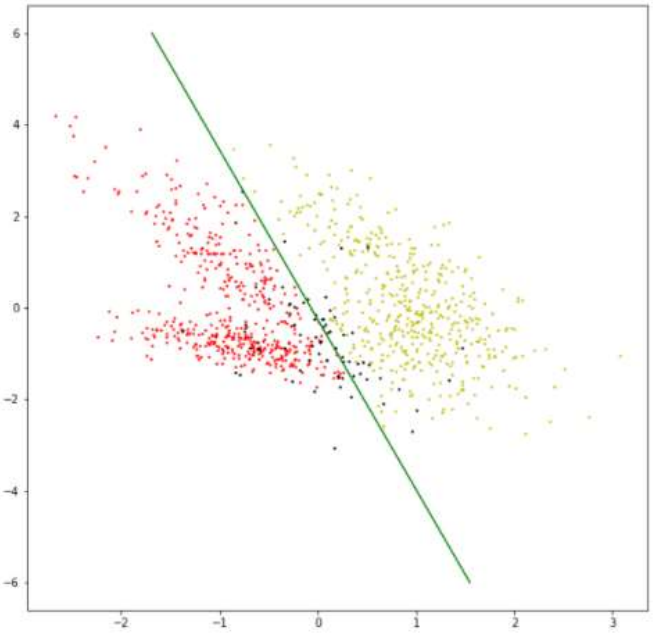
train

```
total =
4000
total_missclassed =
323
misclassed_ratio =
0.08075
correct_ratio =
0.91925
```

test

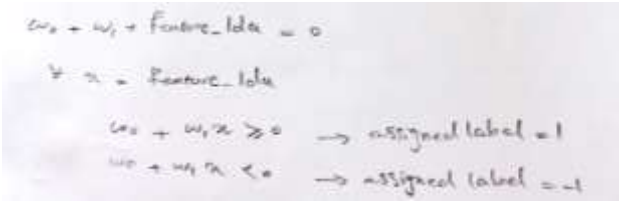
```
total =
1000
total_missclassed =
73
misclassed_ratio =
0.073
correct_ratio =
0.927
```



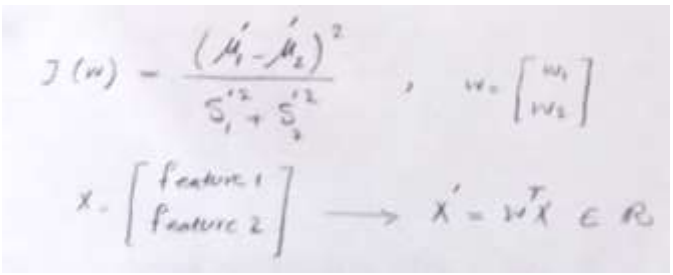


هرچند استفاده از کلاس بندی خطی بهترین مدل برای دسته بندی این داده ها نیست (چون مرز خطی ندارند) اما عملکرد مدل هم روی داه های تست و هم روی داده های آموزشی دقتی بالاتر از ۹۰ درصد دارد. اگر مدل پیچیده تری انتخاب کنم که علاوه بر رابطه ی خطی بین ویژگی‌ها با برچسب، تاثیر موارد دیگری مثل حاصل ضرب ویژگی ها درهم یا رابطه های غیرخطی از آن‌ها را با برچسب در نظر بگیرد دقت مدل هم روی داده های تیت هم آموزش بالاتر می‌رود (تا جایی که دچار خطای بیش برازش و پیچیدگی شدید مدل نشده باشد.) ولی حتی همین مدل خطی هم عملکرد مناسبی دارد.

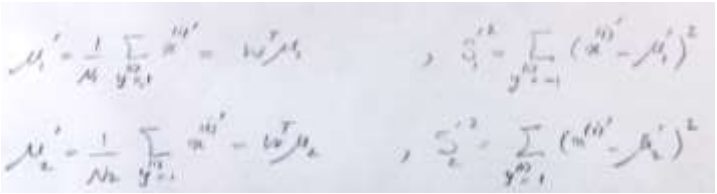
با استفاده از LDA بردار X که دو ویژگی دارد را به فضایی با یک بعد کمتر می‌برم. یعنی تمام داده های را به یک عدد تبدیل میکنم و روی این عدد یک مدل پرسپترون برازش می‌دهم. این بار مدل پرسپترون من به صورت زیر خواهد بود و دو پارامتر دارد.



