

---Paradigma: Algoritmos genéticos/evolutivos---

Tipos de Aprendizaje: paradigmas (recordatorio)

- ☐ Aprendizaje inductivo
- ☐ Aprendizaje analítico o deductivo
- ☐ **Algoritmos genéticos**
- ☐ Métodos conexionistas (enfoque subsimbólico)

☐ Clasificaciones del aprendizaje:

- ☐ Realimentación: supervisado
- ☐ Paradigma: Algoritmos genéticos
- ☐ Qué aprende: encuentra solución a un problema o función

☐ Simulan la evolución de las especies:

- ☐ Crear nuevas generaciones de soluciones
- ☐ Evolucionan mediante selección, cruce y mutaciones
- ☐ Las evalúa y se queda con la mejor

Algoritmos genéticos : Conceptos

- ❑ Población: Conjunto de individuos
 - ❑ estados generados aleatoriamente (N posible soluciones)
 - ❑ Cada nueva generación tiene el mismo N
 - ❑ Las operaciones se aplican a todos los individuos de la población anterior

- ❑ Codificación Individuo: (*cromosoma*)
 - ❑ Dígitos
 - ❑ Cadena de 1's y 0's
 - ❑ Letras

- ❑ Condición parada: num. generaciones, una calidad, convergencia
- ❑ Esquema: patrones de bits que permanecen en los mejores individuos
 - ❑ Ej: dígito 5 y 6 son "1".

Algoritmos genéticos : Pasos del Algoritmo

- ❑ Paso inicial : Generar aleatoriamente la *población*.
- 1. Asignar calidad a cada solución de la población ($f(i)$ ajuste)
- 2. Crear una nueva población de N soluciones nuevas, operaciones:
 - A. *Selección*: Escoger pareja (*ruleta*) y decidir si emparejamiento (*cruce*)
 - B. *Cruce*: Intercambiar un conjunto de genes (*punto de cruce*)
 - C. *Mutación*: cambiar bits de cada cromosoma (m. rate)
 - ❑ Se hace aleatoriamente y usando una probabilidad (*ratio de mutación*)
 - ❑ Si obtienes dos veces el mismo: conservas ambas copias
 - ❑ Se repiten estos pasos hasta tener N nuevas soluciones
- 3. Comprobar que no se han alcanzado las *condiciones de parada*
 - a) Un número de generaciones máximo ó
 - b) Alcanza éxito: condición de solución óptima
- 4. Volver a 1.

OPCIONAL: 1.1 Conservar la *élite* : Ordenar la población por calidad, conservar los **X** mejores, y aplicar a los N - X el resto de los pasos hasta obtener N – X soluciones nuevas

Algoritmos genéticos : Selección con Ruleta

- ❑ Para escoger qué individuos se reproducen en siguiente generación
- ❑ *Función de idoneidad* o ajuste o “fitness” o adaptación: **$f(i)$**
 - ❑ Mide la bondad o calidad del estado o individuo (línea “adaptación”)
 - ❑ Más cerca de la solución o una solución mejor
- ❑ Se usa probabilidad de ser escogido $p_i = f(i) / f_{\text{TOTAL}}$
- ❑ Para calcular f_{TOTAL} : es la suma de la $f(i)$ de cada individuo (= 20 en ej.)
- ❑ Puntuación acumulada $q_0 := 0$, ... $q_i := p_1 + \dots + p_i$
- ❑ Para cada individuo a seleccionar:
 - ❑ Se genera num. aleatorio $0 \leq a \leq 1$
 - ❑ Se selecciona el individuo (i) que cumpla $q_{i-1} < a < q_i$
- ❑ Ejemplo: $a = 0.8751$ selecciono x_7 , $a = 0.1391$ selecciono x_1

Individuo	1	2	3	4	5	6	7	8
$f(i)$ Adaptación	4	1	1	2	3	2	5	2
p_i	0.2	0.05	0.05	0.1	0.15	0.1	0.25	0.1
q_i	0.2	0.25	0.3	0.4	0.55	0.65	0.9	1.0

$f(1) = 4 / 20$

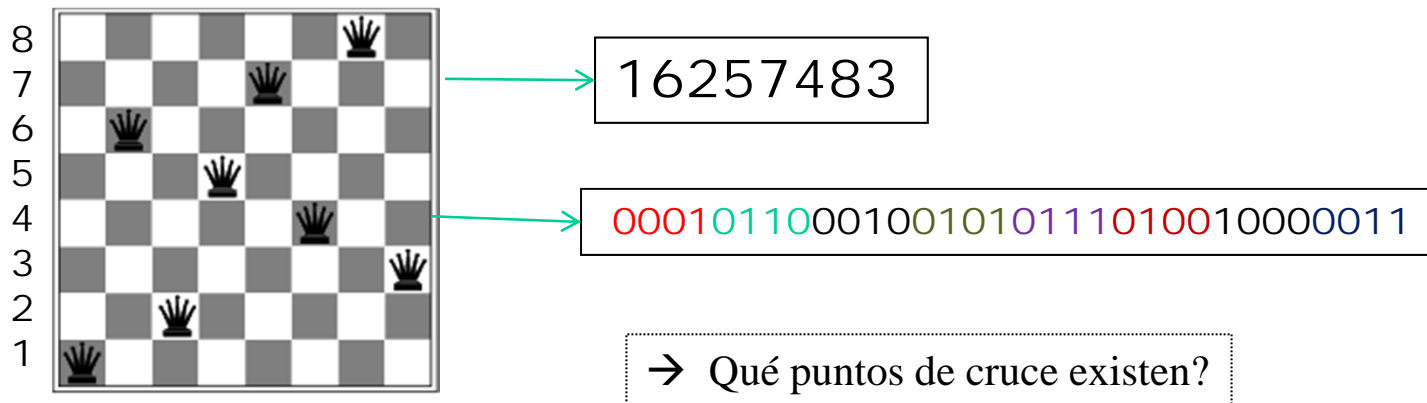
Algoritmos genéticos : Después de seleccionar

