به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

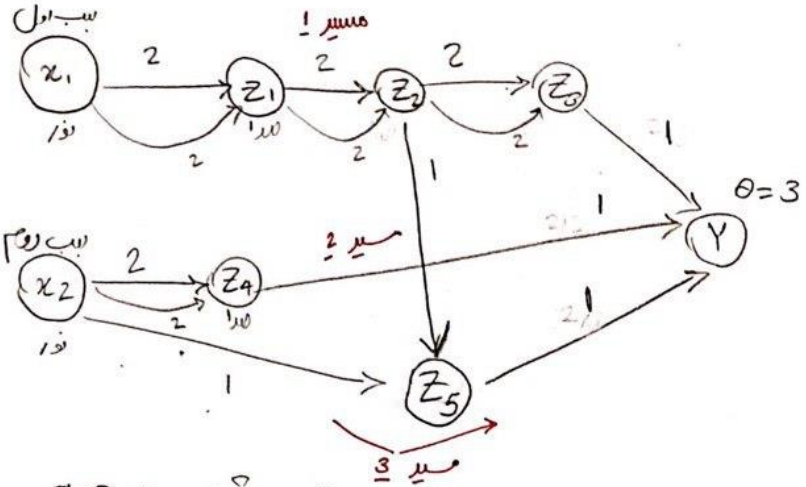
تمرین سری 1

نام و نام خانوادگی	مهسا مسعود
شماره دانشجویی	810196635
تاریخ ارسال گزارش	00/01/13

فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

سوال 1 – McCulloch-Pitts

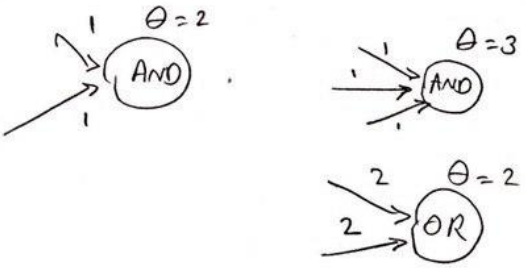
سوال ۱ – الف)



ب و ج)

مقادیر x_1 و x_2 نشان‌دهنده‌ی سیب‌ها هستند. در ابتدا سیب ۱ (x_1) منفجر شده و در STEP ۲ زمانی بعد z_2 یعنی سیب دوم منفجر می‌شود. مقدار threshold برای فعال شدن خروجی ۲، $\theta=3$ می‌باشد و از ۳ مسیر باید همزمان رخ دهد. مقادیر باینری هم State‌ها ۱ ± در نظر گرفته شده.

گیت‌های استفاده شده:



$$(Z_4)\ AND\ (Z_5)\ AND\ (Z_3) = Y$$

$Z_4 \rightarrow x_2$

$Z_5 \rightarrow (z_2\ AND\ x_2)$

$Z_3 \rightarrow z_2$

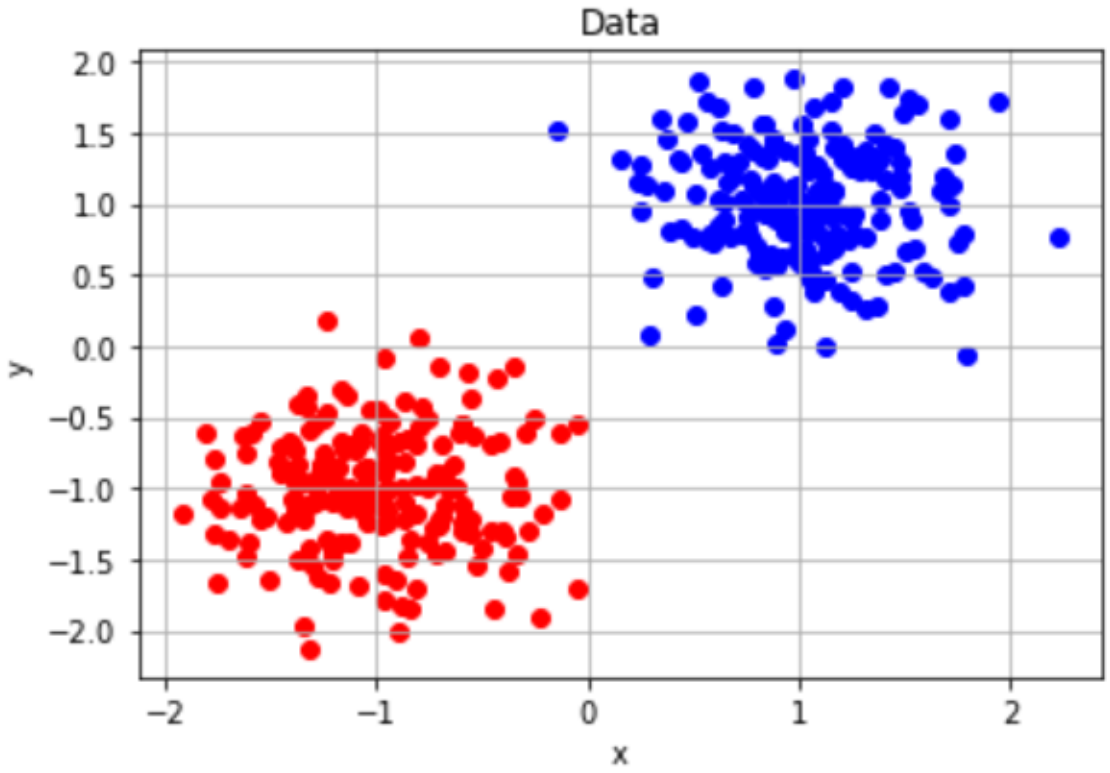
مسیر اول برای منفجر شدن نورون‌های سیب اول است و ۲ است زمانی بعد z_2 فعال می‌شود که همزمان با فعال شدن سیب دوم و دیرتر از آن است. پس در STEP بعد که همزمان با آخرین مسیر سیب دوم است به خروجی می‌رسیم.

