Redes neuronales frente a otras "técnicas"

Las redes neuronales tienen gran potencial en áreas como el reconocimiento de voz e imágenes, en las cuales se persiguen muchas hipótesis en paralelo. A diferencia de la arquitectura Von Neumann (proceso secuencial de instrucciones), las NN exploran varias hipótesis a la vez aprovechando el paralelismo inherente de la red.

Las NN son + robustas (tolerantes ~ fallos). Las NN se adaptan con el tiempo ("aprenden"). Esta cont. es clave para aplicar problemas en los q. aparecen nuevos datos constantes, muchas veces con variaciones no significativas. En cambio, los métodos estadísticos no son adaptativos: procesan todos los datos de una vez.

Las NN son más robustas cuando los datos provienen de procesos no lineales y no Gaussianos.

Las NN tienen múltiples aplicaciones:

- determinar la clase de un patrón corrupto o ruido
- mem. asociativa
- cuantización / clusterización
- optimización

una vez aprovechando el paralelismo inherente de la red.

Las NN son + robustas (tolerantes ~ fallos). Las NN se adaptan con el tiempo ("aprenden"). Esta cont. es clave para aplicar problemas en los q. aparecen nuevos datos constantes, muchas veces con variaciones no significativas. En cambio, los métodos estadísticos no son adaptativos: procesan todos los datos de una vez.

Las redes neuronales artificiales se inspiran en las biológicas, pero eso no debe inducirnos a pensar q. las NN biológicas funcionan como las artificiales. He aquí algunas diferencias:

- las NN biológicas no aplican los ppos de los circuitos digitales, es decir, son analógicas. Las señales biológicas no son asíncronas (duración variable) ni síncronas (reloj central), ni tienen un umbral preciso y cte.
- Ni las neuronas ni las sinapsis son elementos de mem. biestables: no cambian de un estado a otro.
- no hay "instrucciones máquina" ni "códigos de control"
- los circuitos cerebrales no implementan la recursividad
- las NN^{bio} son asíncronas (en el sentido d. q. la cd se reconfigura constantemente), mientras q. las artificiales los pesos se actualizan de forma periódica.
- las bio aprenden con pocos ejs., mientras las artificiales convergen lentamente

- Ni las neuronas ni las sinapsis son elementos de mem. biestables: no cambian de un estado a otro.
- no hay "instrucciones máquina" ni "códigos de control"
- los circuitos cerebrales no implementan la recursividad
- las NN^{bio} son asíncronas (en el sentido d. q. la cd se reconfigura constantemente), mientras q. las artificiales los pesos se actualizan de forma periódica.

REDES NEURONALES

Una red neuronal artificial es un sistema de computación desarrollado como una generaliz. de los modelos matemáticos del conocimiento humano (neurobiología), por eso presenta muchas similitudes con las redes neuronales biológicas:

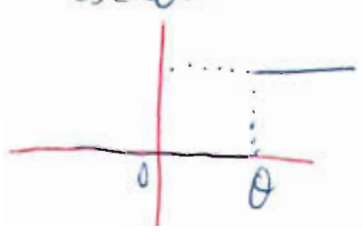
- red formada por muchas unids. de proceso simples, q. tratan la info. localm (paralelismo masivo)
- cada neurona recibe muchas señales de entrada
- cada señal puede ser modificada por un peso en la sinapsis
- la neurona combina las señales de entrada ponderadas para producir una señal de salida, q. a su vez puede propagarse a una o varias neuronas
- la mem. es distribuida: a largo plazo reside en los pesos, a corto plazo en las señales
- los pesos pueden alterarse por la experiencia
- los neurotransmisores pueden ser excitadores o inhibidores

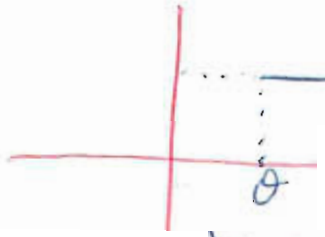
des con las redes neuronales biológicas:

- red formada por muchas unids. de proceso simples, q. tratan la info. localm (paralelismo masivo)
- cada neurona recibe muchas señales de entrada
- cada señal puede ser modificada por un peso en la sinapsis
- la neurona combina las señales de

Las redes neuronales artificiales (NN) se caracterizan por

- arquitectura → nº de capas → monocapa / multicapa
  - interconexiones → flujo directo (feed forward) → sólo ∃ conexiones de la capa N a la N+1
  - recurrentes / iterativas (feedback) con retroalimentación → ∃ conexiones entre N y N+1 y entre N+1 y N
- algoritmos de aprendizaje → supervisado / no supervisado
- algoritmos ← iterativos / recurrentes: van refinando la sol.
  secuenciales: cada paso se aplica 1 sóla vez
- func. de activación: sirven para restringir el rango de valores de una señal y para introducir la "no linealidad" en la red (ya q. una combinación de funcs. lineales es lineal, y puede ser insuficiente para q. el modelo sea fiel)
  - identidad: $f(y_j) = y_{en_j} = y_j$
  - escalón



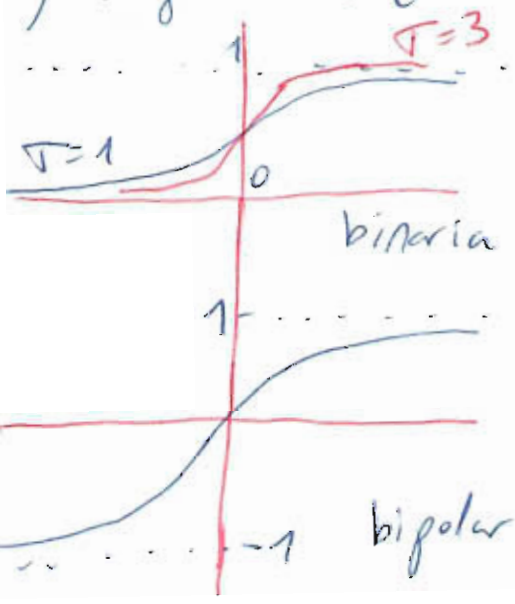


θ = umbral

↓ recurrentes / iterativas (feedback) con retroalimentación → ∃ conexiones entre N y N+1 y entre N+1 y N

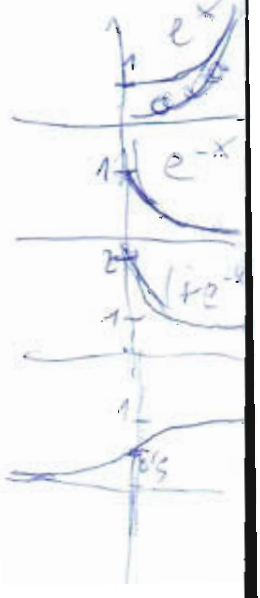
- algoritmos de aprendizaje → supervisado / no supervisado
- algoritmos ← iterativos / recurrentes: van refinando la sol.
  secuenciales: cada paso se aplica 1 sóla vez
- func. de activación: sirven para restringir el rango de valores de una señal y para introducir la "no linealidad" en la red (ya

- sigmoidal: suele utilizarse pq. es acotada y diferenciable



$$f(x) = \frac{1}{1+e^{-\tau x}}$$

$$g(x) = 2f(x) - 1 = \frac{1-e^{-\tau x}}{1+e^{\tau x}}$$



Algunas arquitecturas emplean la derivada de la func. de activación:

$$[f'(x)] \Rightarrow \frac{d}{dx} u^n = n u^{n-1} \frac{du}{dx} \qquad u = 1+e^{-\tau x} \quad \frac{du}{dx} = -\tau e^{-\tau x} \quad n = -1$$

$$\hookrightarrow -1 \cdot (1+e^{-\tau x})^{-2} \cdot (-\tau e^{-\tau x}) = \frac{\tau e^{-\tau x}}{(1+e^{-\tau x})^2}$$

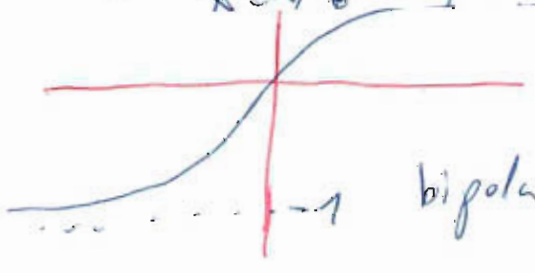
La derivada se puede expresar a partir de la propia $f(x)$:

$$f'(x) = \frac{\tau e^{-\tau x}}{(1+e^{-\tau x})^2} = \frac{1}{(1+e^{-\tau x})} \frac{e^{-\tau x}}{(1+e^{-\tau x})} \cdot \tau = f(x) \cdot (1-f(x)) \cdot \tau$$

Normalo, $\tau = 1$, por lo q. $\boxed{f'(x) = f(x)[1-f(x)]}$

$$[g'(x)] \Rightarrow \frac{dy}{dx} = \frac{dy}{du}$$

$$g(x) = 2[f(x)]^2 - 1 = \frac{1-e^{-\tau x}}{1+e^{\tau x}}$$



Algunas arquitecturas emplean la derivada de la func. de activación:

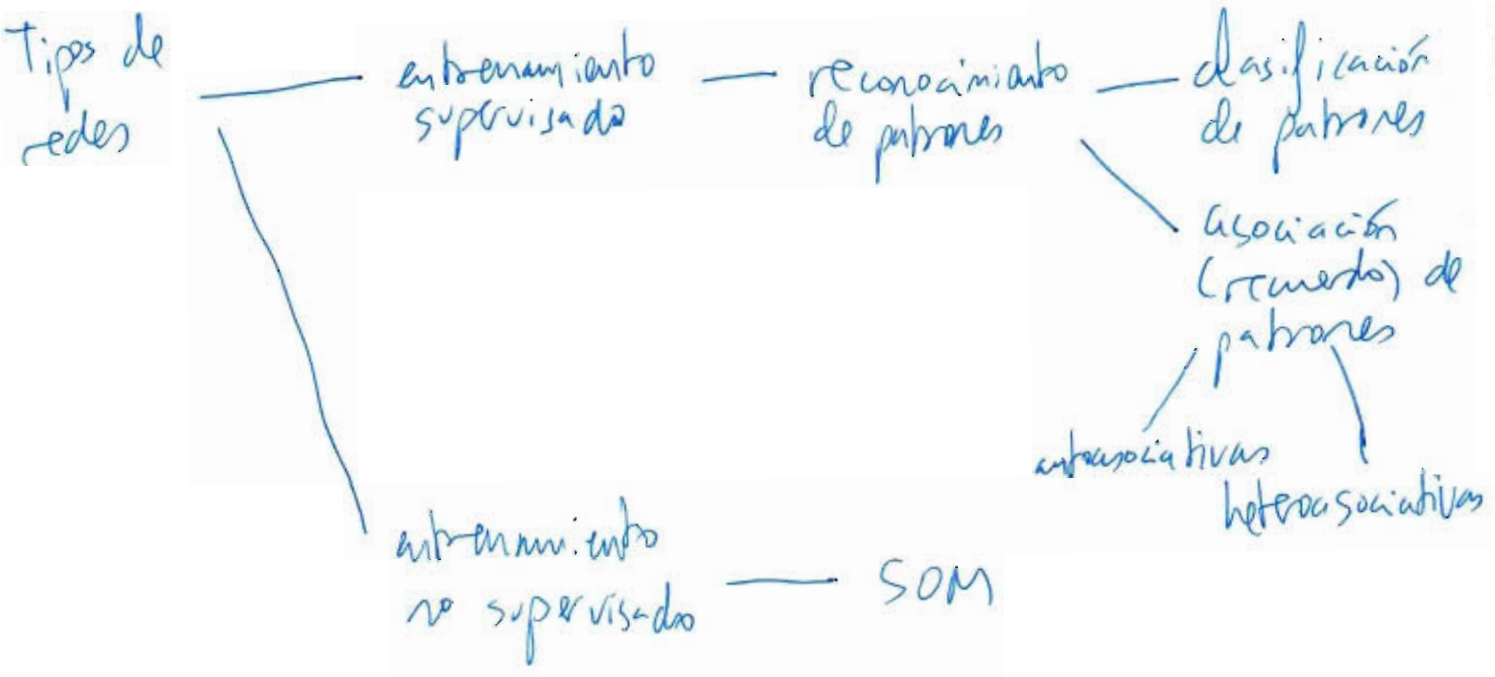
$$[f'(x)] \Rightarrow \frac{d}{dx} u^n = n u^{n-1} \frac{du}{dx} \qquad u = 1+e^{-\tau x} \quad \frac{du}{dx} = -\tau e^{-\tau x} \quad n = -1$$

Repaso

Clasificación redes

- Mc Culloch - Pitts
- Aprendizaje no supervisado
  - Feedback[1] (flujo total)
    - ART
    - Hopfield
    - BAM
    - SOM
    - Aprendizaje competición: Hamming Maxnet, sombrero mejicano,
  - Feedforward-only[2] (flujo directo)
    - Contrapropagación (CP)
- Aprendizaje supervisado
  - Feedback[1]
    - BSB, hetero/autoasociativas
  - Feedforward-only[2]
    - Perceptron
    - Adaline / Madaline
    - Retropropagación (BP)
    - LVQ
    - Hebb

- SOM
- Aprendizaje competición: Hamming Maxnet, sombrero mejicano,
- Feedforward-only[2] (flujo directo)
  - Contrapropagación (CP)
- Aprendizaje supervisado
  - Feedback[1]
    - BSB, hetero/autoasociativas
  - Feedforward-only[2]



Conjunto entrenamiento ⇄ ajuste pesos / comprobar aprendizaje

Conjunto de prueba ⇄ verificar aprendizaje independiente del de entrenamiento

Ambos deben ser significativos (nº suficiente ejs.) y representativos (diversos)

Tipos entrada → clases

- Clasif. de patrones: reconocimiento: cada patrón (vector de entrada) pertenece (o no) a una clase (categoría). Se conocen las clases de los patrones del conjunto de entrenamiento. Puede haber varias clases (lo q. conlleva varias unids. de salida: nº unids. salida = nº clases salida)
  En estas redes el bias y el umbral son equivalentes, por lo no aporta nada incluir ambos. Sin embargo, no

Conjunto entrenamiento ⇄ ajuste pesos / comprobar aprendizaje

Conjunto de prueba ⇄ verificar aprendizaje independiente del de entrenamiento

Ambos deben ser significativos (nº suficiente ejs.) y representativos

lineal de func. es lineal)

→ Mapeo patrón entrada → patrón salida

- Asoc. de patrones: la red almacena un conjunto de asociaciones entre patrones de entrada (s) y salida (t). Si $s=t$, la red es autoasociativa. Si $s \neq t$, heteroasociativa.

Cuando se usa la regla de Hebb, suelen normalizarse los pesos finales con un factor $1/n$ (n = nº unids. de la red)

- Completación: sólo una de las neuronas gana

- Autoorganizativas: la unid. cuyo peso esté más próximo al vector de entrada es la q. se actualiza. Para calcular la proximidad se puede usar el cuadrado de la distancia euclídea o el producto vectorial (correlación)

Solución problemas no lineales separables

- Cambio codif. patrones (p. e., añadir una nueva entrada q. ayude a separar los patrones)

Cond. de parada

- nº fijo de épocas
- el error desciende por debajo de un umbral
- la variación en los pesos es irrelevante

- Completación: sólo una de las neuronas gana

- Autoorganizativas: la unid. cuyo peso esté más próximo al vector de entrada es la q. se actualiza. Para calcular la proximidad se puede usar el cuadrado de la distancia euclídea o el producto vectorial (correlación)

REGLAS DE APRENDIZAJE

Regla de Hebb (Donald Hebb, 1949)

"Cuando un axón de una célula A está lo suficientemente cerca para excitar una célula B, y toma parte repetidas o en el proceso de disparo de dicha célula, se produce algún tipo de cambio metabólico en una de las células (o en las 2) q. hace q. la eficacia con la q. A dispara a B se vea incrementada"

Es decir, cuando 2 neuronas se activan repetidas al mismo tiempo, tienden a asociarse, de modo q. la actividad en una incita la actividad de la otra. En pocas palabras, se refuerzan las conexiones de las neuronas q. se activan simultáneas.

Hebb ponía como ej. de alteración en la sinapsis el aumento del nº de terminales sinápticos o su alargamiento.

