به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سری اول

نام و نام خانوادگی	مریم ریاضی
شماره دانشجویی	810197518
تاریخ ارسال گزارش	1400/12/20

فهرست گزارش سوالات

سوال 1 – Mcculloch Pitts 1

سوال ۲ – Adaline 7

سوال 3 – Madaline 11

سوال 4 – Perception 17

سوال ۱ – Mcculloch Pitts

برای طراحی شبکه عصبی سوال اول، باید در ابتدا رابطه منطقی هرکدام از بیت های خروجی را به دست آوریم. در ادامه با داشتن شبکه های عصبی گیت هایی مانند and, or, xor شبکه عصبی Full Adder دو بیتی به راحتی به دست می آید.

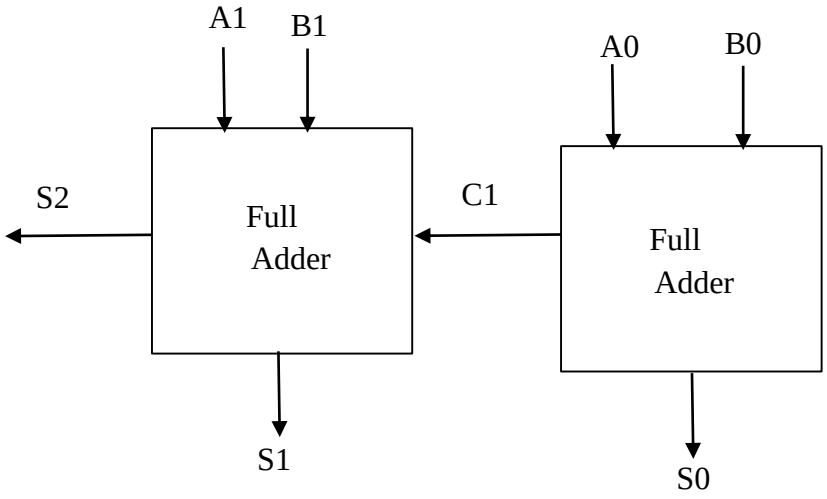
اگر ورودی های دو بیتی به صورت A1A0, B1B0 و خروجی به صورت S2S1S0 باشد، روابط منطقی هر کدام از بیت های خروجی به شرح زیر است:

S0 = A0 xor B0

S1 = C1 xor A1 xor B1

S2 = A1B1 + A1C1 + B1C1

یک Full Adder دو بیتی به صورت زیر کار می کند:

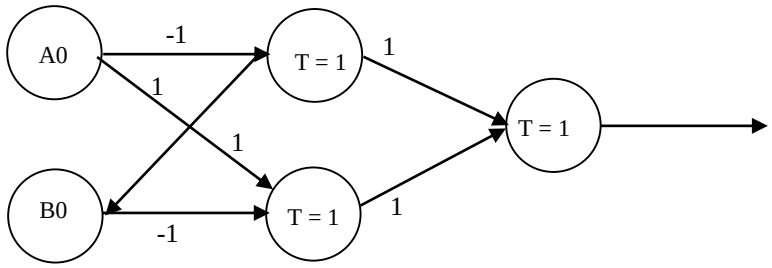


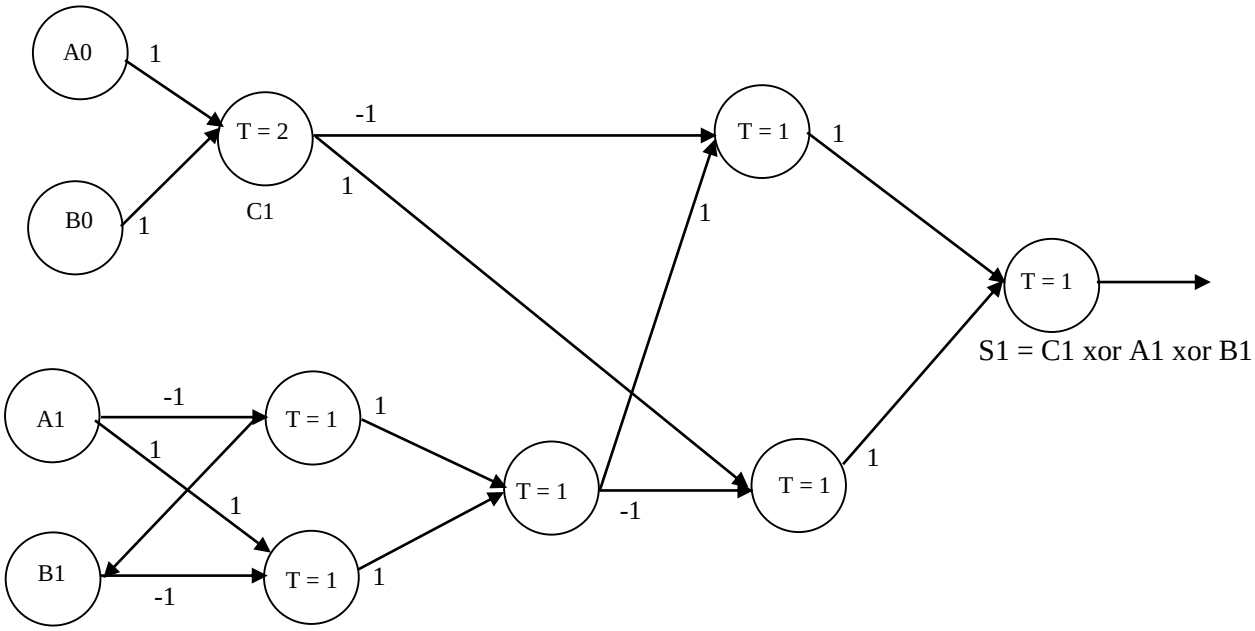
جدول درستی ورودی و خروجی های یک Full Adder دو بیتی به صورت زیر است:

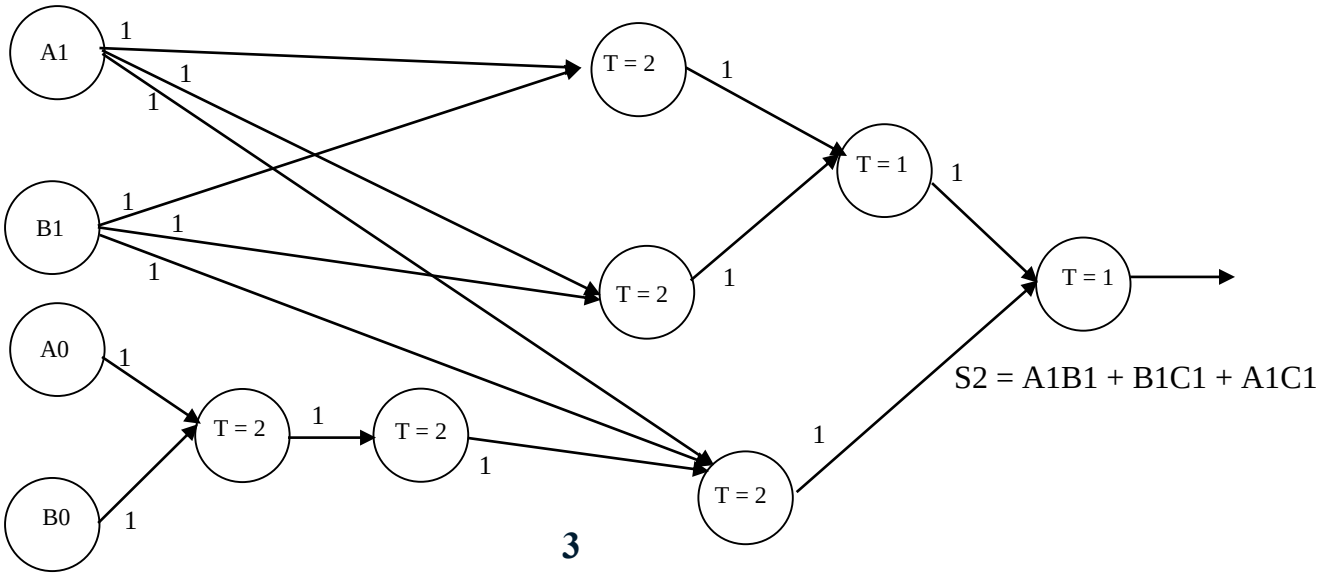
جدول شماره1: ورودی و خروجی های Full Adder دو بیتی

A0	A1	B0	B1	S0	S1	S2
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

شبکه عصبی هر کدام از بیت های خروجی به صورت شکل زیر است:







```
input1 = list(map(int, input().strip().split()))[:2]
A1 = input1[0]
A0 = input1[1]
input2 = list(map(int, input().strip().split()))[:2]
B1 = input2[0]
B0 = input2[1]
```

شکل 1:کد ورودی گرفتن از کاربر

```
def AND(a1, a2):
    weights = [1 , 1]
    T = 2
    net = a1*weights[0] + a2*weights[1]
    if net >= T:
        return 1
    else:
        return 0
```

شکل 2: کد شبکه عصبی گیت **AND**

```
def OR(a1, a2):
    weights = [1 , 1]
    T = 1
    net = a1*weights[0] + a2*weights[1]
    if net >= T:
        return 1
    else:
        return 0
```

شکل 3 :کد شبکه عصبی گیت**OR**

```python
def XOR(a1, a2):
    weights1 = [-1 , 1]
    T = 1
    net1 = a1*weights1[0] + a2*weights1[1]
    net2 = a2*weights1[0] + a1*weights1[1]
    y1 = []
    y2 = []

    if net1 >= T:
        y1 = 1
    else:
        y1 = 0
    if net2 >= T:
        y2 = 1
    else:
        y2 = 0
    return OR(y1, y2)
```

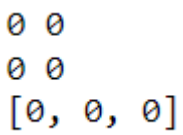
شکل **4**: کد گیت **XOR**

```python
S0 = XOR(A0, B0)
C1 = AND(A0, B0)
S1 = XOR(XOR(C1, A1), B1)
S2 = OR(OR(AND(A1, B1), AND(A1, C1)), AND(B1, C1))
a = [S2, S1, S0]
```

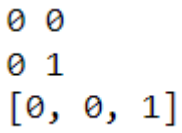
شکل 5 :کد محاسبه هر بیت خروجی

همانطور که در شکل های 1 تا 5 می بینیم، کد شبکه عصبی هرکدام از گیت های منطقی با توجه به اسلایدهای درس و ضرایب به دست آمده برای گیت xor نوشته شده است. در نهایت نیز هر کدام از بیت های خروجی با توجه به رابطه منطقی نشان داده شده در ابتدا، محاسبه شده اند.

