DMC ONLINE

#YoMeQuedoEnCasa

Deep Learning & Web Scraping and Text Mining

PhD. Edwyn Javier Aldana Bobadilla

DMC Perú

Reglas Básicas

Puntualidad

Mantener silenciado el micrófono durante la sesión

Las preguntas se realizarán por el chat/ en caso sea necesario se habilita el micrófono

Realizar las actividades encomendadas

Dinámica de las sesiones

7:00 PM – 8:30 PM	***Parte 1***
8:30 PM – 8:40 PM	***Receso***
8:40 PM – 10:00 PM	***Parte 2***

Los recursos (scripts, datasets, presentaciones) serán compartidos en la nube, en un URL habilitado por el profesor.

Calificaciones

Asistencia (Curso) mínimo 80% sesiones para recibir la certificación

Exámen Final (40%) + ***Trabajo final*** (60%)

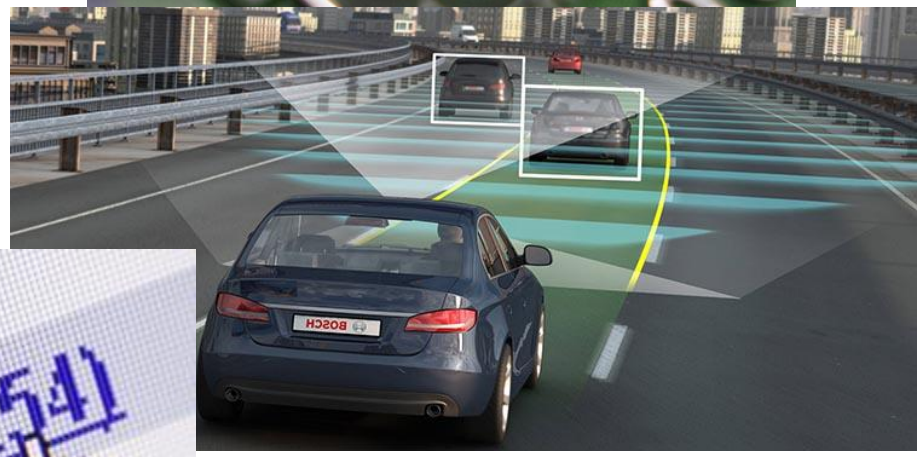
Preliminares

Aprendizaje Automático

Técnicas para crear **modelos computacionales** con capacidad de decisión basada en la **experiencia**.

Ejemplos:

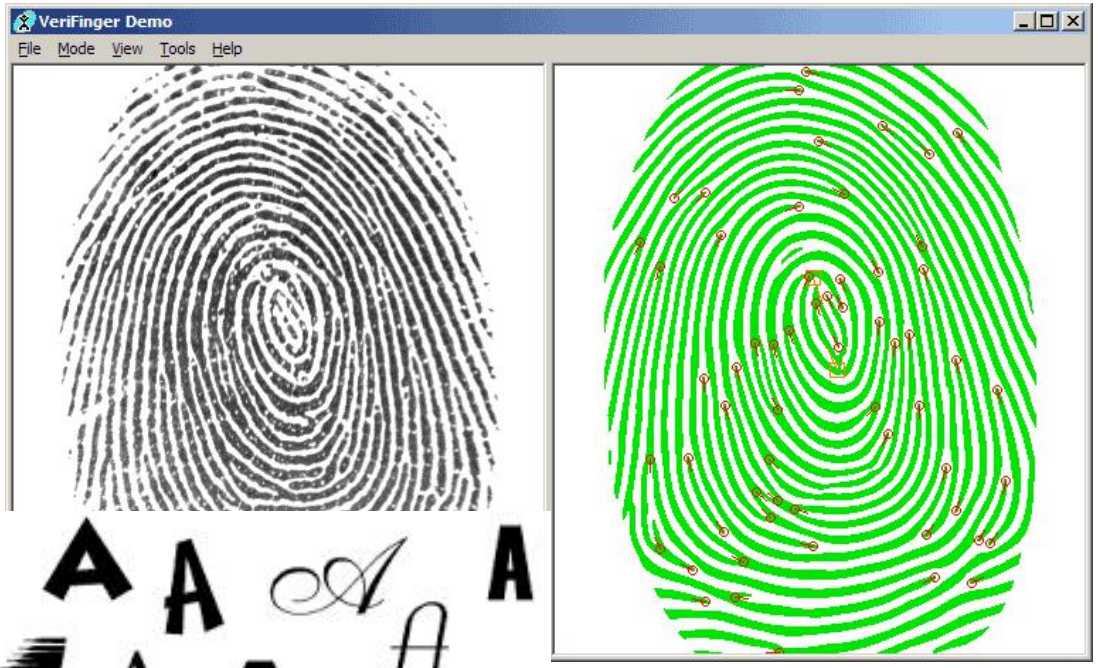
- Detección de transacciones fraudulentas en tarjetas de crédito.
- Correo no deseado
- Vehiculos autónomos

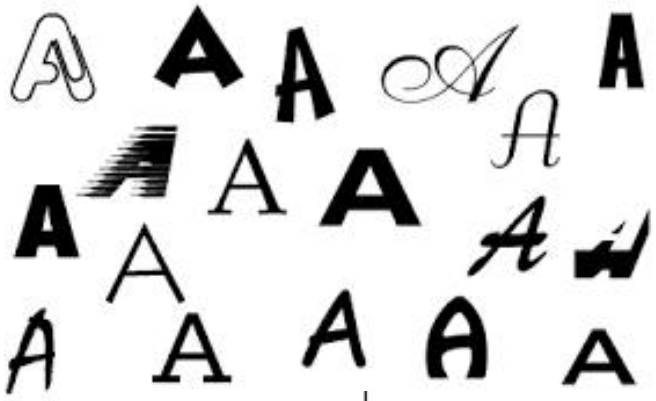


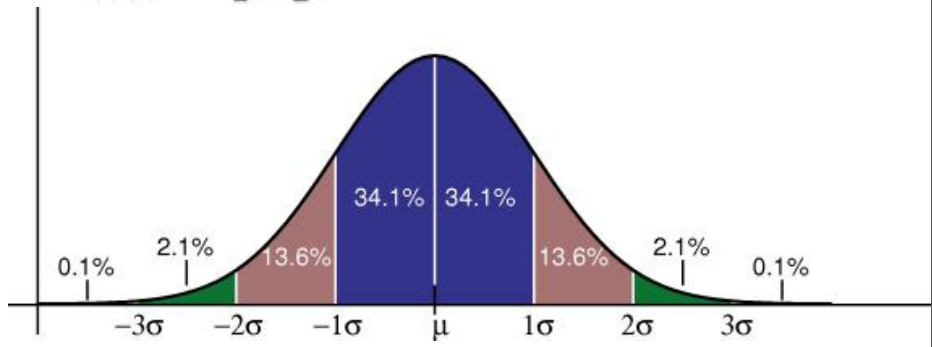
Aprendizaje Automático

Areas relacionadas:

- Reconocimiento de patrones.
- Teoria del aprendizaje
- Estadística
- Probabilidad







Elementos del aprendizaje automático

1) Problema o tarea a resolver (T)

I. Jugar ajedrez.
II. Reconocer caracteres escritos a mano.
III. Conducir un automóvil a través de una autopista.
IV. Realizar el estudio de un crédito bancario a un cliente.
V. Reconocer discurso ofensivo o que incite a la violencia

Elementos del aprendizaje automático

2) Conocimiento previo o experiencia (E)

I. Conjunto de todos los posibles estados del tablero de ajedrez.

II. Muestra de **imágenes** con caracteres escritos a mano.

III. Secuencia de **imágenes** del entorno asociadas a acciones ejecutadas por un conductor humano (complejo).

IV. Información de ingreso, egresos y comportamiento financiero de un conjunto de clientes.

V. Corpus conteniendo **sentencias** del discurso, o un conjunto de **audios** conteniendo dichas sentencias.

Elementos del aprendizaje automático

3) Medida de desempeño (P)

I. Número de juegos ganados por contrincantes humanos.
II. Porcentaje de caracteres “bien clasificados”.
III. Tiempo o distancia recorrida antes que un error ocurra (observado y corregido por intervención humana).
IV. Porcentaje de cartera morosa.
V. Efectividad en la predicción de conductas violentas.

Representación de la experiencia (E)

- Las computadoras trabajan en un **espacio numérico finito**
- Es importante que la información relacionada con la experiencia ***E*** sea mapeada a dicho espacio numérico.
- ***E*** incluirá aquellos atributos o propiedades importantes del problema a ser resuelto (**Extracción de Características**).
- Generalmente ***E*** se representa como un conjunto de vectores de la forma:

$$\vec{x} = [x_1, x_2, \ldots x_n] \in \Re^n$$

Representación de la experiencia (E)

Ejemplo:

- Problema (T): Reconocer las frutas en la imágen:

Representación de la experiencia (E)

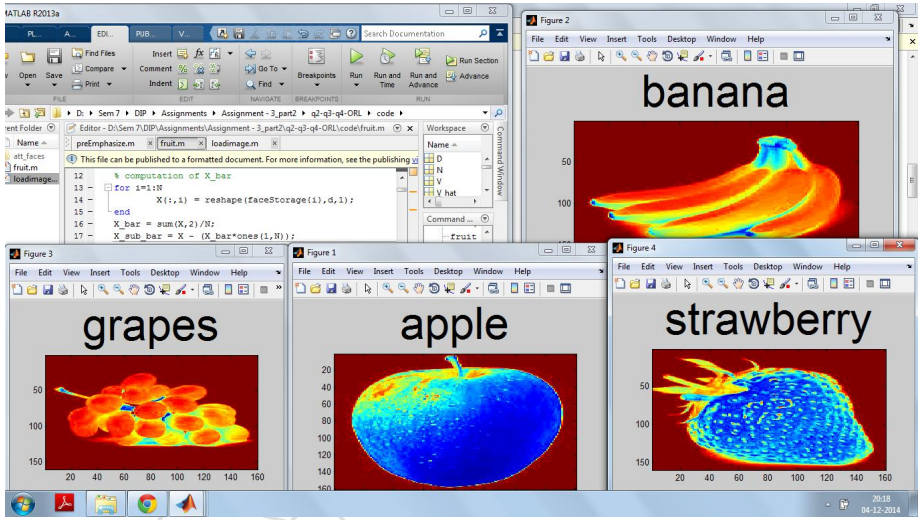
Ejemplo:

- Problema (T): Reconocer las frutas en la imágen
- Experiencia (E): ?

Representación de la experiencia (E)

Ejemplo 1:

- Problema (T): Reconocer las frutas en la imágen
- Experiencia (E): ?



Representación de la experiencia (E)

Ejemplo1 :

- Problema (T): Reconocer las frutas en la imágen
- Experiencia (E):

Extracción de características a partir de los píxeles de la imágen.

255	0	137	137	137	137	0
0	128	255	128	137	255	137
128	0	0	64	128	64	64
128	128	0	255	137	255	0
0	255	128	137	137	137	0
128	137	137	137	0	255	64
255	128	128	128	128	64	64

Representación de la experiencia (E)

Ejemplo 2:

- Problema (T): Realizar estudio de crédito

Representación de la experiencia (E)

Ejemplo 2:

- Problema (T): Realizar estudio de crédito.
- Experiencia (E):

$$\vec{x} = [x_1, x_2, \ldots x_n] \in \Re^n$$

Ingresos

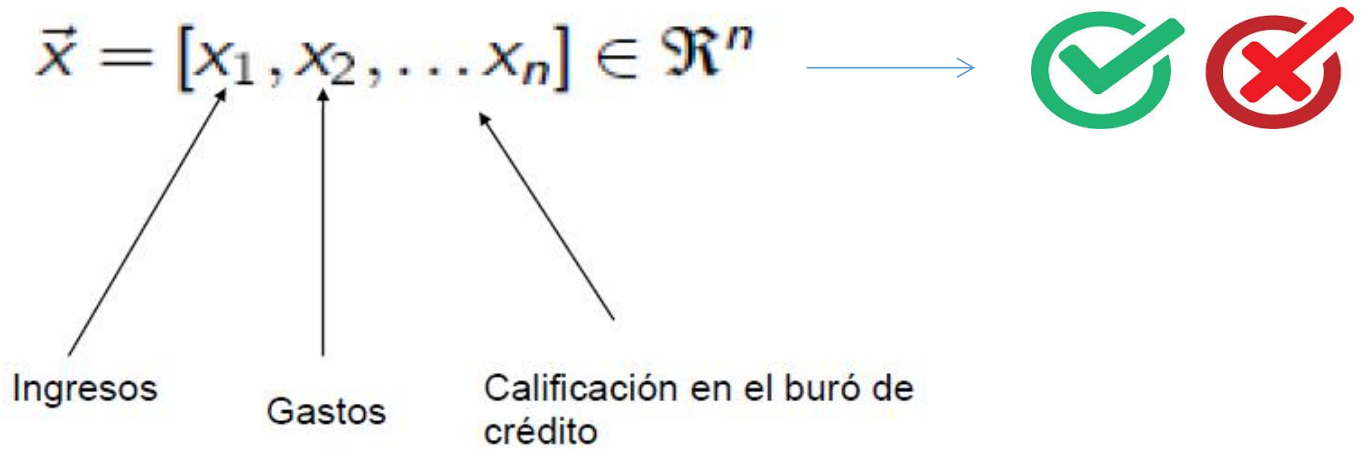
Gastos

Calificación en el buró de crédito

Representación de la experiencia (E)

Ejemplo 2:

- Problema (T): Realizar estudio de crédito.
- Experiencia (E):



Representación de la experiencia (E)

Ejemplo 2:

- Problema (T): Realizar estudio de crédito.
- Experiencia (E):

$$\vec{x} = [x_1, x_2, \ldots x_n] \in \Re^n$$

Ingresos

Gastos

Calificación en el buró de crédito

$\vec{x}$			y
4.9	3.0	1.4	1
4.7	3.2	1.3	0
4.6	3.1	1.5	0
5.0	3.6	1.4	1
5.4	3.9	1.7	1
4.6	3.4	1.4	1
5.0	3.4	1.5	0
4.4	2.9	1.4	1

Es este tipo de problemas usualmente se tienen las caracteristicas, no se requiere el proceso de **extracción**, solo **selección**.

Representación de la experiencia (E)

Ejemplo 3:

- Problema (T): Determinar si un correo electrónico es no deseado SPAM
- Experiencia (E): Miles de correos electónicos categorizados como **spam y no spam**.

Representación de la experiencia (E)

Ejemplo 3:

- Problema (T): Determinar si un correo electrónico es no deseado SPAM
- Experiencia (E): Conjunto de palabras de correos SPAM y NO-SPAM

En este caso la **extracción de características** no es un problema trivial.

Primera aproximación al aprendizaje

- Problema (T): Realizar estudio de crédito
- Experiencia (E): Atributos de clientes en el espacio numérico.

$\vec{x}$			y
4.9	3.0	1.4	1
4.7	3.2	1.3	0
4.6	3.1	1.5	0
5.0	3.6	1.4	1
5.4	3.9	1.7	1
4.6	3.4	1.4	1
5.0	3.4	1.5	0
4.4	2.9	1.4	1

- Debemos encontrar un modelo ***M*** de la forma:

$$Y = F(X)$$

Tipos de aprendizaje

Aprendizaje

- Encontrar un modelo del problema basado en la **experiencia**.
- Como se mencionó la experiencia se debe presentar a la computadora codificada en un espacio numérico.
- La estructura más común es a través de un vector denomidado **vector de características** o **patrón** de la forma:

$$\vec{x} = [x_1, x_2, ..., x_n]$$

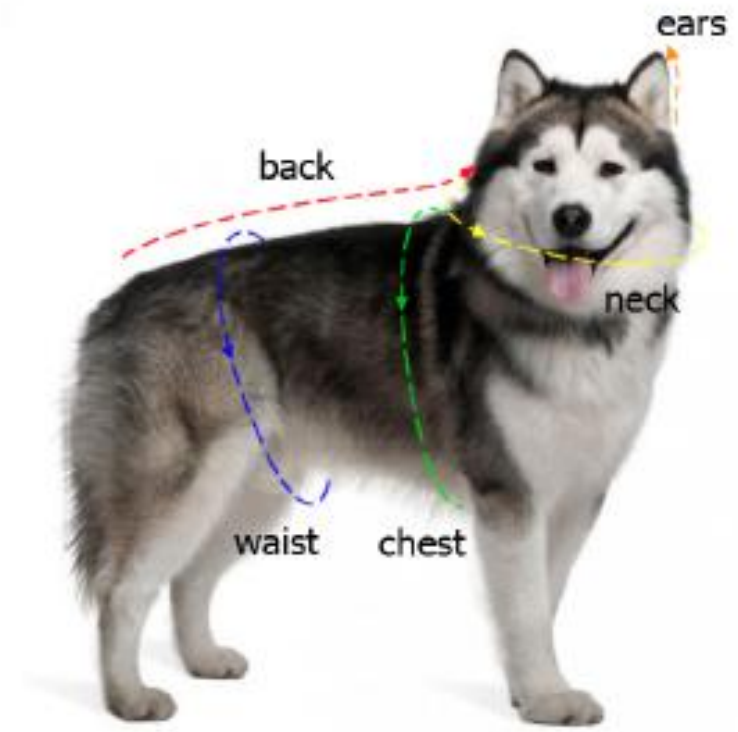
- En tal caso la experiencia es un conjunto de instancias de ***x*** denominadas en común como **dataset.**

Aprendizaje

En general las tareas de aprendizaje se clasifican en :

- Aquellas donde se busca un modelo basado en información a priori de la variable de decisión.
- Aquellas donde el modelo debe inferir el valor de la variable de decisión sin un conocimiento previo.
- En el primer caso hablamos de aprendizaje supervisado.
- En el segundo caso hablamos de aprendizaje no supervisado

Aprendizaje



$$\vec{x} = \begin{bmatrix} 120 & 80 & 50 & 60 & 10 \end{bmatrix}$$

Aprendizaje

$\vec{x} =$ | 120 | 80 | 50 | 60 | 10 | Malamute |

$\vec{x} =$ | 30 | 20 | 10 | 5 | 5 | Chihuahua |

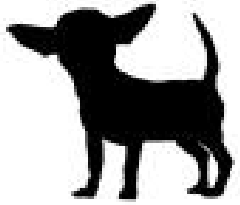
$\vec{x} =$ | 100 | 60 | 40 | 50 | 10 | Pitbull |

$\vec{x} =$ | 80 | 50 | 40 | 45 | 15 | Bull dog |

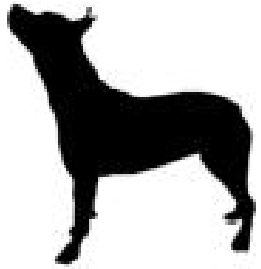
Aprendizaje

$\vec{x} =$ | 120 | 80 | 50 | 60 | 10 | ? |

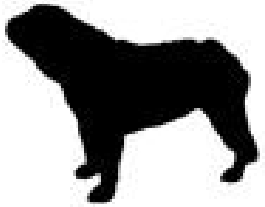
$\vec{x} =$ | 30 | 20 | 10 | 5 | 5 | ? |



$\vec{x} =$ | 100 | 60 | 40 | 50 | 10 | ? |



$\vec{x} =$ | 80 | 50 | 40 | 45 | 15 | ? |

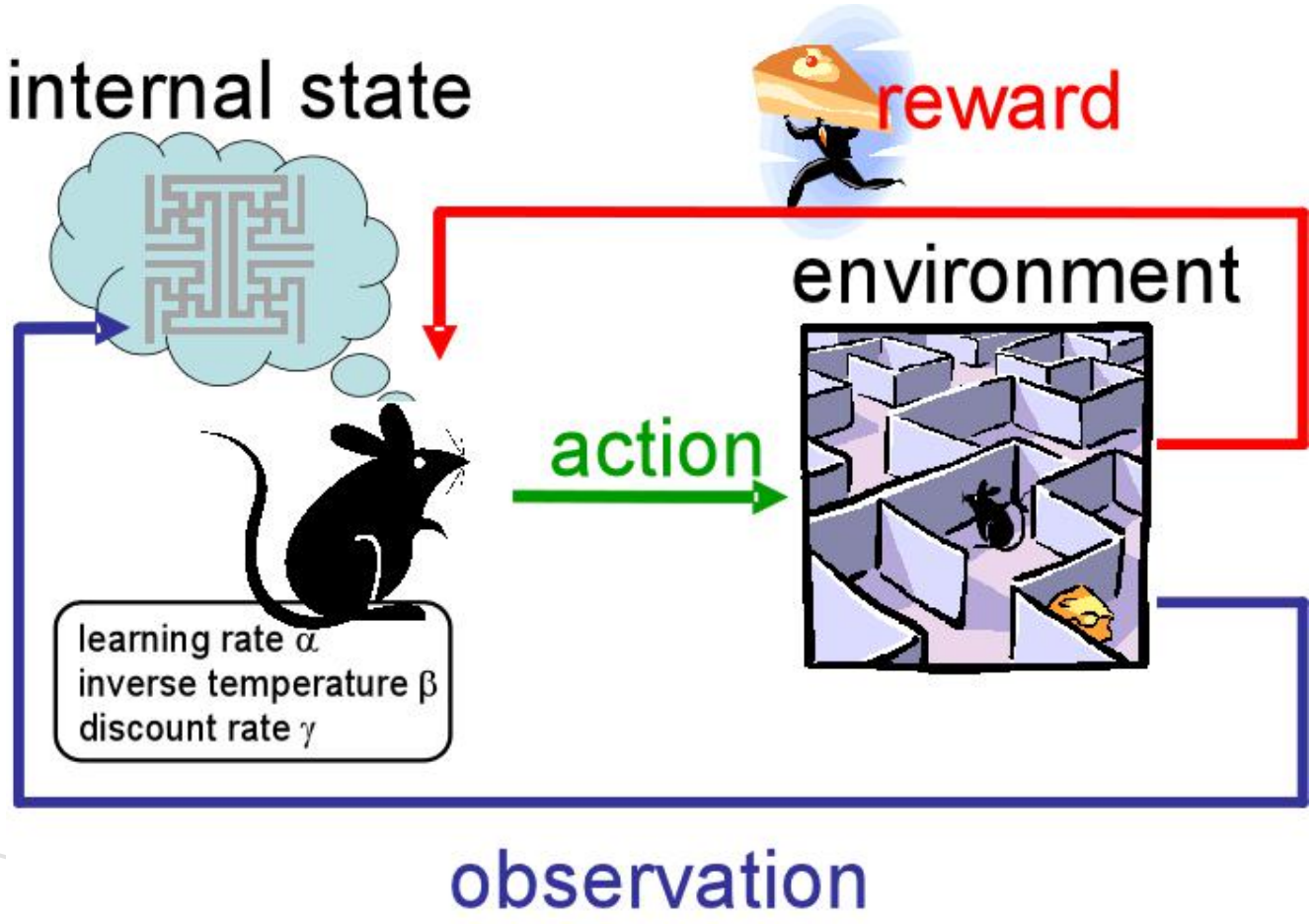


Aprendizaje

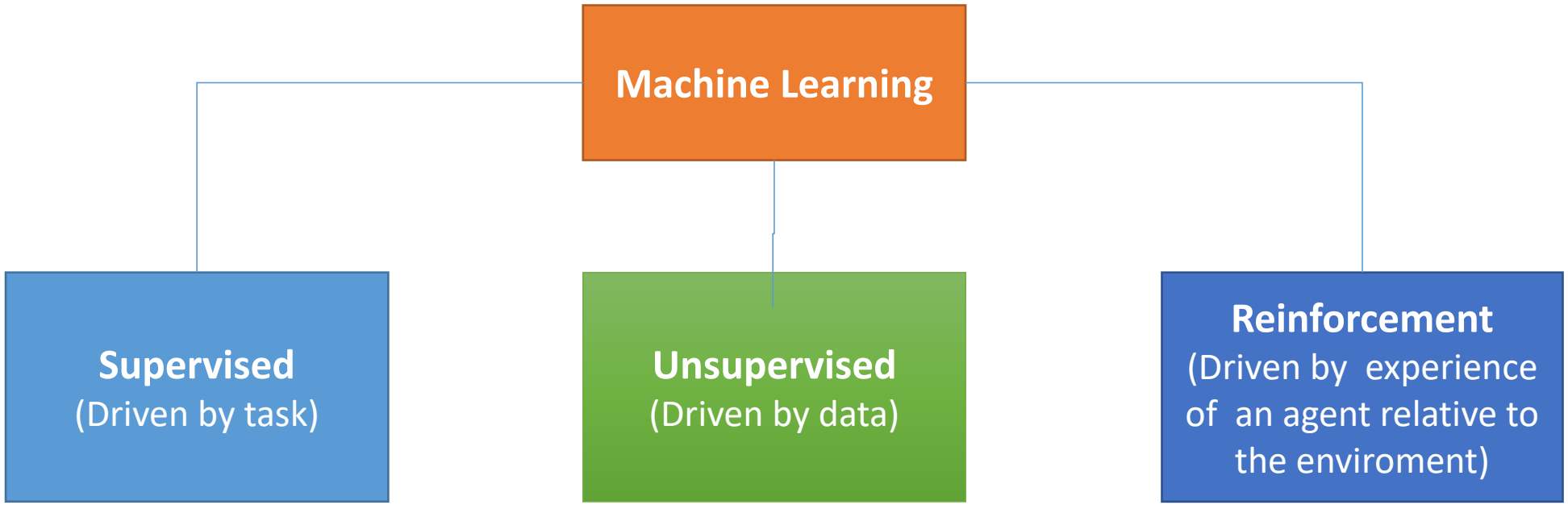
Supervised Learning

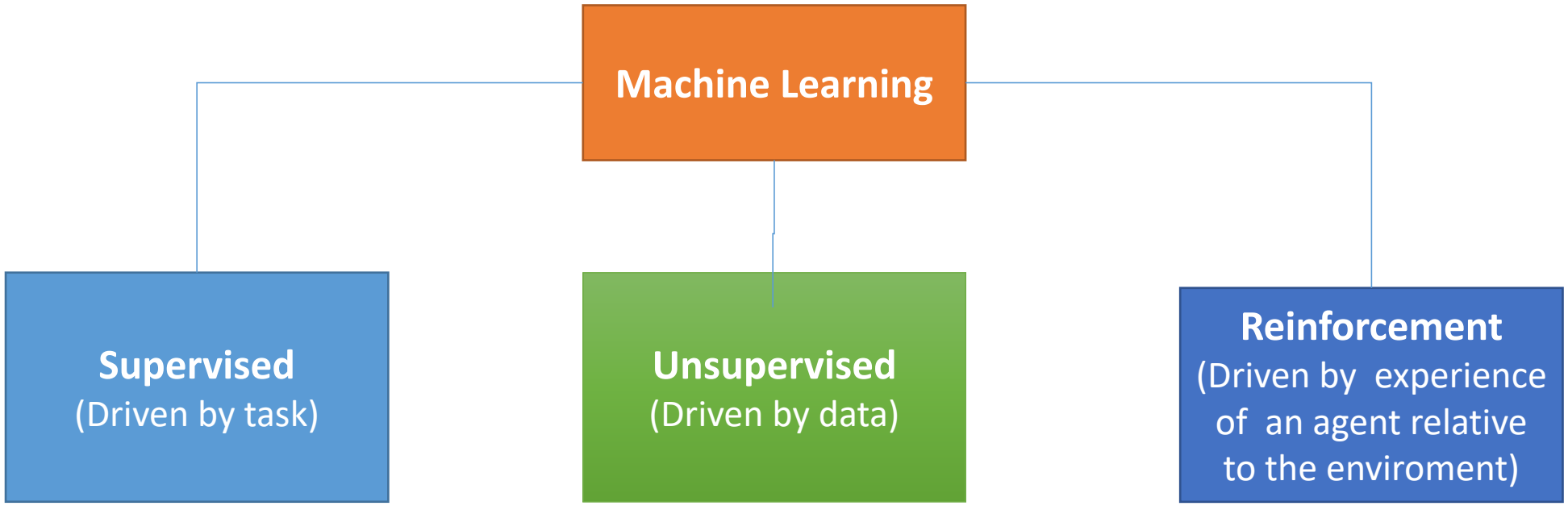
Unsupervised Learning

Otros tipos de aprendizaje



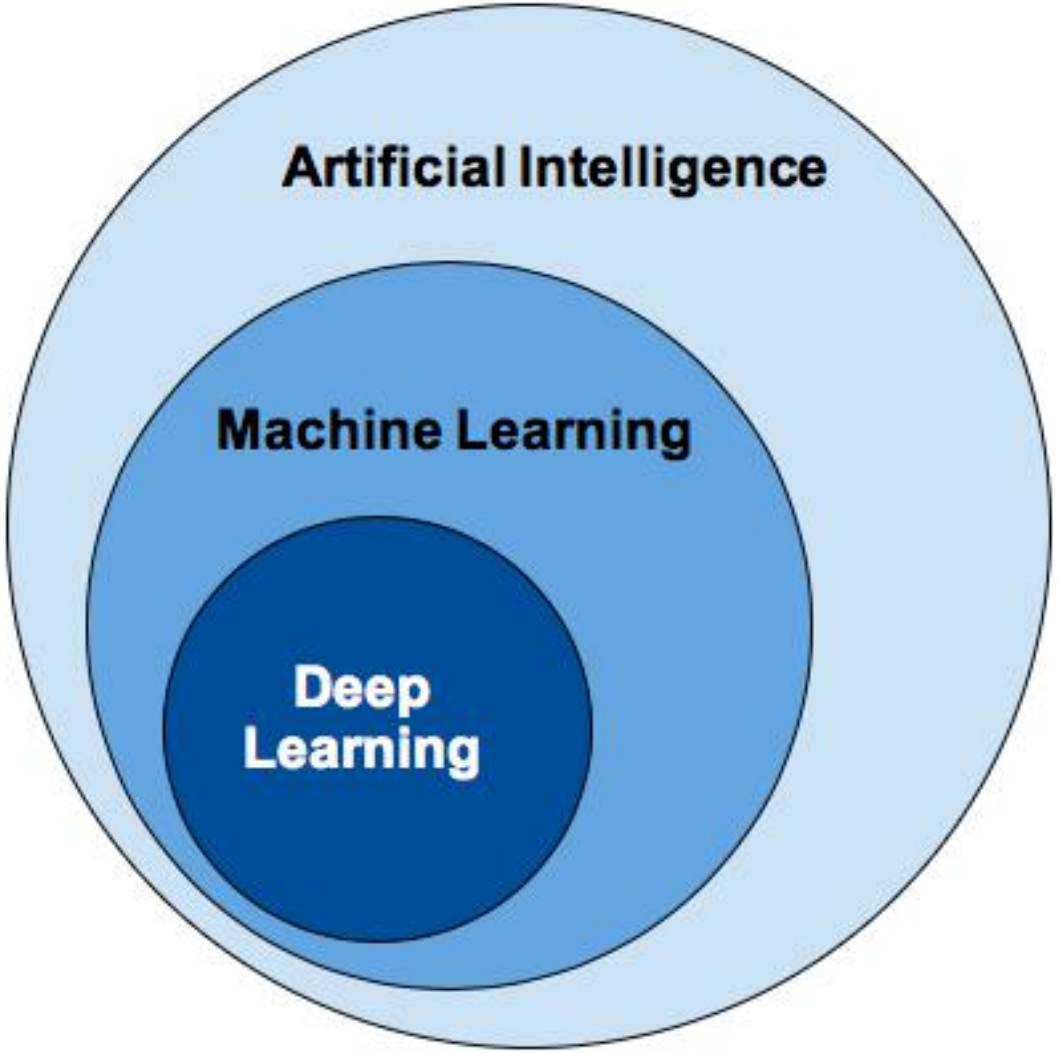
- Teoria de Juegos
- Optimización
- Investigación de operaciones


Machine Learning
Supervised
(Driven by task)
Unsupervised
(Driven by data)
Reinforcement
(Driven by experience of an agent relative to the enviroment)

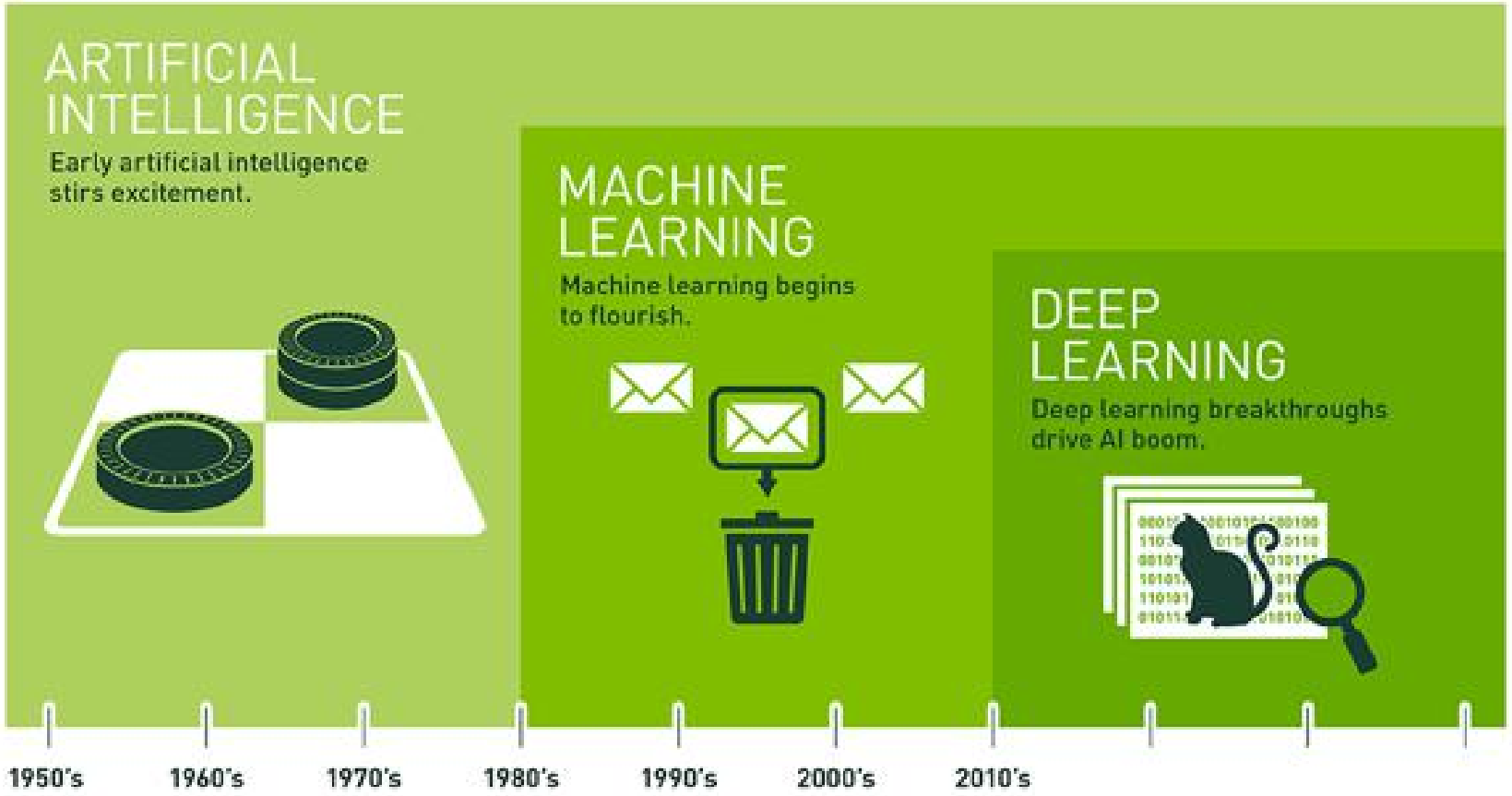


Y el deep learning?

Artificial Intelligence, Machine Learning and Deep Learning

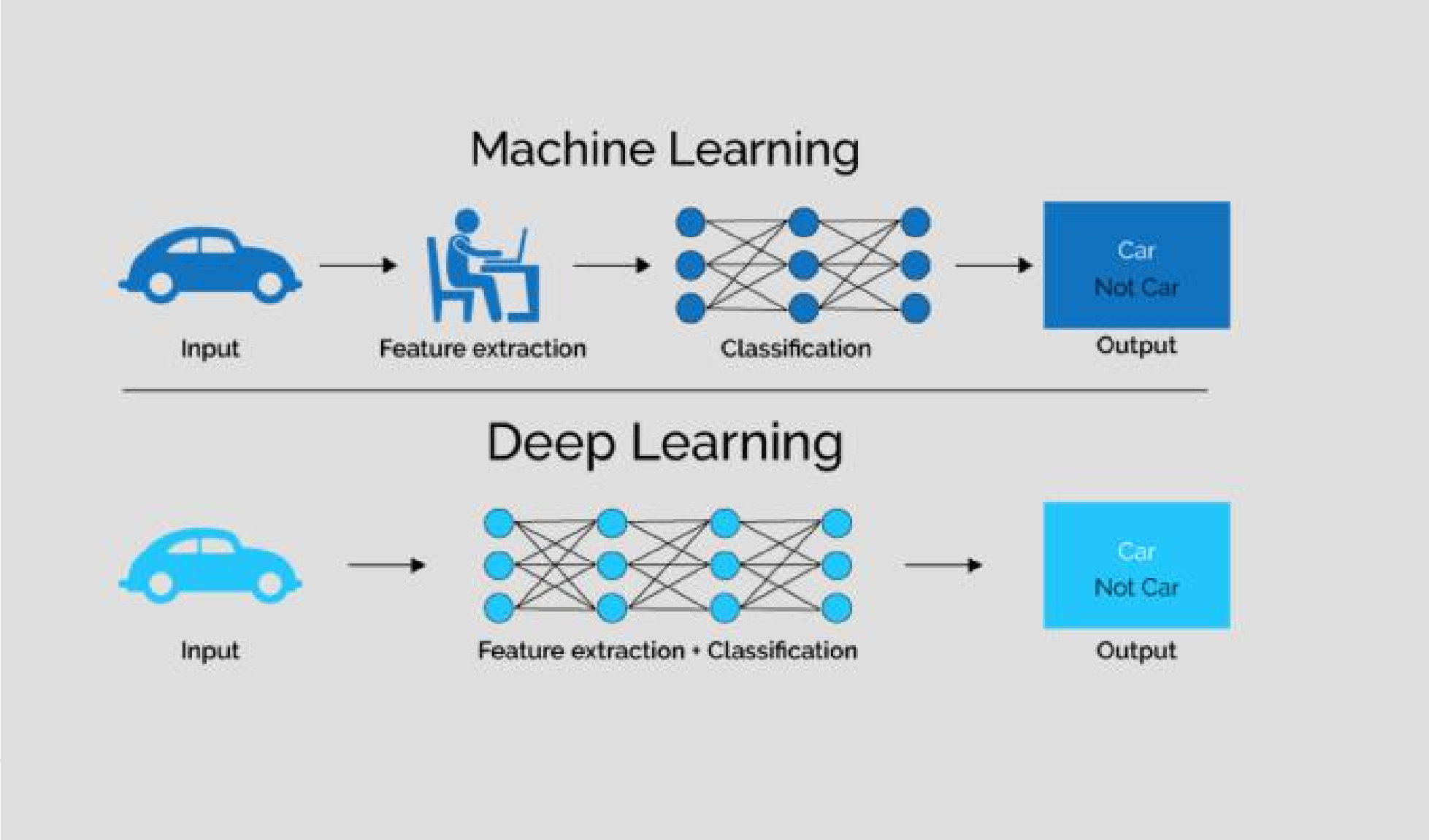


Evolution



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

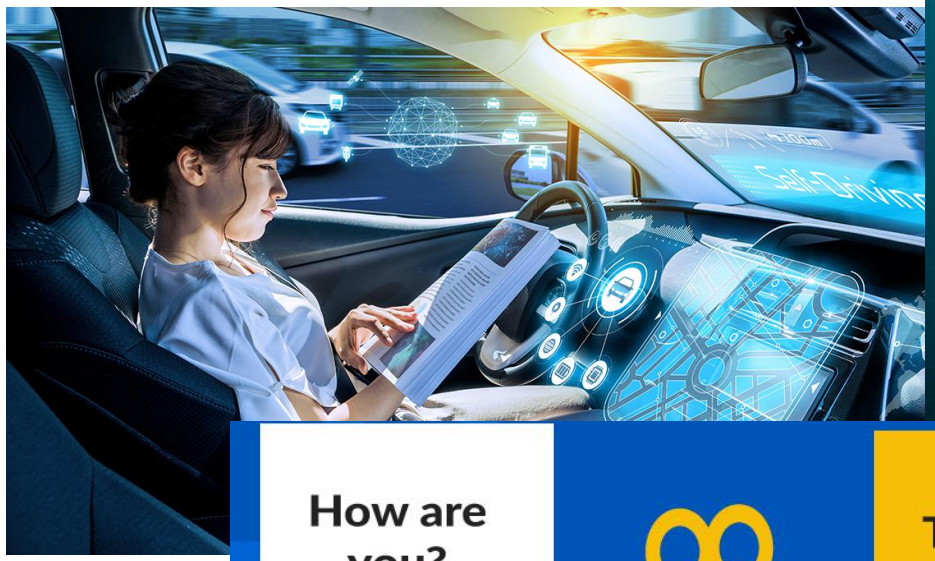
Machine Learning vs. Deep Learning

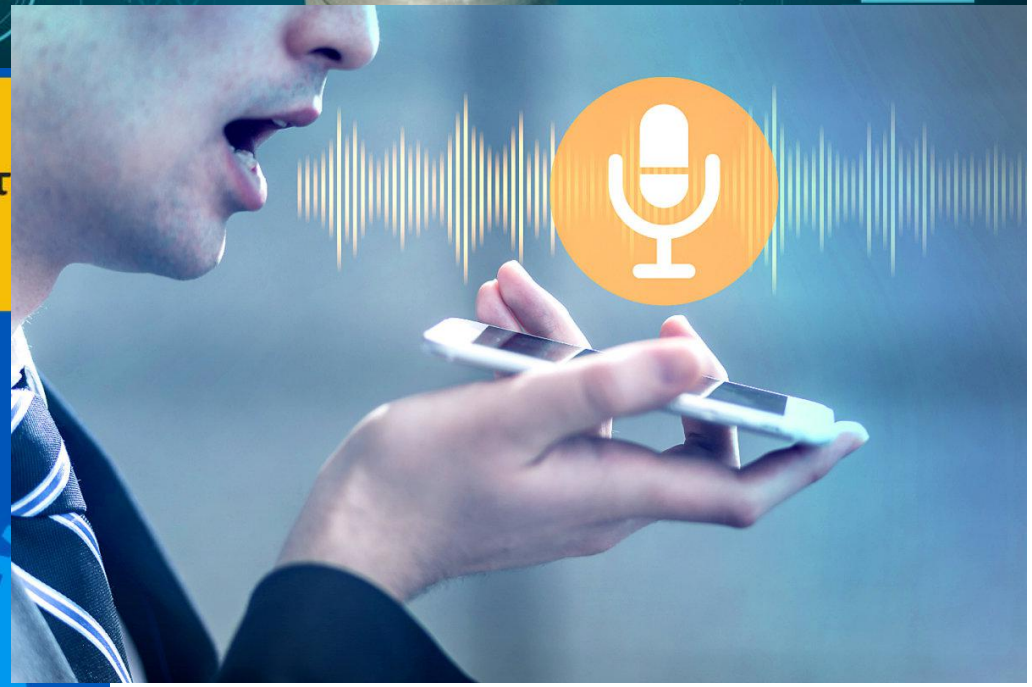


Common Machine Learning Applications

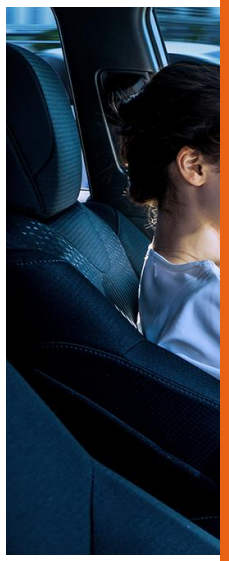
Retail	Marketing	Healthcare	Telco	Finance
• Demand forecasting • Supply chain optimization • Pricing optimization • Market segmentation and targeting • Recommendations	• Recommendation engines & targeting • Customer 360 • Click-stream analysis • Social media analysis • Ad optimization	• Predicting Patient Disease Risk • Diagnostics and Alerts • Fraud	• Customer churn • System log analysis • Anomaly detection • Preventative maintenance • Smart meter analysis	• Risk Analytics • Customer 360 • Fraud • Credit scoring

Deep Learning Applications



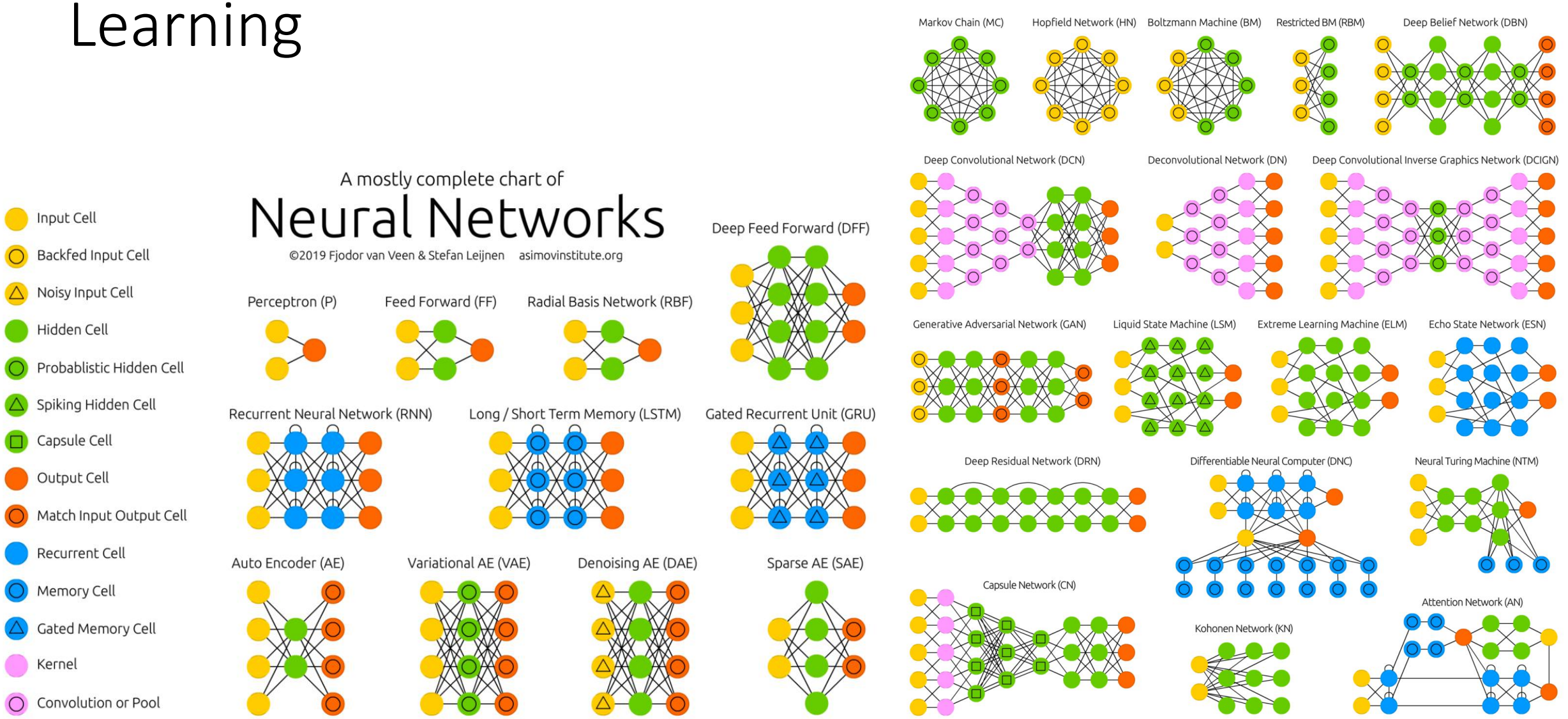


Deep Learning Applications



amazon go

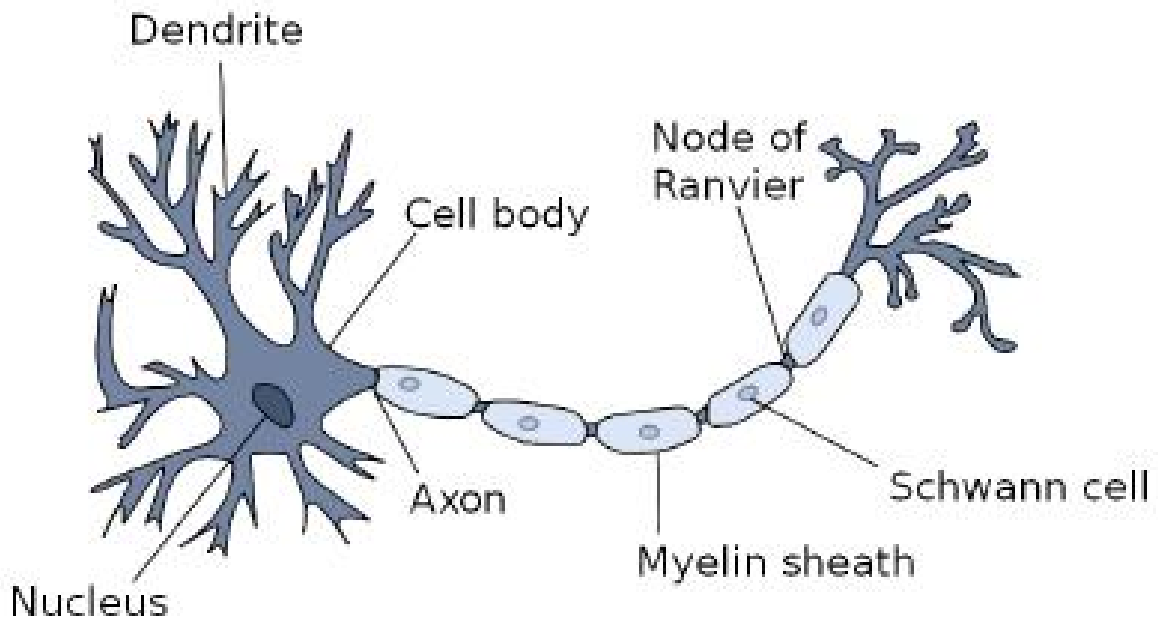
Neural Networks: The foundations of Deep Learning




DMC ONLINE
#YoMeCapacitoEnCasa
Neural Networks and Deep Learning
www.dmc.pe

What is a neuron?