La regla de Hebb como tal es bastante genérica y ha dado pie a numerosas formulaciones. La más simple es:

[1] $w_{ij} = x_i \cdot x_j$ — binaria → $0 \cdot 0 = 0$ (inhibición) — 2 neuronas se inhiben simultáneas

$0 \cdot 1 = 0$ "

dice algún tipo de cambio metabólico en una de las células (o en las 2) q. hace q. la eficacia con la q. A dispara a B se vea incrementada"

Es decir, cuando 2 neuronas se activan repetidas al mismo tiempo, tienden a asociarse, de modo q. la actividad en una incita la actividad de la otra. En pocas

Eso, cuando el refuerzo también se produce cuando varias neuronas se desactivan a la vez, se habla de regla de Hebb extendida.

Una formulación + biológica sería:

$$w(t+1) = w(t) + (apr \cdot apos) \quad [2]$$

(activación presináptica / postsináptica)

La formulación + habitual hoy en día es

$$\Delta w_{ij} = x_i y_j \quad [3]$$

~~Si también se puede plantear~~

En las redes asociativas se aplica la fórmula [1] vectorialmente:

$$w = \bar{x}^T \cdot \bar{y} \quad [4]$$

En el perceptrón se aplica una regla equivalente (cuando la salida obtenida no coincide con la esperada):

$$\Delta w_i = \alpha t x_i$$

$$w(t+1) = w(t) + (apr \cdot apos) \quad [5]$$

(activación presináptica / postsináptica)

La formulación + habitual hoy en día es

$$\Delta w_{ij} = x_i y_j \quad [3]$$

~~Si también se puede plantear~~

Regla Delta: busca minimizar la diferencia entre el valor real producido en la capa de salida de la red y el valor esperado u objetivo:

$$\Delta w_{ij} = \alpha (t_j - y_{in_j}) x_i$$

$t \rightarrow$ valor esperado

$y_{in} \rightarrow$ valor obtenido (antes de aplicar la func. de activación)

$x_i \rightarrow$ valor de entrada

$\alpha \rightarrow$ tasa de aprendizaje

- grande $\rightarrow$ convergencia rápida, con riesgo de perder mínimos u oscilar entre ellos
- peq. $\rightarrow$ convergencia lenta y segura

La regla delta minimiza la func. de error. Para acelerar el proceso se puede recurrir a técnicas de optimización no lineal. En el caso del perceptrón, suele utilizarse el método de descenso del gradiente.

$t \rightarrow$ valor esperado

$y_{in} \rightarrow$ valor obtenido (antes de aplicar la func. de activación)

$x_i \rightarrow$ valor de entrada

$\alpha \rightarrow$ tasa de aprendizaje

- grande $\rightarrow$ convergencia rápida, con riesgo de perder mínimos u oscilar entre ellos

Aprendizaje Kohonen

$$\Delta w_{ij}(t) = \alpha (x_i - w_{ij}(t-1))$$

- Tipos de ~~entradas~~ datos
  - bipolares: -1, 1. Permiten distinguir los datos omitidos (con un 0) de los erroneos (cambio de signo)
  - binarios: 0, 1
  - continuos

Aprendizaje Kohonen $\Delta w_j = \alpha (x - w_j old)$

equivale a Hebb, salvo 4.

Regla aprendizaje perceptrón (se aplica sólo cuando hay fallos)

$w_i(nuevo) = w_i(old) + \alpha t x_i$

t = valor esperado
x_i = entrada
α = tasa aprendizaje

→ regla delta generalizada

- FUNCS. DE ACTIVACIÓN

Hebb
- red de Hebb (B)
- heteroasociativa (E)
- autoasociativa (E)
- BSB (I-E)
- Hopfield (E)
- BAM (E)

Perceptron
- Perceptron (E)(B)

Kohonen
- LVQ
- SOM

Delta
- Adaline (I,E) (B)
- Madaline (E) (B)
- (heteroasociativa)
- (autoasociativa)
- (contrapropagación)
- otras propagación (S) (B)

Otros
- Maxnet (I)
- Sombrero mej. como (I-E)
- Hamming (B)
- ART
- McCulloch-Pitts (E)

equivale a Hebb, salvo 4.

Regla aprendizaje perceptrón (se aplica sólo cuando hay fallos)

$w_i(nuevo) = w_i(old) + \alpha t x_i$

t = valor esperado
x_i = entrada
α = tasa aprendizaje

→ regla delta generalizada

- FUNCS. DE ACTIVACIÓN

Hebb
- red de Hebb (B)
- heteroasociativa (E)

Delta
- Adaline (I,E) (B)
- Madaline (E) (B)

REGLAS DE APRENDIZAJE

Limitaciones: codif. bipolar E/S & separabilidad lineal. Nº entradas limitado

$\Delta w_i = x_i y$

Regla de Hebb (1949): → mecanismo local y dependiente del tiempo

Regla de Hebb: se refuerzan las conexiones de las neuronas q. se activan simultáneamente. No se usa en la actualidad. "Cuando un axón de la célula A está lo bastante próximo a la célula B como para excitarla, y de forma repetida o persistente colabora en su activación, tienen lugar algunos procesos de crecimiento o cambios metabólicos en una o ambas células, tal q. la eficiencia de A, como

(1988) Regla de Hebb extendida: también se refuerzan las conexiones cuando varias neuronas se desactivan a la vez. Dependiendo de la codif. se puede llegar a unos pesos q. separan linealm (o no) los datos, por eso a veces se cambia de codif.

$w_{ij}(new) = w_{ij}(old) + x_i y_j$ ($\bar{x} \cdot \bar{y}$ = correlación)

Si los vectores x_i no son ortogonales, la respuesta de la neurona incluirá interferencias

Regla Delta: minimiza la diferencia entre la entrada neta de la unid. de salida (y_in) y el valor objetivo (t) (o "mínimos cuadrados")

$w_i(new) = w_i(old) + \alpha (t - y_{in}) x_i$

* α grande → convergencia rápida pero con riesgo de perder mínimos
* α peq. → convergencia lenta pero segura

$y = \sum x_i w_i$

La regla de Hebb se puede calcular en varios pasos (tantos como patrones a almacenar). Ej. Una de las células q. activa a B, se incrementa

s: P1: (−1, 1, −1, 1, −1, 1) t: (−1, 1)

P2: (−1, 1, 1, 1, −1, −1) (1, 1)

$W_1 = s_1^T \cdot t_1 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix}$ $W_2 = s_2^T t_2 = \begin{pmatrix} -1 & -1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \end{pmatrix}$ $W = W_1 + W_2 = \begin{pmatrix} 0 & -2 \\ 0 & 2 \\ 2 & 0 \\ 0 & 2 \\ 0 & -2 \\ -2 & 0 \end{pmatrix}$

Regla abierta con varias formulaciones. La + simple es $\Delta w_{ij} = x_i \cdot y_j$

cuando varias neuronas se desactivan a la vez. Dependiendo de la codif. se puede llegar a unos pesos q. separan linealm (o no) los datos, por eso a veces se cambia de codif.

$w_{ij}(new) = w_{ij}(old) + x_i y_j$ ($\bar{x} \cdot \bar{y}$ = correlación)

Si los vectores x_i no son ortogonales, la respuesta de la neurona incluirá interferencias

Regla Delta: minimiza la diferencia entre la entrada neta de la unid. de salida (y_in) y el valor objetivo (t) (o "mínimos cuadrados")

$w_i(new) = w_i(old) + \alpha (t - y_{in}) x_i$

* α grande → convergencia rápida pero con riesgo de perder mínimos
* α peq. → convergencia lenta pero segura

$y = \sum x_i w_i$

Evol. redes neuronales

Redes básicas → McCulloch → Hebb → Perceptrón → Adaline → Madaline

Implementa funciones básicas con NN a partir de valores de ej. Capacidad aprendizaje o pesos fijos. Una vez implementada la red y se puede aplicar a nuevos patrones

- McCulloch → ~~[illegible]~~ pesos fijos, umbral de Lade neuron
- Hebb → aprendizaje (Hebb) (clasif. patrones)
- Perceptrón → tasa aprendizaje, func. activación (umbral), multicapa (opcional), se pasan varias veces los patrones (iteraciones)
- Adaline → Varias salidas, aprendizaje Delta
- Madaline → Capa oculta

Backpropagation
- capa oculta
- propagación atrás

asoc. patrones

Asociativas iterativas → Heteroasociativa → inicializ. pesos Hebb

Se parte de un patrón incompleto y se quiere saber a q. patrón memorizado se parece +

Heteroasociativa → Autoasociativa → Autoasoc. con func. umbral

- Autoasociativa → $x = y$, vectores ortogonales
- Autoasoc. con func. umbral → repetición $x_{in}(i+1) = x(i)W$, θ_i

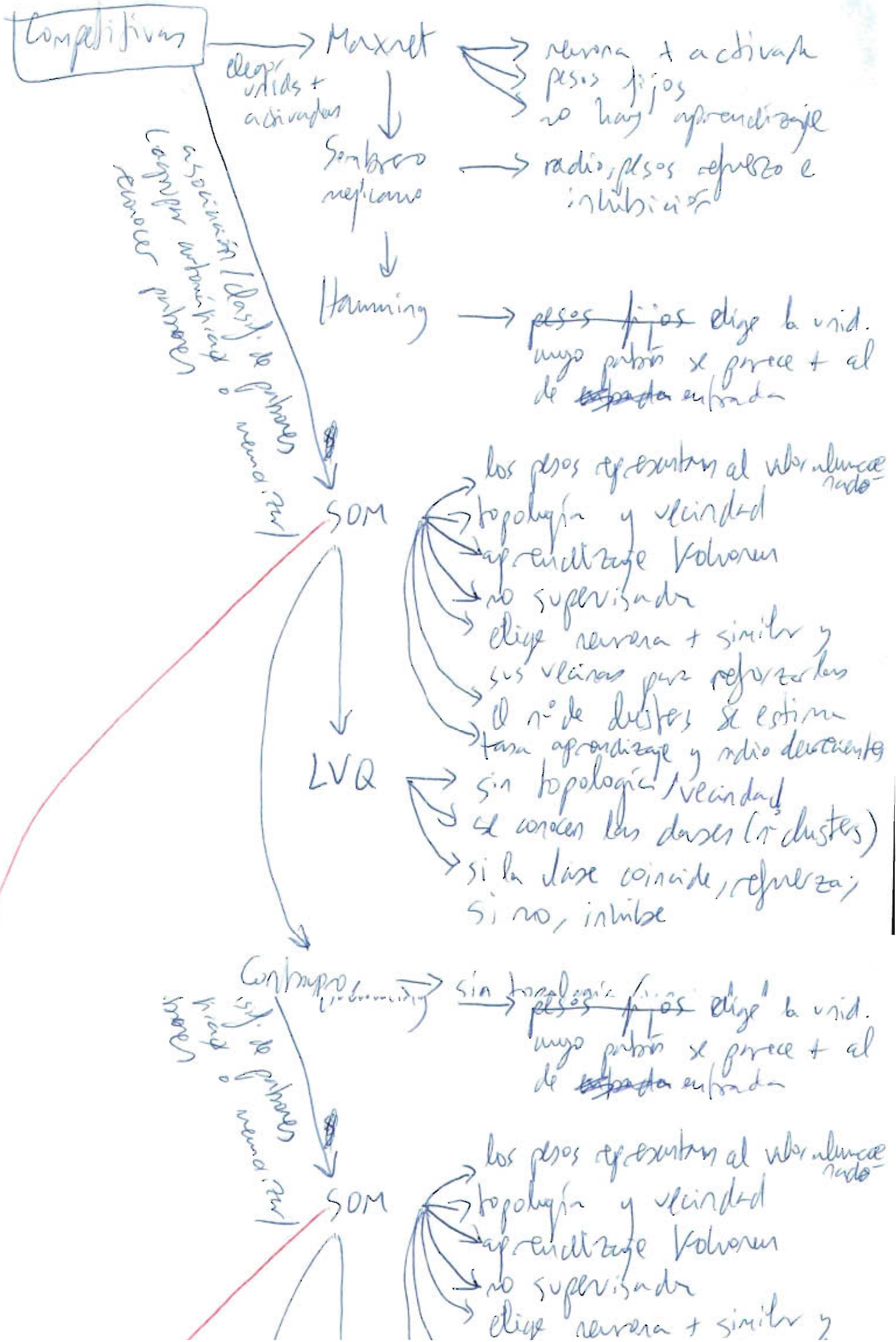
fijos. Una vez implementada la red y se puede aplicar a nuevos patrones

Perceptrón → Adaline → Madaline

- Perceptrón → tasa aprendizaje, func. activación (umbral), multicapa (opcional), se pasan varias veces los patrones (iteraciones)
- Adaline → Varias salidas, aprendizaje Delta
- Madaline → Capa oculta

Backpropagation
- capa oculta
- propagación atrás

asoc. patrones


Competitivas
Maxnet
elegir vrids + activadas
neurona + activada
pesos fijos
no hay aprendizaje
Sombrero mejicano
radio, pesos refuerzo e inhibición
Hamming
pesos fijos elige la vrid. cuyo patrón se parece + al de entrada
asociación / clasif. de patrones (aprender patrones)
SOM
los pesos representan al vector aprendido
topología y vecindad
aprendizaje Kohonen
no supervisado
elige neurona + similar y sus vecinas para reforzarlos
el nº de clusters se estima
tasa aprendizaje y radio decreciente
LVQ
sin topología / vecindad
se conocen las clases (= clusters)
si la clase coincide, refuerza; si no, inhibe
Contrapropagación
sin topología
pesos fijos elige la vrid. cuyo patrón se parece + al de entrada
clasif. de patrones
SOM
los pesos representan al vector aprendido
topología y vecindad
aprendizaje Kohonen
no supervisado
elige neurona + similar y

Redes Neuronales Artificiales – resumen características

Red	Aprendizaje	Tipo		Codif		Func activ		Bias	Entren.	Iterat.	Inic.	Otros
McCulloch-Pitts	No			Binaria		1 y≥θ 0 y<θ		No	No	No	No	θi
Hebb	Hebb	Clasif. supervisada		Bipolar		No		Sí	Sí		W(0)=0	
Perceptrón						1 y>θ 0 -θ≤y≤θ -1 y<θ				Sí		
Adaline	Delta					Y=x (entren.) 1 y≥0 -1 y<0 (apl.)					W(0)=aleat.	
Madaline						1 y≥0 -1 y<0						
Retropropagación	Delta generaliz.			Binaria / Bipolar		Sigmoidal						
Heteroasociativa	Hebb	Asociativas	Heteroasoc.			1 y>0 -1/0 y≤0	0 signif.	No	No	No	W(0)=∑s't	
BAM				Bipolar	Convers.		Y(t) yin(t-1)=θ			Sí		θj
Autoasociativa			Autoasoc.		-					No	W(0)=∑s's	
Autoasoc. Con func. Umbral							0 signif.			Sí	wii = 0	
Hopfield					Convers.							
Maxnet	No	Competitivas		Continua		X x>0 0 x≤0					W(0)=-ε / 1	
Sombrero mejicano						Xmax x>xmax X 0≤x≤xmax 0 x<0						
Hamming			Cluster	Bipolar		No		B(0)=n/2		No	W(0)=e/2	>act.
SOM	Kohonen			Cualquiera				No	Sí	Sí	W(0)=aleat	<dist.
LVQ											W(0)=aleat	
Contrapropagación				Continua							W(0)=aleat	
ART1	"ART"			Binaria							B(0)<L/(L-1+n),	>act.
						-1 y<θ						
Adaline	Delta					Y=x (entren.) 1 y≥0 -1 y<0 (apl.)					W(0)=aleat.	
Madaline						1 y≥0 -1 y<0						
Retropropagación	Delta generaliz.			Binaria / Bipolar		Sigmoidal						
Heteroasociativa	Hebb	Asociativas	Heteroasoc.			1 y>0 -1/0 y≤0	0 signif.	No	No	No	W(0)=∑s't	
BAM				Bipolar	Convers.		Y(t) yin(t-1)=θ			Sí		θj
Autoasociativa			Autoasoc.		-					No	W(0)=∑s's	

