1-

الف) سه بخش اصلی:

- دندریت[1]: وظیفه ی آوردن پیام های صادر شده از سلول های حسی مجاور یا نورون های رده بالاتر به درون سلول را بر عهده دارد.
- جسم سلولی[2]: حاوی هسته می باشد و مسئول حفظ شکل سلول به ویژه آکسون می باشند.
- آکسون[3]: وظیفه ی هدایت پیام های الکتریکی از جسم سلولی به خارج از نورون را دارد.

ب) سیناپس: یک اتصال شیمیایی می باشد که امکان اتصال دندریت به آکسون را فراهم می سازد.

وظایف سیناپس:

1. تبدیل فرکانس به ولتاژ
2. تقویت یا تضعیف خروجی آکسون

پ)

- دندریت: نقش بردار ورودی در نورون مصنوعی را دارد.
- جسم سلولی: نقش تابع جمع (سیگما) را دارد که باعث میشود ورودی ها بسته به یک وزنی که میگیرند با هم جمع شوند.
- آکسون: خروجی یک نورون می باشد که اگر از یک حدی که به کمک تابع انتقال[4] مشخص میشود بیشتر شود، خروجی فعال می شود.
- سیناپس: معادل وزن هایی می باشند که در ورودی ضرب میشوند.

[1] Dendrite
[2] Soma
[3] Axon
[4] Transfer Function

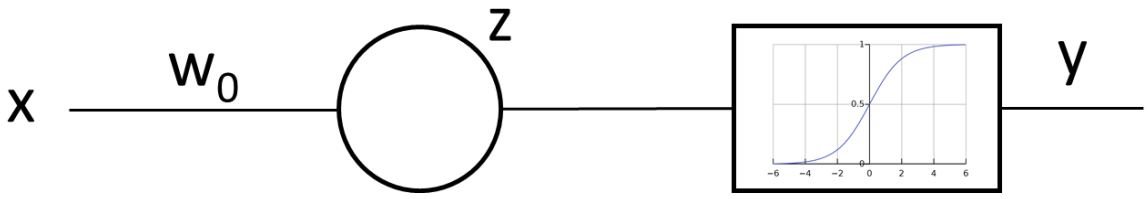
ت)

1. سرعت پردازش در شبکه های عصبی مصنوعی سریع تر از شبکه های عصبی طبیعی است.
2. تخصیص حافظه برای فرایند جدید در شبکه های عصبی مصنوعی غیر ممکن است زیرا که محل قبلی برای فرایند قبلی ذخیره شده است. در حالی که در شبکه های عصبی طبیعی به آسانی این کار تنها با تنظیم کردن رابطه ها صورت میگیرد.
3. پردازش ها در شبکه های عصبی مصنوعی به صورت متوالی صورت میگیرد. در حالی که در شبکه های عصبی طبیعی حجم زیادی از پردازش ها به صورت موازی صورت میگیرد.
4. اگر در شبکه عصبی مصنوعی داده ای خراب شود، امکان بازگردانی آن وجود ندارد. در حالی که در شبکه عصبی طبیعی داده در شبکه بین زیرگره ها پخش شده است. به طوری که اگر داده ای خراب شود امکان بازگردانی آن موجود است.

2-

الـف) در شبکه عصبی بیولوژیکی، ولتاژ بایستی از یک حد آستانه ای بیشتر شود تا آکسون یک شلیک انجام دهد. در شبکه عصبی مصنوعی نیز از این ایده تحت عنوان بایاس جهت منعطف پذیر کردن عملکرد استفاده شده است.

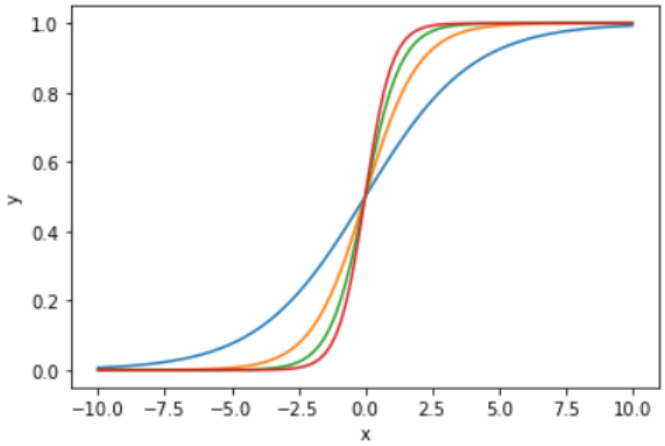
جهت فهم این موضوع، فرض کنید یک پرسپترون بسیار ساده داریم که تنها یک ورودی x را داراست و خروجی آن نیز از یک تابع فعالیت (که در قسمت (ت) درباره آن توضیح میدهیم) سیگموئید عبور میکند. داریم:



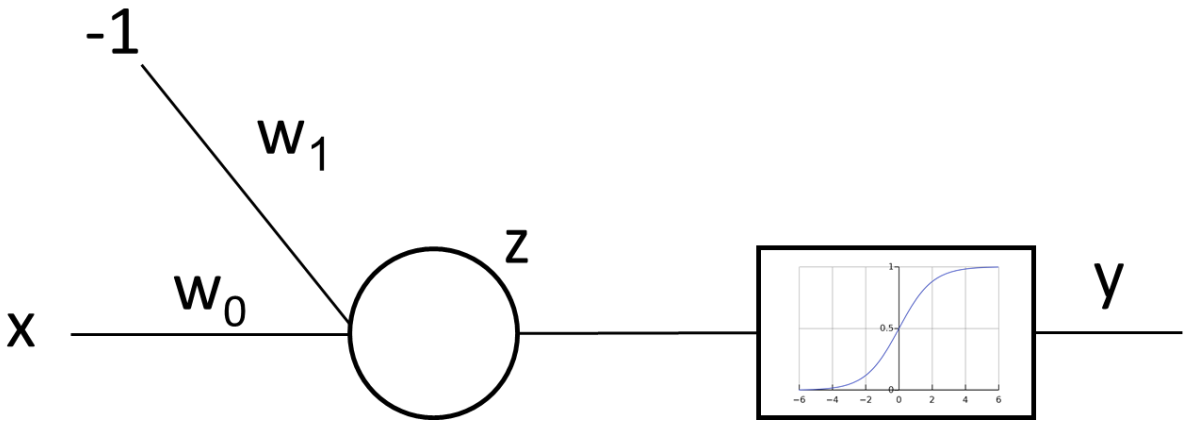
همانطور که از شکل فوق پیداست، ورودی در یک وزن w_0 ضرب میشود تا به خروجی برود. با شبیه سازی این موضوع به کمک پایتون در میابیم که تغییر وزن تنها میتواند شیب تابع سیگموئید را تغییر میدهد:

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
x = np.linspace(-10, 10, 100)

w0_values = [0.5, 1, 1.5, 2]

for w0 in w0_values:
    y = 1/(1 + np.exp(-x * w0))
    plt.plot(x, y)

plt.xlabel("x")
plt.ylabel("y");
```

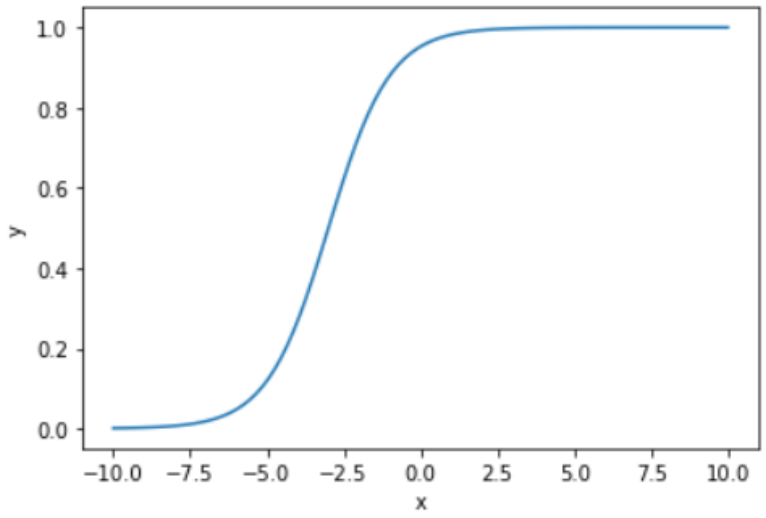


لذا انعطاف زیادی برای حل نخواهیم داشت. برای مثال فرض شود مسئله ما به گونه ایست که بایستی خروجی به ازای x های کوچک تر از 6- برابر با صفر شوند. و برای بزرگتر از آن مقدار داشته باشد. در اینجا بایاس به ما کمک میکند. به گونه ای که داریم:



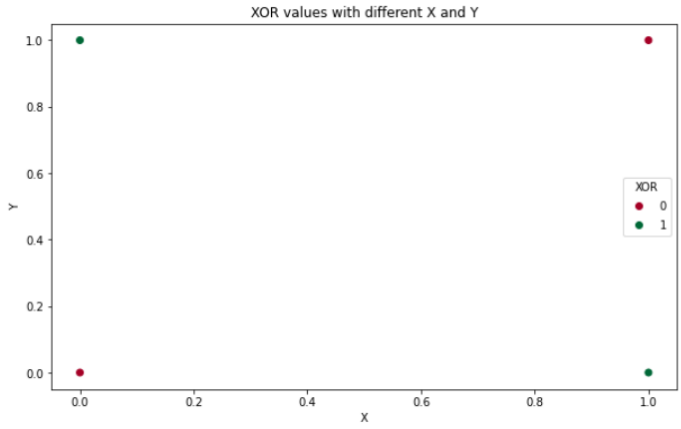
با توجه به پرسپترون فوق، میتوانیم w_1 را طوری تغییر دهیم که با انجام آن یک شیفت در تابع سیگموئید صورت بگیرد.

```
In [7]: x = np.linspace(-10, 10, 100)

        w0 = 1
        w1 = 3

        y = 1/(1 + np.exp(-x * w0 - w1))
        plt.plot(x, y)