- Typically a neuron is defined as an information processing unit.

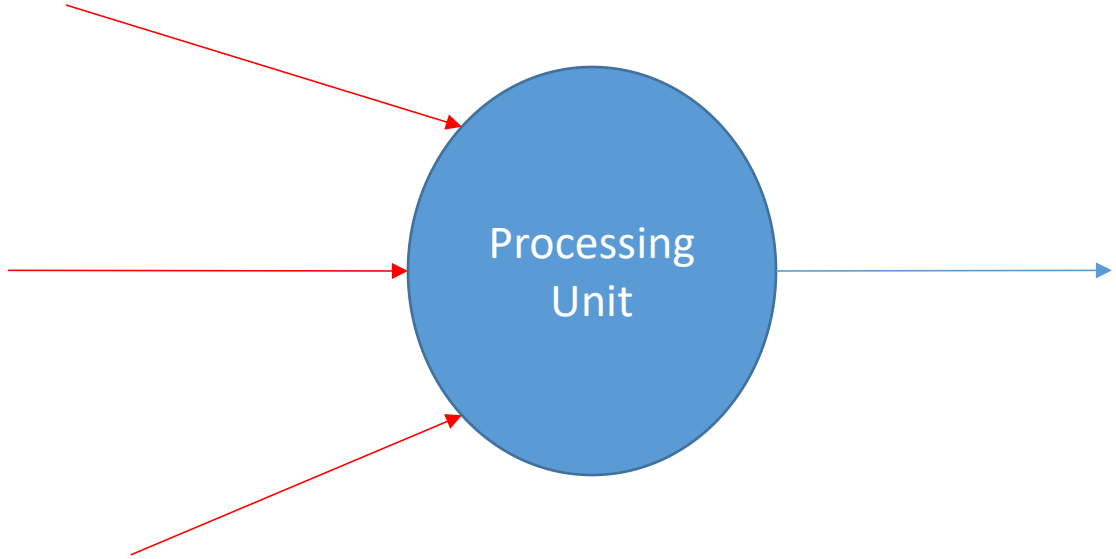


What is a neuron?

- From biology, a neuron processes electrochemical **input signals** which come from others neurons.
- The result of such processing is another electrochemical signal which is called **output**.
- This signal is transmitted to other neuron through the **axon**.

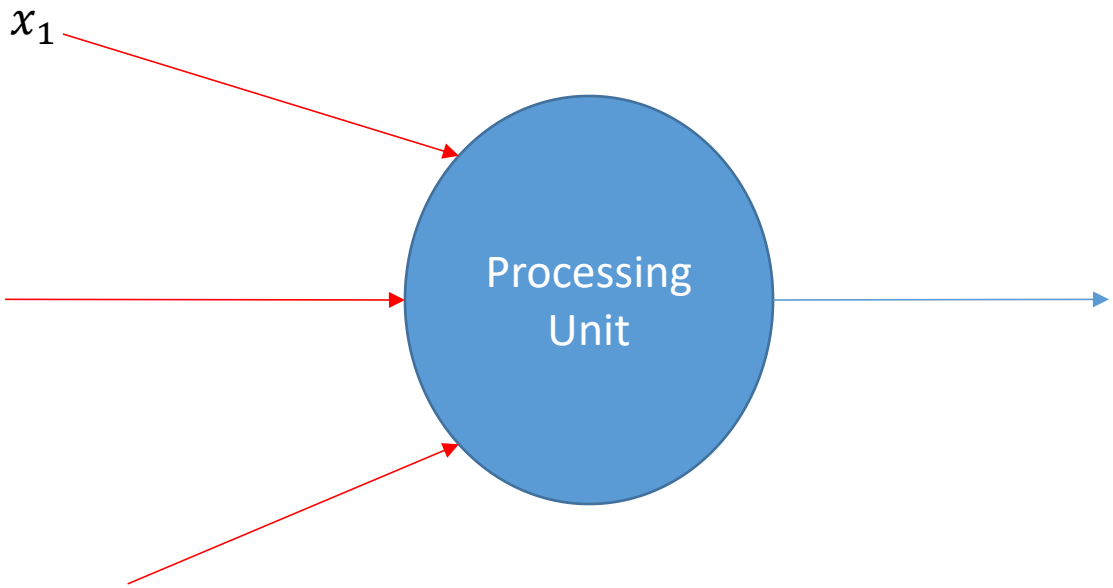
Artificial Neuron

- Inspired by biological neurons



Artificial Neuron

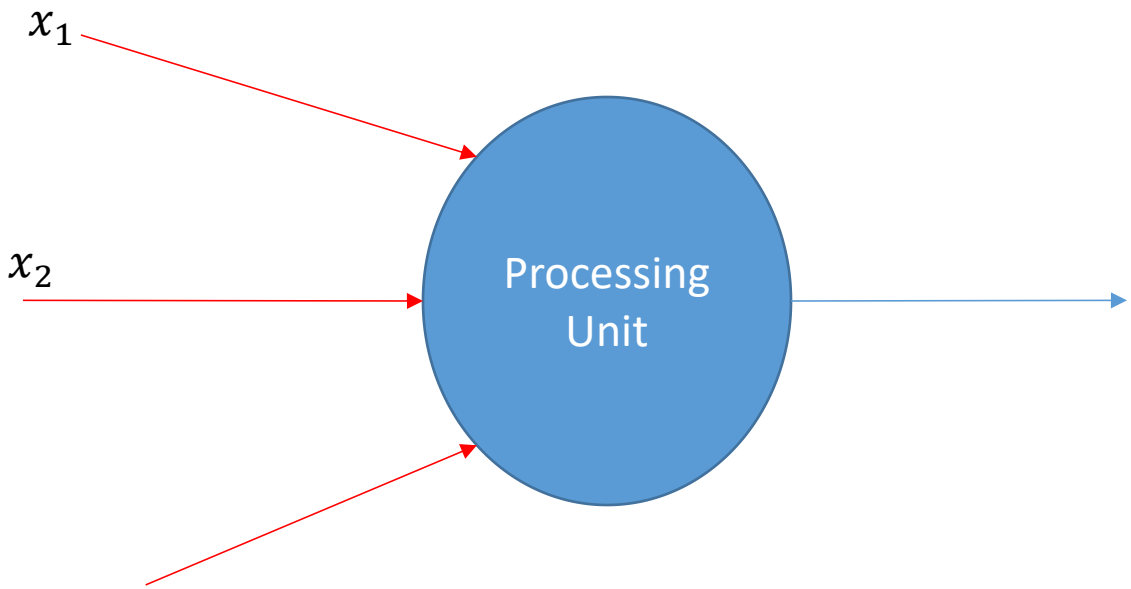
- Inspired by biological neurons



The input of a neuron is a vector $\vec{x}$

Artificial Neuron

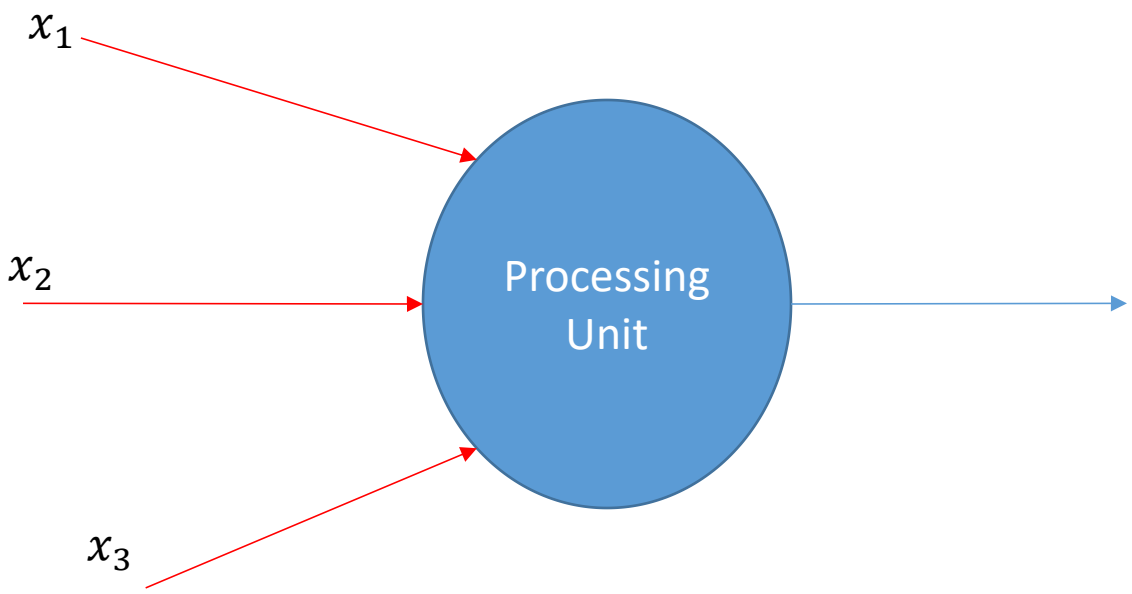
- Inspired by biological neurons



The input of a neuron is a vector $\vec{x}$

Artificial Neuron

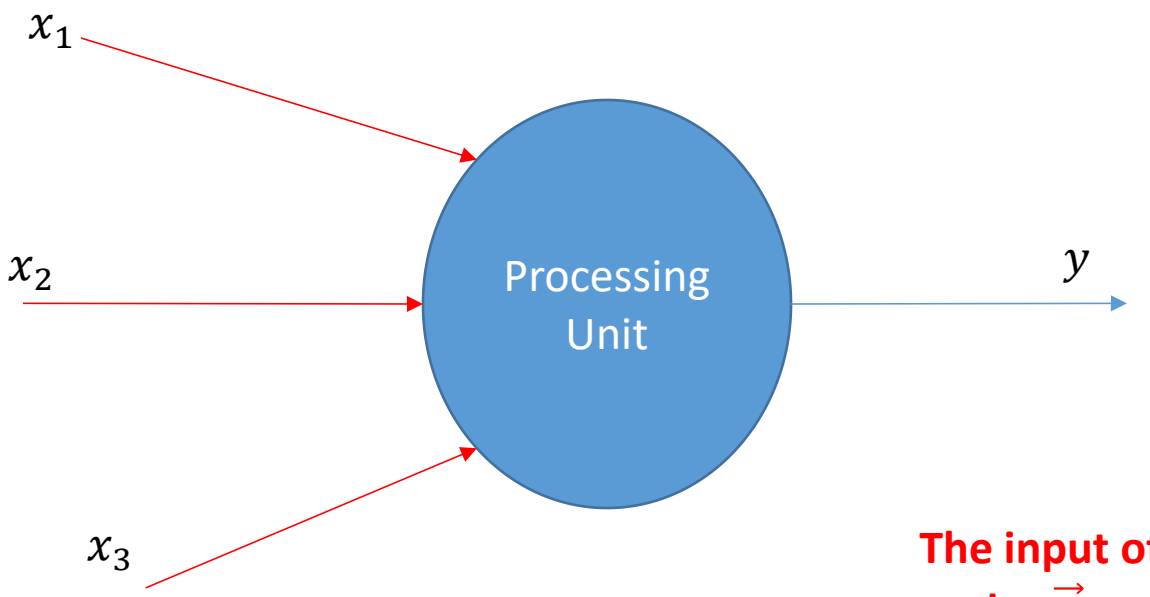
- Inspired by biological neurons.



The input of a neuron is a vector $\vec{x}$

Artificial Neuron

- Inspired by biological neurons.

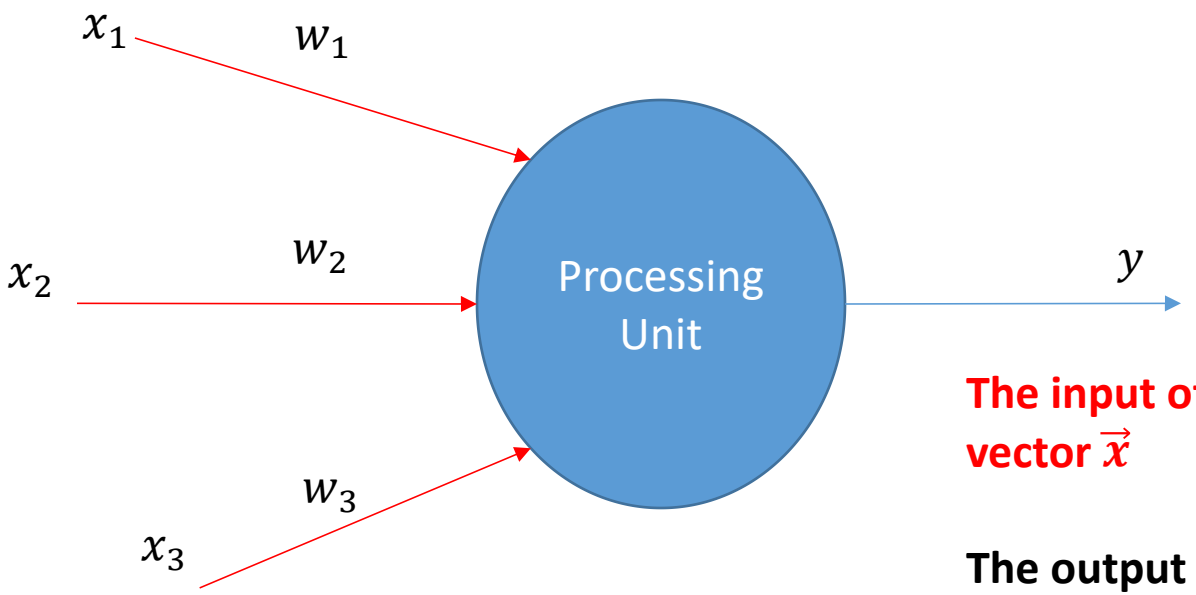


The input of a neuron is a vector $\vec{x}$

The output of the neuron is a scalar value

Artificial Neuron

- Inspired by biological neurons.



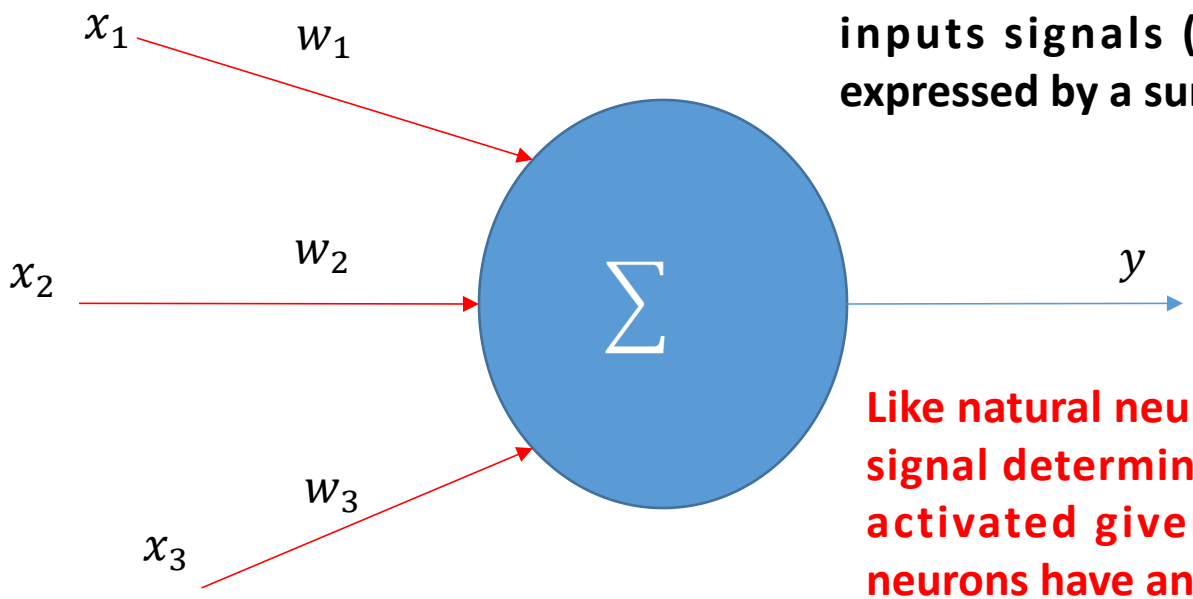
The input of a neuron is a vector $\vec{x}$

The output of the neuron is a scalar value

The importance of a component of $\vec{x}$ is weighted with a value w_i

Artificial Neuron

- Inspired by biological neurons.



The input of a neuron is a vector $\vec{x}$

The output of the neuron is a scalar value

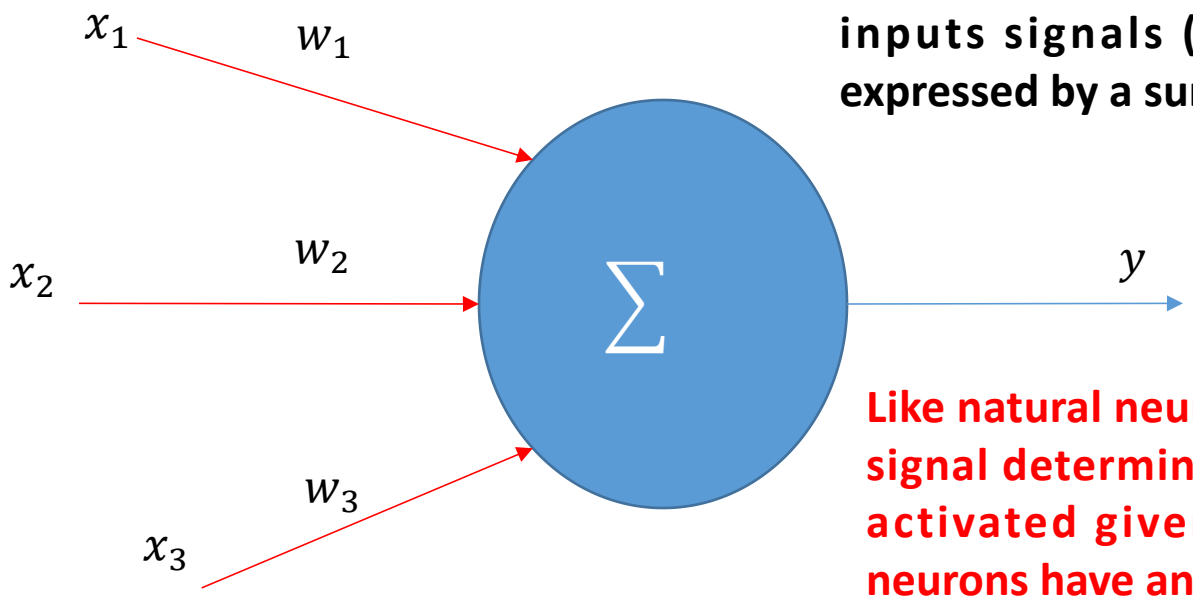
The importance of a component of $\vec{x}$ is weighted with a value w_i

The processing unit summarized the inputs signals (mathematically it is expressed by a sum).

Like natural neurons, the value of output signal determine if the neuron must be activated given its inputs. Thus, the neurons have an activation threshold.

Artificial Neuron

- Inspired by biological neurons.



The input of a neuron is a vector $\vec{x}$

The output of the neuron is a scalar value

The importance of a component of $\vec{x}$ is weighted with a value w_i

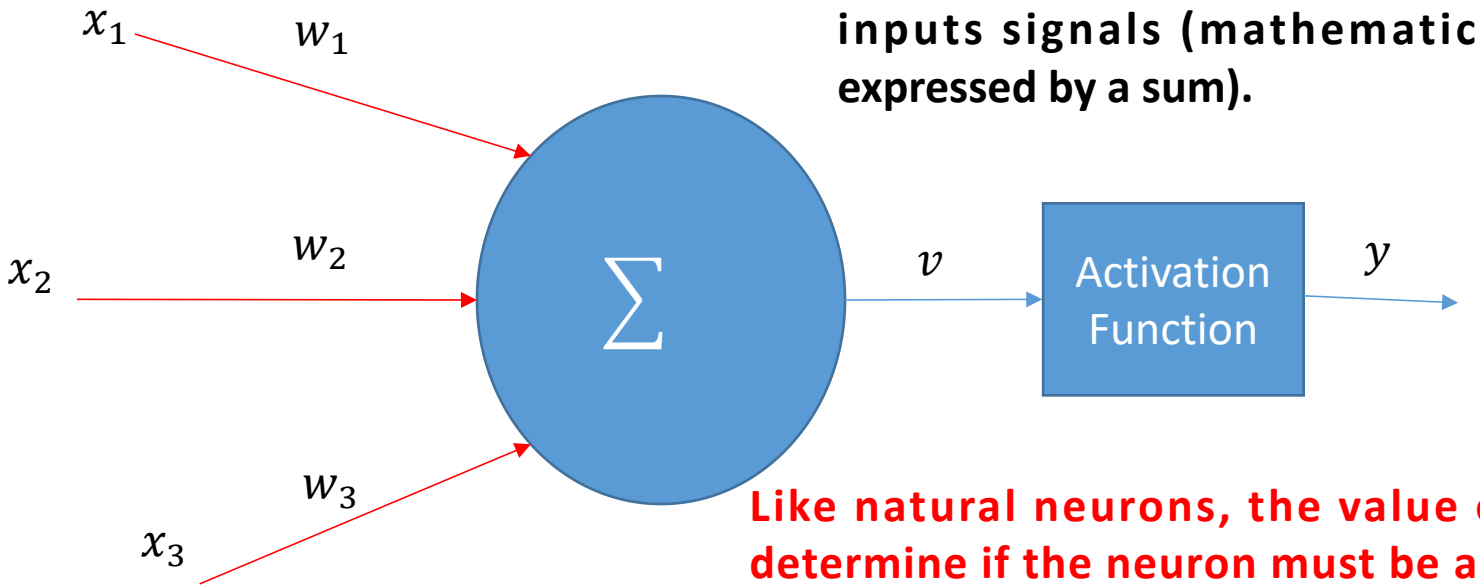
The processing unit summarized the inputs signals (mathematically it is expressed by a sum).

Like natural neurons, the value of output signal determine if the neuron must be activated given its inputs. Thus, the neurons have an activation threshold.

We can emulate such threshold with mathematical functions known as activation functions

Artificial Neuron

- Inspired by biological neurons.



The input of a neuron is a vector $\vec{x}$

The output of the neuron is a scalar value

The importance of a component of $\vec{x}$ is weighted with a value w_i

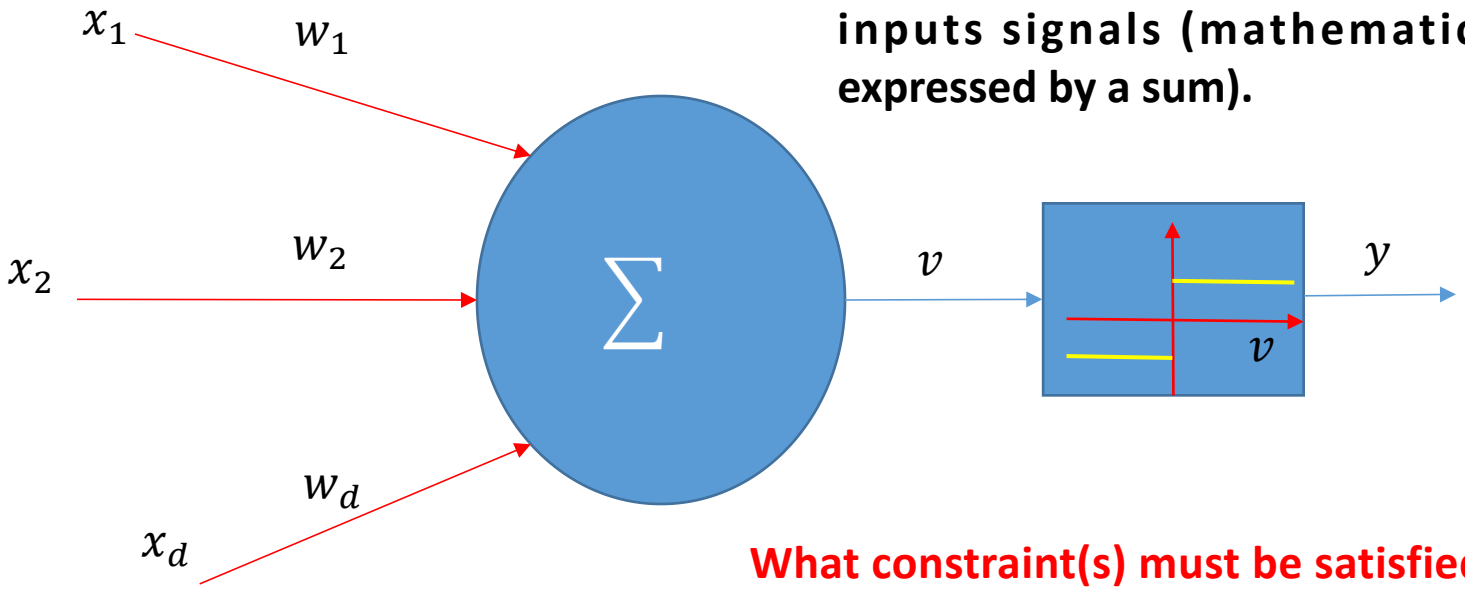
The processing unit summarized the inputs signals (mathematically it is expressed by a sum).

Like natural neurons, the value of output signal determine if the neuron must be activated given its inputs. Thus, the neurons have an activation threshold.

We can emulate such threshold with mathematical functions known as activation functions

Artificial Neuron

- Inspired by biological neurons.



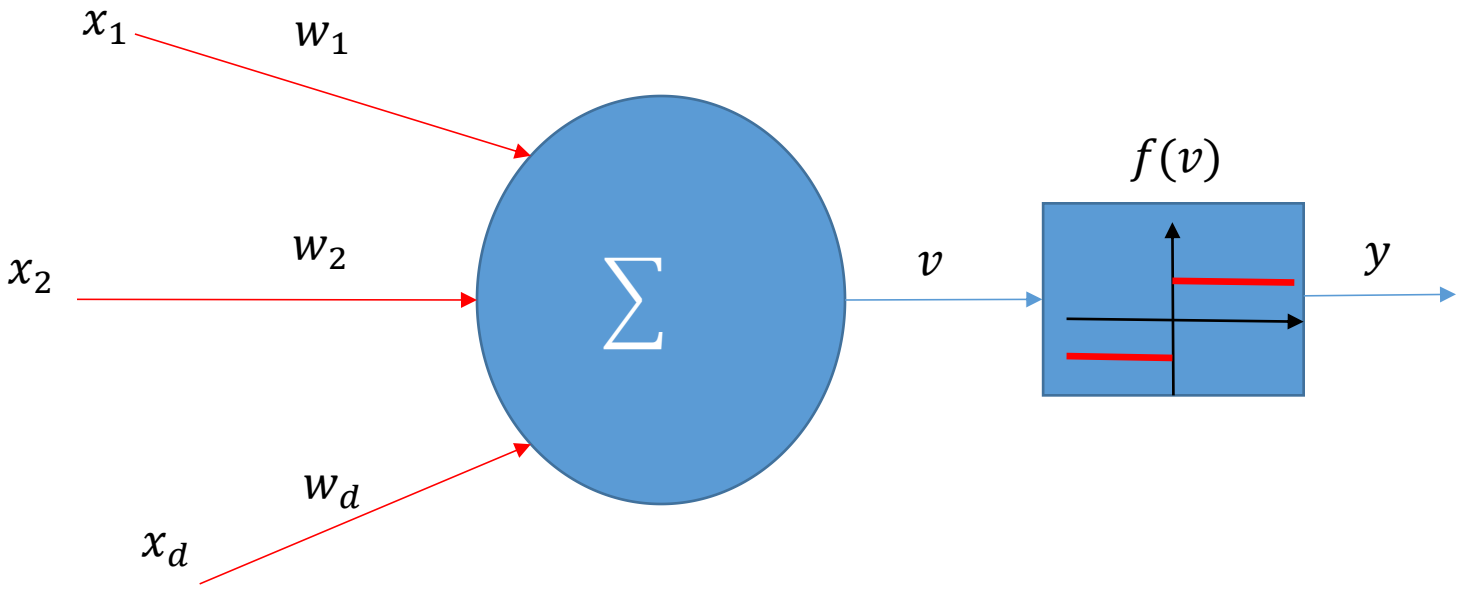
The input of a neuron is a vector $\vec{x}$

The output of the neuron is a scalar value

The importance of a component of $\vec{x}$ is weighted with a value w_i

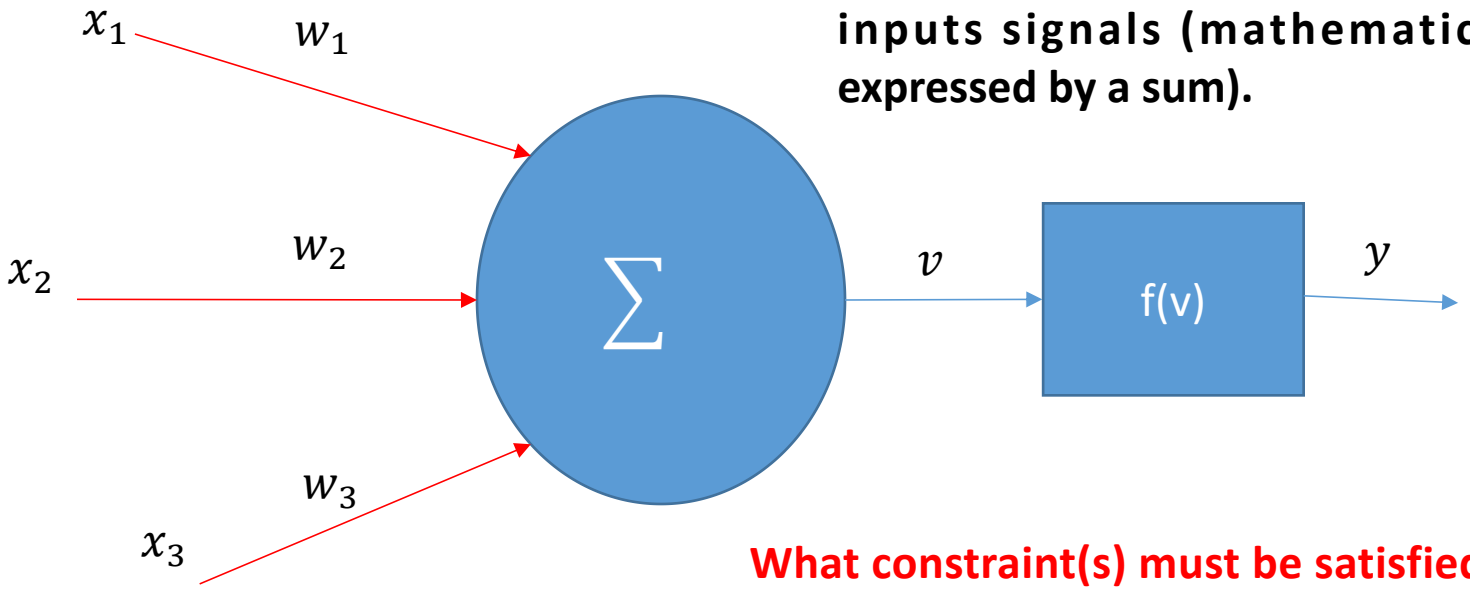
The processing unit summarized the inputs signals (mathematically it is expressed by a sum).

What constraint(s) must be satisfied by the activation function?


x_1
w_1
x_2
w_2
w_d
x_d
$\sum$
v
$f(v)$
y

Artificial Neuron

- Inspired by biological neurons.



The input of a neuron is a vector $\vec{x}$

The output of the neuron is a scalar value

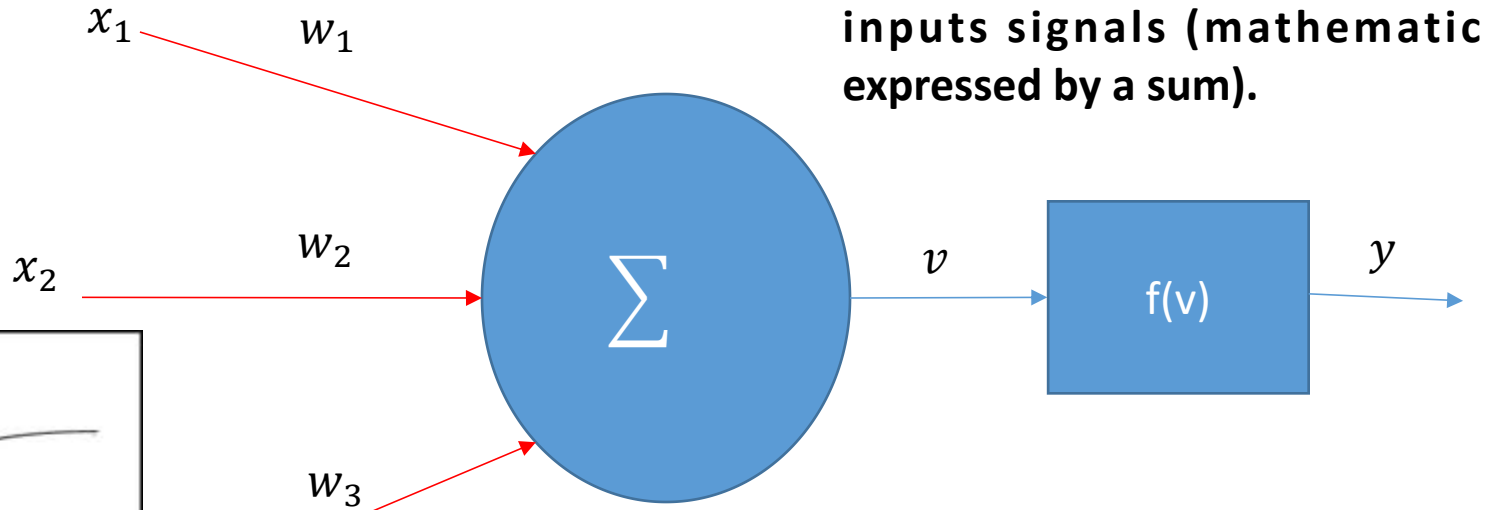
The importance of a component of $\vec{x}$ is weighted with a value w_i

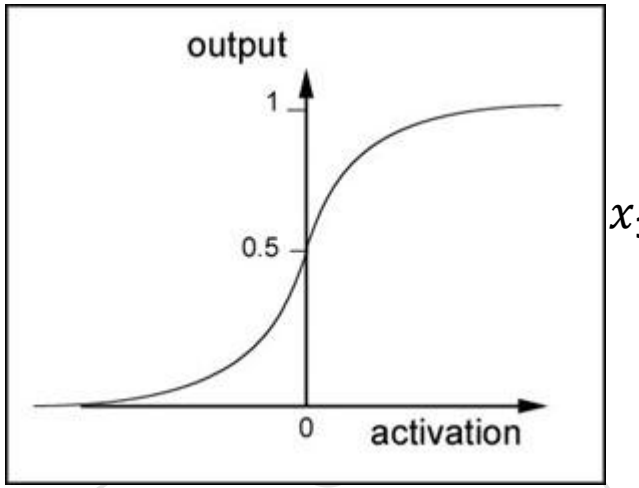
The processing unit summarized the inputs signals (mathematically it is expressed by a sum).

What constraint(s) must be satisfied by the activation function?

Artificial Neuron

- Inspired by biological neurons.





The input of a neuron is a vector $\vec{x}$

The output of the neuron is a scalar value

The importance of a component of $\vec{x}$ is weighted with a value w_i

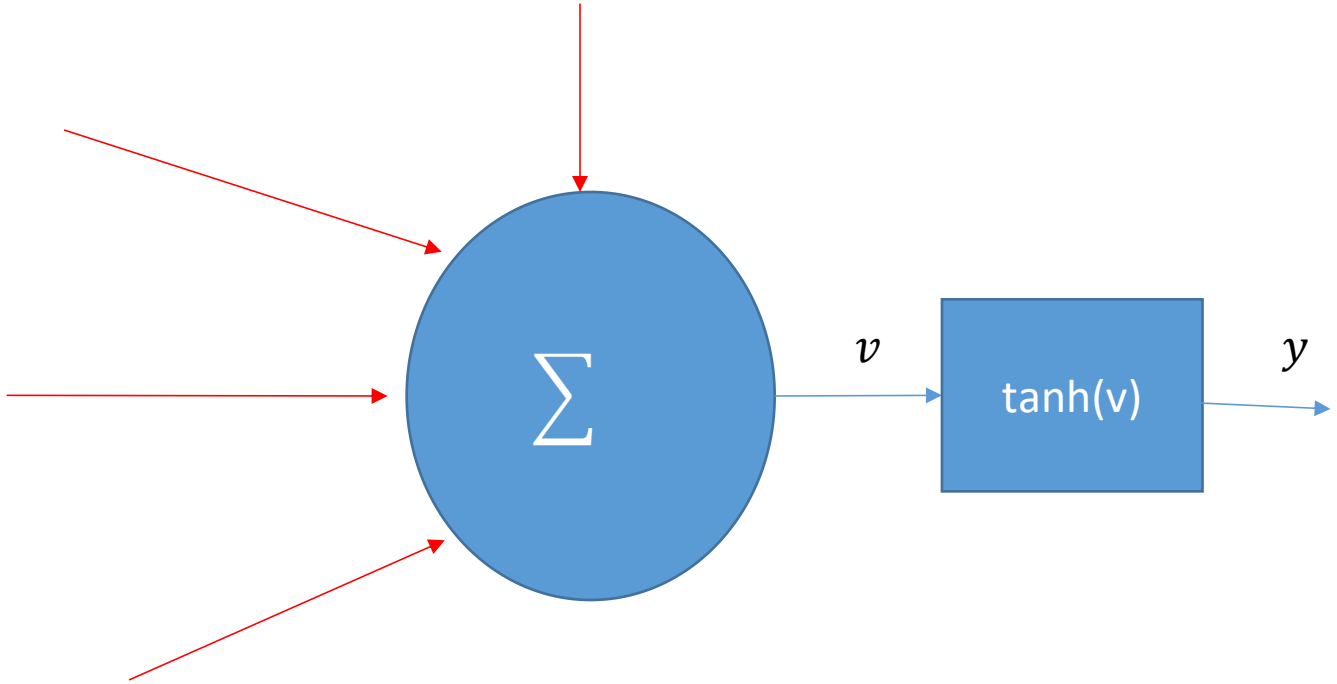
The processing unit summarized the inputs signals (mathematically it is expressed by a sum).

What constraint(s) must be satisfied by the activation function?

Why an artificial network is able to classify?

Artificial Neuron

- Training a neuron (perceptron).