Entrenamiento / Iteraciones

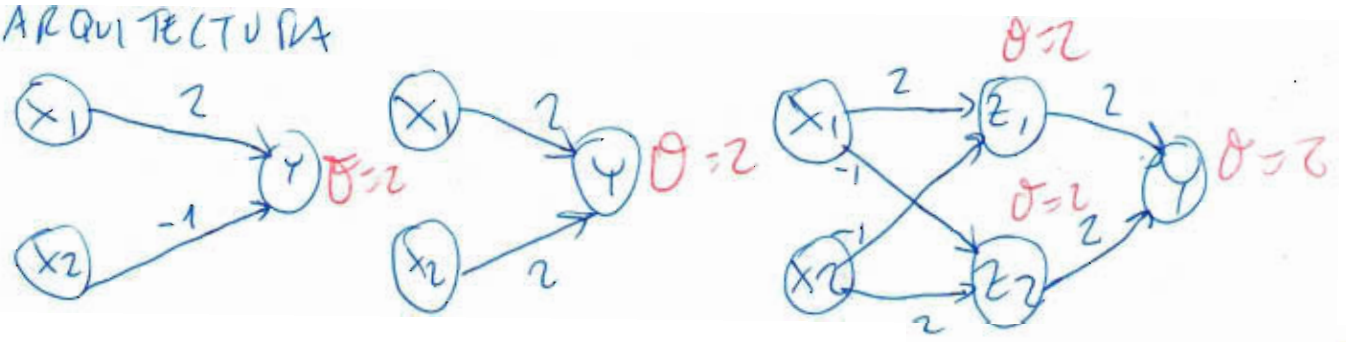
Red	Codif.	Func activ.	Bias	Tipo	Obs
Mc Culloch-Pitts	Binaria	1 $y \geq \theta$ 0 $y < \theta$	No		θ_i NE, NI
Hebb	Bipolar	—	Sí	Clasif. sep.	Hebb E, NI
Perceptron (salida)		1 $y > \theta$ 0 $-\theta \leq y \leq \theta$ -1 $y < \theta$	t ≠ y		E, I $w(0) = 0$
Adaline		$y = x$ (train) 1 $y \geq 0$ (app.) -1 $y < 0$ (app.)			Delta E, I $w(0)$ = aleat.
Madaline		1 $y \geq 0$ -1 $y < 0$	t ≠ y		E, I, $w(0)$ aleat (generalizada)
Backprop.	Binaria Bipolar	Sigmoide	$0'1Q \leq P(N+M+1)+M \leq 0'15Q$		
Heteroasoc.		1 $y > 0$ (o significativo) -1/0 $y \leq 0$	No	Asociat. (memorias)	Hebb NI $(w_{ij} = \sum s^T \cdot t)$
Auto-asoc.	Bipolar			NE	$(w_{ii} = 0)$ NI
Auto-asoc. con func. umbral		0 importante		I	$w(0) = \sum s^T \cdot s$
Hopfield	(conversión)			I	
BAM	(conversión)	$y(t)$, $y_i(t-1) = \theta$		I	$(w_{ij} = \sum s^T \cdot t$
Perceptron (salida)		0 $-\theta \leq y \leq \theta$ -1 $y < \theta$	t ≠ y		$w(0) = 0$
Adaline		$y = x$ (train) 1 $y \geq 0$ (app.) -1 $y < 0$ (app.)			Delta E, I $w(0)$ = aleat.
Madaline		1 $y \geq 0$ -1 $y < 0$	t ≠ y		E, I, $w(0)$ aleat
			$0'1Q \leq P(N+M+1)+$		

MCCULLOCH - PITTS (1943)

TIPO:

USOS:

ARQUITECTURA



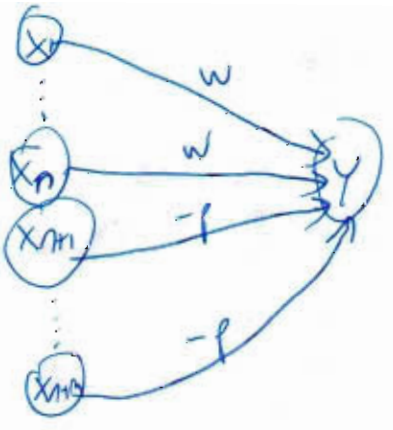
AND-NOT OR XOR (= ANDNOT OR ANDNOT)

$y(t) = z_1(t-1)\ OR\ z_2(t-1) = x_1(t-2)\ AND\ NOT\ x_2(t-2)\ OR\ x_2(t-2)\ AND\ NOT\ x_1(t-2)$

w = Peso positivo → camino excitador

-p = " negativo → " inhibidor

Todos los caminos excitadores tienen los mismos pesos



ALGORITMO

Pesos: fijos (positivos o neg.)

Entrenamiento: no hay

Func. activación

1 ÷ ——— $\theta > nw - p$ (inhibición absoluta, con un solo camino inhibidor ≠ 0

AND-NOT OR XOR (= ANDNOT OR ANDNOT)

$y(t) = z_1(t-1)\ OR\ z_2(t-1) = x_1(t-2)\ AND\ NOT\ x_2(t-2)\ OR\ x_2(t-2)\ AND\ NOT\ x_1(t-2)$

w = Peso positivo → camino excitador

-p = " negativo → " inhibidor

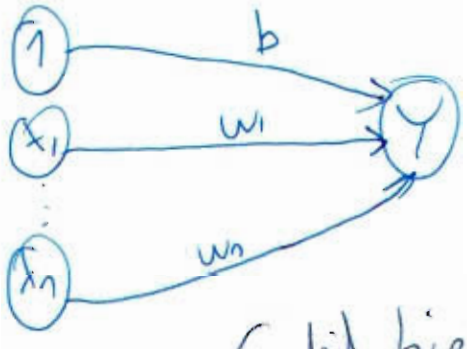
Todos los caminos excitadores tienen los mismos pesos

HEBB

TIPO: Clasif. patrones (ent. supervisado)

USOS:

ARQUITECTURA



Codif. bipolar entrada y salida

ALGORITMO:

Inic. pesos: $w_i = 0 \quad i \in (1,n)$ $\quad (w_0 = b)$

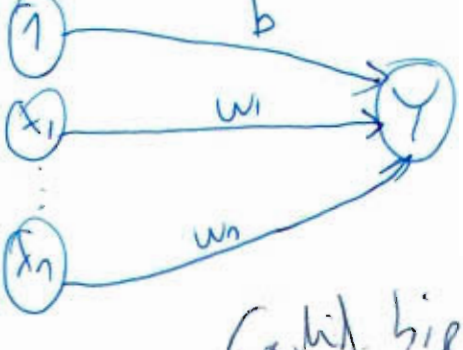
- Regla aprendizaje: Hebb extendida

$$w_i(new) = w_i(old) + x_i t$$

$$b(new) = b(old) + t \quad (x_0 = 1)$$

$x = s$
$y = t$
Ajustar pesos y bias

No hay condición de parada (los patrones se presentan

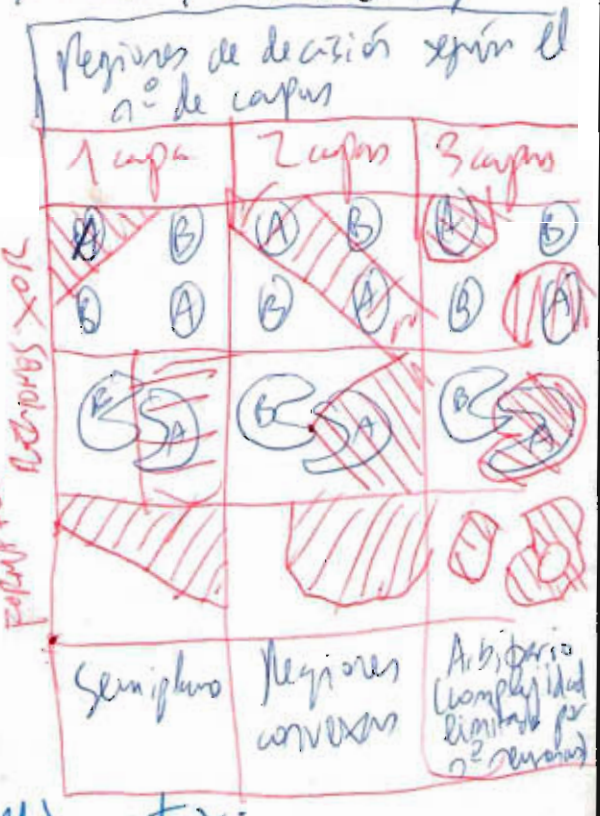


Codif. bipolar entrada y salida

PERCEPTRON

TIPO: Clasif. patrones (ent. supervisado), Feed Forward (FF)

USOS: Clasif. automática



ARQUITECTURA: = Hebb

x_i: entradas
t: valor esperado de y
y: respuesta unid. salida

ALGORITMO ENTRENAMIENTO

Inic. pesos: $w_i = 0$, $i \in [0, N]$

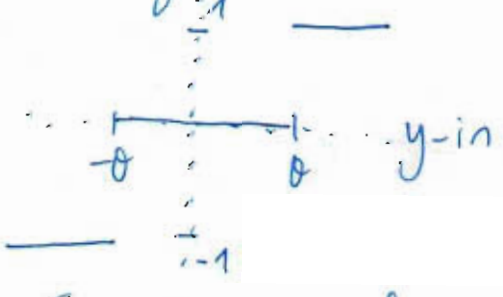
Tasa aprendizaje $0 < \alpha \leq 1$

Regla aprendizaje: (Hebb)

$$w_i(new) = w_i(old) + \alpha t x_i$$

Entrada de y: $y\_in = \sum_{i=0}^{n} x_i w_i$

Func. de activación



$$y = \begin{cases} 1 & y\_in > \theta \\ 0 & -\theta \leq y\_in \leq \theta \\ -1 & y\_in < -\theta \end{cases}$$

Cond. de parada: los pesos no han cambiado en la última época

x_i: entradas
t: valor esperado de y
y: respuesta unid. salida

ALGORITMO ENTRENAMIENTO

Inic. pesos: $w_i = 0$, $i \in [0, N]$

Tasa aprendizaje $0 < \alpha \leq 1$

Regla aprendizaje: (Hebb)

$$w_i(new) = w_i(old) + \alpha t x_i$$

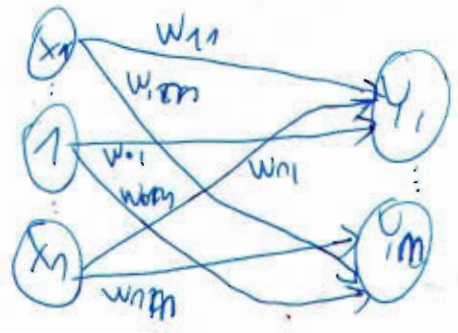
ADALINE

ADAptative Linear ~~NEtwork~~ NEuron
(Widrow & Hoff, 1960)

TIPO: Clasif. patrones (ent. supervisado), FF

USOS:

ARQUITECTURA



$y\_in_j = \sum_{i=0}^{n} x_i w_{ij}$

Q patrones

Codif. bipolar ←

ALGORITMO ENTRENAMIENTO

Inic. pesos: valores aleatorios peq.

α → muy grande: el aprendizaje no converge
→ muy peq.: el " es muy lento
→ para $n=1$, $0'1 \leq n\alpha \leq 1$ $\quad n = nº$ entradas

Regla aprendizaje: Delta ← Se aplica a todas las neuronas

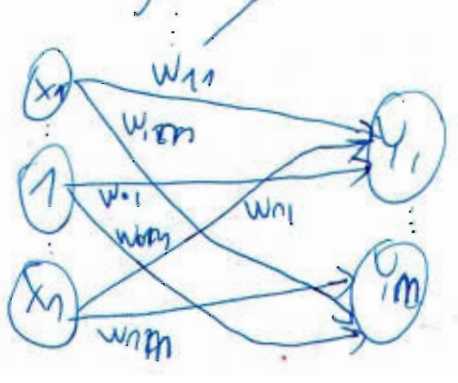
$\Delta w_{ij} = \alpha (t_j - y_{in_j}) x_i$

$w_{ij}(new) = w_{ij}(old) + \Delta w_{ij}$

→ Func. activación: $y_{in} = b + \sum x_i w_i$

$y = f(y\_in)$

$y\_in = \bar{x}\,\bar{w}$



$y\_in_j = \sum_{i=0}^{n} x_i w_{ij}$

Q patrones

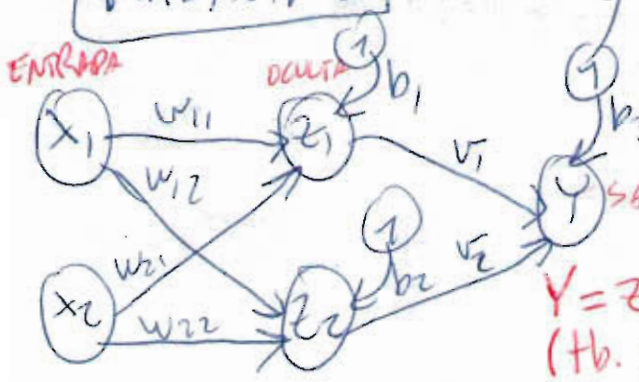
Codif. bipolar ←

ALGORITMO ENTRENAMIENTO

Inic. pesos: valores aleatorios peq.

MADALINE

Many Adaline



$Y = z_1 \text{ OR } z_2$ ($v_1 = v_2 = b_3 = \frac{1}{2}$)
(tb. se pueden utilizar otras func. como AND o "mayoría")

Conectividad total (todas las unids. de una capa conectan con todas las de la sgte)
Codif. bipolar

ALGORITMOS ENTRENAMIENTO

MRE (sólo se ajustan los pesos w_{ij})

Inic. pesos. v_1, v_2 y b_3 vienen predeterminados (fijos)[1]
w_{ij} = valor aleatorio peq.
α = valor peq.

$x_i = s_i$
$z\_in_j = b_j + \sum_i x_i w_{ij}$ $\quad z_j = f(z\_in_j)$
$y\_in = b_3 + z_1 v_1 + z_2 v_2$ $\quad y = f(y\_in)$
Si $y = t$ → salida correcta. No se modifican pesos
Loc si $t = 1$, actualizar pesos w_{ij} en la unid. j con entrada z_in_j + próxima a 0 (2)
Si $t = -1$, actualizar pesos en unids. k con entrada $z\_in_k > 0$

Cond. parada: variación de los pesos es aceptable,

Conectividad total (todas las unids. de una capa conectan con todas las de la sgte)
Codif. bipolar

ALGORITMOS ENTRENAMIENTO

MRE (sólo se ajustan los pesos w_{ij})

Inic. pesos. v_1, v_2 y b_3 vienen predeterminados (fijos)[1]
w_{ij} = valor aleatorio peq.
α = valor peq.