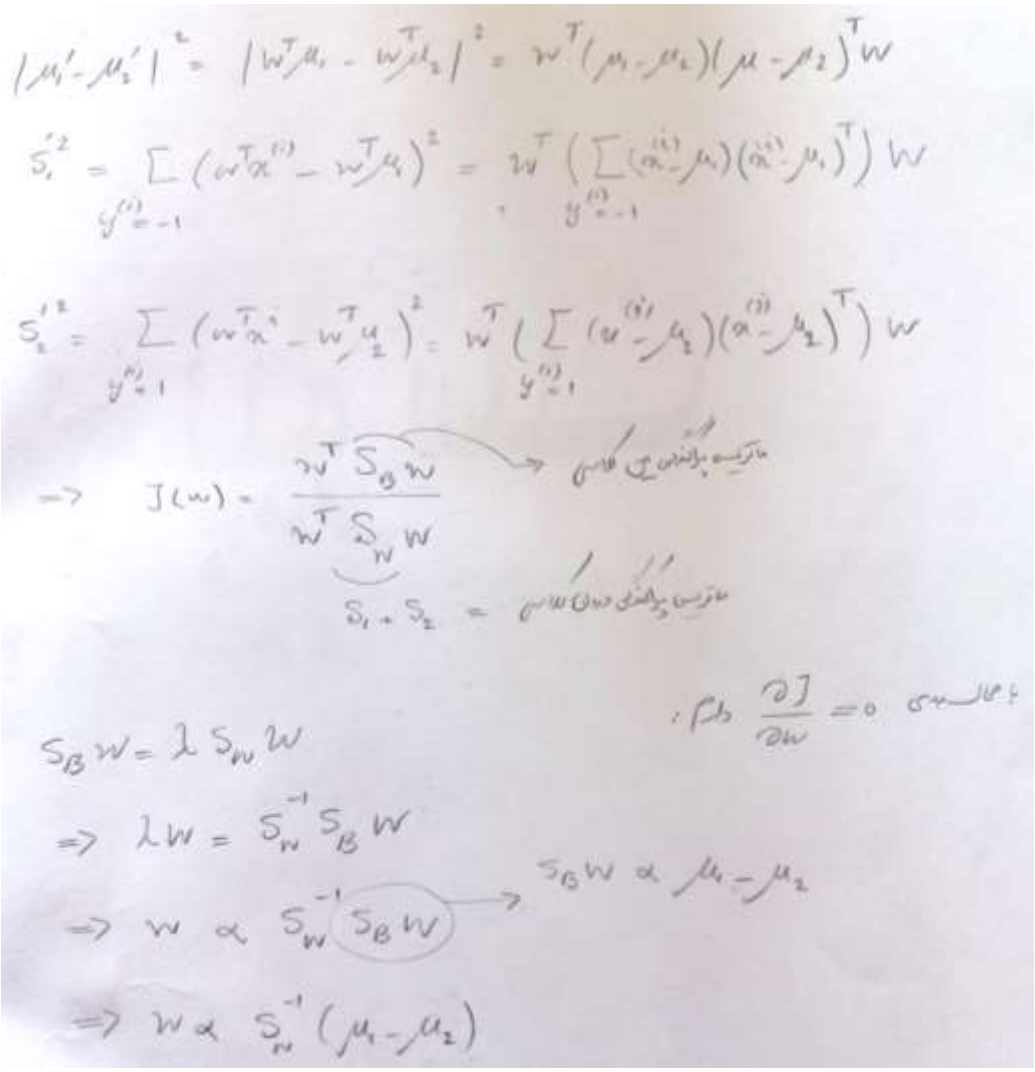
برای محاسبه ی ویژگی LDA نظیر داده ها باید برداری را در راستایی پیدا کنم که اگر داده های دو کلاس روی آن راستا نگاشته شوند، حداکثر فاصله را از داده های کلاس مخالف و حداقل فاصله را از داد های کلاس مشابه داشته باشند. برای این کار به دنبال یک بردار در فضای دوبعدی دو ویژگی feature1 و feature2 می‌گردم که تابع زیر را بیشینه کند.