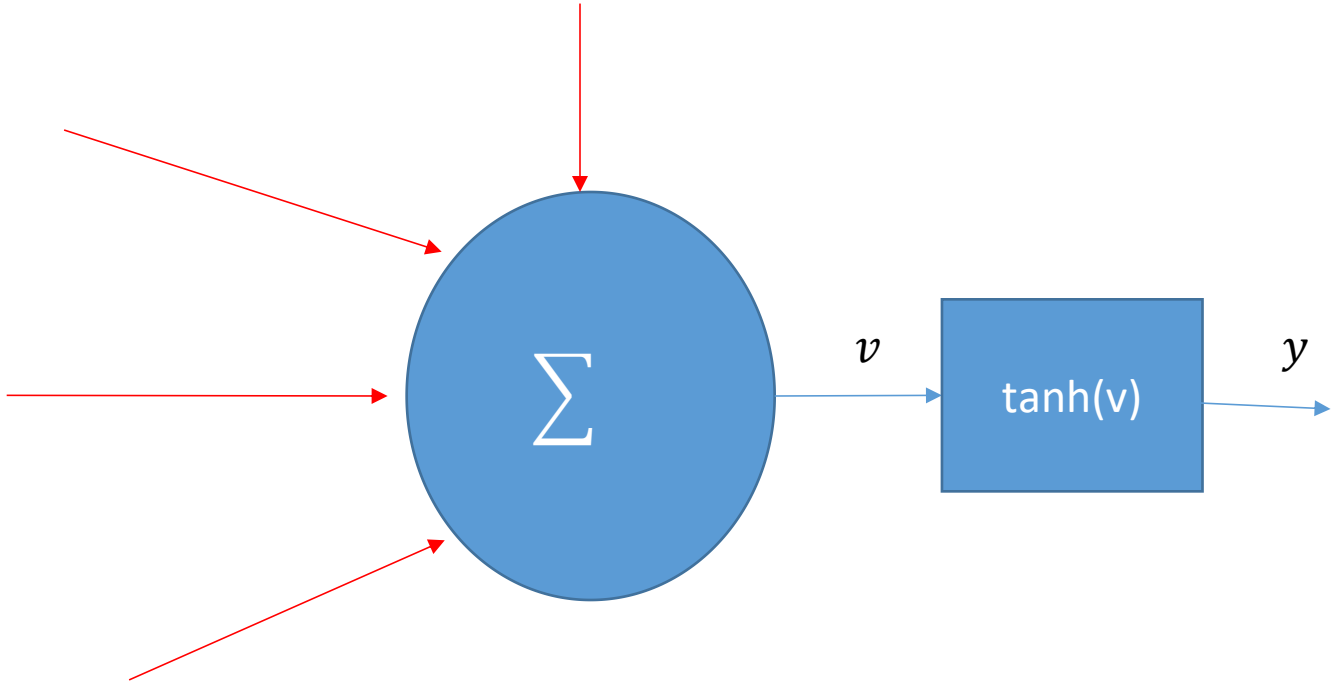
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 1: A random vector W is generated



X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

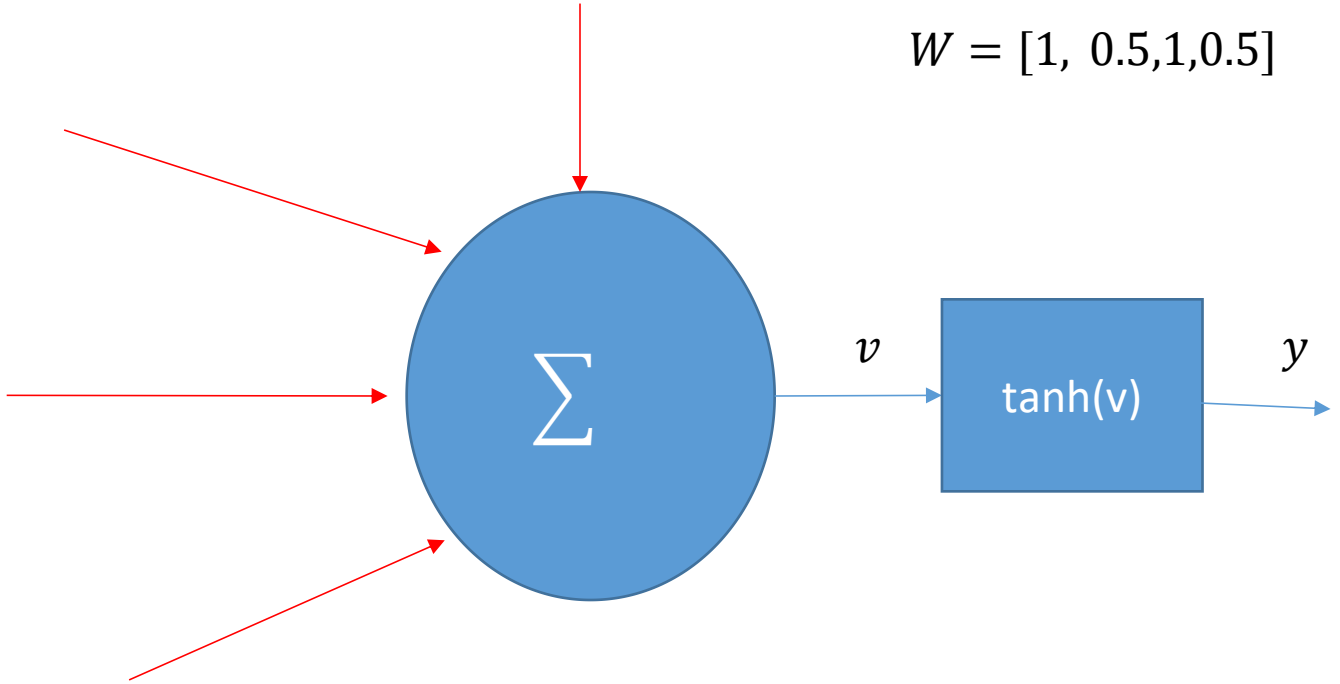
Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 1: A random vector W is generated

$$W = [1,\ 0.5,1,0.5]$$



X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

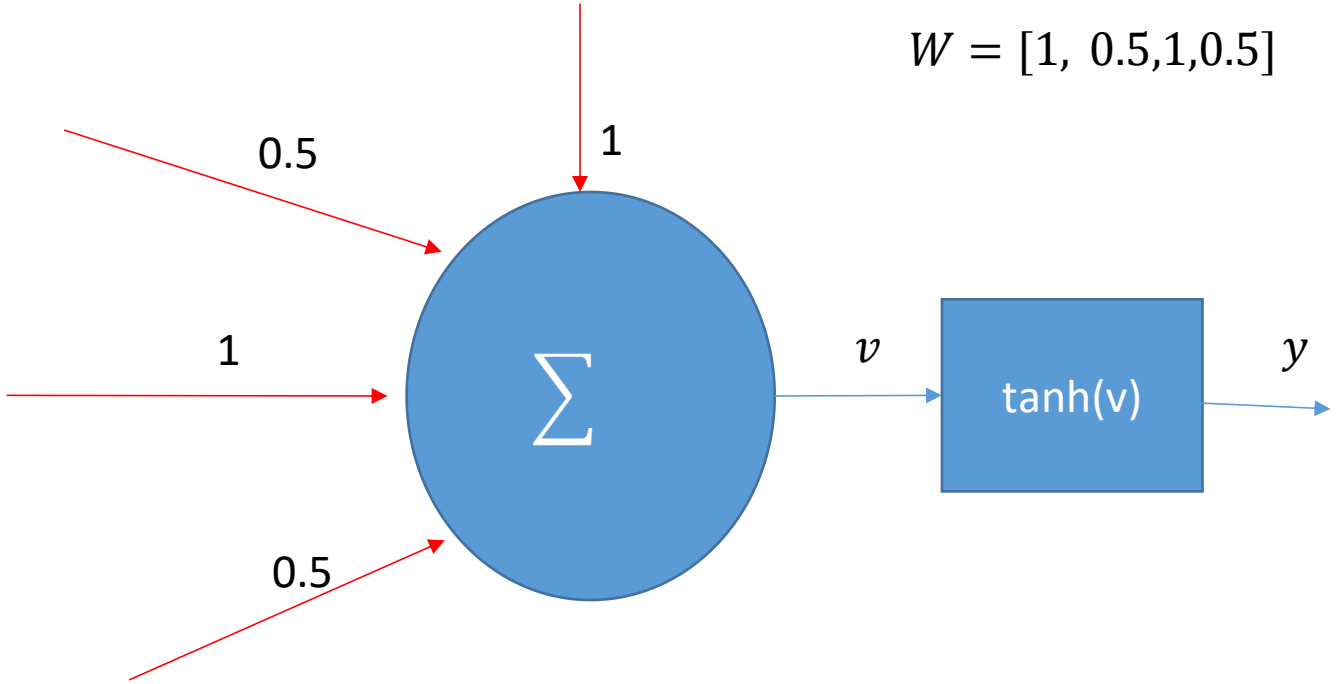
Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 1: A random vector W is generated

$$W = [1,\ 0.5,1,0.5]$$



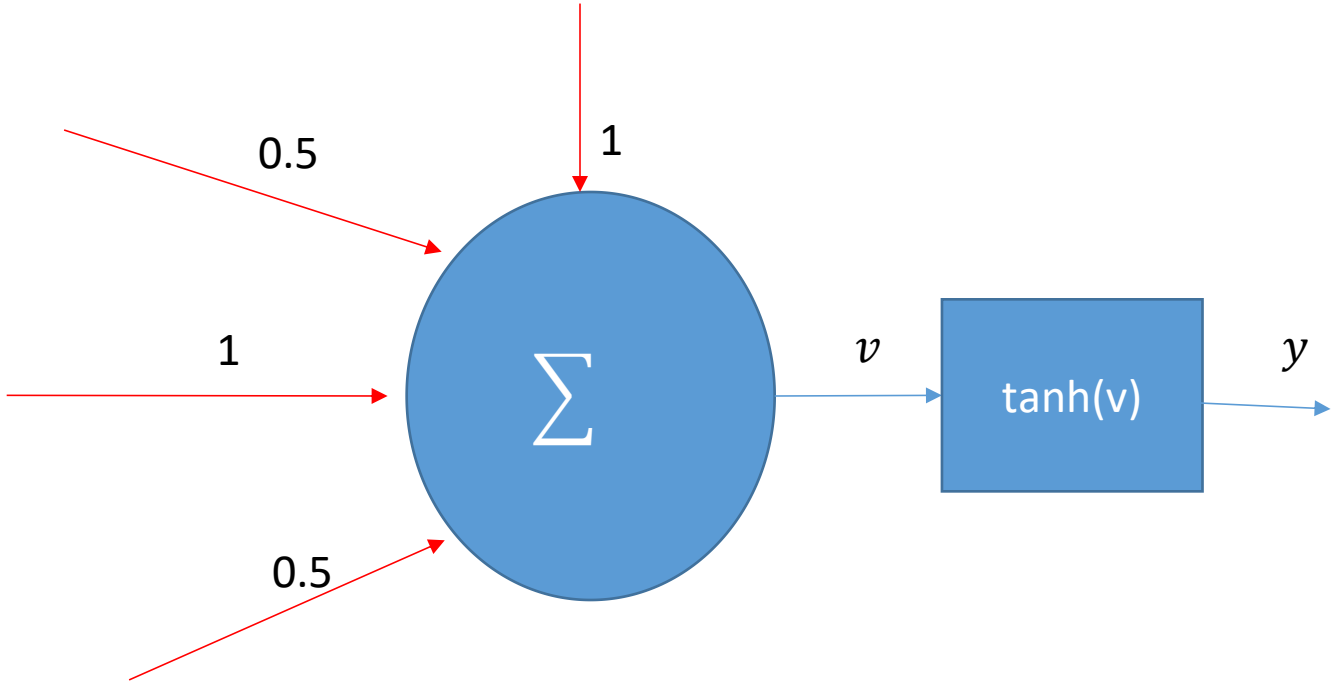
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 2: The first pattern is presented to the perceptron



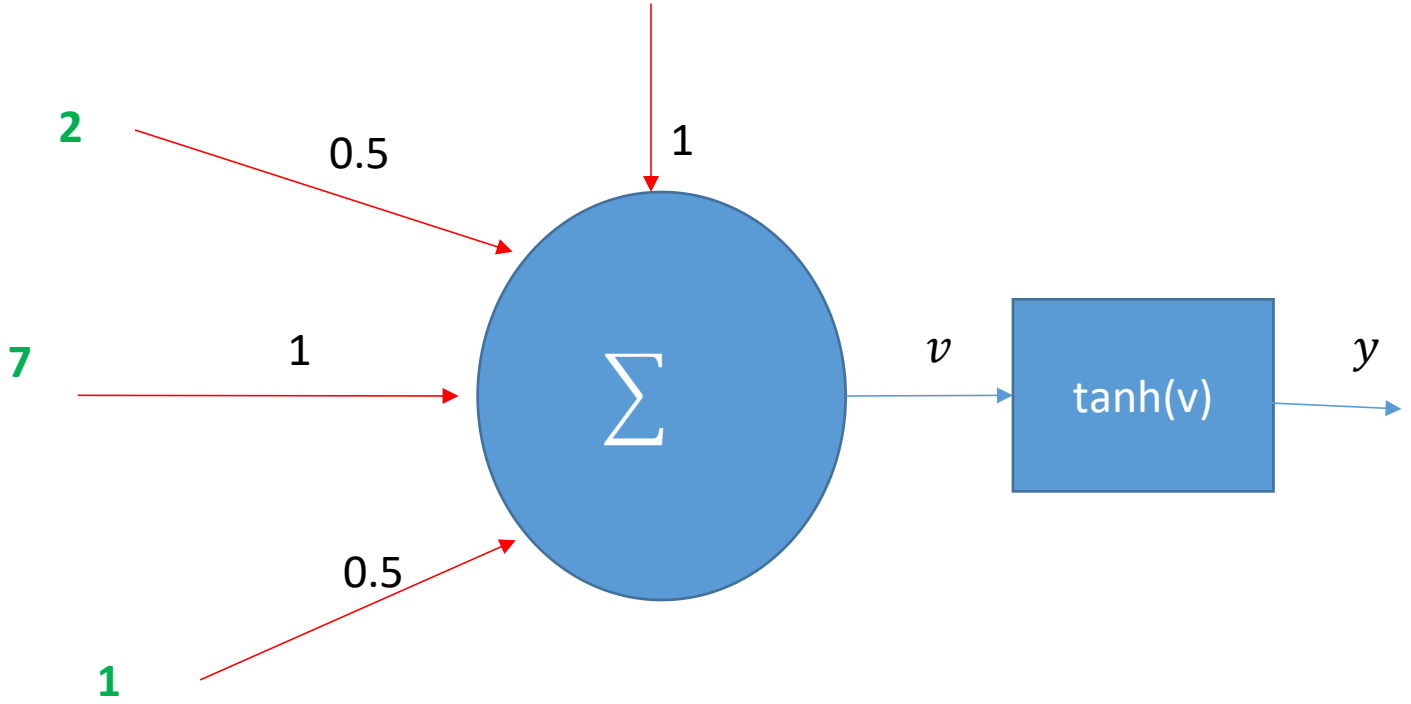
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 2: The first pattern is presented to the perceptron



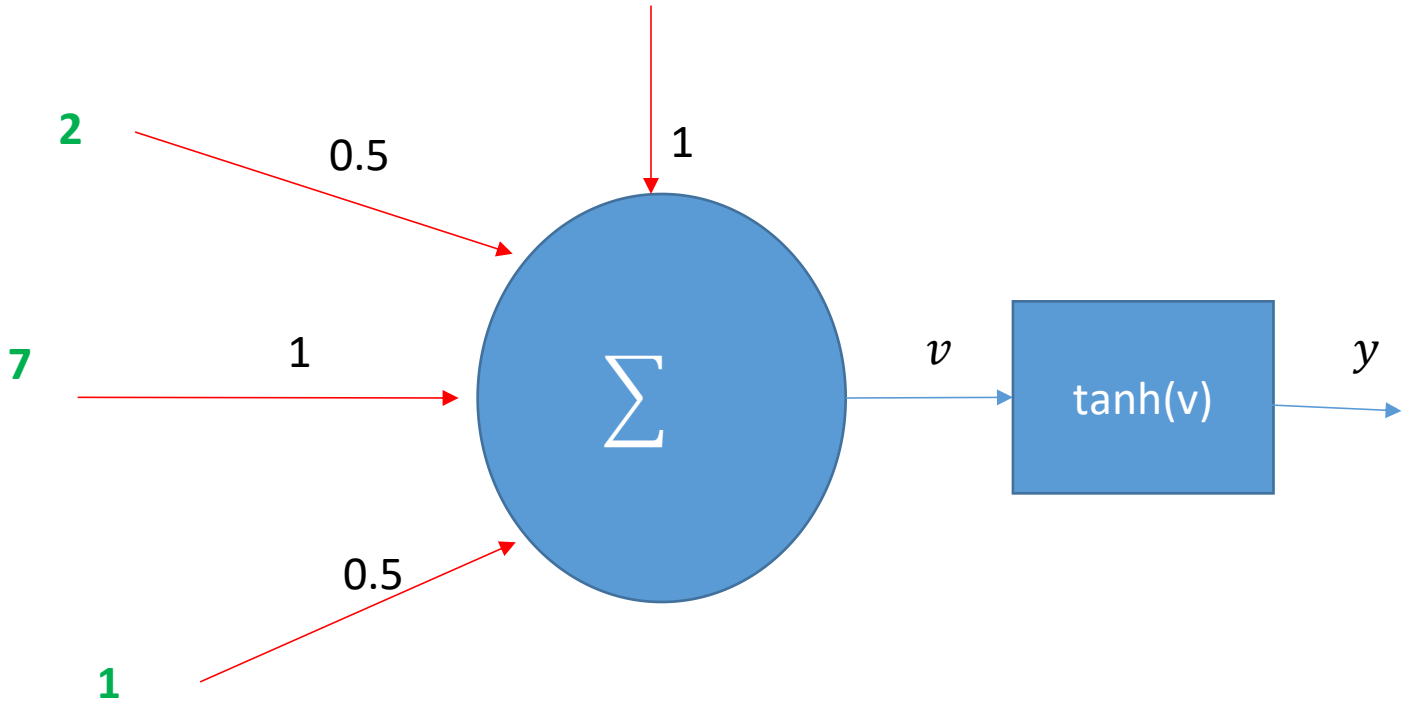
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 3: The output of the perceptron is determined.



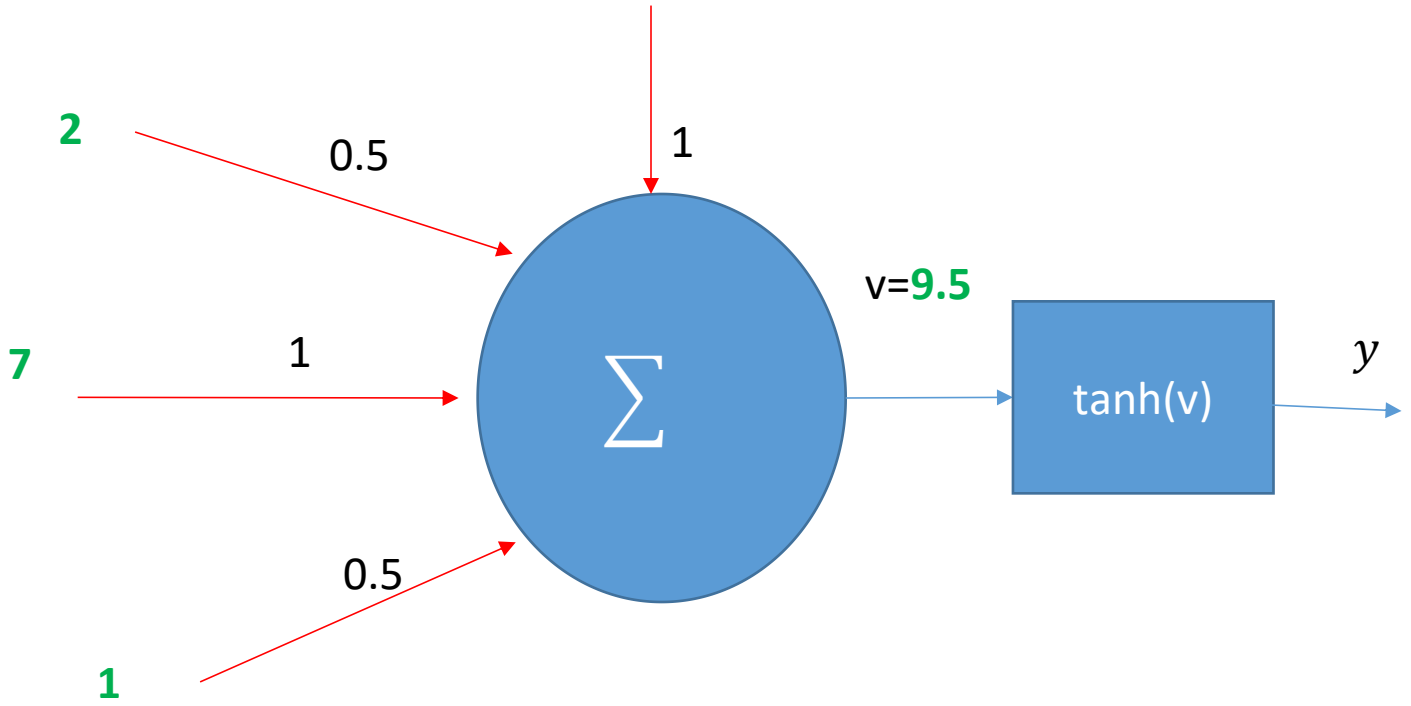
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 3: The output of the perceptron is determined.



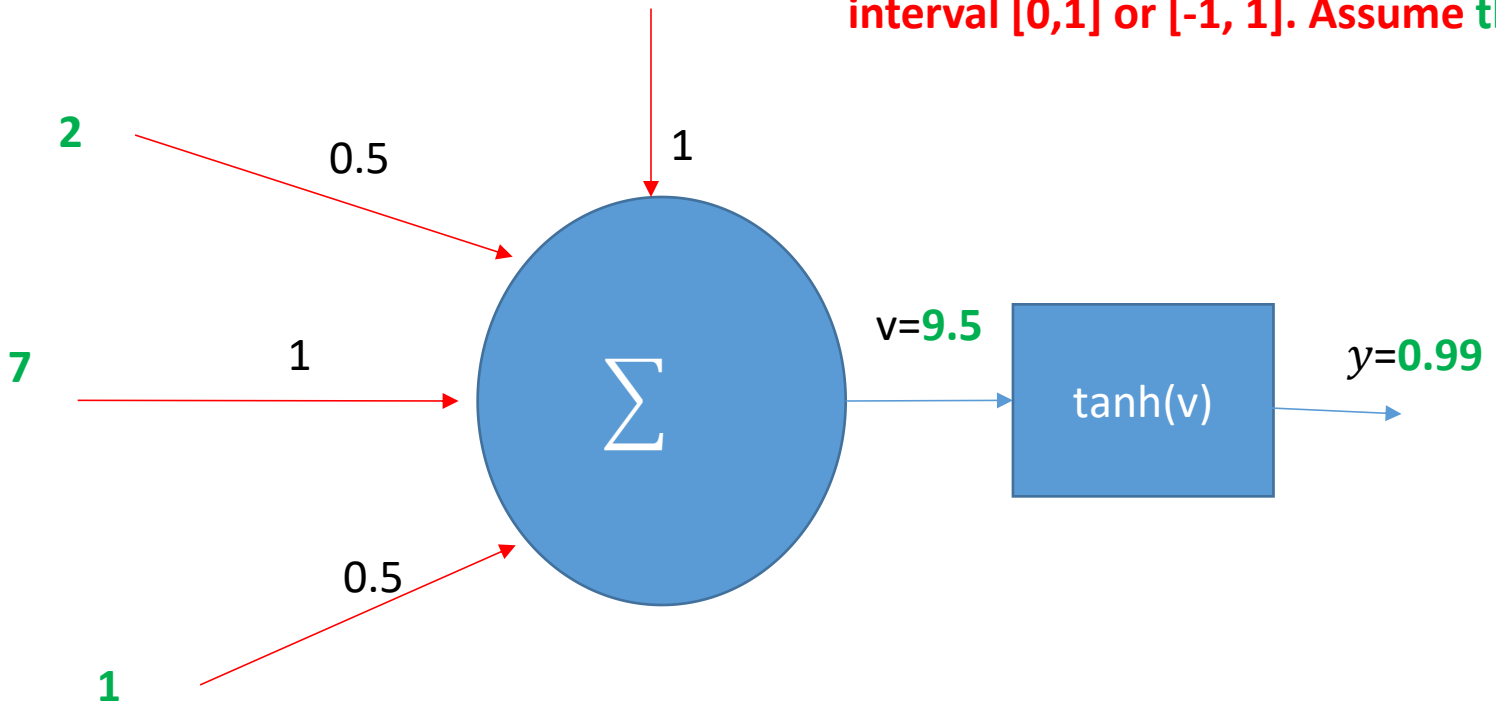
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 4: The value of y is calculated based on activation function. Recall that such function allows to normalize the output as an closed interval [0,1] or [-1, 1]. Assume the last interval.



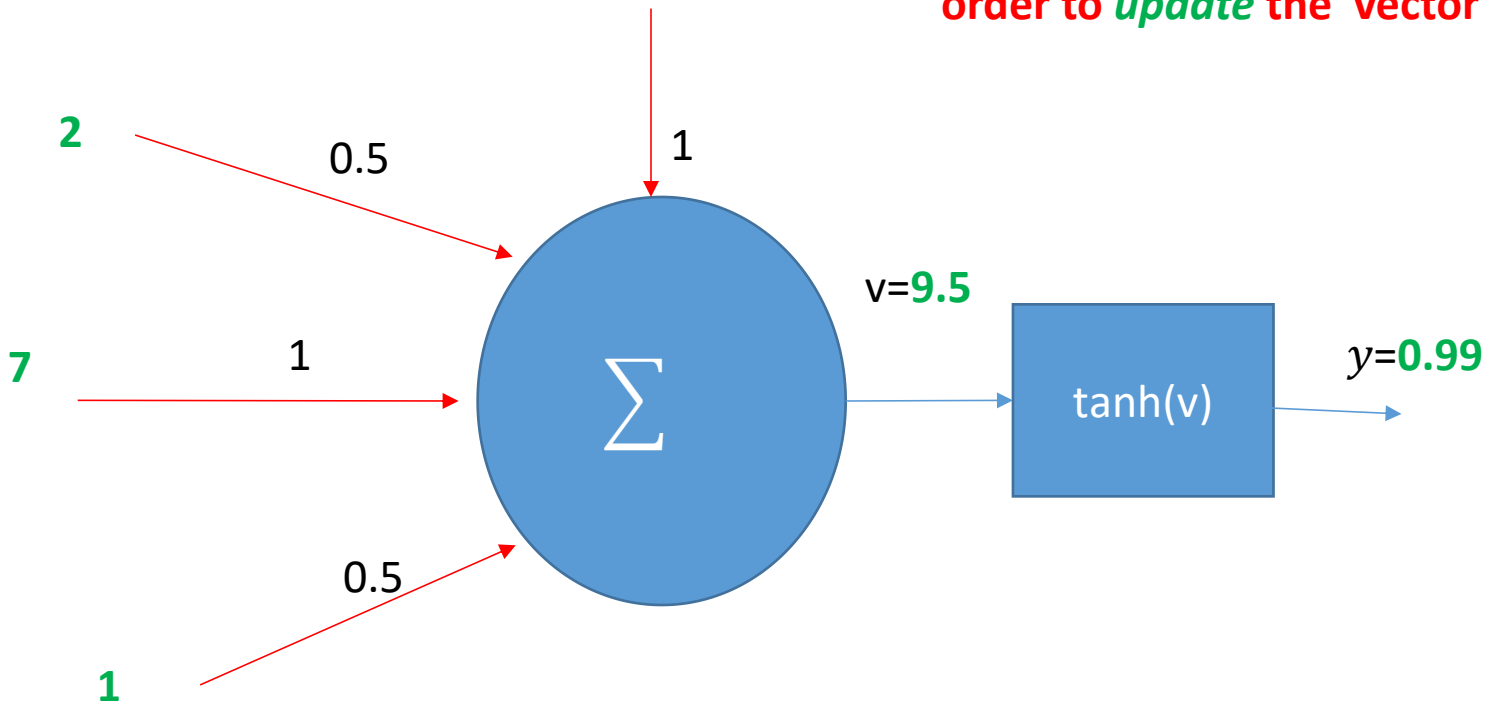
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 5: The value of y is evaluated in order to *update* the vector W.



X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

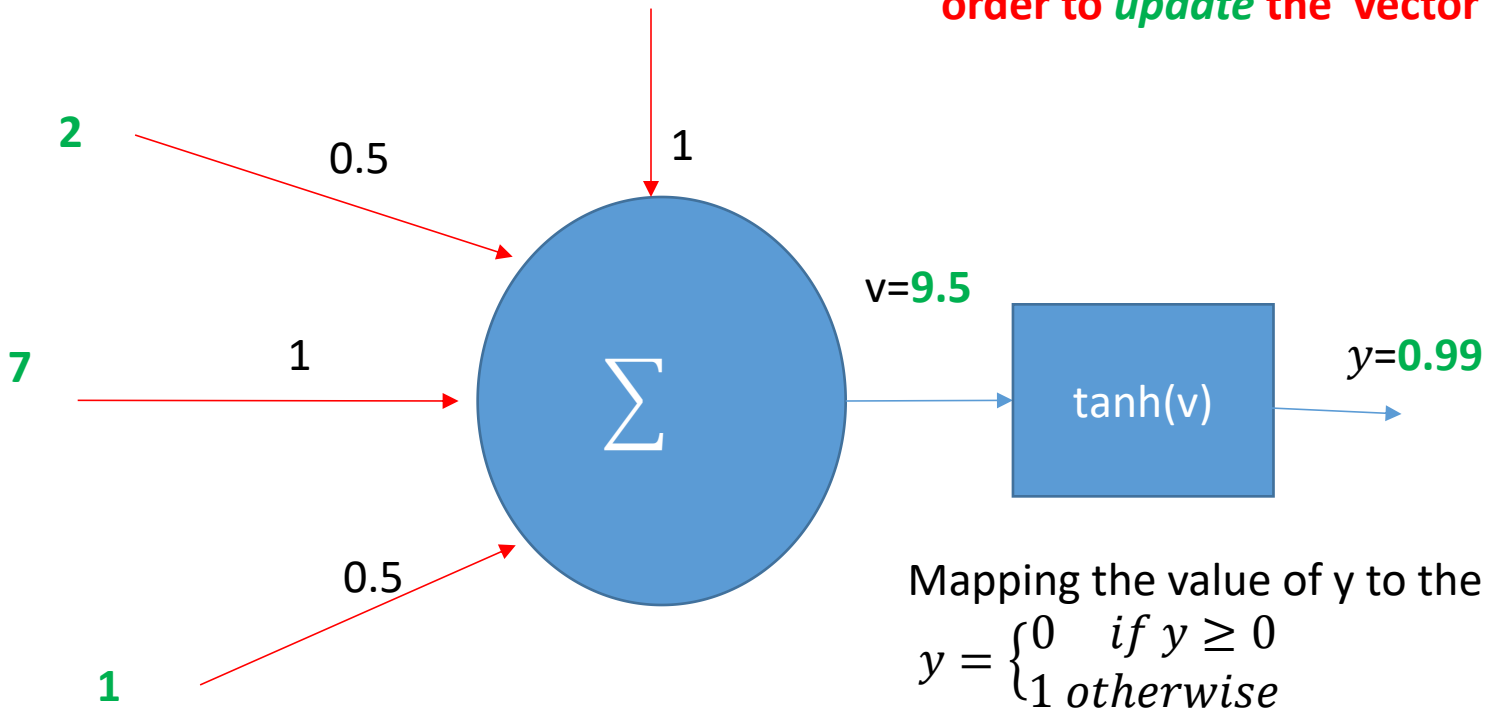
Dataset

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

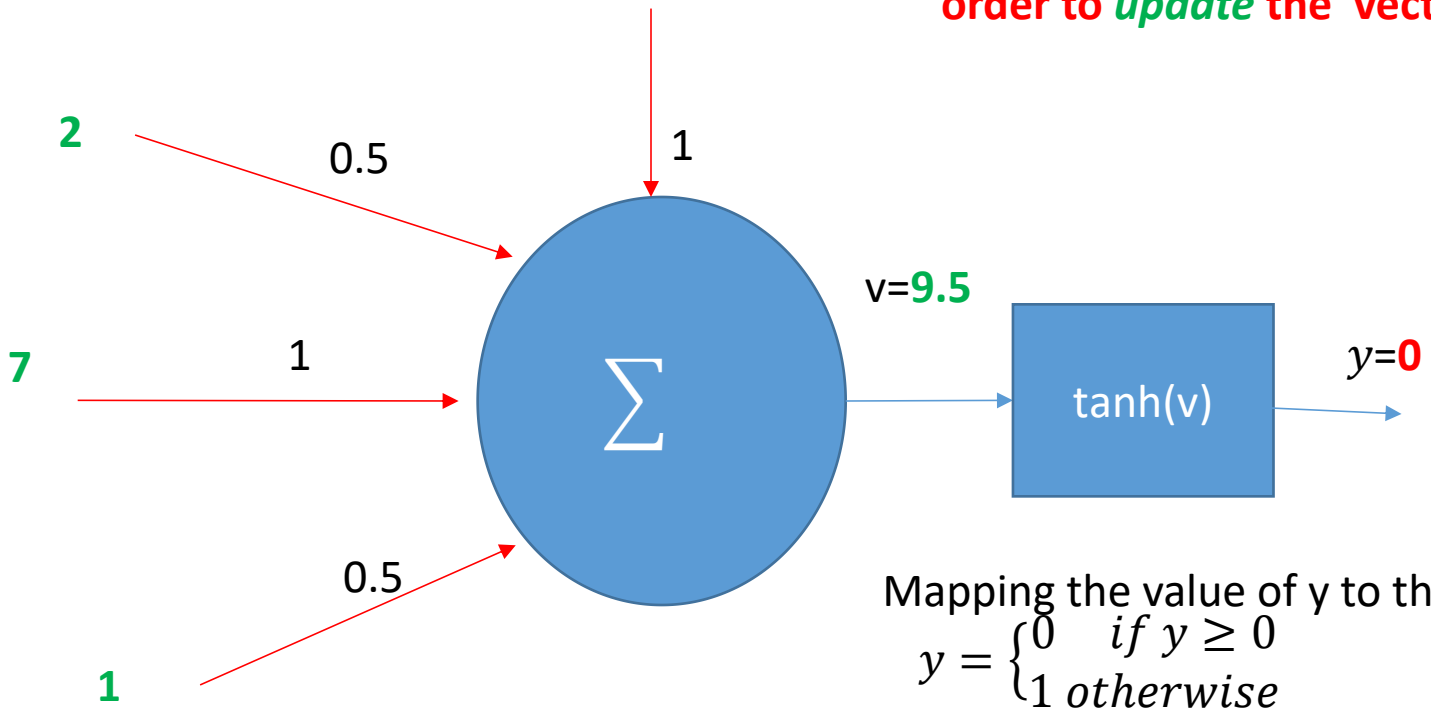
Mapping the value of y to the class label.

$$y = \begin{cases} 0 & if\ y \geq 0 \\ 1 & otherwise \end{cases}$$

Artificial Neuron

- Training a neuron (perceptron).

Step 5: The value of y is evaluated in order to *update* the vector W.



Mapping the value of y to the class label.

$$y = \begin{cases} 0 & if\ y \geq 0 \\ 1 & otherwise \end{cases}$$

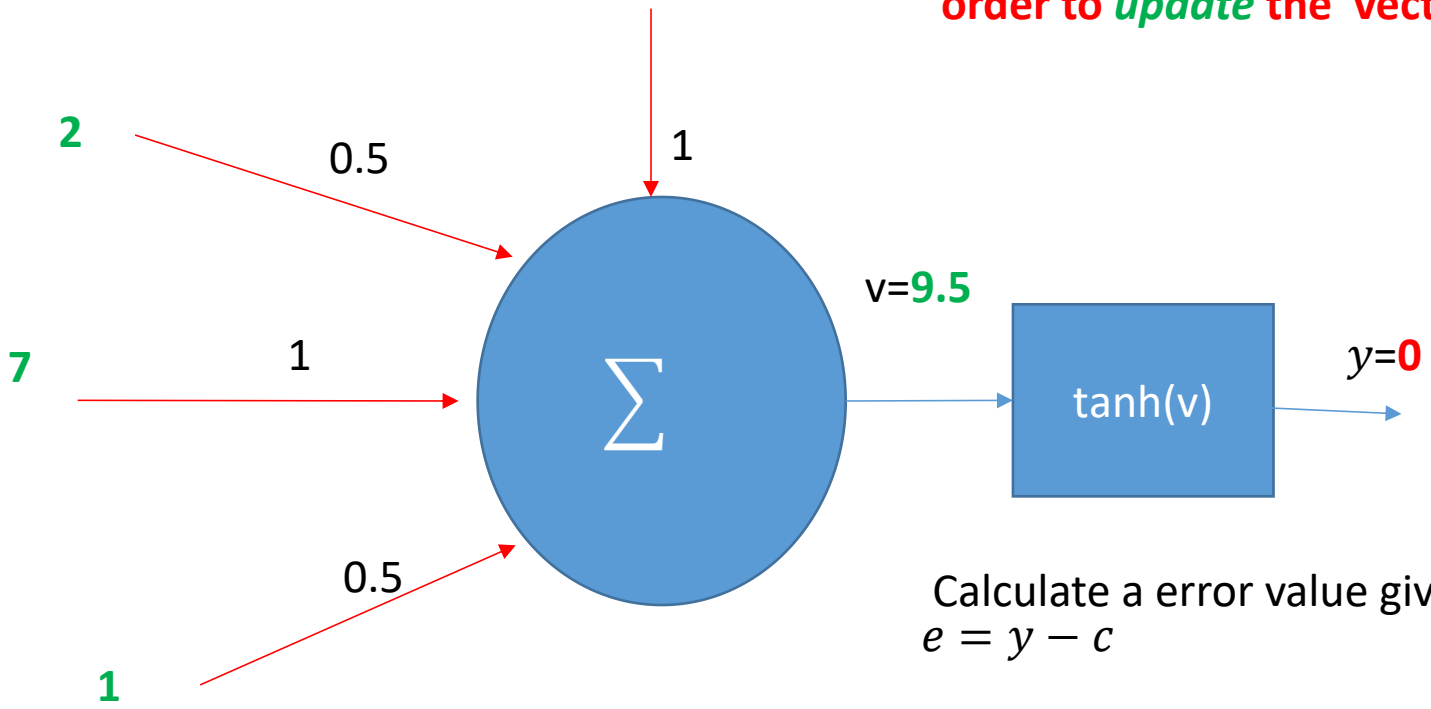
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 5: The value of y is evaluated in order to *update* the vector W.



X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

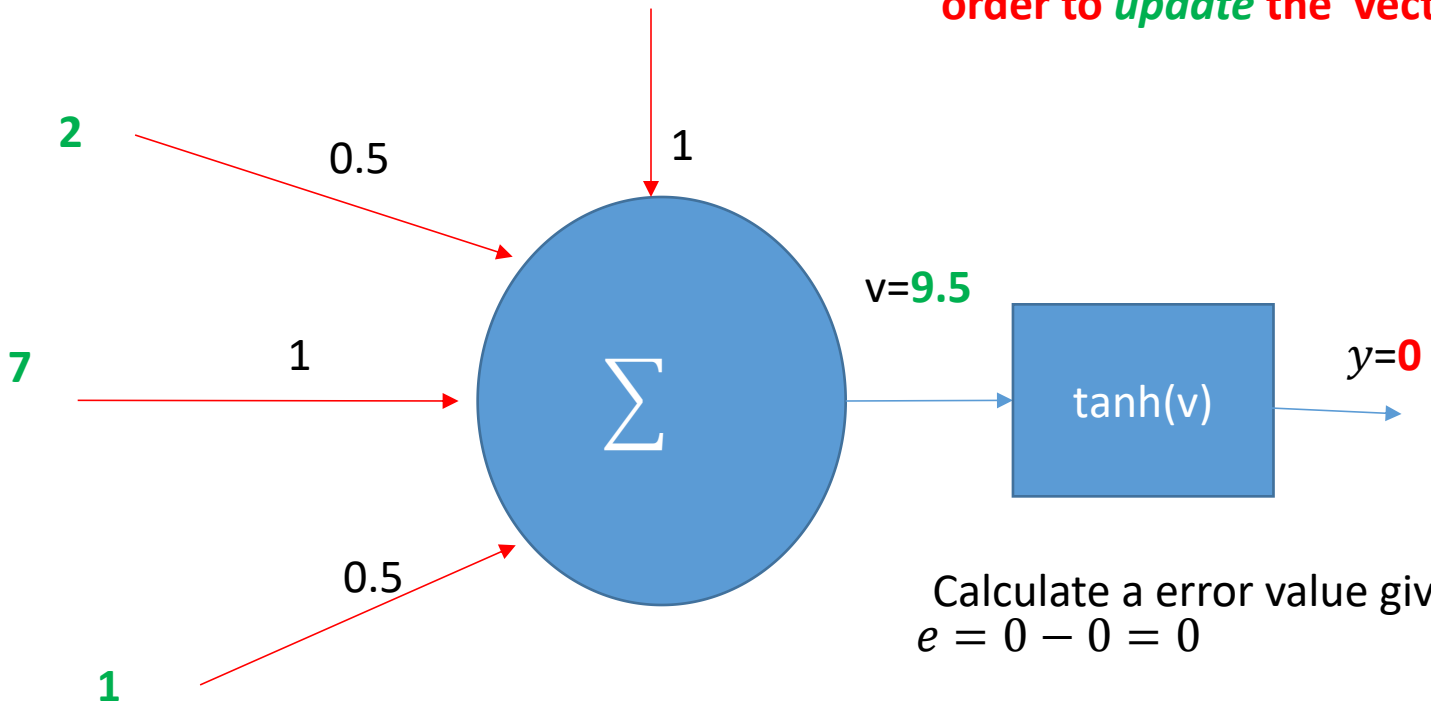
Calculate a error value given the expected class c

$e = y - c$

Artificial Neuron

- Training a neuron (perceptron).

Step 5: The value of y is evaluated in order to *update* the vector W.