        plt.xlabel("x")
        plt.ylabel("y");
```



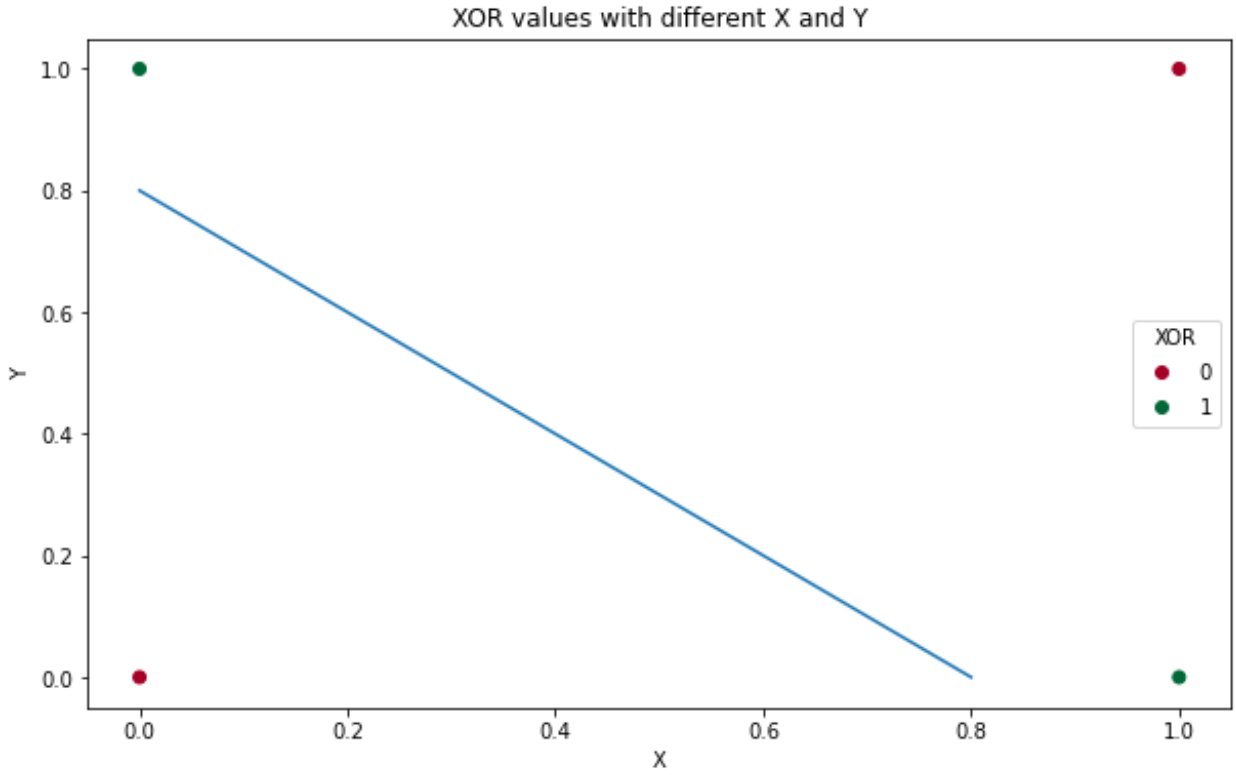
ب) برای درک این مطلب، نمودار scatter برای XOR رسم میکنیم:

```
fig, ax = plt.subplots(figsize=(10, 6))

xor_scatter = ax.scatter(x=xor_data['x'],
                         y=xor_data['y'],
                         c=xor_data['XOR'],
                         cmap='RdYlGn')


ax.set(title="XOR values with different X and Y",
       xlabel='X',
       ylabel="Y")

