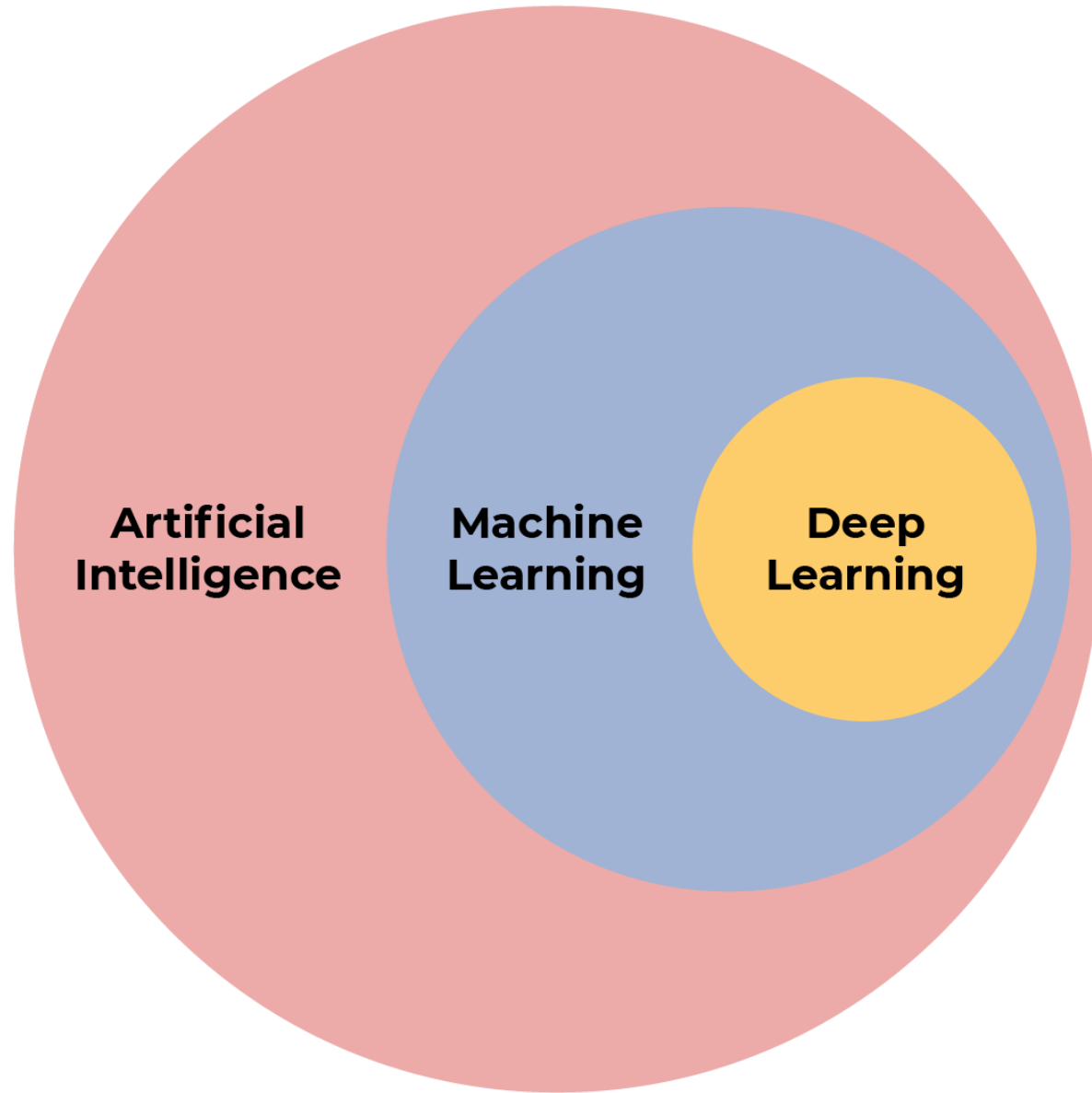


# Computational Intelligence

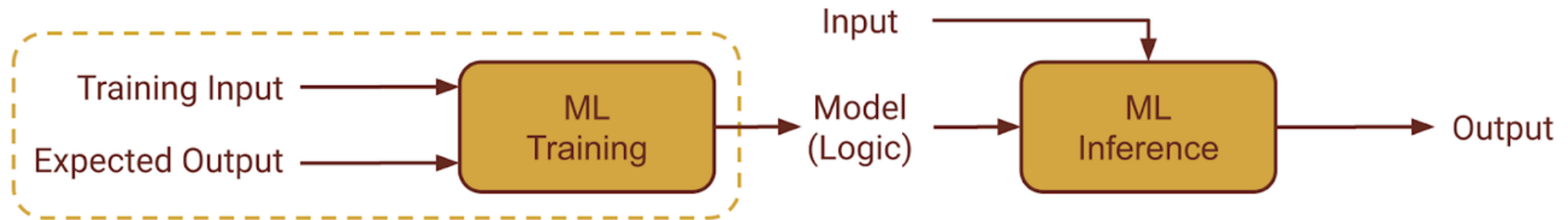
Professor: Dr. Mohammad Zare  
Teaching Assistant: Ali Kohan