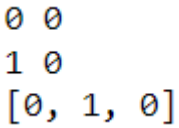
در ادامه همه ورودی های محتمل را مانند جدول 1 امتحان می کنیم تا از صحت خروجی مطمئن شویم:



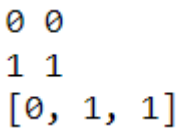
شکل 6: امتحان خروجی 1



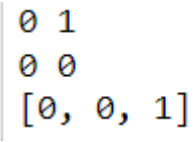
شکل 7: امتحان خروجی 2



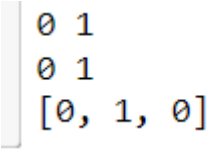
شکل 8: امتحان خروجی 3



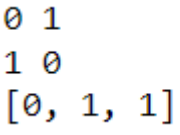
شکل9: امتحان خروجی 4



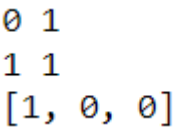
شکل 10: امتحان خروجی 5



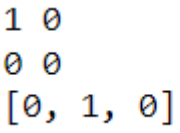
شکل 11: امتحان خروجی 6



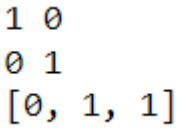
شکل 12: امتحان خروجی 7



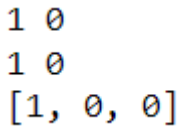
شکل 13: امتحان خروجی 8



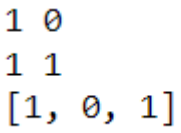
شکل 14: امتحان خروجی 9



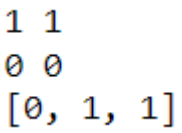
شکل 15: امتحان خروجی 10



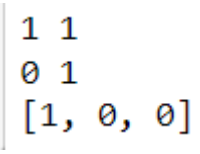
شکل 16: امتحان خروجی 11



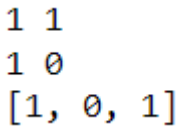
شکل 17: امتحان خروجی 12



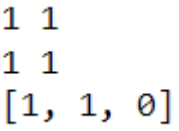
شکل 18: امتحان خروجی 13



شکل 19: امتحان خروجی 14



شکل 20: امتحان خروجی 15



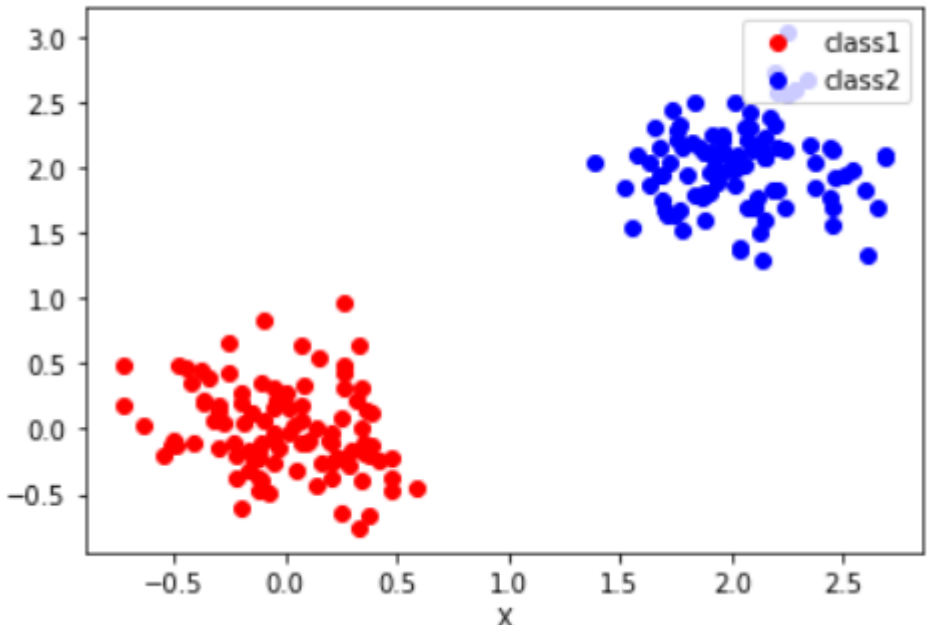
شکل 21: امتحان خروجی 16

همانطور که در شکل های 7 تا 21 مشاهده می شود، تمام خروجی ها مطابق با جدول 1 است.

سوال ۲ – Adal i ne

الف)

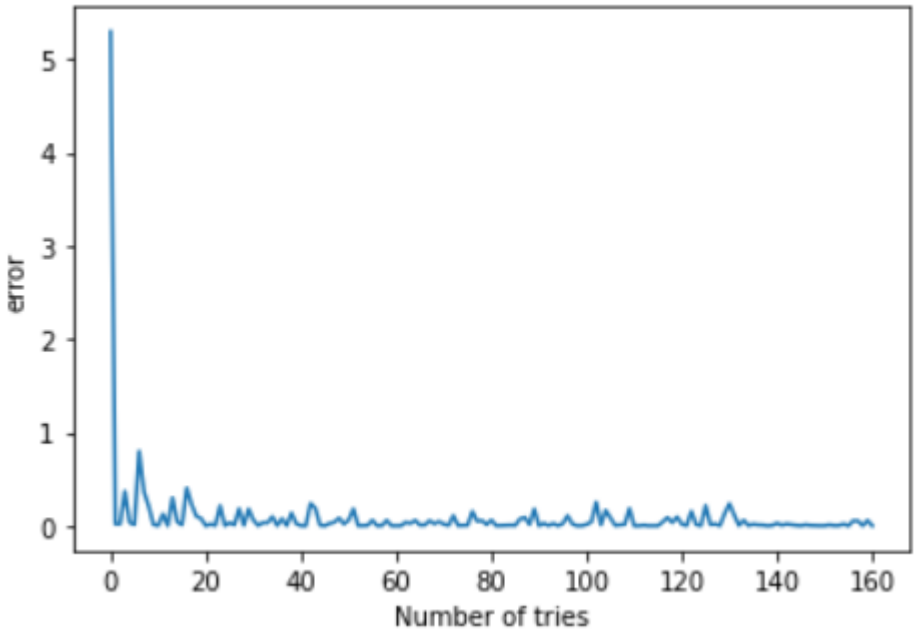
ابتدا دو دسته داده به فرض های داده شده در سوال تولید کردم که نمودار پراکندگی آن ها به صورت شکل زیر است.



