به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری اول

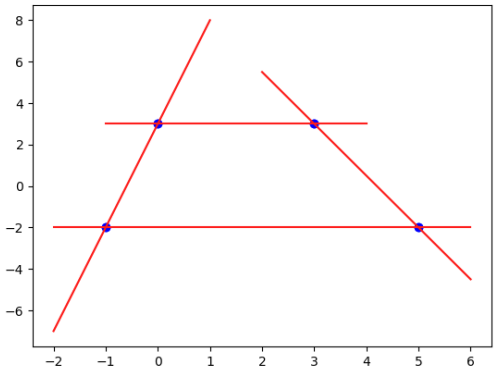
نام و نام خانوادگی	علی ایزدی
شماره دانشجویی	۸۱۰۱۹۹۱۰۲
تاریخ ارسال گزارش	۹ آبان ۱۴۰۰

سوال ۱- Mcculloch pits

ابتدا معادله چهار خط را به دست می آوریم.

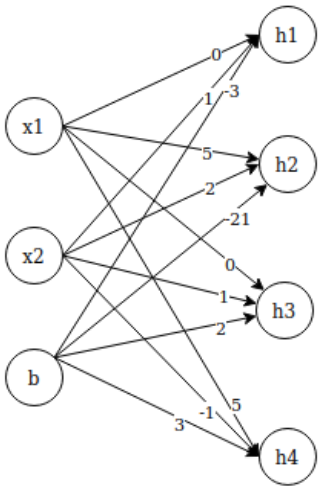
$x_2 - 3 = 0$
$5x_1 + 2x_2 - 21 = 0$
$x_2 + 2 = 0$
$5x_1 - x_2 + 3 = 0$

در شکل ۱-۱ خطوط به دست آمده و نقاط داده شده را رسم کرده ایم تا مشخص شود معادله خطوط به درستی به دست آمده است.



شکل ۱-۱ خطوط رسم شده از معادله خطهای به دست آمده

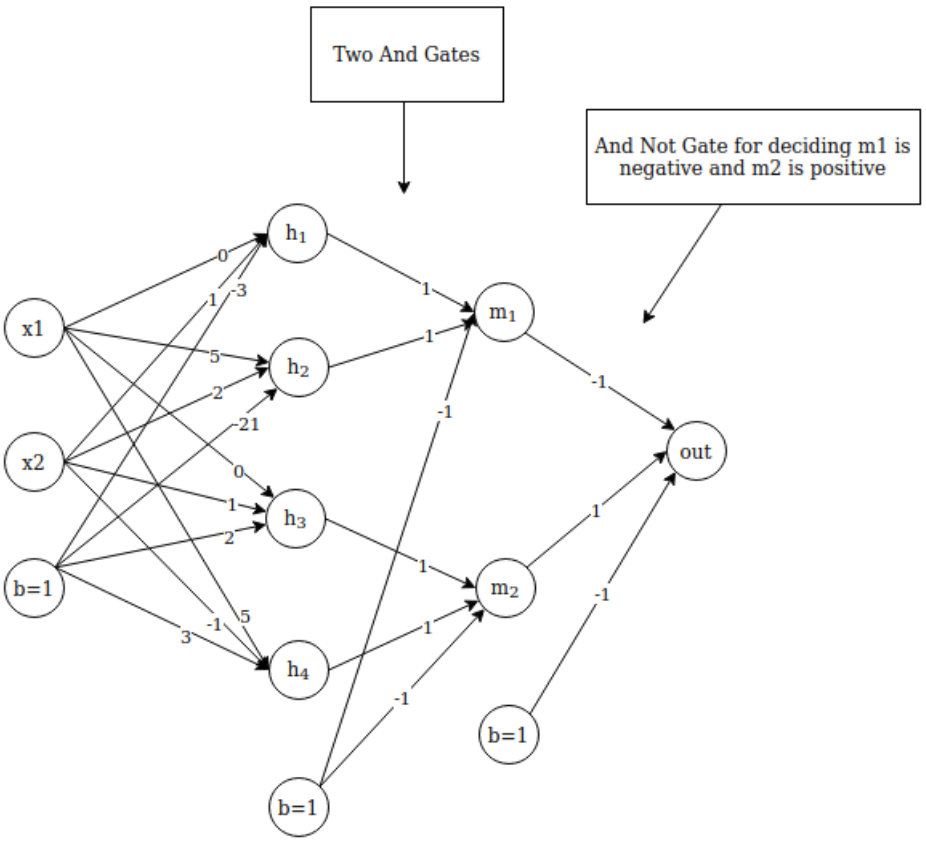
ابتدا باید با استفاده از ضرایب x1 , x2 , b هر خط لایه اول شبکه مک کلاچ پیتز مطابق با شکل ۱-۲ بسازیم.