X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Calculate a error value given the expected class c

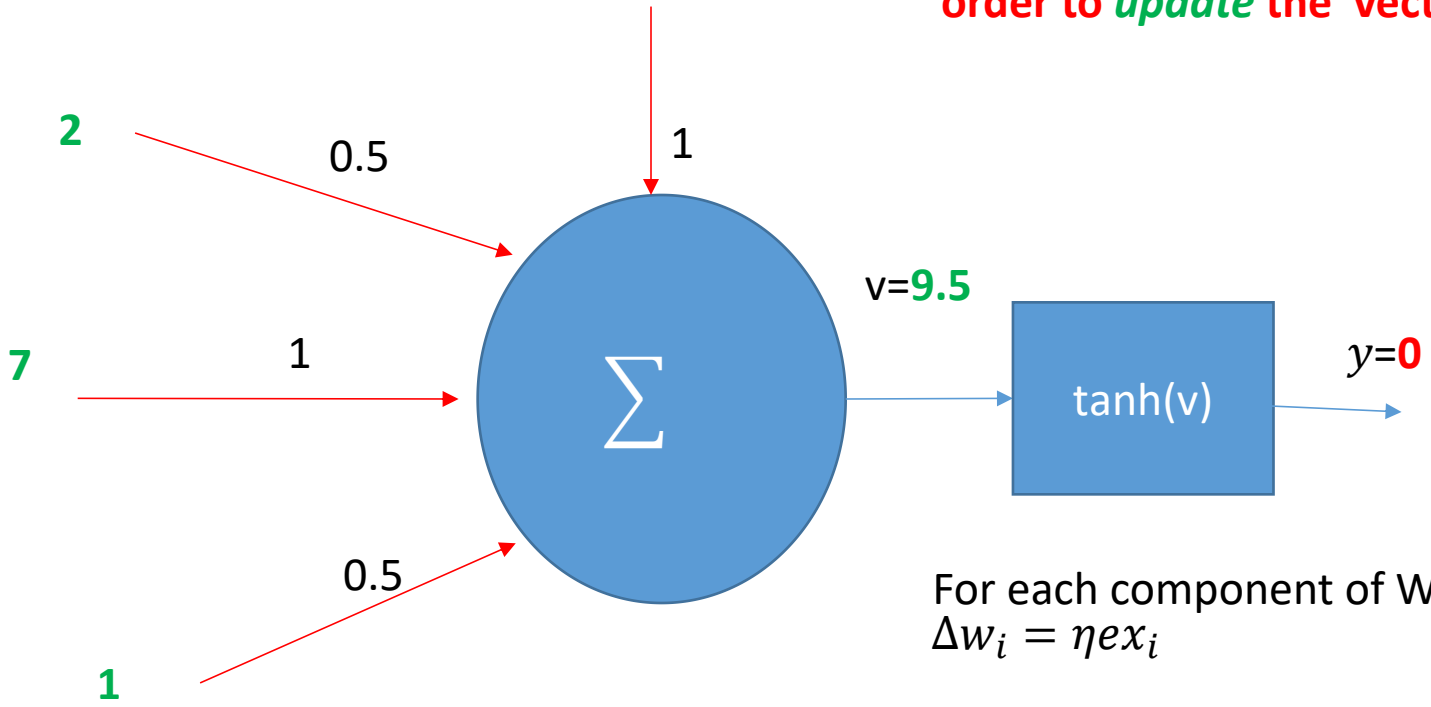
$e = 0 - 0 = 0$

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

For each component of W, calculate a variation Δw_i

$\Delta w_i = \eta e x_i$

Where η is a perturbation value, usually it is a small positive value. Assume η=0.01

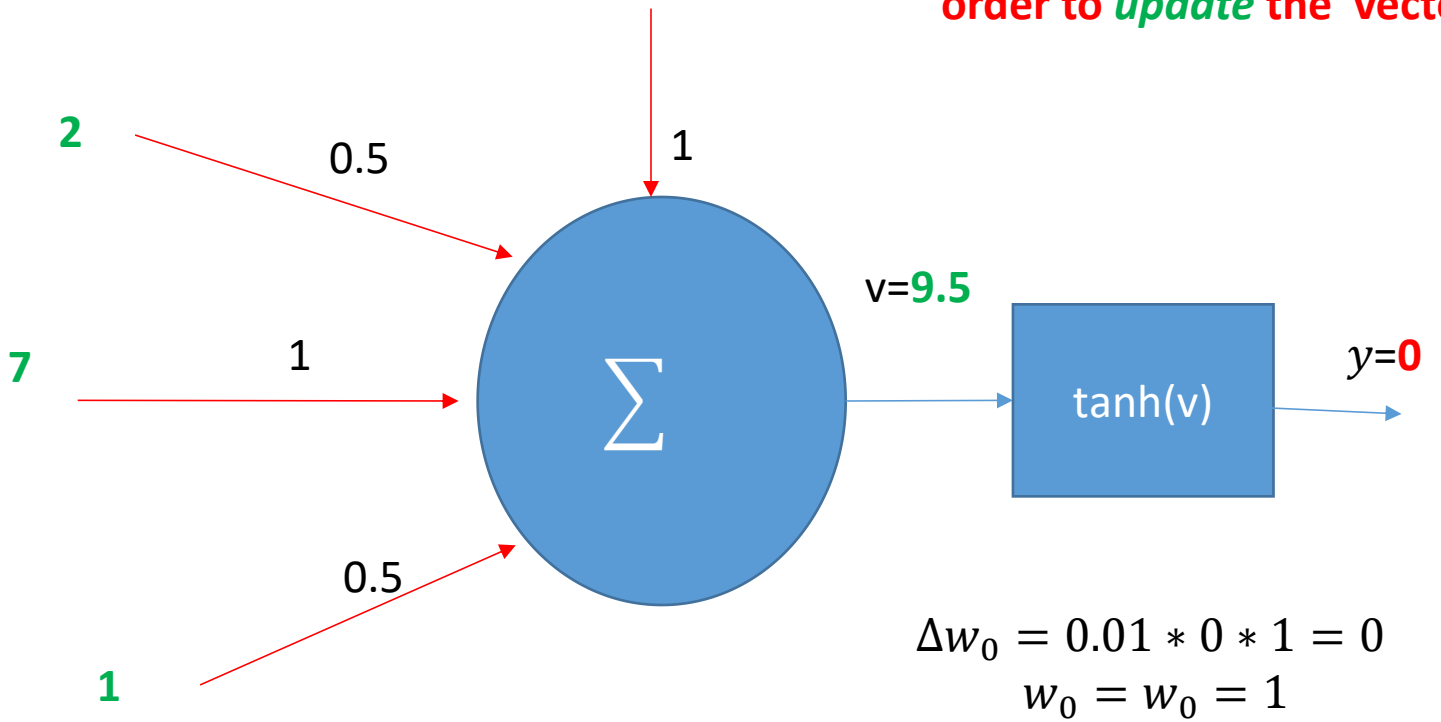
Update $w_i = w_i + \Delta w_i$

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

$$\Delta w_0 = 0.01 * 0 * 1 = 0$$
$$w_0 = w_0 = 1$$

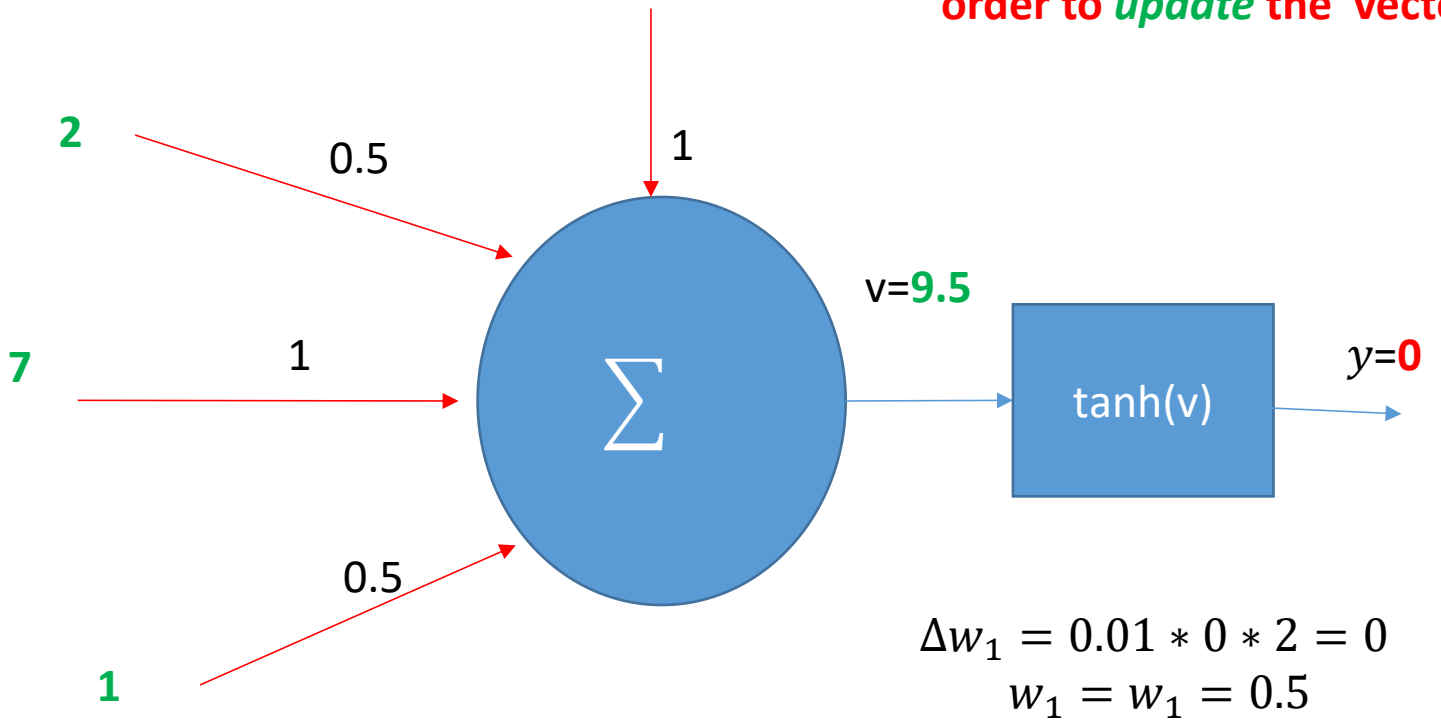
$$W = [1$$

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

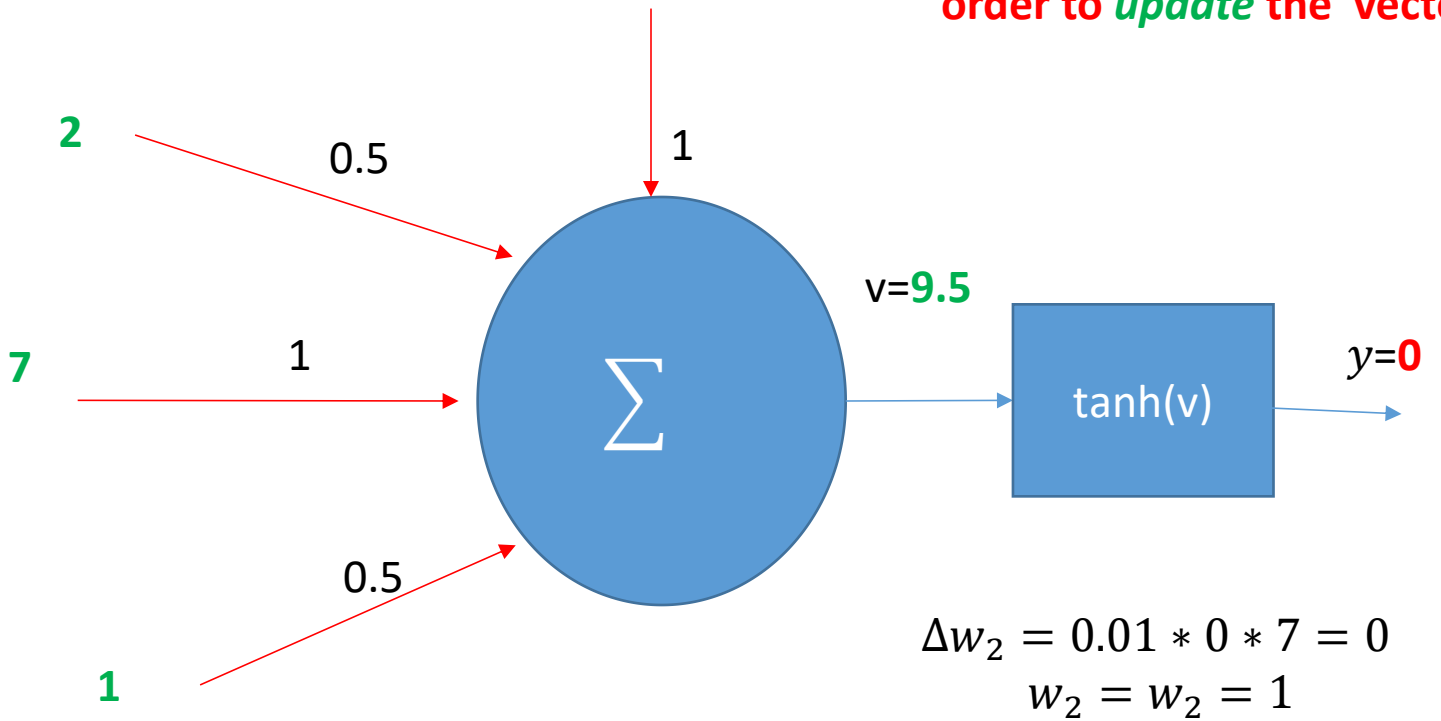
$$\Delta w_1 = 0.01 * 0 * 2 = 0$$
$$w_1 = w_1 = 0.5$$

$$W = [1, 0.5$$

Artificial Neuron

- Training a neuron (perceptron).

Step 5: The value of y is evaluated in order to *update* the vector W.



X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

$$\Delta w_2 = 0.01 * 0 * 7 = 0$$
$$w_2 = w_2 = 1$$

$$W = [1,\ 0.5,1,$$

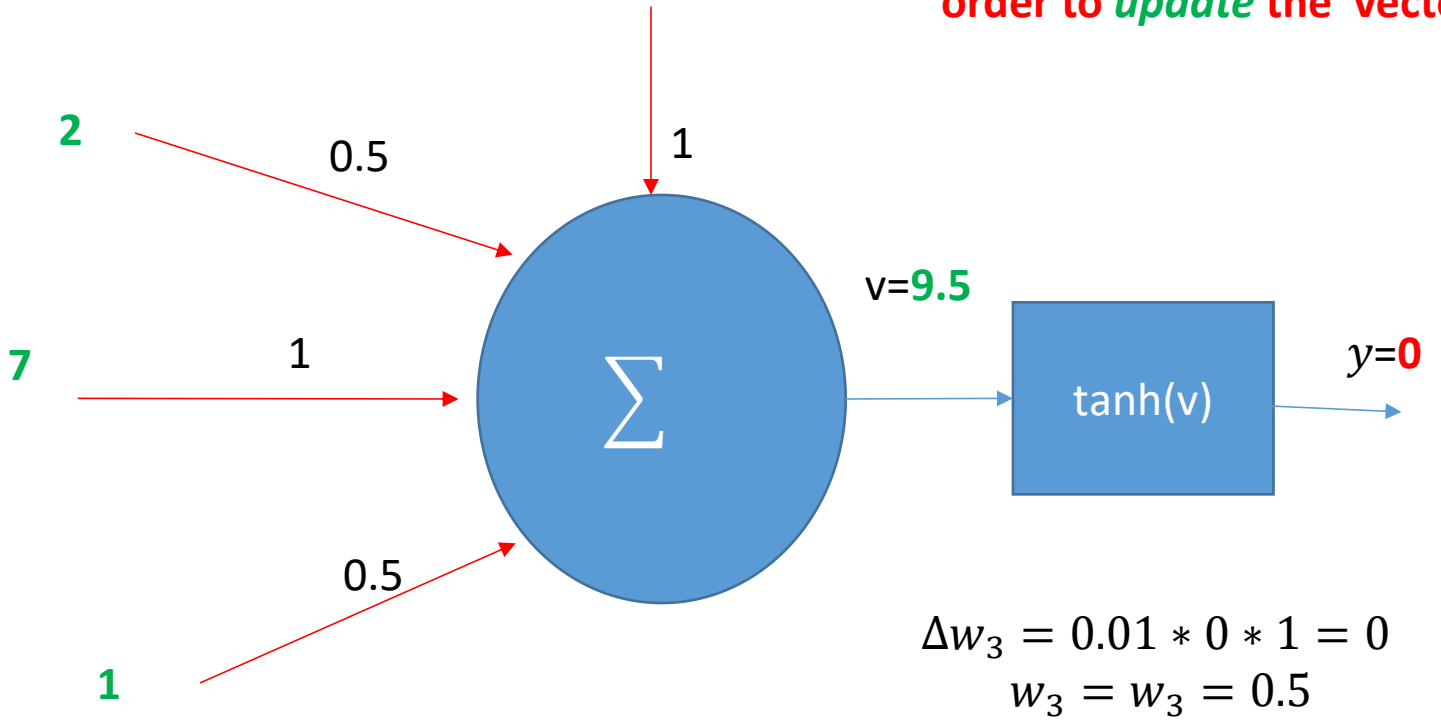
Artificial Neuron

- Training a neuron (perceptron).

Step 5: The value of y is evaluated in order to *update* the vector W.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



$$\Delta w_3 = 0.01 * 0 * 1 = 0$$
$$w_3 = w_3 = 0.5$$

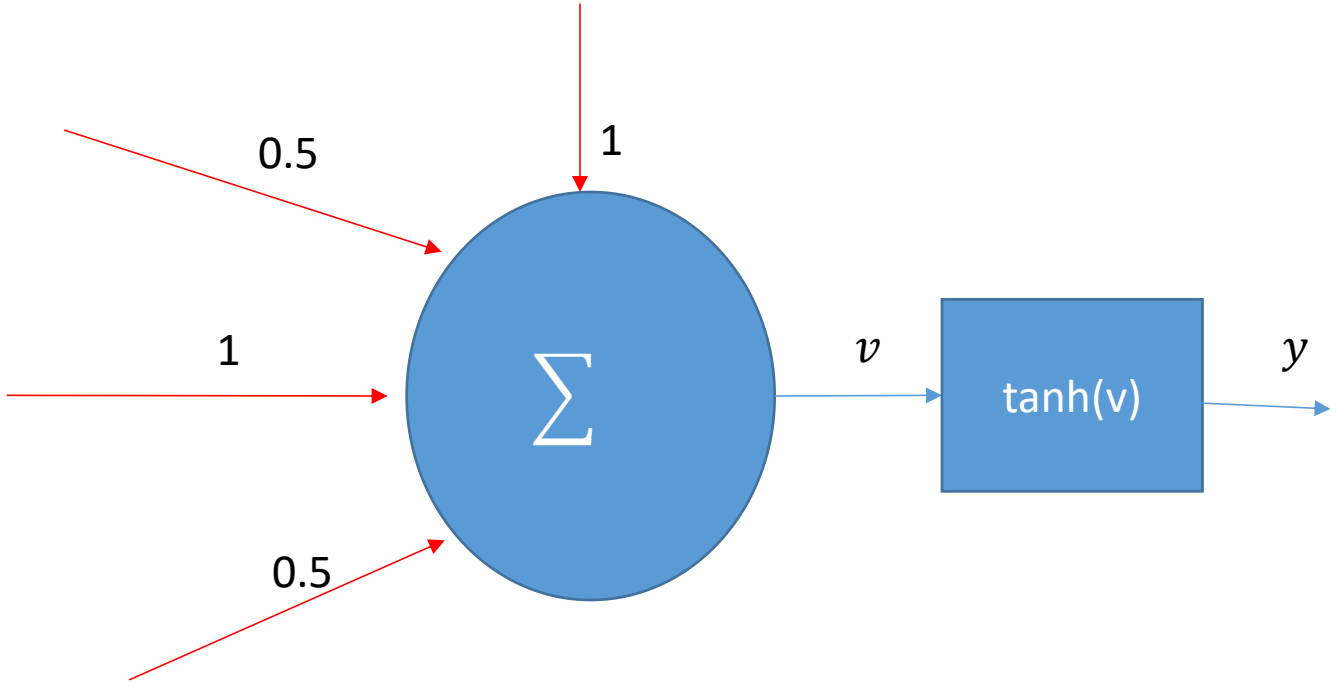
$$W = [1,\ 0.5,1,0.5]$$

In this case W does not change, Repeat steps 2-5 again with **other pattern**

Artificial Neuron

- Training a neuron (perceptron).

Step 2: The first pattern is presented to the perceptron



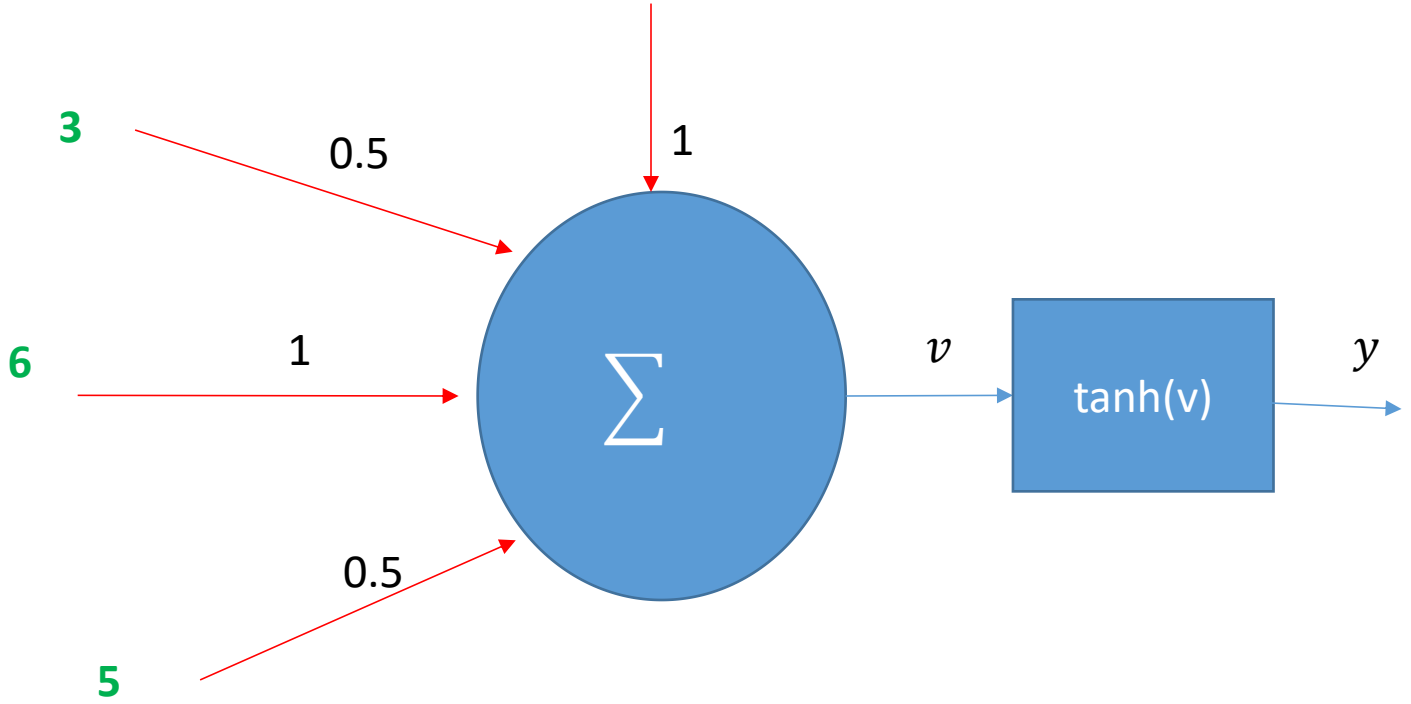
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 2: The first pattern is presented to the perceptron



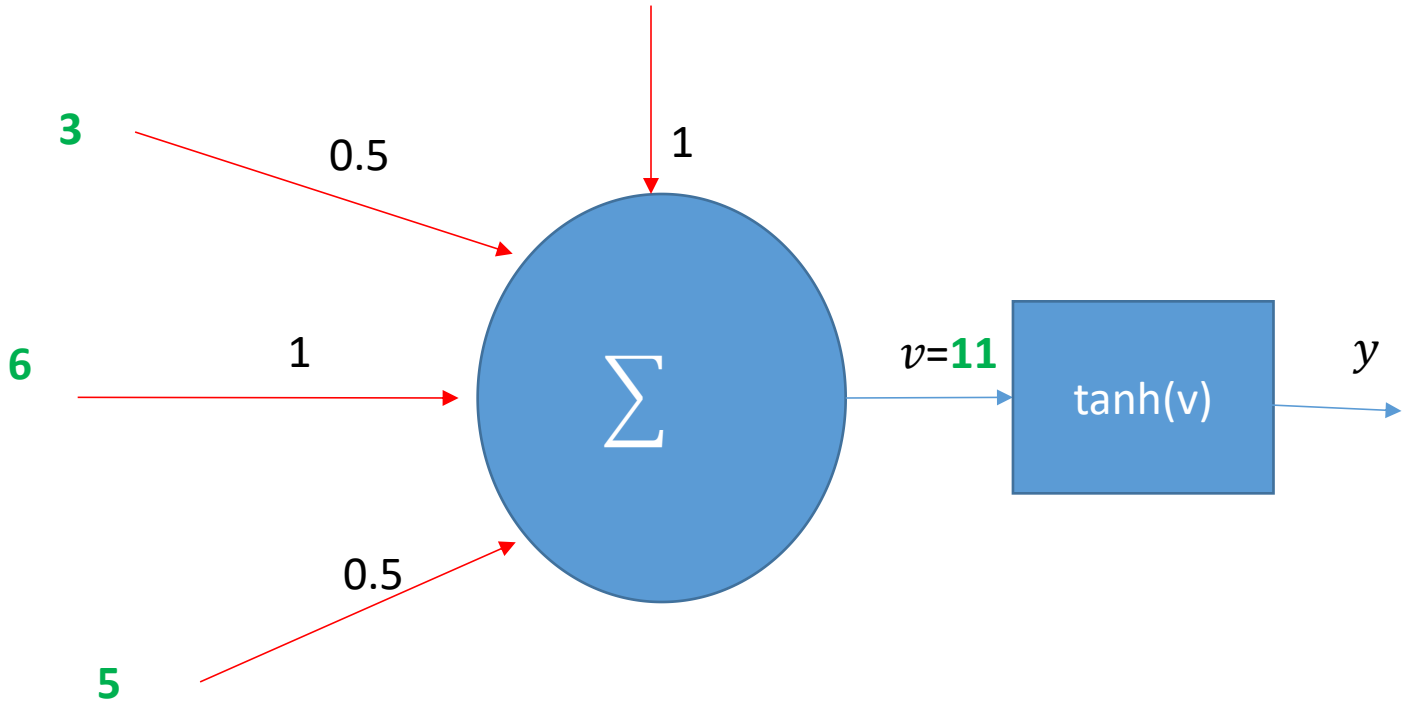
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Artificial Neuron

- Training a neuron (perceptron).

Step 3: The output of the perceptron is determined.



X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

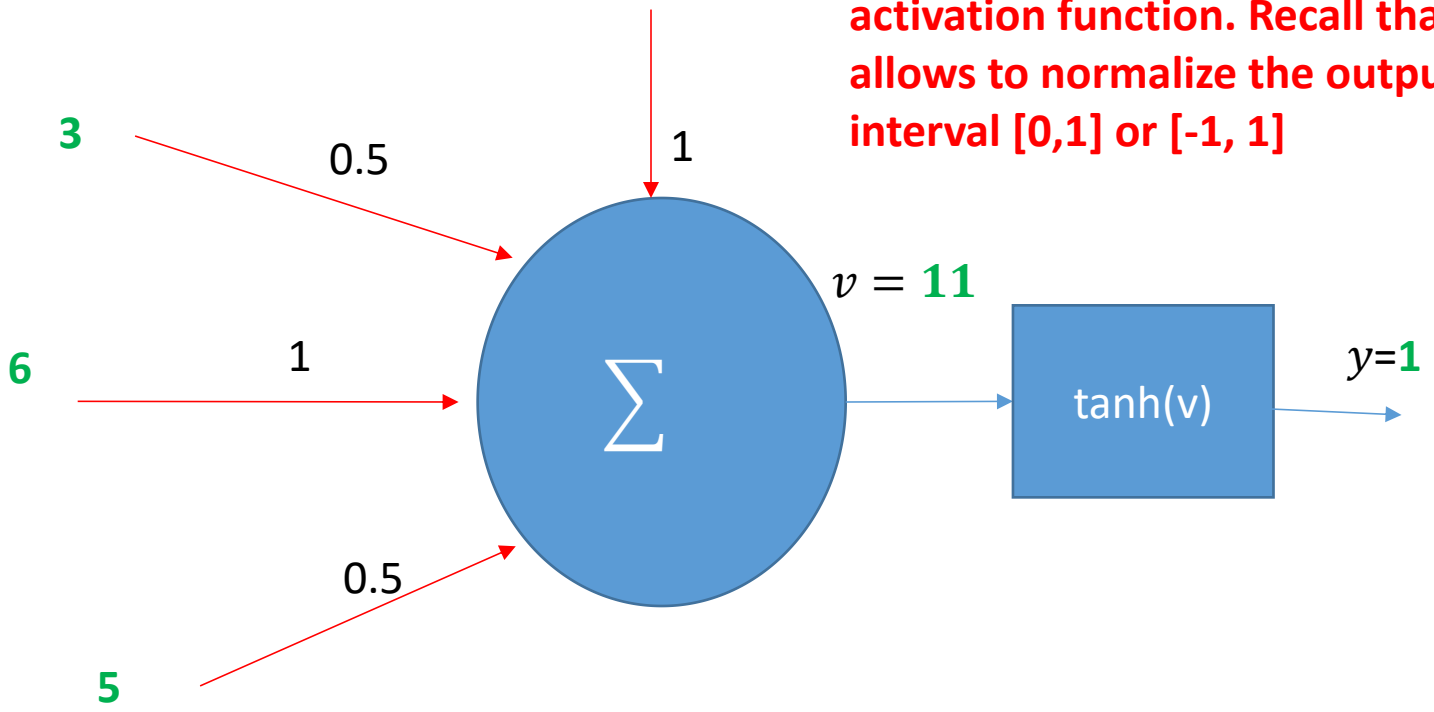
Dataset

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

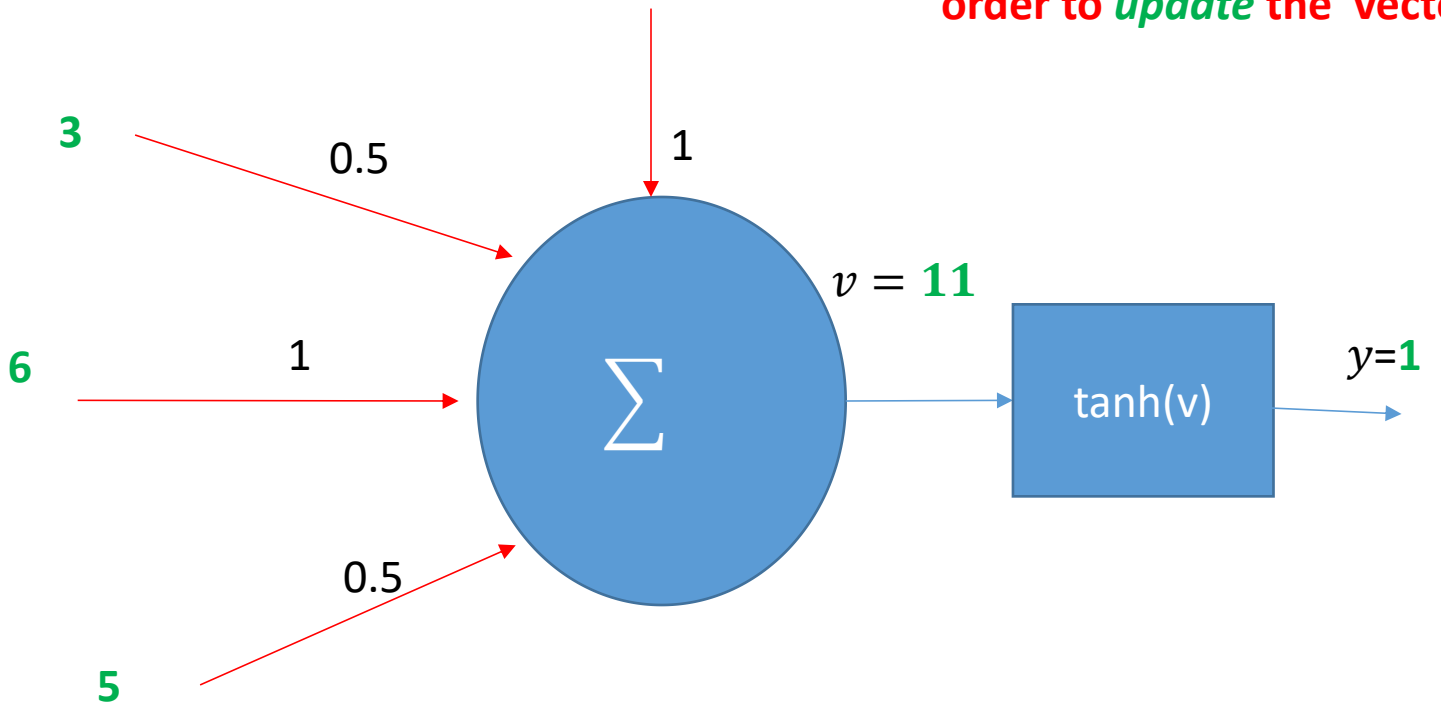


Step 4: The value of y is calculated based on activation function. Recall that such function allows to normalize the output as an closed interval [0,1] or [-1, 1]

Artificial Neuron

- Training a neuron (perceptron).

Step 5: The value of y is evaluated in order to *update* the vector W.



X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

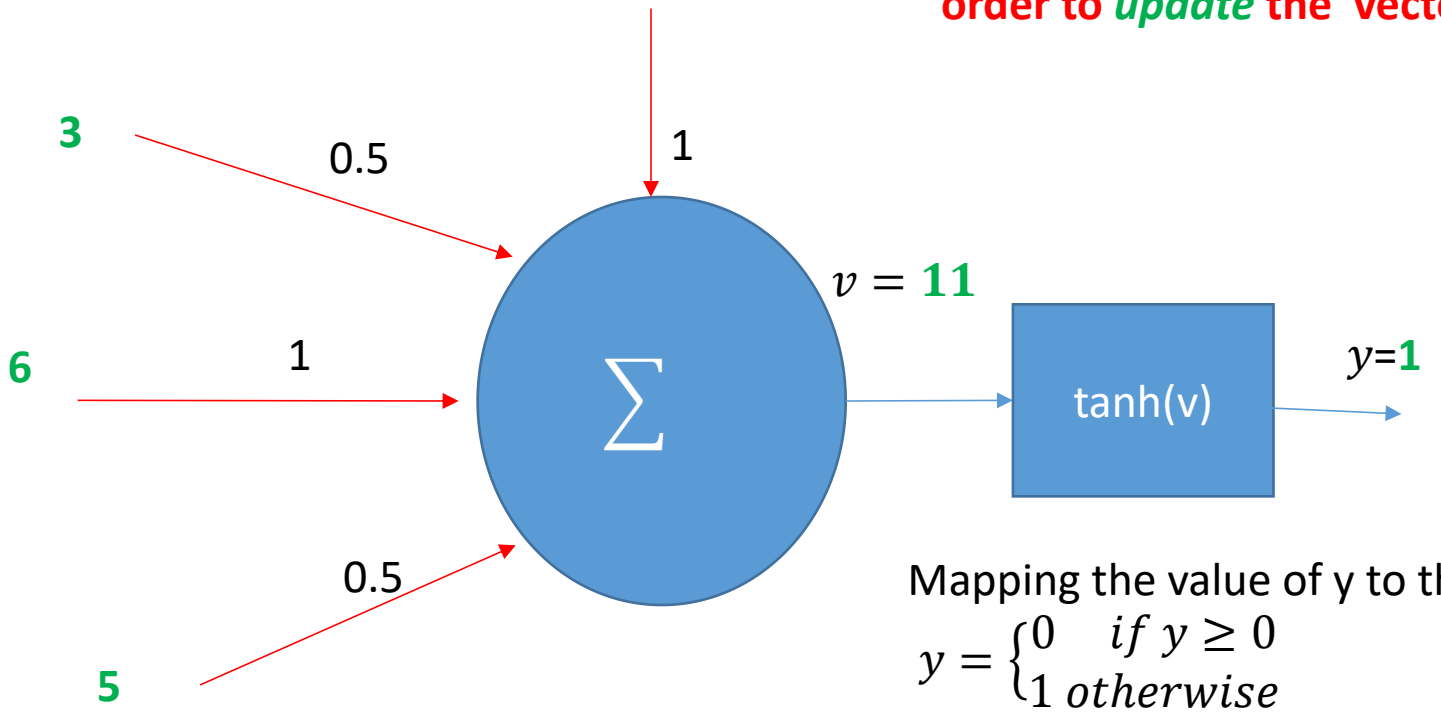
Dataset

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

Mapping the value of y to the class label.

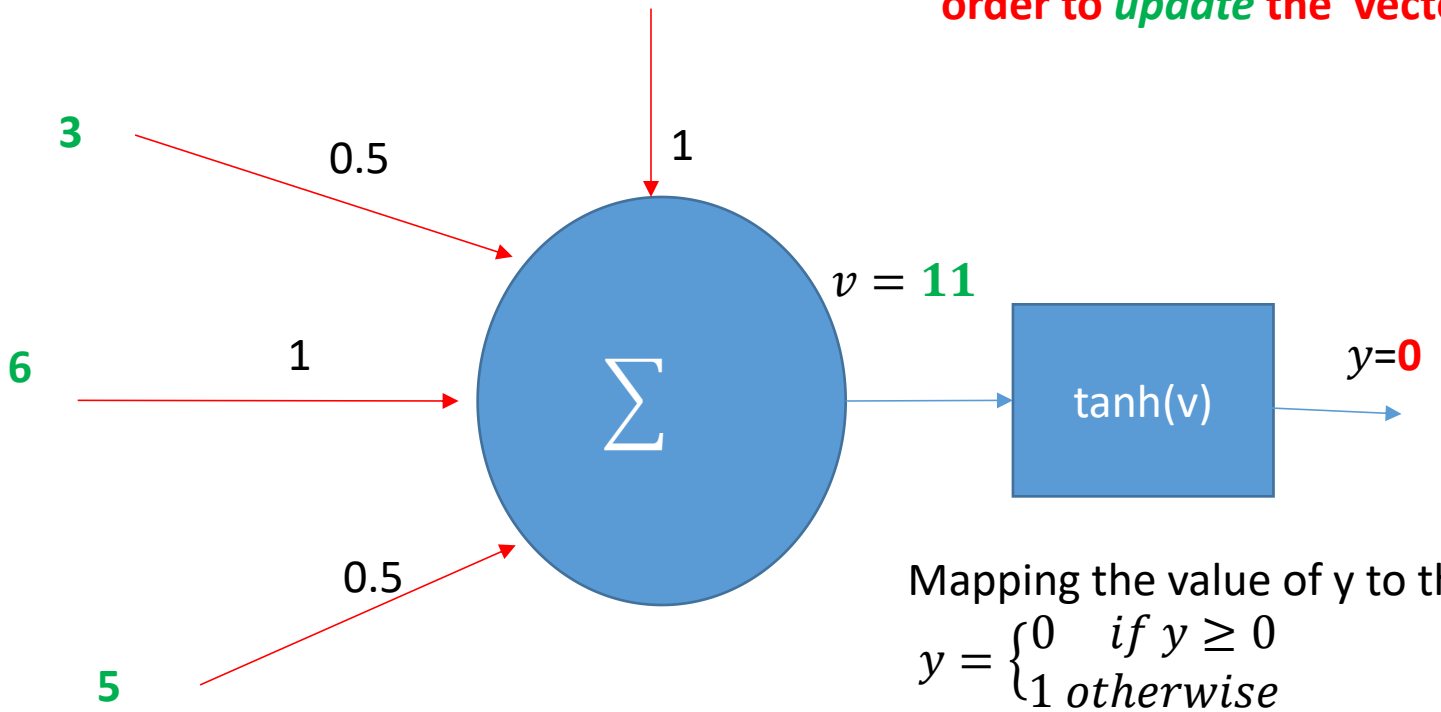
$$y = \begin{cases} 0 & if\ y \geq 0 \\ 1 & otherwise \end{cases}$$

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

Mapping the value of y to the class label.

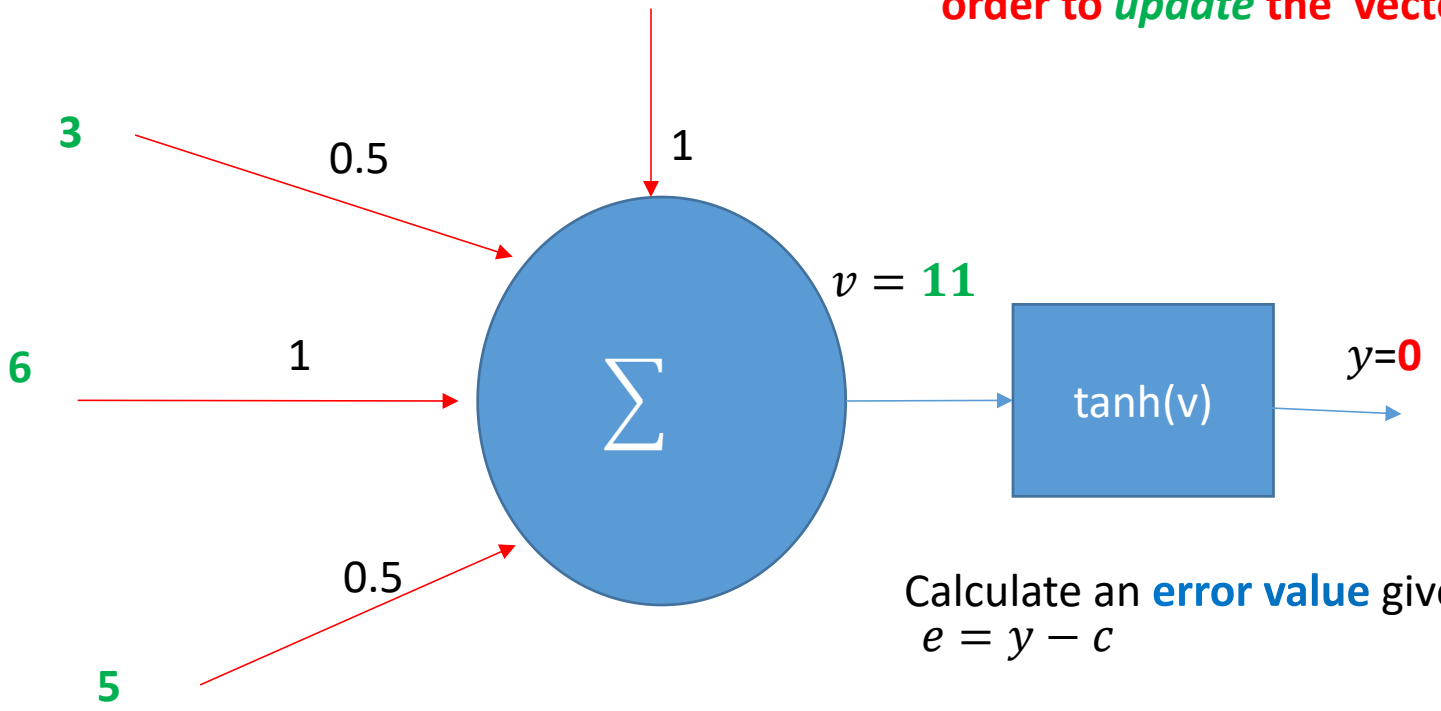
$$y = \begin{cases} 0 & if\ y \geq 0 \\ 1 & otherwise \end{cases}$$

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

Calculate an **error value** given the expected class c

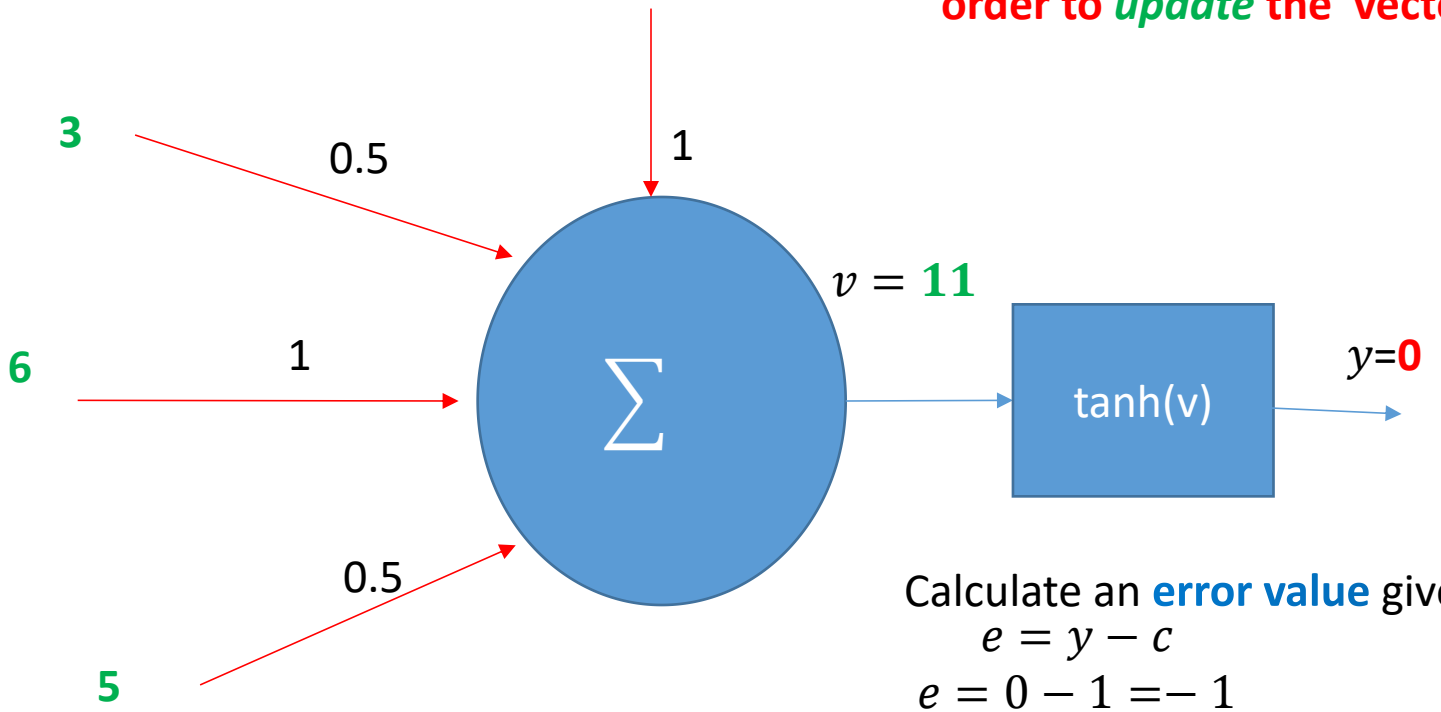
$$e = y - c$$

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

Calculate an **error value** given the expected class c

$$e = y - c$$

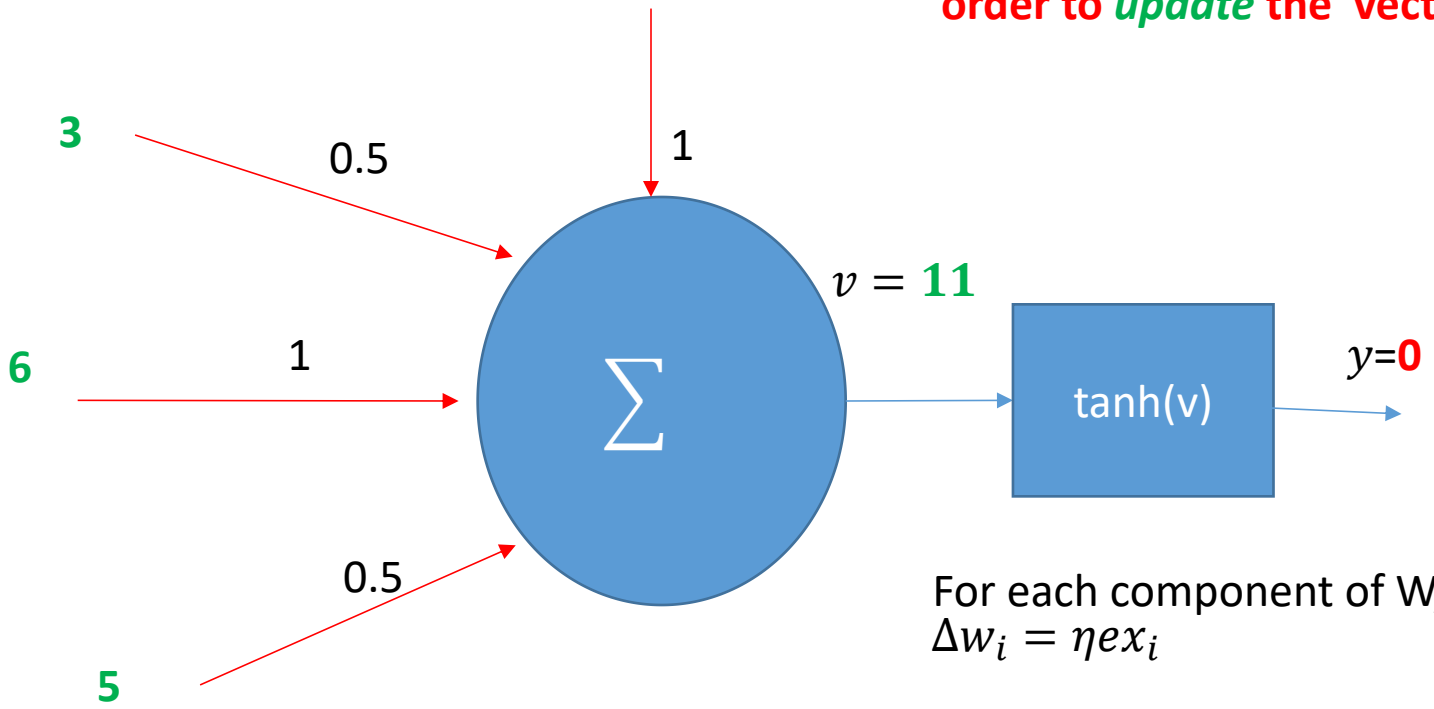
$$e = 0 - 1 = -1$$

Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



Step 5: The value of y is evaluated in order to *update* the vector W.