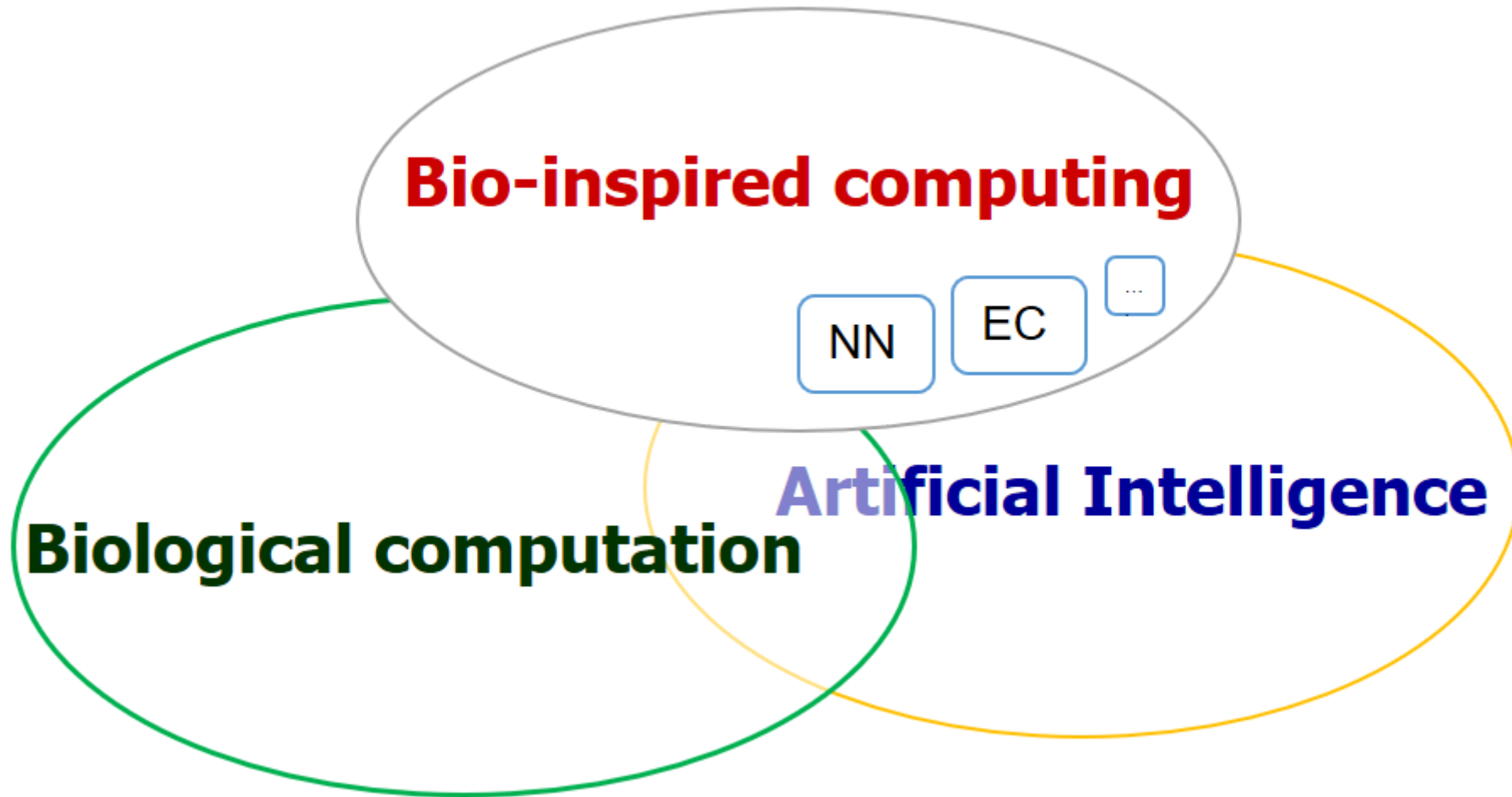


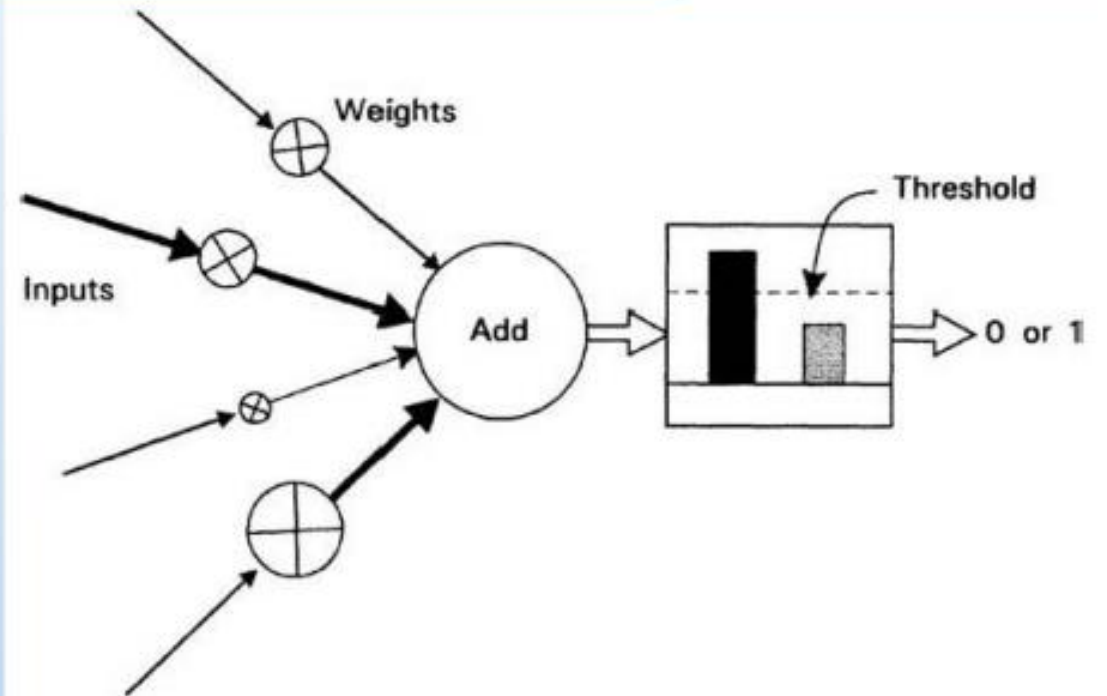
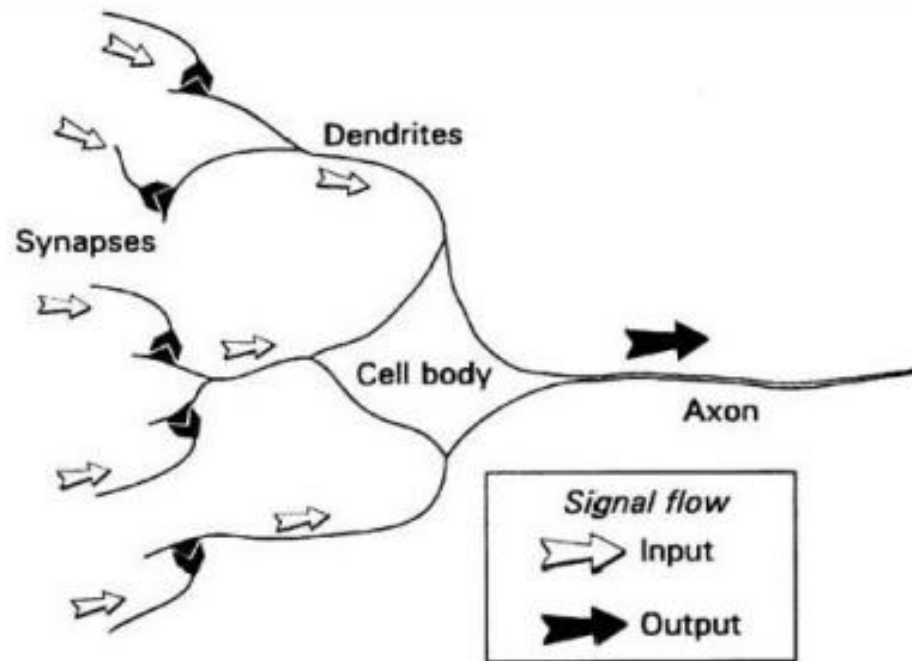
**Traditional Programs:** Define algo/logic to compute output



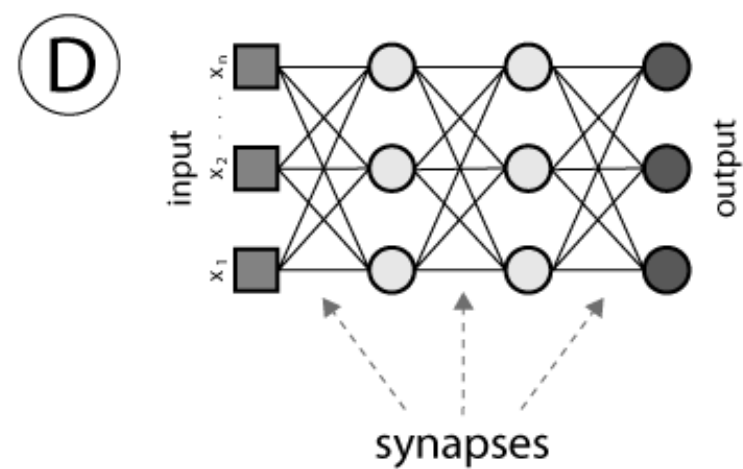
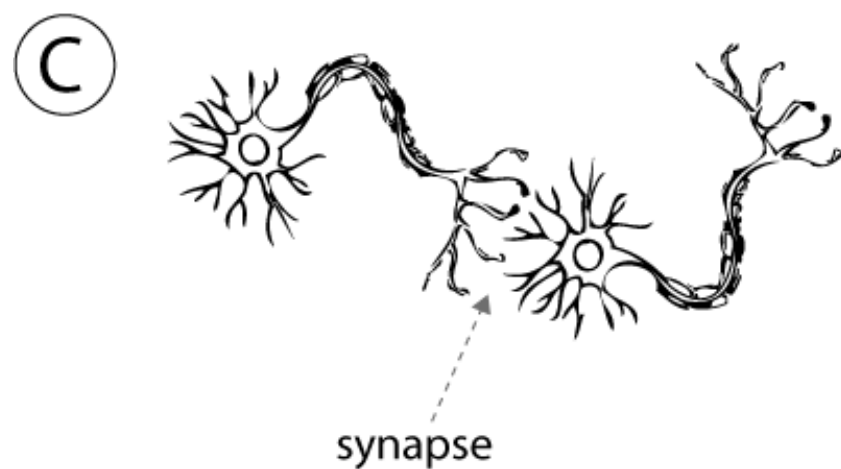
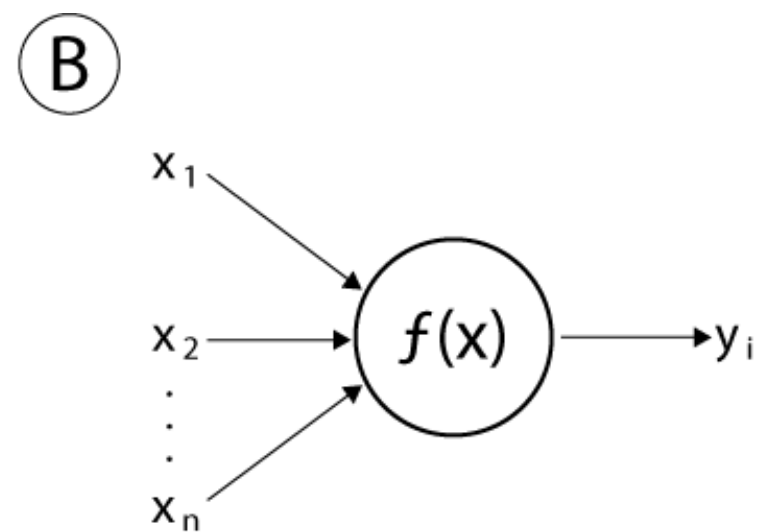
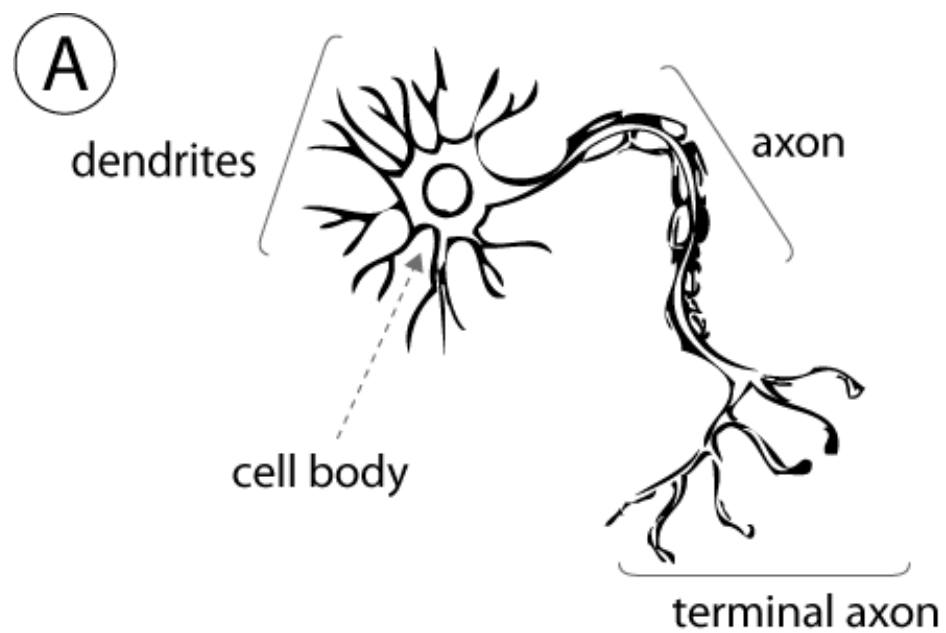
**Machine Learning:** Learn model/logic from data