ax.legend(*xor_scatter.legend_elements(), title="XOR");
```



همانطور که در شکل میبینیم، XOR در نقاط سمت راست و پایین سمت چپ برابر با 0 شده که میخواهیم جاهایی که XOR = 1 و جاهایی

اما مشکلی که داریم این است که همانطور که میدانیم با یک پرسپترون تنها یک خط میتوان تولید کرد. از طرفی خطی موجود نیست که بتواند نمودار را به دو بخش XOR = 0 و XOR = 1 تقسیم کند.

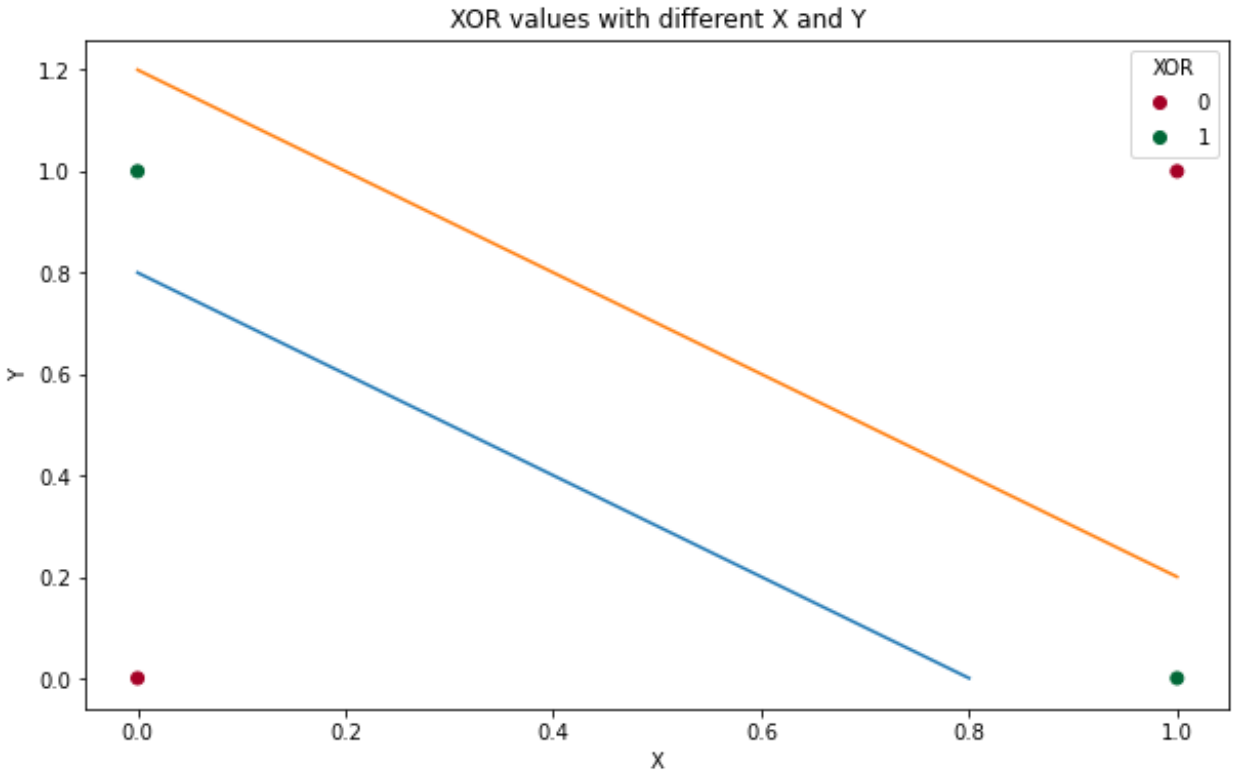


برای مثال اگر همچین خطی را داشته باشیم، در سمت بالای آن هم XOR = 0 هم XOR = 1 موجود است. با تغییر شیب و عرض از مبدا به هر شکل نمیتوان این 2 را از هم تفکیک کرد.

پ) اگر با توجه به قسمت (ت) فرض بر این است که تا به الان تابع فعالیتی نداریم، اضافه کردن لایه و نورون بی فایده است. زیرا که خروجی یک پرستپرون ترکیب خطی ورودی در وزن می باشد. حال اگر چند پرسپترون در لایه های مختلف بدون تابع فعالیت داشته باشیم، گویی یک ترکیب خطی از یک ترکیب خطی داریم که در نهایت انگار همچنان یک پرسپترون ولی با ورودی ها و وزن های بیشتر است! پس نتیجه همچنان قابلیت غیرخطی بودن را ندارد و توانایی تفکیک را ندارد.

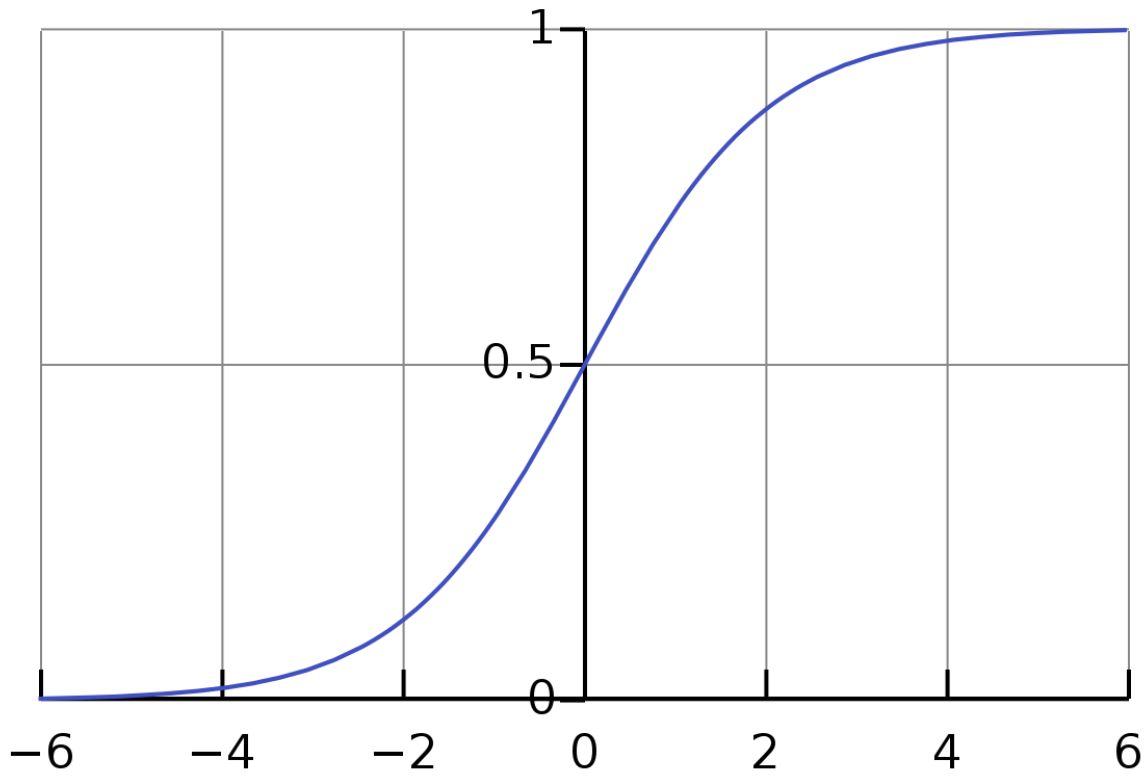
ت) مهم ترین وظیفه تابع فعالیت این است که مدل ما را غیر خطی میکند. در این صورت افزایش نورون ها به تنهایی قادر به حل این مساله نیست. اما به روش ریاضی اثبات شده است در صورتی که از یک MLP با 3 لایه و تعداد نورون کافی استفاده کنیم، قادر هستیم هر ابرمنحنی را مدل کنیم (با افزایش لایه ها تنها تعداد

وزن ها بیشتر و لذا مدل سازی را دقیق تر خواهد کرد. که البته به سایز داده های آموزشی هم وابسته است که overfitting صورت نگیرد). پس در صورتی که از یک مدل پیچیده تر استفاده کنیم، قادر هستیم یک جداسازی غیرخطی را انجام دهیم. به عنوان نمونه میتوان به شکل زیر اشاره کرد:



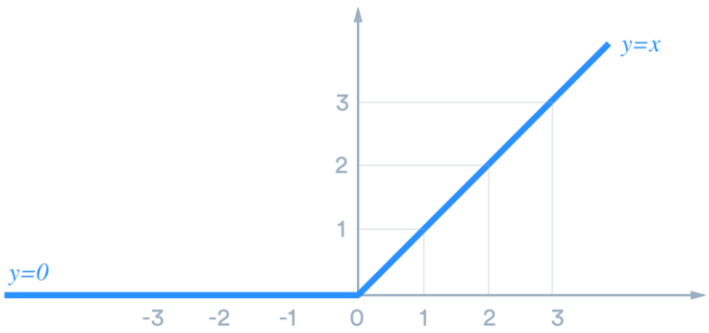
به کمک تقسیم بندی فوق، میتوان صفحه را به 3 بخش تقسیم کرد و به خوبی XOR را مدل کرد.

ث) تـابـع Sigmoid:



این تابع همانطور که از شکل فوق پیداست هر ورودی را به یک عدد در بازه [0, 1] تبدیل میکند. از عیب آن میتوان به پاسخگو نبودن در شبکه های عمیق استفاده کرد. چرا که در backpropagation بایستی مدام مشتق تابع فعالیت ضرب شود و در صورتی که تعداد لایه ها زیاد باشد، تعداد زیادی از مشتق تابع فعالیت سیگموئید داریم که بین 0 و 1می باشند و با ضرب شدن آن سبب می شود که خروجی به صفر میل بکند و یادگیری مفیدی صورت نگردد. لذا برای شبکه های عمیق بهتر است از سیگموئید استفاده نشود.