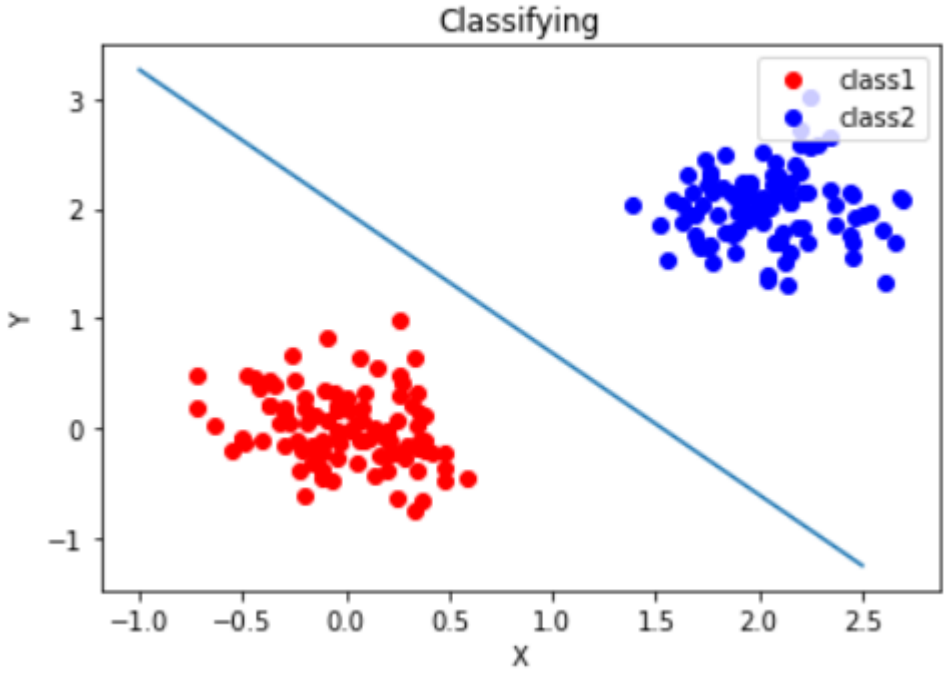
شکل22: نمودار پراکندگی داده های رندوم

ب)

برای آموزش شبکه عصبی به روش Adaline در ابتدا دو وزن به صورت رندوم مقداردهی کردم. مقدار b را نیز به صورت عددی رندوم انتخاب کردم. در ادامه به صورت دلخواه برای هر کدام از دسته داده ها تارگتی انتخاب کردم. در این سوال برای داده های کلاس 1، تارگت را برابر 1 و برای داده های کلاس 2، تارگت را برابر -1 قرار دادم. پس از آن، داده ها را به صورت زوج هایی کنار هم و کنار تارگت خودشان قرار دادم. به این صورت لیستی از تمام داده ها و تارگت هرکدام خواهیم داشت. در مرحله بعد برای محاسبه net و آپدیت کردن هرکدام از وزن ها، زوج ها به صورت رندوم از این لیست انتخاب خواهند شد. مراحل آپدیت کردن تا جایی ادامه پیدا می کند که مقدار خطا از threshold کمتر شود. مقدار threshold نیز با آزمون و خطا به دست آمد. در نهایت نمودار خطا و خطی که داده ها را از هم جدا می کند به صورت دو شکل زیر است.



