

# Redes Neuronales

## Perceptrón y Adeline

Universidad San Buenaventura, Cali

Junio de 2021

# Contenido

- 1 Perceptrón
- 2 Limitaciones del perceptrón
- 3 Adeline
- 4 Ejercicios

# Contenido

1 Perceptrón

2 Limitaciones del perceptrón

3 Adeline

4 Ejercicios

# Perceptrón

## Definición

- Fue introducido por Rosenblatt a finales de los años 50
- Se inspira en los procesos de aprendizaje de los animales (ejemplo la visión), en los cuales la información va atravesando diferentes capas de neuronas
- Es un modelo unidireccional, compuesto por dos capas de neuronas, una de entrada y otra de salida
- La operación de este tipo puede darse con  $n$  neuronas de entrada y  $m$  de salida

# Perceptrón

## Definición

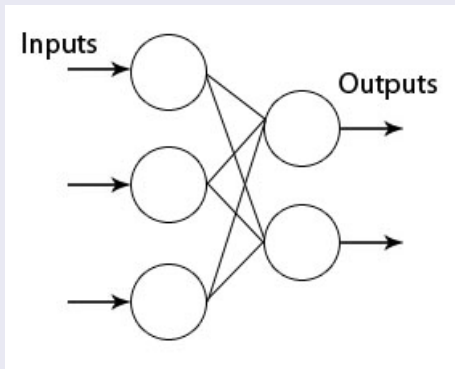


Figura 1: Modelo de Perceptrón, tomado de <http://neuroph.sourceforge.net/tutorials/Perceptron.html>

# Perceptrón

## Definición

- Las neuronas de entrada no realizan ningún computo
- Se consideran señales discretas 0 o 1
- La operación para  $n$  neuronas de entrada y  $m$  de salida puede considerarse así:

$$y_i = H\left(\sum_{j=1}^n w_{ij}x_j - \Theta_i\right), \forall i, 1 \leq i \leq m$$

Donde  $H(x)$  es la función escalón.

# Perceptrón

## Definición

- El Perceptrón permite clasificar dos conjuntos linealmente separables en un plano o hiperplano
- La respuesta de la neurona es 1 si pertenece a la clase o 0 si no pertenece

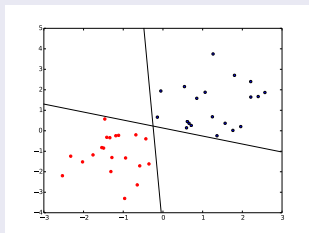



Figura 2: Conjunto linealmente separable, tomado de <https://en.wikipedia.org/>

# Perceptrón

## Ejemplo

- Sea una neurona tipo perceptron con entrada  $x_1$  y  $x_2$
- Entonces la operación se define como:

$$y = H(w_1x_1 + w_2x_2 - \Theta)$$




# Perceptrón

## Ejemplo

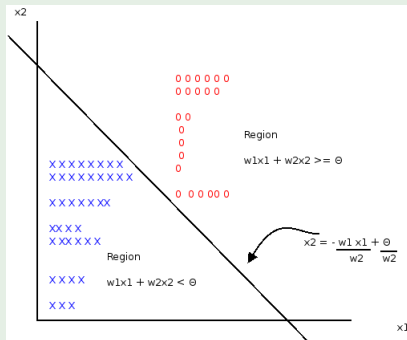


Figura 3: Regiones de decisión del plano, tomado de [Brio and Molina, 2005]

# Perceptrón

## Definición

Como se puede ver se divide el plano en dos regiones. Como se puede ver se requiere que el problema a solucionar tenga **solución lineal**

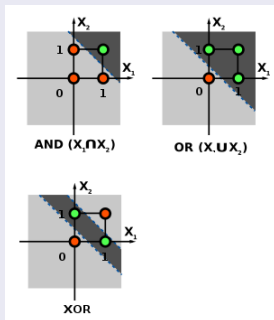


Figura 4: Casos de compuertas <https://en.wikipedia.org/>

# Perceptrón

## Algoritmo de aprendizaje













Estructura	Regiones de Decisión	Problema de la XOR	Clases con Regiones Mezcladas	Formas de Regiones más Generales
1 Capa 	Medio Plano Limitado por un Hiperplano			
2 Capas 	Regiones Cerradas o Convexas			
3 Capas 	Complejidad Arbitraria Limitada por el Número de Neuronas			

Figura 5: Regiones de decisión perceptron [Lippmann, 1988]