شکل ۱-۲ معماری شبکه مک کلاچ پیتز

حال بر اساس لایه اول به دست آمده و معادله خطوط برای مشخص شدن نقاط داخل صفحه h1 و h2 باید خروجی منفی و h3 و h4 باید خروجی مثبت داشته باشند.

برای این کار میتوان از نورون های مک پلاچ پیتز and و and not مطابق با کتاب استفاده کرد.
به گونه ای که h1 و h2 باهم و h3 با h4 با هم and شوند و لایه بعدی را با دو خروجی ایجاد m1 و m2 به ترتیب ایجاد کنند.
سپس در لایه سوم m1 و m2 با هم and not میشوند تا مشخص شود m1 باید منفی و m2 باید مثبت باشد.
در نهایت خروجی ای که به دست می آید اگر +1 باشد یعنی نقطه داخل ذوزنقه و اگر -1 باشد خارج آن است.
نورون های and و and not با ورودی های bipolar یا 1 و 1- کار میکنند بنابراین در لایه نهان اول و دوم هر نورون از activation function عبور میکند تا اگر مثبت بود خروجی ۱ و اگر منفی بود خروجی ۱- دهد.
معماری نهایی شبکه مطابق شکل ۱-۳ است.



شکل ۱-۳
معماری نهایی شبکه مک کلاچ پیتز

پیاده سازی با استفاده از ضرب ماتریسی انجام شده است و وزن های بین لایه اول و دوم نهان نیز بین آن هایی که در شکل ۳ کشیده نشده است صفر قرار داده شده است تا بتوان با ضرب ماتریسی لایه نهان سوم را به آسانی به دست آورد.

خروجی شبکه برای نقاط زیر که داخل ذوزنقه اند ۱ به دست آمد:

(2, 2) (1, 1) (0, 2)

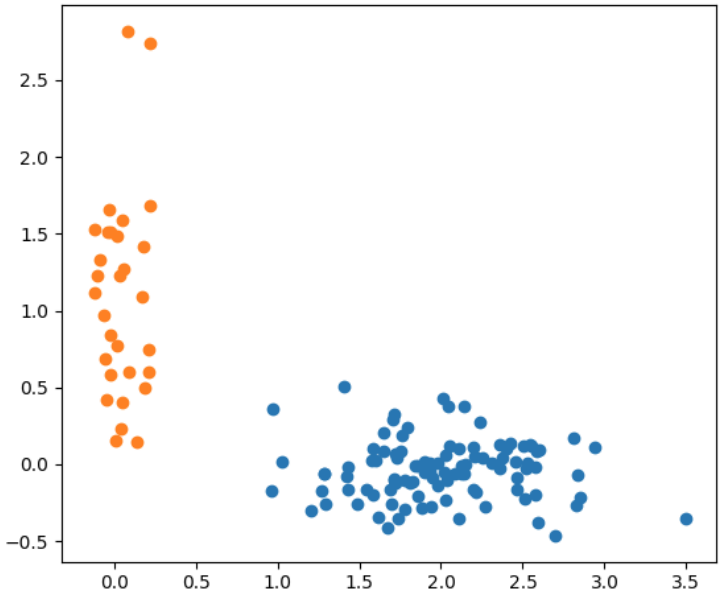
برای نقاط بیرون از ذوزنقه خروجی شبکه ۱- به دست آمد:

(-1, 2) (4, 4) (-5, -5)

سوال ۲ - Adaline

الف)

داده های تولید شده مطابق شکل ۲-۱ رسم شده اند.