- ❑ Decidir si dos cromosomas seleccionados se emparejan
- ❑ **Pe**: Probabilidad de emparejamiento (crossover rate) :
 - ❑ Es un dato que hace de umbral: 0.7 por ejemplo
 - ❑ Se elige **a**, un num. aleatorio.
 - ❑ Si **a** \leq **Pe** se emparejan y pasan a hacer cruce.
 - ❑ En caso contrario sigue con la mutación
- ❑ *Crossover rate*: *Punto* *cruce* en la cadena desde donde cambiar genes
 - ❑ Elige **a** valor al azar, escoge el punto de cruce según...
 - ❑ en qué intervalo cae: $x / (\text{num.genes} - 1)$ donde $x : [1..\text{genes}]$
 - ❑ ej.: hay 8 posibles puntos para 9 genes $1/8 \dots 8/8$
 - ❑ Si **a** está entre 0 y $1/8$ se cruza después del gen 1º
- ❑ **Pm** *Ratio de mutación*: a partir de este umbral el gen muta (dato,ej: 0.1)
 - ❑ Para cada cromosoma:
 - ❑ Para cada gen: se elige **a** núm. aleatorio
 - ❑ Si **a** $<$ **Pm** , ese gen muta (cómo muta es un dato)
 - > Definir cómo muta es específico del dominio del problema

Qué puntos de cruce existen?: decide el dominio.
Condición: genes enteros.

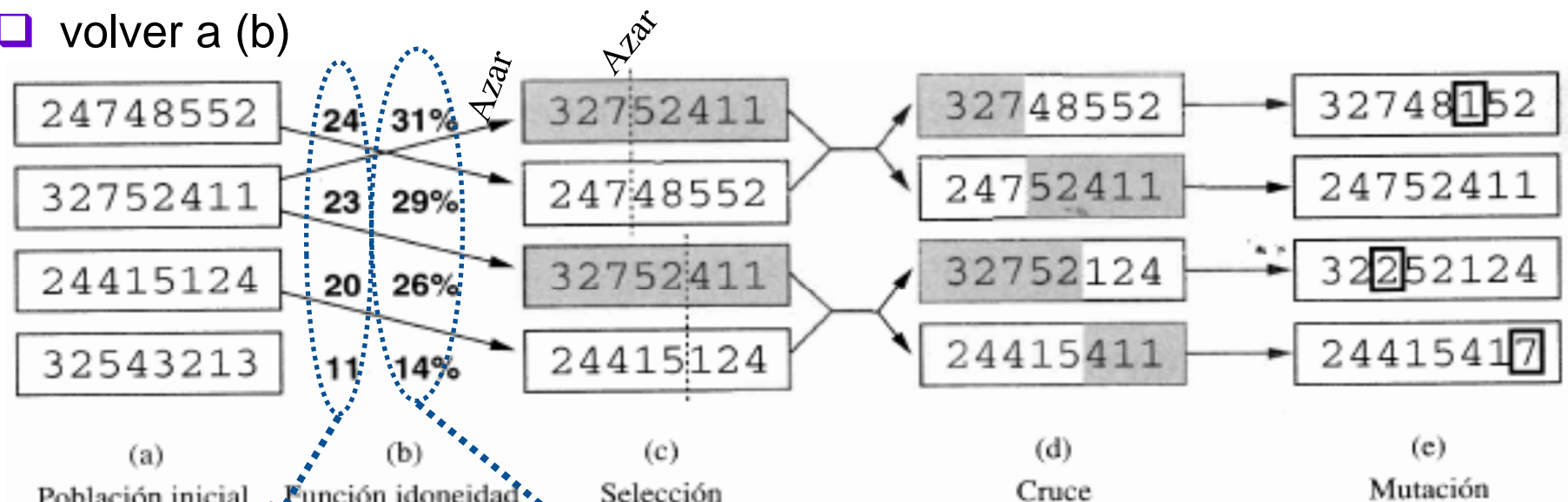
Algoritmos genéticos : Ejemplo 8 reinas

- ❑ Población (a) : Combinaciones de 8 reinas en un tablero
 - ❑ Cada gen es una reina
 - ❑ Cada cromosoma o individuo es una combinación de 8 reinas
- ❑ Codificación Individuo: (*cromosoma*) *Dos posibles modos*
 - A) Lista de 8 dígitos con valor de 1 al 8 : (1,2,3,4,5,6,7,8)
 - ❑ Su posición en la lista es la columna
 - ❑ Su valor es la fila
 - B) Cadena de 1's y 0's . Total = $8 * 4 = 32$ bits