# Perceptrón

## Algoritmo de aprendizaje

Vamos a trabajar el perceptrón de una capa

- Se basa en la corrección de errores
- Vamos a introducir una tasa de aprendizaje  $\eta$ : Indica el ritmo de aprendizaje
- Dados unos patrones  $x^u$ , salidas obtenidas  $y^u$  y salidas deseadas  $t^u$
- Los pesos iniciales son aleatorios entre -1 y 1.
- Se examina cada patrón y aplicamos la relación de cambio:

$$\Delta w_{ij}^u(t) = \eta \cdot (t_i^u - y_i^u) x_j^u$$

A esto se le conoce como **regla del perceptrón**

# Perceptrón

## Algoritmo de aprendizaje

Vamos a trabajar el perceptrón de una capa

- Se basa en la corrección de errores
- Vamos a introducir una tasa de aprendizaje  $\eta$ : Indica el ritmo de aprendizaje
- Dados unos patrones  $x^u$ , salidas obtenidas  $y^u$  y salidas deseadas  $t^u$

# Perceptrón

## Algoritmo de aprendizaje

Vamos a trabajar el perceptrón de una capa

- Los pesos iniciales son aleatorios entre -1 y 1. Se utiliza la función de activación  $f$  como escalón o sigmoide. Las entradas están en el conjunto  $\{0, 1\}$ .
- Se examina cada patrón y aplicamos la relación de cambio:

$$\Delta w_{ij}^u = \eta \cdot (t_i^u - y_i^u) x_j^u$$

A esto se le conoce como **regla del perceptrón**

# Perceptrón

## Algoritmo de aprendizaje

Para comprender el Perceptrón se mostrará en una forma gráfica

$$y_i^u = f\left(\sum_{j=1}^n w_{ij}x_j^u - \Theta_i\right) = f(\|w_i\| \cdot \|x^u\| \cos(\phi))$$

# Perceptrón

## Algoritmo de aprendizaje

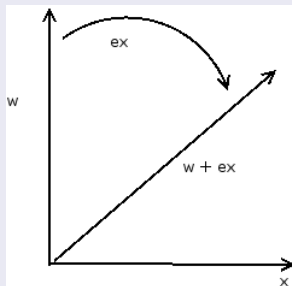


Figura 6: Aplicación regla perceptron [Brio and Molina, 2005]



# Perceptrón

## Algoritmo de aprendizaje

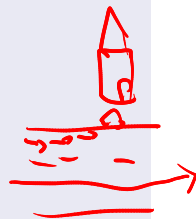
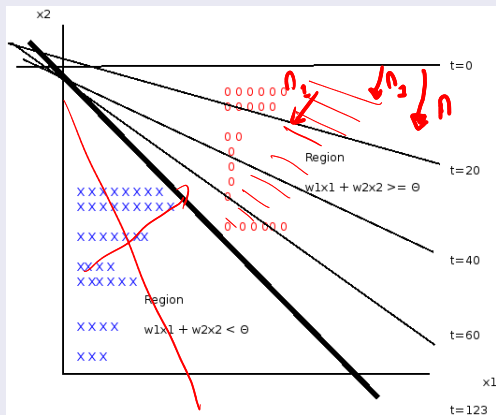


Figura 7: Aplicación iterativa de las reglas de decision  
[Brio and Molina, 2005]

# Perceptrón

## Algoritmo de aprendizaje

Es un algoritmo para clasificación binaria (0 o 1).

- 1 Inicializar los pesos aleatoriamente entre  $[-1$  y  $1]$
- 2 Para el estado  $t$ . Calcular:

$$y^u(k) = f\left(\sum_{j=1}^n (w_j x_j)\right)$$

- 3 Corregir pesos sinápticos (Si  $t_j^u \neq y_j^u$ )

$$w_j = w_j + \eta[t_j^u - y_j^u]x^u$$

- 4 Para si no se han modificado los pesos en los últimos  $p$  patrones o se ha llegado a un número de iteraciones especificado.