$x_i = s_i$

MRII (se ajustan todos los pesos)

Inicializar pesos

" α

- Calcular y como en algoritmo MRII
- Determinar error y actualizar pesos:
  - Si $y \neq t$,
    - $\forall j$ / $z_j \in (-0'25, 0'25)$
      - $z_j = -1 \cdot z_j$
      - Recalcular y
      - Si se ha reducido el error, actualizar los w_{ij} (aplicar la regla delta con z_j)

Condición parada: igual q. MRI

$\forall j$ / $z_j \in (-0'25, 0'25)$

$z_j = -1 \cdot z_j$

Recalcular y

Si se ha reducido el error, actualizar los w_{ij} (aplicar la regla delta con z_j)

Condición parada: igual q. MRI

RETROPROPAGACIÓN (Backpropagation)

TIPO: aprendizaje supervisado, feed forward

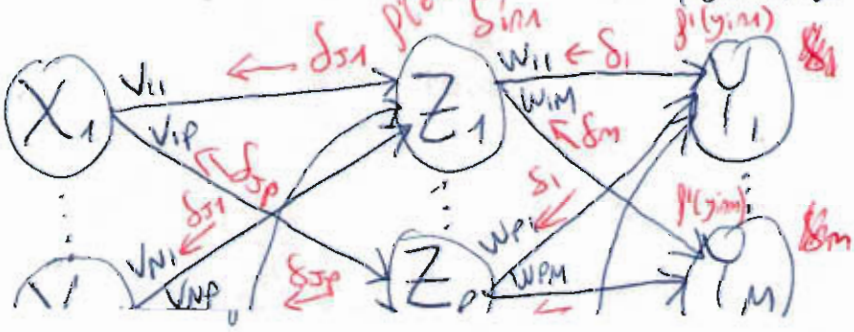
USOS: ⟹ aproximador universal (en particular, relaciones no lineales)
- filtrado de ruido
- compresión

Codif. binaria o bipolar

ARQUITECTURA

Perceptrón multicapa. Para cada problema se diseña una red adaptando los sgtes. parám.:

- nº entradas (N)
- nº salidas (M)
- nº neuronas capa oculta (P_c)
- nº de capas ocultas (C). Una capa oculta es suficiente en la mayoría de los problemas, por lo q. trabajaremos con $C = 1$ para simplificar
- func. de activación ⟹ sigmoidal ⟹ binaria / bipolar
- tasa aprendizaje α ⟹ identidad



ARQUITECTURA

Perceptrón multicapa. Para cada problema se diseña una red adaptando los sgtes. parám.:

- nº entradas (N)
- nº salidas (M)
- nº neuronas capa oculta (P_c)
- nº de capas ocultas (C). Una capa oculta es

ALGORITMO DE ENTRENAMIENTO

El proceso de aprendizaje del perceptrón multicapa equivale a encontrar un mín. en la func. de error E:

$$E = \frac{1}{Q}\sum_{n=1}^{Q} e(n) \qquad e(n) = \frac{1}{2}\sum_{i=1}^{M}[t_i(n) - y_i(n)]^2$$

Para encontrar el mín. se recurre a técnicas de optimiz. no lineal, en las cuales la búsqueda sigue la dir. negativa del gradiente de E. En la práctica lo q. se minimiza son los errores de cada patrón, $e(n)$

1) Inicializ. pesos. Heurística → valores aleatorios entre -0'5 y 0'5 (conocimiento previo)
2) Alimentación hacia adelante

$$z_{inj} = \sum_{i=0}^{N} x_i v_{ij} \qquad z_j = f(z_{inj})$$

$$y_{ink} = \sum_{j=0}^{P} z_j w_{jk} \qquad y_k = f(y_{ink})$$

Se puede definir una func. de activación diferente para cada neurona, pero en la práctica suele escogerse la misma f para todas (sigmoidal)

Para encontrar el mín. se recurre a técnicas de optimiz. no lineal, en las cuales la búsqueda sigue la dir. negativa del gradiente de E. En la práctica lo q. se minimiza son los errores de cada patrón, $e(n)$

1) Inicializ. pesos. Heurística → valores aleatorios entre -0'5 y 0'5 (conocimiento previo)

4.2. Capa oculta

$$\delta in_j = \sum_{K=1}^{M} \delta_K w_{jK}$$

$$\delta_j = \delta in_j \, f'(z in_j)$$

~~$\Delta v_{ij} = \alpha \delta_j x_i$~~

(regla delta generalizada)

5) Actualizar pesos $\Delta w_{jk} = \alpha \delta_k z_j$

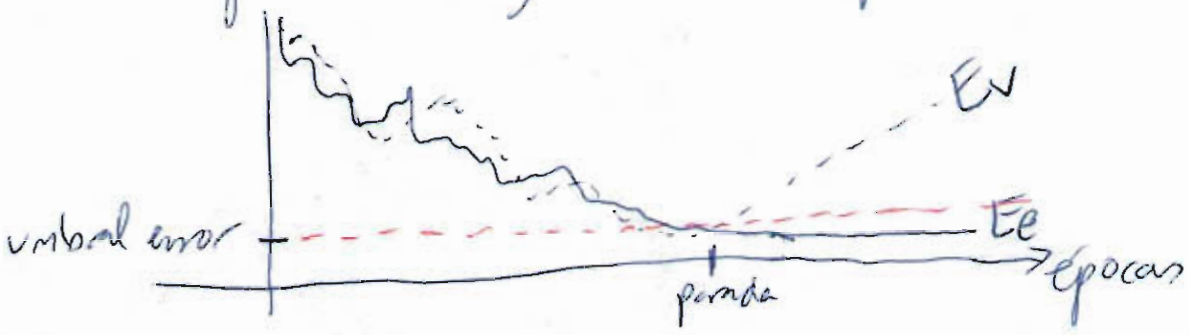
$\Delta v_{ij} = \alpha \delta_j x_i$

$$v_{ij}(t+1) = v_{ij}(t) + \Delta v_{ij}$$

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}$$

6) Evaluar condición de parada

Se calcula el error Ee (Error entrenamiento) y se observa su evol. en cada época. En las primeras debe decrecer globalmente hasta sobrepasar un umbral preestablecido. Después se controla el error de validación Ev. Cuando comienza a crecer indica q se está perdiendo generalidad y conviene parar



$$v_{ij}(t+1) = v_{ij}(t) + \Delta v_{ij}$$

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}$$

6) Evaluar condición de parada

Se calcula el error Ee (Error entrenamiento) y se observa su evol. en cada época. En las primeras debe decrecer globalmente hasta sobrepasar un umbral preestable-

El diseño de la red afecta a estos problemas:

- α
  - grande → convergencia rápida
    - mínimos locales
    - oscilación entre mínimos
  - peq. → convergencia lenta
- Pesos → Heurísticas → Nguyen-Widrow:
  $v_{ij} \in (-\beta, \beta)$, $\beta = 0{,}7\sqrt[N]{P}$

- Codif. entradas y salidas: una entrada con valor 0 impide el aprendizaje, ralentizando la convergencia. Por eso se recomienda la codif. bipolar.
- No es imprescindible normalizar entradas y salidas, pero obviarlo si no se normalizan las salidas hay q. utilizar las func. de activación identidad
- Nº de parám. = $NP + P + MP + M = P(N+M+1) + M$

  N y M suelen venir dados por el problema
  Para no atascarse en mín. locales, conviene aumentar P; pero un P alto lleva al sobre-aprendizaje
  Heurística: $10\%.Q \leq$ nº parám. $\leq 15\%.Q$
  También se puede intentar reducir N con técnicas

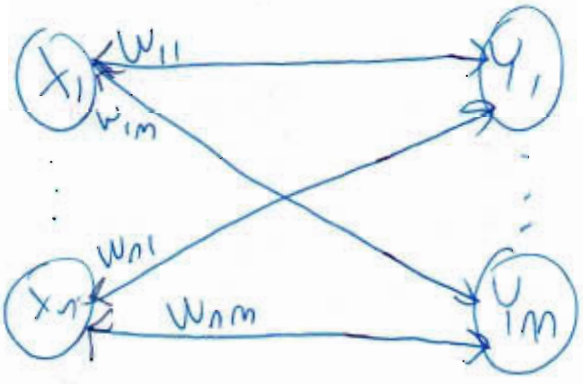
- Codif. entradas y salidas: una entrada con valor 0 impide el aprendizaje, ralentizando la convergencia. Por eso se recomienda la codif. bipolar.
- No es imprescindible normalizar entradas y salidas, pero obviarlo si no se normalizan las salidas hay q. utilizar las func. de activación identidad
- Nº de parám. = $NP + P + MP + M =$

HETEROASOCIATIVA (de flujo directo)

TIPO: Asoc. patrones (ent. supervisado). Pesos fijos

USOS:

ARQUITECTURA:



pesos simétricos

$n \neq m$ (Normalo, $m < n$)

Codif. bipolar o binaria, pero la misma para entradas y salidas

ALGO. ENTREN. $w_{ij}(0) = 0$

No hay. Los pesos se inicializan con la regla de Hebb o Delta. La matriz de pesos Hb se puede calcular como la suma de las matrices q. almacenan cada patrón por separado: $W_i = s(i)^T \cdot t(i)$; $W = \sum_{i=1}^{p} W_i$ (p = nº patrones)

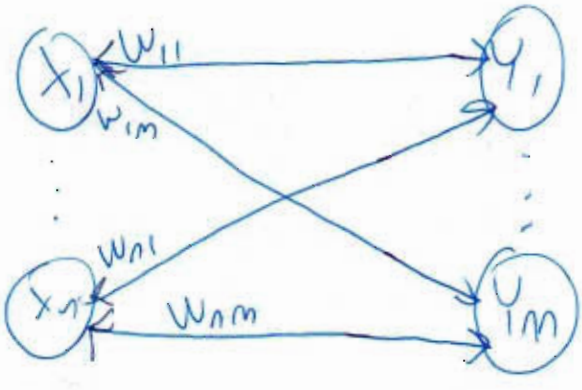
ALGO. APLICACIÓN

$$y\_in_j = \sum_{i=1}^{n} x_i w_{ij}$$

A partir de los pesos obtenidos durante el entrenamiento, se calcula la salida correspondiente para cada patrón

Func. activación y_j:

$$f(y) = \begin{cases} 1 & y > \theta \\ 0 & y = \theta \\ -1 & y < \theta \end{cases}$$ bipolar



pesos simétricos

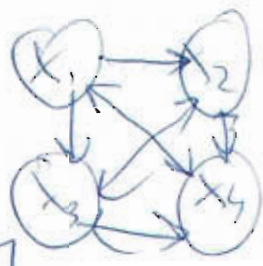
$n \neq m$ (Normalo, $m < n$)

Codif. bipolar o binaria, pero la misma para entradas y salidas

ALGO. ENTREN. $w_{ij}(0) = 0$

AUTOASOCIATIVA

(~~Heteroasociativa~~)
(no iterativa)

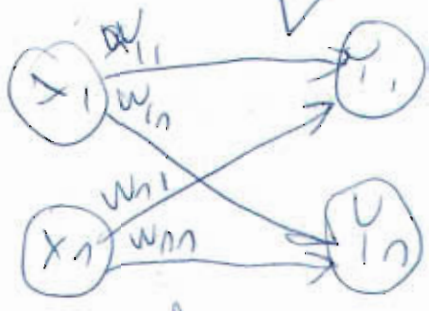


TIPO: Aspc. patrones (ent. supervisado)
Pesos fijos

USOS:

ARQUITECTURA:

= Heteroasociativa
Bipolar



Normalmente, $w_{ii} = 0$ (sin autoconexiones la red funciona mejor)

ALGO. ENTREN.

Inic. pesos: $w_{ij} = 0$

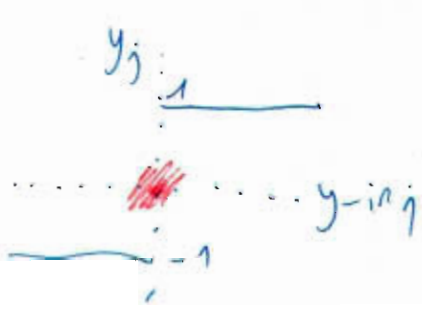
Regla aprendizaje: Hebb para vectores ortogonales ←

Delta en otro caso (en Fs sólo pueden almacenarse vectores ortogonales)

Poner a 0 los pesos de la diagonal

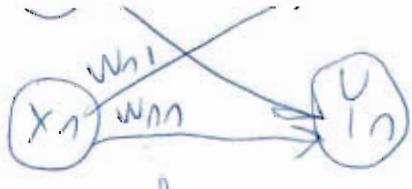
ALGO. APLICACIÓN

$y\text{-in}_j = \sum_{i=1}^{n} x_i w_{ij}$



$$f(y) = \begin{cases} 1 & y \geq 0 \\ -1 & y < 0 \end{cases}$$

= Heteroasociativa
Bipolar



Normalmente, $w_{ii} = 0$ (sin autoconexiones la red funciona mejor)

ALGO. ENTREN.

Inic. pesos: $w_{ij} = 0$

Regla aprendizaje: Hebb para vectores ortogonales ←

Delta en otro caso (en Fs sólo pueden almacenarse vectores ortogonales)

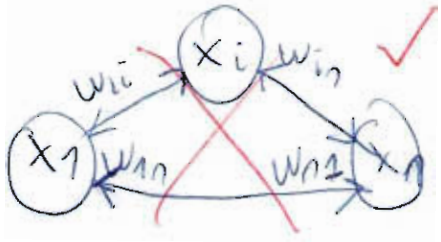
Poner a 0 los pesos de la diagonal

AUTOASOCIADOR CON FUNCIÓN UMBRAL

TIPO: Asoc. patrones (aut. supervisado)
Autoasociativa iterativa. Pesos fijos
(o casi real recurrente)

USOS:

ARQUITECTURA: como BSB pero sin autoconexiones (todas conectadas con todas)



$w_{ii} = 0$

ALGO. ENTRENAMIENTO. APLICACIÓN

Entradas y salidas bipolares

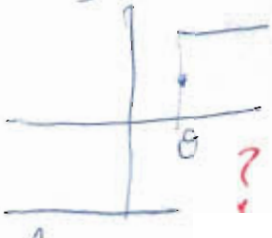
FASE 1 (aprendizaje)
Inicializ. pesos (regla de Hebb). Normalmente, $\theta_i = 0$ (cada neurona tiene su propio umbral)

FASE 2 (aplicación)
Para cada vector de entrada x_0

Activar la activación de todas las unids:

$$x_{i\,new} = \begin{cases} 1 & \text{si } \sum_j x_{j\,old} w_{ij} > \theta_i \\ x_{i\,old} & \text{si } \sum_j x_{j\,old} w_{ij} = \theta_i \\ -1 & \text{si } \sum_j x_{j\,old} w_{ij} < \theta_i \end{cases}$$

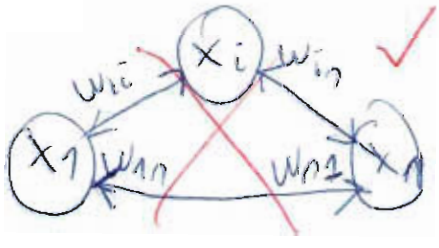
$x_{i\,in} = \sum x_j w_{ji}$
$x = f(x_{in})$



$x(0) = x$
until $x(i) \neq t$
$x_{in}(i+1) = x(i) \cdot W$
$x(i+1) = f(x_{in}(i+1))$

$t \rightarrow$ Condición de parada: x coincide con alguno de los vectores almacenados en la matriz.

ARQUITECTURA: como BSB pero sin autoconexiones (todas conectadas con todas)



$w_{ii} = 0$

ALGO. ENTRENAMIENTO. APLICACIÓN

Entradas y salidas bipolares

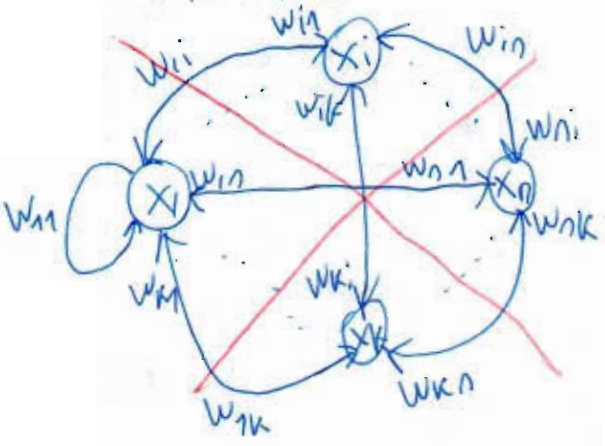
FASE 1 (aprendizaje)
Inicializ. pesos (regla de Hebb). Normalmente, $\theta_i = 0$ (cada neurona tiene su propio

BRAIN-STATE-IN-A-BOX (BSB)

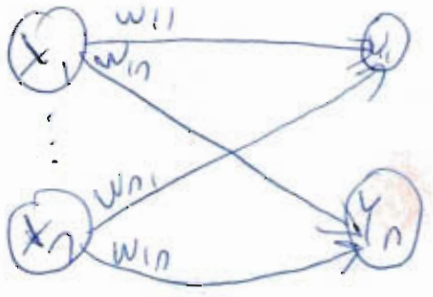
TIPO: Autoasociativa iterativa (Asoc. patrones/ent. supervisado)
Recurrente

USOS: Bipolar

ARQUITECTURA



Matriz de pesos simétrica



ALGO. ENTRÉN.

