# REDES NEURONALES

## Universidad Nacional Mayor de San Marcos
## Facultad de Ingeniería de Sistemas e Informática

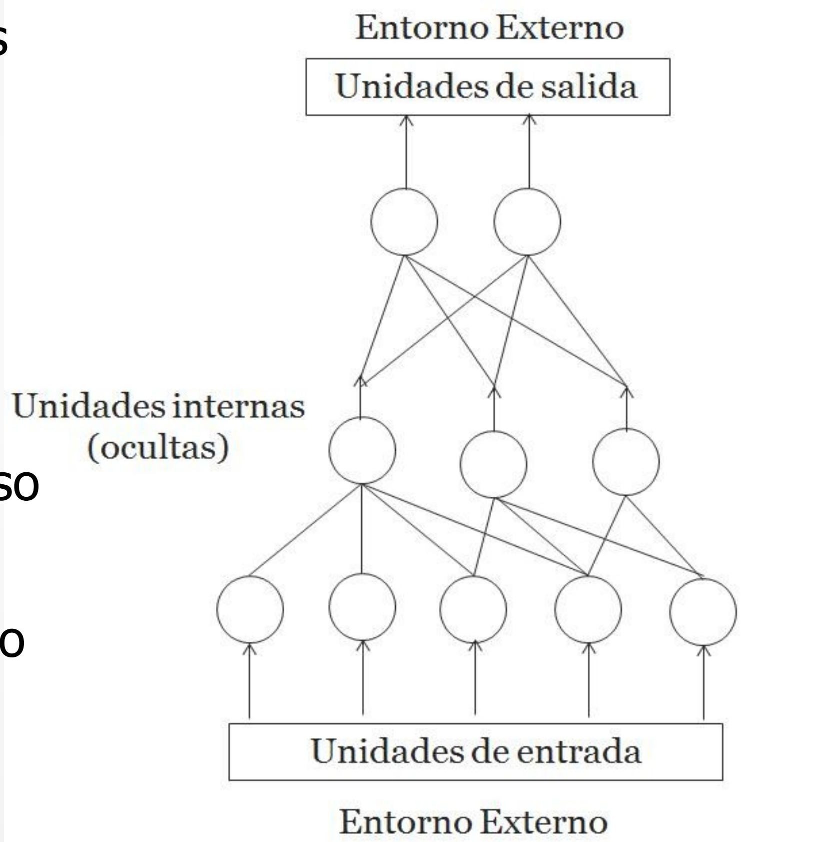
 Ing. Mg. Rolando A. Maguiña Pérez

# Agenda

- ✓ Introducción a las RN
  - ✓ Motivación/justificación del uso de RN
  - ✓ Cerebro como modelo computacional
  - ✓ Definición de RNA
- ✓ Neurona Biológica y Neurona artificial
- ✓ RNA
- ❖ Tipos de problemas abordados- cont
- ❖ Perceptrón Unicapa
  - Características
  - Ejms de aplicación

# Una Red Neuronal Artificial

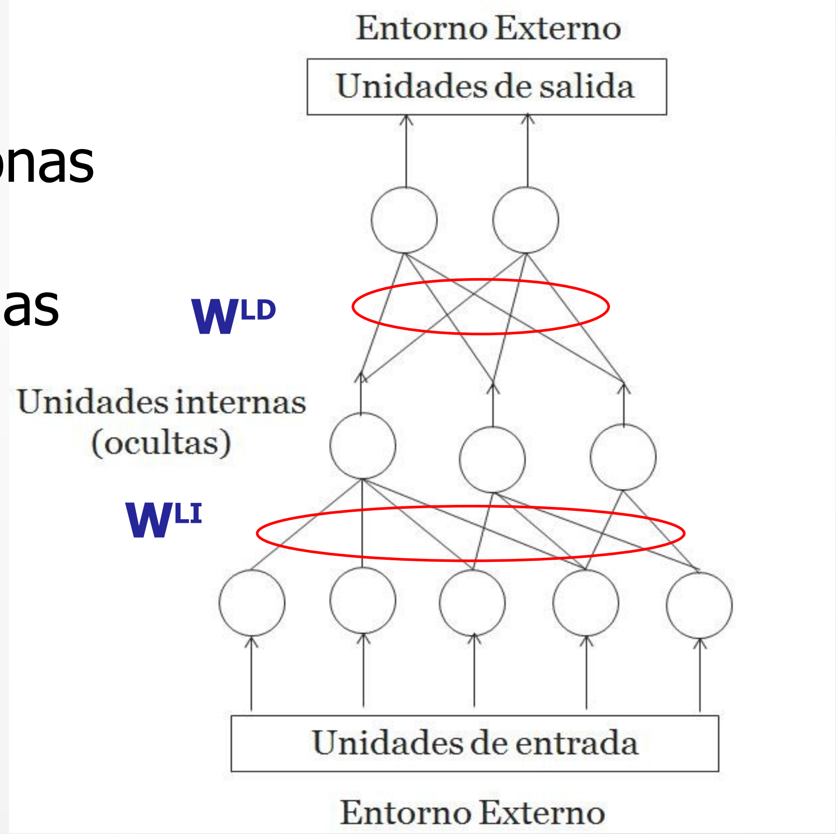
- Una red neuronal se comunica con el entorno externo a través de sus unidades de entrada y de salida. A todos los otros elementos se les denomina unidades internas u ocultas
- Las unidades son conectadas mediante enlaces unidireccionales
- Una conexión es caracterizada por un peso y un signo
- En las neuronas de la capa de entrada no se realiza operación alguna!
- Los pesos son los parámetros libres de la red!