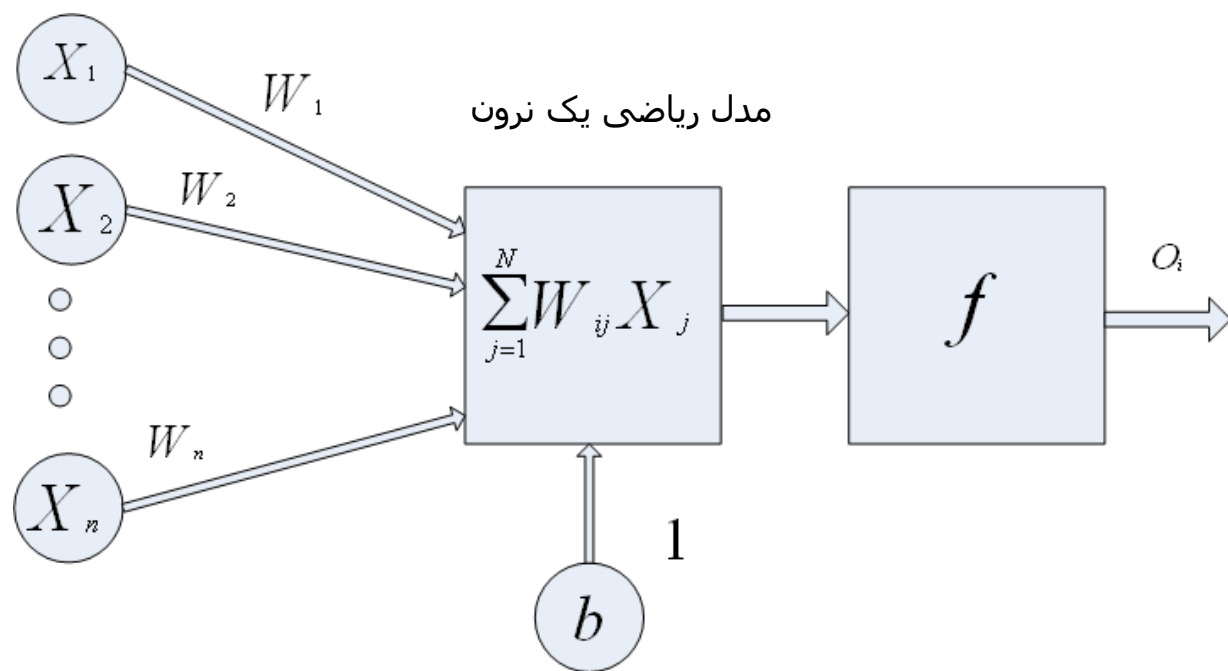




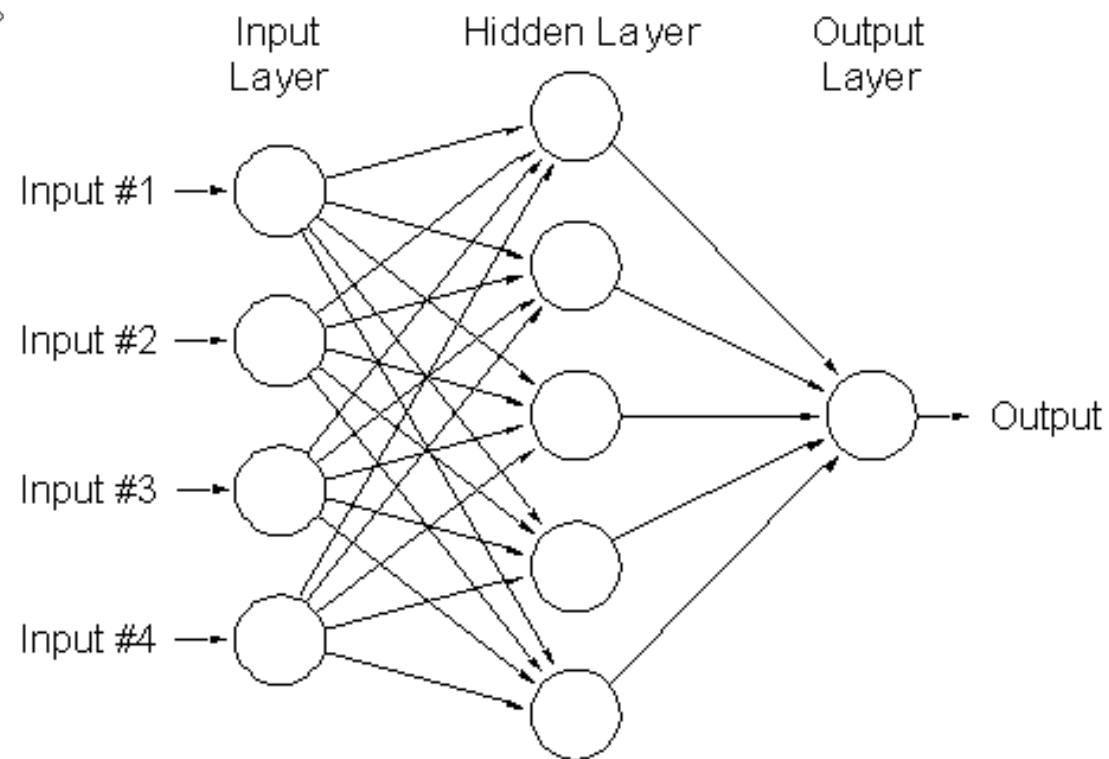


<i>Biological Neural Network</i>	<i>Artificial Neural Network</i>
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

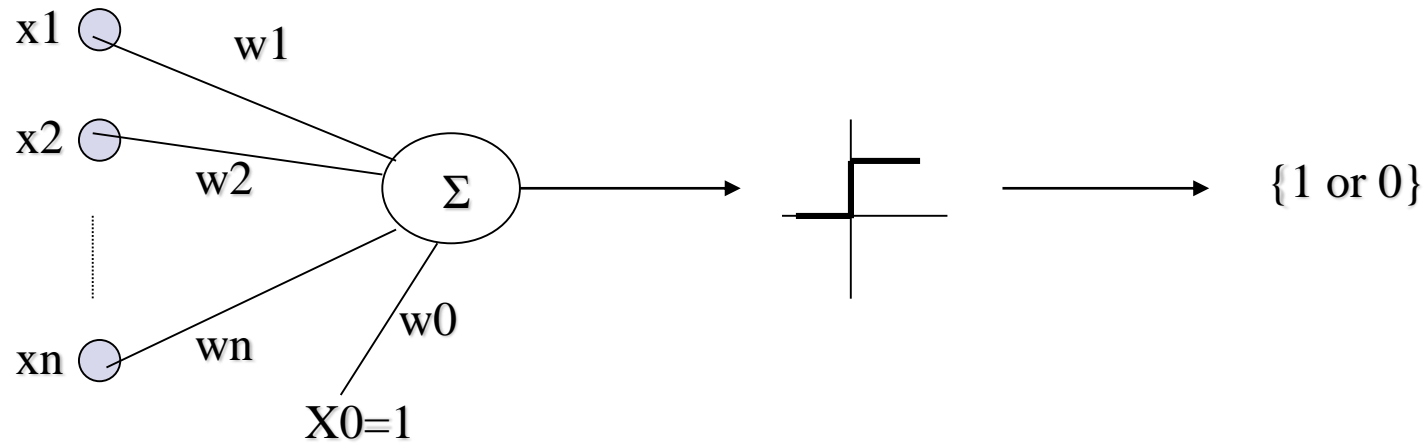




ساختار کلی یک شبکه عصبی مصنوعی

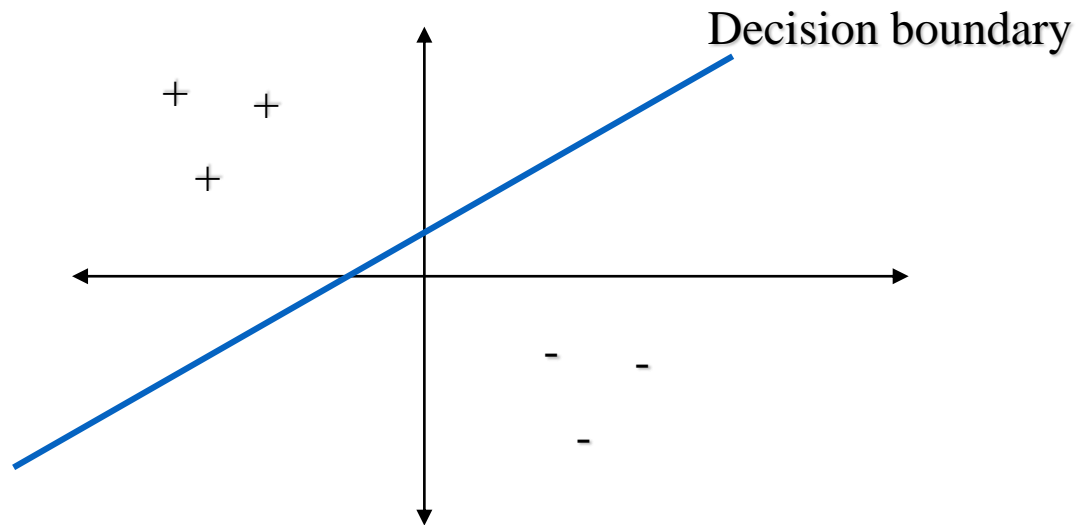


- یکی از ساده ترین مدل های شبکه عصبی بر مبنای یک واحد محاسباتی به نام پرسپترون ساخته می شود.
- یک پرسپترون، برداری از ورودی های با مقادیر حقیقی را گرفته و یک ترکیب خطی از این ورودی ها را محاسبه میکند.
- اگر نتیجه از یک مقدار آستانه بیشتر بود خروجی پرسپترون برابر با 1 و در غیر اینصورت معادل 0 (یا -1) خواهد بود.

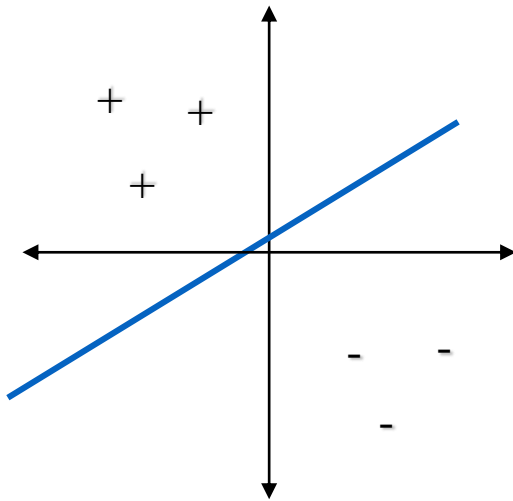




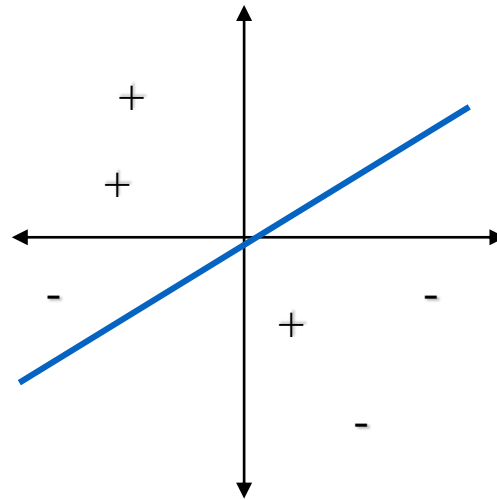
- پرسپترون را می توان به صورت یک hyperplane ( $n-1$  بعدی) در فضای  $n$  بعدی نمونه ها در نظر گرفت
- پرسپترون برای نمونه های یک طرف صفحه مقدار 1 و برای مقادیر طرف دیگر مقدار 0 بوجود می آورد.