z_5 برای این است که حتماً سیب اول در ابتدا detect شده باشد از مسیر ۳ و مسیر ۲ برای لزوم شنیده شدن صدای سیب دوم است.

سوال ۲ – Perceptron

الف)

ابتده داده ورودی را خوانده و سپس با توجه به نوع هر کلاس، داده ها را با استفاده از تابع scatter از کتابخانه matplotlib رسم میکنیم.



شکل 2-1 دیتا

کلاس 1 با رنگ قرمز و کلاس 1- با رنگ آبی مشخص شده اند.

ب)

Updating rules:

$$net = \sum_{i=1}^{n} w_i x_i + b$$

$$w_i = w_i + \alpha x_i t$$

$$b_i = b_i + \alpha t$$

با انتخاب threshhold به مقدار 1 و learning_rate برابر با 0.01 ، الگوریتم perceptron را به شکل زیر پیاده سازی میکنیم:

```
#activation function
def sign(x,theta):
    if (x > theta):
        return 1
    elif (x < -theta):
        return -1
    else:
        return 0
```

```
learning_rate = 0.01
#learning_rate = 0.0001

# generating the weights & the bias
w1 = 0
w2 = 0
b = 0
th = 1

for epoch in range(1000) :

    for i in range(len(x_train)) :
        x = x_train[i]
        net = x[0] * w1 + x[1] * w2 + b

        y = sign(net,th)

        target = x[2]

        # Updating the weights & the bias
        if target != y :

            w1 = w1 + learning_rate * (target) * x[0]
            w2 = w2 + learning_rate * (target) * x[1]
            b = b + learning_rate * (target)
```

بعد از اعمال الگوریتم بالا روی داده ی train مقادیر w1 ,w2,b را به صورت زیر استخراج میکنیم:

```
W1 = -1.8894557397506908
```

```
W2 = -2.1435332564640066
b = -0.3000000000000001
```

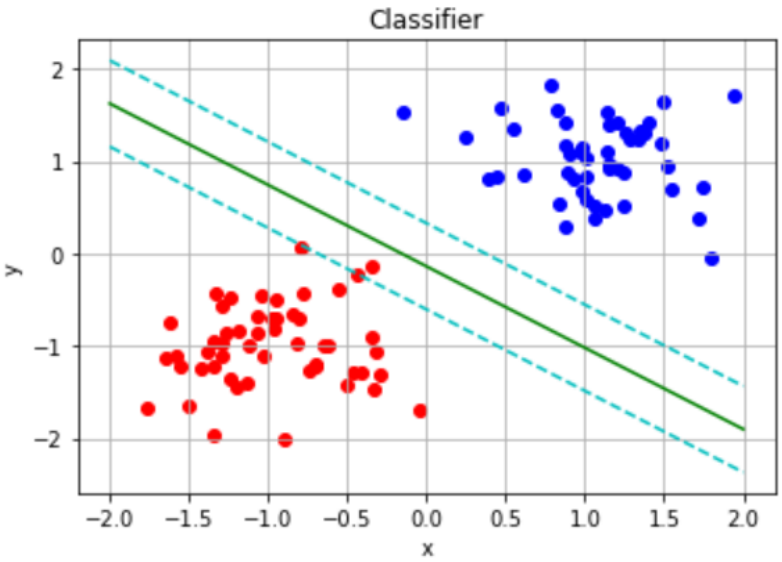
ج)