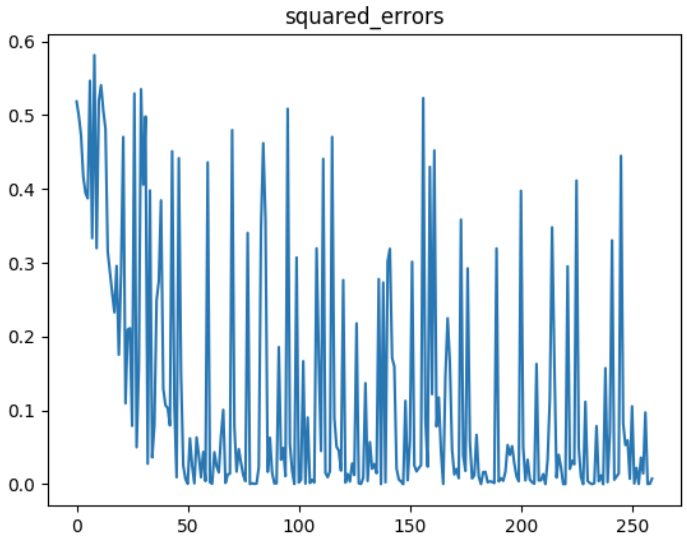


شکل ۲-۱ داده های تولیدی

با استفاده از الگوریتم مطرح شده و پارامترهای زیر الگوریتم Adaline برای جدا سازی این دو دسته آموزش داده شد.

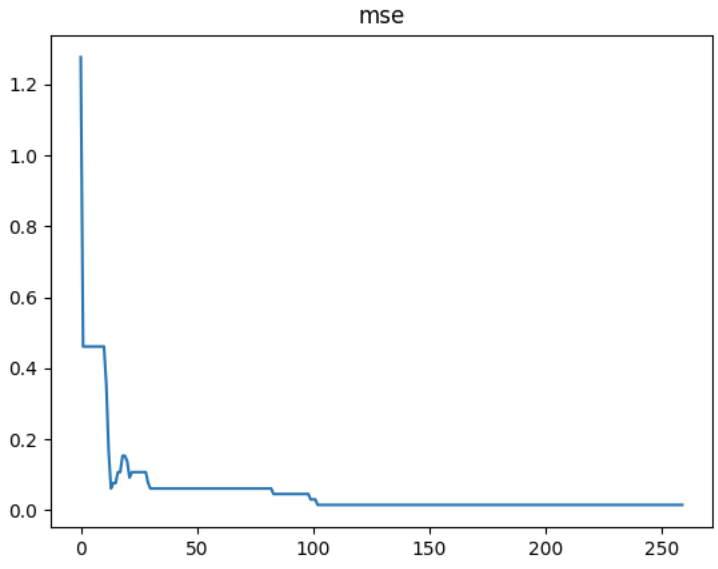
Learning_rate: 0.1
Cost Threshold: 0.001

نمودار $0.5 \times (t - net)^2$ را بعد از هر آپدیت وزن ها برای هر داده مطابق با شکل ۲-۲ رسم شده است.



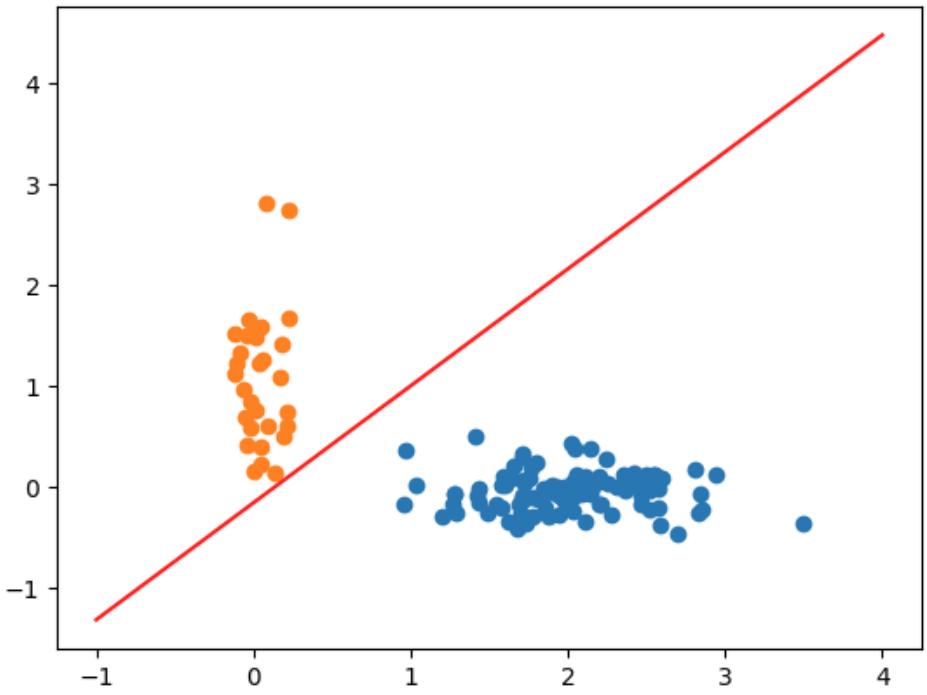
شکل ۲-۲

هم چنین برای بررسی بهتر خطا میانگین مربعات خطا بعد از هر آپدیت وزن برای کل داده ها در شکل ۲-۳ رسم شده است که نشان از کاهش هزینه در طول یادگیری دارد.