Algoritmos genéticos : Ejemplo 8 reinas

- ❑ Función de idoneidad o fitness (b):
 - ❑ Número de parejas de reinas que NO se atacan (max = solución 28)
- ❑ Con esa función se determina la probabilidad de ser escogido
- ❑ Selección: (c) se selecciona una pareja
- ❑ Cruce: (d) intercambio genes en la pareja con proporción azar (0.7)
- ❑ Mutación: (e) mover una reina cualquiera a una fila cualquiera,
 - ❑ Al azar (ratio de mutación 0.1)
- ❑ volver a (b)



fitness

Total=78

Probabilidad de ser escogido ese cromosoma
para reproducirse $24 / 78 = 0.31$

Algoritmos genéticos : uso y limitaciones

- ❑ Para problemas de optimización:
 - ❑ Diseño de circuitos, asignación de horarios, funciones matemáticas
- ❑ No requieren conocimiento profundo del dominio
 - ❑ Aunque mejoran rendimiento si se añade ese conocimiento:
 - ❑ Restricciones sobre el dominio para escoger cromosomas mejores:
 - ❑ En la primera generación o/y en las posteriores
- ❑ Algoritmo
 - ❑ Búsqueda en paralelo por el espacio de conceptos,
 - ❑ Donde cada línea de proceso realiza un algoritmo de escalada
- ❑ Limitaciones
 - ❑ Optimos locales: colocar individuos en distintos puntos alejados
 - ❑ Impiden localización diversificada de soluciones
 - ❑ en problemas que lo requieren

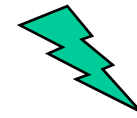
Algoritmos genéticos: demos

- ❑ Demo: dada una superficie con círculos de distintos radios, encontrar sitio para otro círculo de radio lo más grande posible
 - ❑ Genes: centro y radio
 - ❑ Fitness: tamaño del círculo



<http://www.AI-junkie.com>

- ❑ Dado un pequeño circuito, saber qué hacer en cada momento
 - ❑ Un gen por cada sector, que indica qué movimiento hacer
 - ❑ Fitness: cuánto de lejos hemos llegado



Buckland, M. "Building Better Algorithms", AI Wisdom Programming 2
(*en la biblioteca; código fuente y ejecutable en el CD*)

- ❑ Genetic Algorithm Viewer 1.0 GAV

<http://www.rennard.org/alife/english/gavgb.html>

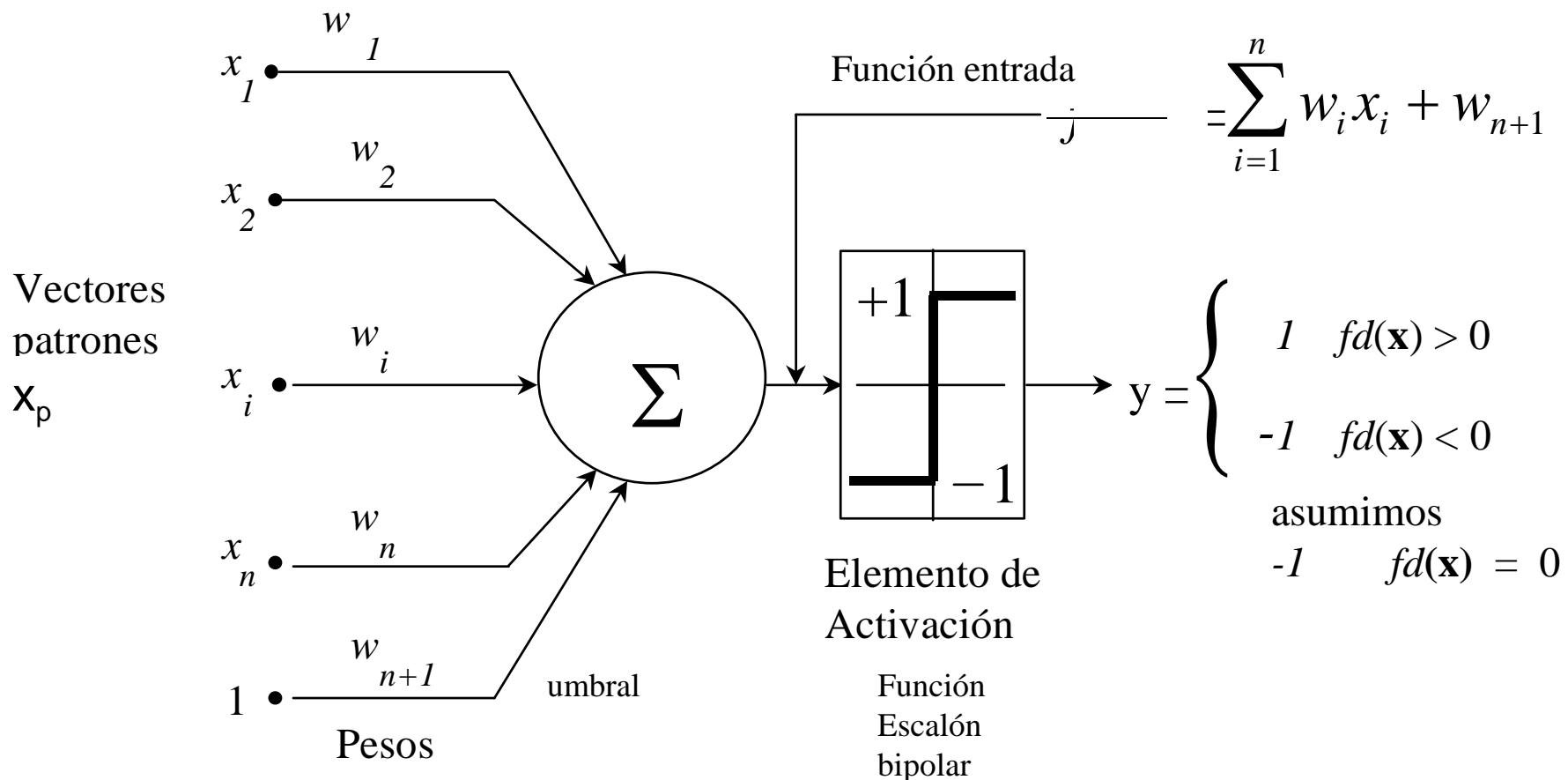
LIBRO: Algoritmos Evolutivos : Un enfoque práctico (L.Araujo, C. Cervigón, RA-MA)