Theta = 1:

$$w_1 = w_1 + \alpha \times target \times x1$$
$$w_2 = w_2 + \alpha \times target \times x2$$
$$\mathrm{b} = b + \alpha \times target$$
$$net = w_1 \times x1 + w_2 \times x2 + b$$

برای تست شبکه یک تابع guess تعریف میکنیم و با توجه به مقادیر بایاس و وزن های بدست آمده ، طبق فرمول های بالا کلاس مورد نظر را حدس خواهیم زد.

با تتای 1 به دقت 98 درصد دست میابیم.



شکل 2-2 طبقه بند با تتا = 1

د)

Theta = 100:

بعد از اعمال الگوریتم بالا روی داده ی train مقادیر w1 ,w2,b را به صورت زیر استخراج میکنیم:

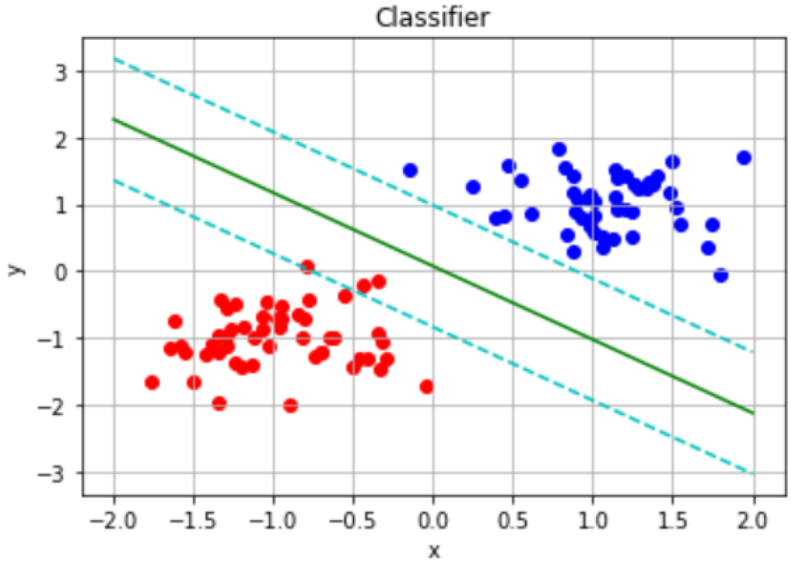
```
W1 = -120.59718441540234
```

```
W2 = -109.9895695159157
b = 8.109999999999872
```

$$w_1 = w_1 + \alpha \times target \times x1$$
$$w_2 = w_2 + \alpha \times target \times x2$$
$$\mathrm{b} = b + \alpha \times target$$
$$net = w_1 \times x1 + w_2 \times x2 + b$$

برای تست شبکه یک تابع guess تعریف میکنیم و با توجه به مقادیر بایاس و وزن های بدست آمده ، طبق فرمول های بالا کلاس مورد نظر را حدس خواهیم زد.

با تتای 100 به دقت 97 درصد دست میابیم.



شکل 2-3 طبقه بند با تتا = 100

$$\sum_{i=1}^{n} w_i x_i = threshold$$
$$\sum_{i=1}^{n} w_i x_i = b \rightarrow b = -threshold$$

مقدار آستانه خود را در قالب بایاس نشان می دهد و بیش از حد بزرگ انتخاب کردن آن می تواند از دقت الگوریتم بکاهد.

با انتخاب آستانه ی بزرگتر مارجینی که برای خط های اطراف در نظر گرفته شده بیشتر میشود و آزادی عمل افزایش می یابد ،یعنی در معادله قسمت بدون تصمیم (non-decision) بزرگتر خواهد شد :

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

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

سوال 3 – Adaline

الف)

شباهت ها:

1- هر دو روش از کلسیفایر ها برای binary classification استفاده میکنند..
2- در هر دو روش یک مرز تصمیم مشخص خطی وجود دارد.