شکل 23: نمودار مقدار خطا در هر دفعه تکرار

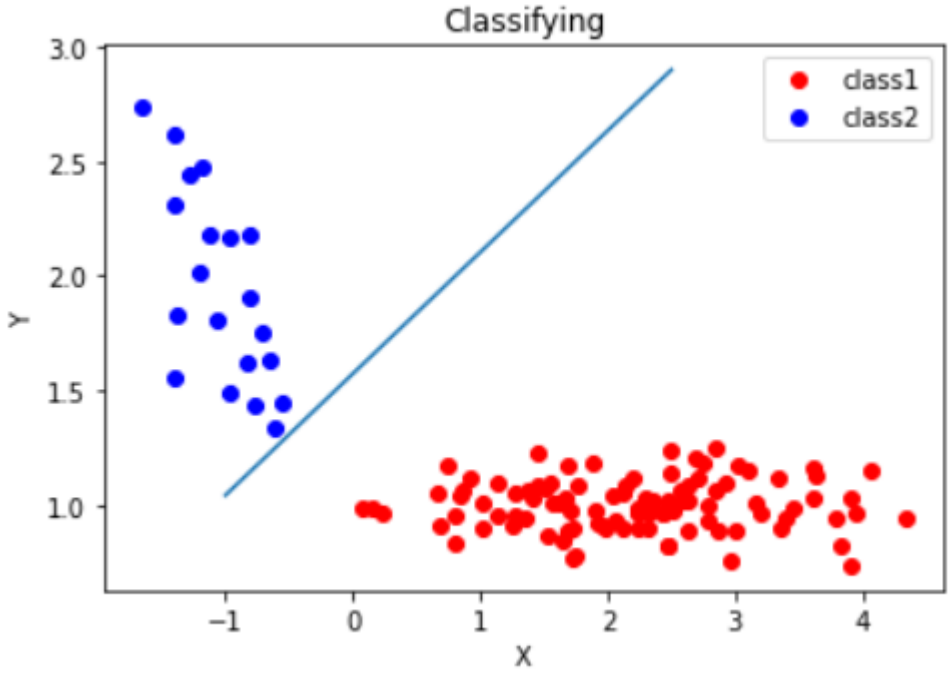


شکل 24: نمودار داده ها و خط جداکننده

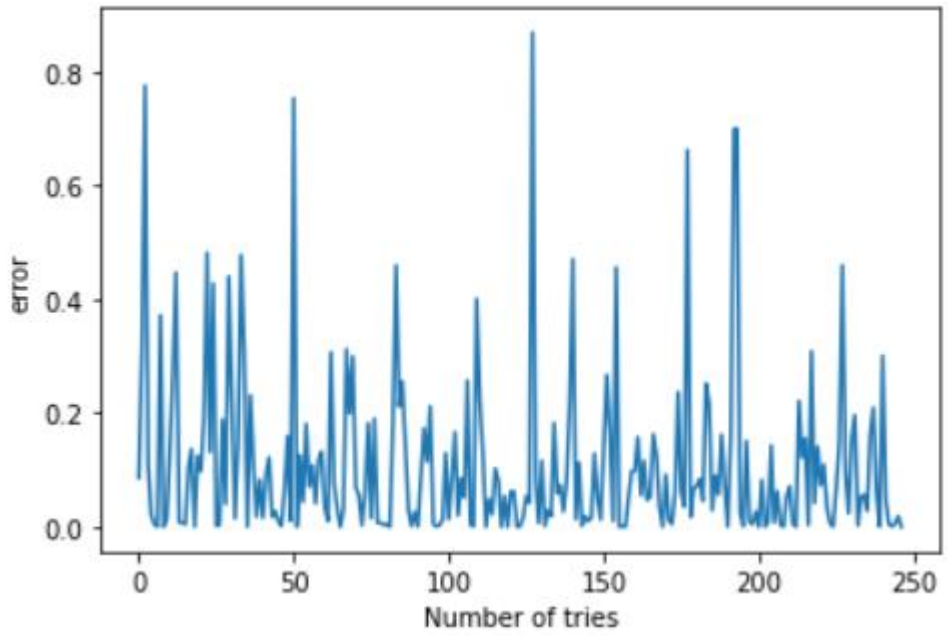
همانطور که در درس آموختیم، الگوریتم Adaline در جداسازی داده هایی که نسبت به هم متقارن هستند، بسیار خوب عمل می کند. در این قسمت نیز با توجه به شکل 22 می توان دید که نوعی تقارن در داده ها وجود دارد. از طرف دیگر در هر دسته از داده ها، میانگین و انحراف معیار x,y ها با هم برابر بود. این ویژگی باعث می شود که داده ها خیلی پراکنده از همدیگر نباشند و به همین دلیل جداسازی آن ها ساده تر می شود. تعداد برابر داده ها نیز ویژگی دیگری بود که به تقارن آن ها کمک می کرد.

ج)

در این قسمت نیز با الگوریتمی که در بخش ب به کار برده شد، خط جداکننده داده ها به دست آمد.