Tipos de aprendizaje: según el paradigma utilizado

- ☐ Aprendizaje inductivo
- ☐ Aprendizaje analítico
- ☐ Algoritmos genéticos
- ☐ **Métodos conexionistas (enfoque subsimbólico)**
 - ☐ **Redes Neuronales**

Métodos conexionistas - Redes Neuronales

- ❑ Simulan modelos biológicos de neuronas
- ❑ Procesan señales que entran (valores no codificados ni símbolos)
- ❑ Propagan salida a otras neuronas



Redes Neuronales : Componentes de Neuronas

- ❑ Entradas x_i : señales del exterior (ejemplos x_p) o de otras neuronas
- ❑ Pesos w_i : importancia de la entradas x_i
- ❑ Umbral u : (bias, sesgo)
 - ❑ Para normalizar el valor con el que se activa la neurona
 - ❑ Es otra entrada $x_{n+1} = 1$ con peso w_{n+1} aleatorio inicial
- ❑ Objetivo : combinación “mejor” de pesos
 - ❑ Búsqueda en un espacio de pesos
 - ❑ Se obtiene entrenando la neurona con ejemplos (se sabe si son + o -)
 - ❑ Compromiso entre: tiempo, error y nivel de generalización
- ❑ Hablaremos del Perceptrón
 - ❑ Todas las señales de entrada y salidas son 0 ó 1
 - ❑ No hay unidades (neuronas) ocultas
 - ❑ Funciones de activación usadas :
 - ❑ escalón o escalón bipolar (esta última en transpa anterior)

Redes Neuronales : Componentes de Neuronas

□ Ejemplo entrenamiento *<patrón entrada, salida esperada>*

□ Patrón de entrada $\mathbf{x_p} = \{x_1, x_2, \dots, x_n\}$

□ Salida esperada $\mathbf{fd(k)}$ (es un dato)

1 si el patrón pertenece a la clase A

-1 si es de la clase B

□ Función de entrada: suma ponderada de entradas de

$$\sum_{i=1}^n w_i x_i + w_{n+1}$$

□ Función de activación \mathbf{f} : (discriminante)

□ Decide cuando activar la neurona basada en la función de entrada

□ Varios tipos: escalón, sigmoides y otras


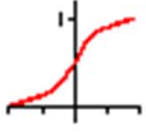
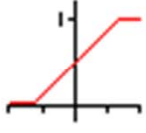
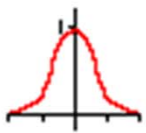
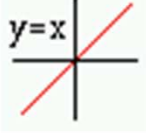
□ Salida observada \mathbf{y} : es lo que devuelve \mathbf{f} : $\{0, 1\}$ ó $\{-1, 1\}$

□ Función activación aplicada a la suma ponderada de entradas de $\mathbf{x_p}$

$$\mathbf{y} = \mathbf{f} \left(\sum_{i=1}^n w_i x_i + w_{n+1} \right)$$

Redes Neuronales : Funciones de Activación $f(x)$

- La x representa el valor obtenido de la función de entrada

Unit Step		$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$
Sigmoid		$f(x) = \frac{1}{1+e^{-\beta x}}$
Piecewise Linear		$f(x) = \begin{cases} 0 & \text{if } x \leq x_{min} \\ mx+b & \text{if } x_{max} > x > x_{min} \\ 1 & \text{if } x \geq x_{max} \end{cases}$
Gaussian		$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
Identity		$f(x) = x$