شکل ۲-۳

در نهایت با استفاده از وزن های یادگرفته شده خط جداساز دو کلاس در شکل ۲-۴ رسم شده است.



شکل ۲-۴

ب)

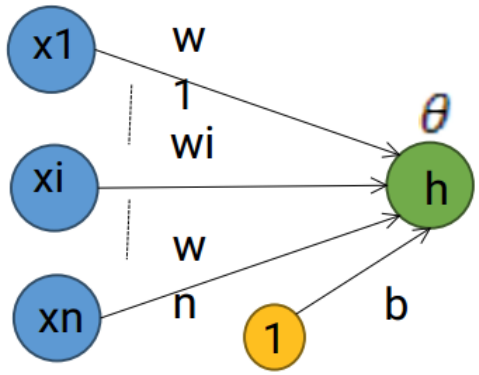
بله.

با توجه به نتایج به دست آمده و به خاطر این که داده ها به صورت خطی از هم جدا پذیرند الگوریتم میتواند خطی برای جدا سازی دو کلاس پیدا کند.

سوال ۳- Perceptron

الف)

هدف از الگوریتم perceptron یادگیری وزن های شبکه ای مطابق با شکل ۳-۱ است.



شکل ۳-۱

خروجی شبکه از روابط زیر به دست آمده که به تعیین یک تنای مناسب دو خط تصمیم گیری به دست می آید که با استفاده از آن ها خروجی ۱ یا ۱- یا بدون تصمیم برای داده ها به دست می آید.

$$\text{net} = w_1x_1 + \cdots . + w_ix_i + \cdots + w_nx_n + b$$

$$(\#) \quad h = \begin{cases} 1 & \text{if} \quad \text{net} > \theta & \text{Active} \\ 0 & \text{if} \quad |\text{net}| < \theta & \text{non - descision} \\ -1 & \text{if} \quad \text{net} < -\theta & \text{Passive} \end{cases}$$

$$\theta : \text{non} - \text{negative threshold}$$

برای یادگیری وزن های W از قاعده به روز رسانی زیر بر اساس داده وروی و برچسب خروجی استفاده میکنیم.

$$(*) \quad w_i(new) = w_i(old) + \alpha x_i t \qquad \alpha : \text{ learning rate}$$

وزن ها ابتدا با مقدار صفر جایگیزین میشود و به ازای هر داده مطابق با عبارت بالا به روز رسانی میشود اگر مقدار تفاضل خروجی شبکه با برچسب t یا h-t صفر نباشد.
هم چنین اگر برای همه داده ها مقدار این خطا صفر شود الگوریتم پایان میابد.

ب)

1) net = 0.1*0 + 0.6*1 + 0.3*1 - 0.5 = 0.4 => h = 1

2) W1 = 0.1 + 0.2*0* -1 = 0.1
   W2 = 0.6 + 0.2*1*-1 = 0.4
   W3 = 0.3 + 0.2*1*-1 = 0.1
   b = -0.5 + 0.2*-1 = -0.7
   net = 0.1*0 + 0.4*1 + 0.1*1 - 0.7 = -0.2 => h = -1

3) خطا صفر است پس دیگر وزن ها برای این داده طبق الگوریتم به روز رسانی نمیشود.