# Perceptrón

## Algoritmo de aprendizaje

Miremos la compuerta AND

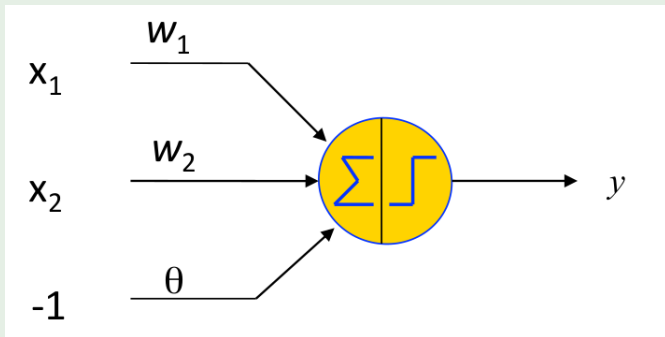


Figura 8: Perceptrón compuesta AND

# Perceptrón

## Algoritmo de aprendizaje

Miremos la compuerta AND

Entrada	Salida $t^u(k)$
(0,0)	0
(0,1)	0
(1,0)	0
(1,1)	1

Cuadro 1: Función AND con lógica función signo

# Perceptrón

## Algoritmo de aprendizaje

- 1 Inicialización de pesos. Elegimos  $\eta = 0,5$

$$w_1 = 0,4, w_2 = -0,2, \Theta = 0,6$$

- 2 Con  $t = 1$ , patrón  $(0,0)$ ,  $y^u = f(-0,6) = 0$ . Igual a salida esperada.
- 3 Para  $t = 1$ , patrón  $(0,1)$ ,  $y^u = f(-0,8) = 0$ . Igual a salida esperada.

# Perceptrón

## Algoritmo de aprendizaje

- 4 Para  $t = 1$ , patrón  $(1,0)$ ,  $y^u = f(-0,2) = 0$  Igual a salida esperada.
- 5 Para  $t = 1$ , patrón  $(1,1)$ ,  $y^u = f(-0,4) = 0$  Debemos corregir los pesos, ya que la salida debe ser 1. Recordando
- $$w_j = w_j + \eta[t_j^u - y_j^u]x^u$$

$$w_1 = 0,4 + 0,5[1 - 0](1) = 0,9$$

$$w_2 = -0,2 + 0,5[1 - 0](1) = 0,3$$

$$\Theta = 0,6 + 0,5[1 - 0](-1) = 0,1$$

$(1,1)$

0.9

0.3

0.1

 $t=2$  $s_t$  $(0, 0)$ 

$$0.9 \times 0 + 0.3 \times 0 - 0.1 = -0.1$$

0


 $(0, 1)$   
 $\uparrow$ 

$$0.9 \times 0 + 0.3 \times 1 - 0.1 = 0.2$$

1



$$w_1 = 0.9 + 0.5(0 - 1) \times 0$$

$$w_2 = 0.3 + 0.5(0 - 1) \times 1$$

$$w_2 = -0.2$$

$$\theta = 0.1 + 0.5(0 - 1) \times (-1)$$

$$\theta = 0.6$$

$$(1, 0)$$

$$w_1 = 0.9$$

$$w_2 = -0.2$$

$$\theta = 0.6$$

$$0.9 \times 1 + -0.2 \times 0 - 0.6 = 0.3$$

$$\frac{0.3}{1}$$

$$w_1 = 0.9 + 0.5(0 - 1) \times 1$$

$$w_1 = 0.4$$

$$\theta = 0.6 + 0.5(0 - 1) \times -1$$

$$\theta = 1.1$$

$$(1, 1)$$

$$w_1 = 0.4$$

$$w_2 = -0.2$$

$$\theta = 1.1$$

$$0.4 - 0.2 - 1.1 =$$

$$0$$



$$w_1 = 0.4 + 0.5(1 - 0) \times 1$$

$$w_1 = 0.9$$

$$w_2 = -0.2 + 0.5(1 - 0) \times 1$$

$$w_2 = 0.3$$

$$\Theta = 1.1 + 0.5(1 - 0) \times -1$$

$$0.6$$

$(0, 0)$	$-0.6$	$0 \checkmark$
----------	--------	----------------

$(0, 1)$	$-0.3$	$0 \checkmark$
----------	--------	----------------

$(1, 0)$	$0.3$	$1 \times$
----------	-------	------------

$$w_1 = 0.9 \quad w_2 = 0.3 \quad \theta = 0.6$$

$$w_1 = 0.9 + 0.5(0 - 1) \times 1$$

$$0.4$$

$$\theta = 0.6 + 0.8(0 - 1) \times 1$$

$$\theta = 1.1$$

		$w_1 = 0.4 \quad w_2 = 0.3$
		$\theta = 1.1$
$(0, 0)$	$< 0$	0 ✓
$(0, 1)$	$< 0$	0 ✓
$(1, 0)$	$< 0$	0 ✓
$(1, 1)$	$< 0$	0 ✗

$$w_1 = 0.4 + 0.5(1-0) \times 1$$

$$w_1 = 0.9$$

$$w_2 = 0.3 + 0.5(1-0) \times \underline{1}$$

$$w_2 = 0.8$$

$$\theta = 1.1 + 0.5(1-0) \times 1$$

$$\theta = 0.6$$

⋮

.