Redes Neuronales : Regla de Aprendizaje

- ❑ Cómo se obtienen los *pesos nuevos* (Rossenblatt)
- ❑ α : tasa de aprendizaje (es dato)
 - ❑ [0 .. 1] Velocidad de aprendizaje,
 - ❑ Aumenta acercándose a 1
- ❑ Cálculo (k es el paso en el que se recalcula el valor)
 - ❑ *Error = salida esperada – salida observada = [fd(k) - y(k)]*
 - ❑ $w_{NUEVO} = w_{ANTIGUO} + \alpha * \text{Error} * \text{entrada}_i$
 - ❑ $w_i(k+1) = w_i(k) + \alpha(k) * \text{Error} * x_i$
- ❑ También se actualiza igual el umbral **u** (es w_{n+1} ó w_0)
 - ❑ $u(k+1) = u(k) + \alpha(k) * \text{Error}$

$$\sum_{i=1}^n w_i x_i + w_{n+1}$$

F. entrada

Redes Neuronales : *Algoritmo*

1.- Inicialización de los pesos y del umbral:

asignar valores aleatorios a cada pesos $w_i, i = 1, 2, \dots, n$ y al umbral w_{n+1}

2.- Presentación un ejemplo con el par <Entrada, Salida esperada>

$$< \mathbf{x}^p = \{x_1, x_2, \dots, x_n\}, \quad \mathbf{fd}(k) >$$

3.- Cálculo de la salida actual: (incluye w_{n+1} como otra entrada más)

$$y(k) = f[\mathbf{w}^t \mathbf{x}^p], \quad (f: \text{escalón en este caso})$$

4.- Adaptación de los pesos:

❑ $\text{Error} = [\mathbf{fd}(k) - y(k)]$

❑ Si el error es 0 no se adaptan los pesos

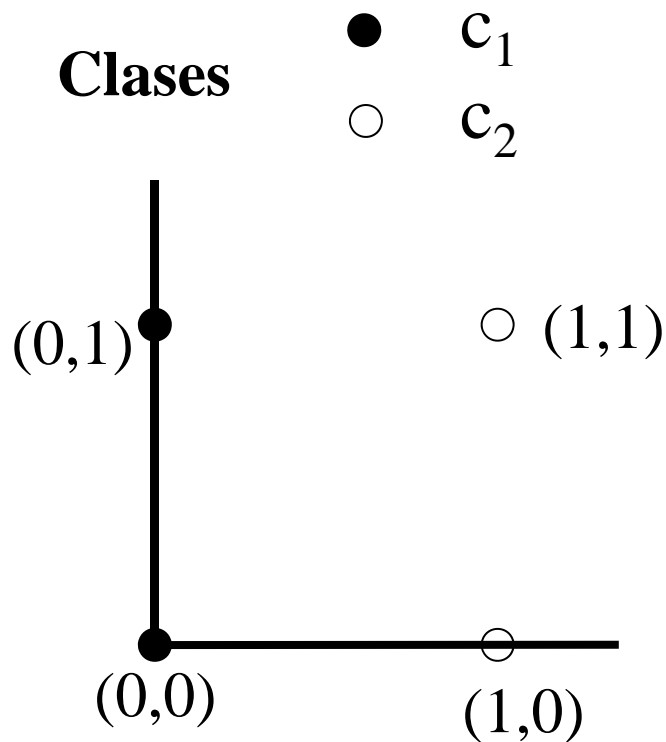
❑ $w_i(k+1) = w_i(k) + \alpha(k) * \text{Error} * x_i$ (Rosenblatt)

5.- Si no hay convergencia (pesos han cambiado) volver al paso 2

(repite hasta que los pesos no cambien en **ningún** ejemplo)

Redes Neuronales : *Algoritmo , Ejemplo*

Aplicar el algoritmo al siguiente ejemplo biclase (asumir $\alpha = \frac{1}{2}$)



Pesos iniciales: $w_1 = w_2 = w_3 = 0$

Vectores patrón aumentados:

$$c_1: \begin{aligned} x_1 &= (0,0,1)^t \\ x_2 &= (0,1,1)^t \end{aligned}$$

Salida esperada
 $fd_i = 1$

$$c_2: \begin{aligned} x_3 &= (1,0,1)^t \\ x_4 &= (1,1,1)^t \end{aligned}$$

$fd_i = -1$

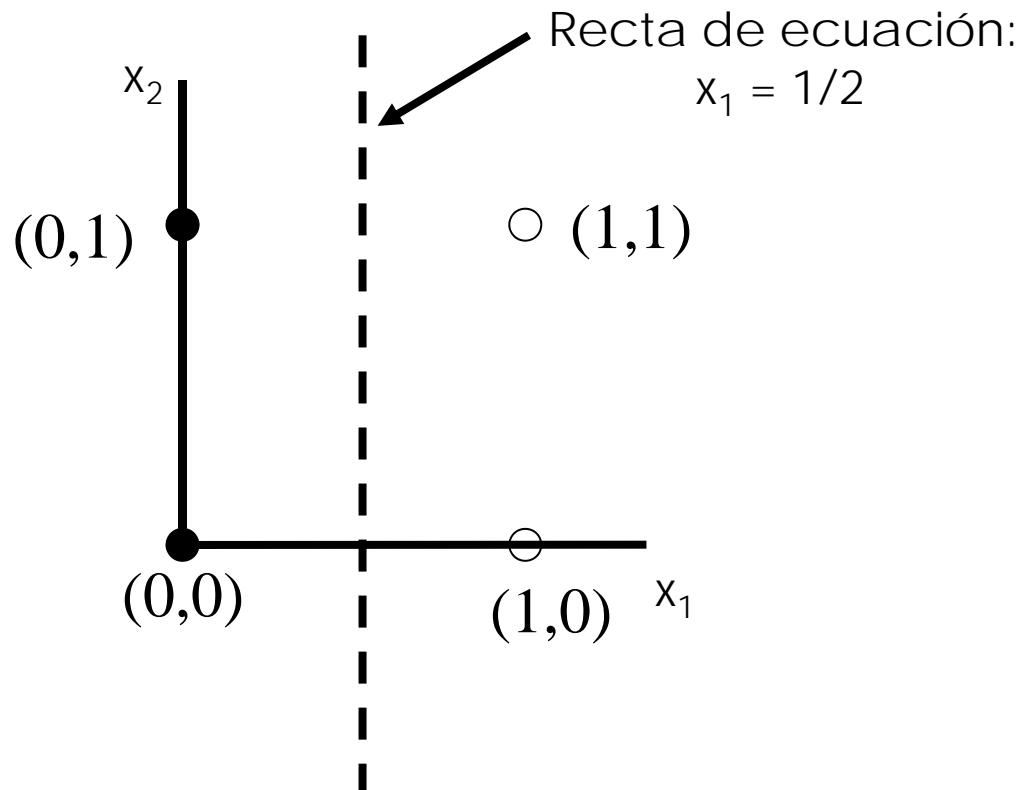
Usar F. Activación Bipolar: 1 para $x > 0$,
-1 para el resto

Redes Neuronales : *Algoritmo, Ejemplo*

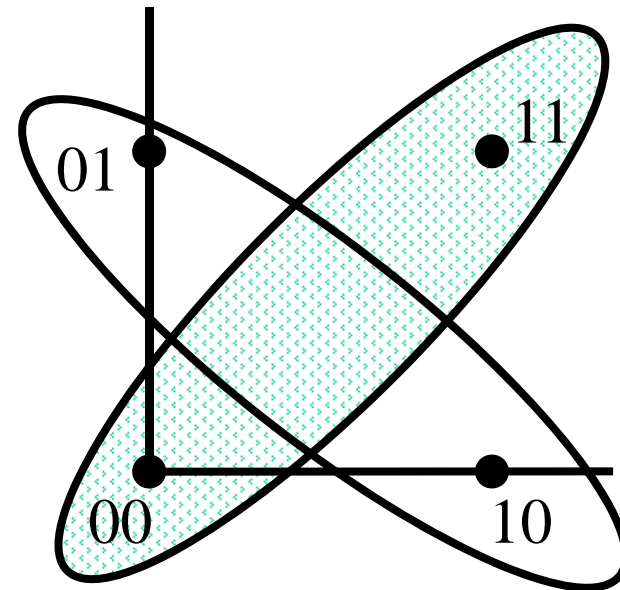
- **1. Inicialización:** $\mathbf{w}^t(1) = (0,0,0)$; $\alpha = 0.5$
- **3.1 Patrón $\mathbf{x} = \{0,0,1\}$:** $\mathbf{w}^t \mathbf{x} = 0(0)+0(0)+0(1)=0$; $y = f(0) = -1$,
error = (fdi - y)=1+1= 2
4.1 Pesos modificados: $\mathbf{w}^t(2) = \mathbf{w}^t(1) + \mathbf{x} = (0,0,1)$
- **3.2 Patrón $\mathbf{x} = \{0,1,1\}$:** $\mathbf{w}^t \mathbf{x} = 1$; $y = 1$, error = (fdi - y)=1 -1= 0 \Rightarrow
4.2 Pesos *no* modificados: $\mathbf{w}^t(3) = \mathbf{w}^t(2)$
- **3.3 Patrón $\mathbf{x} = \{1,0,1\}$:** $\mathbf{w}^t \mathbf{x} = 1$; $y = 1$, error = (fdi - y)=-1 - 1= -2
4.3 Pesos modificados: $\mathbf{w}^t(4) = \mathbf{w}^t(3) - \mathbf{x} = (-1,0,0)$
- **3.4 Patrón $\mathbf{x} = \{1,1,1\}$:** $\mathbf{w}^t \mathbf{x} = -1$; $y = -1$, error = (fdi - y)=-1 +1= 0
4.4 Pesos *no* modificados: $\mathbf{w}^t(5) = \mathbf{w}^t(4)$
- **Repetir otra serie de entrenamiento con los mismos patrones:**
Llegando en 10ª iteración a: $\mathbf{w}^t(10) = (-2,0,1)$