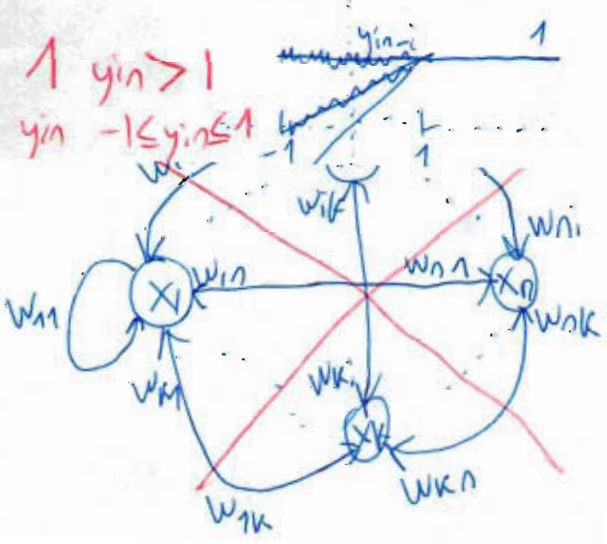
Inic. pesos: valores aleatorios peq.
Tasas de aprendizaje α y β
Regla de aprendizaje: Variante de Hebb

$$\Delta w_{ij} = \beta\, y_i\, y_j$$

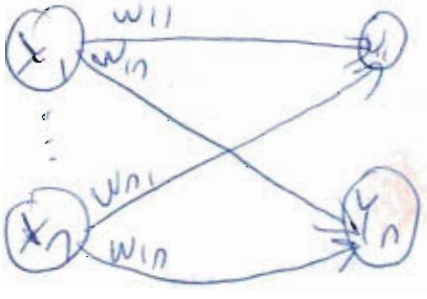
Func. activación: $y_i = x_i$ (codif. bipolar)

los valores se restringen a $(-1, 1)$

$$y\text{-}in_i = y_i + \alpha \sum_{j=1}^{n} y_j w_{ji}$$



Matriz de pesos simétrica



ALGO. ENTRÉN.

Inic. pesos: valores aleatorios peq.

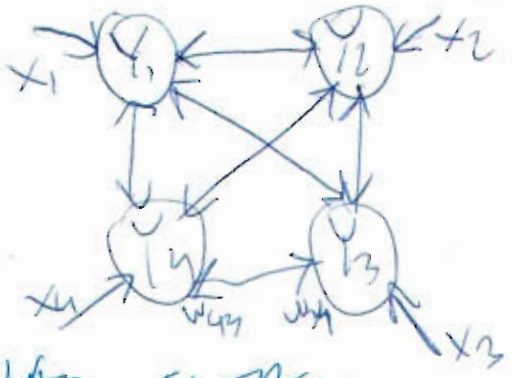
HOPFIELD DISCRETA

Bipolar o binario. La matriz es siempre bipolar, así q. si los patrones son binarios hay q. convertirlos a bipolar antes de almacenarlos

TIPO. Autoasoc. iterativa (asoc. patrones / entren. no supervisado)
Recurrente. Pesos fijos

USOS: Mem. de acceso por contenido, reconocimiento de patrones (imágenes / voz...), optimización con restricciones (problema del viajante...)

ARQUITECTURA: = BSB



pesos simétricos $w_{ij} = w_{ji}$

$w_{ii} = 0$

la matriz de pesos siempre es bipolar, por eso si las entradas son binarias se convierten a bipolar durante la fase de entrenamiento. En la fase de aplic. las entradas se dejan en su codif., y el resultado se compara con los vectores almac. en la aplic. orig.

ALGO. ENTREN.

No supervisado. Los pesos se calculan mediante la sgte. fórmula (Hebb)

Hebb $w_{ij} = \begin{cases} 0 & \text{si } i = j \\ \sum_{p=1}^{P} s_i(p)\, s_j(p) & \end{cases}$

simplemente los patrones sobre la misma

binaria $\rightarrow w_{ij} = \sum_p [2s_i(p) - 1][2s_j(p) - 1]$

Hebb supervisada $\rightarrow s_i(p)\, t_j(p)$

bipolar no supervisada

, donde $s_i(p)$ son las componentes del patrón $s(p)$. P es el nº de patrones. Matricialmente,

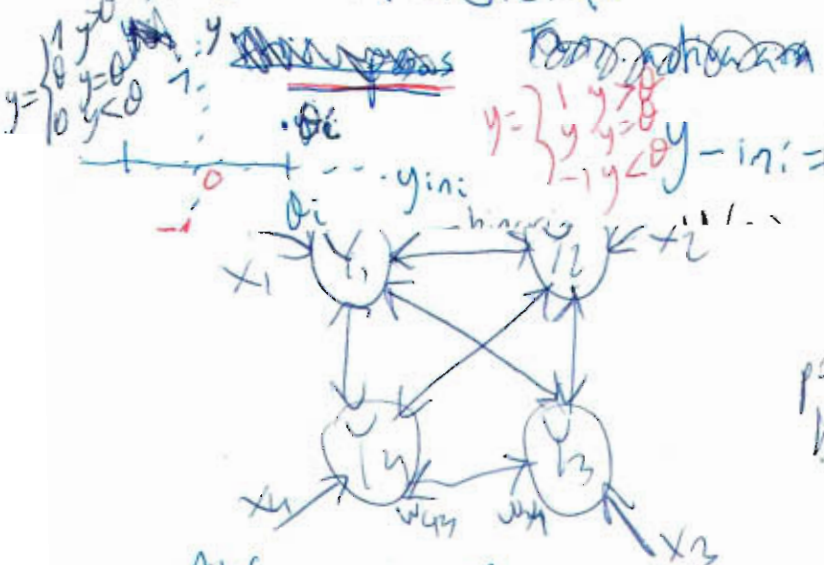
$$\sum_p [s(p)^T s(p)]$$

(suma cada matriz correspondiente al producto de $s(p)^T$ por $s(p)$)

ALGO. APLICACIÓN

$$y = \begin{cases} 1 & y > \theta \\ 0 & y = \theta \\ 0 & y < \theta \end{cases} \qquad y = \begin{cases} 1 & y > \theta \\ y & y = \theta \\ -1 & y < \theta \end{cases}$$

$$y\_in_i = x_i + \sum_j y_j w_{ji}$$



$w_{ii} = 0$

la matriz de pesos siempre es bipolar, por eso si las entradas son binarias se convierten a bipolar durante la fase de entrenamiento. En la fase de aplic. las entradas se dejan en su codif., y el resultado se compara con los vectores almac. en la aplic. orig.

ALGO. ENTREN.

No supervisado. Los pesos se calculan mediante la sgte. fórmula (Hebb)

Hebb $w_{ij} = \begin{cases} 0 & \text{si } i = j \end{cases}$

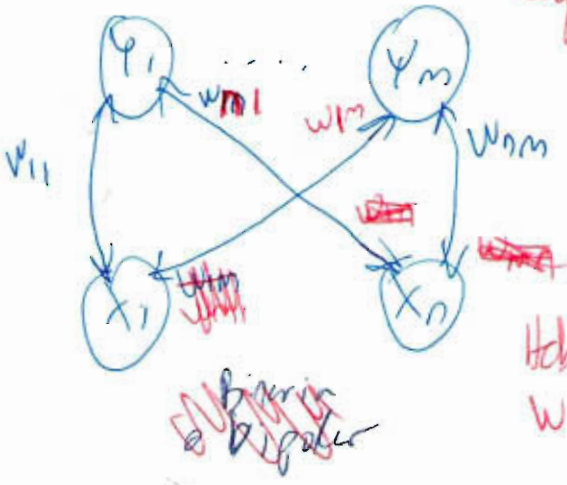
BIDIRECTIONAL ASSOCIATIVE MEMORY (BAM)

TIPO: ~~Autoasoc.~~ Heteroasoc. iterativa (no supervisado, recurrente), pesos fijos

USOS: X o Y pueden ser incompletos / erróneos

→ binaria o bipolar (mejor bipolar) → binario se convierte a bipolar

ARQUITECTURA



la recuperación es iterativa: mientras no se converge, se calculan alternativamente X e Y (explicando la bidireccionalidad)

Pesos señales $X \to Y = W$

(SIMÉTRICOS) $Y \to X = W^T$

Peso $x_i \to y_j$ = Peso $y_j \to x_i$

$$\begin{pmatrix} 11 & 12 \\ 21 & 22 \\ 31 & 32 \\ 41 & 42 \end{pmatrix}$$

Hebb patrones binarios

$$w = \sum_p (2s(p)^T - 1) \cdot (2t(p) - 1)$$

ALGO. ENTREN. / APLICACIÓN

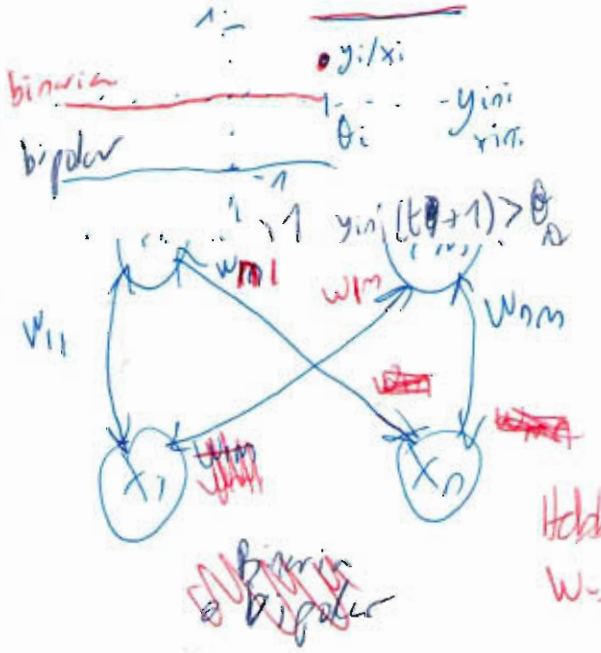
FASE 1 Inic. pesos (almacenamiento): Regla Hebb (patrones bipolares) entrada p, objetivo p

$$w_{ij} = \sum_p s_i(p)\, t_j(p) \qquad s_i(p),\ t_i(p)$$

FASE 2 Presentar patrones a la capa X ~~y~~ ← un patrón incompleto en X o en Y sirve de pto. de partida, ayudando a completar posiciones y la red lo puede

(recuperación)

Func. activación



Calcular respuesta capa Y (y_j)

$$y\_in_j = \sum_{i=1}^{n} w_{ij} x_i \qquad y_j = f(y\_in_j)$$

Calcular respuesta capa X (x_i)

$$x\_in_i = \sum_{j=1}^{m} w_{ij} y_j \qquad x_i = f(x\_in_i)$$

(SIMÉTRICOS) $Y \to X = W^T$

Peso $x_i \to y_j$ = Peso $y_j \to x_i$

$$\begin{pmatrix} 11 & 12 \\ 21 & 22 \\ 31 & 32 \\ 41 & 42 \end{pmatrix}$$

Hebb patrones binarios

$$w = \sum_p (2s(p)^T - 1) \cdot (2t(p) - 1)$$

ALGO. ENTREN. / APLICACIÓN

FASE 1 Inic. pesos: Regla Hebb (patrones bipolares) entrada p, objetivo p

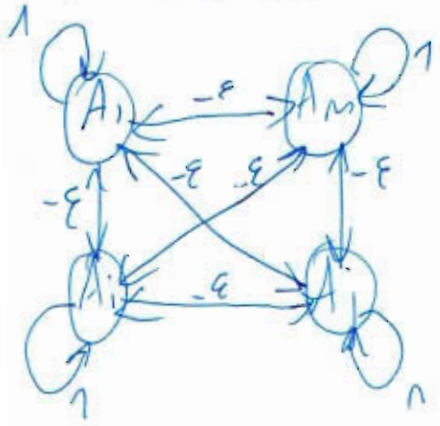
MAXNET

Coef. conexión positiva

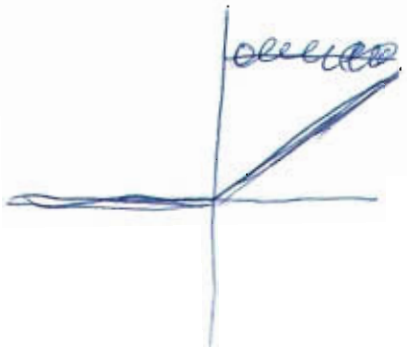
TIPO: Competitiva pesos fijos

USOS:

ARQUITECTURA



FUNC. ACTIVACIÓN



$x \quad x > 0$

$0 \quad x \leq 0$

ALGO. ENTREN.

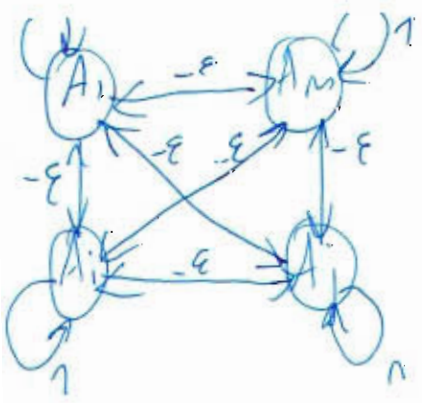
No hace. $w_{ij} = \begin{cases} 1 & \text{si } i = j \\ -\varepsilon & \text{si } i \neq j \end{cases} \quad (0 < \varepsilon < \frac{1}{m})$

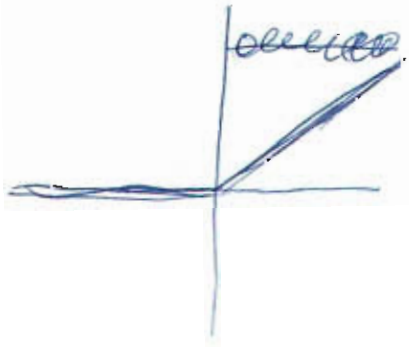
ALGO. APLICACIÓN

Actualizar activaciones:

$$a_j(new) = f\left[a_j(old) - \varepsilon \sum_{k \neq j} a_k(old)\right] = f\left(\sum_i w_{ij}\, a_j(old)\right)$$

Cond. parada: sólo hay una nodo activado





$x \quad x > 0$

$0 \quad x \leq 0$

ALGO. ENTREN.

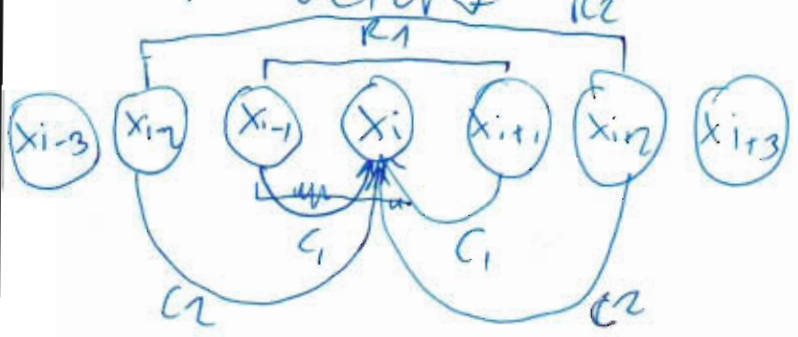
SOMBRERO MEXICANO

Tipo: competitiva pero fijo

Usos: refuerzo contraste

Codif. continua

ARQUITECTURA



R1: conexiones excitadoras
R2: conexiones inhibidoras

cooperación

R1: radio de refuerzo positivo
$[W_K = C_1, \; C_1 > 0)$

R2: radio competición
$[W_K = C_2, \; C_2 < 0)$

región cooperación: $\{x_i, x_{i+1}, x_{i-1}\}$
competencia: $\{x_{i-2}, x_{i+2}\}$

ALGO. APLICACIÓN

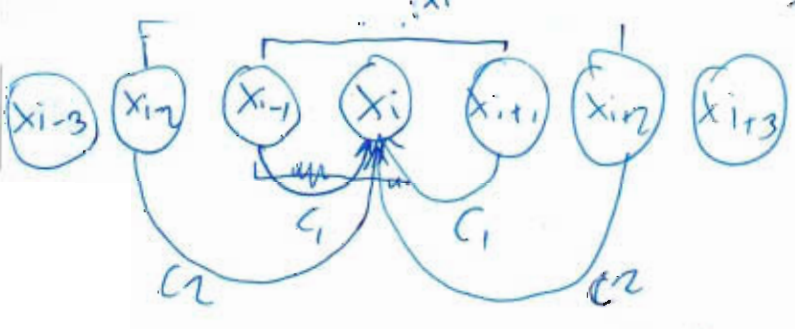
Para cada unidad de activación x_i, calcular la entrada neta,

$$x_{in_i}^1 = C_1 \sum_{K=-R1}^{R1} x_{i+K}(old) + C_2 \sum_{K=-R2}^{-R1-1} x_{i+K}(old) + C_2 \sum_{K=R1+1}^{R2} x_{i+K}(old)$$

región cooperación (anillo interno) — región competición (anillo externo)

$x_i(old) = x_i$

Func. activación: $x_i = f(x_{in_i})$



R1: conexiones excitadoras
R2: conexiones inhibidoras

R1: radio de refuerzo positivo
$[W_K = C_1, \; C_1 > 0)$

R2: radio competición
$[W_K = C_2, \; C_2 < 0)$

región cooperación: $\{x_i, x_{i+1}, x_{i-1}\}$
competencia: $\{x_{i-2}, x_{i+2}\}$