که در این تابع :



آنچه برای محاسبه ی LDA داده ها نیاز دارم راستای بردار است. پس لازم نیست مقدار بیشینه را حساب کنم. تنها به دست آوردن راستا کافی است. با توجه به روابط زیر :



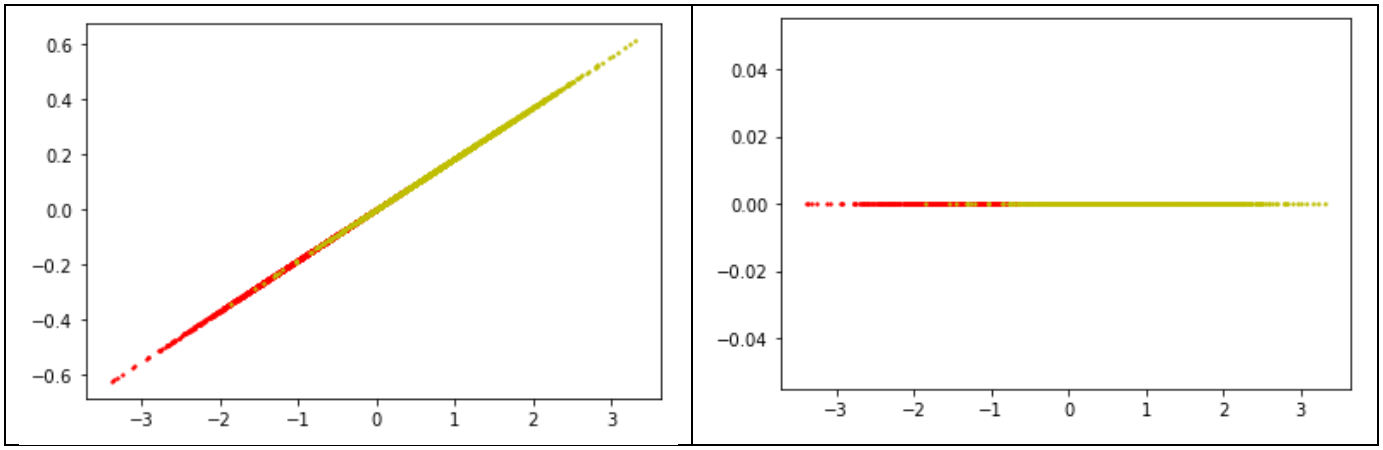
کافیست وارون ماتریس Sw و تفاضل میانگین های دو ویژگی داده های آموزشی را محاسبه کنم تا راستای w پیدا شود. تابع LDA(d) این کار را پیاده سازی کرده و تمام داده ها را روی این راستا نگاشت می‌هد و عدد به دست آمده را در ستونی با نام lda روی داده ها نگه می‌دارد. این ستون همان ورودی پرسپترون جدید برای ساختن مدل و کلاس بندی خواهد بود. در نهایت یک بردار یکه در راستای مورد نظر برمی‌گرداند.

قبل از اینکه داده ها را به LDA بدهم، یک استانداردسازی خواسته شده روی ستون های feature1 و feature2 از رابطه زیر انجام می‌دهم. (همین استانداردسازی باید برای داده های تست هم قبل از ارسال به تابع LDA با همین میانگین و واریانس بدست امده از روی داده های آموزش انجام بگیرد. این میانگین و واریانس داده های آموزش را براوردی از میانگین و واریانس توزیع اصلی و مجهول داده ها در نظر گرفتم.)