Redes Neuronales: Limitación

*Función de decisión
del Ejemplo:*



*...pero No existe recta que
separe estas clases:*



XOR

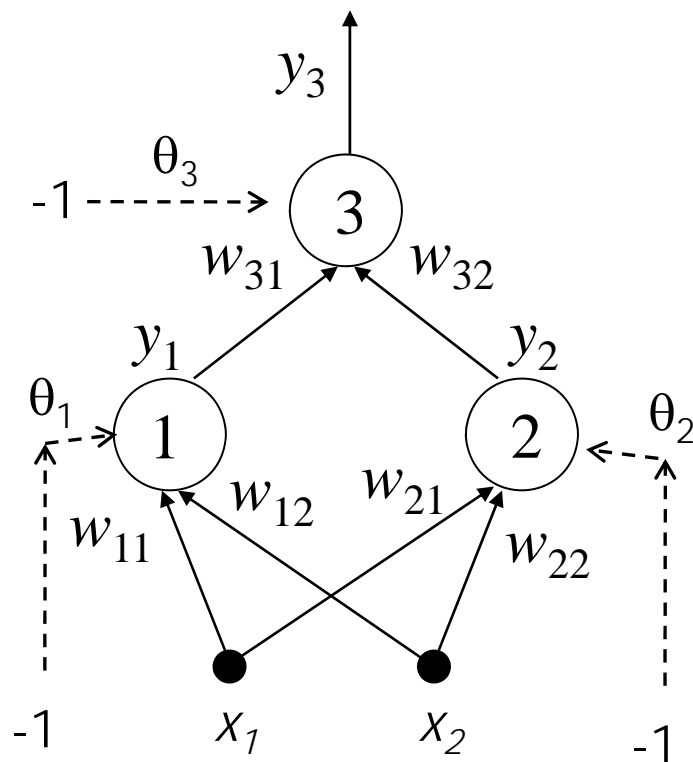
*...necesitamos dos rectas
para separar estas clases*

... y cómo se consiguen?

Tema 5 - 19

Redes Neuronales: Perceptrón Multicapa

Ejemplo: solución para XOR



Pesos

$$w_{11} = w_{12} = w_{21} = w_{22} = w_{31} = 1; w_{32} = -1.5$$

$$\theta_1 = 0.5; \theta_2 = 1.5; \theta_3 = 0.5$$

Salidas de las neuronas

(valores umbrales = -1)

$$y_1 = f(w_{11}x_1 + w_{12}x_2 - \theta_1) = f(x_1 + x_2 - 0.5)$$

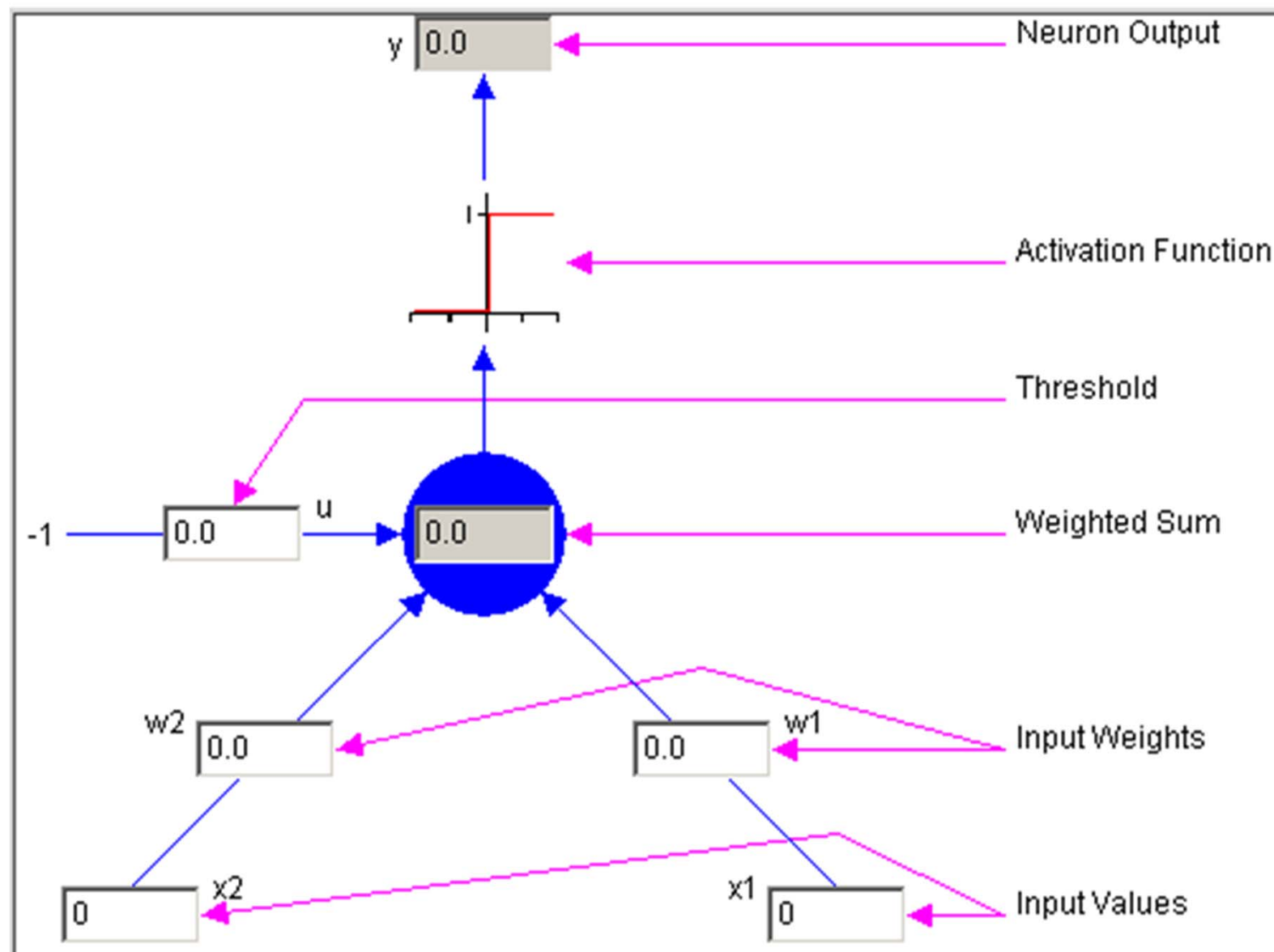
$$y_2 = f(w_{21}x_1 + w_{22}x_2 - \theta_2) = f(x_1 + x_2 - 1.5)$$