For each component of W, calculate a variation Δw_i

$\Delta w_i = \eta e x_i$

Where η is a perturbation value, usually it is a small positive value. Assume η=0.01

Update $w_i = w_i + \Delta w_i$

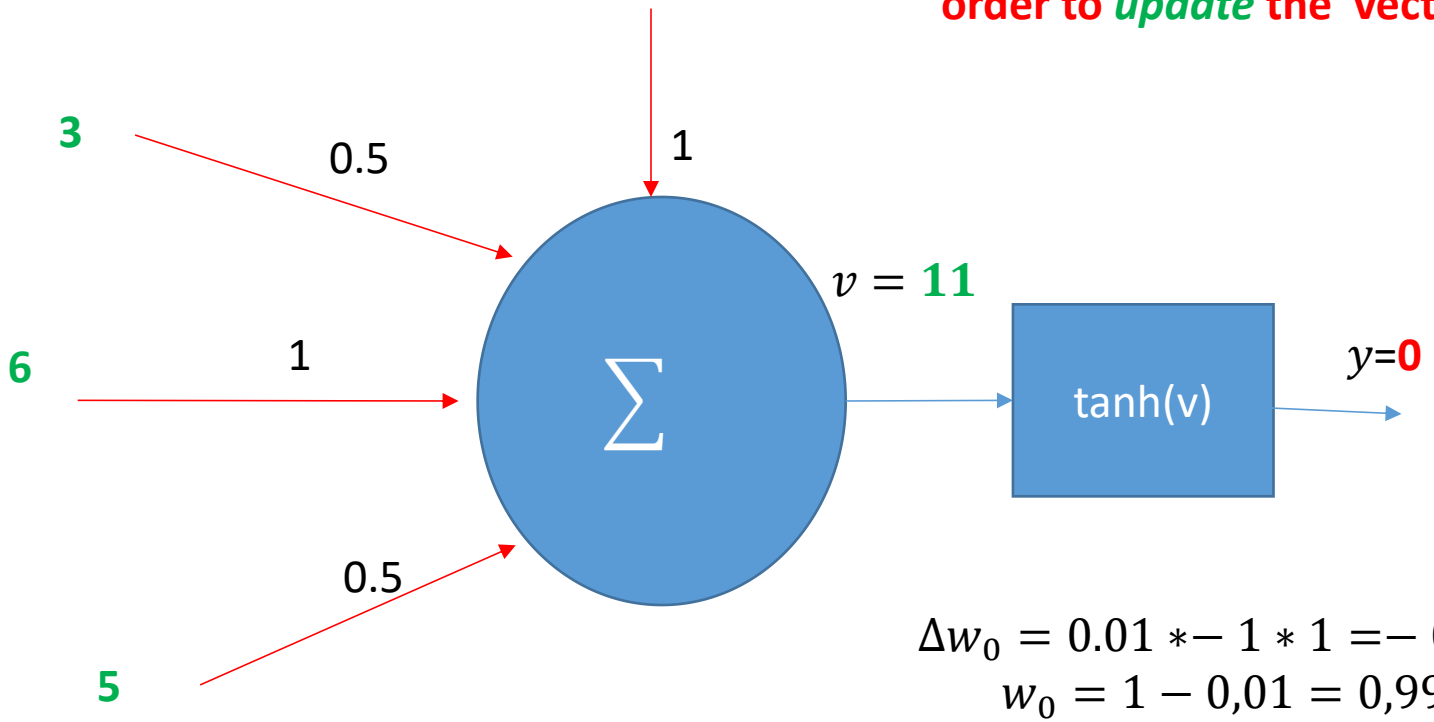
Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Step 5: The value of y is evaluated in order to *update* the vector W.



$$\Delta w_0 = 0.01 * -1 * 1 = -0{,}01$$
$$w_0 = 1 - 0{,}01 = 0{,}99$$

$$W = [0.99$$

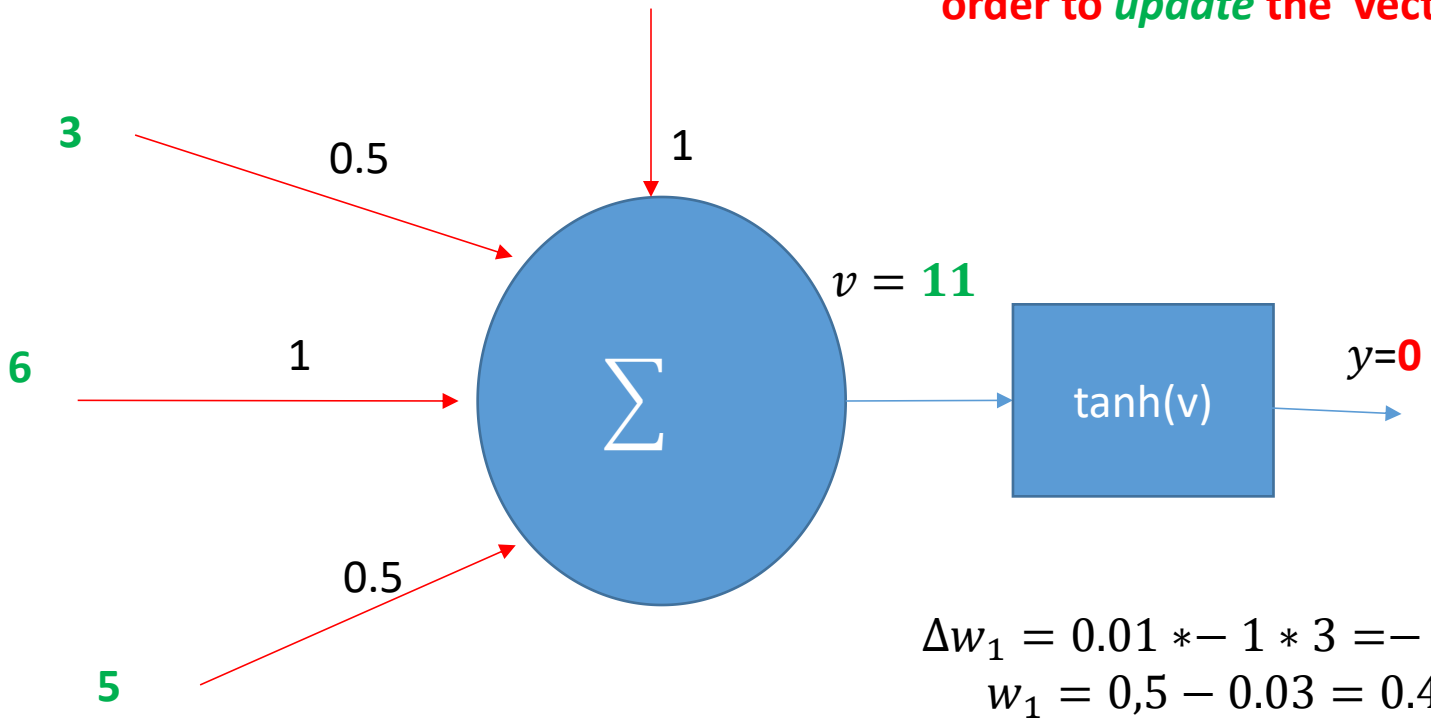
Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Step 5: The value of y is evaluated in order to *update* the vector W.



$$\Delta w_1 = 0.01 * -1 * 3 = -0.03$$
$$w_1 = 0{,}5 - 0.03 = 0.47$$

$$W = [0.97, 0.47,$$

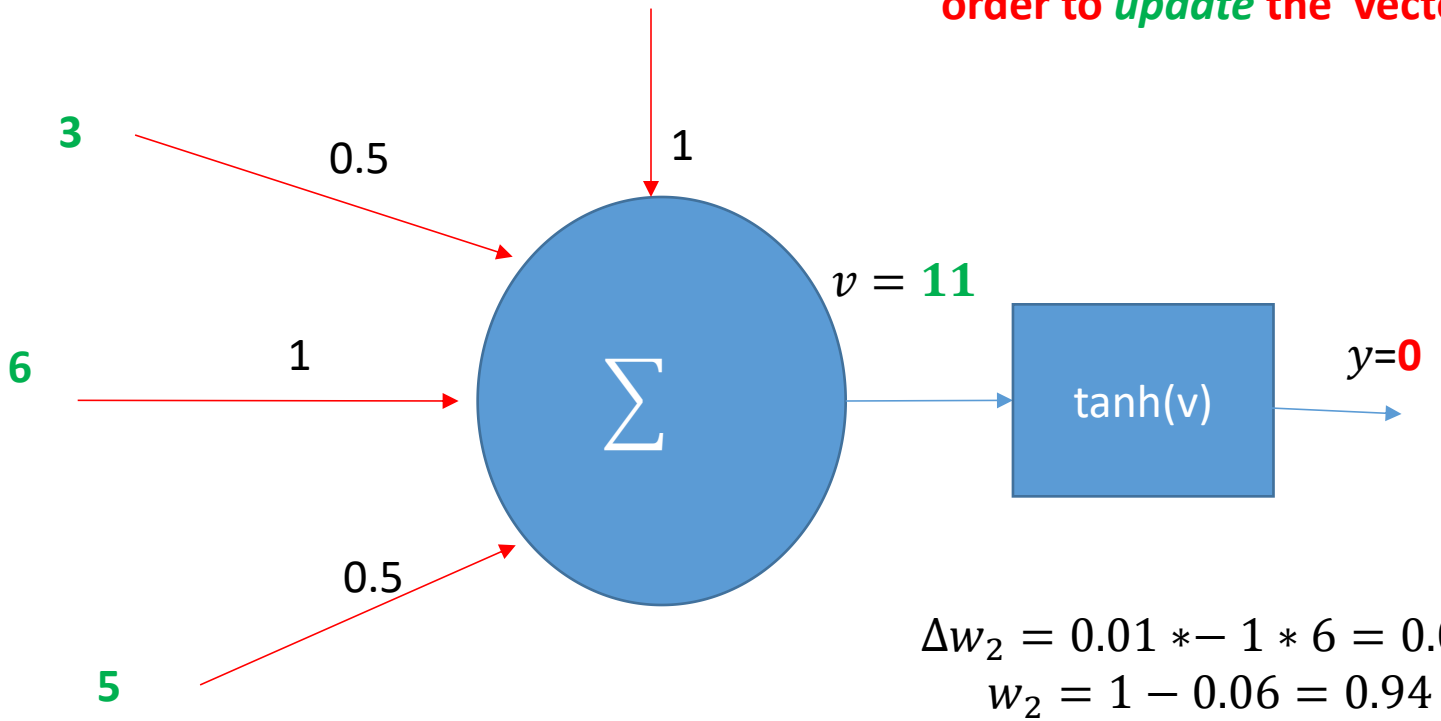
Artificial Neuron

- Training a neuron (perceptron).

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset

Step 5: The value of y is evaluated in order to *update* the vector W.



$$\Delta w_2 = 0.01 * -1 * 6 = 0.06$$
$$w_2 = 1 - 0.06 = 0.94$$

$$W = [0.97, 0.44,\ 0,94$$

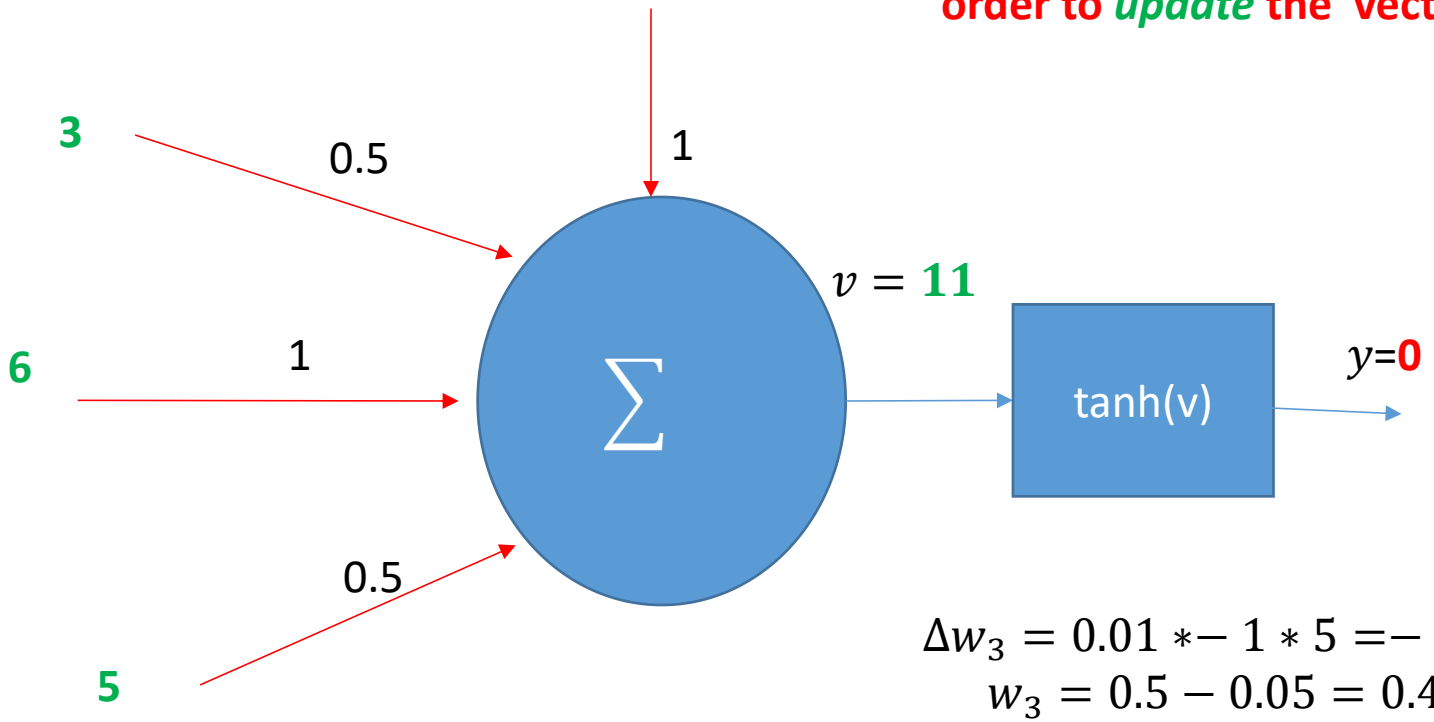
Artificial Neuron

- Training a neuron (perceptron).

Step 5: The value of y is evaluated in order to *update* the vector W.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	0
6	8	5	0
7	9	7	1
8	0	8	1

Dataset



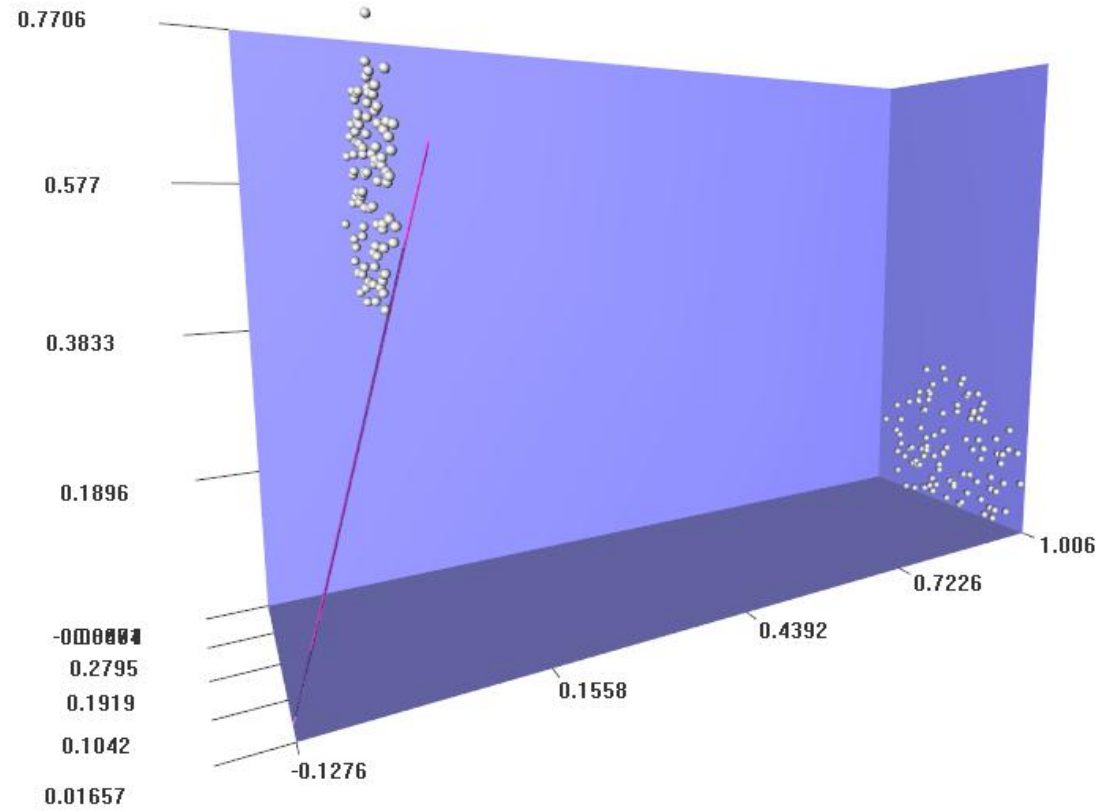
$v = 11$

$$\Delta w_3 = 0.01 * -1 * 5 = -0.05$$
$$w_3 = 0.5 - 0.05 = 0.45$$

$$W = [0.97, 0.44,\ 0, 94, 0.45]$$

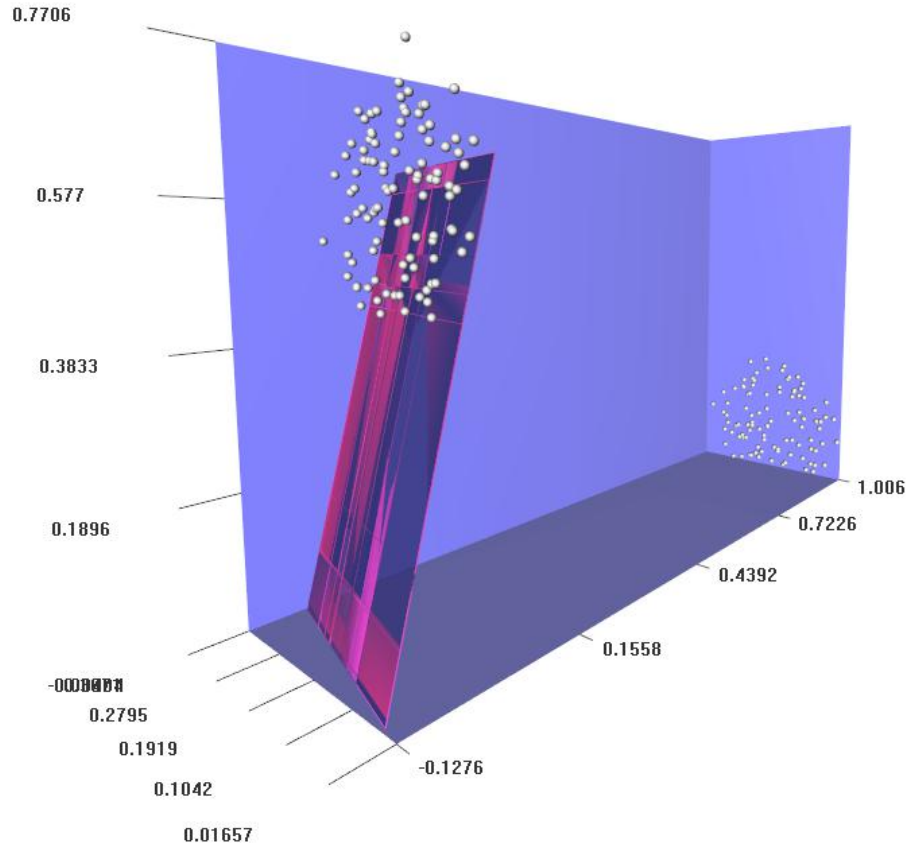
Artificial Neuron

- Example of classification based on perceptron.



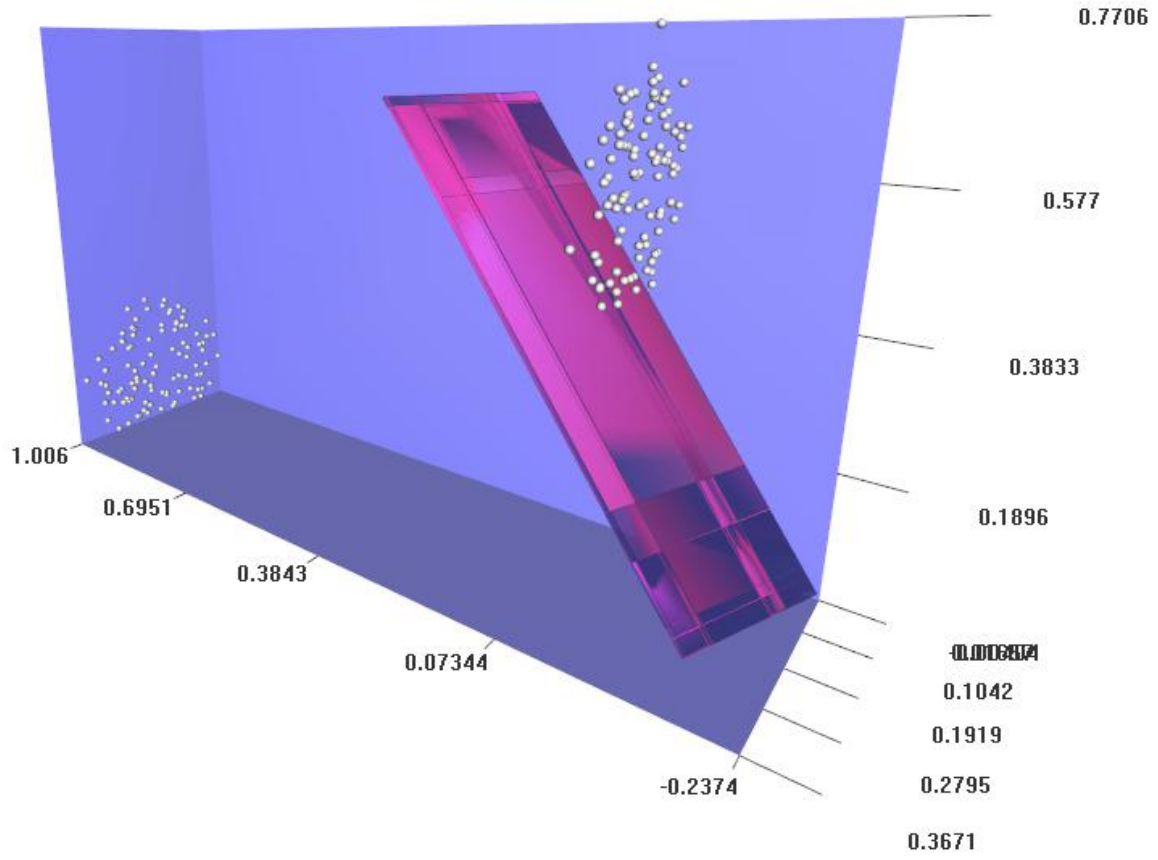
Artificial Neuron

- Example of classification based on perceptron.



Artificial Neuron

- Example of classification based on perceptron.



Artificial Neuron

- Although the perceptron rule finds a successful weight vector when the training ***n*** examples are linearly separable, it can fail to converge if the examples are not linearly separable.
- A second training rule, called the ***delta rule,*** is designed to overcome this difficulty.
- If the training examples are not linearly separable, the delta rule converges toward a best-fit approximation to the target concept.

Artificial Neuron (The Delta Rule)

- The key idea behind the delta rule is to use ***gradient descent*** to search the hypothesis space of possible weight vectors to find the weights that best fit the training examples.
- This rule is important because gradient descent provides the basis for the **BACKPROPAGATION** algorithm, which can learn networks with many interconnected neurons.

Artificial Neuron (The Delta Rule)

- The key idea behind the delta rule is to use ***gradient descent*** to search the hypothesis space of possible weight vectors to find the weights that best fit the training examples.
- This rule is important because gradient descent provides the basis for the **BACKPROPAGATION** algorithm, which can learn networks with many interconnected neurons.
- The delta training rule is best understood by considering the task of training an perceptron; that is, **a linear unit** for which the output o is given by:

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

Artificial Neuron (The Delta Rule)

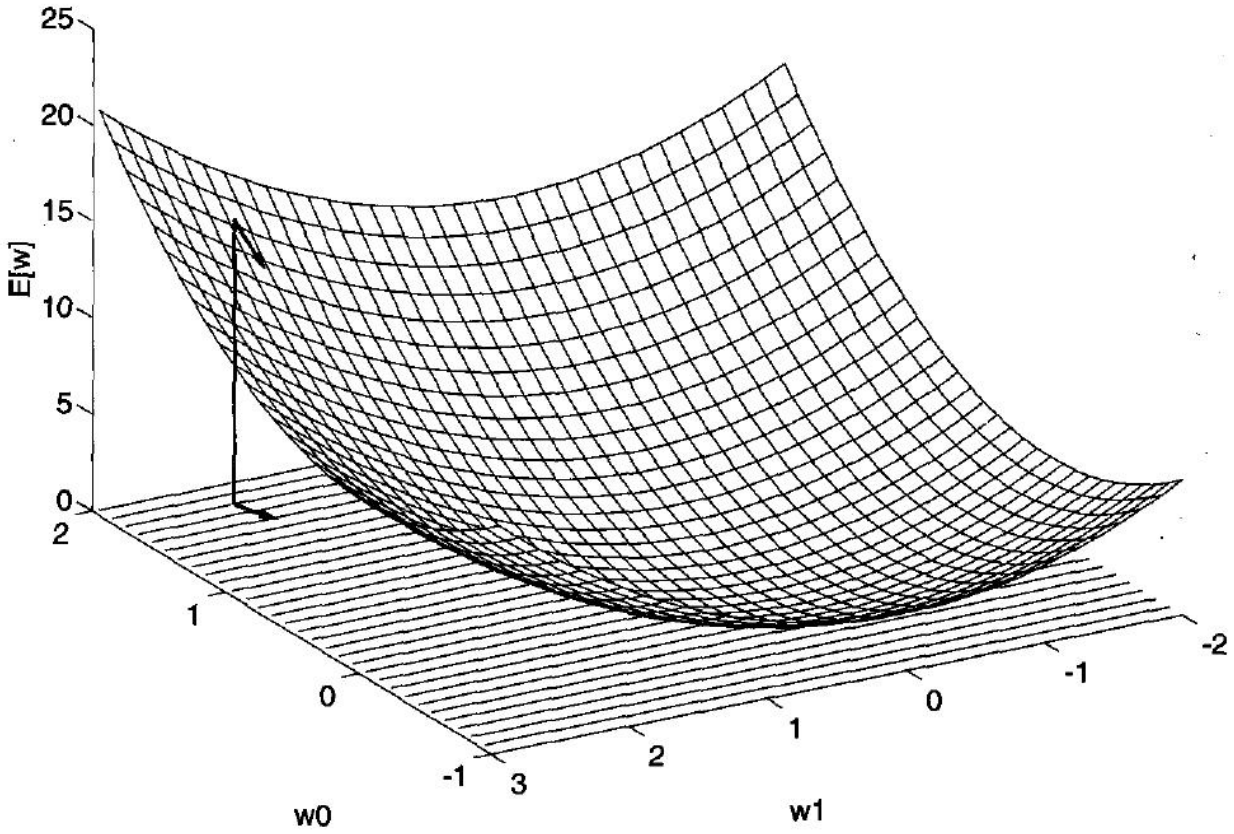
- The key idea behind the delta rule is to use ***gradient descent*** to search the hypothesis space of possible weight vectors to find the weights that best fit the training examples.
- We can derive a weight learning rule for linear units, specifying a measure for the ***training error*** of the weight vector, relative to the training examples.

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- Where t_d and o_d are the target output for training example the output of the linear unit for training example d in a dataset D.

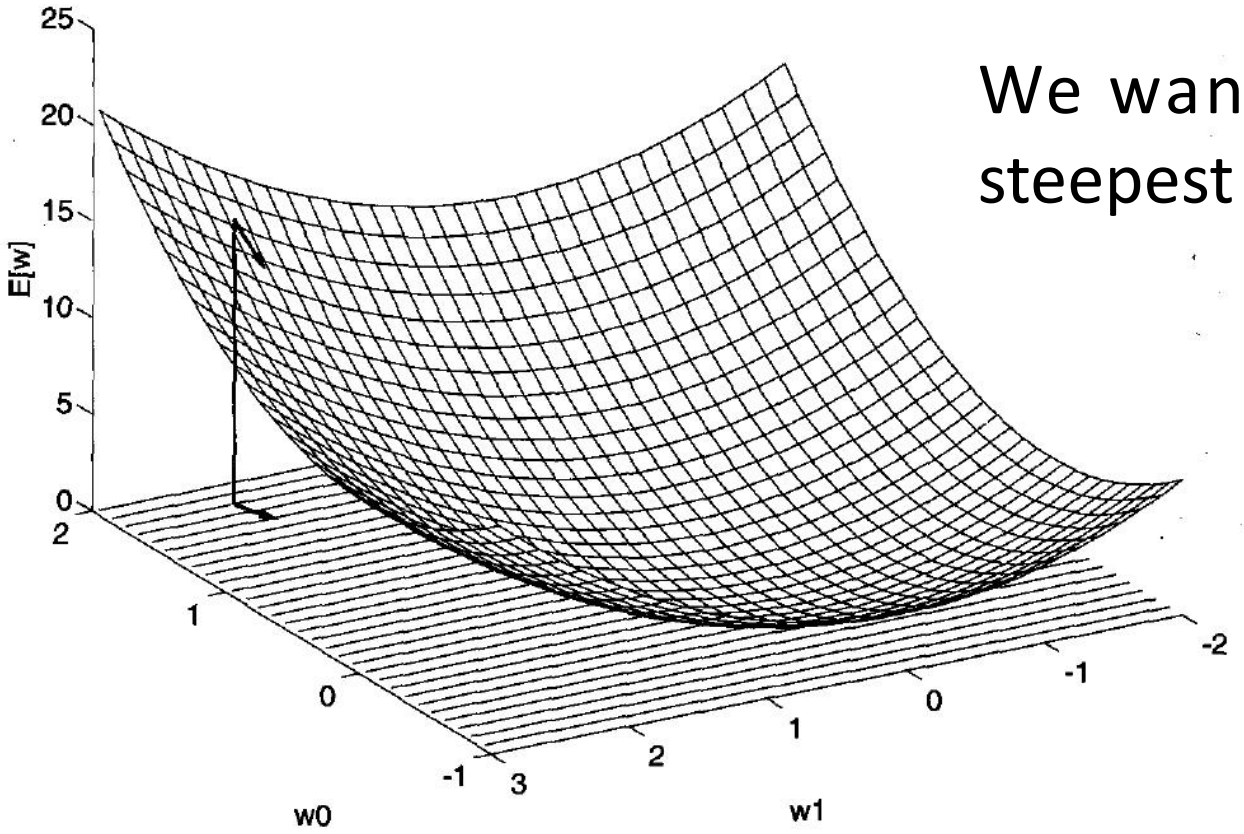
Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



Artificial Neuron (The Delta Rule)

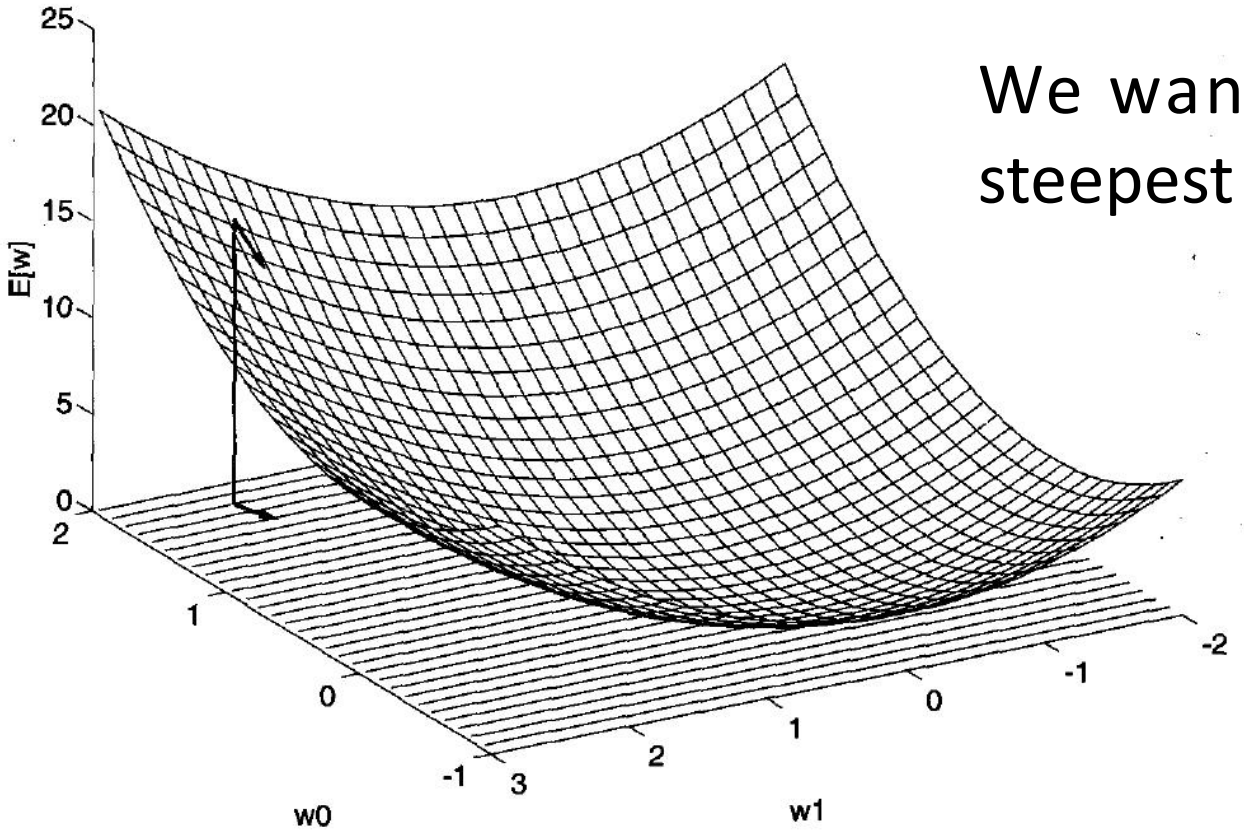
- Visualizing the gradient descendent method



We want to calculate the direction of steepest descent along the error surface.

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method

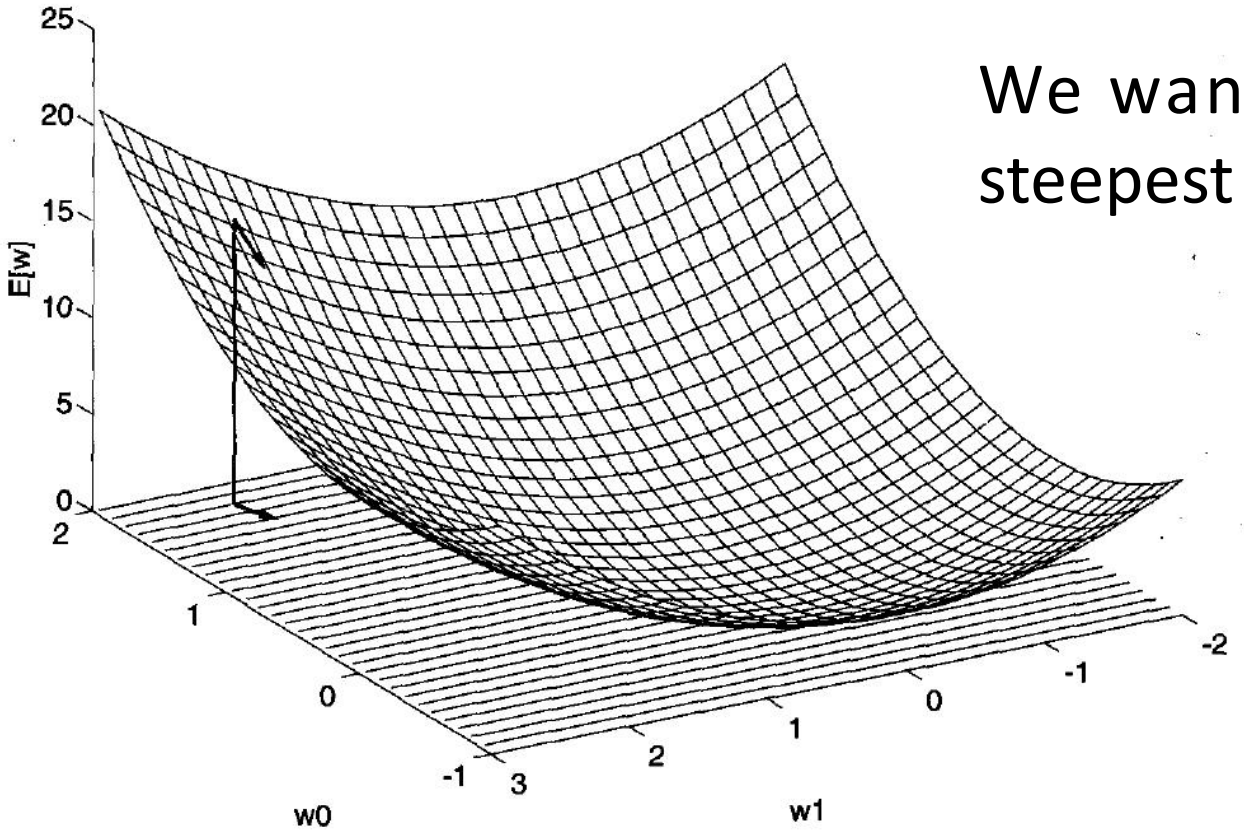


We want to calculate the direction of steepest descent along the error surface.

$$\nabla E(\vec{w})$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method

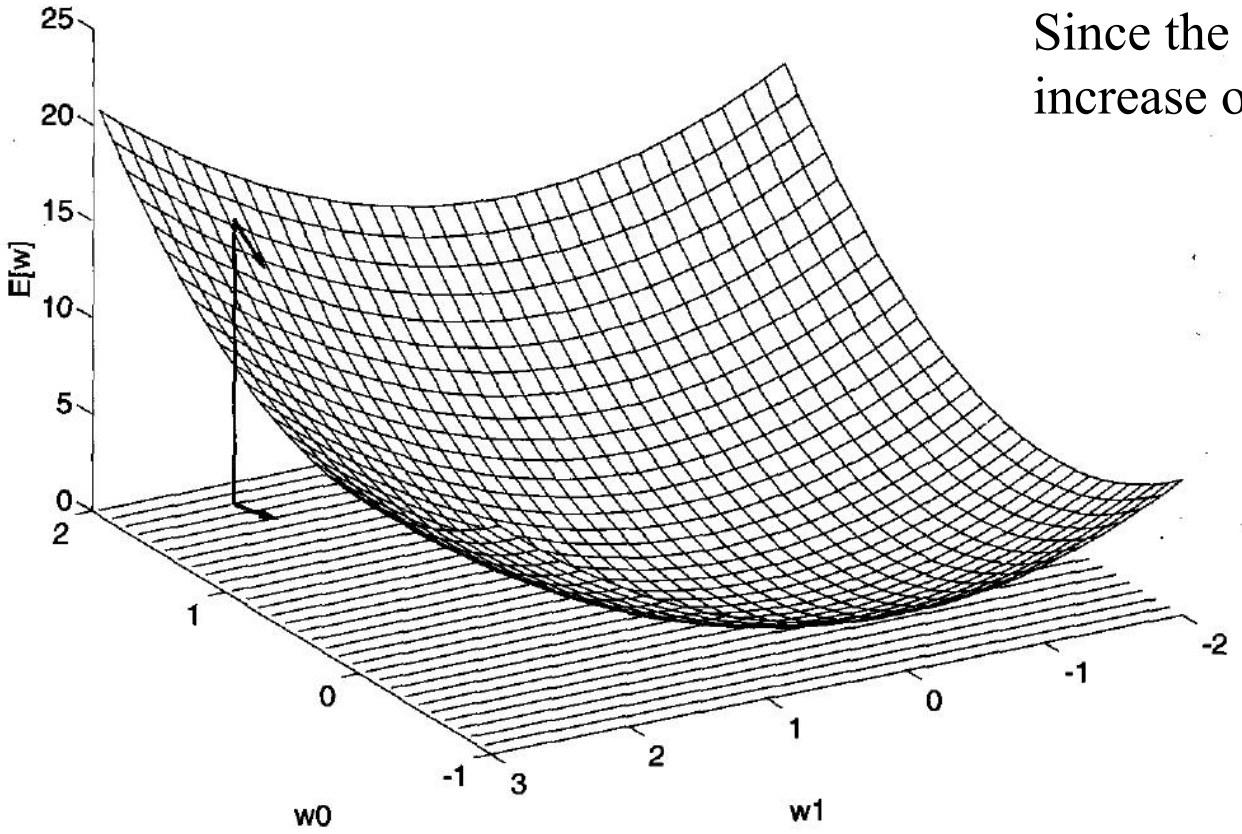


We want to calculate the direction of steepest descent along the error surface.

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n} \right]$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method

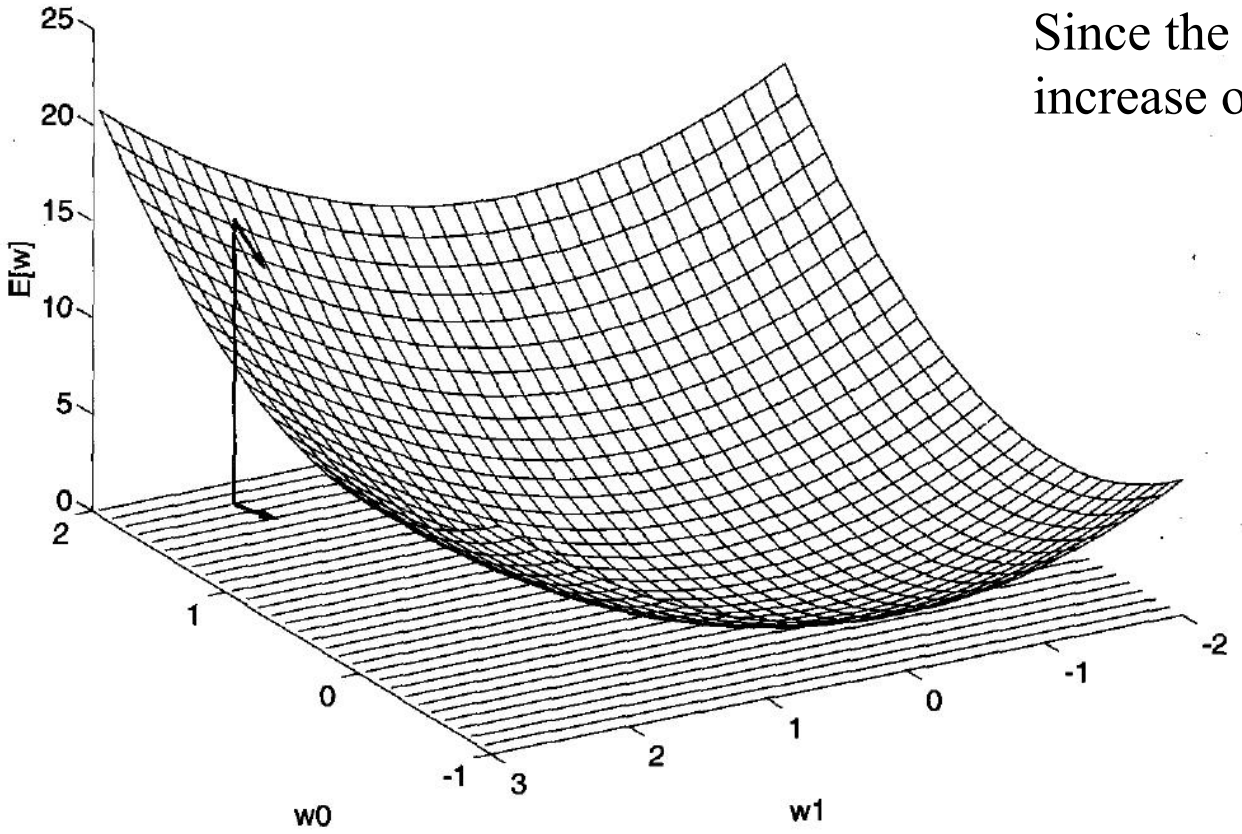


Since the gradient specifies the direction of steepest increase of ***E***, the training rule for gradient descent is:

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



Since the gradient specifies the direction of steepest increase of ***E***, the training rule for gradient descent is:

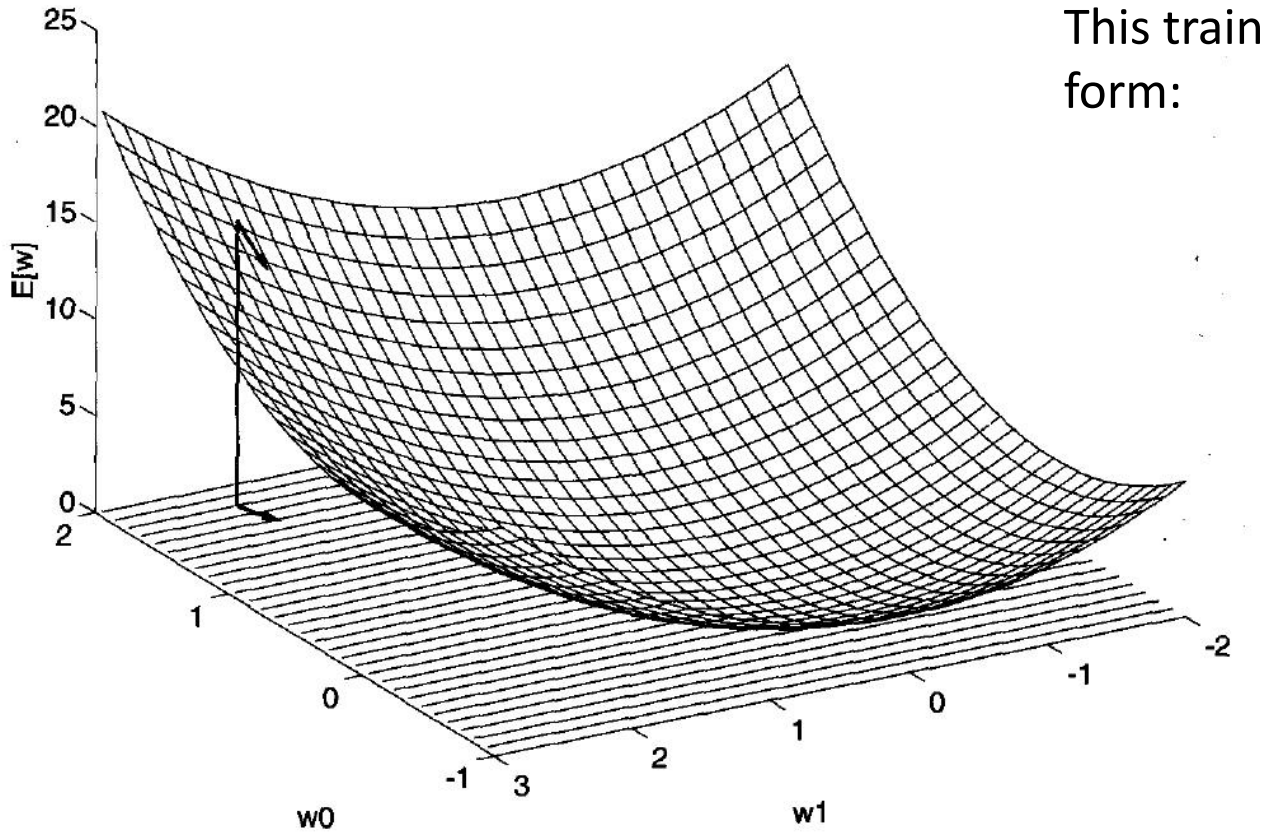
$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

where

$$\Delta\vec{w} = -\eta\nabla E(\vec{w})$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



This training rule can also be written in its component form:

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

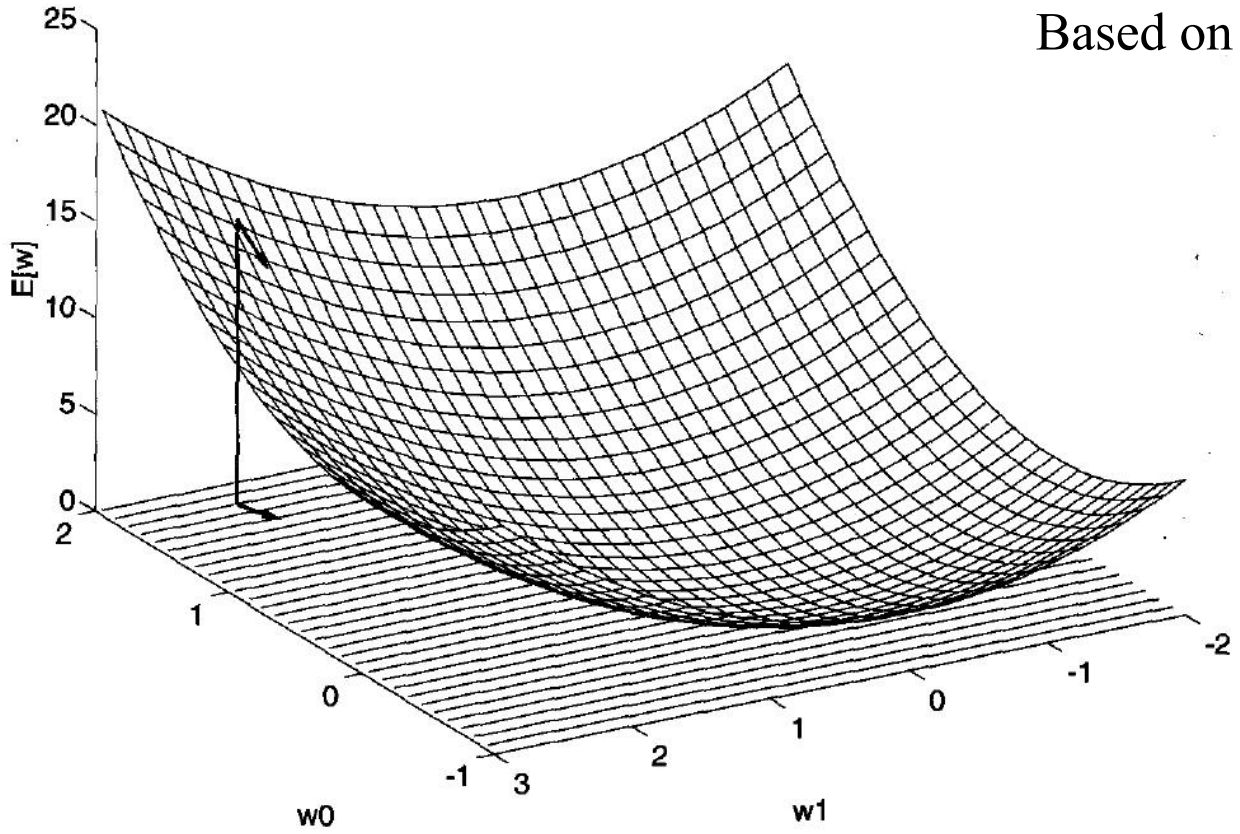
where

$$\Delta\vec{w} = -\eta\nabla E(\vec{w})$$

What it means the negative sign?