- یک پرسپترون ساده تنها قادر است مسائلی را یاد بگیرد که به صورت خطی جداپذیر باشند.
- اینگونه مثال ها مواردی هستند که بطور کامل توسط یک hyperplane قابل جدا سازی میباشند.



Linearly separable



Non-linearly separable

- خروجی پرسپترون توسط رابطه زیر مشخص میشود:

$$O(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } W_0 + W_1.X_1 + W_2.X_2 + \dots + W_n.X_n > 0 \\ 0 & \text{otherwise} \end{cases}$$

- که برای سادگی آنرا میتوان بصورت زیر نشان داد:

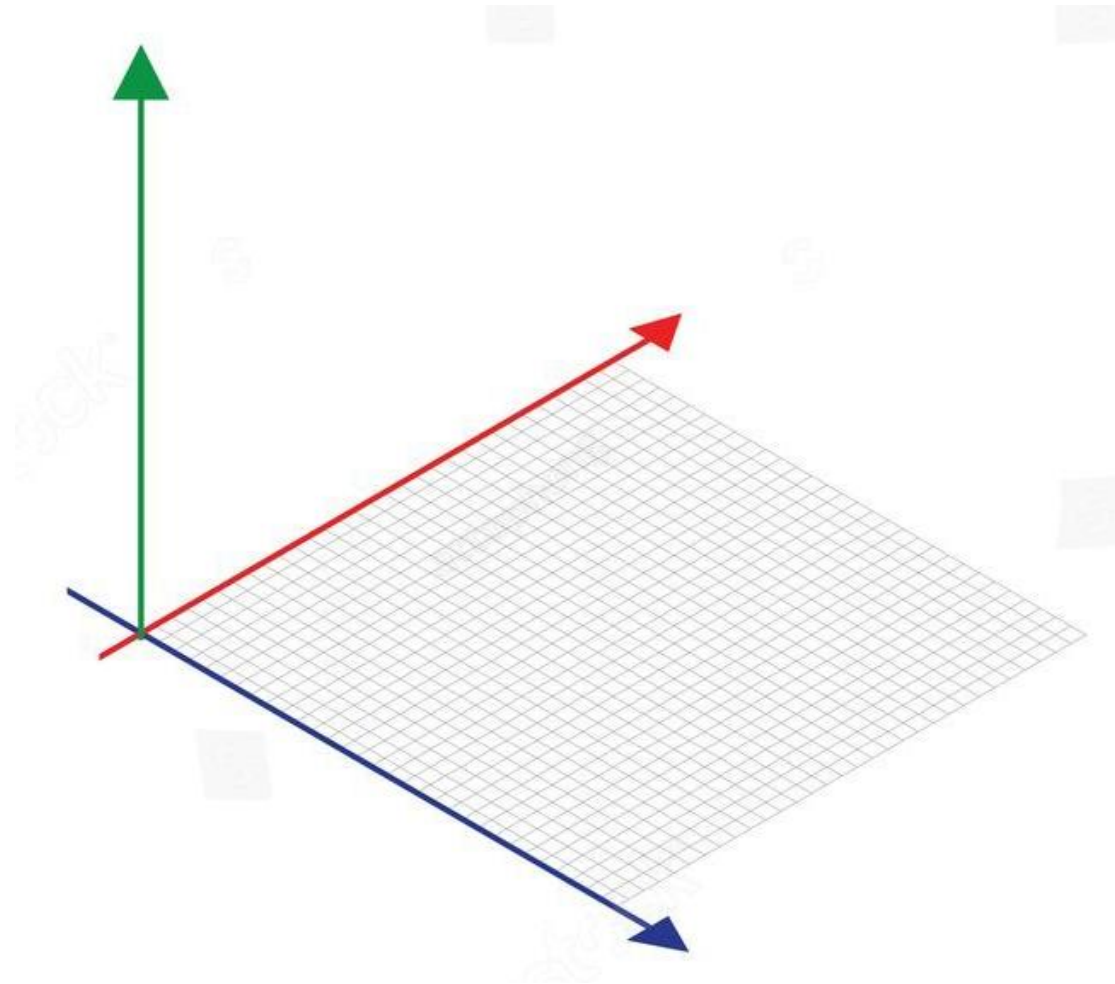
$$O(\mathbf{X}) = \text{sgn}(\mathbf{W}\mathbf{X}) \text{ where}$$

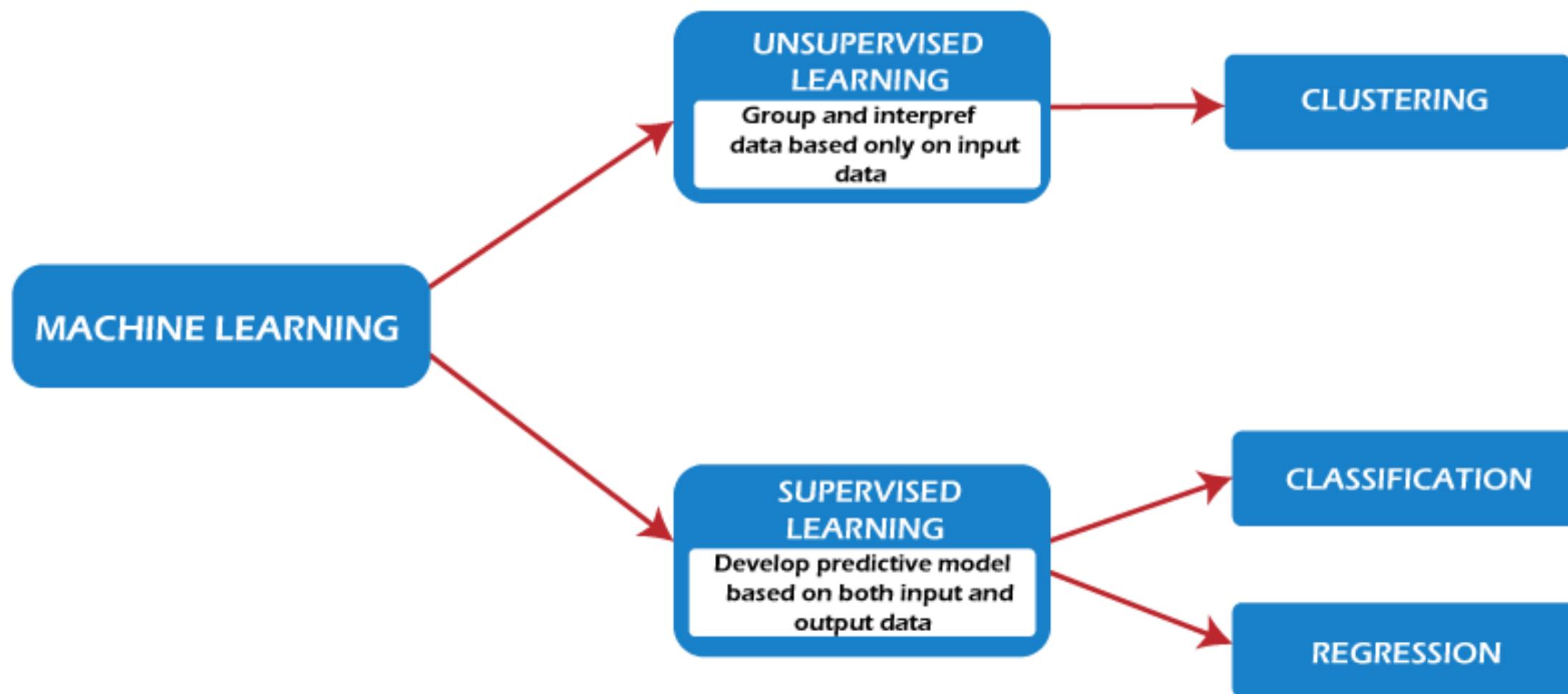
$$\text{sgn}(y) = \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{otherwise} \end{cases}$$

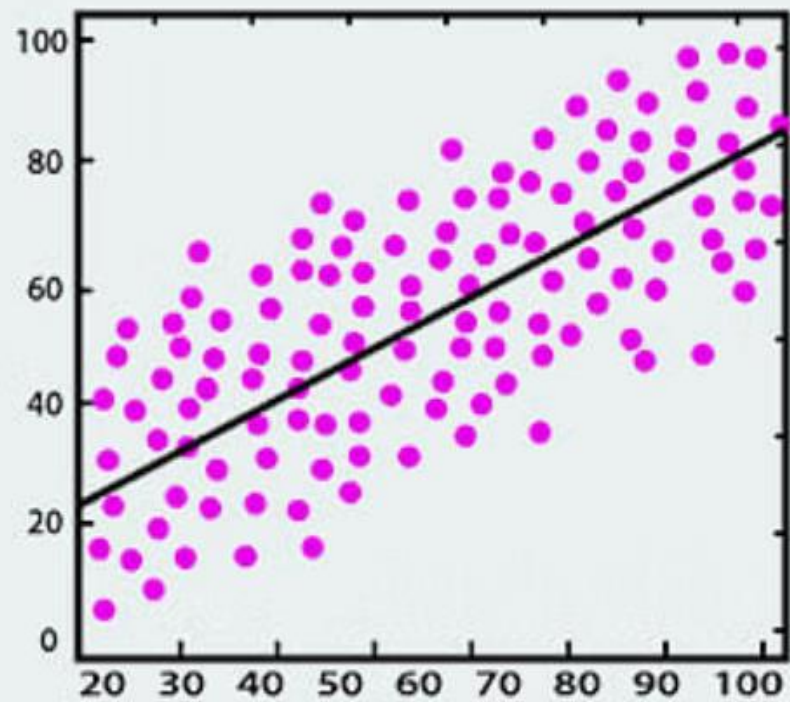
یادگیری پرسپترون عبارت است از:

پیدا کردن مقادیر مناسبی برای  $W$

عرض از مبدا  $\rightarrow$  Bias

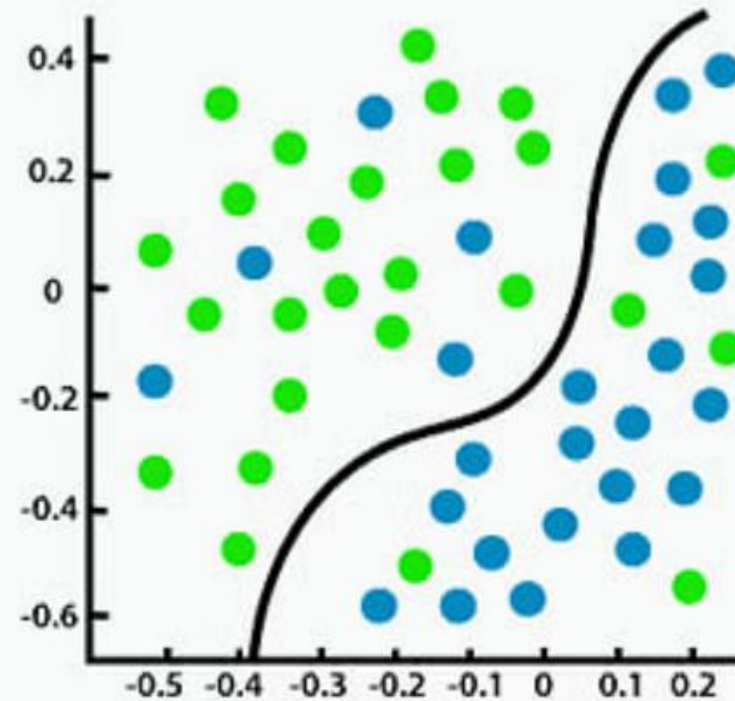






**Regression**

***versus***



**Classification**

# Example

## Regression

Feature	Description
Size (sqft)	The size of the house in square feet
Bedrooms	The number of bedrooms in the house
Bathrooms	The number of bathrooms in the house
Price (USD)	The price of the house

Size (sqft)	Bedrooms	Bathrooms	Price (USD)
1500	3	2	250000
2000	4	3	350000
1200	2	1	180000
...	...	...	...

## Classification

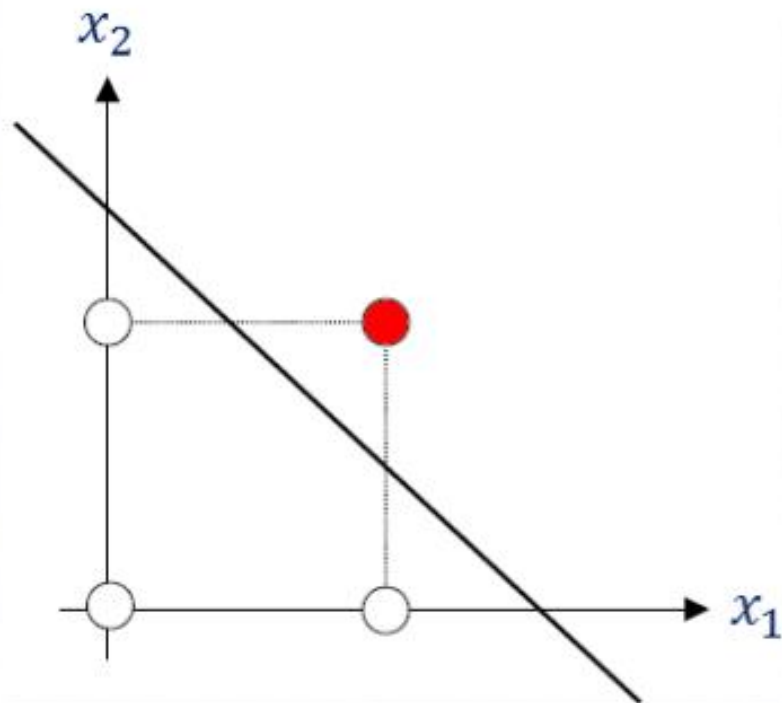
Feature	Description
Words	The words present in the email
Length	The length of the email
HasAttachment	Whether the email has an attachment or not
Label	Spam or Not Spam

Words	Length	HasAttachment	Label
"free", "money"	100	1	Spam
"hello", "friend"	50	0	Not Spam
"buy", "now"	200	1	Spam
...	...	...	...

- For simple **logic gate** problems, decision boundaries between classes are **linear**:
- Decision boundary:  $x_1w_1 + x_2w_2 - \theta = 0$

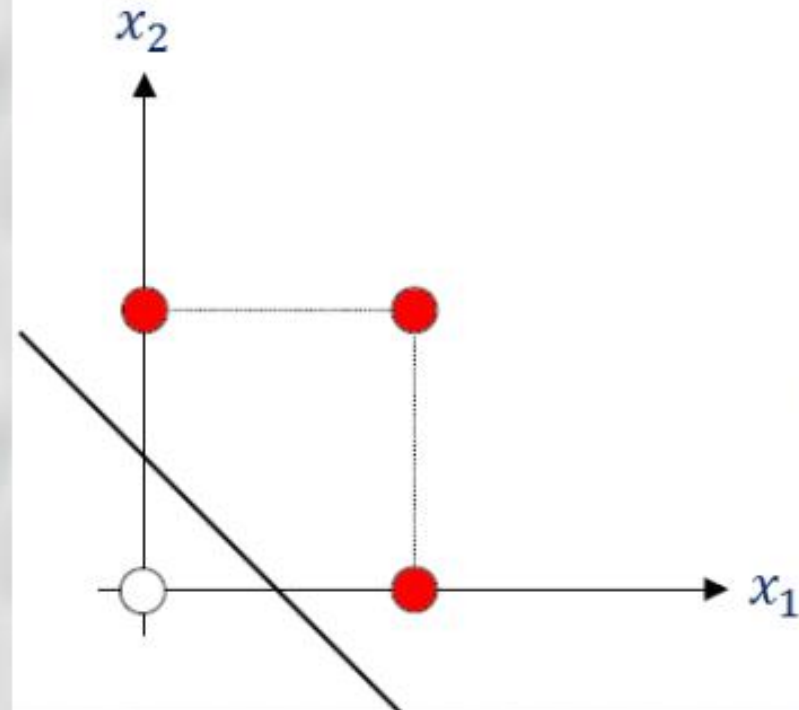
### AND

$$w_1 = 1, w_2 = 1, \theta = 1.5$$

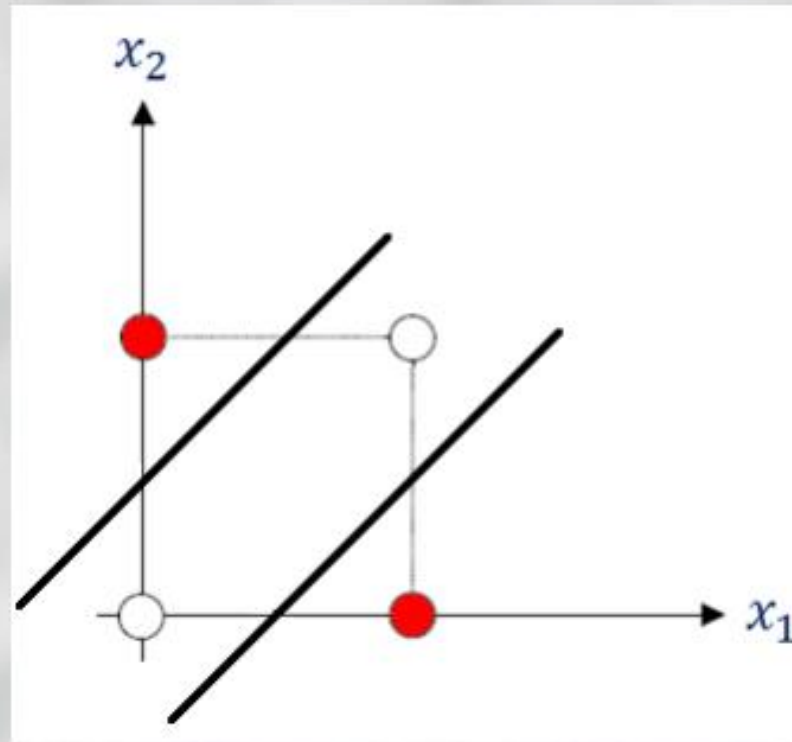


### OR

$$w_1 = 1, w_2 = 1, \theta = 0.5$$

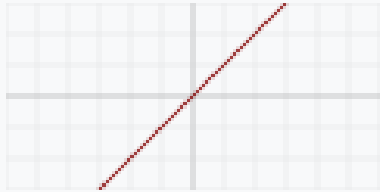
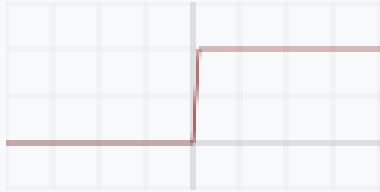
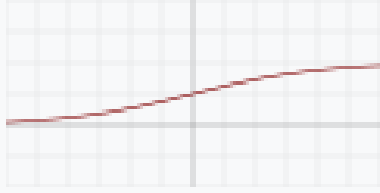
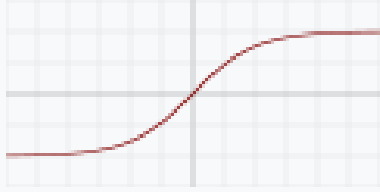