شکل 25: نمودار داده ها و خط جداکننده



شکل 26: نمودار خطا در هر دفعه تکرار

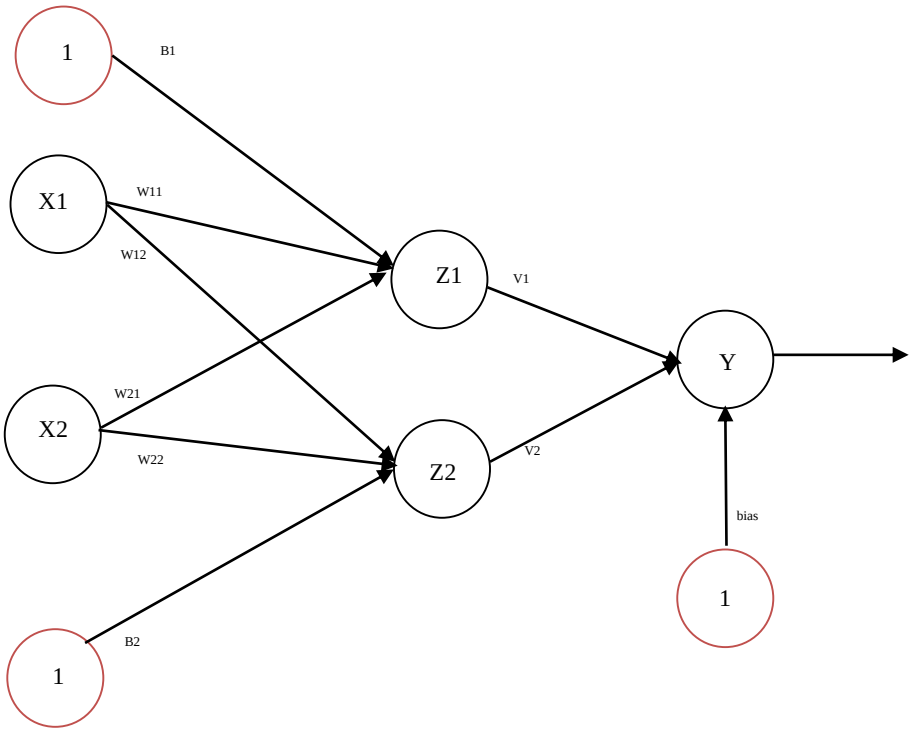
همانطور که در شکل های 25 و 26 مشاهده می شود، جداسازی در این دسته از داده ها مانند بخش ب دقیق و باکیفیت نیست. در بخش ب و در شکل 23 دیدیم که مقدار خطا در هر دفعه تکرار کم و کمتر شد؛ اما در این بخش در شکل 26 می بینیم که مقدار خطا نموداری نوسانی دارد و مدام کم یا زیاد می شود. علت این موضوع این است که اولا تعداد داده ها در دو دسته یکسان نیست. دومین علت نیز این است که در هر دسته از داده ها برخلاف قسمت ب، هر کدام از داده های x, y میانگین و انحراف معیار خود را دارند که با هم برابر نیست. این تفاوت باعث می شود که داده ها بسیار پراکنده شوند. همین پراکندگی و عدم تقارن باعث می شود که الگوریتم AdaLine نتواند به درستی به optimal point برسد.

سوال ۳ – Madaline

الف)

در این بخش الگوریتم MR1 را توضیح می دهم. نحوه عملکرد این الگوریتم مانند Adaline است با این تفاوت که بعد از نورون های Adaline یک لایه دیگر از نورون افزوده می شود که خروجی های لایه قبلی را باهم or می کند.

شکل شبکه عصبی این الگوریتم مانند زیر است.



در این الگوریتم تابع فعال سازی به شرح زیر است:

$$f = \begin{cases} 1 & net \geq 0 \\ -1 & net < 0 \end{cases}$$

نحوه کار این الگوریتم به شرح زیر است:

قدم صفر: در ابتدا باید مقادیر bias , V را به صورت رندوم اعداد کوچکی در نظر گرفت.

قدم اول: شرط توقف را تعریف کنید و تا وقتی این شرط برقرار نشده قدم های 2 تا 8 را انجام دهید.

قدم دوم: برای هر زوج ورودی و خروجی s, t قدم های 3 تا 7 را انجام دهید.

قدم سوم: $x_i = s_i$

قدم چهارم:

$Z\_in1 = b_1 + x_1w_{11} + x_2w_{21}$

$Z\_in2 = b_2 + x_1w_{12} + x_2w_{22}$

قدم پنجم:

$Z_1 = \text{activation function}(Z\_in1)$

$Z_2 = \text{activation function}(Z\_in2)$

قدم ششم:

$Y\_in = bias + v_1Z_1 + v_2Z_2$

$Y = \text{activation function}(Y\_in)$

قدم هفتم: مقدار Y باید با مقدار تارگت مقایسه شود. اگر یکی نبود باید وزن ها به روز شوند. اما این به روزرسانی به مقدار تارگت بستگی دارد.

اگر مقدار تارگت اصلی، 1 بود: این موضوع به این معنی است که تمام Zها مقدار 1- داشتند؛ بنابراین حداقل یکی از آنها باید به مقدار 1 تغییر پیدا کند. برای این کار نزدیکترین Z_in به صفر را پیدا می کنیم و وزن ها را به شیوه الگوریتم Adaline برای آن به روز می کنیم.

اگر مقدار اصلی تارگت، 1- بود: این موضوع به این معنی است که حداقل یکی از Z ها برابر 1 بود. برای نتیجه درست، همه Zها باید مقدار 1- بگیرند. برای این کار وزن ها را برای تمام Z_inهایی که مقدار مثبت دارند، به روز می کنیم. مجددا در اینجا وزن ها را به شیوه الگوریتم Adaline به روز می کنیم.

قدم هشتم: اگر جداسازی به حدی قابل قبول رسید یا آپدیت کردن وزن ها به اندازه یک عدد ماکسیمم انجام شد، به روز رسانی را متوقف کن.

ب)

به شیوه ای که در شکل نشان داده شده، داده را در کد بارگذاری می شود و نمودار پراکندگی آن رسم شد. ازآنجایی که قرار است از الگوریتم MR1 برای جداسازی استفاده شود و در این الگوریتم، تارگت ها با مقادیر 1 و 1-نشان داده شده اند، تارگت های 0 در داده به تارگت 1- تبدیل شدند تا استفاده از الگوریتم ساده تر شود.