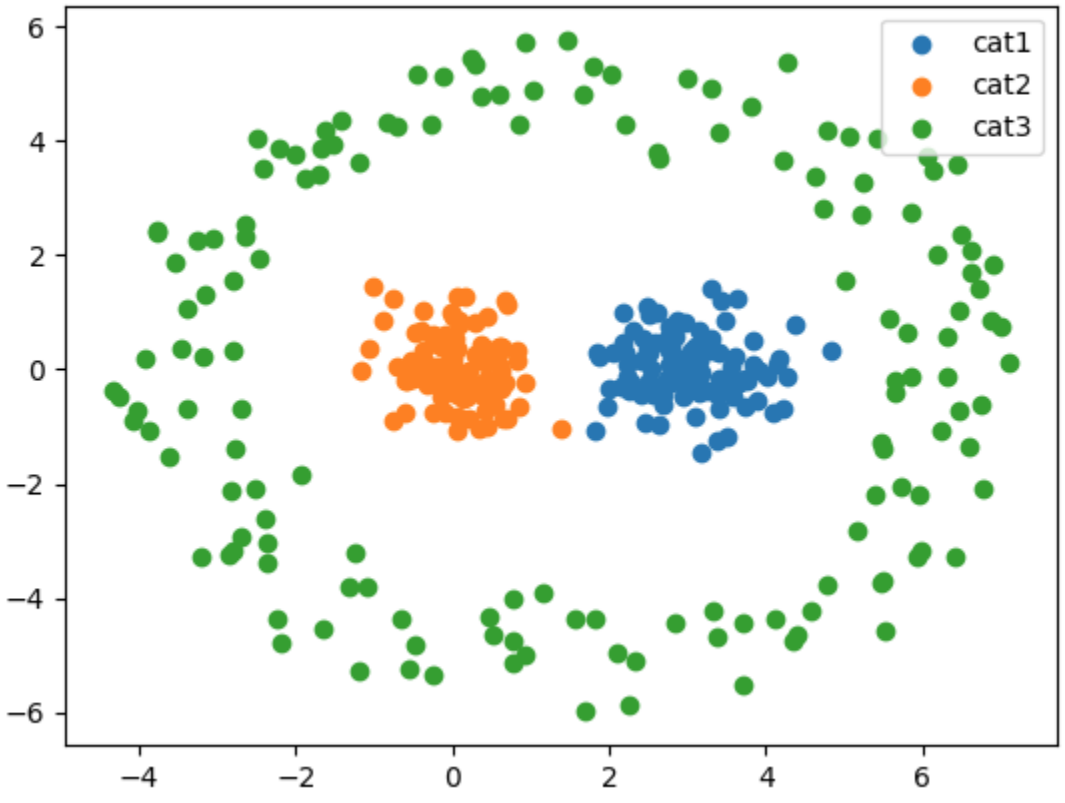
	Sample	W1	W2	W3	b	h	error
1	-1	0.1	0.6	0.3	-0.5	1	2
2	-1	0.1	0.4	0.1	-0.7	-1	0
3	-1	0.1	0.4	0.1	-0.7	-1	0

سوال ۴- Madaline

داده ها را مطابق با شکل ۴-۱ رسم میکنیم.

برای تولید رینگ با استفاده از cos و sin زاویه های ۰ تا ۳۶۰ x و y داده ها تولید شود.

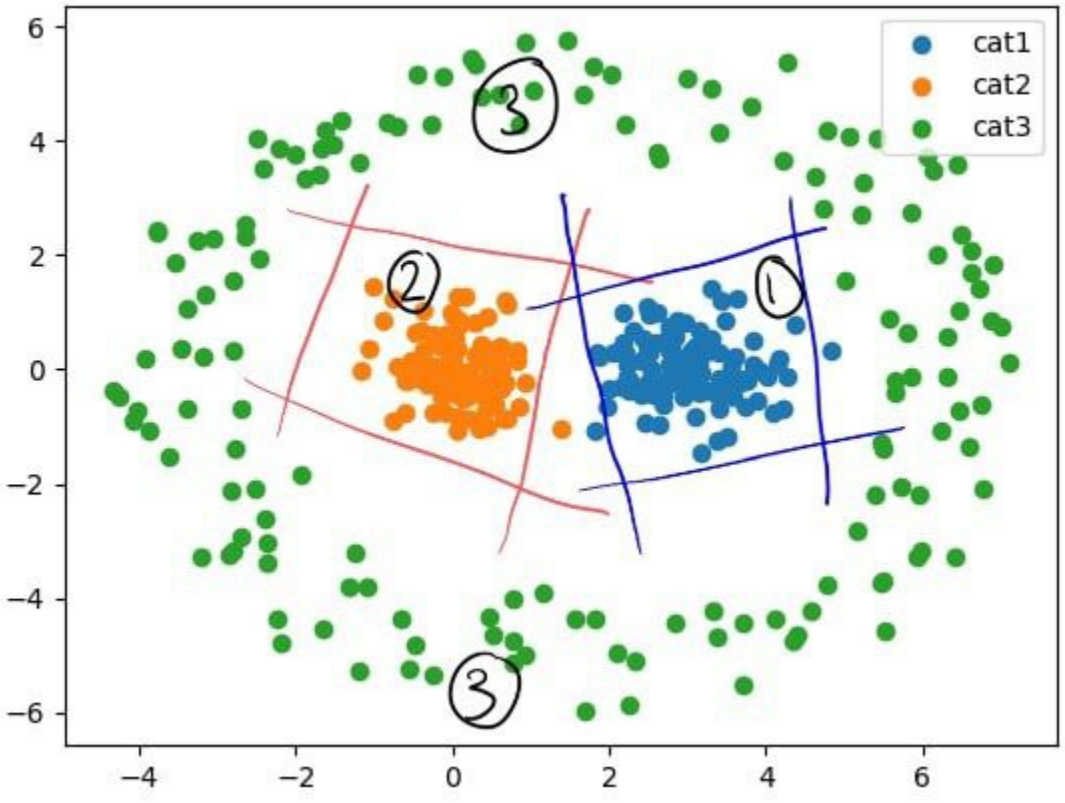
سپس نقاط در 5 ضرب شده اند تا در دایره ای با شعاع ۵ قرار گیرند تا هنگامی که با عددی رندوم بین بازه ۱ و ۱- جمع شوند ring محدود به شعاع ۴ و ۶ شود. سپس برای این که داده ها بر روی نقطه x=1.5 قرار گیرند بعد x آن ها با مقدار 1.5 جمع شده است.