تـــابـــع ReLU :



همانطور که در شکل میبینیم، در تابع ReLU خروجی در x>0 برابر با y=x و در x<0 برابر با y=0 می باشد. این امکان سبب میشود که بتوانیم ReLU را در شبکه های عمیق استفاده کنیم، زیرا مشکل سیگموئید را

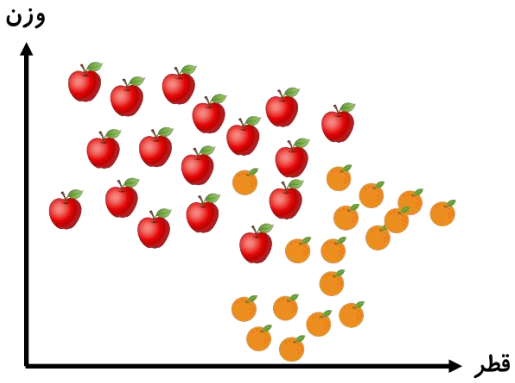
نخواهد داشت که همگی بین 0 و 1 باشند و به صفر میل بکند.

از طرفی از معایب ReLU این است که در $x < 0$ برابر با $y=0$ می باشد که در نتیجه در برخی مسائل مشکل ساز می شود. برای راه حل برخی پیشنهاد داده اند که در $x < 0$ نیز یک مقدار ناچیزی را اختیار کند.

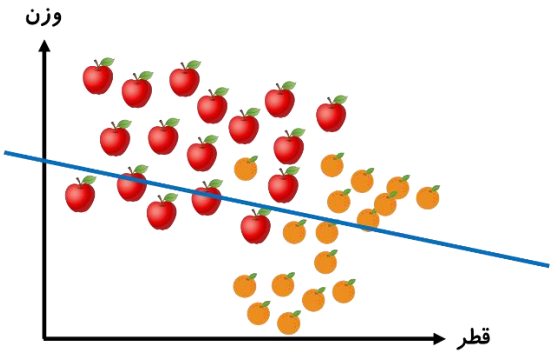
با توجه به 2 تابع فعالیت فوق مشاهده میکنیم که در زمانی که تعداد لایه های شبکه عصبی کم است، سیگموئید گزینه مناسبی می تواند باشد چرا که انعطاف بیشتری در حول 0 دارد ولی برای شبکه های عمیق با لایه های بالا انتخاب ReLU معقول تر است.

3-

الـف) برای فهم این موضوع داده های زیر را در نظر بگیرید:

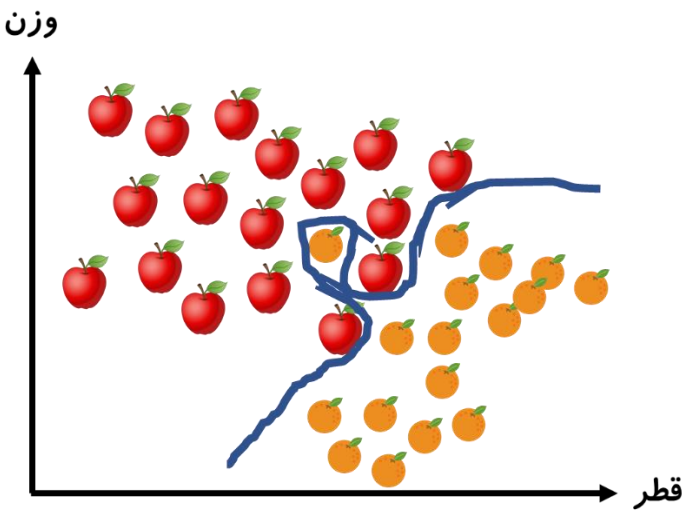


فرض شود میخواهیم نمودار را به دو بخش تقسیم کنیم. به طوری که سیب ها و پرتغال ها به خوبی از هم تفکیک شوند.



اگر خطی به روش فوق جهت جدا سازی رسم شود، به خوبی توانایی جداسازی را ندارد. به عبارت دیگر خطای زیادی را شامل می شود. در این گونه مواقع میگوییم دچار بایاس بالا یا Underfitting شده ایم.

اما حالت زیر را در نظر بگیرید:



در حالت فوق منحنی رسم شده به شکل غیرقابل عادی و دقیق عمل جداسازی را انجام میدهد. هرچند خطا به نظر کم میرسد اما این عمل اشکال دارد. زیراکه به شدت به داده های آموزشی وابسته شده است و هنگام مشاهده داده جدید دیگر به خوبی قابلیت جداسازی را ندارد و خطا زیاد میشود. به این حالت واریانس بالا یا Overfitting می گویند.

ب) برای این کار داده های موجود به سه بخش تقسیم میشود. حدود 70 الی 80 درصد آن جهت داده های آموزشی استفاده میشوند و عملیات یادگیری روی آن صورت میگیرد. حدود 10 الی 15 درصد آن برای Validation جدا میشوند تا Hyperparameter هایی نظیر تعداد EPOC ها، تعداد لایه ها و نورون ها مشخص شود. و در آخر 10 الی 15 درصد داده نیز برای تست جدا سازی می شود تا ارزیابی جهت خطا صورت گیرد.