# Una Red Neuronal Artificial

❖ La red neuronal de la figura presenta 3 capas:

- una capa de entrada con 'n' neuronas
- una capa oculta con 'p' neuronas
- una capa de salida con 'm' neuronas

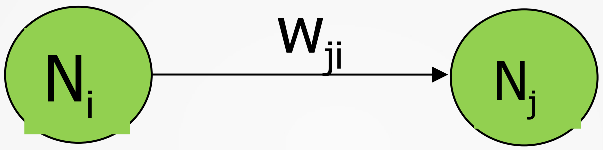



❖ Matrices de conexión en red ejemplo:

- $W^{LI}$ : pesos o sinapsis entre las neuronas de entrada y las ocultas
- $W^{LD}$ : pesos o sinapsis entre las neuronas ocultas y las de salida

# Una Red Neuronal Artificial

- La sinapsis entre “axón” de neurona $N_i$ y “dendrita” $N_j$ , es representada por el escalar $w_{ji}$




- El conjunto de pesos de la red forman la matriz de conexión

# Una Red Neuronal Artificial

- En una neurona artificial "j" se debe considerar las siguientes reglas:

Regla de propagación ($u_j$)

$$= \sum$$

Regla de activación $f(u_j)$

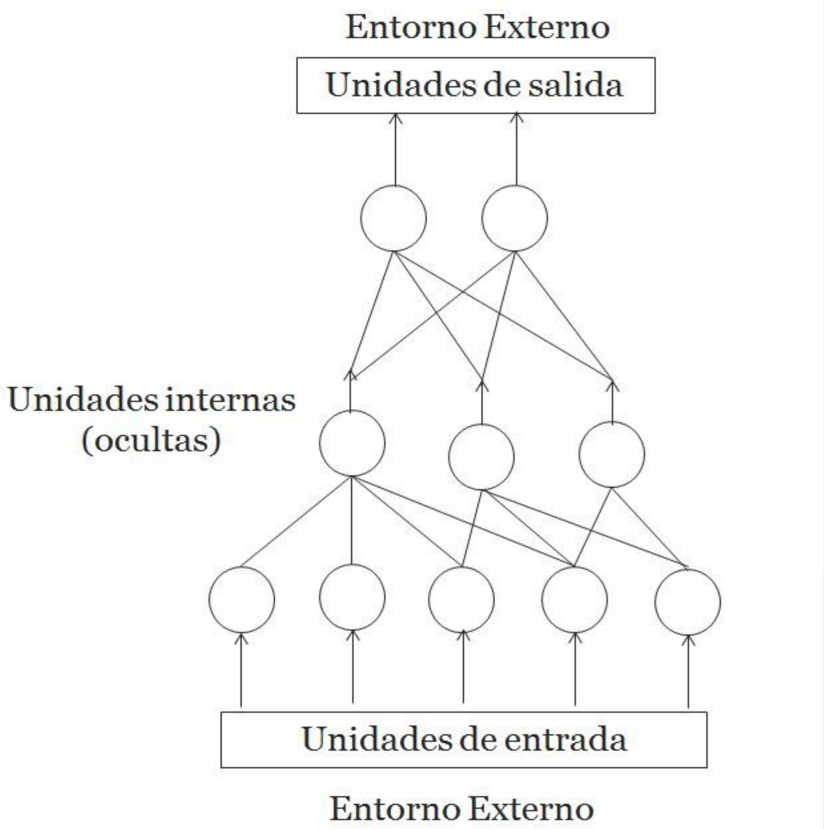
Función escalón y función signo.

Además: función sigmoidal, lineal, etc.