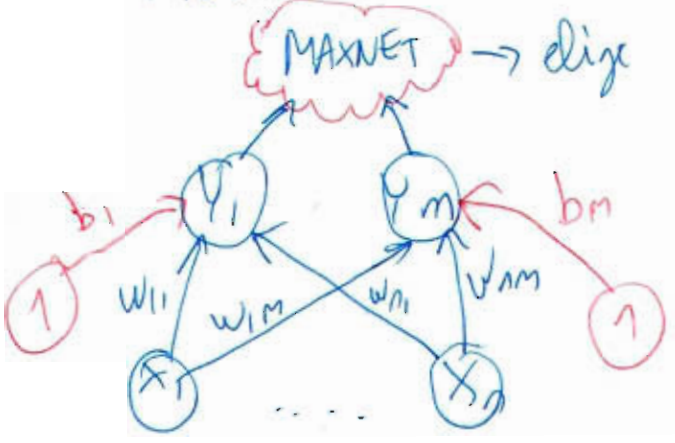
HAMMING

Bipolar / ~~binaria~~ → tiene sentido usar codif. binaria

TIPO. Competitive peso fijo

USOS: determinar a q. vector ejemplar se parece + el vector de entrada

ARQUITECTURA



ALGO. ~~APLICACIÓN~~

Almacenamiento

Inic. pesos: $w_{ij} = \frac{e_i(j)}{2}$

bias $b_j = \frac{n}{2}$

$e(j)$ es el patrón de referencia (ejemplar) de la unid Y_j

$e_i(j)$ es cada una de sus componentes

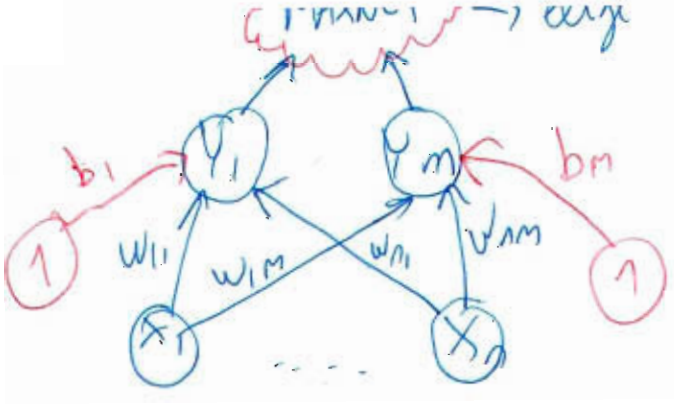
Para cada patrón de entrada,

Comparación

Entrada neta: $y\_in_j = b_j + \sum_{i=1}^{n} x_i w_{ij}$ (x e y son vectores bipolares)

$y_j(\text{new}) = y\_in_j$.

Buscar el mejor ejemplar con MAXNET aplicada sobre el vector y



ALGO. ~~APLICACIÓN~~

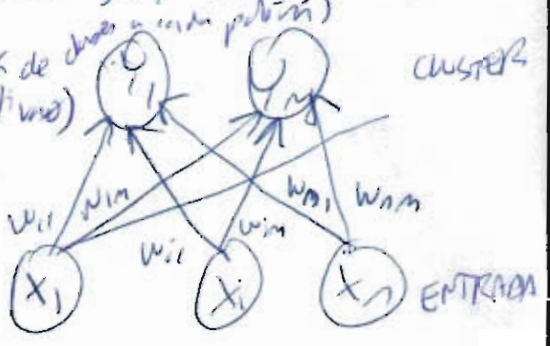
MAPA AUTOORGANIZATIVO (KOHONEN)

Cualquier codif.

TIPO: Competitiva / entrena. no supervisada, recurrente

USOS: agrupar (clasificar) los patrones en clusters sin conocimiento

2/ q. la clasif. sea estable (no hay cambios en la asignación de clase a cada patrón)
3/ q. la clasif. obtenida (buscada) patrón se asigna a una clase determinada)



ARQUITECTURA: = heteroasociativa (pero con topología)

El nº de clases (salidas) se desconoce a priori, ALGO. ENTRENEN. por lo q. se estima

Inicializ. pesos: pueden reflejar un conocimiento previo, o simplem ser valores aleatorios

Elegir Radio vecindad R, Func. vecindad (lineal, rectangular, hexagonal)[2]

Inic. tasa aprendizaje α y tasa de variación de R

Para cada patrón:

Buscar la neurona J + próxima al vector de entrada x [aquella con la distancia euclídea D menor, $D(j) = \sum_{i=1}^{n} (w_{ij} - x_i)^2$]

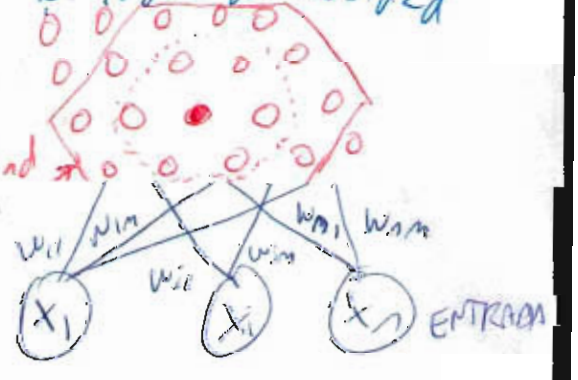
Regla aprendizaje Kohonen: $\Delta w_{ij} = \alpha (x_i - w_{ij}(t))$ respecto a los pesos

Actualizar pesos de la unid. ganadora y_j y sus vecinas

En cada época se actualiza α y el radio de vecindad

Cond. parada: 1) iteraciones $\geq 500 \cdot n$ épocas

COMENTARIOS



ARQUITECTURA: = heteroasociativa (pero con topología)

El nº de clases (salidas) se desconoce a priori, ALGO. ENTRENEN. por lo q. se estima

Inicializ. pesos: pueden reflejar un conocimiento previo, o simplem ser valores aleatorios

Elegir Radio vecindad R, Func. vecindad (lineal, rectangular, hexagonal)[2]

Inic. tasa aprendizaje α y tasa de variación de R

LEARNING VECTOR QUANTIZATION (LVQ)

TIPO: Asoc. patrones / entren. supervisado, FF, competencia

USOS: Clasificar patrones nuevos en base a otros conocidos
→ SOM para aprendizaje supervisado

ARQUITECTURA
= alg. SOM, pero sin topología. Se conoce a q. clase pertenece cada patrón [es decir, se sabe de antemano el nº de clases, y por tanto, el nº de salidas]

ALGO. ENTREN. (LVQ1)

Inicializ. vectores de pesos de referencia w_j (pares):
1) Usar los m 1os vectores de entrenamiento
2) Inicializ. los pesos aleatorios
Inicial. tasa aprendizaje α
Buscar la unid. de salida con menor distancia euclídea → $\|x - w_j\| = \sqrt{\sum (x_i - w_{ij})^2}$ entre el patrón y los pesos de ref.
Si la categoría obtenida es = a la categoría correcta del patrón
obtenida → $w_j(new) = w_j(old) + \Delta w_j$
eoc → $w_j(new) = w_j(old) - \Delta w_j$
$\Delta w_j = \alpha (x - w_j(old))$ (Kohonen)
→ Reducir α (en cada época)
Cond. parada — nº fijo iteraciones

= alg. SOM, pero sin topología. Se conoce a q. clase pertenece cada patrón [es decir, se sabe de antemano el nº de clases, y por tanto, el nº de salidas]

ALGO. ENTREN. (LVQ1)

Inicializ. vectores de pesos de referencia w_j (pares):
1) Usar los m 1os vectores de entrenamiento

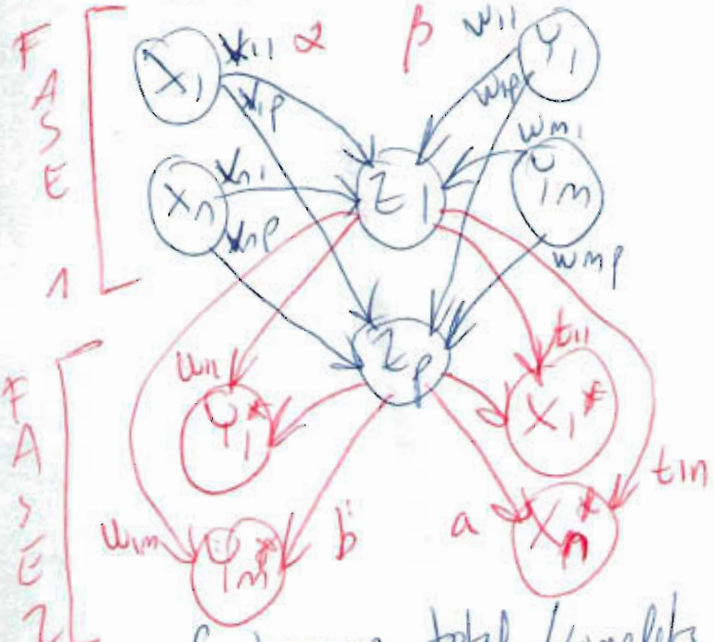
CONTRAPROPAGACIÓN (Counterpropagation)

Codif. Gaussiana Real

TIPO: Competitiva

Usos: Aprox. func. } contraprop. total
Asociar patrones }
Comprimir info. (contrapropagación de flujo directo)

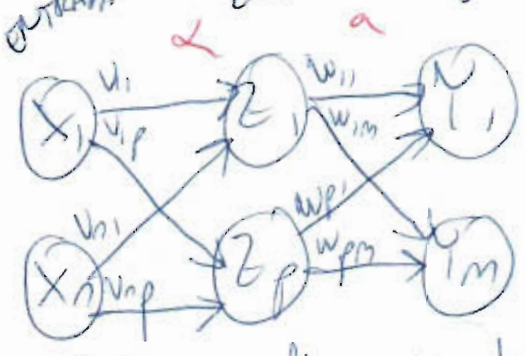
ARQUITECTURA



Contraprop. total / completa

Func. invertible: $x \to y$

X, Y: entradas; Z: cluster; X^*, Y^*: salidas



Contraprop. flujo directo / sólo hacia adelante

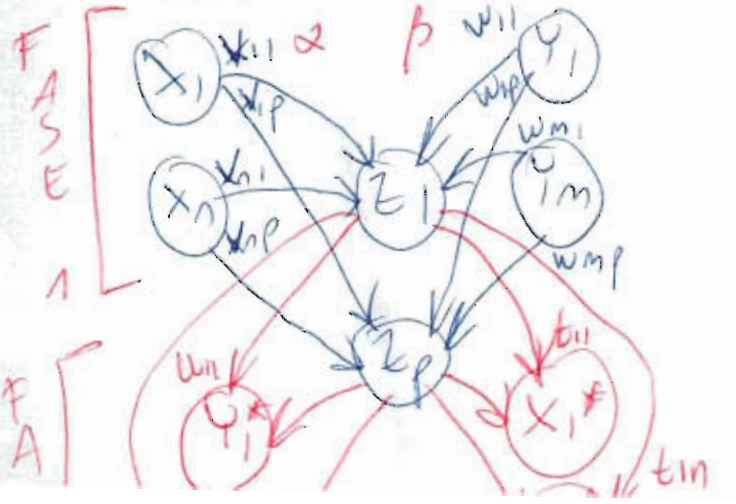
Func. no invertible: $x \to y$

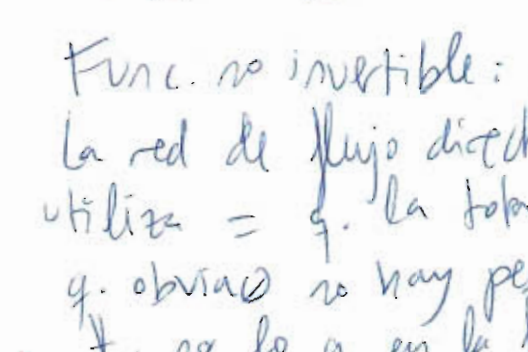
La red de flujo directo se utiliza = q. la total, salvo q. obviamente no hay pesos u y t, por lo q. en la fase 1 se calculan sólo los v_{ij}, y en la 2 los t_{jk} (q. aquí se etiquetan como w_{jk})

ALGO. ENTREN.

Inicializar pesos: v, w, t, u

ARQUITECTURA





Contraprop. flujo directo / sólo hacia adelante

Func. no invertible: $x \to y$

Actualizar pesos v y w con Δv y Δw

Reducir factores aprendizaje α y β (después de cada época, no de cada patrón)

Cond. parada Fase 1 (pesos no cambian, entradas se asocian siempre a la misma unid, factor aprendizaje muy peq...)

Fase 2: refinamiento pesos cluster → salida

(en esta fase, α y β han quedado reducidos a valores peq)

Entradas → $X = x$, $Y = y$

Buscar unid. ganadora J

Ajuste pesos → v, w → mín (α y β son muy peq)

→ u_{JK} y t_{Ji} → usar Δu y Δt:

$$\Delta u_{JK} = a\,(y_k - u_{JK}\,old)$$
$$\Delta t_{Ji} = b\,(x_i - t_{Ji}\,old)$$

objetivos de X^*_k e Y^*_i — activación unids. X^*_k e Y^*_a

por eso puede considerarse regla Delta

Reducir factores aprendizaje a y b

Cond. parada Fase 2

APLICACIÓN

Suministrar patrones x e y a la red (x o y pueden ser desconocidos), con lo q. la red activa una neurona Z_J del cluster (la de < distancia). Los pesos u_{JK} y t_{Ji} asociados a dicha neurona son la aprox. x^* e ...

Buscar unid. ganadora J

Ajuste pesos → v, w → mín (α y β son muy peq)

→ u_{JK} y t_{Ji} → usar Δu y Δt:

$$\Delta u_{JK} = a\,(y_k - u_{JK}\,old)$$
$$\Delta t_{Ji} = b\,(x_i - t_{Ji}\,old)$$

objetivos de X^*_k e Y^*_i — activación unids. X^*_k e Y^*_a

por eso puede considerarse regla Delta

ART — Adaptative Resonance Theory

Diseñada para ser estable y plástica

ART 1: entradas binarias — ART 2: entradas continuas

TIPO: No supervisada

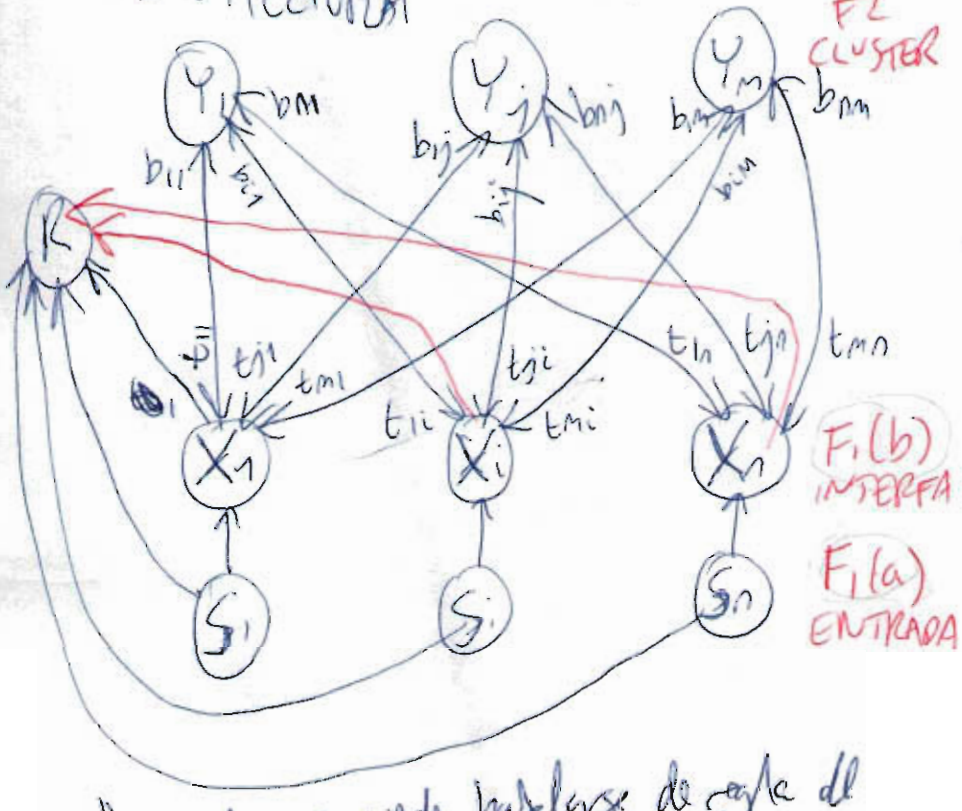
~~Binaria, bipolar~~ ?