حال اگر خطای مورد نظر ما در داده های آموزشی و داده های Validation زیاد باشد، یعنی به خوبی نتوانستیم مدل کنیم و لذا دچار بایاس بالا هستیم.

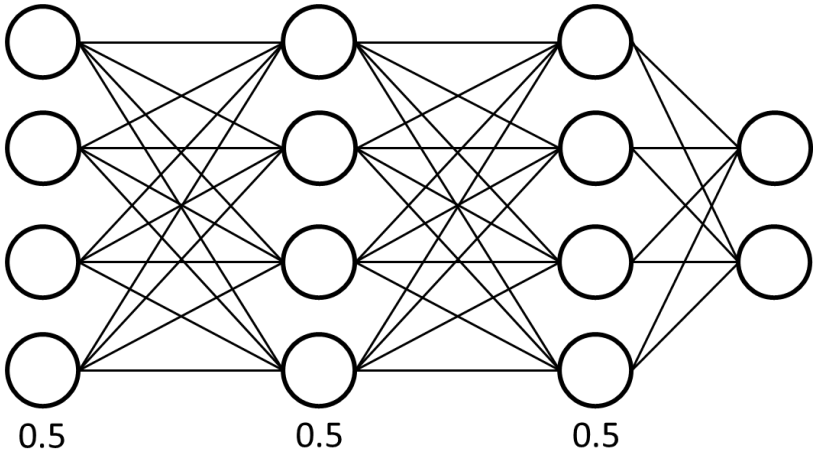
اگر داده های آموزشی خطای خیلی کم داشته باشند ولی داده های Validation خطای زیادی داشته باشند آنگاه دچار واریانس بالا هستیم.

پ) راه های مختلفی را میتوان امتحان کرد:

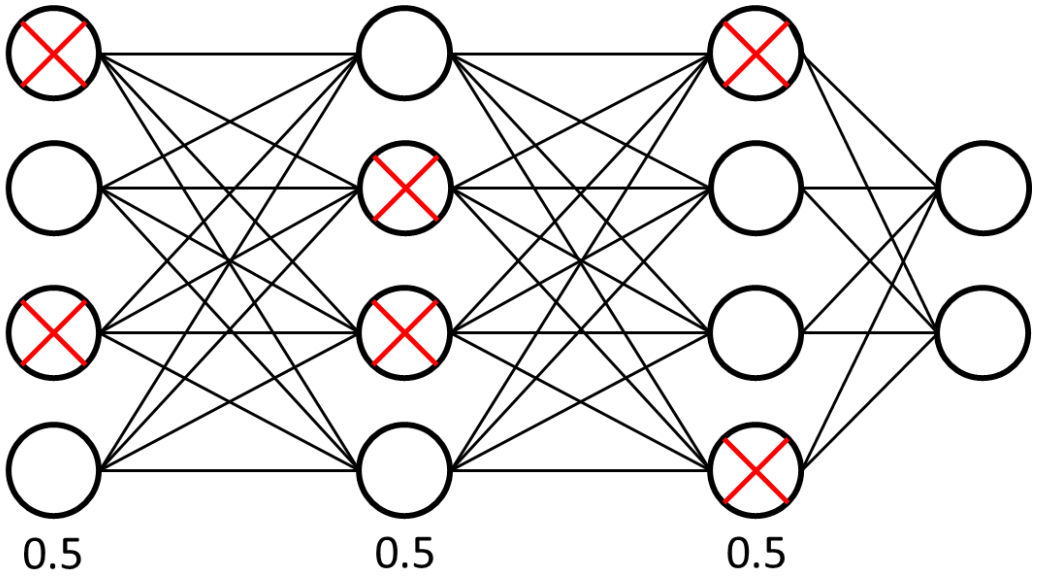
- **درست کردن یک شبکه بزرگتر:** استفاده از لایه های پنهانی بیشتر و تعداد نورون های بیشتر
- **تغییر در شیوه بهینه سازی:** استفاده از الگوریتم های بهینه سازی موثر تر

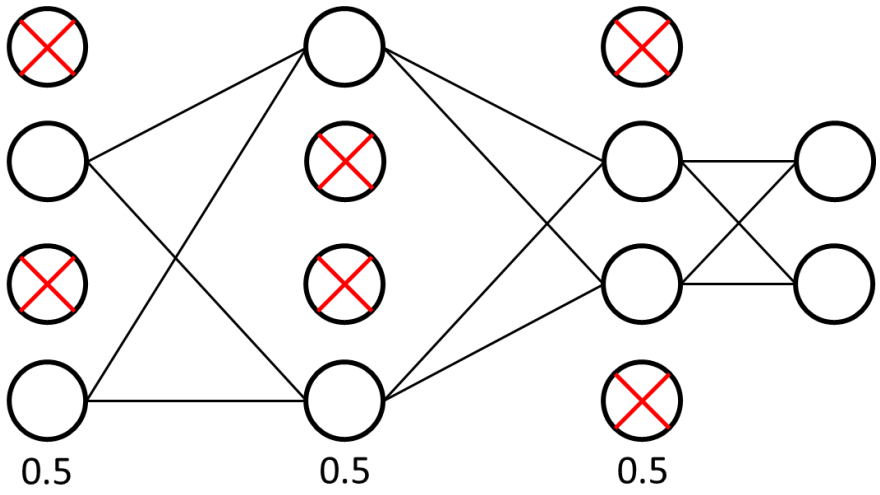
- **استفاده از معماری شبکه عصبی بهتر**

ت) Dropout: در این روش برای هر لایه در شبکه عصبی یک احتمال در نظر میگیریم. برای مثال فرض شود برای هرکدام 0.5 در نظر بگیریم:



سپس برای هر پرسپترون در هر لایه یک عدد رندم بین 0 و 1 تولید میشود. در مثال ما اگر این عدد بین 0 تا 0.5 باشد پرسپترون حذف وگرنه باقی میماند. فرض شود پرسپترون هایی که میخواهند حذف شوند برابرند با:





با حذف ارتباطات خروجی و ورودی از این پرسپترون ها، شبکه عصبی از واریانس بالا خارج میشود و مدل ساده تری خواهیم داشت.

Regularization: از معروف ترین آن ها L2 و L1 می باشند که هردوی آن ها تلاش بر آپدیت کردن Cost Function را دارند. به طوری که داریم:

$$\text{Cost function} = \text{Loss} + \text{Regularization Term}$$

با افزایش Regularization Term، مقدار وزن های شبکه عصبی برای رسیدن به Cost Function مورد نظر کاهش می یابند. به طور شهودی میتوان چنین فرض کردن کاهش وزن ها به گونه ای میباشند که گویی مقداری از لایه های پنهان نادیده گرفته میشوند و حذف میشوند. پس با ساده تر شدن مدل همراه خواهیم شد که نتیجه ی آن جلوگیری از واریانس بالا خواهد بود.

$$\text{L2: Cost Function} = \text{Loss} + \frac{\lambda}{2m}\sum||w||^2$$

$$\text{L1: Cost Function} = \text{Loss} + \frac{\lambda}{2m}\sum||w||$$

در دو رابطه فوق λ برابر با پارامتر Regularization و m برابر با تعداد ورودی ها می باشد. مشاهده میکنیم تفاوت L2 و L1 در قسمت سیگما میباشد. به طوری که L2 از w^Tw (که w وزن ها می باشند) استفاده خواهد کرد در صورتی که L1 از قدر مطلق آن استفاده خواهد کرد. لذا در L1 احتمال این که به صفر میل بکند و مدل فشرده تری داشته باشیم بهتر است. اما تاثیر چندانی نخواهد داشت و معمولا از L2 استفاده خواهد شد.