تفاوت ها :

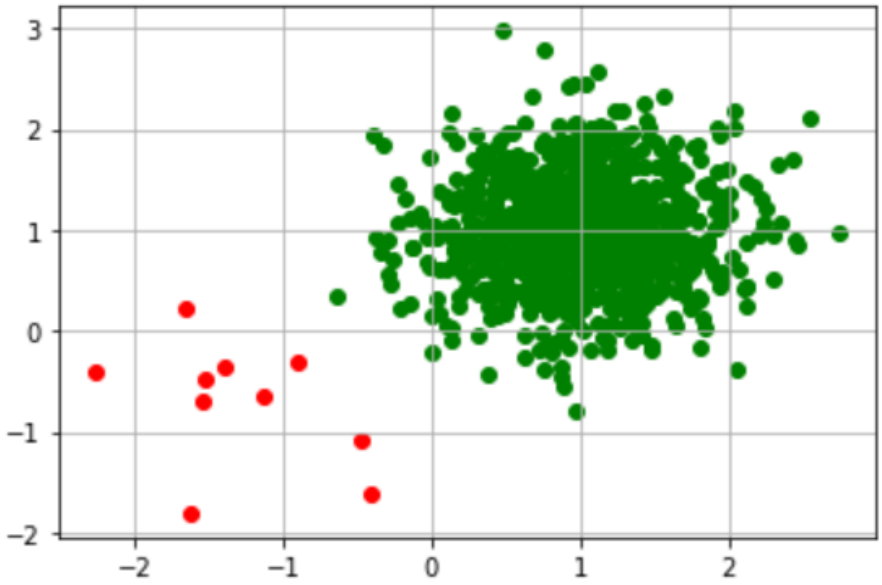
1- در قاعده پرسپترون این تضمین وجود دارد که ما با هر p روی ورودی ها شروع کنیم و با بردار وزن ها w (دلخواه) داشته باشیم: $t(p) = f(x(p)w)$ ، قطعا این روش ما را به یک بردار وزنی صحیح که پاسخ طبقه بندی شده صحیح را خروجی میدهد، می رساند. در حالیکه در روش adaline و قاعده دلتا، تضمینی برای همگرایی وجود ندارد و مشخص نیست به جواب بهینه می رسیم یا خیر(البته اگر از tanh به عنوان activation function استفاده کنیم این روش نیز جواب میدهد.)
2- در مدل پرسپترون برای به روز رسانی ضرایب از class label ها استفاده میکنیم و خطا ها را در هر epoch اصلاح میکنیم ولی روش ادلاین از طرق gradient-based و استفاده از continuous predicted values برای آپدیت کردن ضرایب استفاده میکند.

ب)

از فرمول $net = \sum_{i=1}^{n} w_i x_i + b$ استفاده می کنیم. و با استفاده از تابع h به صورت زیر، تتا را از این رابطه حذف میکنیم وآن را اعمال میکنیم.

$$h = f(net) = \begin{cases} +1 & \text{net } \geq 0 \\ -1 & \text{net } < 0 \end{cases}$$

داده های قسمت اول با توجه به تعداد و میانگین و واریانس ذکر شده تولید و در شکل زیر نمایش داده می شوند:



شکل 3-1 دیتا ی اول

برای به روز رسانی مقادیر وزن ها و بایاس از فرمول های زیر استفاده میشود:

$$w_i = w_i + \alpha(t_i - net)x_i$$

$$b_i = b_i + \alpha(t_i - net)$$

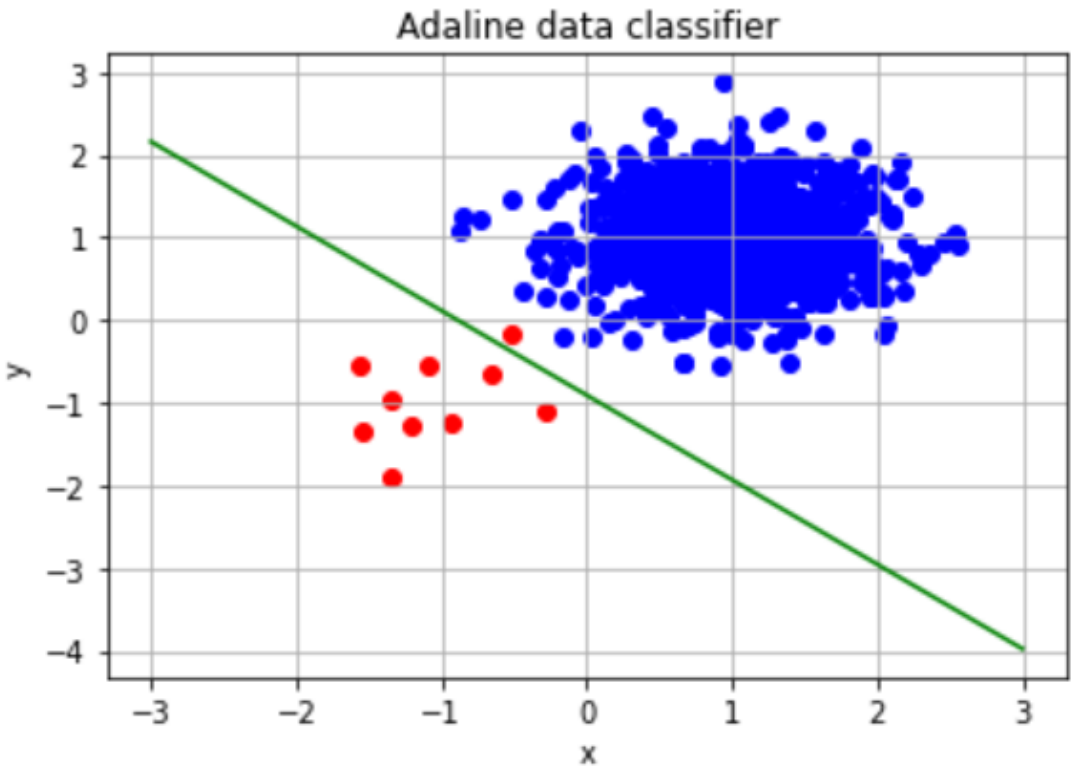
مقادیر w1,w2,b را با استفاده از تابع random از کتابخانه numpy به صورت رندوم انتخاب میکنیم.

همچنین مقدار learning rate را نیز برابر با 0.01 قرار میدهیم.

همچنین میدانیم که روش دلتا لزوما همگرا نخواهد شد به این دلیل که net به t نزدیک می شود و نه h.

بعد از گذشت 150 epoch به نتیجه زیر میرسیم:

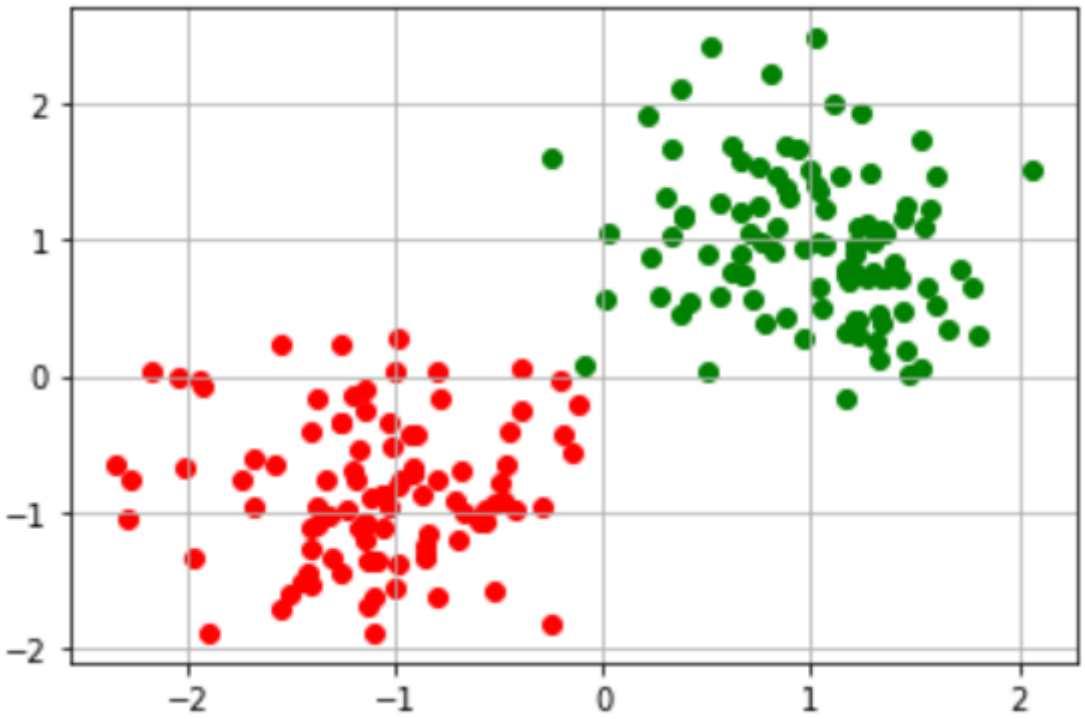
-شکل اول:



شکل 3-2 دیتای اول و طبقه بندی با adaline

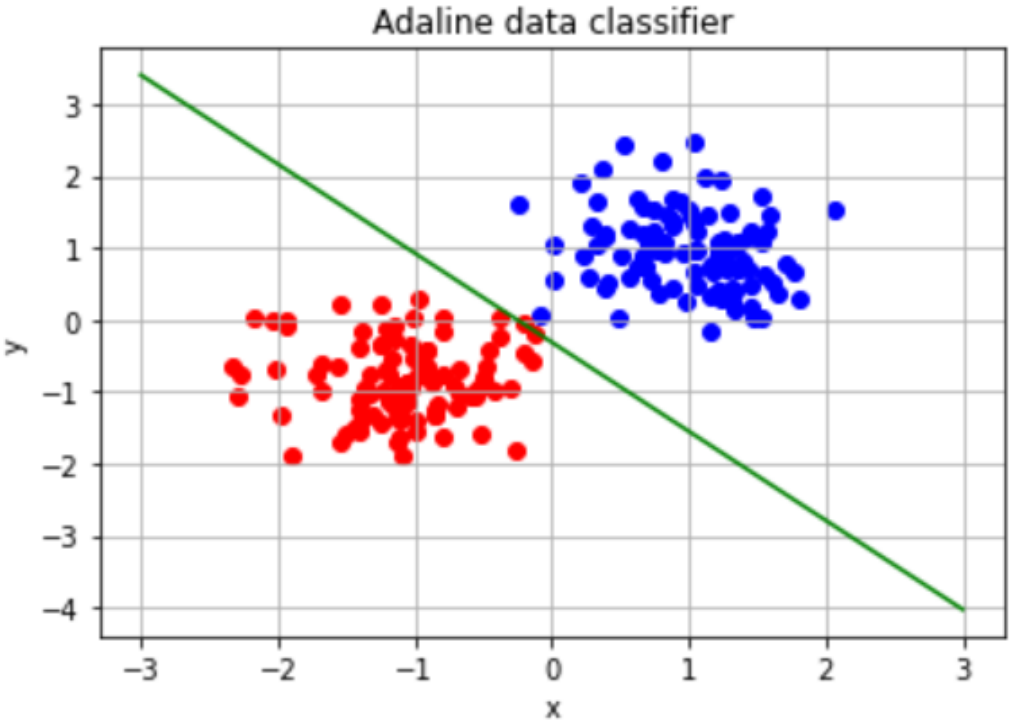
-شکل دوم:

نمایش داده ها در ابتدا:



شکل 3-3دیتای حالت دوم

داده ها به یکدیگر نزدیک تر هستند و population و نوع distribution آن ها یکسان است (بر خلاف شکل اول که تفاوت زیادی داشتند).



شکل 3-4 دیتای حالت دوم با طبقه بند adaline

ج)

روش دلتا تضمینی برای همگرایی نمیدهد وطبق تعریف الگوریتم دلتا ، از تفاضل net , t استفاده می شود و نه t-h و این به دلیل استفاده از اکتیویشن فانکشن sign بود حال برای رفع این مشکل می توان از تابع soft-sign مثل tanh که فرم نرم تری دارد استفاده کرد و تابع هزینه را به این صورت نوشت:

$$J_p(w,b) = 0.5 * (\, t_p - \tanh(y * \left(net(x_p, w, b)\right))^2$$

د)

با استفاده از روش های گرادیانی و اعمال آن ها به فرمول بالا می توان به مقادیر زیر دست یافت:

$$\delta W_i = W_i^+ - W_i^- = -\alpha \frac{dJ_p(w,b)}{dw_i} = \alpha(t - net)x_i$$

$$\delta b_i = b^+ - b^- = -\alpha \frac{dJ_p(w,b)}{db} = \alpha(t - net)$$

همچنین باید نرخ یادگیری از بزرگترین مقدار ویژه ماتریس correlation کوچکتر باشد.

سوال 4 – Madaline

الف و ب)

در این سوال با استفاده از الگوریتم MRI برای مدل کردن داده ها در سیستم madaline استفاده میکنیم.

نرخ یادگیری = 0.001

Step های این الگوریتم از کتاب فاست برداشته شده است.