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



Based on the above the component representation is:

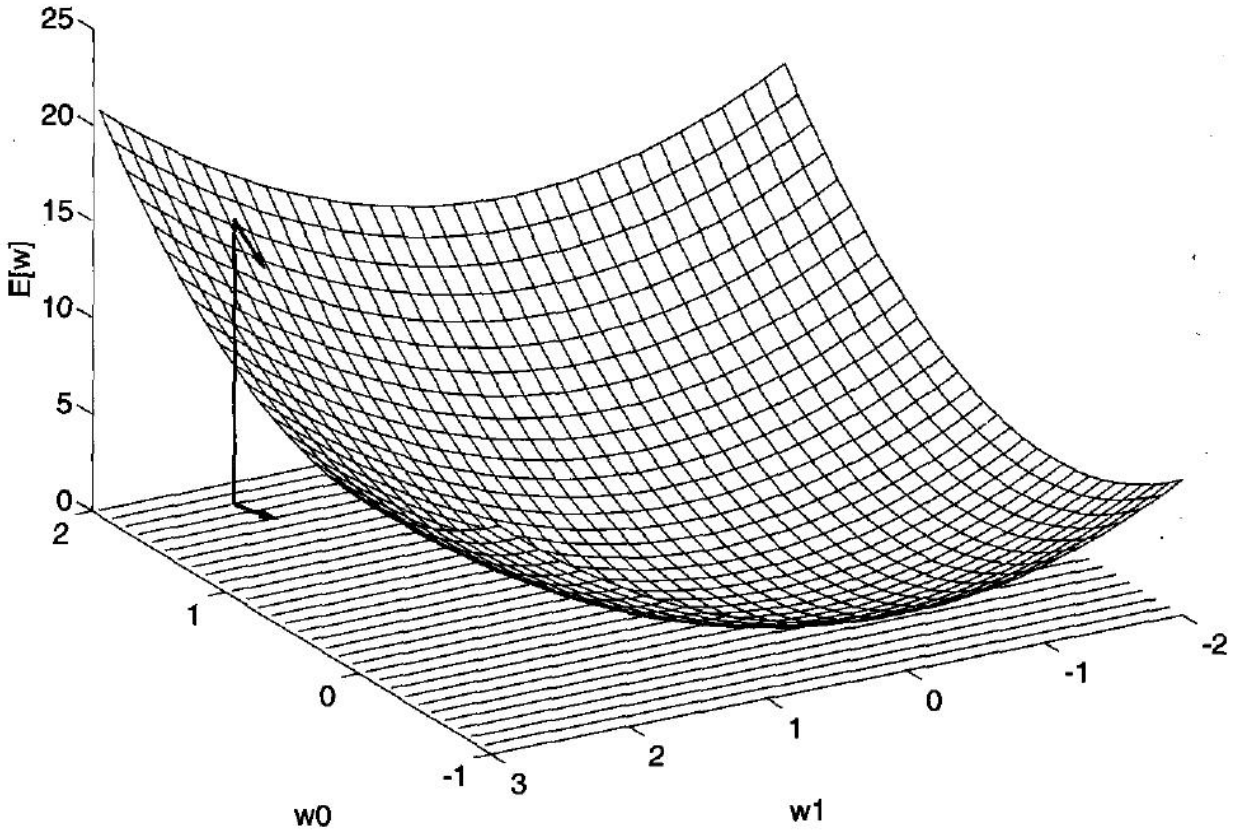
$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method

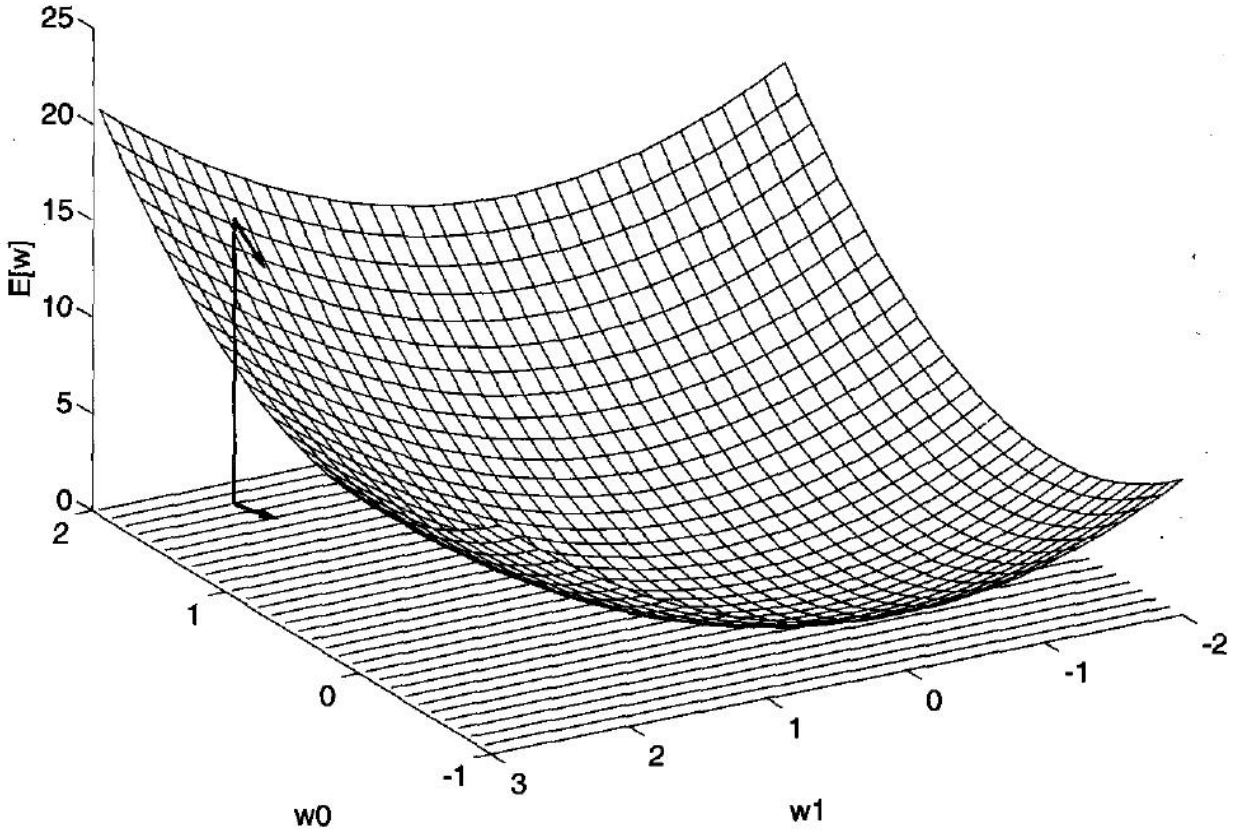


$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



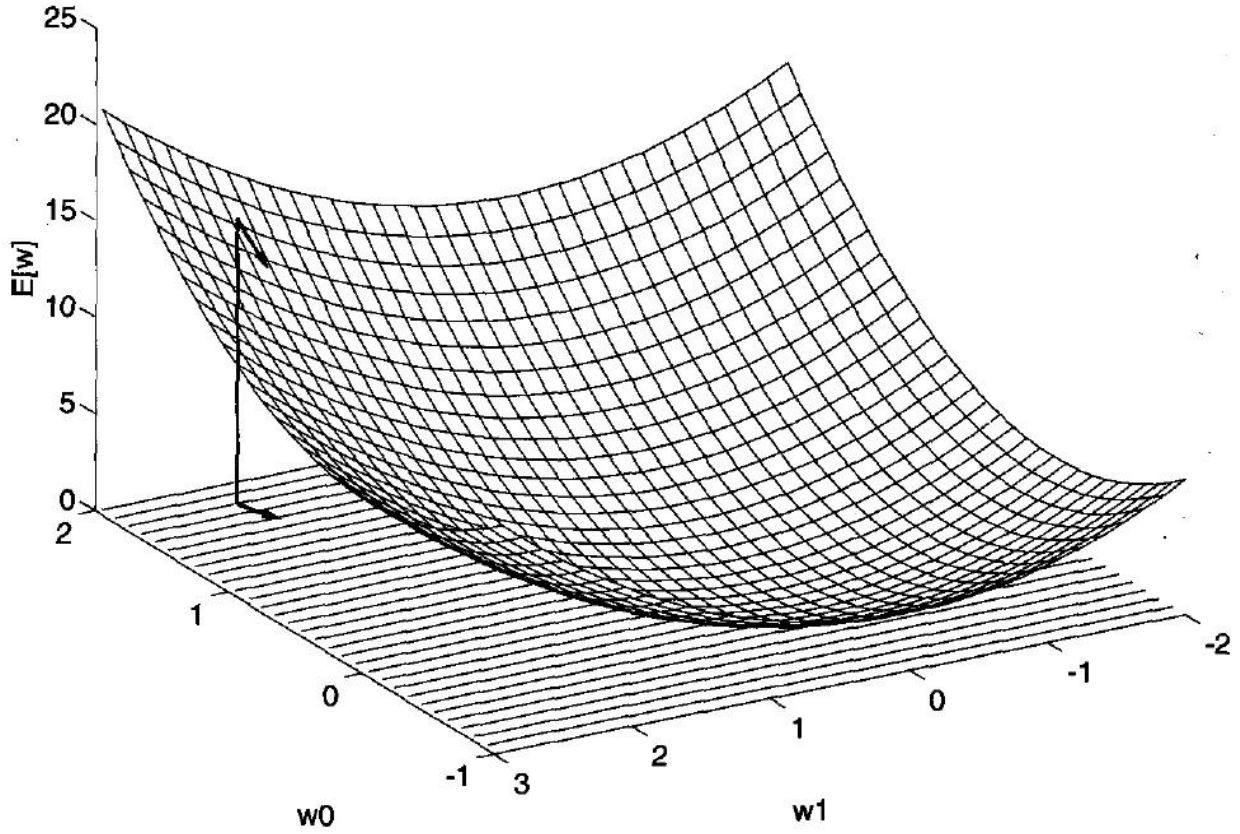
$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

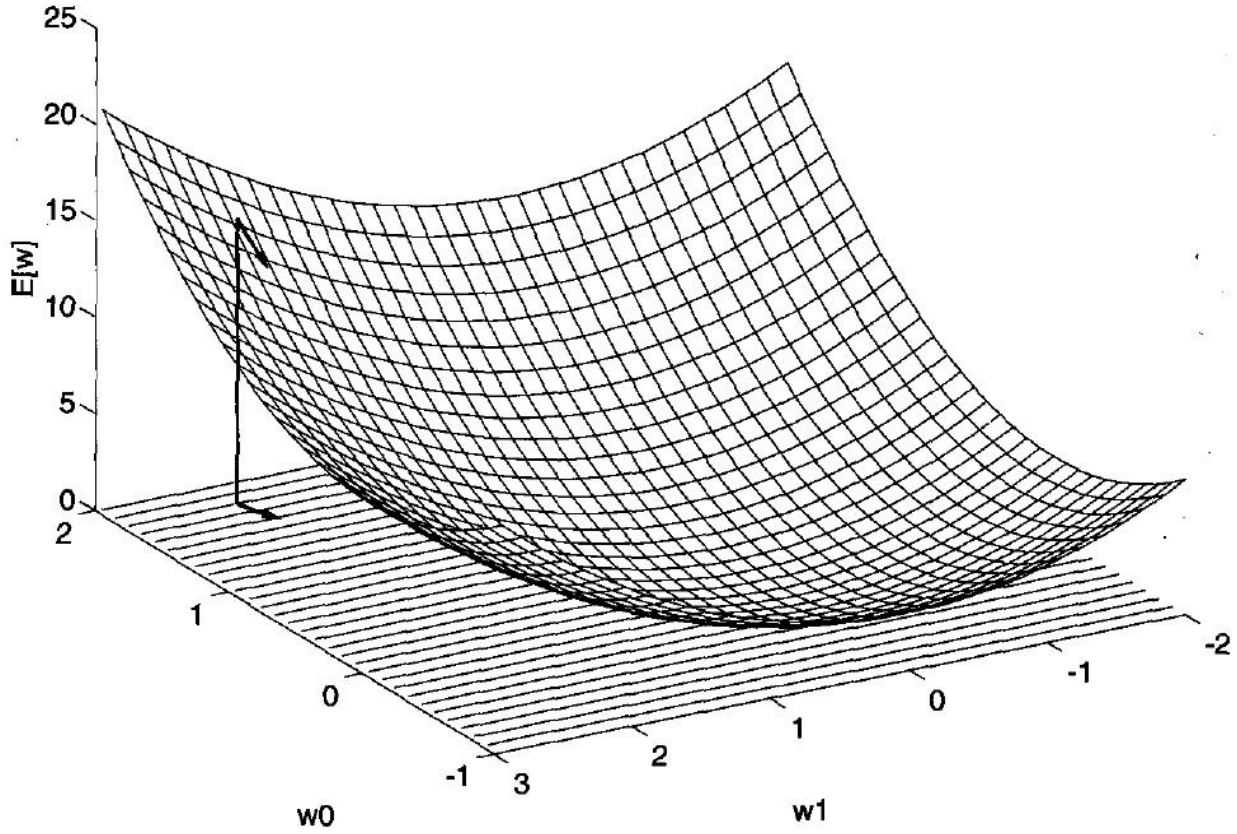
$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d)$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



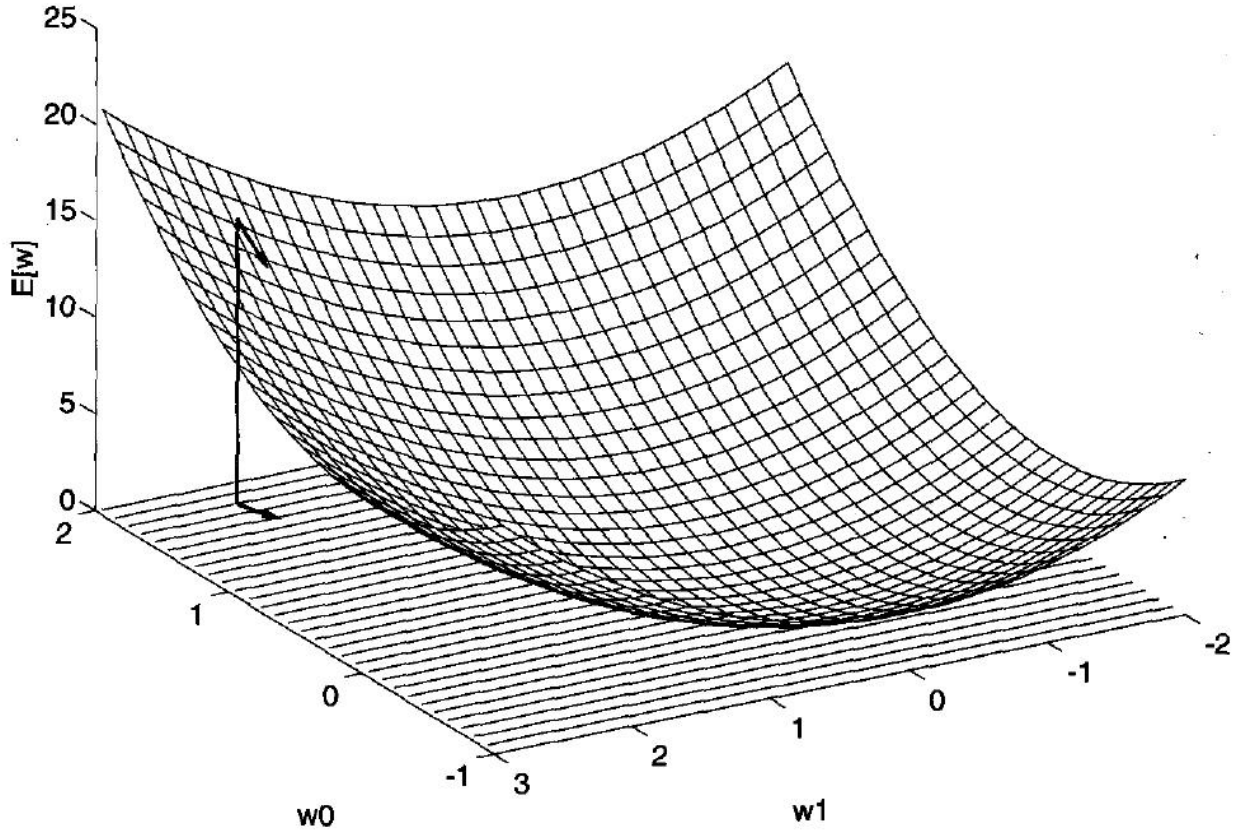
$$= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d)$$

$$= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d)$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



$$= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2$$

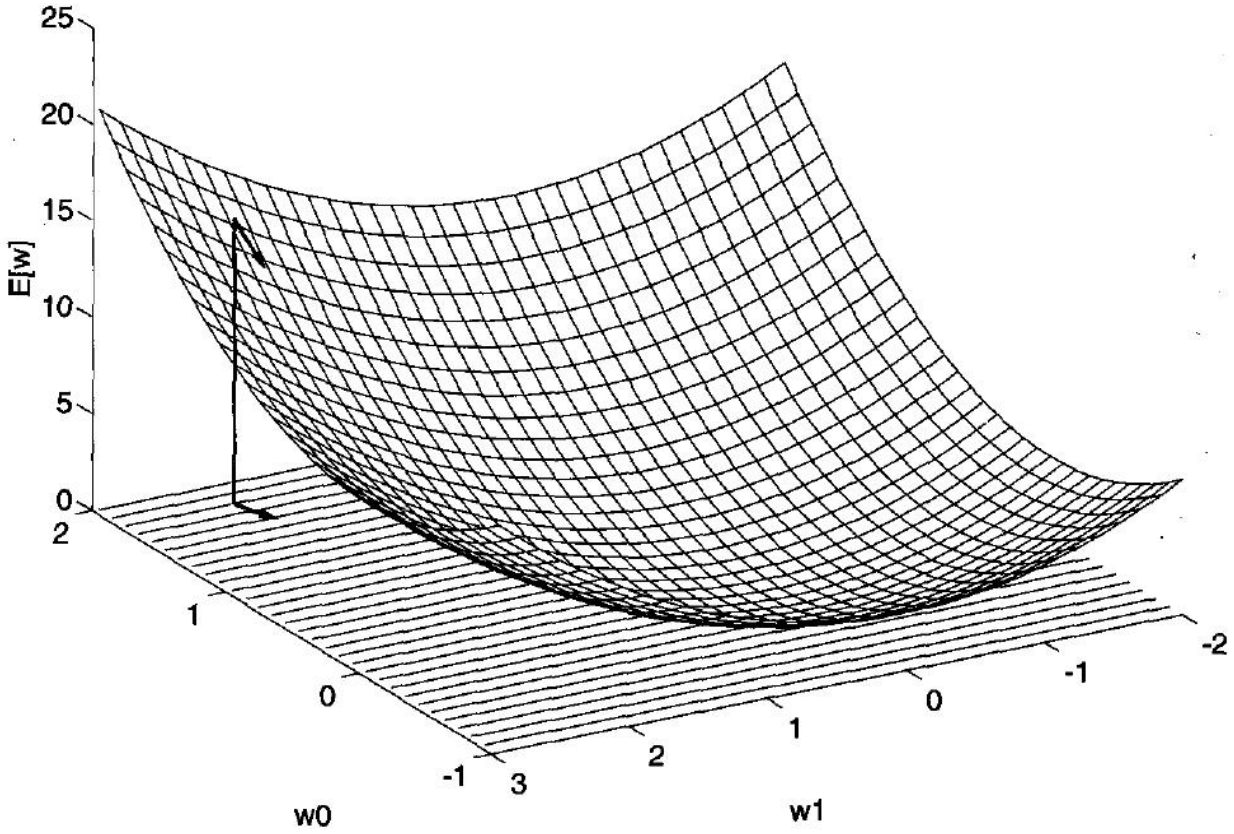
$$= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d)$$

$$= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x_d})$$

$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d)(-x_{id})$$

Artificial Neuron (The Delta Rule)

- Visualizing the gradient descendent method



Thus:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d)\, x_{id}$$

Exercises

Exercise 1

- Review the script in which a perceptron implementation is shown.

Exercise 2

- Review the script using a perceptron implementation based on **sklearn** API

Multilayer Neural Networks

Multilayer networks

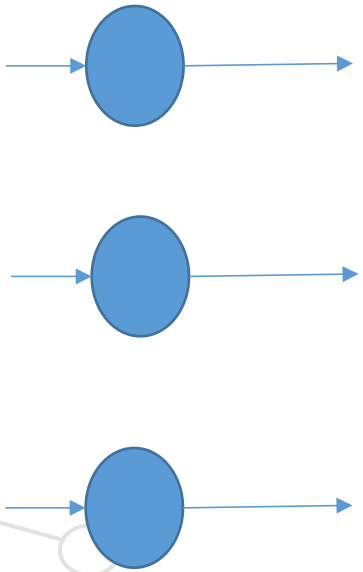
- Assume a problem that requires to learn about a set of patterns classified into **three classes**. How to solve this learning problem through artificial neurons?

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2

Multilayer networks

- Assume a problem that requires to learn about a set of patterns classified into **three classes**. How to solve this learning problem through artificial neurons?

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2

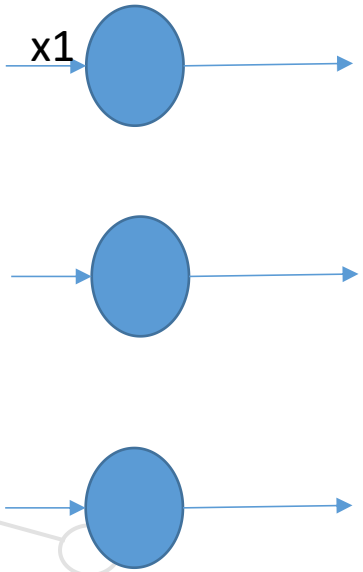


For each feature, a neuron is defined.

Multilayer networks

- Assume a problem that requires to learn about a set of patterns classified into **three classes**. How to solve this learning problem through artificial neurons?

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2

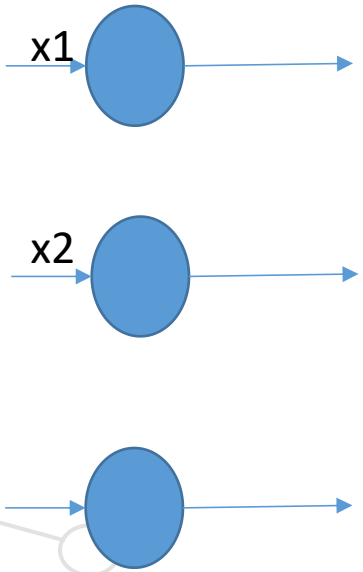


For each feature, a neuron is defined.

Multilayer networks

- Assume a problem that requires to learn about a set of patterns classified into **three classes**. How to solve this learning problem through artificial neurons?

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2

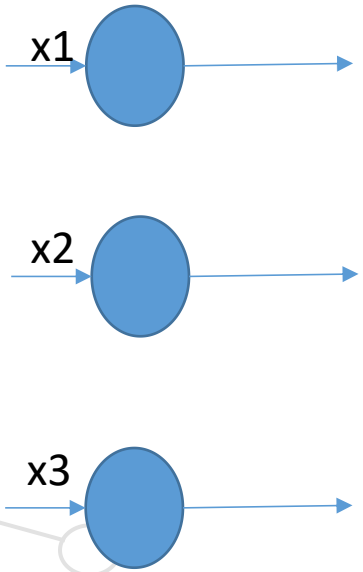


For each feature, a neuron is defined.

Multilayer networks

- Assume a problem that requires to learn about a set of patterns classified into **three classes**. How to solve this learning problem through artificial neurons?

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



For each feature, a neuron is defined.

Multilayer networks

- Assume a problem that requires to learn about a set of patterns classified into **three classes**. How to solve this learning problem through artificial neurons?

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2

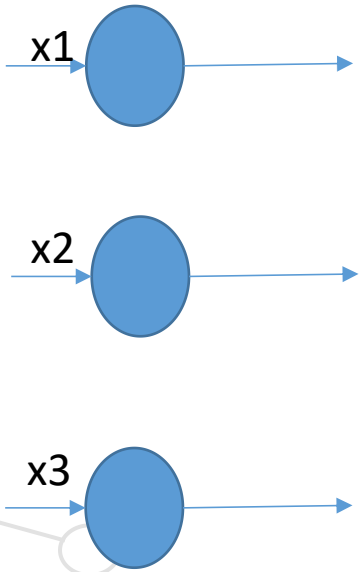
A set of neurons that encodes the class label of the input pattern is created.

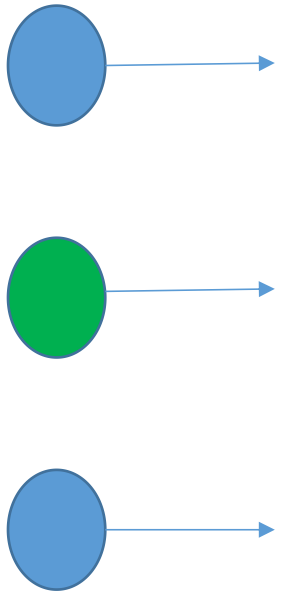
For each feature, a neuron is defined.

Multilayer networks

- Assume a problem that requires to learn about a set of patterns classified into **three classes**. How to solve this learning problem through artificial neurons?

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2





A set of neurons that represent the class label of the input pattern is created.

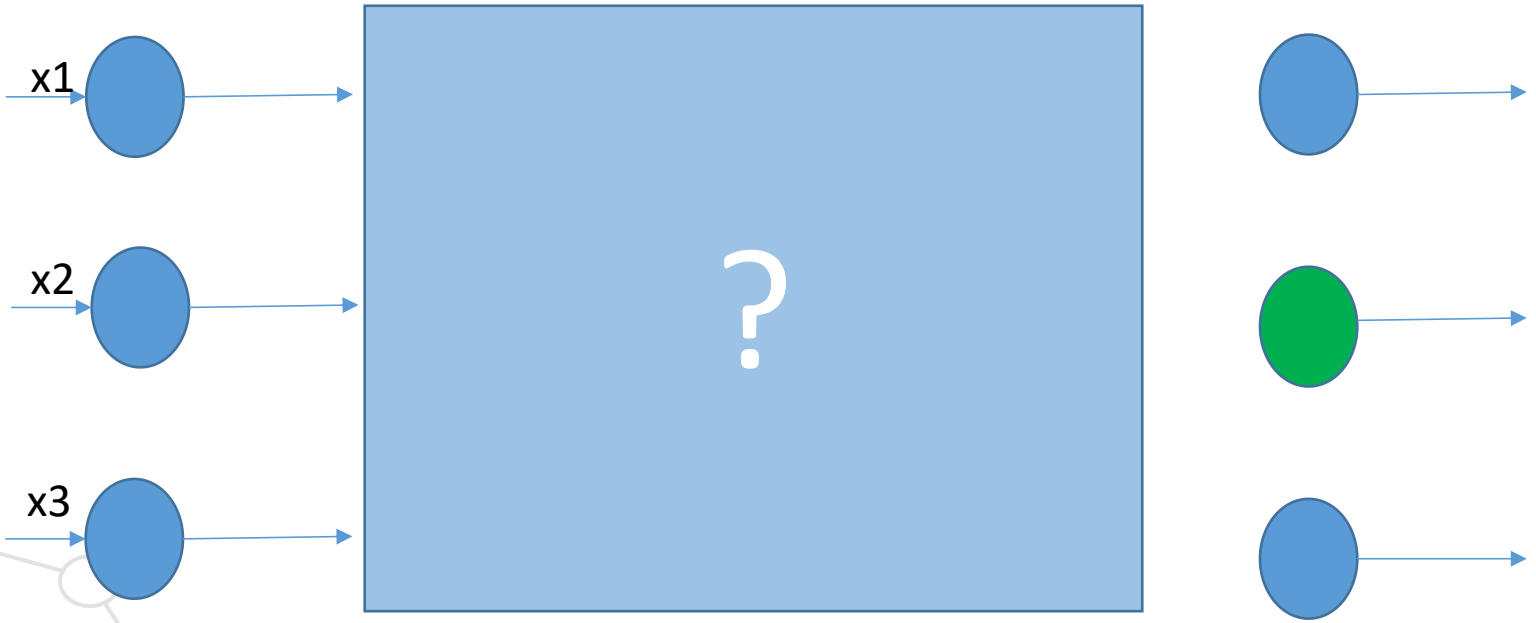
Note that only one neuron will be activated

For each feature, a neuron is defined.

Multilayer networks

- The mystery box represent a set of neurons and interconnections that propagate the signals from input neurons to output neurons.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



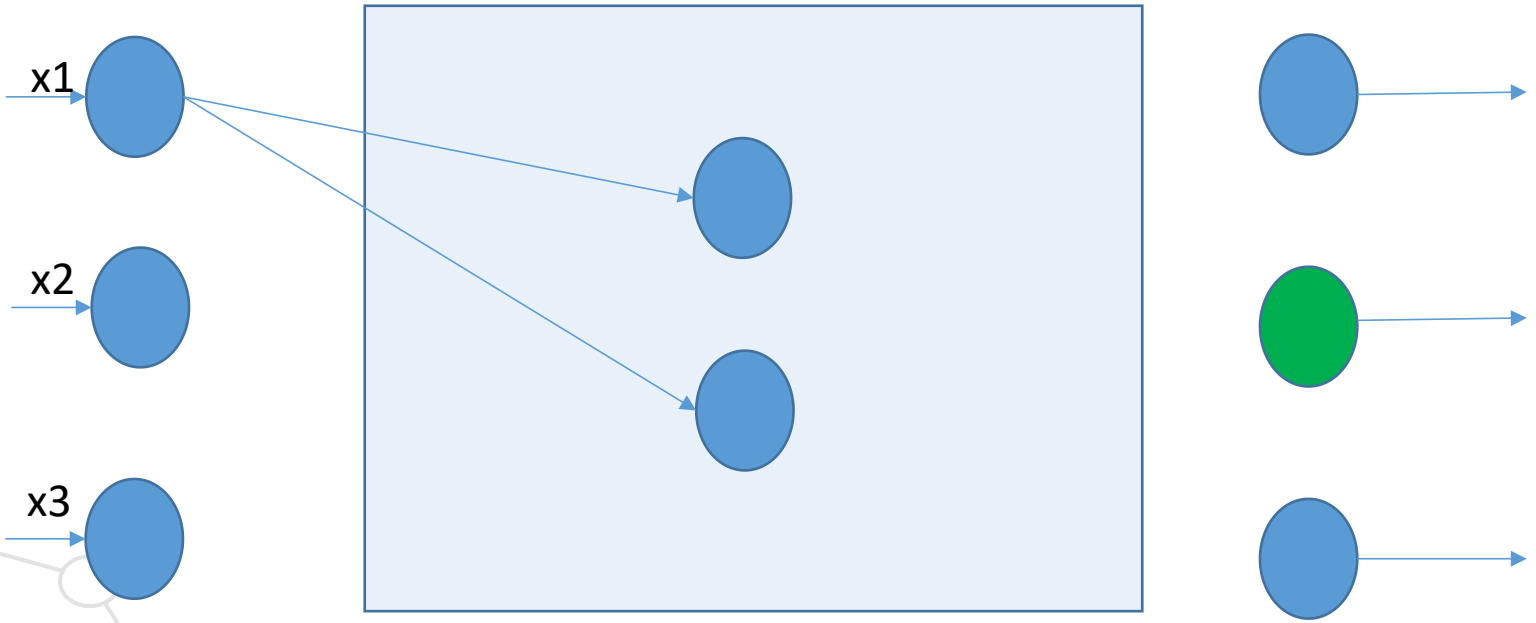
Input Neurons

Output Neurons

Multilayer networks

- The mystery box represent a set of neurons and interconnections that propagate the signals from input neurons to output neurons.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



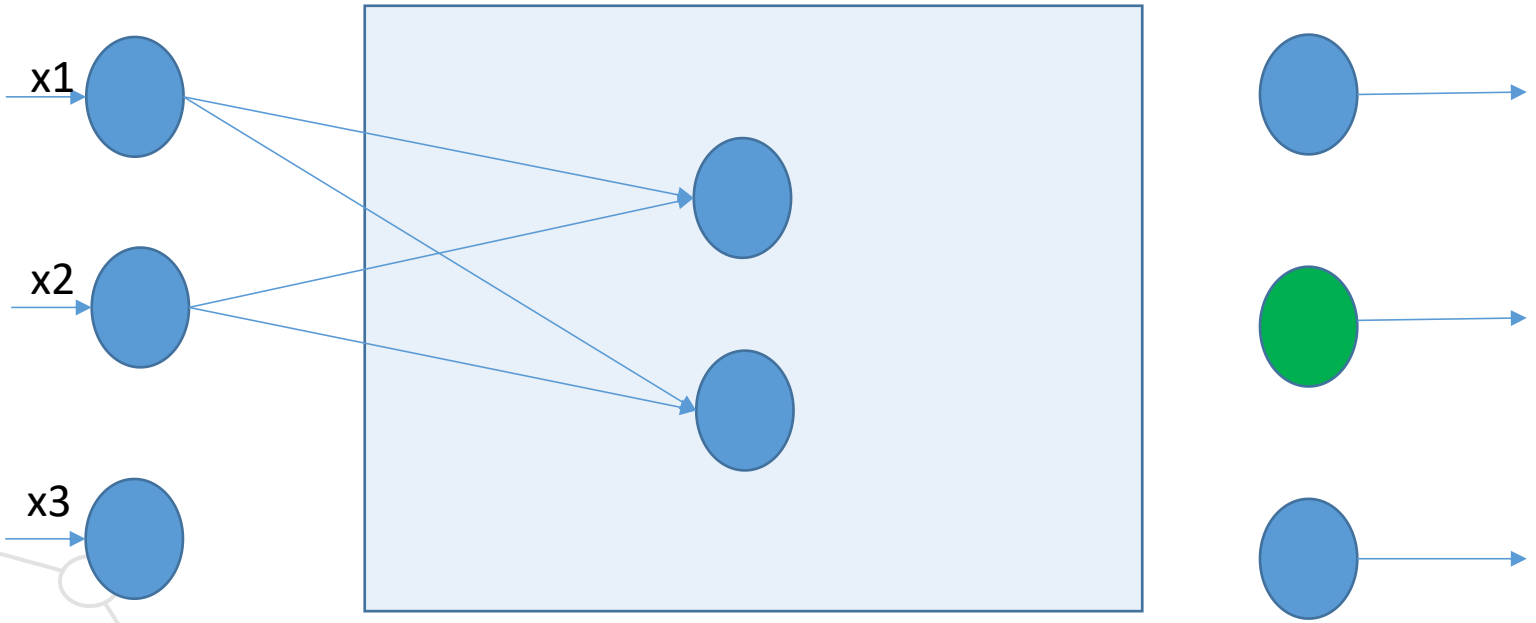
Input Neurons

Output Neurons

Multilayer networks

- The mystery box represent a set of neurons and interconnections that propagate the signals from input neurons to output neurons.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



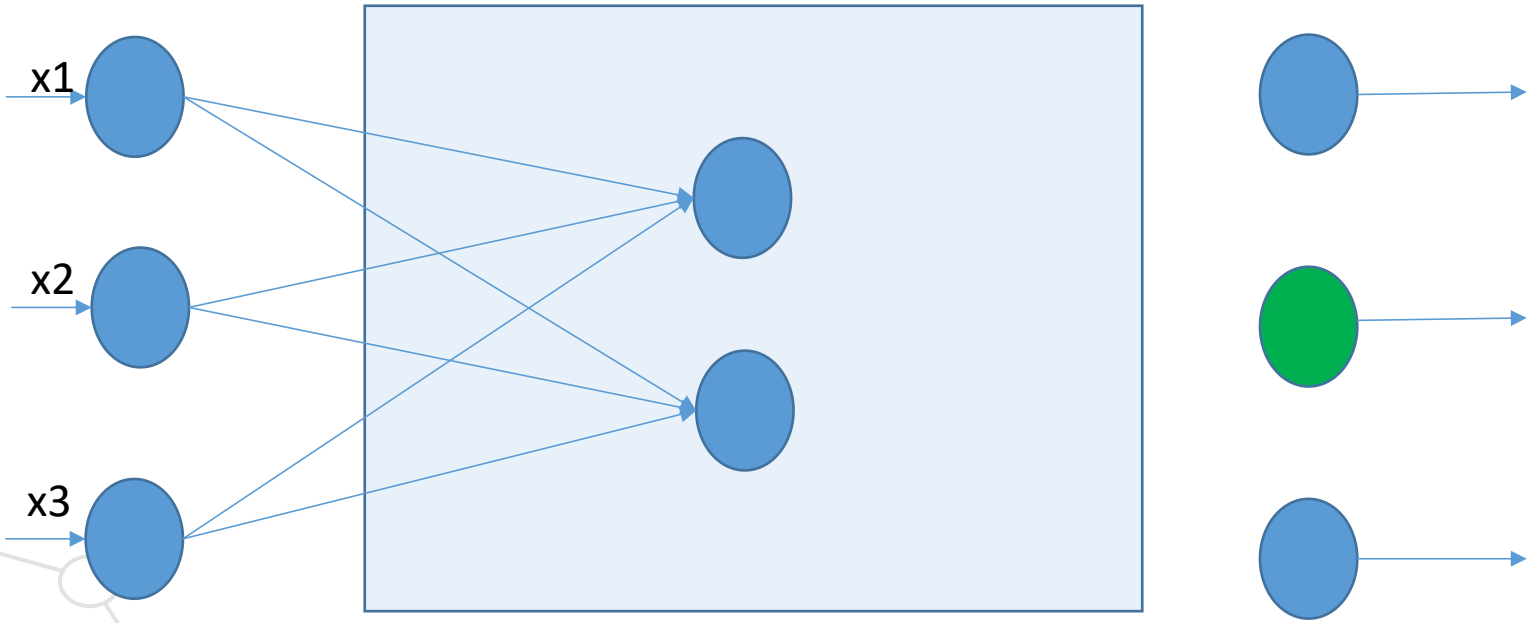
Input Neurons

Output Neurons

Multilayer networks

- The mystery box represent a set of neurons and interconnections that propagate the signals from input neurons to output neurons.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



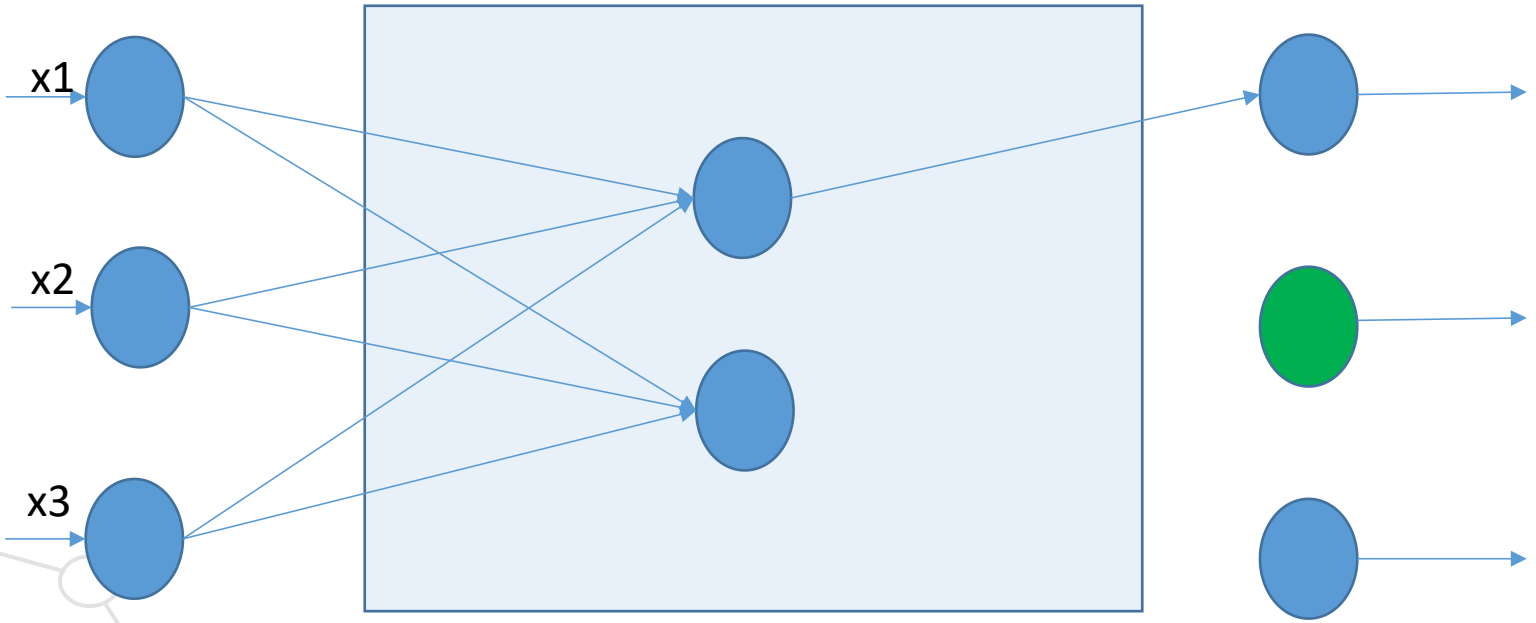
Input Neurons

Output Neurons

Multilayer networks

- The mystery box represent a set of neurons and interconnections that propagate the signals from input neurons to output neurons.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



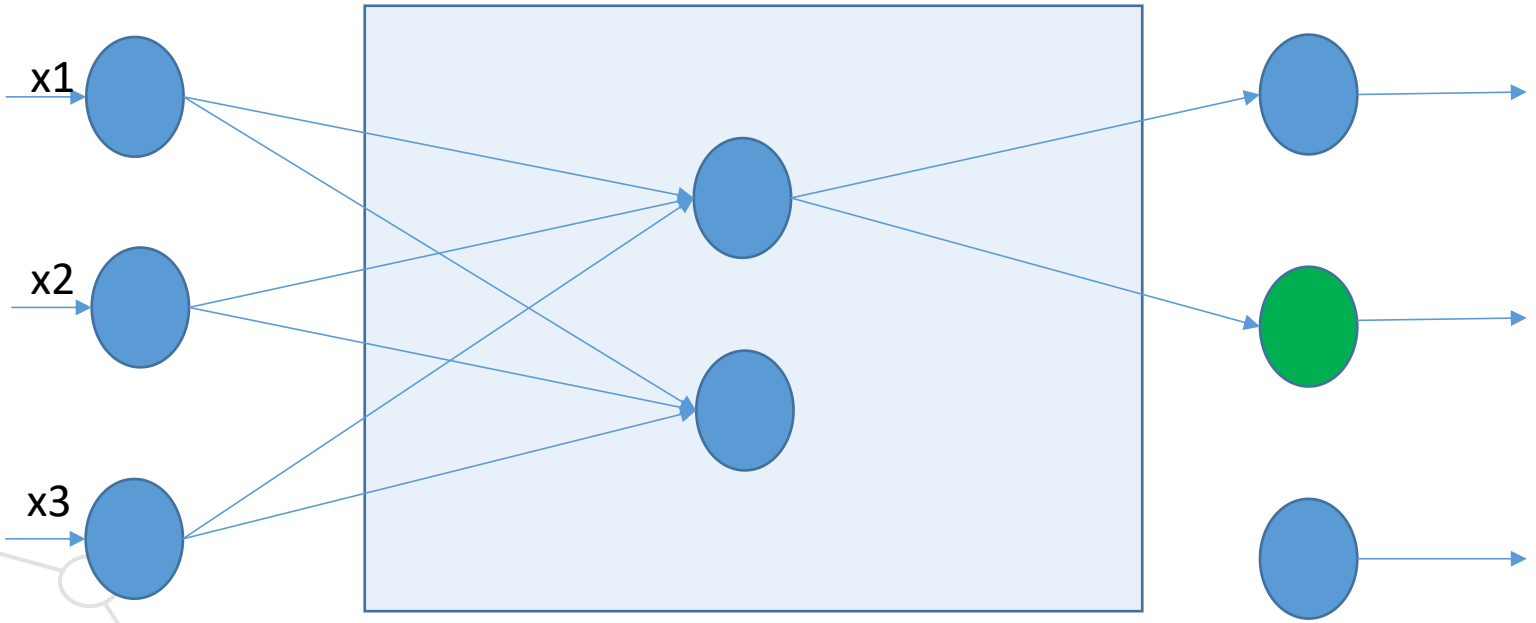
Input Neurons

Output Neurons

Multilayer networks

- The mystery box represent a set of neurons and interconnections that propagate the signals from input neurons to output neurons.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



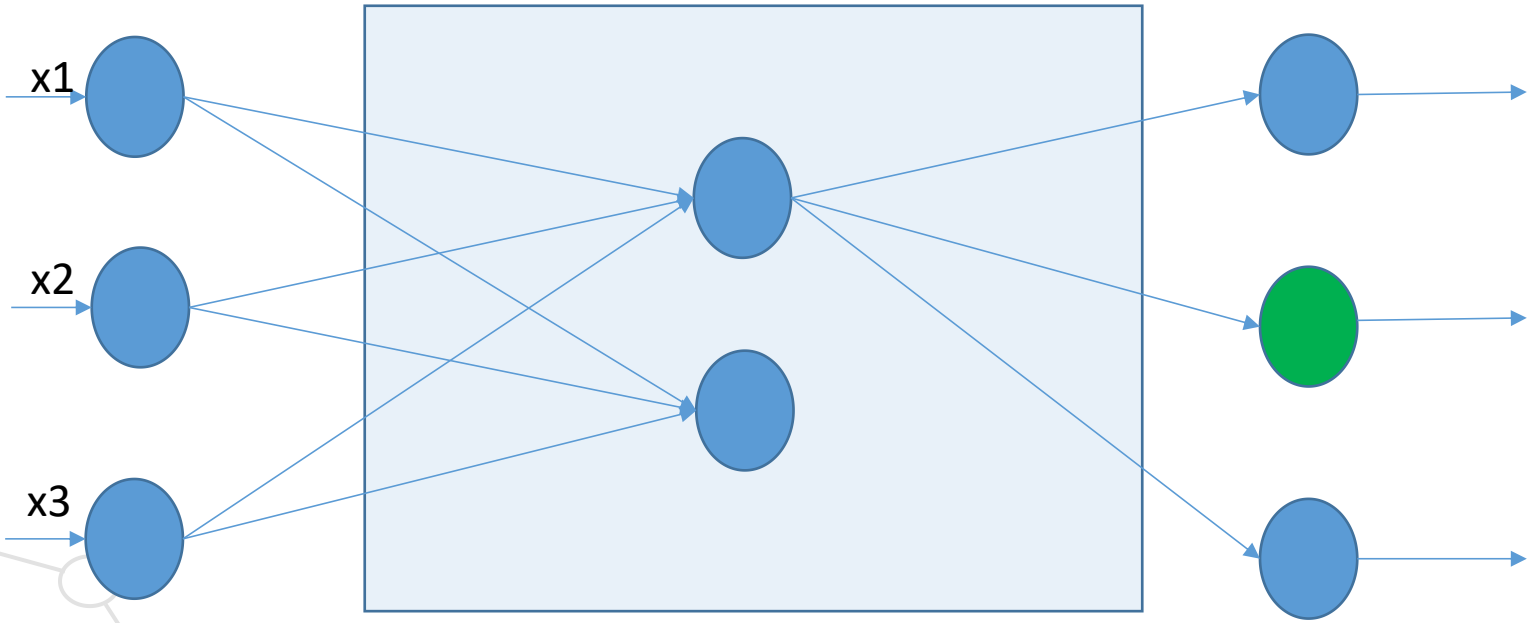
Input Neurons

Output Neurons

Multilayer networks

- The mystery box represent a set of neurons and interconnections that propagate the signals from input neurons to output neurons.