Regla de salida ($y_j$)

$$y_j = f(u_j)$$

Regla de aprendizaje ($w_{ji}$)



# Una Red Neuronal Artificial

## Procedimiento general de la técnica

❖La aplicación de la técnica llamada RN comprende, a grandes rasgos:

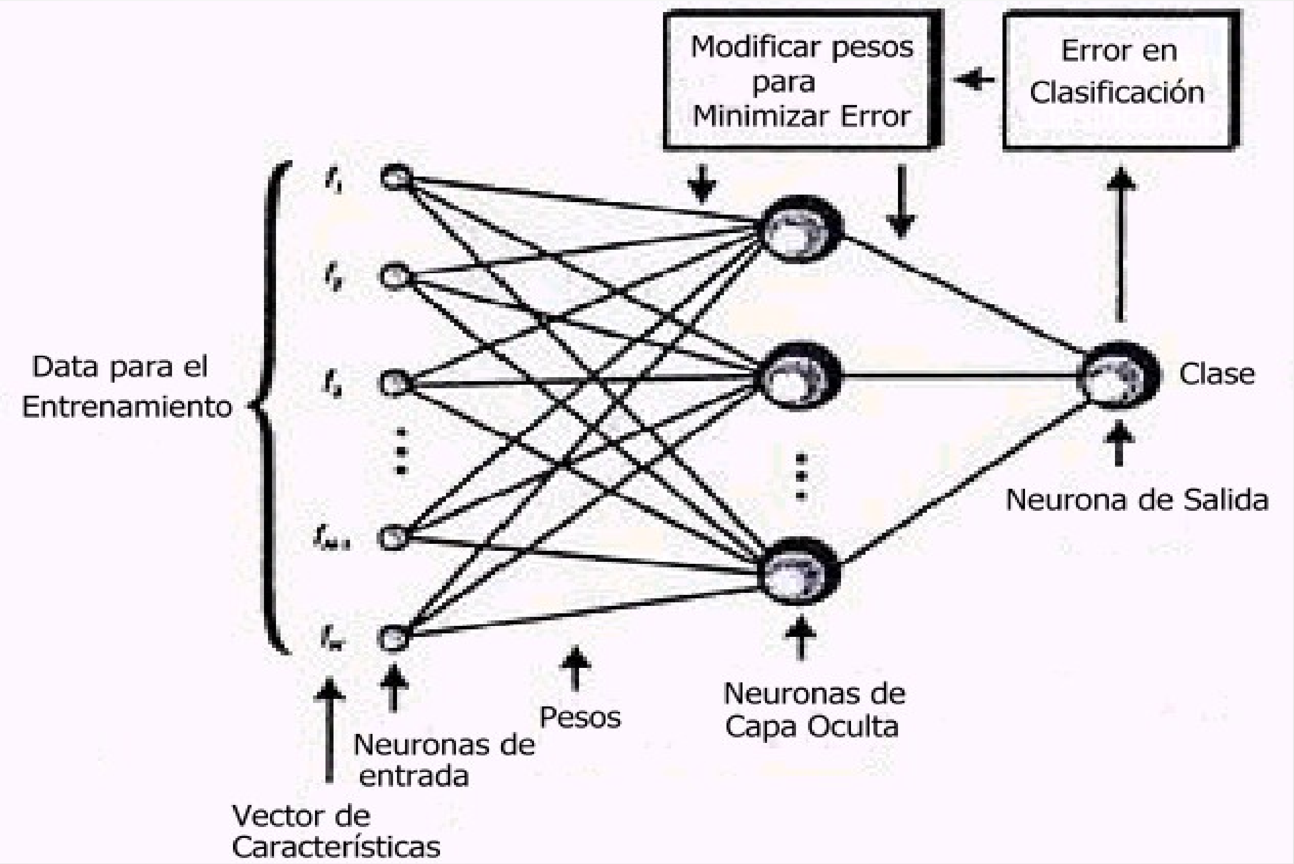
- Seleccionar los ejemplos para el entrenamiento.

Definir un conjunto de L ejemplos de entrenamiento; cada ejm está constituido por un par (*s*,*t*) donde *s* representa a las entradas que deben generar las salidas *t*

- Determinar la arquitectura de la red
- Entrenamiento de la red
- Validación del entrenamiento