- Con el entrenamiento se consigue q. en los pesos t_{ji} se almacenen los representantes de cada categoría

USOS: Clasificación. Basada en la comparación q. se hace entre la info. sin procesar que llega al cuerpo geniculado y la info. procesada de la corteza visual primaria

ARQUITECTURA

capa competitiva



n = nº componentes vector entrada

M = nº máx. de clusters

b_{ij} [2] = pesos $F_1(b) \to F_2$

t_{ji} [3] = pesos $F_2 \to F_1$

Las neuronas se representan en mayúscula, y su activación en minúscula: Y_1 vs y_j

R → mecanismo de Reset (previo a la actualiz. de los pesos)

- En esta red no puede hablarse de regla de aprendizaje, es una simple actualiz. de pesos

grado de similaridad entre los patrones de un mismo cluster

factor de vigilancia

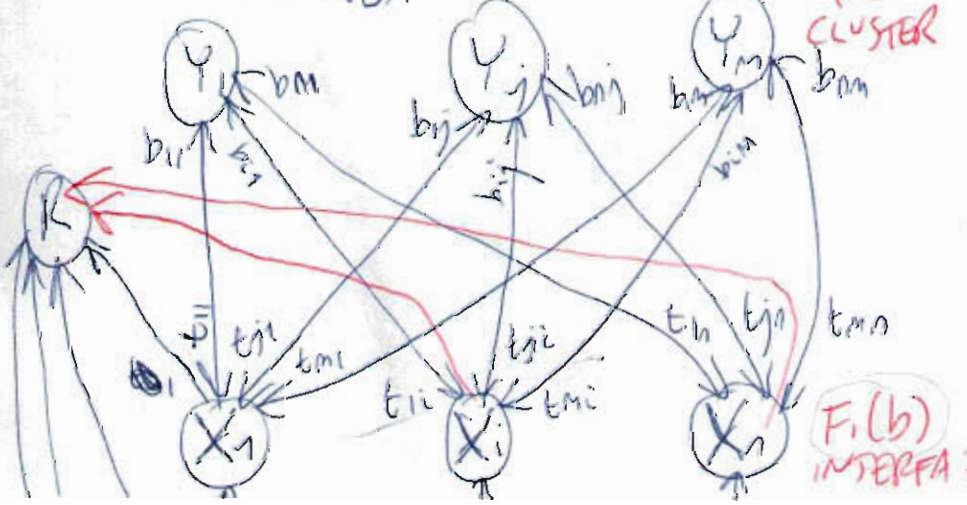
$0 < \rho \leq 1$

ALGO. ENTRENAMIENTO (ART 1)

Inicializar parámetros: $L > 1$

Inicializar pesos: $0 < b_{ij}(0) < \frac{L}{L-1+n}$; $t_{ij}(0) = 1$

ARQUITECTURA



n = nº componentes vector entrada

M = nº máx. de clusters

b_{ij} [2] = pesos $F_1(b) \to F_2$

t_{ji} [3] = pesos $F_2 \to F_1$

Las neuronas se representan en mayúscula, y su activación en minús-

- Calcular la norma de s^1 (el vector de entrada):

$$\|s\| = \sum_i s_i$$

- Mandar las señal de entrada de $F_1(a)$ a $F_1(b)$

$$x_i = s_i$$

- Para cada nodo de F_2 (q. no está inhibido, $(y_j \neq -1)$), calcular el valor de activación $y_j = \sum_i b_{ij} x_i$ → $\vec{y} = x\,b$ (vectorial)

- Mientras Reset sea verdadero,
  - encontrar la neurona Y_J ganadora, i.e. aquella tal q.

$$Y_J \geq y_j \quad \forall j$$

  - recalcular las activaciones de $F_1(b)$

$$x_i = s_i t_{Ji} \qquad x = s\,t_J \text{ (escalar)}$$

  - calcular la norma de x

Test de Reset $\|x\| = \sum_i x_i$

Si $\frac{\|x\|}{\|s\|} < \rho \rightarrow y_J = -1$ (se comparan x con s)

RESET (inhibir nodo ganador eliminándolo de la competición)

EOC, actualizar pesos para nodo J y poner Reset = False:

$$b_{iJ}(new) = \frac{L x_i}{L - 1 + \|x\|}$$

$$t_{Ji}(new) = x_i$$

(una vez q. se asigna 0 a t_{Ji}, ya no puede volver a valer 1, con lo q. se consigue la estabilidad del objeto de ART)

- Evaluar condición de parada (una de las 3 sgtes):
  - los pesos no han cambiado
  - no se han reiniciado unids.
  - se ha alcanzado el nº máx. de épocas

el valor de activación $y_j = \sum_i b_{ij} x_i$

El Reset es un mecanismo para controlar el grado de similaridad de los patrones colocados en un mismo cluster

- Mientras Reset sea verdadero,
  - encontrar la neurona Y_J ganadora, i.e. aquella tal q.

$$Y_J \geq y_j \quad \forall j$$

  - recalcular las activaciones de $F_1(b)$

$$x_i = s_i t_{Ji} \qquad x = s\,t_J \text{ (escalar)}$$

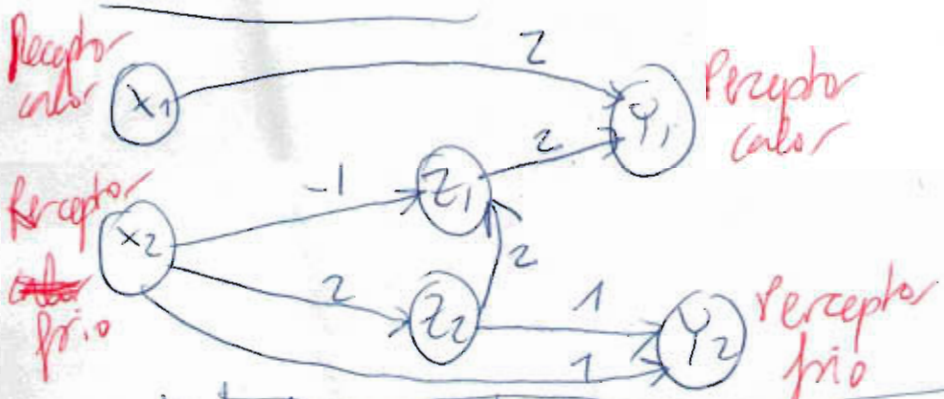
  - calcular la norma de x

$$\|x\| = \sum_i x_i$$

El Reset es un mecanismo para controlar el grado de similaridad de los patrones colocados en un mismo cluster

EJEMPLOS

McCulloch-Pitts



Todas las neuronas tienen un umbral de activación 2

t	1				2					
x_1	0	0	1	1	0	0	0	1	1	0	0	1	1	0
x_2	0	1	0	1	1	1	1	0	0	0	1	1	0	0
z_1	0	-1	0	-1	-1+4=3	4	0+4	0	0+0	0	-1+0	-1	0+4	0
z_2	0	2	0	2	2	0		0	0		2		0	
y_1	0	0	0	2	0+-2=-2	-2		2+0=2	0+0		0+0		2+-2	
y_2	0	1	2	1	1+2=3	2		0+0	1+0		0+2			
	calor	frío	frío	calor	frío	frío		frío	calor		frío			

$t=1$ $x_1=1$ (calor) $y_2=1$ (frío) → $x_1=1$ (calor) $y_1=1$ (calor)
→ $x_2=1$ (frío) $y_1=y_2=0$

$x_2=1$ (frío) $y_2=1$ (frío) → $x_1=1$ (calor) $y_2=1$ (frío)
→ $x_2=1$ (frío) $y_2=1$ (frío)

Observando los valores, se deduce q. si se recibe frío en 2 instantes consecutivos, se percibe frío en ambos. Si se recibe calor y luego en los 2 instantes, 1º se percibe frío y luego calor. Si se recibe primero frío, primero se percibe frío y

t	1				2					
x_1	0	0	1	1	0	0	0	1	1	0	0	1	1	0
x_2	0	1	0	1	1	1	1	0	0	0	1	1	0	0
z_1	0	-1	0	-1	-1+4=3	4	0+4	0	0+0	0	-1+0	-1	0+4	0
z_2	0	2	0	2	2	0		0	0		2		0	
y_1	0	0	0	2	0+-2=-2	-2		2+0=2	0+0		0+0		2+-2	
y_2	0	1	2	1	1+2=3	2		0+0	1+0		0+2			
	calor	frío	frío	calor	frío	frío		frío	calor		frío			

Intro. el tiempo

$Y_1(t) = X_1(t-1)$ OR $Z_1(t-1)$

$Y_2(t) = X_2(t-1)$ AND $Z_2(t-1)$

$Z_1(t) = Z_2(t-1)$ AND NOT $X_2(t-1)$

↳ $Z_1(t-1) = Z_2(t-2)$ AND NOT $X_2(t-2)$

$Z_2(t) = X_2(t-1) \rightarrow Z_2(t-2) = X_2(t-3)$

$Y_1(t) = X_1(t-1)$ OR $(Z_2(t-2)$ AND NOT $X_2(t-2))$

$= \boxed{X_1(t-1) \text{ OR } (X_2(t-3) \text{ AND NOT } X_2(t-2))}$

$Y_2(t) = \boxed{X_2(t-1) \text{ AND } X_2(t-2)}$

Es decir, se percibe calor si se ha ~~percibido~~ recibido calor en el instante anterior, o se ha ~~experimentado~~ recibido frío solo un instante. Se percibe frío si durante 2 instantes se ha ~~percibido~~ recibido frío.

$Y_1(t) = X_1(t-1)$ OR $(Z_2(t-2)$ AND NOT $X_2(t-2))$

$= \boxed{X_1(t-1) \text{ OR } (X_2(t-3) \text{ AND NOT } X_2(t-2))}$

$Y_2(t) = \boxed{X_2(t-1) \text{ AND } X_2(t-2)}$

Es decir, se percibe calor si se ha ~~percibido~~ recibido calor en el instante anterior, o se ha ~~experimentado~~ recibido frío solo un instante. Se percibe frío si durante 2 instantes

Hebb

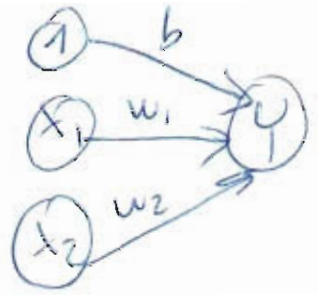
Func. AND

$x(1) = (1 \quad 1)$ $\quad t(1) = 1$

$x(2) = (1 \quad -1)$ $\quad t(2) = -1$

$x(3) = (-1 \quad 1)$ $\quad t(3) = -1$

$x(4) = (-1 \quad -1)$ $\quad t(4) = -1$



Inicializ. pesos

$w = (0 \quad 0)$

$x(1) \longrightarrow w_i(new) = w_i(old) + x_i t_i$

$w_1(new) = 0 + 1 \cdot 1 = 1$

$w_2(new) = 0 + 1 \cdot 1 = 1$

$b(new) = 0 + 1 \cdot 1 = 1$

$1 + x_1 + x_2 = 0$

$x_2 = -x_1 - 1$ R1

$x(2) \longrightarrow w_1(new) = 1 + (1 \cdot (-1)) = 0$

$w_2(new) = 1 + (-1 \cdot -1) = 2$

$b(new) = 1 - 1 = 0$

$2x_2 = 0$

$x_2 = 0$ R2

$x(3) \longrightarrow w_1(new) = 0 + (-1 \cdot -1) = 1$

$w_2(new) = 2 + (1 \cdot (-1)) = 1$

$b(new) = 0 - 1 = -1$

$-1 + x_1 + x_2 = 0$

$x_2 = 1 - x_1$ R3

$x(4) \longrightarrow w_1(new) = 1 + (-1(-1)) = 2$

$w_2(new) = 1 + (-1(-1)) = 2$

$b(new) = -1 + -1 = -2$

$-2 + 2x_1 + 2x_2 = 0$

$x_2 = \frac{2 - 2x_1}{2} = 1 - x_1$ R4

$x(3)$ x_2 $+ x(1)$

R1

Inicializ. pesos

$w = (0 \quad 0)$

$x(1) \longrightarrow w_i(new) = w_i(old) + x_i t_i$

$w_1(new) = 0 + 1 \cdot 1 = 1$

$w_2(new) = 0 + 1 \cdot 1 = 1$

$b(new) = 0 + 1 \cdot 1 = 1$

$1 + x_1 + x_2 = 0$

$x_2 = -x_1 - 1$ R1

$x(2) \longrightarrow w_1(new) = 1 + (1 \cdot (-1)) = 0$

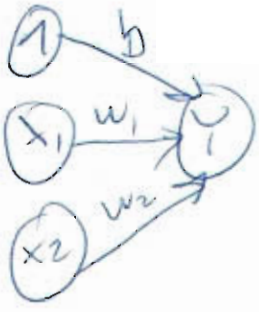
$2x_2 = 0$

Perceptrón

función AND

$x(1) = (0\ \ 0)$ $t(1) = -1$
$x(2) = (0\ \ 1)$ $t(2) = -1$
$x(3) = (1\ \ 0)$ $t(3) = -1$
$x(4) = (1\ \ 1)$ $t(4) = 1$

$\alpha = 1$
$\theta = 0'2$



$w_1 = w_2 = b = 0$

x(1) $y_{in} = 0 \longrightarrow y = 0$ $y \neq t(1)$

$w_i(new) = w_i old + \alpha t x_i$

$w_1(new) = 0 + 1 \cdot (-1) \cdot 0 = 0$

$w_2(new) = 0 = 0$

$b(new) = 0 + 1 \cdot (-1) = -1$

no define recta

x(2) $y_{in} = -1 \longrightarrow y = -1$ $y \neq t(2)$

$w_1(new) = 0 + 1 \cdot (-1) \cdot 0 = 0$

$w_2(new) = 0 + 1 \cdot (-1) \cdot 1 = -1$

$b = -1 + 1 \cdot -1 = -2$

$-2 - x_2 = \pm\theta = \pm 0'2$

$x_2 = -2'2$ R1

$x_2' = -1'8$ R1'

x(3) $y_{in} = 0 + -1 \rightarrow y = -1$ $y = t(3)$

x(4) $y_{in} = -1 \rightarrow y = -1$ $y \neq t(4)$

$w_1(new) = 0 + 1 \cdot 1 = 1$

$w_2(new) = 0 + 1 \cdot 1 = 1$

$b(new) = -1 + 1 = 0$

$x_1 + x_2 = \pm 0'2$

$x_2 = 0'2 - x_1$ R2

$x_2' = -0'2 - x_1$ R2'

R2' R2 x(3) x(4) R3

Tras esta 1ª época, los pesos son (0 1 1)

$w_1 = w_2 = b = 0$

x(1) $y_{in} = 0 \longrightarrow y = 0$ $y \neq t(1)$

$w_i(new) = w_i old + \alpha t x_i$

$w_1(new) = 0 + 1 \cdot (-1) \cdot 0 = 0$

$w_2(new) = 0 = 0$

$b(new) = 0 + 1 \cdot (-1) = -1$

no define recta

x(2) $y_{in} = -1 \longrightarrow y = -1$ $y \neq t(2)$

$w_1(new) = 0 + 1 \cdot (-1) \cdot 0 = 0$

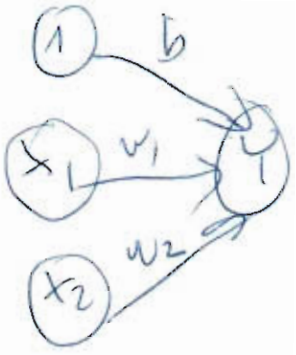
$w_2(new) = 0 + 1 \cdot (-1) \cdot 1 = -1$

$-2 - x_2 = \pm\theta = \pm 0'2$

$x_2 = -2'2$ R1

Adaline

Func. OR



$x(1) = (1 \quad 1)$ $\quad t(1) = 1$
$x(2) = (1 \quad -1)$ $\quad t(2) = 1$
$x(3) = (-1 \quad 1)$ $\quad t(3) = 1$
$x(4) = (-1 \quad -1)$ $\quad t(4) = -1$