شبکه 4 نورون :

وزن های اولیه :

b1 = -1

b2 = 1.5

b3 = -0.5

b4 = 0.75

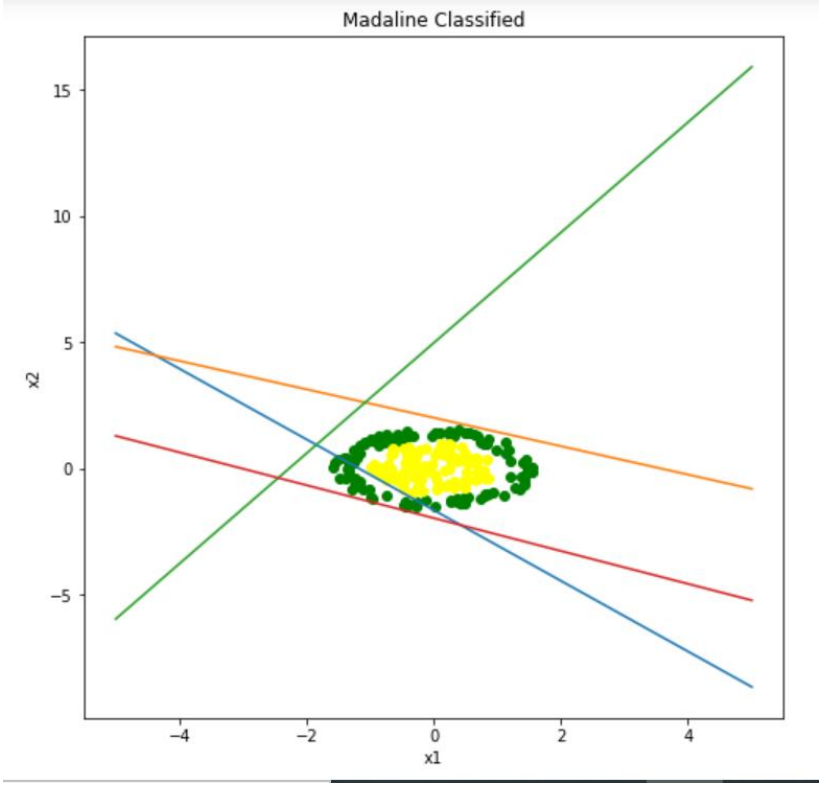
w1 = [-0.8,0.3]

w2 = [1,0.425]

w3 = [0.25,0.5]

w4 = [0.5,-0.5]

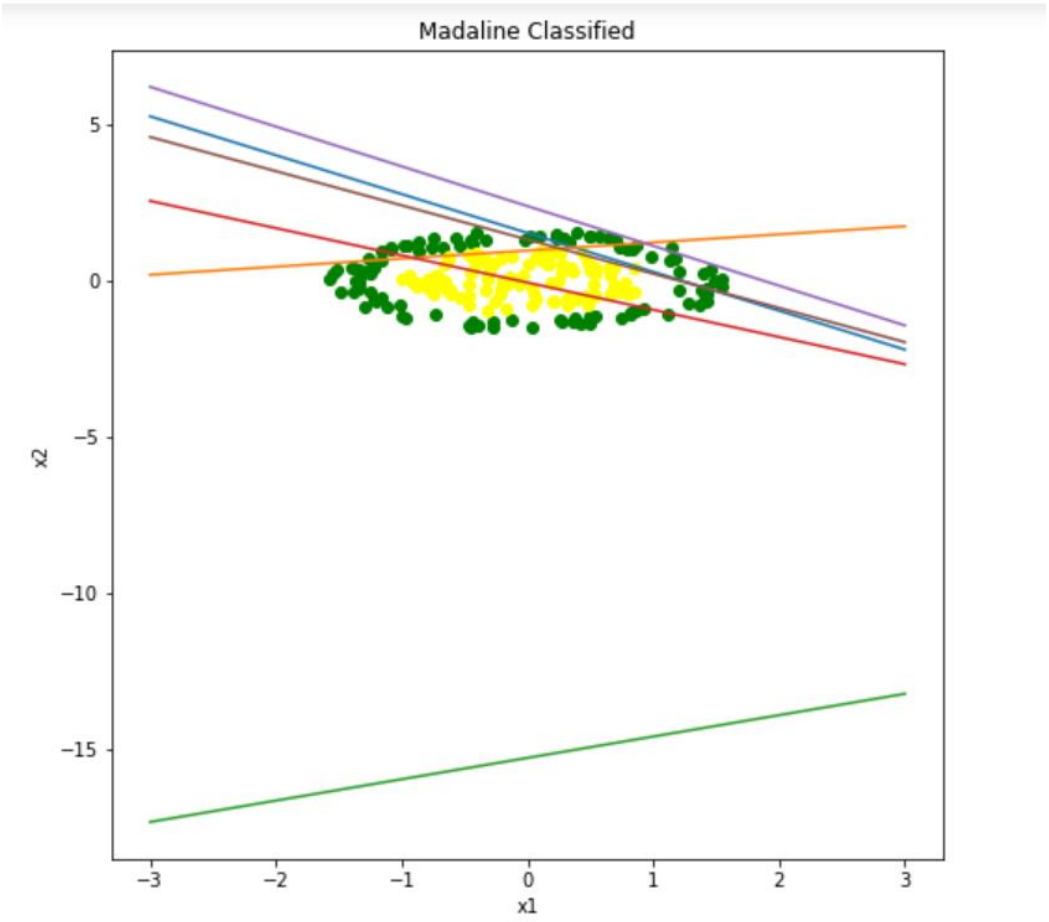
در این شبکه به حدود 500 ایپاک برای رسیدن به دقت مطلوب نیاز داریم.