$$y_3 = f(w_{31}y_1 + w_{32}y_2 - \theta_3) = f(y_1 - 1.5y_2 - 0.5)$$

Resultados

x_1	x_2	y_1	y_2	$y_3 = \text{XOR}$
0	0	$f(-0.5) = 0$	$f(-1.5) = 0$	$f(-0.5) = 0$
0	1	$f(0.5) = 1$	$f(-0.5) = 0$	$f(0.5) = 1$
1	0	$f(0.5) = 1$	$f(-0.5) = 0$	$f(0.5) = 1$
1	1	$f(1.5) = 1$	$f(0.5) = 1$	$f(-1.0) = 0$

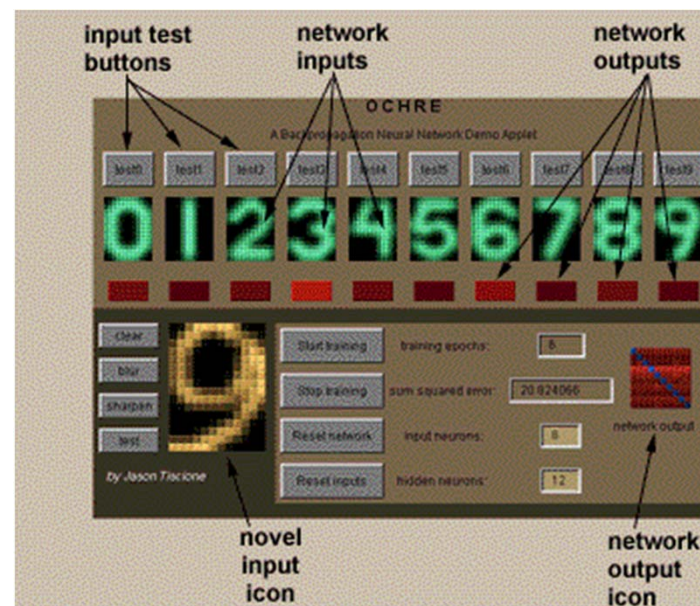
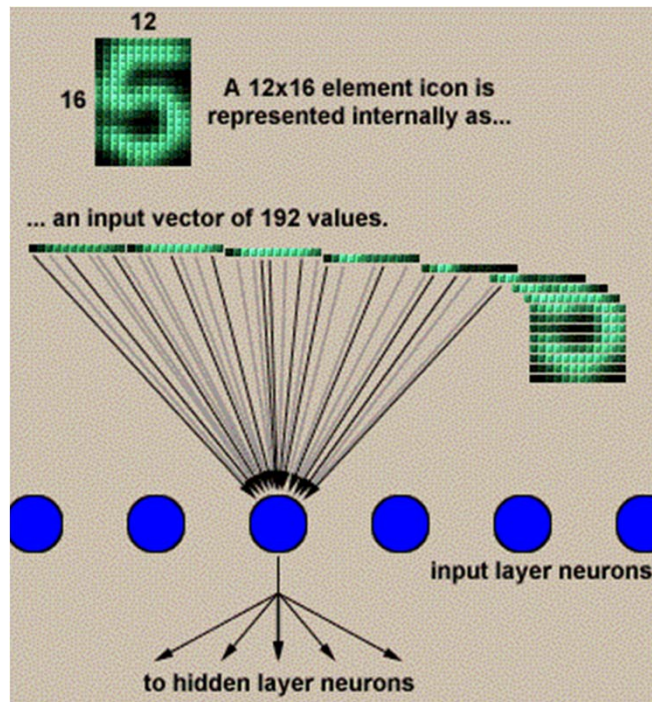
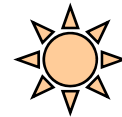
Redes Neuronales: Applet de una Neurona



Redes Neuronales : Aplicaciones

- ❑ Reconoce números escritos a mano, 12 x 16 pixels = 193 entradas
- ❑ Neuronas: 8 en capa de entrada, 12 en capa oculta y 10 en la de salida

<http://sund.de/netze/applets/BPN/bpn2/ochre.html>



LIBRO: Redes Neuronales Artificiales (J.R.Hilera, V.J. Martínez) Ra-Ma

Uso de redes neuronales: Aplicaciones

- ❑ Muy buenos resultados en problemas en los que los datos de entrenamientos proceden de sensores complejos que contienen ruido (información de cámaras o micrófonos)
- ❑ ALVINN: sistema de conducción automático