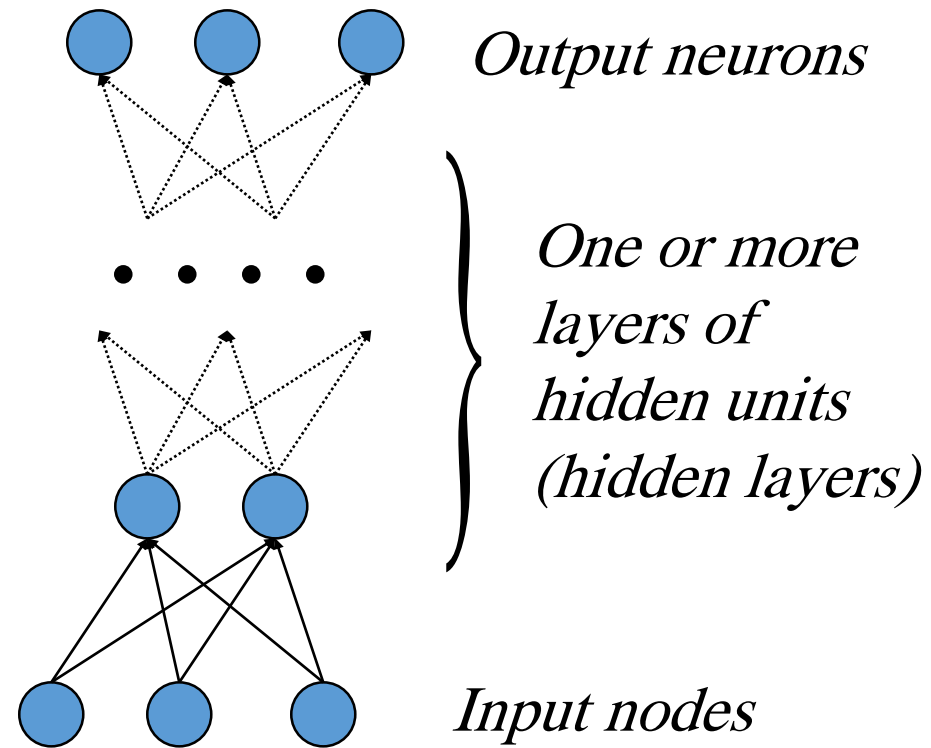


- For **XOR**, there are **two** obvious remedies:
  - Either change **activation function** so that it has more than one **decision boundary**
  - Use a more **complex network** that is able to generate more **complex decision boundaries**

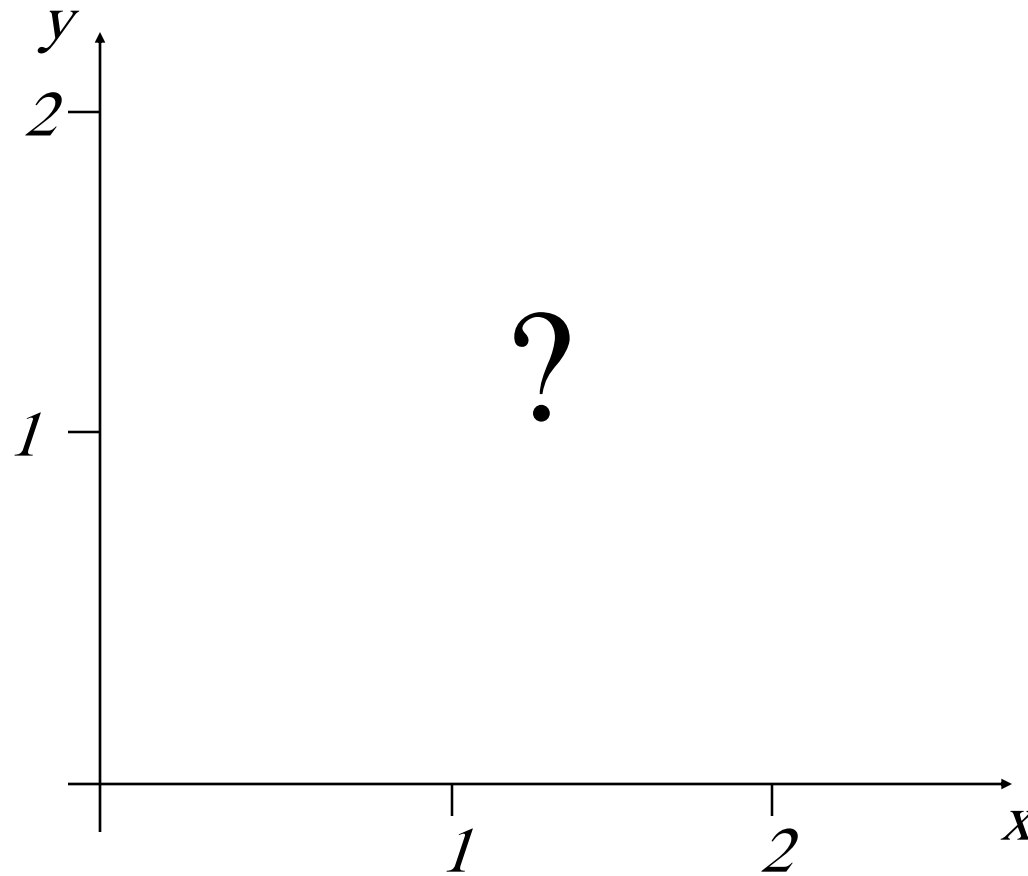
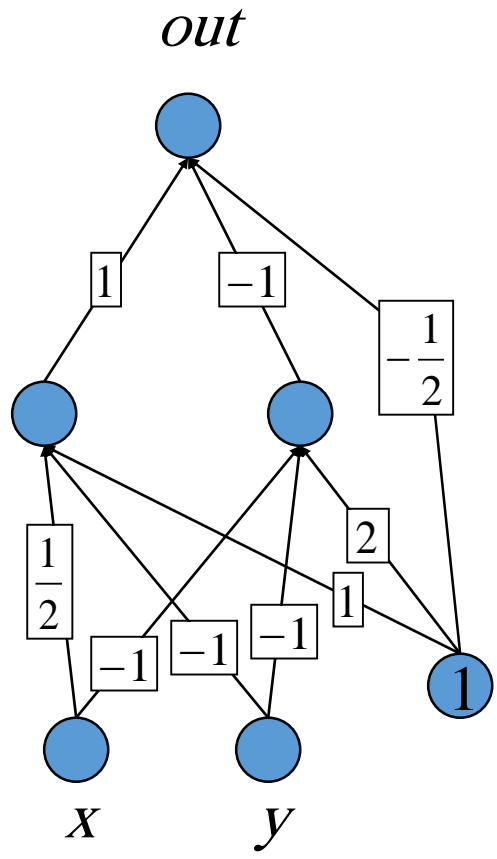
# Activation function

Name	Plot	Function, $g(x)$	Derivative of $g$ , $g'(x)$	Range
Identity		$x$	1	$(-\infty, \infty)$
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	0	$\{0, 1\}$
Logistic, sigmoid, or soft step		$\sigma(x) \doteq \frac{1}{1 + e^{-x}}$	$g(x)(1 - g(x))$	$(0, 1)$
Hyperbolic tangent (tanh)		$\tanh(x) \doteq \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - g(x)^2$	$(-1, 1)$