-4

الـف) اگر $Z_1 = BT + b_3$ ، $Z_2 = AV + b_2$ و $Z_3 = WX + b_0$ در نظر بگیریم، داریم:

$$\frac{\partial\, Cost}{\partial B_0} = \frac{\partial\, Cost}{\partial O_1} \times \frac{\partial\, O_1}{\partial Z_1} \times \frac{\partial\, Z_1}{\partial B_0}$$

حال داریم:

$$\frac{\partial\, Cost}{\partial O_1} = 2(O_1 - y_t)$$

$$\frac{\partial\, O_1}{\partial Z_1} = S'(Z_1) = S(Z_1)\big(1 - S(Z_1)\big)$$

با توجه به این که $Z_1 = B_0 t_0 + b_3$ می باشد داریم:

$$\frac{\partial\, Z_1}{\partial B_0} = t_0$$

$$\frac{\partial\, Cost}{\partial B_0} = 2(O_1 - y_t)S(Z_1)\big(1 - S(Z_1)\big)t_0$$

حال به یک لایه قبل تر میرویم و داریم:

$$\frac{\partial\, Cost}{\partial A_0} = \frac{\partial\, Cost}{\partial B_0} \times \frac{\partial\, B_0}{\partial Z_2} \times \frac{\partial\, Z_2}{\partial A_0}$$

$$\frac{\partial\, B_0}{\partial Z_2} = S'(Z_2) = S(Z_2)\big(1 - S(Z_2)\big)$$

با توجه به این که $Z_2 = A_0 v_0 + A_1 v_1 + b_2$ می باشد داریم:

$$\frac{\partial\, Z_2}{\partial A_0} = v_0$$

$$\frac{\partial\, Cost}{\partial A_0} = 2(O_1 - y_t)S(Z_1)\big(1 - S(Z_1)\big)t_0 S(Z_2)\big(1 - S(Z_2)\big)v_0$$

و در آخر برای رسیدن به خواست سوال داریم:

$$\frac{\partial\, Cost}{\partial w_1} = \frac{\partial\, Cost}{\partial A_0} \times \frac{\partial\, A_0}{\partial Z_3} \times \frac{\partial\, Z_3}{\partial w_1}$$

$$\frac{\partial\, A_0}{\partial Z_3} = S'(Z_3) = S(Z_3)\big(1 - S(Z_3)\big)$$

با توجه به این که $Z_3 = w_0X_0 + w_1X_1 + w_2X_2 + b_0$ می باشد داریم:

$$\frac{\partial\, Z_3}{\partial w_1} = X_1$$

بنابراین داریم:

$$\frac{\partial\, Cost}{\partial w_1} = 2(O_1 - y_t)S(Z_1)\big(1 - S(Z_1)\big)t_0 S(Z_2)\big(1 - S(Z_2)\big)v_0 S(Z_3)\big(1 - S(Z_3)\big)X_1$$

ب) برای به دست آوردن خروجی داریم:

$$O_1 = Sigmoid(B_0t_0 + b_3)$$
$$B_0 = Sigmoid(A_0v_0 + A_1v_1 + b_2)$$
$$A_1 = Sigmoid(u_0X_0 + u_1X_1 + u_2X_2 + b_1)$$
$$A_0 = Sigmoid(w_0X_0 + w_1X_1 + w_2X_2 + b_0)$$