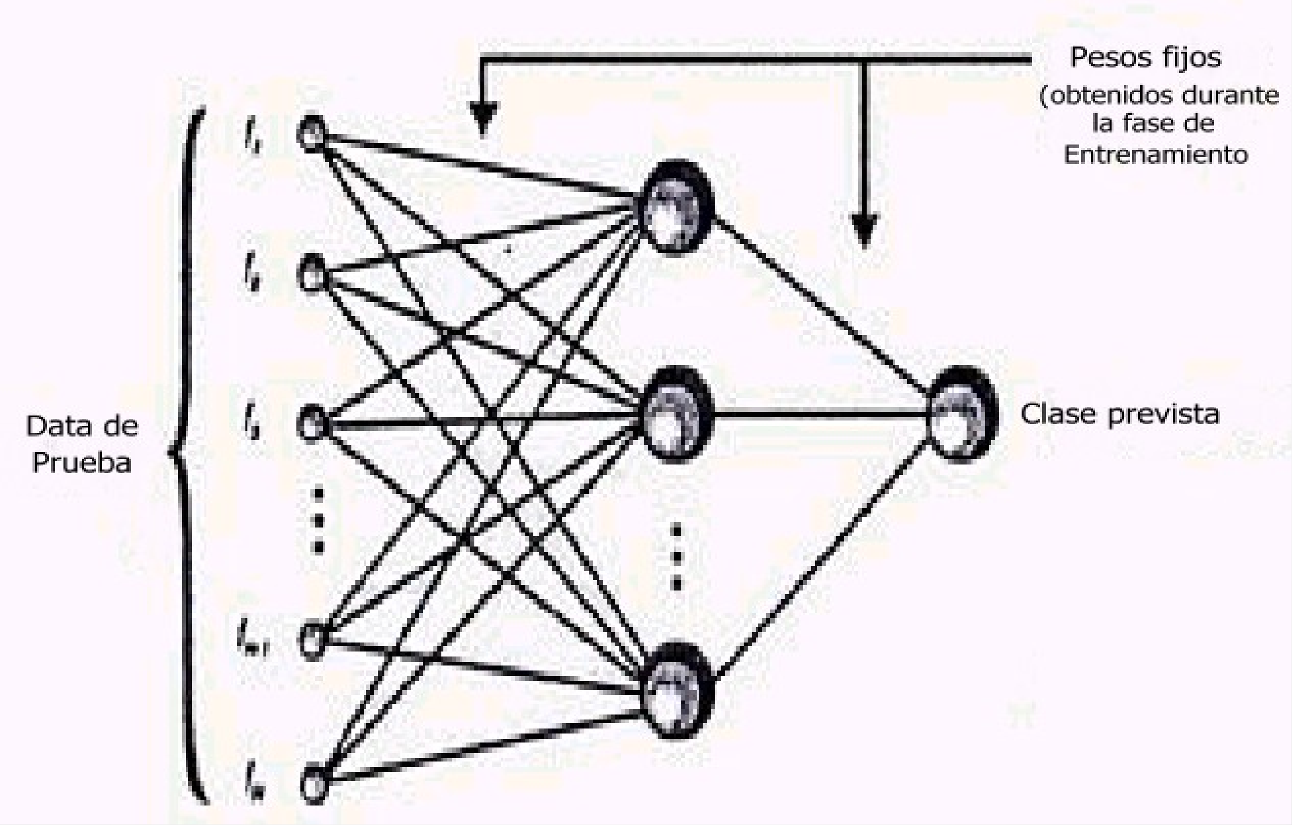
# Fases de operación de una RNA

## Fase de Entrenamiento (memoria)



# Fases de operación de una RNA

## Fase de ejecución (recuerdo)



# PERCEPTRON UNICAPA

Concepto

Características

Aprendizaje

Problemas

# Perceptrón unicapa

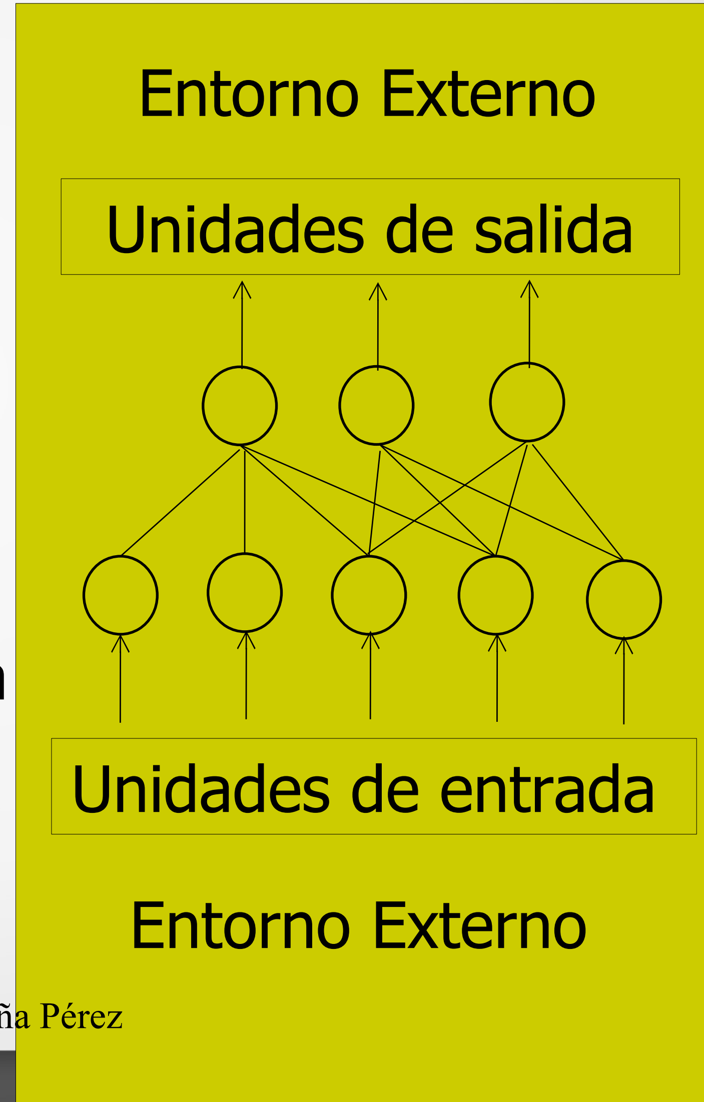
## Características

- Presentada por Rosenblat (1957)
- Red unidireccional monocapa
- Regla de propagación