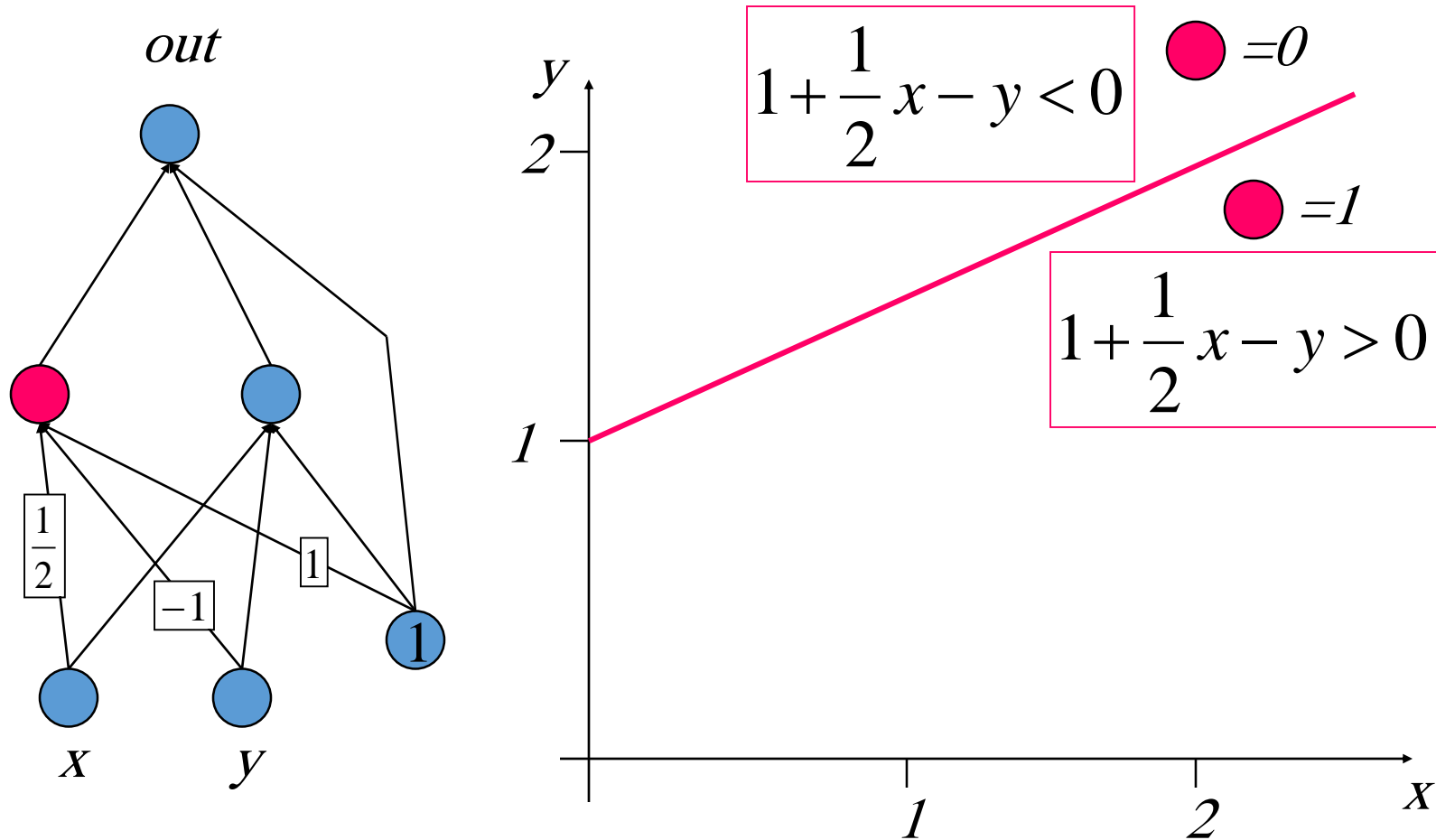
# Multilayer Perceptron



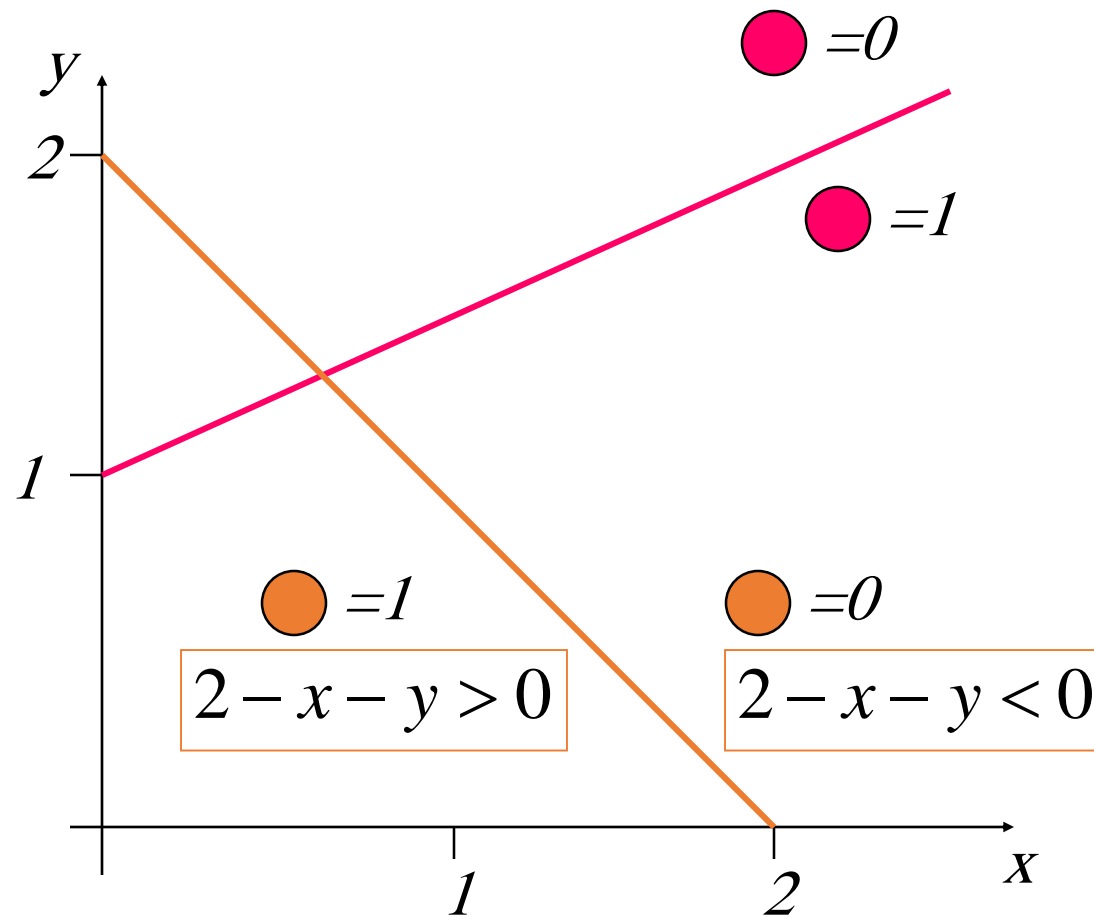
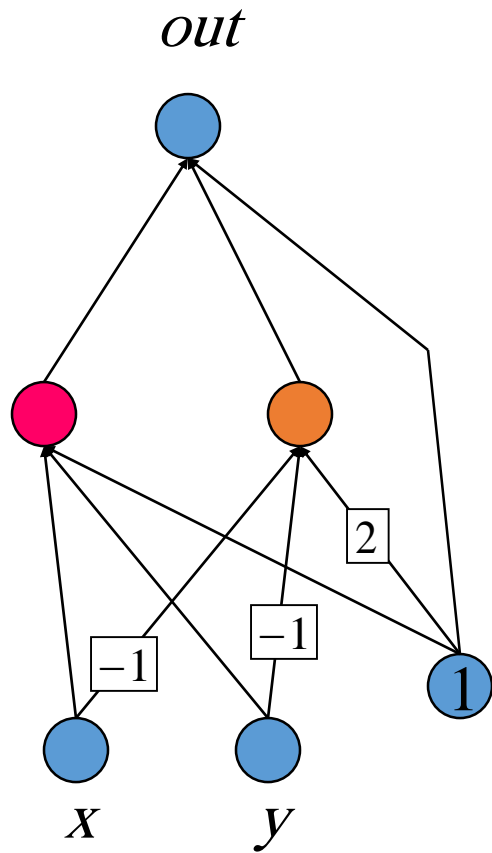
## Example: Perceptrons as Constraint Satisfaction Networks



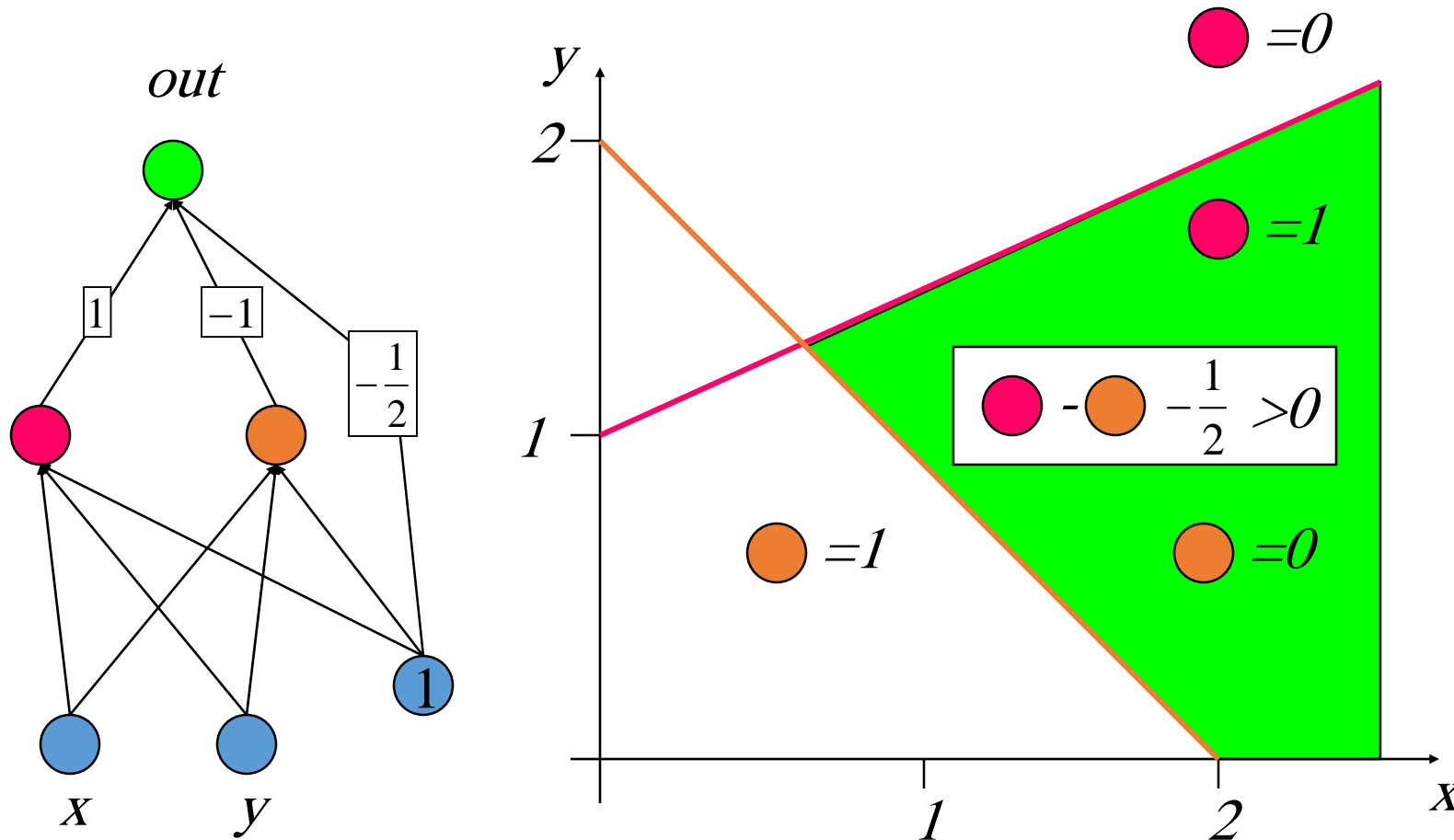
## Example: Perceptrons as Constraint Satisfaction Networks