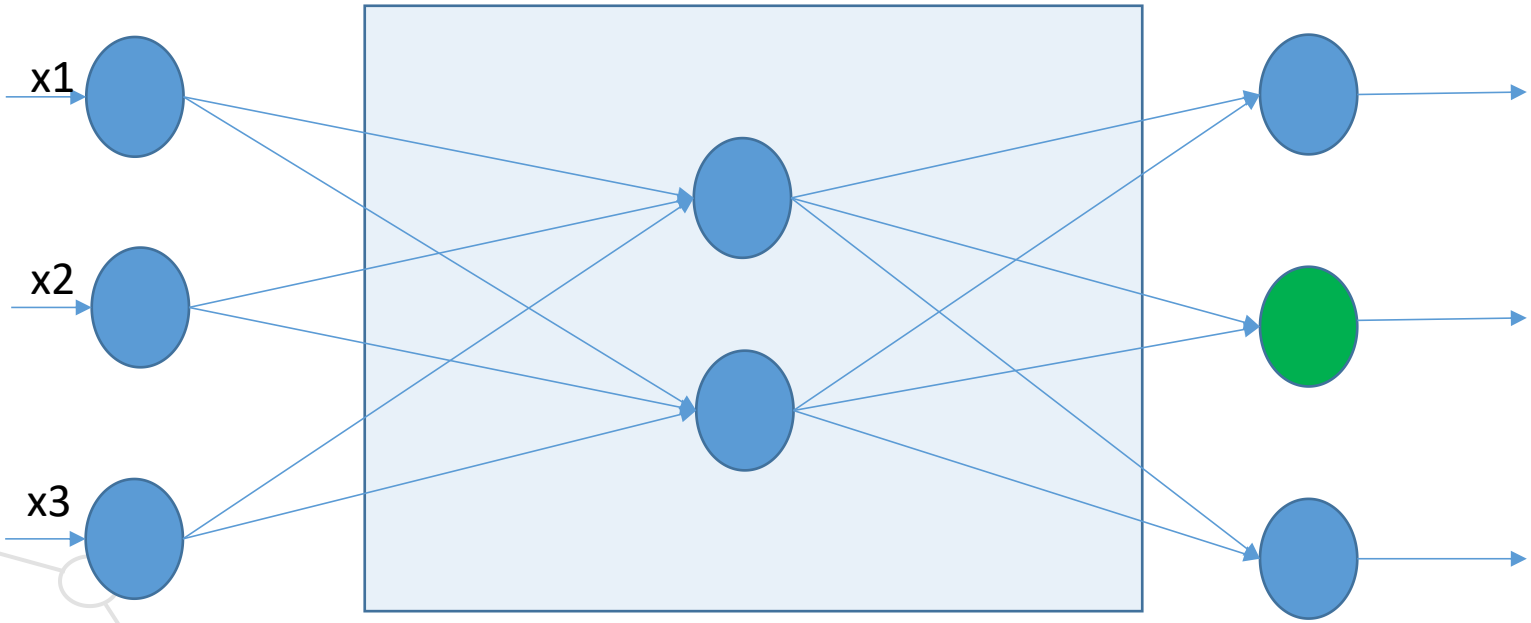
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks

- The mystery box represent a set of neurons and interconnections that propagate the signals from input neurons to output neurons.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



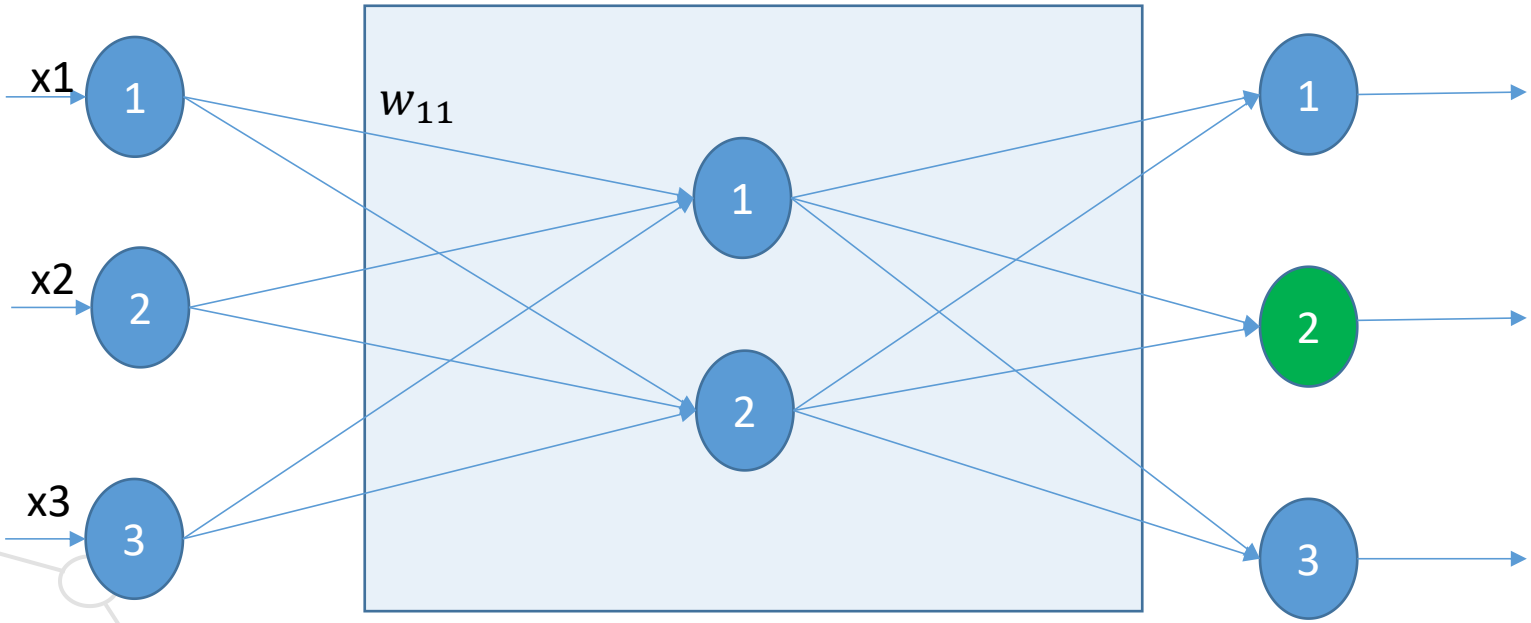
Input Neurons

Output Neurons

Multilayer networks

- A set of weights must be defined.

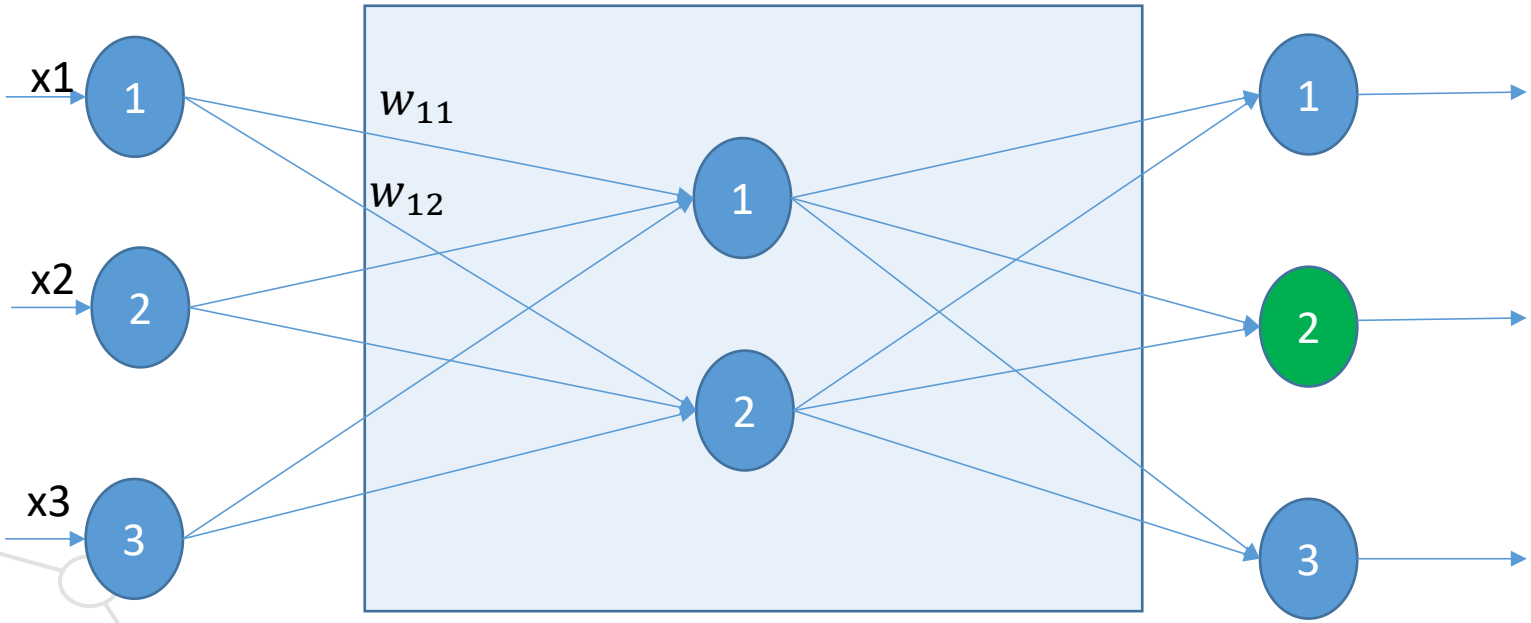
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks

- A set of weights must be defined.

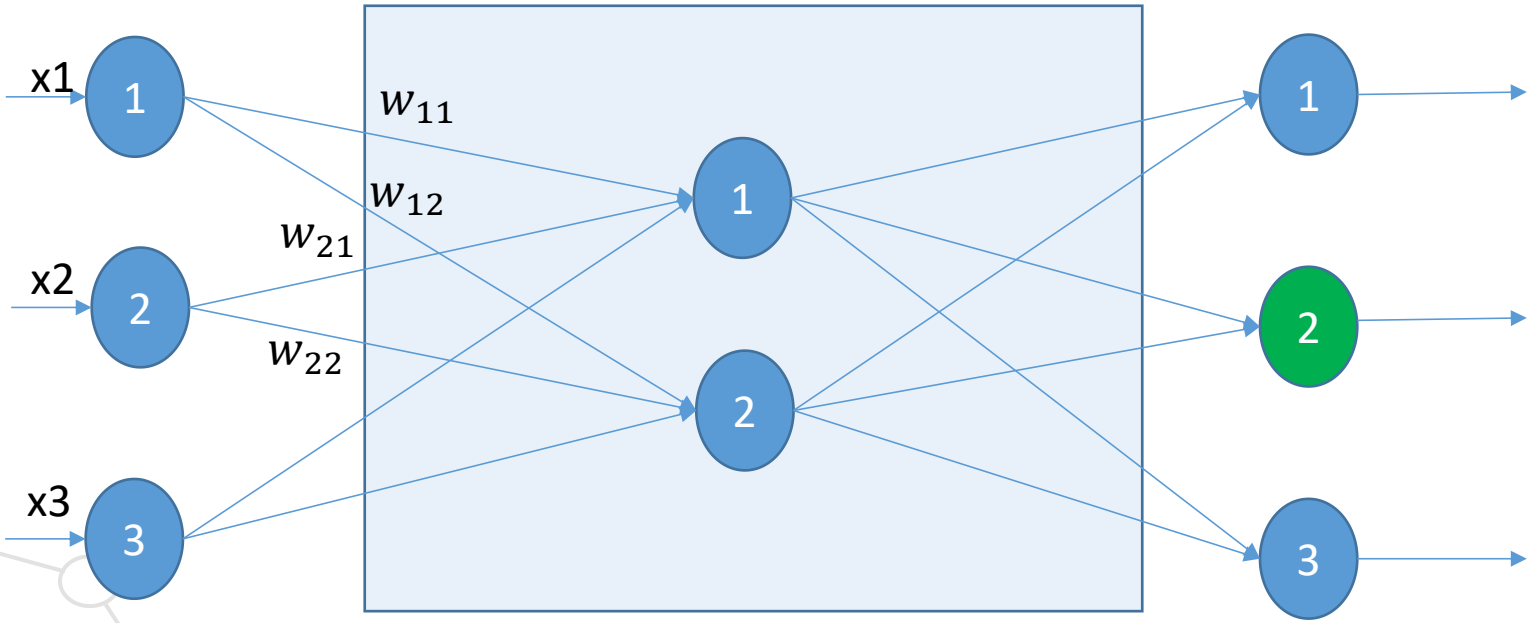
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks

- A set of weights must be defined.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



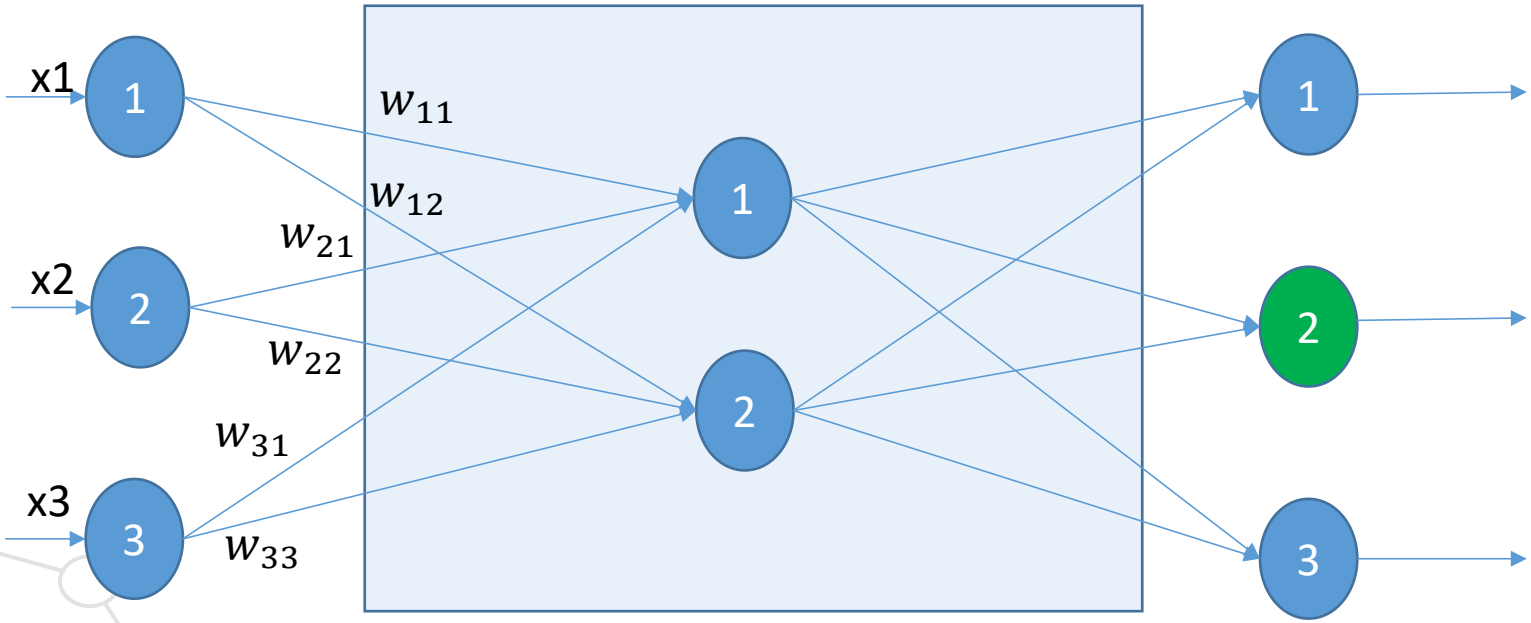
Input Neurons

Output Neurons

Multilayer networks

- A set of weights must be defined.

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



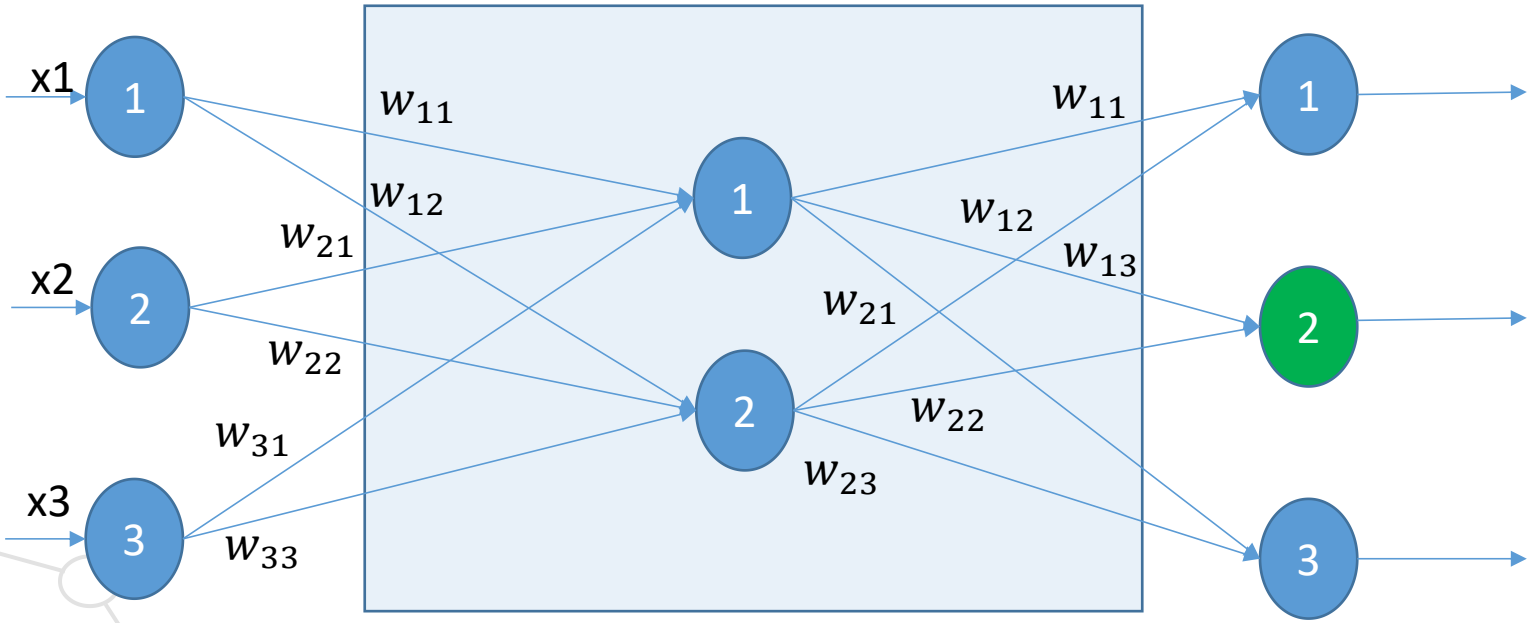
Input Neurons

Output Neurons

Multilayer networks

- A set of weights must be defined.

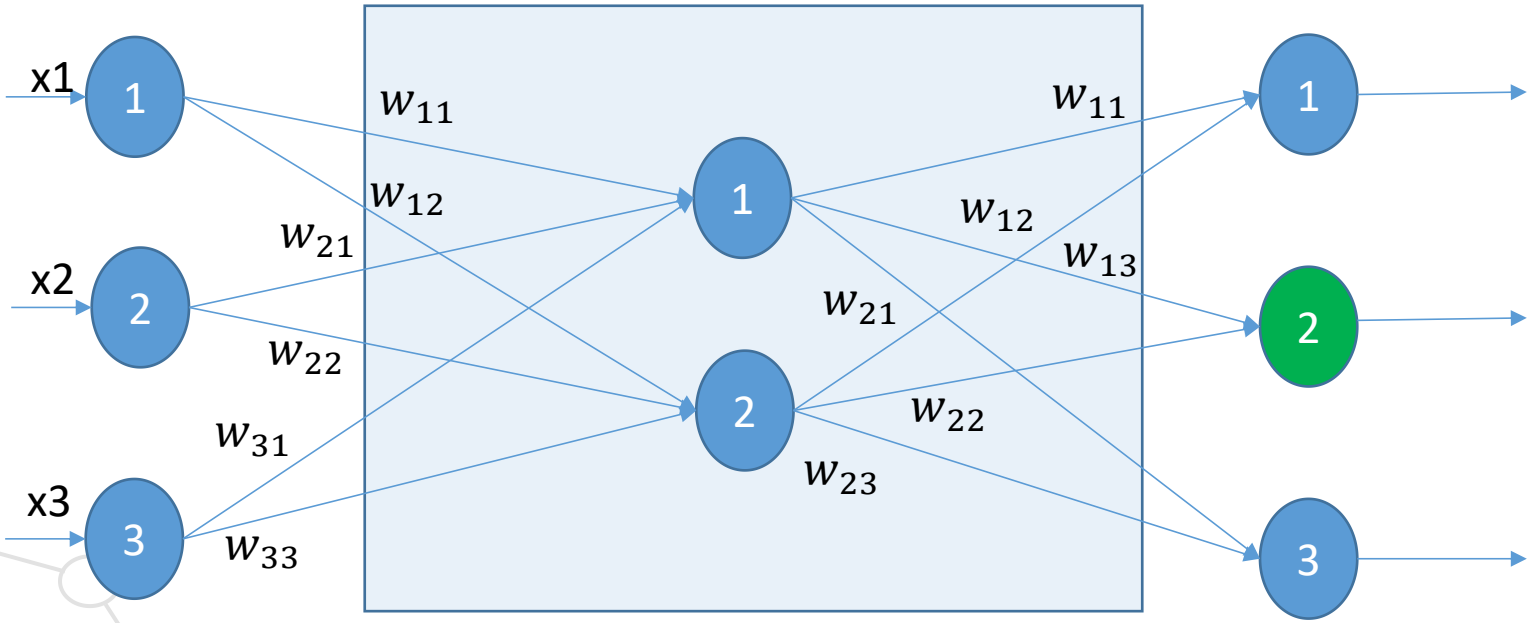
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks

- A set of weights must be defined.
- Like perceptron, for the output of each neuron, an **activation function** is applied.

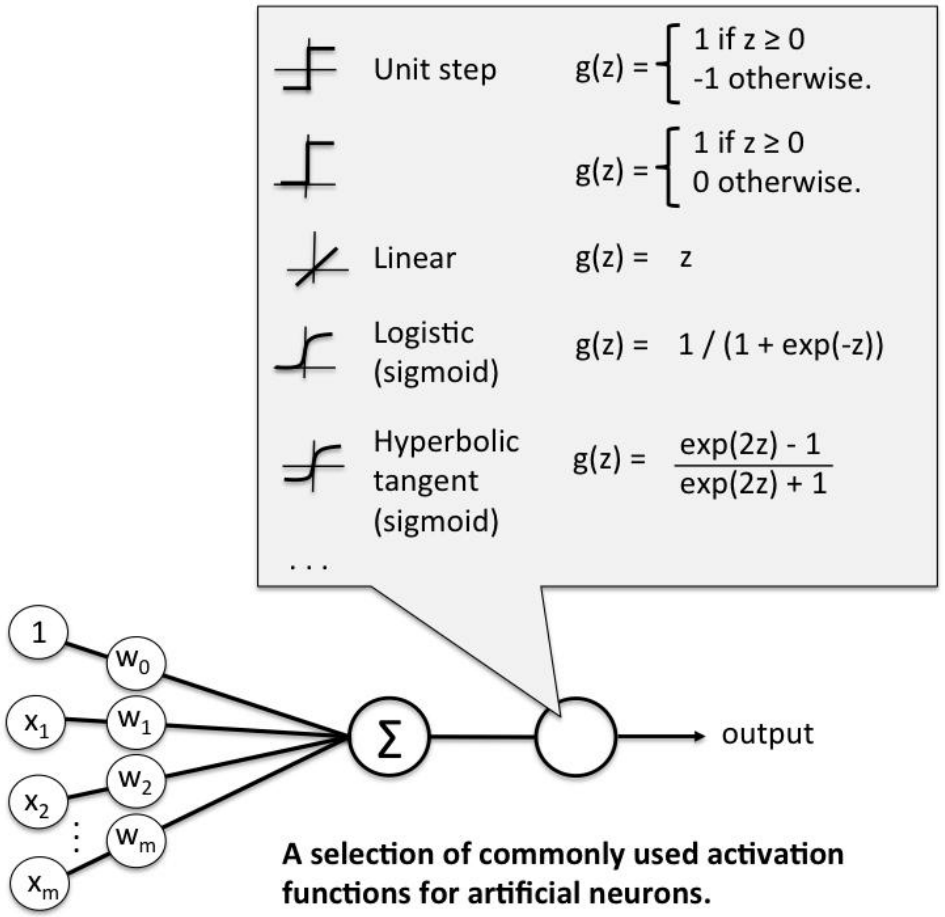
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Input Neurons

Output Neurons

Activation Function



A selection of commonly used activation functions for artificial neurons.

Typically these functions are within **Hidden layers of a Neural Network Model**.

For output layers we should use a **Softmax function** for a Classification problem to compute the probabilites for the classes.

For a regression problem it should simply use a **linear function**.

Softmax Function

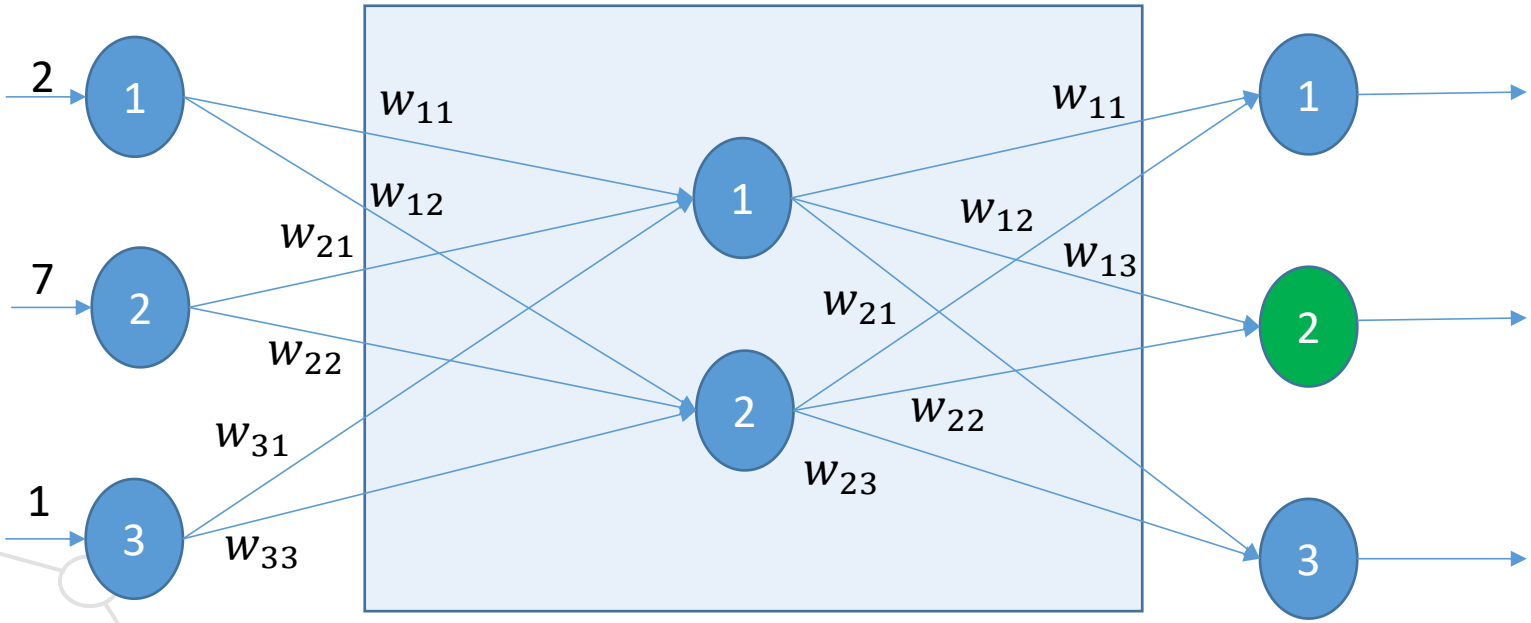
The softmax function converts its inputs, known as logit or logit scores, to be between 0 and 1, and also normalizes the outputs so that they all sum up to 1. In other words, the softmax function turns your logits into probabilities. Mathematically, the softmax function is defined as follows:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

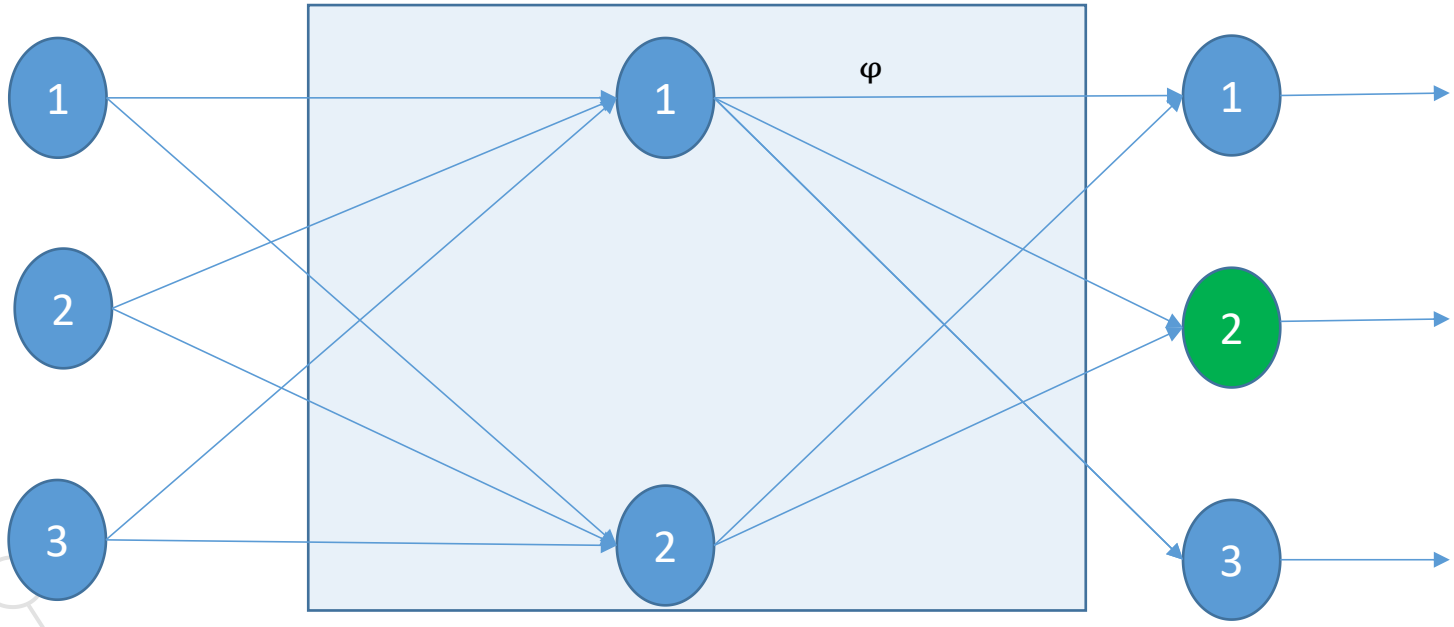
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network
- φ

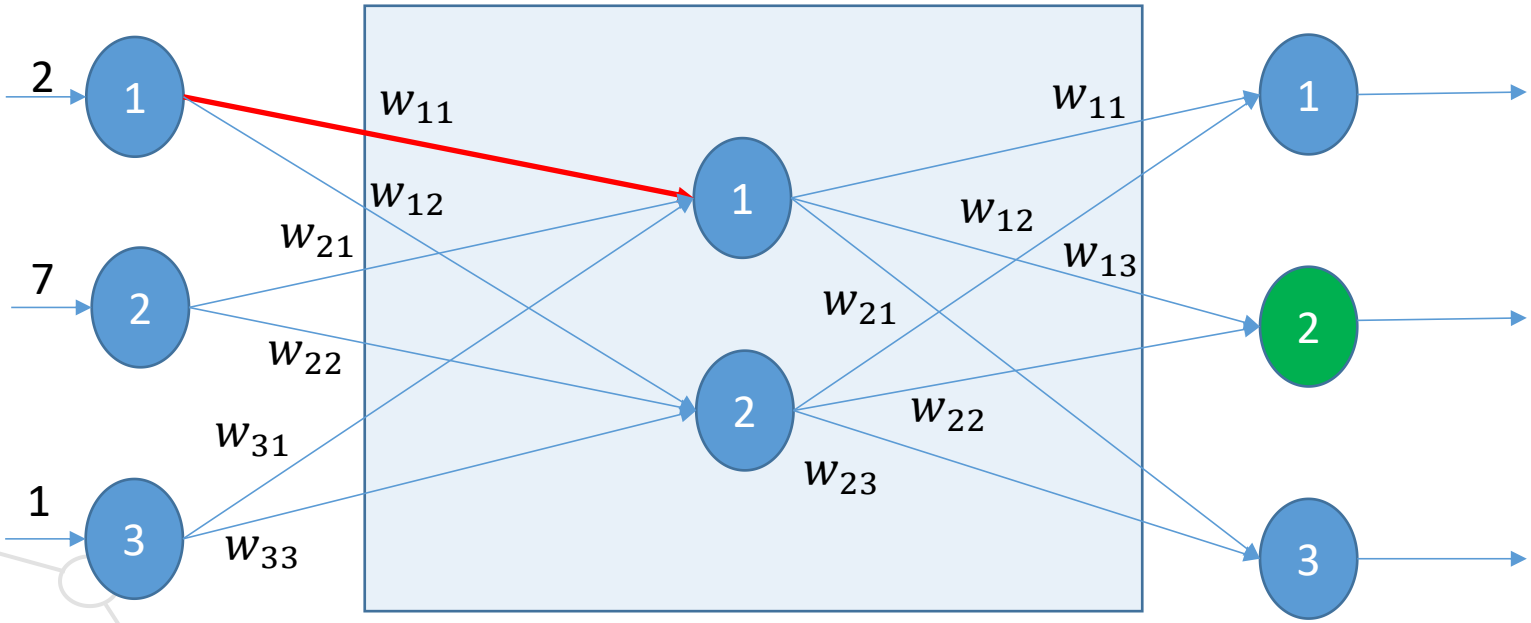
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

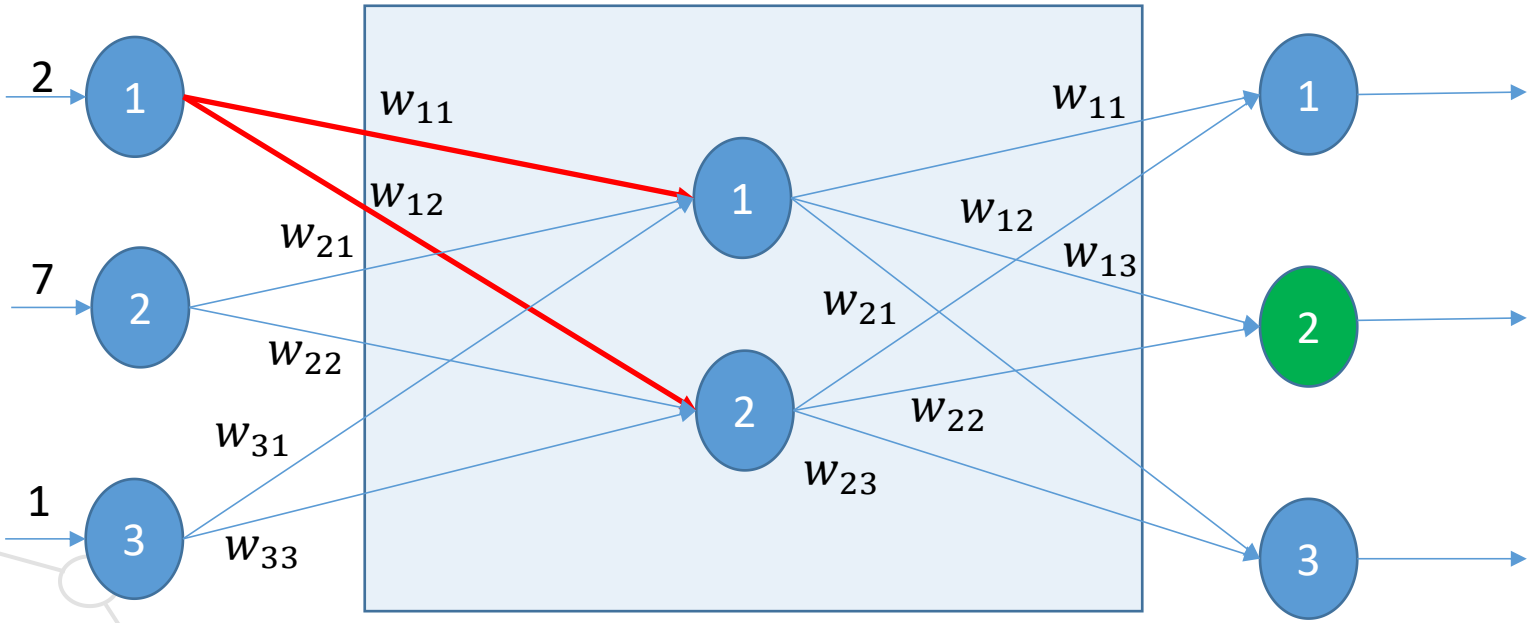
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

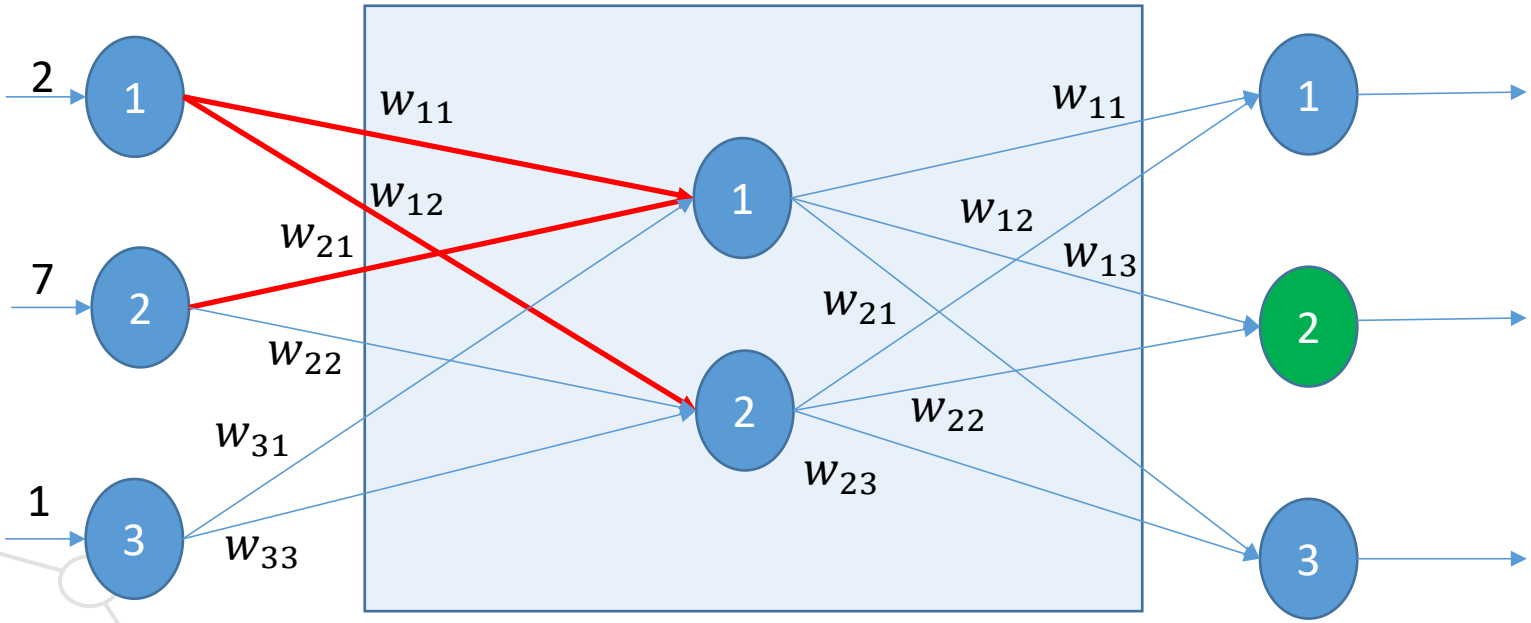
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