بردار w به دست آمده :

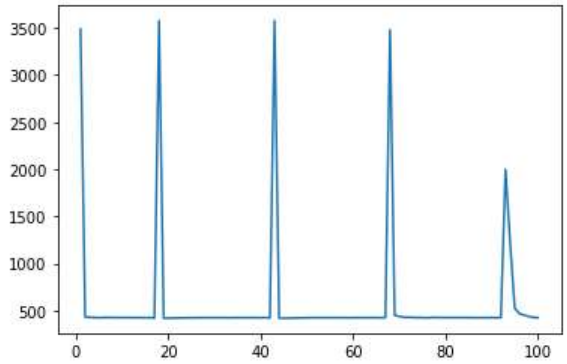
0.98342694
0.18130485

مقدار ستون lda رو روی خط افقی نشان داده شده میبینیم و اگر این عدد را برای هر داده روی راستای w پیدا کنم، خط به دست آمده شکل زیر همان راستای پیدا شده است که تابع J را بیشینه کرد. در زمان کار روی داده های تست و ارزیابی مدل، مقدار lda داده های تست با همین بردار محاسبه خواهد شد.



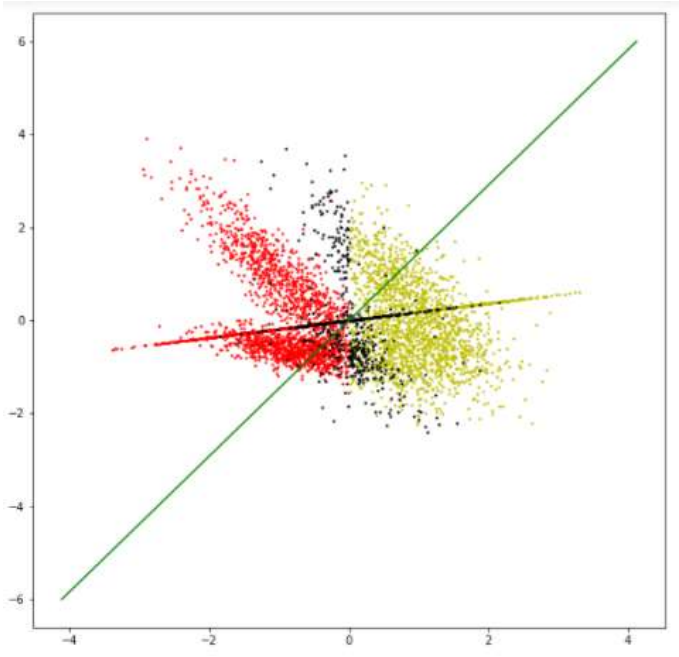
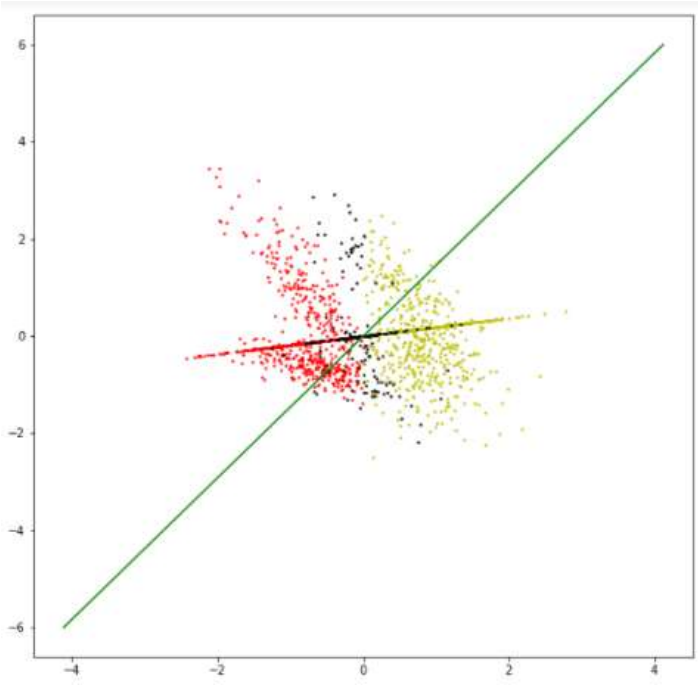
مدل پرسپترون را با همان روشی در قسمت قبل توضیح دادم آموزش می‌دهم. همان طور که در روابط بالاتر گفته شد این مدل یک ورودی و دو پارامتر دارد و این ورودی همان lda است. مدل به دست آمده :