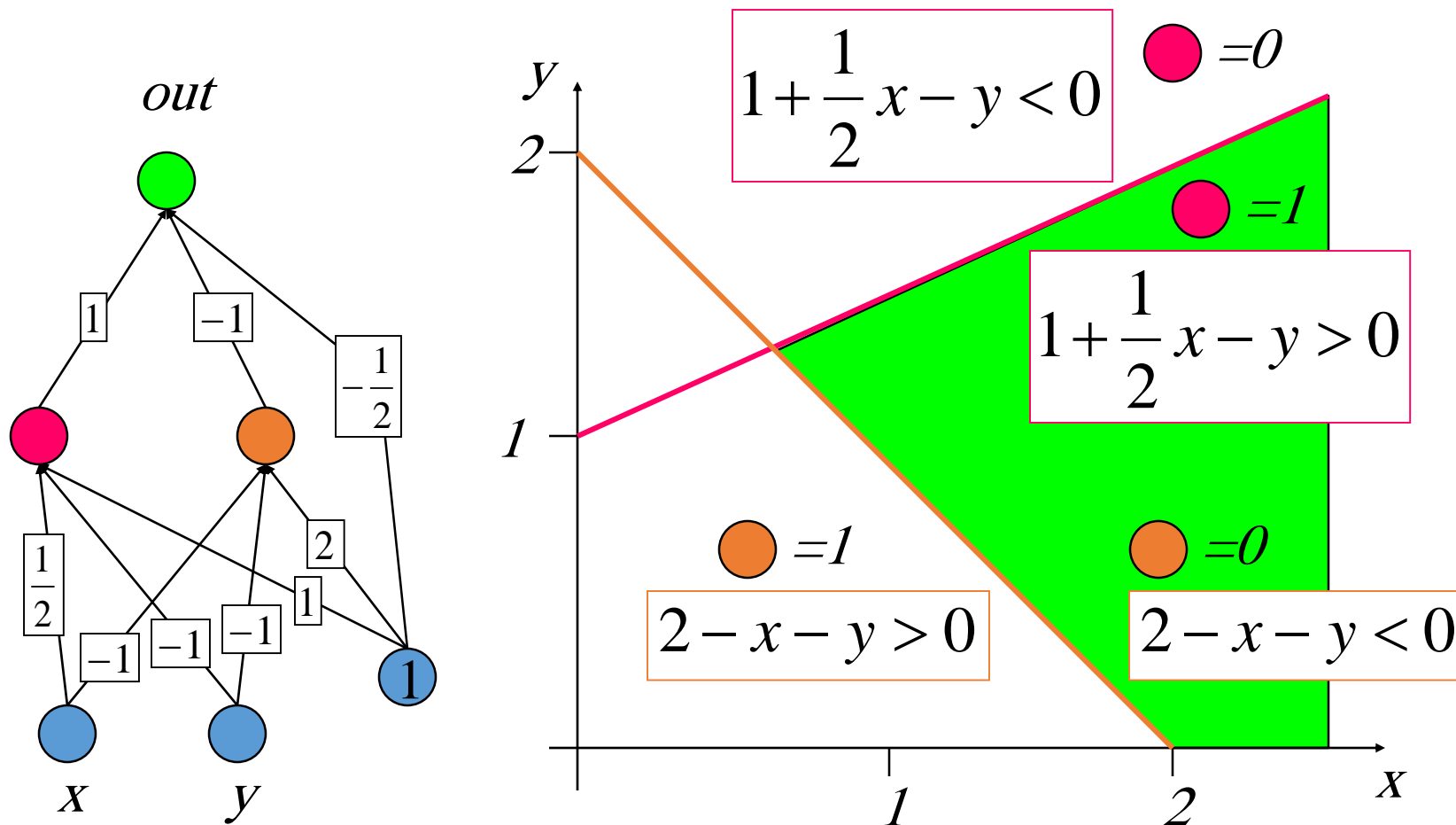
## Example: Perceptrons as Constraint Satisfaction Networks



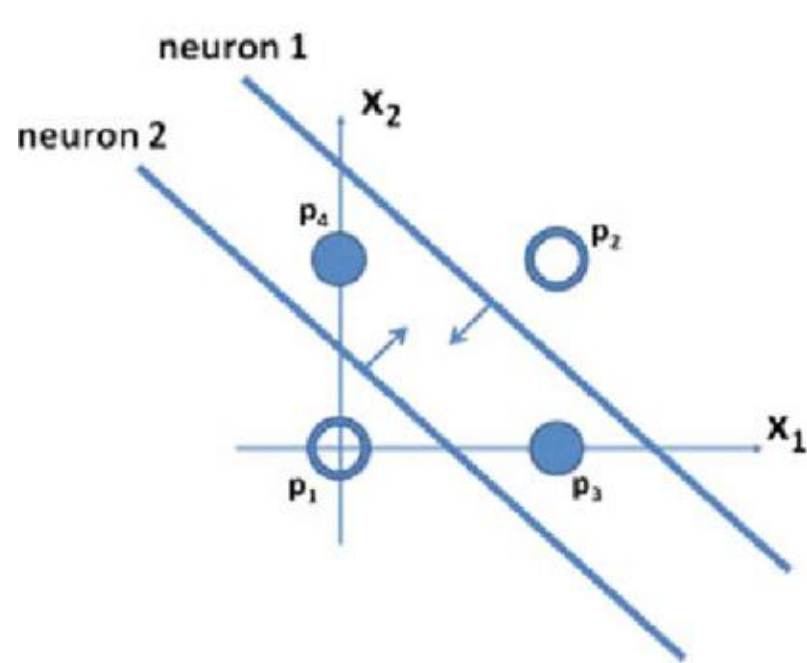
## Example: Perceptrons as Constraint Satisfaction Networks



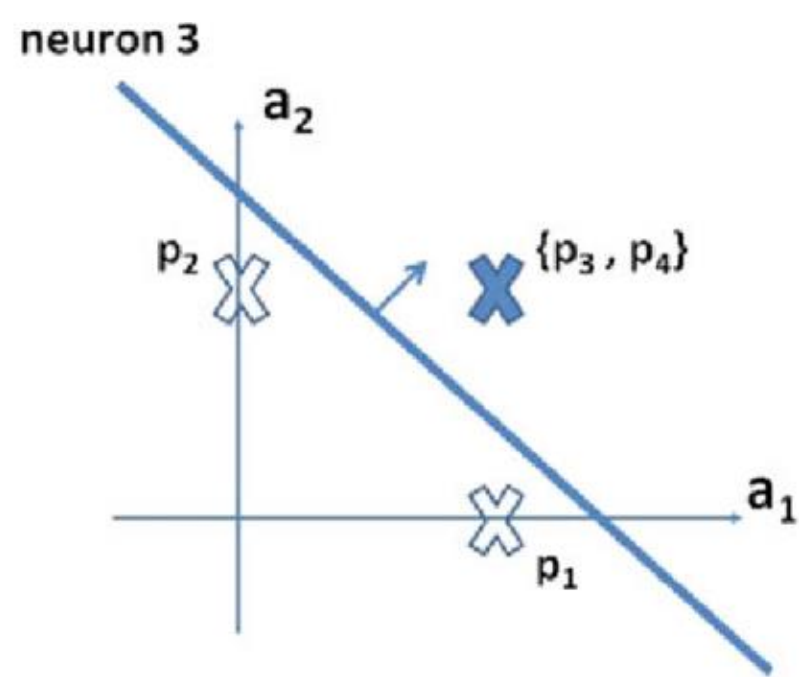
# Perceptrons as Constraint Satisfaction Networks



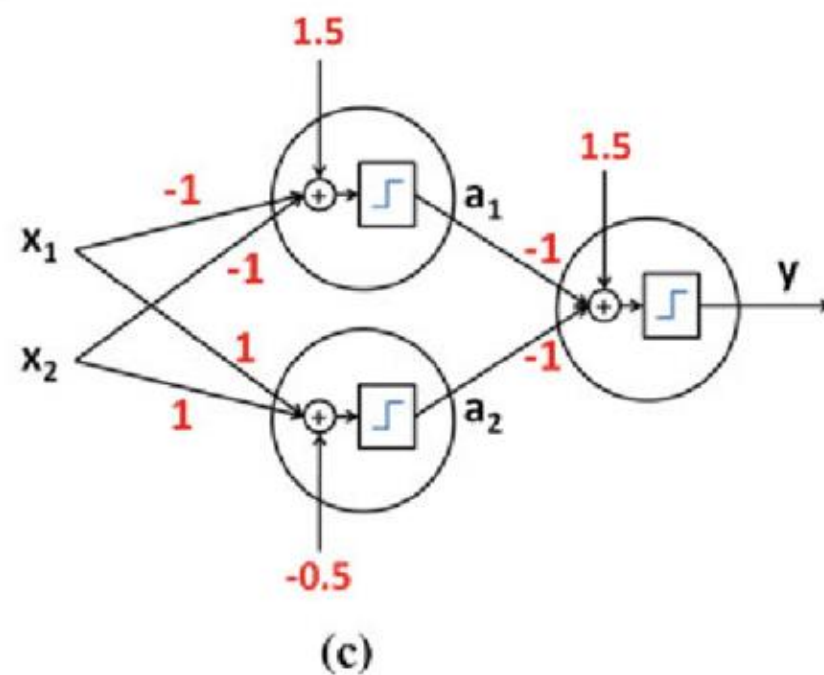




(a)

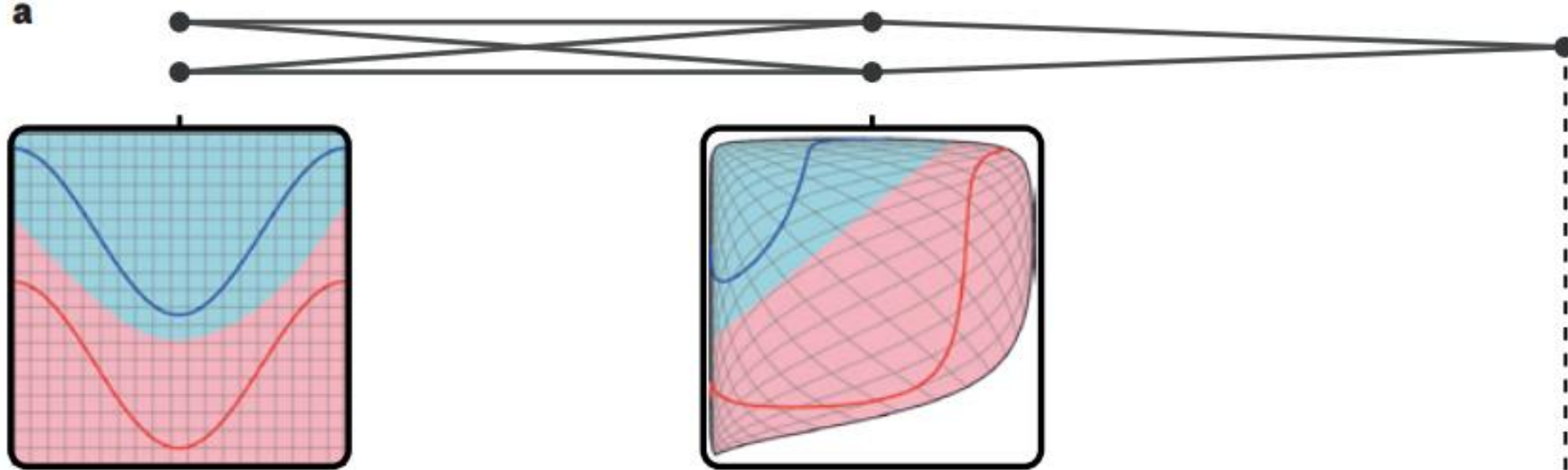


(b)



(c)

**a**



Playground:

<https://deeperplayground.org/>