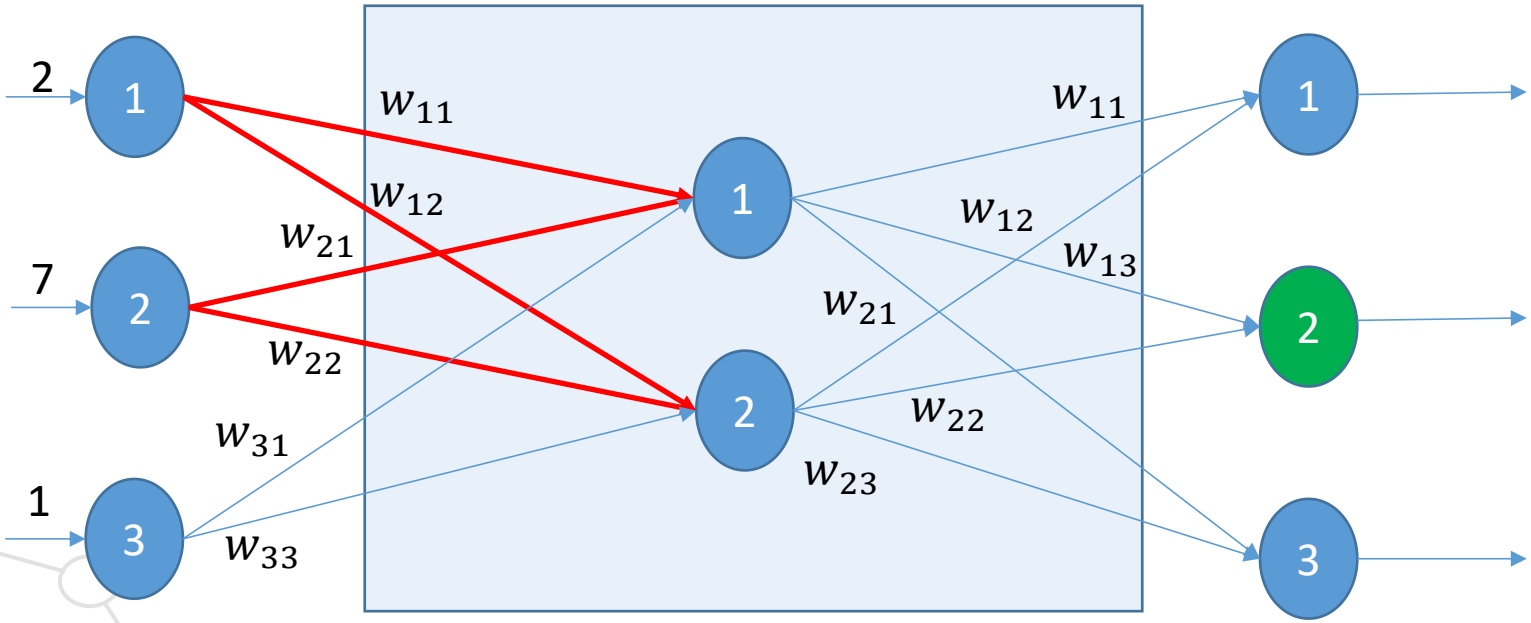
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

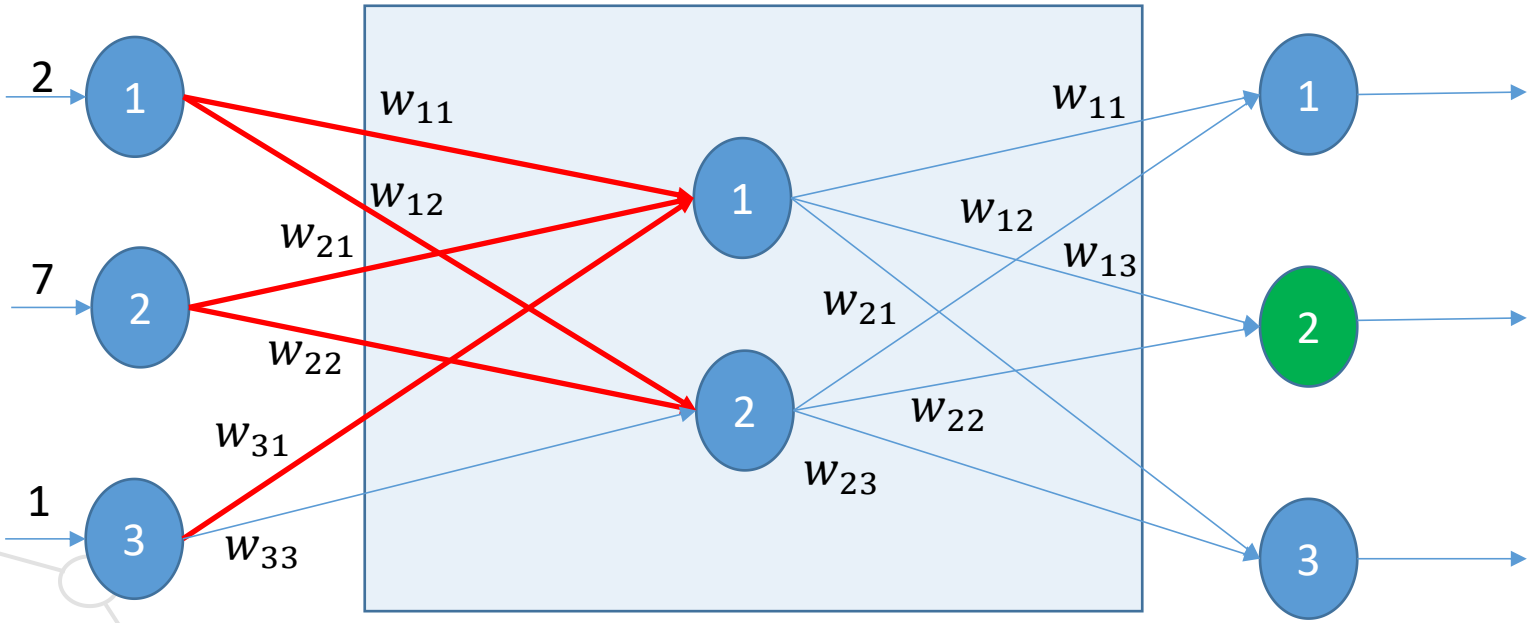
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



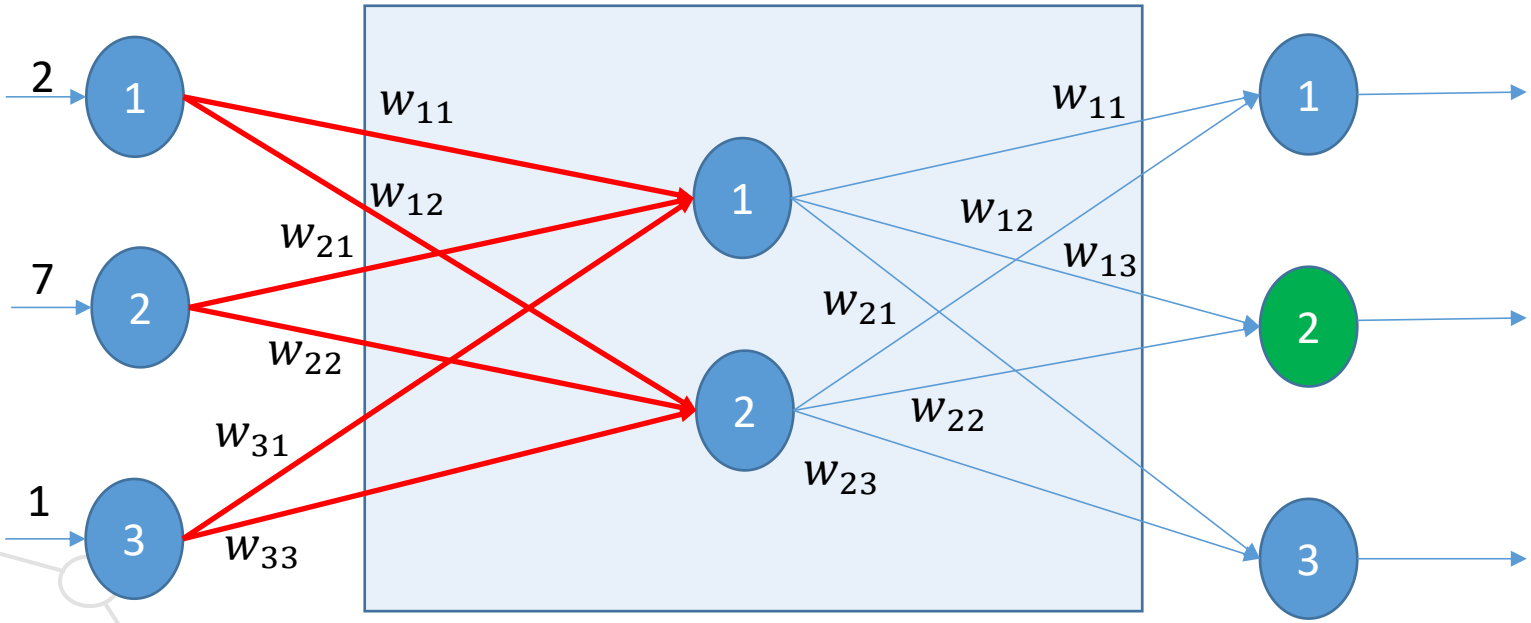
Input Neurons

Output Neurons

Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

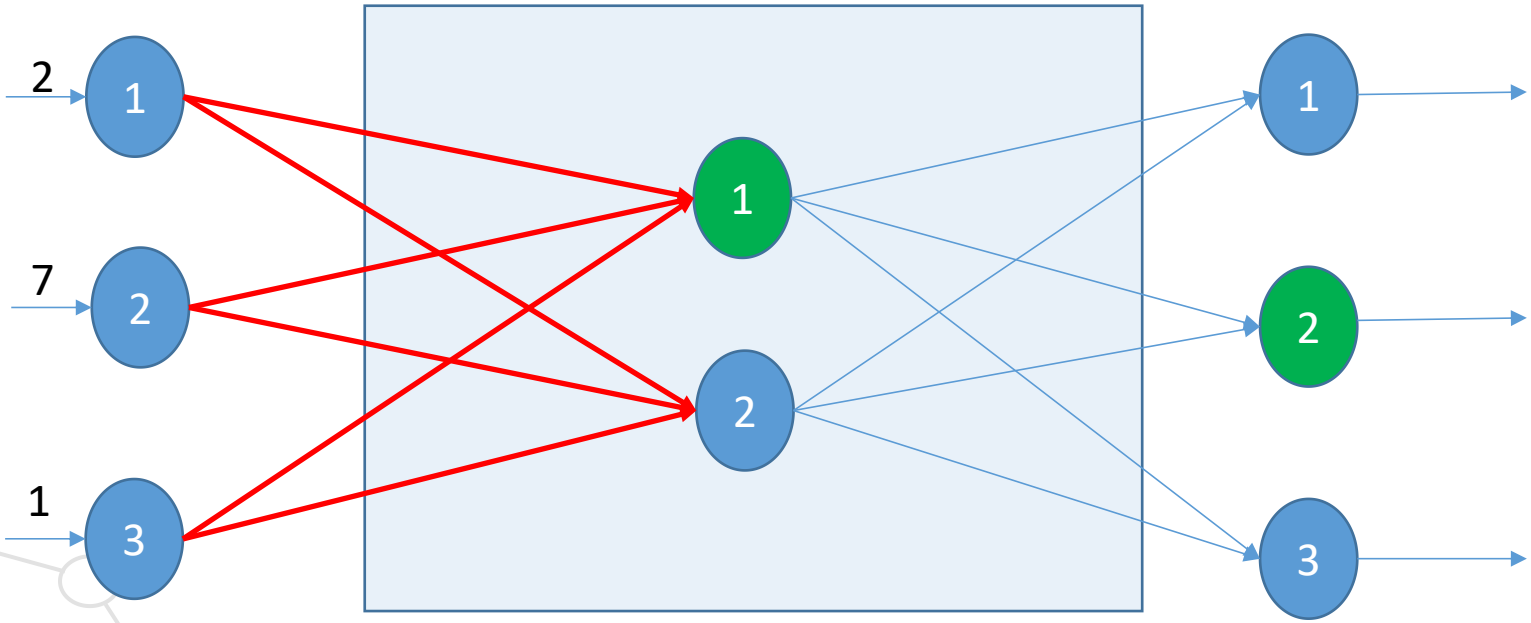
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

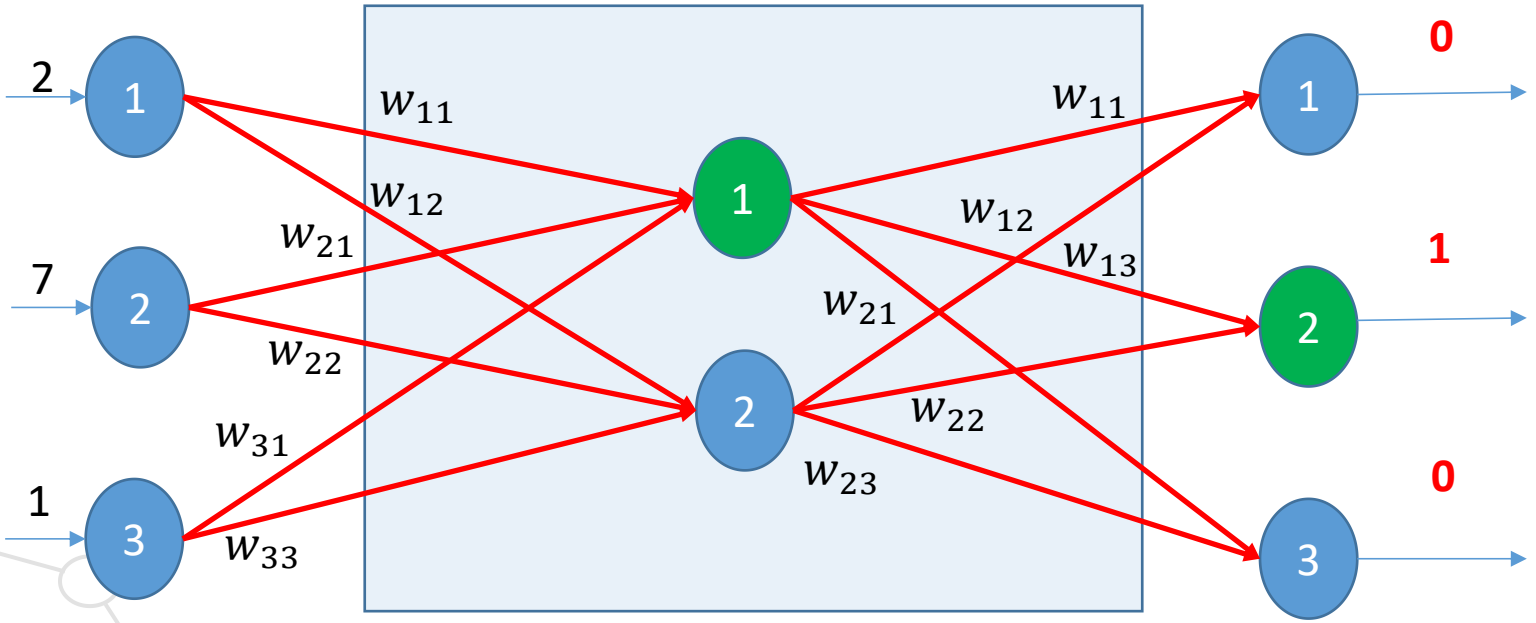
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 1: Propagate the input forward through the network

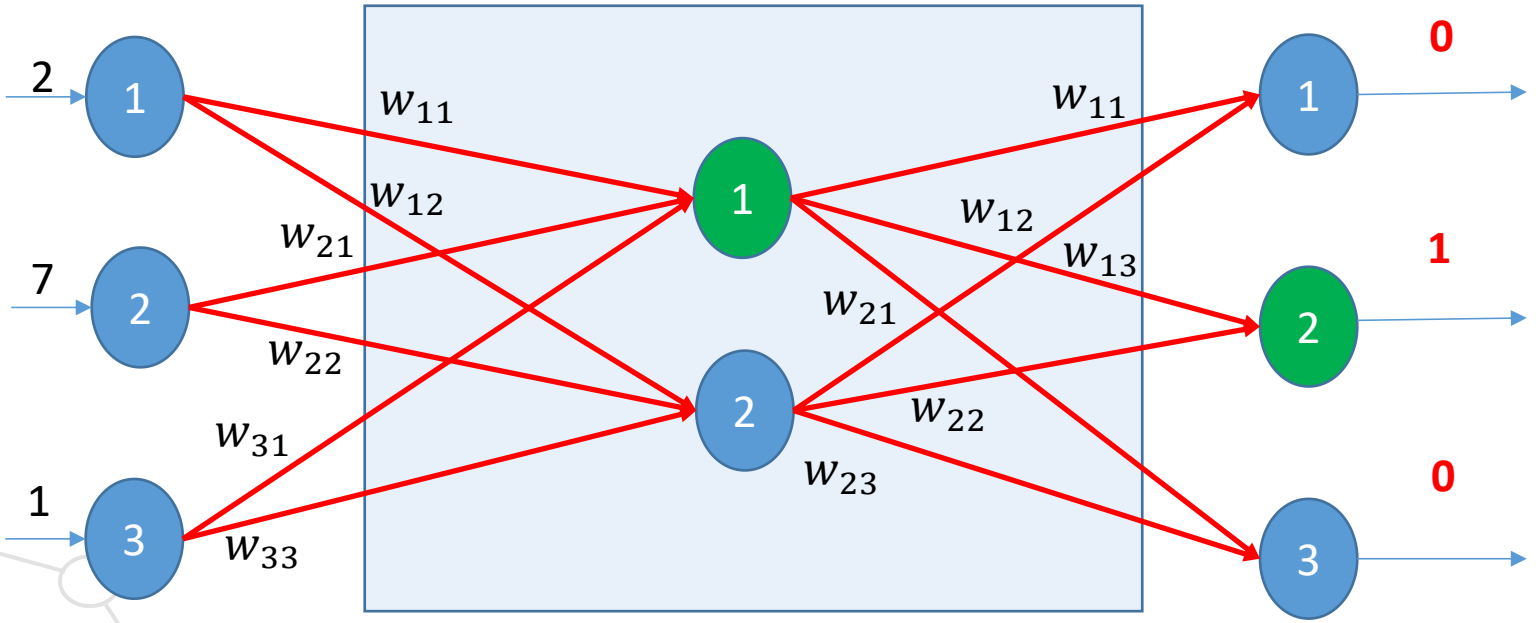
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 2: Propagate the input forward through the network

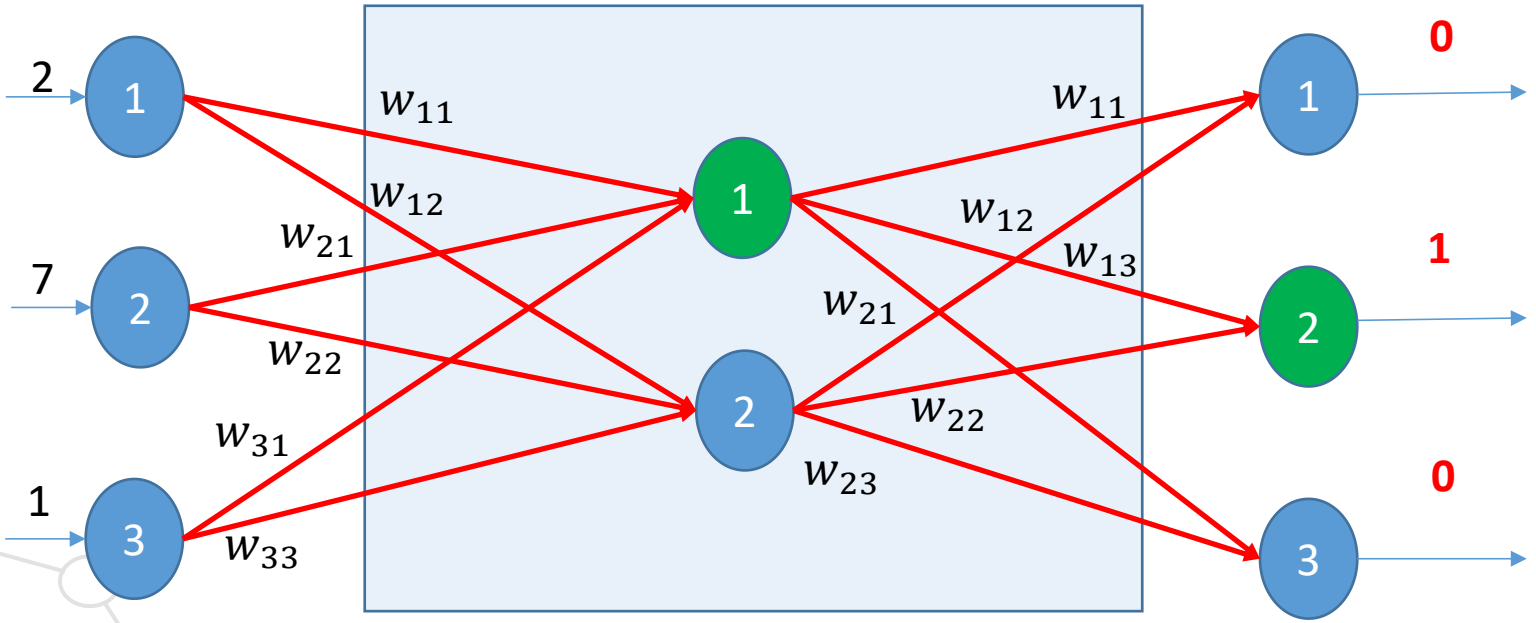
X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Multilayer networks-Learning Process

- Step 3: Learning Process (propagate the **error** backward)

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2

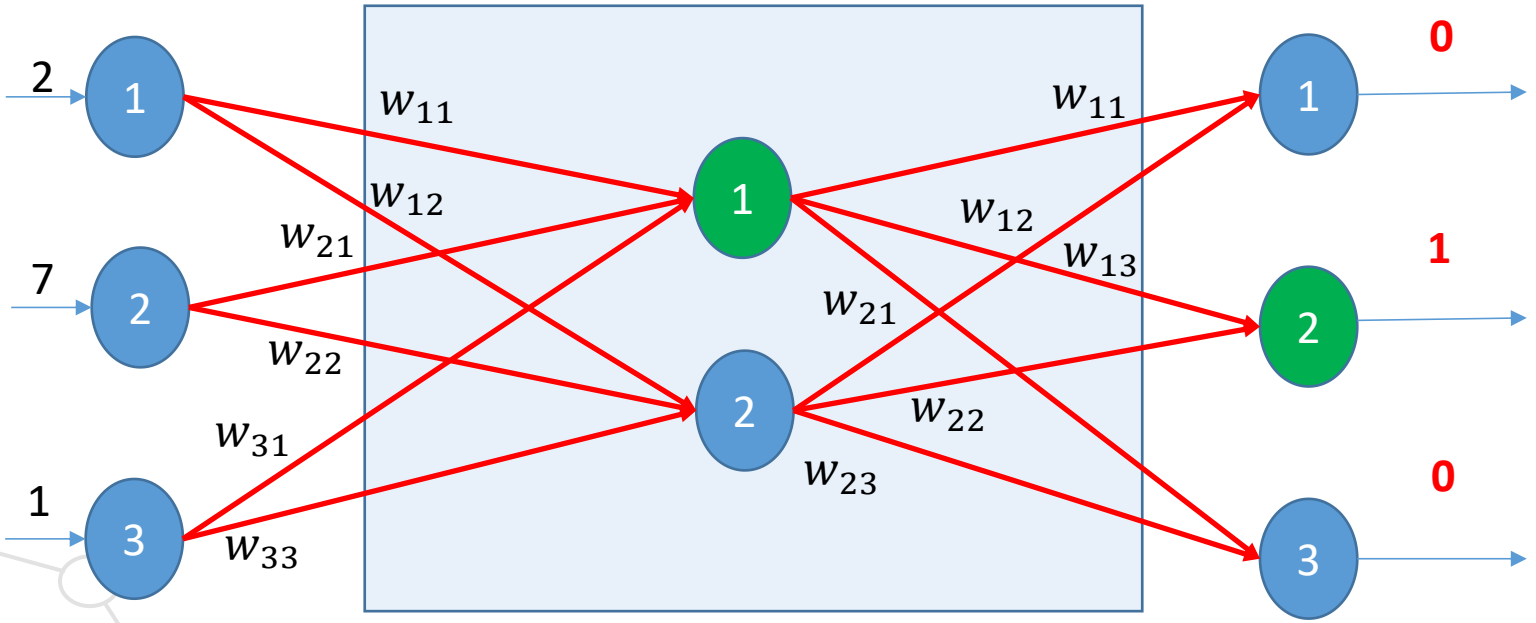


Multilayer networks-Learning Process

- Like single perceptron, we want to update the weigths in this form:

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



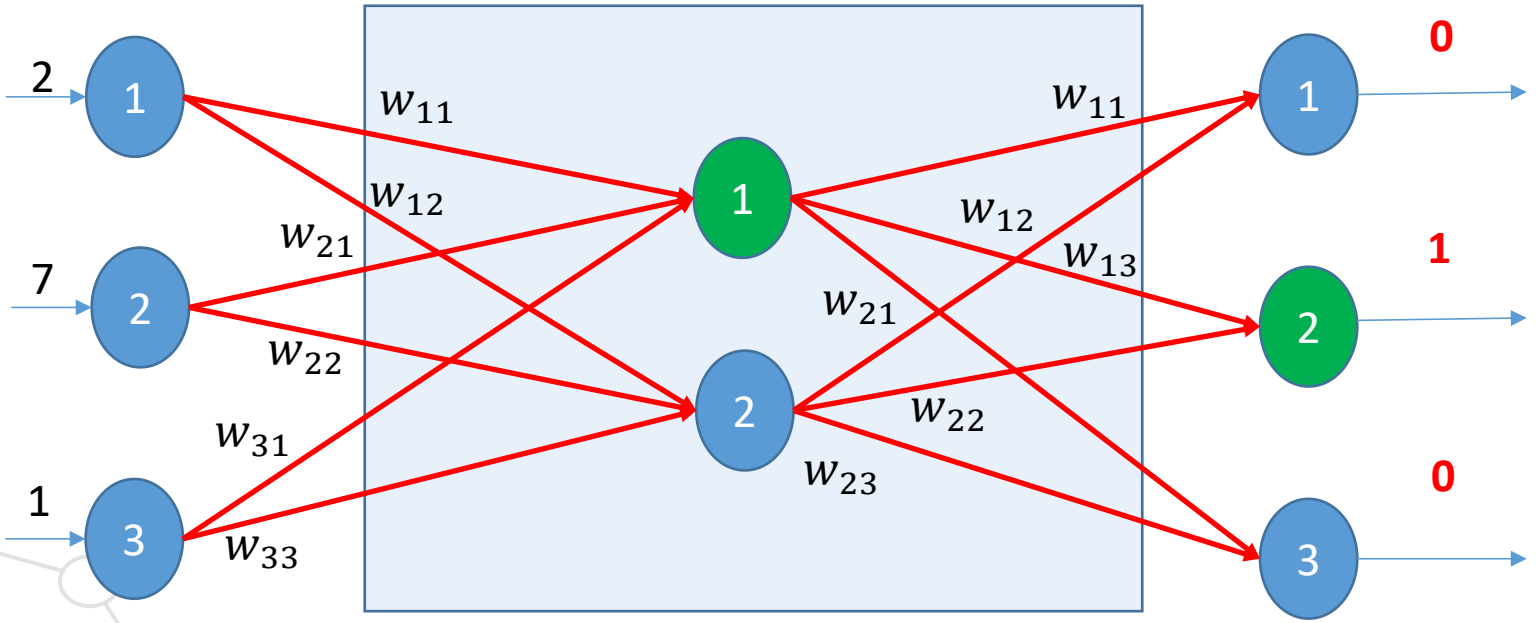
Multilayer networks-Learning Process

- Like single perceptron, we want to update the weigths in this form:

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

BACKPROPAGATION METHOD

X1	X2	X3	c
2	7	1	0
3	6	5	1
5	4	7	2
6	8	5	0
7	9	7	1
8	0	8	2



Input Neurons

Output Neurons

Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

Design the neural network architecture

Assume a problem with the following dataset:

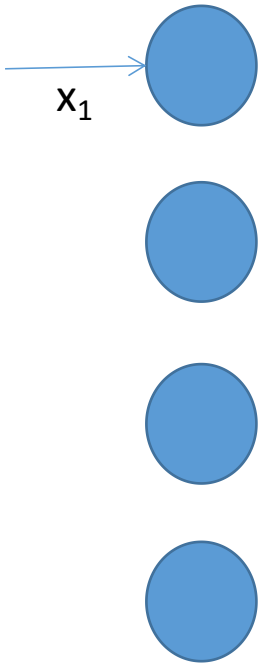
x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4

Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

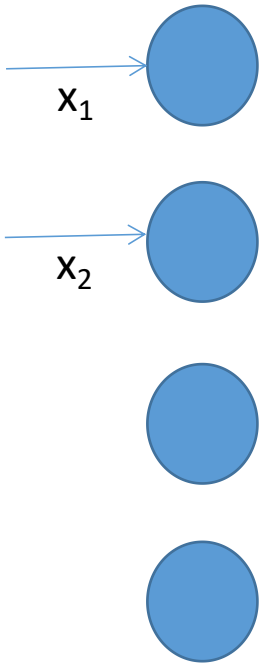


We have three classes and feature vectors in R^4

Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

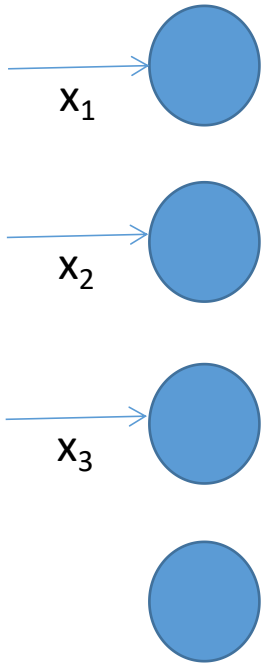


We have three classes and feature vectors in R^4

Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1



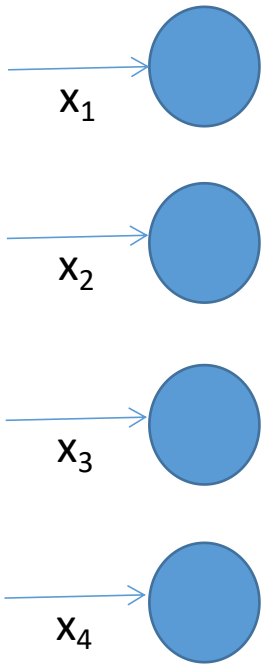
We have three classes and feature vectors in R^4

Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

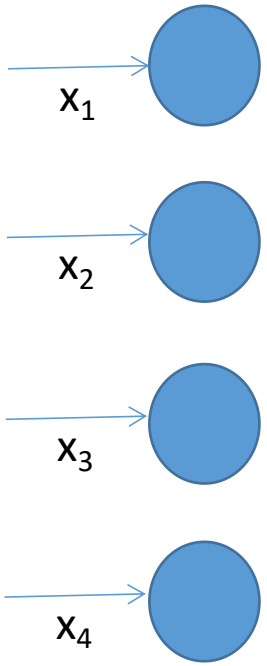
How many outputs?

Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

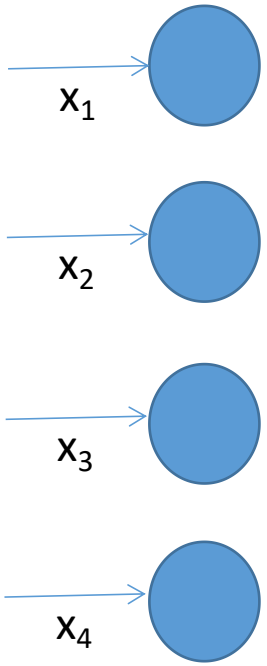
Option 1: As many as the number of classes

Design the neural network architecture

Assume a problem with the following dataset:

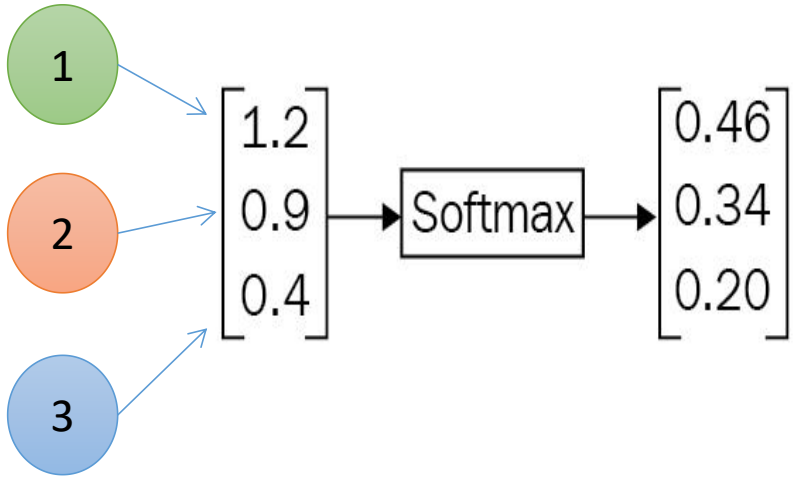
x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

Option 1: As many as the number of classes

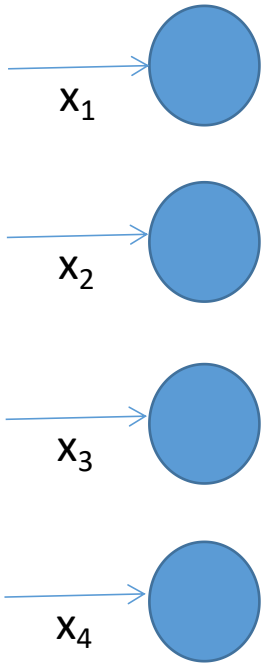


Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

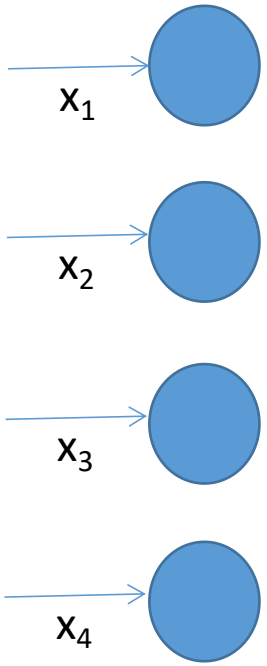
Option 2:
Use binary neurons and encode the class labels with them

Design the neural network architecture

Assume a problem with the following dataset:

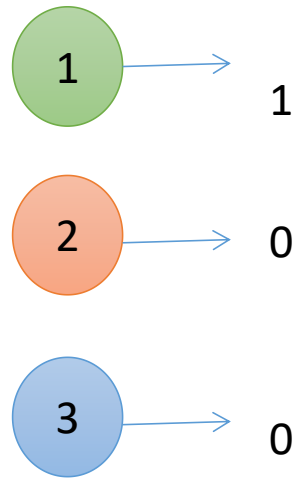
x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

Option 2:
Encode the class labels based on binary output of the neurons



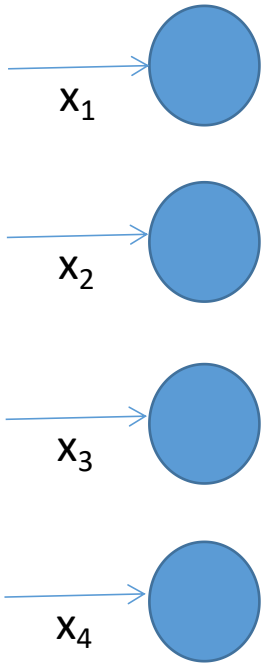
Class 1

Design the neural network architecture

Assume a problem with the following dataset:

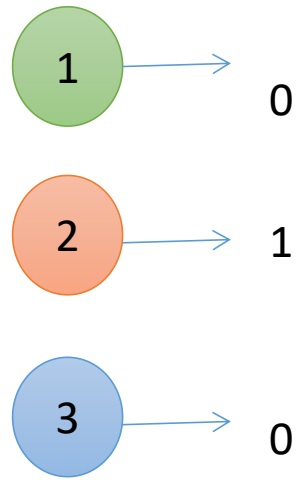
x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

Option 2:
Encode the class labels based on binary output of the neurons



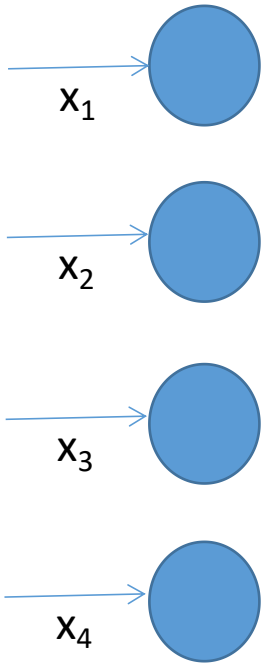
Class 2

Design the neural network architecture

Assume a problem with the following dataset:

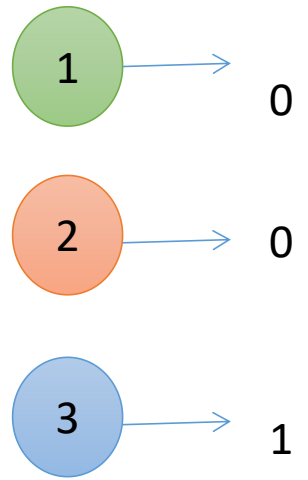
x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

Option 2:
Encode the class labels based on binary output of the neurons



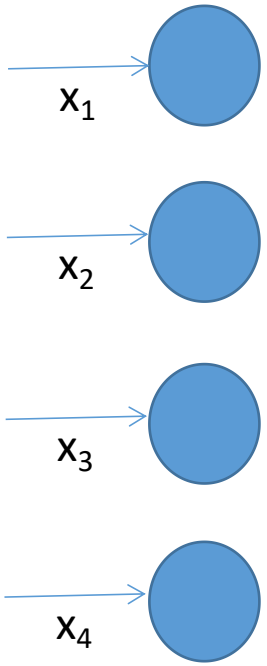
Class 3

Design the neural network architecture

Assume a problem with the following dataset:

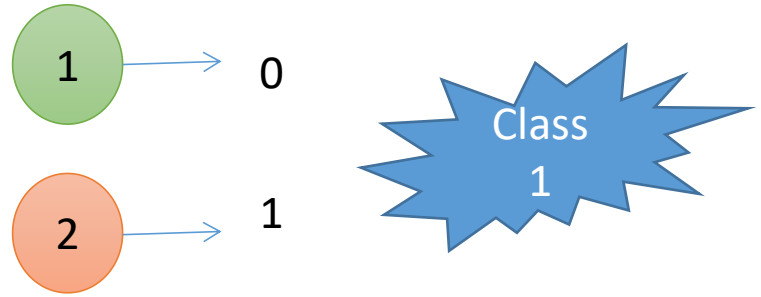
x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

Option 3: Code the class as binary number based on the outputs values

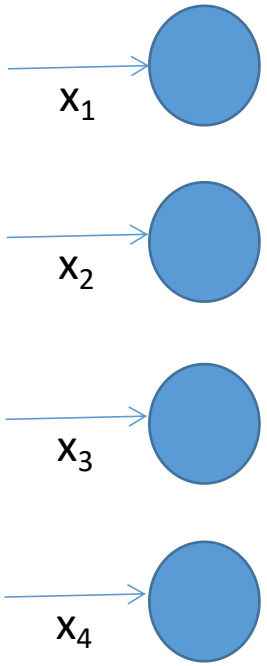


Design the neural network architecture

Assume a problem with the following dataset:

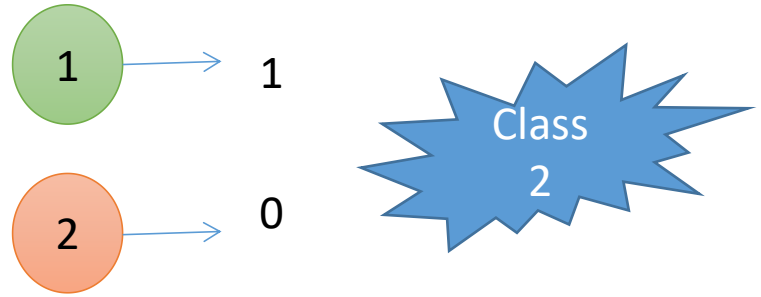
x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

Option 3: Code the class as binary number based on the outputs values

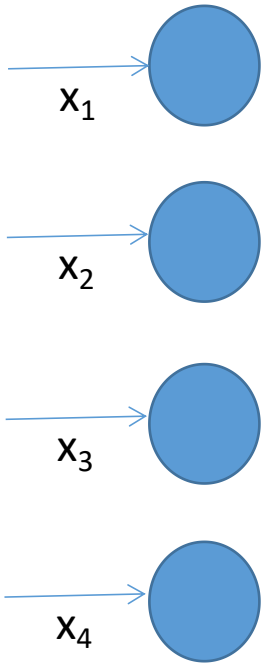


Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons

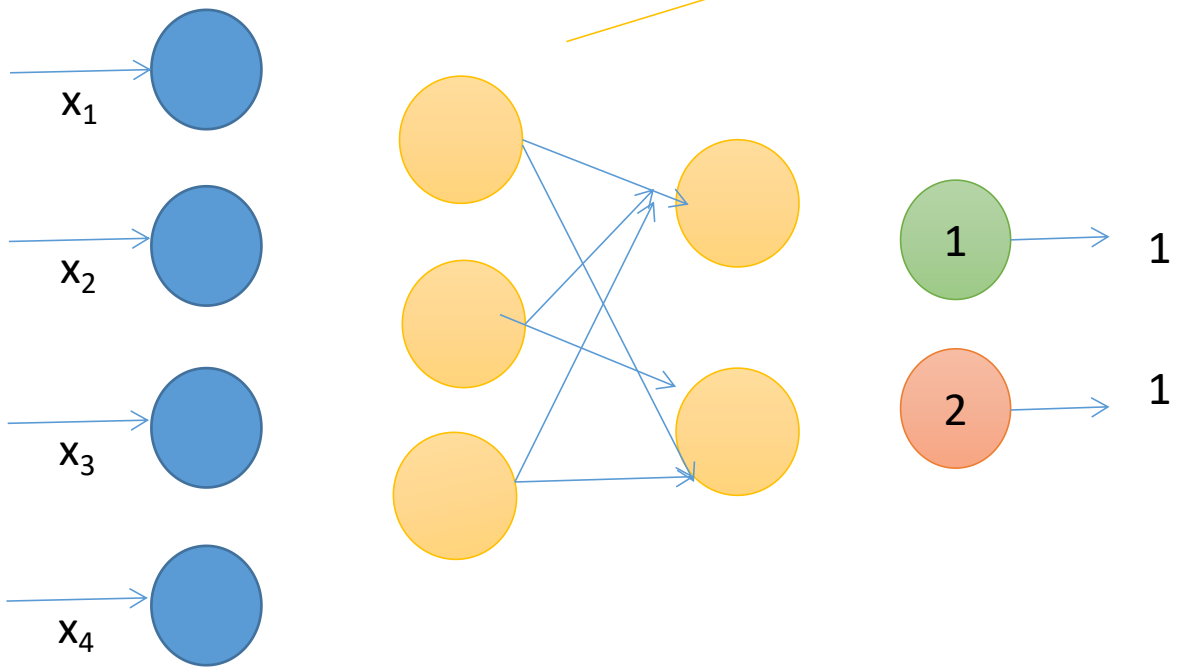
Option 3: Code the class as binary number based on the outputs values

Design the neural network architecture

Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



The number of hidden neurons is a parameter

Input neurons

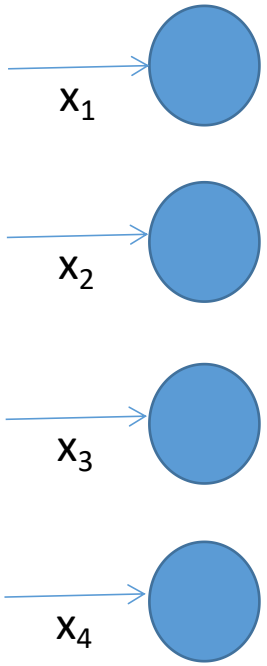
Output neurons

Design the neural network architecture

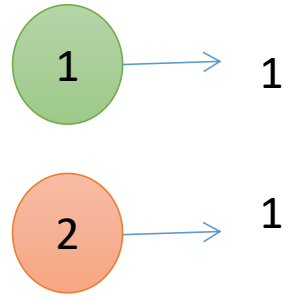
Assume a problem with the following dataset:

x1	x2	x3	x4	y
3	6	12	5	3
5	5	1	7	2
1	8	4	3	1
4	9	5	9	3
6	10	7	4	1

We have three classes and feature vectors in R^4



Input neurons



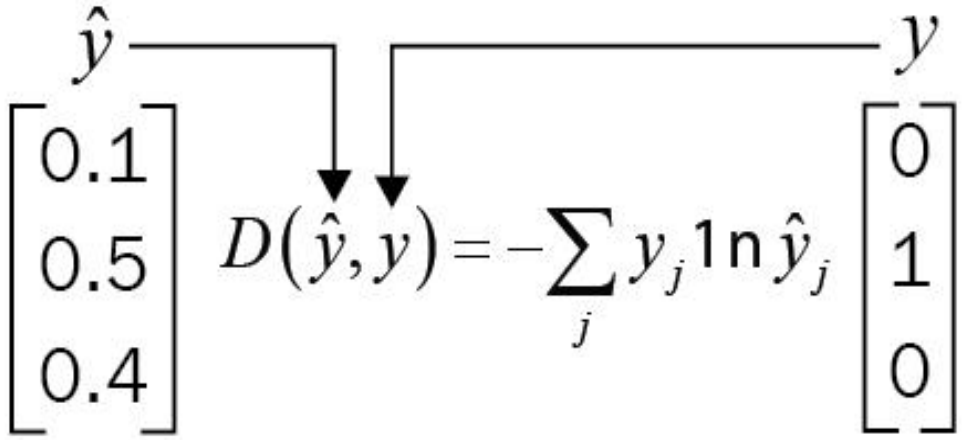
Output neurons

Loss Function

- The MLPN use different type of error functions in order to update the weights.
- What loss function to use? Depends on the problem
- For classification problems (binary and multiclass), you can use **Cross-Entropy**
- For regression problems, we can use **Mean Squared Error**.

Loss Function- Cross Entropy

- Cross entropy measures the similarity distance between two probability vectors by means of **cross-entropy**.
- In neural networks, we can use this measure to determine the loss (error) of a neural model. This measure is defined as:



Model Prediction

Actual Values