W0	*W1*
-0.006724	*1.456784*



در مورد قله های نمودار آبی و شرط توقف الگوریتم همچنان آنچه گفته بودم صادق است ولی اینجا چون روی خطای کمی متوقف شده، کار را ادامه می‌دهم. بردار مدل پرسپترون به دست آمده بایاس تقریبا صفر دارد. در واقع این مدل چون روی داده های یک بعدی کار می‌کند که این داده ها از برازش روی یک بردار یکه به دست آمده اند و ابرصفحه ی این فضا یک نقطه ست پس تبدیل می‌شود به یک مقدار ثابت یا آستانه که یک تابع پله ای طبق آن مقدار بین دو کلاس تمایز قایل می‌شود. (تابع پله ای اینجا sign بوده.)

برازش داده های تست و آموزش نتایج زیر را دارد. روی نمودارهاس این بخش، خط سبز مدل پرسپترون است. خط دوم همان نگاشت مقادیر lda روی راستای w هست. محل تلاقی این دو خط، معیار دیته بندی است. یعنی به ازای داده هایی که سمت چپ آن نقطه مقدارlda شان باشد در دسته قرمز -۱ و در غیر این صورت در دسته ی زرد ۱ قرار خواهند گرفت. داده های رنگی داده های درست دسته بندی شده هستند و داده های سیاه اشتباه دسته بندی شده اند. توجه شود که نقاط پراکنده ی نشان داده شده (خارج خط) همان نقاط اصلی feature1 و feature2 هستند و فقط برای مشخص شدن درست و غلط کلاس بندی شدنشان نمایش داده شدند تا مقایسه این حالت با حالت بدون LDA از روی شکل واضح تر یاشد. این مقادیر feature1و feature2مستقیما در مدل وارد نشده اند و مدل اینجا یک پرسپترون دو پارامتری است که صرفا روی داده های خط lda عمل می‌کند. و فضای داده های ورودی تک بعدی هست.

train	*test*
total = 4000 total_missclassed = 428 misclassed_ratio = 0.107 correct_ratio = 0.893	total = 1000 total_missclassed = 99 misclassed_ratio = 0.099 correct_ratio = 0.901
	

وضعیت کلی دسته بندی روی هر دو سری داده ها به هم بسیار نزدیک است مثل حالت قبل. اگر نمودار خط افقی داده های تست و آموزش را در نظر بگیرم، اط روس این داده های تک نقطه ای که مدل روی آن ساخته شده واضح است که نمیتوان یک نقطه ی آستانه روی آن پیدا کرد. پس داده های یک بعدی LDA این مساله همچنان به حالت خطی جداپذیر نیستند. و روی دو نمودار بالا و نقاط سیاه رنگ، خط LDA را اگر تنها و بدون شیب در نظر بگیرم، همان خط افقی که قبل تر برای داده های آموزش رسم کردم را دارم که به جای برچسب های واقعی، با مقدار های براورد شده ی مدل رسم شده اند و رنگ سیاه تمام نقاطی ست که کلاس واقعی شان در آستانه ی پیدا شده برای این مدل صدق نمی‌کند. با وجود این درهم رفتگی داده ها بازهم مدل کارایی چندان بدی ندارد. ولی اگر نتایج را با نتایج مدل پرسپترون قبلی مقایسه کنم، هرچند اختلاف خیلی زیاد نیست اما افت معناداری در دقت مدل هم در داده های تست و هم آموزش پیدا شده. در بخش قبل گفتم که اصل خطای مدل ناشی از پیچیدگی کم آن است. اتفاقی که افتاد اینکه با تک بعدی کردن داده های مساله، مدل و ورودی هایش بازهم کمی ساده تر شدند و آن جداکنندگی که مدل قبلی قادر بود روی داده های با دو ویژگی بیاید در حالت اسکالر شدن داده ها و با مدل دوم که فقط از یک نقطه تشکیل شده بود محدودتر شد. نتیجه اینکه حدود سه درصد که افت دقت مدل روی داده های تست مدل دوم نسبت به اولی. استفاده از LDA روی این داده ها نتیجه ی مطلوب تر یبه دست نمی‌دهد.