شکل ۴-۱

الف)

مطابق با شکل ۲-۴ میتوان شبکه‌ی madalineای طراحی کرد که دو convex hyper polygon زیر را یاد بگیرد.



شکل ۲-۴

برای این کار شبکه ای طراحی میکنیم در لایه اول نهان 8 نورون داشته باشد تا بتواند ۸ معادله خط شکل ۲-۴ را یادبگیرد.

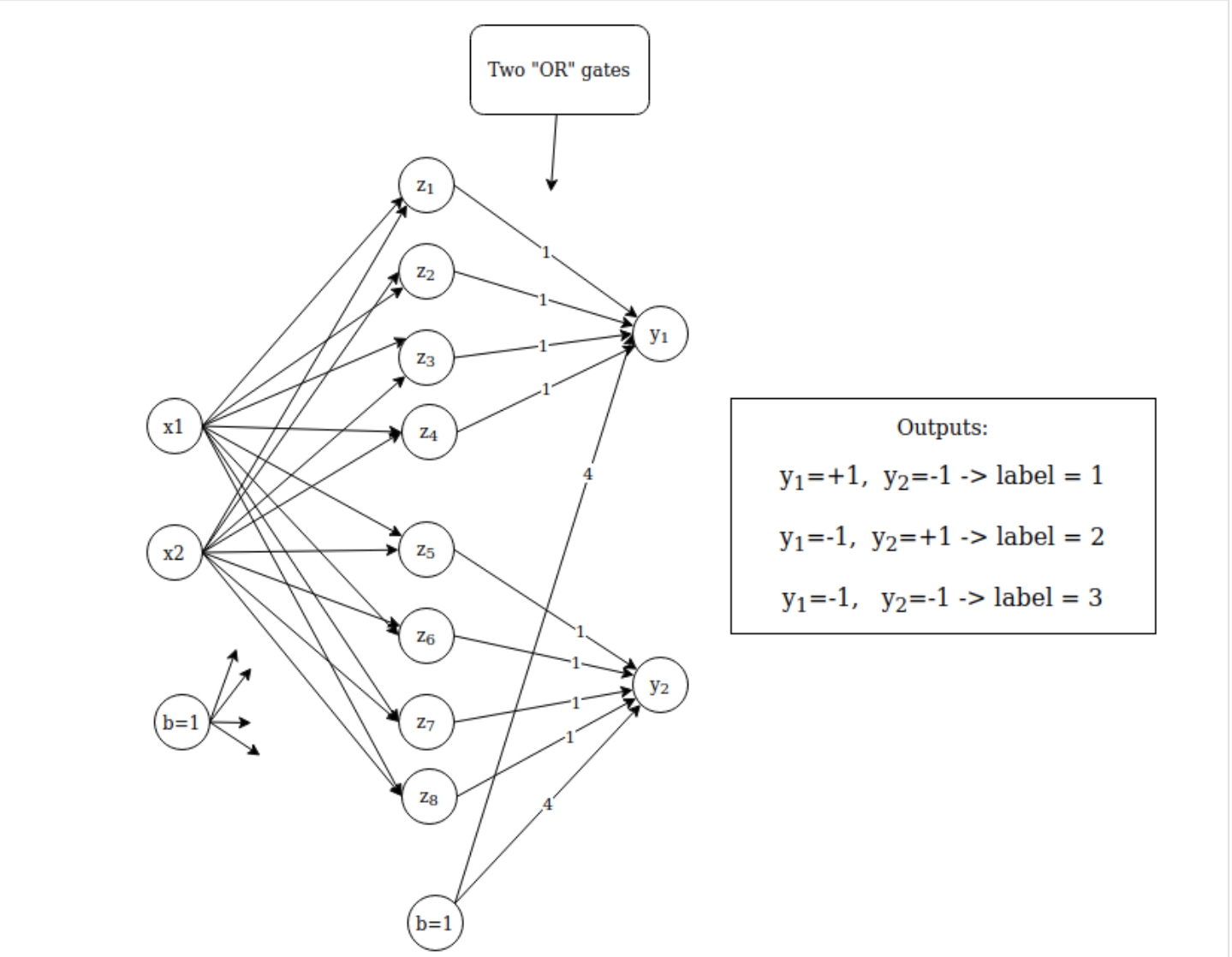
سپس در لایه بعدی نیاز به دو گیت OR داریم تا هر کدام مشخص کننده وجود داده ها داخل یا بیرون از polygon باشد.