$$w_i[i] = w[i] + \eta e \times x_i$$

$$[0.2 \ 0.4] + \overset{\eta}{0.5} \overset{e}{[0.2 \ 0.4]} \times \overset{w}{[0.2 \ 0.4]} \times \overset{x_i^T}{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}$$

$$0.2 + 0.5 \times 0.2 \times 1 = 0.3$$

$$0.4 + 0.5 \times 0.4 \times 0 = 0.4$$

$$W = W + \eta \times e \times W \times X^T$$

$$w = [\underline{0.6} \quad 0.3]$$

$$y = 1 \quad y^d = 0 \quad \eta = 0.5$$

$$x_i = (\underline{0}, 1)$$

$$w[0] = 0.6 + 0.5 \times (-1) \times 0$$

$$w[1] = 0.3 + 0.5 \times (-1) \times 1$$

$$w = [\underline{0.6} \quad 0.3] + 0.5 \times (-1) \times \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$w[0] = 0.6 + 0.5 \times (-1) \times 0$$

$$w[1] = 0.3 + 0.5 \times (-1) \times \underline{1}$$

# Perceptrón

## Algoritmo de aprendizaje

$$w_1 = 0,9, w_2 = 0,3, \Theta = 0,1$$

- 6 Para  $t = 2$ ; patrón  $(1, 1)$ ,  $f(1,1) = 1$ . Correcto
- 7 Para  $t = 2$ ; patrón  $(1, 0)$ ,  $f(0,8) = 1$ . Correcto
- 8 Para  $t = 2$ ; patrón  $(0, 1)$ ,  $f(0,2) = 1$ . Correcto
- 9 Para  $t = 2$ ; patrón  $(0, 0)$ ,  $f(-0,1) = 0$  Correcto

# Contenido

1 Perceptrón

2 Limitaciones del perceptrón

3 Adeline

4 Ejercicios

# Limitaciones del perceptrón

## Limitaciones

- 1 El perceptrón monocapa no puede trabajar con problemas que no sean linealmente separables
- 2 Para esto utilizamos el perceptrón multicapa, la cual cuenta con una capa de entrada, capas ocultas y una capa de salida

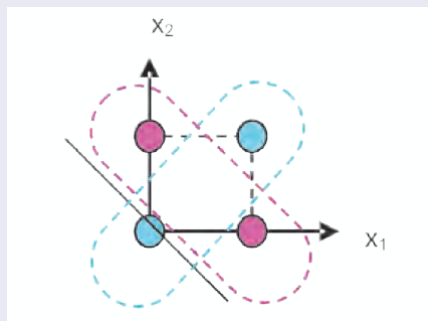


# Limitaciones del perceptrón

## Limitaciones

Para el caso de la función XOR, tenemos el siguiente problema:

**Figura 9:** Problema del perceptrón con compuerta XOR, tomado de [Eduardo and Jesus Alfonso, 2009]



# Contenido

1 Perceptrón

2 Limitaciones del perceptrón

3 Adeline

4 Ejercicios

## Definición

- Fue introducido por Widrow [Widrow and Hoff, 1988], [Widrow and Winter, 1988] entre 1959 y 1988.
- Es de respuesta lineal a diferencia del perceptrón
- Puede trabajar con entradas continuas
- Se incorpora un elemento adicional llamado **bias** u **umbral**  $\Theta$ . La cual se suma a la entrada (usualmente es -1)
- Utiliza mínimos cuadrados para el cálculo del error.
- Se tiene una tasa de aprendizaje  $\eta$