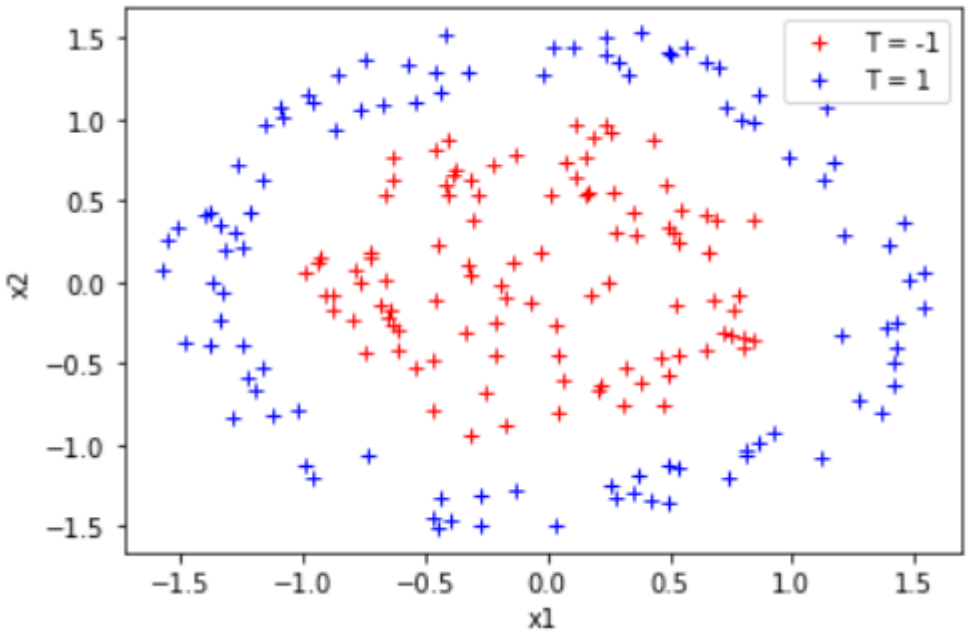
```
data = pd.read_csv('Question3.csv', header = None)
data.columns = ['x1', 'x2', 'T']
data['T'] = data['T'].map({0:-1, 1:1})

set1 = data[(data['T'] == -1)]
x1 = set1.iloc[:]['x1']
y1 = set1.iloc[:]['x2']

set2 = data[(data['T'] == 1)]
x2 = set2.iloc[:]['x1']
y2 = set2.iloc[:]['x2']

plt.plot(x1, y1, 'r+', label = 'T = -1')
plt.plot(x2, y2, 'b+', label = 'T = 1')
plt.legend(loc = "upper right")
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```

شکل27 : کد بارگذاری داده و رسم نمودار پراکندگی

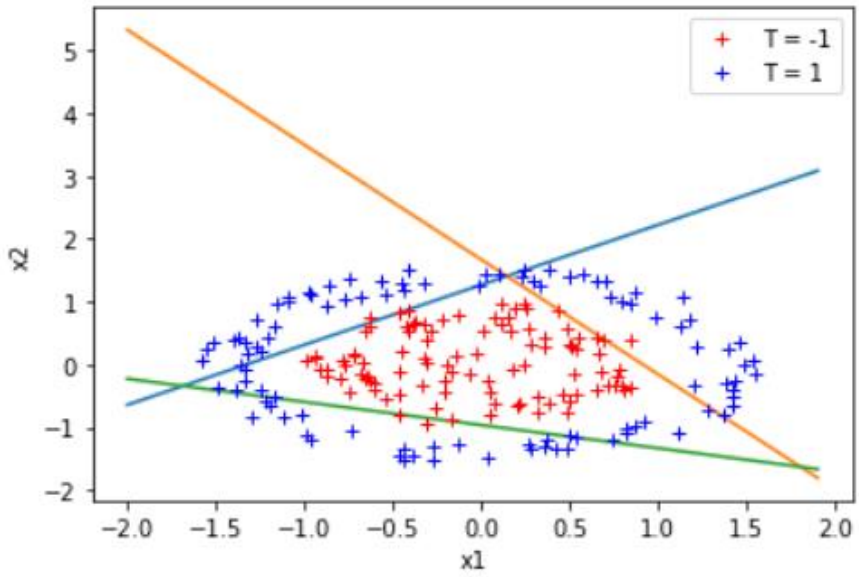


شکل 28: نمودار پراکندگی داده ها

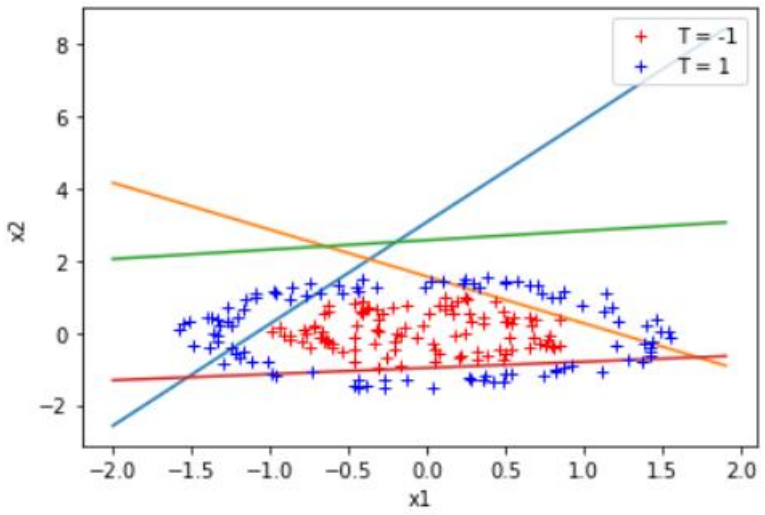
ج)

در این بخش برای هر سه حالت: 3 نورون، 4نورون و 8 نورون یادگیری شبکه مطابق با قدم های ذکرشده در بخش الف انجام شد. برای انجام این بخش، شرط توقف خاصی در نظر گرفته نشده چون می دانیم که ممکن است خط های حاصل از شبکه Madaline هیچ گاه به جداسازی دقیق و کامل نرسند. برای رسیدن به نتیجه، تعدادهای متفاوت ایپاک برای هرکدام از شبکه ها امتحان شد تا ببینیم با کدام تعداد نتیجه بهتری حاصل می شود.

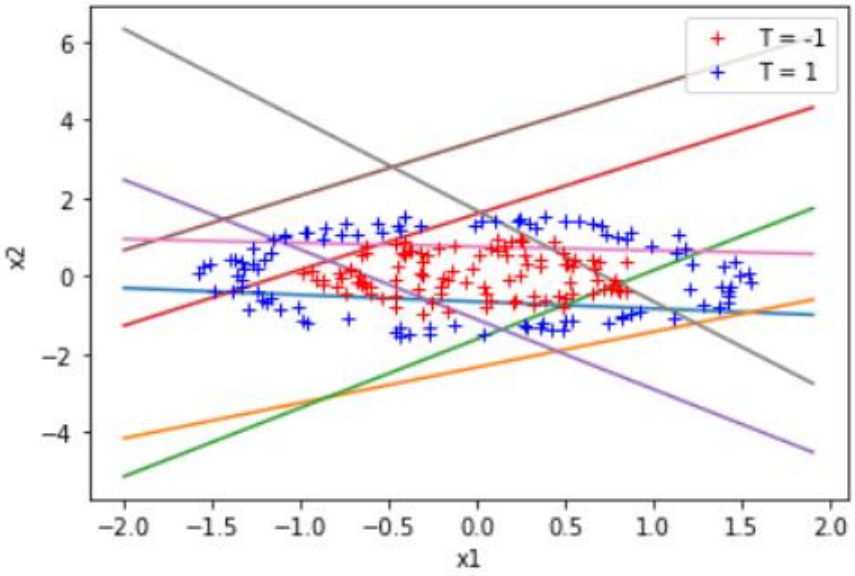
نتایج جداسازی برای هر کدام از حالت ها به صورت شکل های زیر است:



شکل 29: جداسازی داده ها با شبکه 3 نورونی



شکل 30: جداسازی داده ها با شبکه 4 نورونی



شکل31: جداسازی داده ها با شبکه 8 نورونی

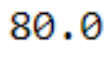
د)