در نتیجه در لایه آخر دو خروجی y1 و y2 خواهیم داشت که مطابق با جدول ۴-۱ ترکیب y1 و y2 به صورت binary سه دسته اول تا سوم را مشخص خواهد کرد.

y1	y2	caetgory
1	-1	1
-1	1	2
-1	-1	3

جدول ۴-۱

معماری شبکه مطابق با شکل ۴-۳ خواهد بود.



شکل ۴-۳

دو گیت OR لایه دوم به صورت Z1 + Z2 + Z3 + Z4 + 4 = 0 مشخص شده اند تا زمانی که همه‌ی Z1 تا Z4 منفی باشند خروجی ۱- شود و در غیر این صورت خروجی ۱+ شود.

بنابراین اگر شبکه بتواند وزن ها درست را یادبگیرد به ازای داده های ورودی خروجی متناسب با y1 و y2 مطابق با جدول ۱-۴ باید تولید شود.

برای آموزش شبکه از الگوریتم MR1 استفاده شده است.

۱- ابتدا خروجی لایه اول و دوم مطابق با زیر به ازای داده s محاسبه میشود.

```
z_in = np.dot(s, W1.T) + b1.T
z = f1_active(z_in)

y_in = np.dot(z, W2.T) + b2.T
y = f2_active(y_in)
```

توابع activation نیز به این گونه عمل میکنند که اگر ورودی بزرگتر صفر بود خروجی ۱+ و در غیر این صورت خروجی ۱- میدهند.

۲- برای به روز رساتی وزن های لایه اول مطابق با زیر عمل میکنیم.

طبق الگوریتم MR1 دو حالت برای به روزرسانی داریم.

A*: اگر خروجی ۱- بود و برچسب ۱+ بود باید وزن های نورون Zای که به صفر نزدیک تر است را به صورت زیر به روز رسانی میکنیم.

```
W = W + alpha * (1 - z_in) * s
```