با وجود روابط فوق، ابتدا به محاسبه A_0 و A_1 می پردازیم:

$$A_0 = Sigmoid(0.4 \times 1 + 0.3 \times 1 + 0.2 \times 1 + 0.5)$$
$$= Sigmoid(1.4) = \frac{1}{1 + e^{-1.4}} = \frac{1}{1.2466} = 0.8022$$

$$A_1 = Sigmoid(0.3 \times 1 + 0.4 \times 1 + 0.5 \times 1 + 0.2)$$
$$= Sigmoid(1.4) = \frac{1}{1 + e^{-1.4}} = \frac{1}{1.2466} = 0.8022$$

$$B_0 = Sigmoid(0.8022 \times 0.6 + 0.8022 \times 0.7 + 0.5)$$
$$= Sigmoid(1.543) = \frac{1}{1 + e^{-1.543}} = \frac{1}{1.2137} = 0.8239$$

$$O_1 = Sigmoid(0.8239 \times 0.9 + 0.5)$$

$$= Sigmoid(1.2415) = \frac{1}{1 + e^{-1.2415}} = \frac{1}{1.2889} = 0.775$$

خطا نیز برابر است با:

$$cost\ func = (0.775 - 0)^2 = 0.602$$

5- داریم:

$$W^{new} = W^{old} - \eta \nabla Cost(W^{old})$$

لذا بایستی برای بدست آوردن W^{new}، ابتدا به محاسبه گرادیان بپردازیم:

$$\nabla Cost(W^{old}) = \begin{bmatrix} \frac{\partial\ Cost}{\partial w_0} \\ \frac{\partial\ Cost}{\partial w_1} \\ \frac{\partial\ Cost}{\partial w_2} \end{bmatrix}$$

از طرفی با توجه به شکل مساله و مشتق بدست آورده در سوال 4 (الف) میتوان نوشت:

$$\frac{\partial\ Cost}{\partial w_0} = 2(O_1 - y_t)S(Z_1)(1 - S(Z_1))t_0 S(Z_2)(1 - S(Z_2))v_0 S(Z_3)(1 - S(Z_3))X_0$$

$$\rightarrow \frac{\partial\ Cost}{\partial w_0} = 2(0.775 - 0) \times 0.775(1 - 0.775) \times 0.9 \times 0.8239(1 - 0.8239) \times 0.6 \times 0.8022(1 - 0.8022) \times 1 = 0.0033$$

(0.225 → 0.1978, 0.1761)

$$\frac{\partial\ Cost}{\partial w_1} = 2(O_1 - y_t)S(Z_1)(1 - S(Z_1))t_0 S(Z_2)(1 - S(Z_2))v_0 S(Z_3)(1 - S(Z_3))X_1 = 0.0033$$

$$\frac{\partial\ Cost}{\partial w_2} = 2(O_1 - y_t)S(Z_1)(1 - S(Z_1))t_0 S(Z_2)(1 - S(Z_2))v_0 S(Z_3)(1 - S(Z_3))X_2 = 0.0033$$

پس بردار گرادیان برابر است با:

$$\nabla Cost(W^{old}) = \begin{bmatrix} 0.0033 \\ 0.0033 \\ 0.0033 \end{bmatrix}$$

با این وجود با توجه به این که ضریب یادگیری0.1 می باشد داریم:

$$W^{new} = \begin{bmatrix} 0.4 \\ 0.3 \\ 0.2 \end{bmatrix} - 0.1 \begin{bmatrix} 0.0033 \\ 0.0033 \\ 0.0033 \end{bmatrix} = \begin{bmatrix} 0.39967 \\ 0.29967 \\ 0.19967 \end{bmatrix}$$