انتظار ما از شبکه Madaline این است که اطراف داده ها یک n ضلعی بسیازد که n تعداد نورون های به کار رفته در شبکه است. همانطور که در شکل های 29 تا 31 می بینیم، در شبکه 3 و 4 نورونی تا حد خوبی این چندضلعی ایجاد شده است. هرچقدر تعداد نورون های لایه پنهانی افزایش پیدا کرد، میزان دقت و جداسازی داد ها نیز بهتر شد. در شبکه 8 نورونی اما اگرچه یک 8 ضلعی خوبی نشان داده نمی شود اما وقتی دقت را محاسبه می کنیم، می بینیم که جداسازی داده ها در این شبکه با دقت بهتری صورت گرفته است.

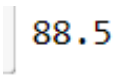
با افزیش تعداد نورون ها، خود به خود دقت بالاتر می شود؛ پس طبیعی است که هرچقدر لایه پنهانی نورون های بیشتری داشته باشد، تعداد ایپاک های لازم برای رسیدن به نتیجه ای قابل قبول کمتر شود. تعداد ایپاک ها در شبکه 3 و 4 نورونی، 10000 تا و در شبکه 8 نورونی، 6000 تا بود.

محاسبه دقت نیز به این صورت انجام گرف که بعد از خارج شدن از حلقه ایپاک ها و تمام شدن روند به روز رسانی وزن ها، وزن های نهایی را روی داده های موجود اعمال کردم. برای محاسبه دقت، مقدار Y تولید شده را با مقدار تارگت اصلی مقایسه کردم. با تقسیم تعداد حدس های درست بر تعداد کل داده ها، میزان دقت به دست آمد.

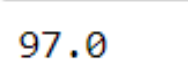
میزان دقت برای شبکه 3 نورونی، 80 درصد. برای شبکه 4 نورونی، 88.5 درصد و برای شبکه 8 نورونی، 97 درصد بود. همانطور که حدس می زدیم، میزان دقت با افزایش تعداد نورون های لایه پنهان، بیشتر شد.



شکل32: میزان دقت
شبکه 3 نورونی



شکل 33: میزان دقت شبکه
4 نورونی



شکل34: میزان دقت
شبکه 8 نورونی

سوال ۴ – Perceptron

$w_1 = 0.2, w_2 = 0.7, w_3 = 0.9, bias = -0.7$

$x_1 = 0, x_2 = 1, x_3 = 1, t = -1$

$\alpha = 0.3$

$$h = \begin{cases} 1 & net \geq 0 \\ -1 & net < 0 \end{cases}$$

---

w_1	w_2	w_3	$bias$	net	h	$error$
0.2	0.7	0.9	-0.7	0.9	1	2
0.2	0.4	0.6	-1	0	1	2
0.2	0.1	0.3	-1.3	-0.9	-1	0