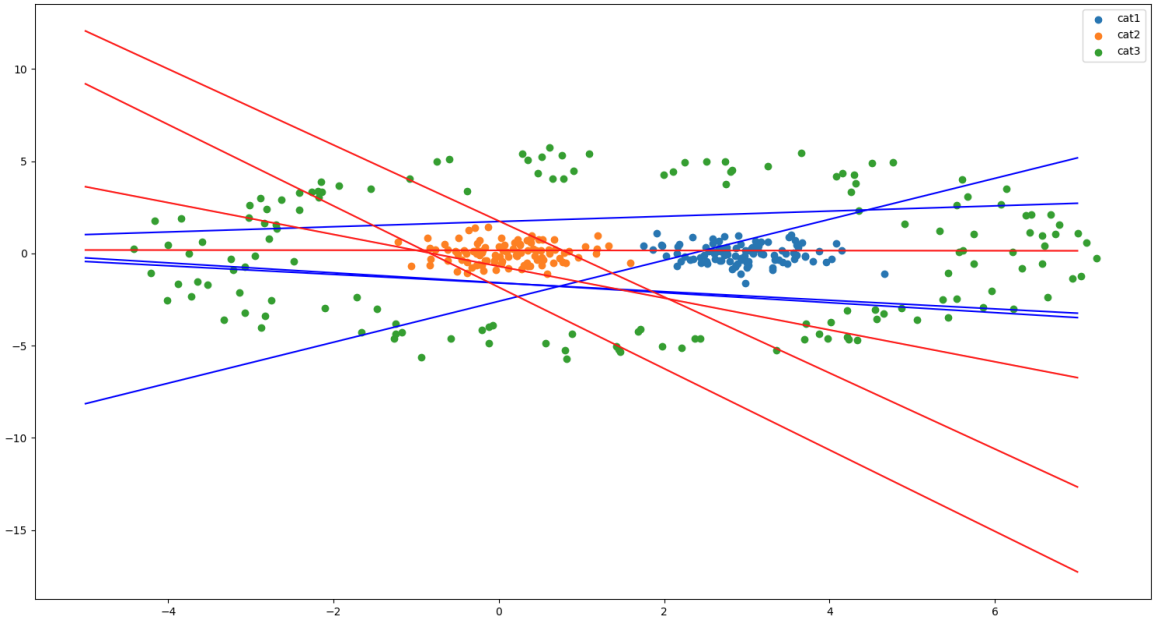
B*: اگر خروجی ۱+ بود و برچسب ۱- بود باید وزن های نورون Zای که Z_in آن ها مثبت است را به صورت زیر به روز رسانی میکنیم.

```
W = W + alpha * (-1 - z_in) * s
```

اما در این معماری دو خروجی y1 و y2 داریم اما میتوان به روز رسانی را برای خروجی های y1 و y2 مطابق با دو حالت A* و B* به روز رسانی کرد به گونه ای که مثلا برای دسته داده اول خروجی مورد انتظار y1=1 و y2=-1 است بنابراین اگر هر کدام از y1 و y2 مقدار متفاوتی خروجی دادند وزن ها را مطابق با A* و B* به روز رسانی میکنیم.

یا مثلا اگر داده مربوط به دسته سوم باشد خروجی مورد انتظار y1=-1 و y2=-1 است بنابراین اگر خروجی شبکه غیر از این دو خروجی بود پس باید وزن های نورون های z1 تا z8 مطابق با B* به روز رسانی شوند.

پیاده سازی دقیقا مطابق با الگوریتم بالا انجام شد و نتایج خطوط جدا ساز مطابق با شکل ۴-۴ به دست آمد. پارامترها learning_rate برابر 0.1 در نظر گرفته شد.



شکل ۴-۴

همان طور که مشاهده میشود آموزش به خوبی انجام نشده است و شبکه توانایی پیدا کردن چهار معادله خط درست را ندارد.

متاسفانه زمان زیادی برای حل این مسئله گذاشتم و به نظرم الگوریتم دقیقا مطابق با MR1 پیاده سازی شده است و دلیل عدم توانایی الگوریتم در پیدا کردن خطوط جدا ساز درست احتمالا به نحوه به روز رسانی وزن های شبکه برمیگردد.
به خاطر این که الگوریتم MR1 در کتاب تنها برای گیت های OR روابط به روز رسانی را مشخص کرده است و طبق این روابط وزن های شبکه به سمتی میروند که خروجی های شبکه را هر چه سریع تر 1- یا مثبت یک کنند و در هر مرحله کل وزن های شبکه به روز رسانی نمی شوند که این باعث

میشود دسته ۱ و ۲ که نیاز به یک خروجی ۱+ و یک خروجی ۱- دارند نتواند به روز رسانی های درست را انجام دهند و مقدار هزینه به خوبی نمیتواند کاهش پیدا کند.

بنابراین طبق نتیجه گیری من این الگوریتم نمیتواند خطوط جدا ساز را برای این حالت به درستی پیدا کند و شاید اگر گیت AND وجود داشت این کار بهتر میتوانست انجام شود گرچه الگوریتم MR1 برای گیت AND در کتاب آورده نشده است.

ب)

الگوریتم به ازای پارامترهای learning rate مختلف تست شد که در نتیجه خروجی اثر چندانی نداشت اما در آموزش پارامترهای نزدیک صفر باعث تغییرات کمتر وزن های W و پارامترهای نزدیک به 1 نیز باعث تغییر سریع وزن ها میشدند. هم چنین به ازای اعداد بزرگتر مساوی 1 وزن ها به بی نهایت میل میکردند که به خاطر انجام ضرب برای به روز رسانی وزن ها منطقی است.