$$=\sum \qquad =\sum$$

- Regla de activación de tipo lógica
- Regla de salida

$$y_j = f(u_j)$$



# Perceptrón unicapa

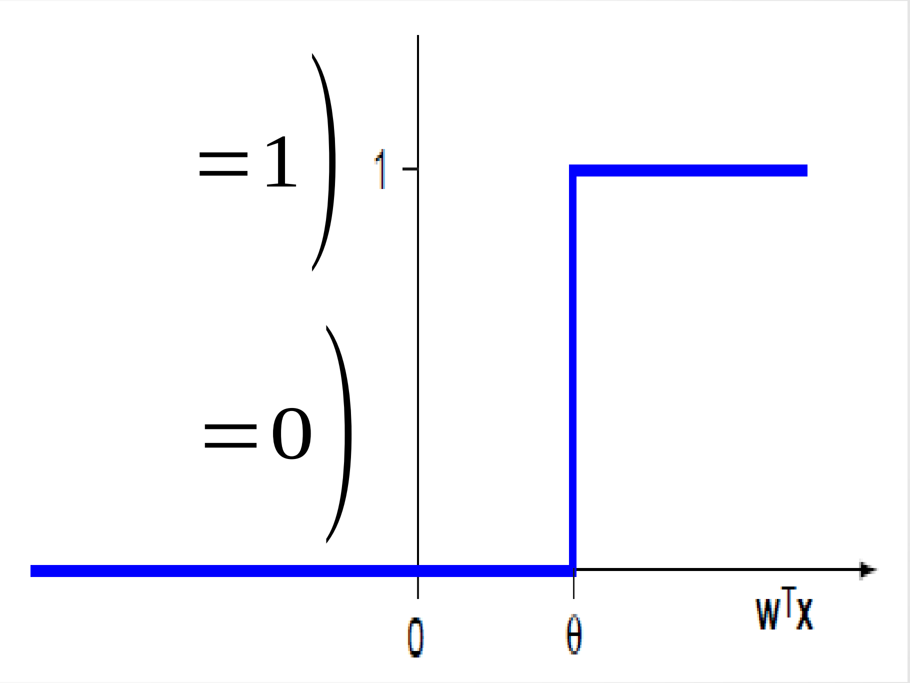
- Regla de activación

$\geq$

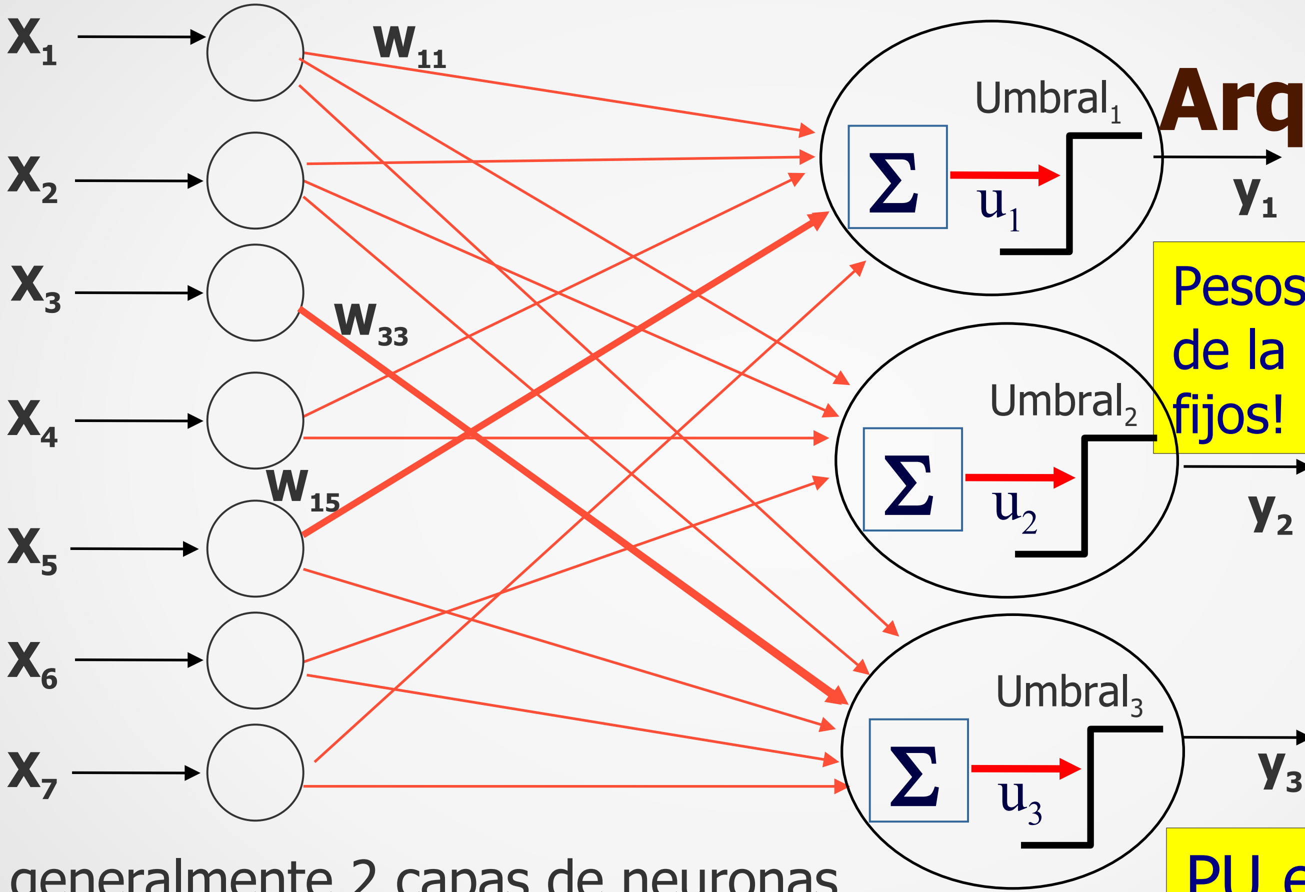
$<$

$=1)$

$=0)$



# Perceptrón unicapa




## Arquitectura

Pesos y Umbrales de la red no son fijos!

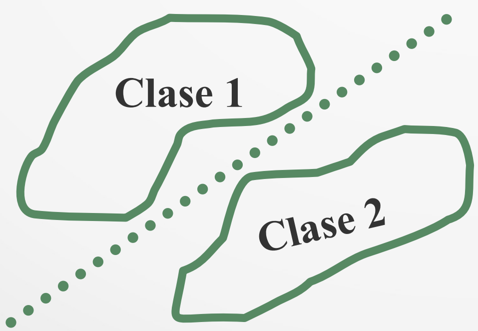
- generalmente 2 capas de neuronas
- función de activación: función lógica
- aprendizaje hebbiano

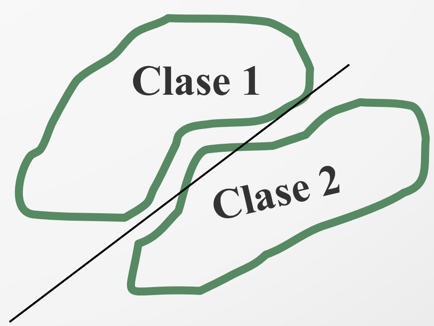
PU es usado en problemas de clasificación!

# Perceptrón unicapa

## Características

- Forma más Simple de una Red Neuronal
- Utilizada para la Clasificación de Patrones Linealmente Separables
- Cada neurona clasifica en dos clases diferentes



# Perceptrón unicapa

## Regla de propagación

suma ponderada de valores de entrada por sus pesos

$$u_j = \sum_{i=0}^{N} w_{ji}\, x_i$$

## Regla de activación

función no lineal aplicada al resultado de sumatoria

$$f(u_j) = \begin{cases} 1 & \text{si } u_j \geq \text{Umbral}_j \\ 0 & \text{si } u_j < \text{Umbral}_j \end{cases}$$

# Perceptrón unicapa

## Regla de aprendizaje

- Basada en estudio efectuado por Donald Hebb quien en 1949 propuso un mecanismo de aprendizaje para la neurona biológica
  - Enunciado:

"Cuando un axón presináptico causa la activación de cierta neurona postsináptica, la eficacia de las sinapsis que las relaciona se refuerza"

- Representación matemática de ese postulado:

$$\Delta w_{ji} = \eta \, y_j \, x_i$$

# Perceptrón unicapa

## Regla de aprendizaje

Modificar los pesos cada vez que se equivoca en su respuesta, según:

$$\Delta w_{ji} = \eta \, (t_j - y_j) \, x_i$$

$$w_{ji}\,(\text{nuevo}) = w_{ji}\,(\text{anterior}) + \eta \, (t_j - y_j) \, x_i$$

donde:

$t_j$: salida deseada para la j-ésima neurona

$y_j$ : salida estimada por la red para la j-ésima neurona

$x_i$: i-ésima entrada para la neurona j

$\eta$: tasa de aprendizaje, paso del entrenamiento

valor típico de $\eta$: 0.1

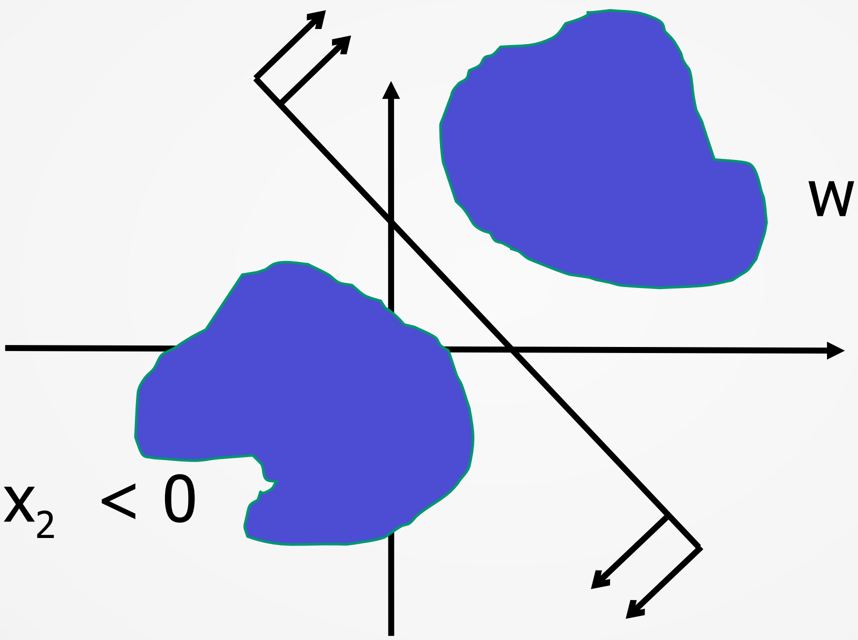
# Perceptrón unicapa

## Teorema de convergencia

- Si las clases son linealmente separables, el algoritmo converge a una solución correcta en un número finito de pasos para cualquier elección inicial de pesos
  - Interpretación: dado que la red convergerá, los ajustes o modificaciones de los pesos se hacen hasta que no haya error en la salida de las neuronas de la red

# Perceptrón unicapa

## Separabilidad lineal



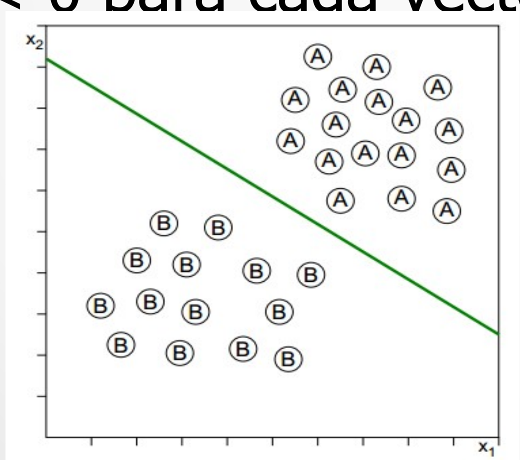
$$w_{11}\, x_1 + w_{22}\, x_2 \geq 0$$

$$w_{11}\, x_1 + w_{22}\, x_2 < 0$$

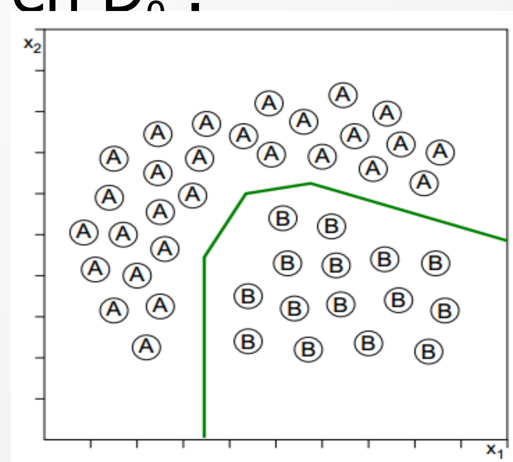
# Perceptrón unicapa

## Separabilidad lineal

- Dos cjtos $D_1$ y $D_0$ que contienen vectores en um espacio *p*-dimensional son linealmente separables si existen *p+1* números reales $w_1$ , $w_2$ ..., $w_p$ tal que $w_{11}\ x_1 + w_{22}\ x_2 \geq 0$ para cada vector en $D_1$ y $w_{11}\ x_1 + w_{22}\ x_2 < 0$ para cada vector en $D_0$ .