شکل 4-1 madaline classified 4 neurons result

شبکه 6 نورون:

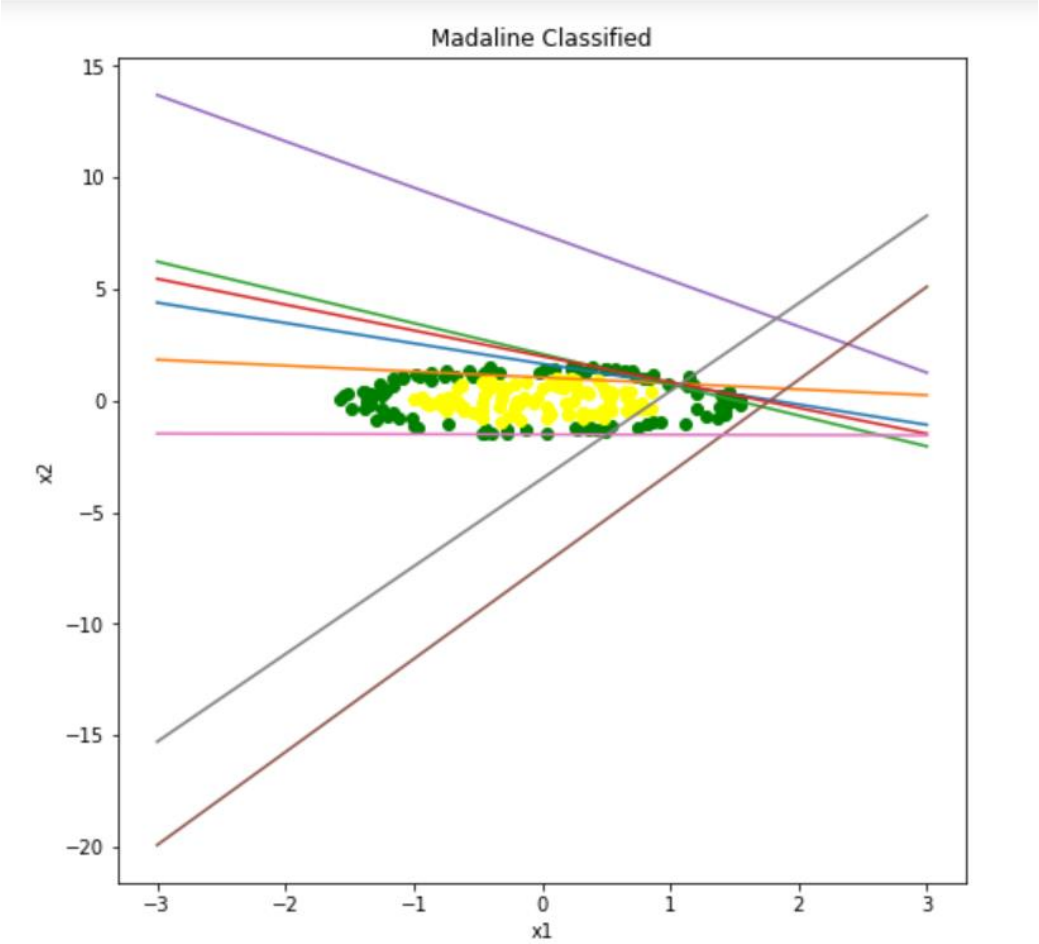
400 epochs



شکل 4-2 madaline classified 6 neurons result

شبکه 8 نورون:

250 epochs



شكل 4 -3 madaline classified 8 neurons result

ج)مشاهده می شود که به دلیل بیشتر شدن تعداد لایه ها ، طبقه بندی به صورت جزیی تری انجام می شود و تعداد خطوط بالا می رود که نشان از دقیق تر شدن طبقه بندی در تعداد ایپاک کمتر میشود.