## Regla del gradiente descendiente

- Si las entradas tiene vectores de entrada ortogonales, se podrá llegar a asociaciones perfectas
- La salida de la neurona es  $y = \sum_{j=1}^n x_j w_j + \Theta$ . Ya que la función de activación es lineal
- El cambio se basa en el cálculo del gradiente descendiente para los patrones de entrada. En este caso el error cuadrático

$$Err = \frac{1}{2} \sum_{j=0}^n (t^u(j) - y^u(j))^2$$

## Regla del gradiente descendiente

- Lo que se busca es modificar los valores de forma iterativa mediante la regla del descenso del gradiente:

$$\Delta_p w_j = -\eta \frac{\partial Err^P}{\partial w_j}$$

## Regla del gradiente descendiente

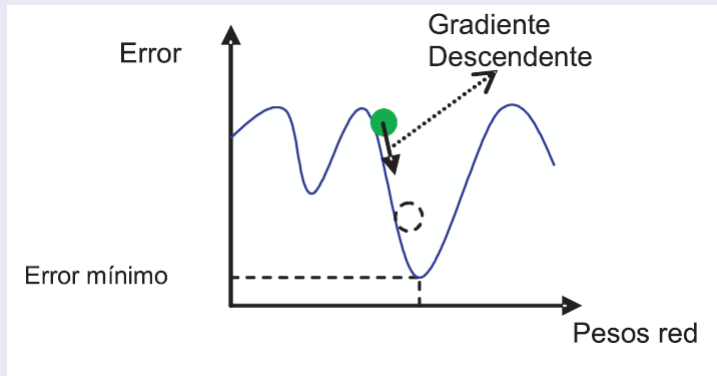


Figura 10: Regla del gradiente descendiente, tomado de [Eduardo and Jesus Alfonso, 2009]

## Regla del gradiente descendiente

- Tomando en cuenta que el error global cuadrático medio es:

$$E = \frac{1}{2}(t^u - y^u)^2$$

- La regla del gradiente descendiente busca el mínimo global, aplicando la regla en un patrón  $p$  cualquiera de la siguiente forma:

$$\Delta w_j = -\eta \frac{\partial E^p}{\partial w_j}$$

Esta va en sentido contrario, ya que deseamos hacer la derivada del error igual a 0

## Regla del gradiente descendiente

- Al aplicar la derivada obtenemos que:

$$\Delta w_j = \alpha(t^u - y^u) * x_j$$

Como se puede ver está regla corresponde a la regla perceptrón. La diferencia es que podemos trabajar con valores en todo el dominio de los números reales.



## Algoritmo de aprendizaje

- 1 Inicialice los pesos aleatorios
- 2 Para cada patrón, actualice los pesos a razón de:

$$w_i = w + \eta * \Delta w_j$$

- 3 Actualizamos los pesos y  $\Theta$
- 4 Puede detenerse cuando todos los patrones cumplen la salida deseada o bien se han cumplido cierto número de iteraciones.

# Contenido

1 Perceptrón

2 Limitaciones del perceptrón

3 Adeline

4 Ejercicios

# Perceptrón y adeline

## Ejercicio 1

Utilizar Perceptrón para reconocer la función binaria de 5 bits a,b,c,d,e:

**1**  $(a \text{ AND } b) \text{ OR } (c \text{ AND } d) \text{ OR } (\text{NOT } d \text{ AND } e)$

**2**  $(a \text{ AND } b \text{ AND } c) \text{ OR } (d \text{ AND } e)$

Utilice 16 entradas para entrenar. Genere otras 16 para probar y grafique el error de entrenamiento y de prueba.

# Perceptrón y adeline




## Ejercicio 2

Utilizar Adeline para crear un codificador binario-decimal.

$x_1$	$x_2$	$y$
0	1	1
1	0	2
1	1	3

Cuadro 2: Codificador binario a decimal

# Referencias I

-  Brio, B. and Molina, A. (2005).  
*Redes neuronales y sistemas difusos.*  
Textos universitarios. Alfaomega.  
Pages 41–63.
-  Eduardo, C. and Jesus Alfonso, L. (2009).  
*Una aproximación práctica a las redes neuronales artificiales.*  
Colección Libros de Texto. Programa Editorial Universidad del Valle.
-  Lippmann, R. P. (1988).  
An introduction to computing with neural nets.  
*SIGARCH Comput. Archit. News*, 16(1):7–25.

# Referencias II



Widrow, B. and Hoff, M. E. (1988).

Neurocomputing: Foundations of research.

chapter Adaptive Switching Circuits, pages 123–134. MIT Press, Cambridge, MA, USA.



Widrow, B. and Winter, R. (1988).

Neural nets for adaptive filtering and adaptive pattern recognition.

*Computer*, 21(3):25–39.

# ¿Preguntas?

Próximo tema:  
Perceptrón multicapa