Linealmente separables




No Linealmente separables

# PU – algoritmo de aprendizaje

Paso 0: Inicializar los pesos y umbrales (0 o valores aleatorios pequeños)

Establecer la tasa de aprendizaje $\eta$ $(0 < \eta <= 1)$

Paso 1: Mientras la condición de parada sea falsa, hacer pasos 2-6

Paso 2: Para cada par de entrenamiento (binario o bipolar) s:t, hacer pasos 3-5

Paso 3: Establecer la activación de cada unidad de entrada i= 1,..,m

$$x_i = s_i$$

Paso 4: Calcular la respuesta de cada unidad de salida j= 1,..,n

$$u_j = w_{0j} \times x_0 + w_{ji} \times x_i$$

$$y_j = \begin{cases} 1, & \text{si } u_j \geq 0 \\ 0, & \text{si } u_j < 0 \end{cases}$$

Paso 5: Actualizar los umbrales y pesos de la red (j=1,..,n; i= 1,..,m)

Si $t_j \neq y_j$ entonces

$$w_{ji}\,(\text{nuevo}) = w_{ji}\,(\text{anterior}) + \eta \times (t_j - y_j) \times x_i$$

$$w_{oj}\,(\text{nuevo}) = w_{oj}\,(\text{anterior}) + \eta \times (t_j - y_j) \times x_0$$

sino

$$w_{ji}\,(\text{nuevo}) = w_{ji}\,(\text{anterior})$$

$$w_{oj}\,(\text{nuevo}) = w_{oj}\,(\text{anterior})$$

//pesos y umbrales no se modifican

Paso 6: Verificar condición de parada

Si no ocurre cambio alguno de pesos y umbrales para todos los patrones entonces parar, sino regresar a paso 2

Fin mientras

# PU – algoritmo de entrenamiento

Paso 0: Inicializar los pesos y umbrales (0 o valores aleatorios pequeños)

Establecer la tasa de aprendizaje $\eta$ $(0 < \eta <= 1)$

Paso 1: Mientras la condición de parada sea falsa, hacer pasos 2-6

Paso 2: Para cada par de entrenamiento (binario o bipolar) s:t, hacer pasos 3-5

Paso 3: Establecer la activación de cada unidad de entrada i= 1,..,n

$$x_i = s_i$$

Paso 4: Calcular la respuesta de cada unidad de salida j= 1,..,m

$$u_j = w_{0j} \times x_0 + \sum w_{ji} \times x_i$$

$$y_j = \begin{cases} 1, & \text{si } u_j \geq 0 \\ 0, & \text{si } u_j < 0 \end{cases}$$

Paso 5: Actualizar los umbrales y pesos de la red (j=1,..,m; i= 1,..,n)

Si $t_j \neq y_j$ entonces

$w_{ji}$ (nuevo) = $w_{ji}$ (anterior) + $\eta \times (t_j - y_j) \times x_i$

$w_{oj}$ (nuevo) = $w_{oj}$ (anterior) + $\eta \times (t_j - y_j) \times x_0$

sino

$w_{ji}$ (nuevo) = $w_{ji}$ (anterior)

$w_{oj}$ (nuevo) = $w_{oj}$ (anterior)

Paso 6: Verificar condición de parada

Si no ocurre cambio alguno de pesos y umbrales para todos los patrones entonces parar, sino regresar a paso 2

Fin mientras

# PU – algoritmo de recuerdo

En esta fase, después de haber encontrado los pesos ideales en el entrenamiento, se aplica sólo la etapa Forward del mismo

Paso 0: Establecer los pesos ideales (aplicar algoritmo de entrenamiento)
//FEEDFORWARD
Paso 1: Para cada vector de entrada, hacer pasos 2-3
Paso 2: Para i=1 hasta n, establecer la activación de la unidad de entrada $x_i$
Paso 3: Para j=1 hasta m

$$u_j = w_{0j} \times x_0 + w_{ji} \times x_i$$

$$y_j = \begin{cases} 1, & \text{si } u_j \geq 0 \\ 0, & \text{si } u_j < 0 \end{cases}$$

# Perceptrón unicapa

EJM. = PU