تمرین سوم :

فرض ، N داده‌ی m کلاسه ، n ویژگی و از کلاس $1 \le j \le m$ برابر N_j داده وجود دارد .

$$X^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \dots \\ x_n^{(i)} \end{bmatrix}_{n\times 1} \longrightarrow X'^{(i)} = \begin{bmatrix} x_1'^{(i)} \\ \dots \\ x_{m-1}'^{(i)} \end{bmatrix}_{(m-1)\times 1} , \quad X'^{(i)} = W^T X^{(i)}$$

داده‌هایمان از فضای n بعدی ⟵ به فضای $m-1$ بعدی ببریم

$$\Rightarrow W = \begin{bmatrix} w_1 & w_2 & \dots & w_{m-1} \end{bmatrix}_{n\times(m-1)}$$

از قبل برای حالت دو کلاسه ماتریس پراکندگی درون کلاسی :

$$S_W = S_1 + S_2$$

$$S_j = \sum_{y^{(i)}=j} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T \qquad \forall j \in \{1, 2\}$$

همین تعریف به ازای $\forall j \in \{1, 2, \dots m\}$ کلاس :

و S_j دقیقا همان رابطه بالا که در آن μ_j برابر است با :

$$S_w = \sum_{j=1}^{m} S_j$$

هر S_j یک ماتریس $n\times n$ است .

$$\mu_j = \frac{1}{N_j} \sum_{y^{(i)}=j} x^{(i)}$$

S_W هم یک ماتریس $n\times n$ هست .

حالا برای ماتریس پراکندگی بین کلاسی دو حالته داشتیم ، $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$

یک μ روی کل داده‌ها تعریف می‌کنیم ،

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$$

حالا به ازای هر $1 \le j \le m$ می‌خواهیم μ_j ها را تا جایی که می‌شود دور از μ (میانگین کل داده‌ها) داشته باشیم .

از طرفی می‌خواهیم این تاثیر به صورت وزن دار باشد ، یعنی هرچه تعداد اعضای کلاس j بیشتر باشد ، داده‌های کلاس j ناحیه‌ی بیشتری با μ داشته باشند تا جدایی بیشتری بین داده‌های X' پیدا شود پس :

$$S_B = \sum_{j=1}^{m} N_j (\mu_j - \mu)(\mu_j - \mu)^T$$

که یک ماتریس $n\times n$ هست .

$$J(w) = \frac{w^T S_B w}{w^T S_w w}$$

مانند حالت دو کلاسه داریم.

که برای آنکه حاصل بین داده های دو کلاس مختلف بیشترین و داده های بین هر کلاس در خودش کمترین پراکندگی را داشته باشد، $J(w)$ باید بیشینه شود.

$$\max \quad J(w)$$

$$\Rightarrow \frac{\partial J}{\partial w} = 0 \qquad \text{رهشتق گیری مثل حالت دو کلاسی ...}$$

در حالت دو کلاسه داشتیم:

$$S_w^{-1} S_B w = \underbrace{\lambda w}_{\hookrightarrow = J(w) \in R}$$

حالا برای حالت m کلاسی که W را به صورت $n-1$ بردار n تایی داریم، برای هر w_j:

$$S_w^{-1} S_B w_j = \underbrace{\lambda_j w_j}_{J(w_j) \in R} \quad , \quad j \in \{1, 2, \dots m-1\}$$

یادآوری: $W = \underset{n \times (m-1)}{[w_1 \; w_2 \; \dots \; w_{m-1}]}$

و هر w_j مثل حالت دو کلاسه برابر بردار ویژه ی ماتریس $S_w^{-1} S_B$ است.

اگر هرکدام از w_j های متناظر بزرگترین λ_j بدست آمده را پیدا کنیم و در W قرار دهیم، مقدار بیشینه ی $J(w)$ را می توان پیدا کرد و $W = [w_1 \; \dots \; w_j]$ همان ابرصفحه ی مورد نظر برای جداسازی داده ها است.