$\alpha = 0'4$

Inic. pesos: $\underset{b}{(0'2} \quad \underset{w_1}{0'1} \quad \underset{w_2}{0'2)}$

$x(1) \rightarrow y_{in} = 0'2 + 0'1 + 0'2 = 0'5$

$w_{ij}(new) = w_{ij}(old) + \alpha(t_j - y_{in_j})x_i$

$w_1(new) = 0'1 + 0'4(0'5) \cdot 1 = 0'3$

$w_2(new) = 0'2 + 0'4 \cdot 0'5 \cdot 1 = 0'4$

$b(new) = 0'2 + 0'4 \cdot 0'5 = 0'4$

$x(2) \rightarrow y_{in} = 0'4 + 0'3 - 0'4 = 0'3$

$w_1(new) = 0'3 + 0'4 \cdot 0'7 \cdot 1 = 0'58$

$w_2(new) = 0'4 + 0'4 \cdot 0'7 \cdot (-1) = 0'12$

$b(new) = 0'4 + 0'28 = 0'68$

$x(3) \rightarrow y_{in} = 0'68 - 0'58 + 0'12 = 0'22$ $\quad 0'312$

$w_1(new) = 0'58 + 0'4 \cdot 0'78 \cdot (-1) = 0'27$

$w_2(new) = 0'12 + 0'4 \cdot 0'78 \cdot 1 = 0'43$

$b(new) = 0'68 + 0'4 \cdot 0'78 = 0'99$

$x(4) \rightarrow y_{in} = 0'99 - 0'27 - 0'43 = 0'29$ $\quad 0'516$

$0'7 + 0'4 \cdot (-1'79)(-1) = 0'78$

Inic. pesos: $\underset{b}{(0'2} \quad \underset{w_1}{0'1} \quad \underset{w_2}{0'2)}$

$x(1) \rightarrow y_{in} = 0'2 + 0'1 + 0'2 = 0'5$

$w_{ij}(new) = w_{ij}(old) + \alpha(t_j - y_{in_j})x_i$

$w_1(new) = 0'1 + 0'4(0'5) \cdot 1 = 0'3$

$w_2(new) = 0'2 + 0'4 \cdot 0'5 \cdot 1 = 0'4$

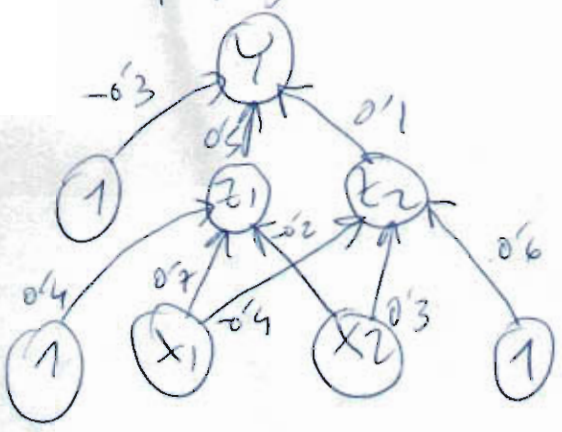
$b(new) = 0'2 + 0'4 \cdot 0'5 = 0'4$

$x(2) \rightarrow y_{in} = 0'4 + 0'3 - 0'4 = 0'3$

Para saber el valor correspondiente a (-1 1), usamos la red con la func. de activación escalón:

$$y_{in} = 0'5 - 0'5 + 0'5 = 0'5 \rightarrow y = 1$$

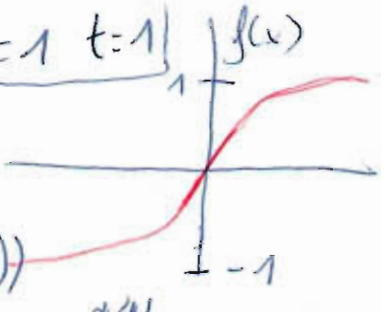
Retropropagación



$\alpha = 0'25$

Coef. y func. activación bipolares

$x_1 = -1 \quad x_2 = 1 \quad t = 1$



$f(x) = \frac{1-e^{-x}}{1+e^{-x}}$

$f'(x) = \frac{1}{2}(1+f(x))(1-f(x))$

$z_{in1} = 0'4 - 0'7 - 0'2 = -0'5 \rightarrow z_1 = f(-0'5) \simeq \frac{0'4}{1'88} \simeq -0'24$

$z_{in2} = 0'6 + 0'4 + 0'3 = 1'3 \rightarrow z_2 \simeq \frac{1'64}{2'64} \simeq 0'57$

$y_{in} = -0'3 + (0'5 \cdot 0'24) + (0'1 \cdot 0'57) = -0'3 + 0'12 + 0'057 = -0'036$

$Y = f(-0'36) = \frac{0'64}{1'88} = -0'18$

$\delta_y = (1 + 0'18) \cdot f'(-0'36) = 0'57 \cdot \frac{1}{2}(1+0'96)(1-0'96) = 0'57 \cdot 0'04 = 0'57$

$\delta in_1 = 0'57 \cdot 0'5 = 0'285 \quad \delta_1 = 0'285 \cdot f'(-0'5) = 0'285 \cdot 0'44 = 0'13$

$\delta in_2 = 0'57 \cdot 0'1 = 0'057 \quad \delta_2 = 0'057 \cdot f'(0'3) = 0'057 \cdot 0'33 = 0'02$

Actualiz. pesos $\quad \Delta w_0 = 0'25 \cdot 0'57 \cdot 1 = 0'14$

$\Delta w_{10} = 0'25 \cdot 0'57 \cdot (-0'24) = -0'03 \quad \Delta w_{02} = 0'25 \cdot 0'57 \cdot (0'57) = 0'08$

$\Delta v_{11} = 0'25 \cdot 0'13 \cdot (-1) = -0'01 \quad \Delta v_{12} = 0'25 \cdot (0'02) \cdot (-1) = -0'002$

$\Delta v_{21} = 0'25 \cdot 0'13 \cdot 1 = 0'01 \quad \Delta v_{22} = 0'25 (0'02) \cdot 1 = 0'002$

$v_{11} = 0'7 - 0'01 = 0'69 \quad v_{12} = -0'4 - 0'002 = -0'402$

$f'(x) = \frac{1}{2}(1+f(x))(1-f(x))$

$z_{in1} = 0'4 - 0'7 - 0'2 = -0'5 \rightarrow z_1 = f(-0'5) \simeq \frac{0'4}{1'88} \simeq -0'24$

$z_{in2} = 0'6 + 0'4 + 0'3 = 1'3 \rightarrow z_2 \simeq \frac{1'64}{2'64} \simeq 0'57$

$y_{in} = -0'3 + (0'5 \cdot 0'24) + (0'1 \cdot 0'57) = -0'3 + 0'12 + 0'057 = -0'036$

$Y = f(-0'36) = \frac{0'64}{1'88} = -0'18$

$\delta_y = (1 + 0'18) \cdot f'(-0'36) = 0'57 \cdot \frac{1}{2}(1+0'96)(1-0'96) = 0'57$

Ej. Madaline XOR

x_1	x_2	t
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

$v_1 = v_2 = b_3 = \frac{1}{2}$ $\alpha = 0'5$

	w_{11}	w_{12}	w_{21}	w_{22}	b_1	b_2
Iniz.	0'05	0'1	0'2	0'2	0'3	0'15
11	-0'725	-0'625	-0'525	-0'525	-0'475	-0'375

1.1 $x_1 = 1$ $x_2 = 1$

$z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21} = 0'3 + 0'05 + 0'2 = 0'55 \rightarrow 1$

$z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22} = 0'15 + 0'1 + 0'2 = 0'45 \rightarrow 1$

$y_{in} = b_3 + z_1 v_1 + z_2 v_2 = 0'5 + 0'5 + 0'5 = 1'5 \Rightarrow 1$

$t = -1 \rightarrow$ actualizar pesos z_1 y z_2

$\Delta w_{11} = 0'5(-1 - 0'55)1 = -0'77$ $\Delta w_{12} = 0'5(-1 - 0'45) \cdot 1 = -0'72$

~~$\Delta w_{21} = 0'5(1 - 0'55)1 = 0'225$~~ $\Delta w_{22} = 0'5(1 - 0'55)1 = 0'225$

$w_{11} = 0'05 - 0'77 = -0'725$ $w_{21} =$

$\Delta b_1 = 0'5(1-1) = 0$ $\Delta b_2 = 0'5 \cdot 0'45 = 0'225$

1.2. $x_1 = 1$ $x_2 = -1$

$z_{in1} = 0'3 + 0'05 - 0'425 < 0$ (-1) $z_{in2} = 0'375 + 0'425 - 0'425 = 0'375$ (1)

$y_{in} = 0'5 + 0'5 + 0'5 = 0'5 \rightarrow 1$ $t = 1 \rightarrow$ pesos no cambian

1.3 $x_1 = -1$ $x_2 = 1$

$z_{in1} = 0'3 - 0'05 - 0'425 = -0'175$ $z_{in2} = 0'375 - 0'425 + 0'425 = 0'375$

1.1 $x_1 = 1$ $x_2 = 1$

$z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21} = 0'3 + 0'05 + 0'2 = 0'55 \rightarrow 1$

$z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22} = 0'15 + 0'1 + 0'2 = 0'45 \rightarrow 1$

$y_{in} = b_3 + z_1 v_1 + z_2 v_2 = 0'5 + 0'5 + 0'5 = 1'5 \Rightarrow 1$

$t = -1 \rightarrow$ actualizar pesos z_1 y z_2

$\Delta w_{11} = 0'5(-1 - 0'55)1 = -0'77$ $\Delta w_{12} = 0'5(-1 - 0'45) \cdot 1 = -0'72$

~~$\Delta w_{21} = 0'5(1 - 0'55)1 = 0'225$~~ $\Delta w_{22} = 0'5(1 - 0'55)1 = 0'225$

Autoasoc. con func. umbral

FASE 1: "entrenar" la red para q. almacene el vector $V=(1,1,1,-1)$

$$\begin{pmatrix}1\\1\\1\\-1\end{pmatrix}(1\ 1\ 1\ -1)=\begin{pmatrix}1&1&1&-1\\1&1&1&-1\\1&1&1&-1\\-1&-1&-1&1\end{pmatrix}\xrightarrow[\text{autoconexiones}]{\text{eliminar}}W=\begin{pmatrix}0&1&1&-1\\1&0&1&-1\\1&1&0&-1\\-1&-1&-1&0\end{pmatrix}$$

FASE 2: Probar la red con los vectores $(1\ 0\ 0\ 0)$ y $(0,0\ 0\ -1)$

$\theta=0$

$x_0=(1\ 0\ 0\ 0)$ $x_1=x_0\cdot W=(0\ 1\ 1\ -1)$

$x_2=x_1\cdot W=(3\ 2\ 2\ -2)\to(1\ 1\ 1\ -1)=V\Rightarrow$ stop

$x_0=(0\ 0\ 0\ -1)$ $x_1=x_0\cdot W=(1\ 1\ 1\ 0)$

$x_2=x_1\cdot W=(2\ 2\ 2\ -3)=(1\ 1\ 1\ -1)$

$x_2=V\to$ stop

La red ha reconocido a los 2 vectores, cada uno de los cuales tenía 3 ~~errores~~ elementos ~~erróneos~~ desconocidos.

Probar con un vector con 2 errores: $V3=(-1\ -1\ 1\ -1)$

$x_0=(-1\ -1\ 1\ -1)$ $x_1=x_0\cdot W=(1\ 1\ -1\ 1)$

$x_2=x_1\cdot W=(-1\ -1\ 1\ -1)=x_0$ Sta.

$$\begin{pmatrix}-1&-1&-1&1\end{pmatrix}\qquad\begin{pmatrix}-1&-1&-1&0\end{pmatrix}$$

FASE 2: Probar la red con los vectores $(1\ 0\ 0\ 0)$ y $(0,0\ 0\ -1)$

$\theta=0$

$x_0=(1\ 0\ 0\ 0)$ $x_1=x_0\cdot W=(0\ 1\ 1\ -1)$

$x_2=x_1\cdot W=(3\ 2\ 2\ -2)\to(1\ 1\ 1\ -1)=V\Rightarrow$ stop

$x_0=(0\ 0\ 0\ -1)$ $x_1=x_0\cdot W=(1\ 1\ 1\ 0)$

Red autoasoc.

Obtener la matriz de pesos para almacenar los sgtes. vectores (ortogonales)

$$(1\ 1\ 1\ 1)$$
$$(1\ 1\ -1\ -1)$$
$$(1\ -1\ 1\ -1)$$
$$(1\ -1\ -1\ 1)$$

$$\begin{matrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{matrix}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{matrix} b_{11}\ b_{12}\ b_{13}\ b_{14} \\ (1\ 1\ 1\ 1) \end{matrix} = \begin{matrix} \\ \\ \\ W_1 \end{matrix}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}\left\{\begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}(1\ 1\ -1\ -1) = \begin{matrix} \\ \\ \\ W_2 \end{matrix}\begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix}\right.$$

$4\times1 \cdot 1\times4 \to 4\times4$

$K,l \to a_{k1}b_{1l} + a_{k2}b_{2l} + \dots a_{kn}b_{nl}$

$1,1 \to a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$

$$\begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}(1\ -1\ 1\ -1) = \begin{matrix} \\ \\ \\ W_3 \end{matrix}\begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix}\left\{\begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}(1\ -1\ -1\ 1) = \begin{matrix} \\ \\ \\ W_4 \end{matrix}\begin{pmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\right.$$

$$W_1 + W_2 = \begin{pmatrix} 2 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{pmatrix}$$

$$(1\ -1\ -1\ 1)$$

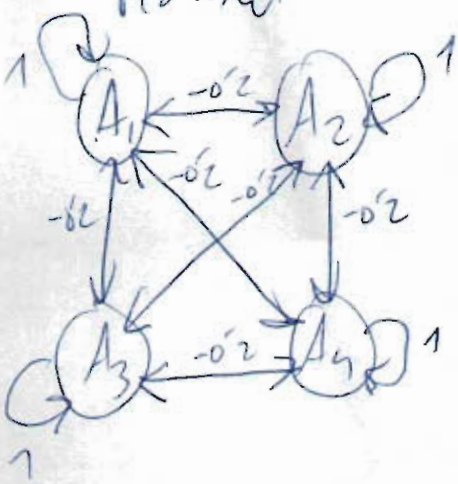
Comprobamos si cumplen los vectores 1 y 2:

$$\begin{matrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{matrix}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{matrix} b_{11}\ b_{12}\ b_{13}\ b_{14} \\ (1\ 1\ 1\ 1) \end{matrix} = \begin{matrix} \\ \\ \\ W_1 \end{matrix}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}\left\{\begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}(1\ 1\ -1\ -1) = \begin{matrix} \\ \\ \\ W_2 \end{matrix}\begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix}\right.$$

$4\times1 \cdot 1\times4 \to 4\times4$

$K,l \to a_{k1}b_{1l} + a_{k2}b_{2l} + \dots a_{kn}b_{nl}$

Maxnet



0 $x = (0'2 \quad 0'4 \quad 0'6 \quad 0'8)$

1 $x_1(1) = (x_1(0) - 0'2(0'4+0'6+0'8)) = 0'07 \to 0$

$x_2(1) = (0'4 - 0'2(0'2+0'6+0'8)) = 0'08 \to 0'08$

$x_3(1) = (0'6 - 0'2(0'2+0'4+0'8)) = 0'32 \to 0'32$

$x_4(1) = (0'8 - 0'2(0'2+0'4+0'6)) = 0'56 \to 0'56$

$x = (0 \quad 0'08 \quad 0'32 \quad 0'56)$

2 $x_1(2) = 0 - K = -K \to 0$

$x_2(2) = [0'08 - 0'2(0'32+0'56)] = -0'168 \to 0$

$x_3(2) = (0'32 - 0'2(0+0'08+0'56)) = 0'192$

$x_4(2) = (0'56 - 0'2(0+0'08+0'32)) = 0'48$

$x = (0 \quad 0 \quad 0'192 \quad 0'48)$

3 $x_3(3) = [0'192 - 0'2(0'48)) = 0'096$

$x_4(3) = (0'48 - 0'2 \cdot 0'192)] = 0'442$

$x = (0 \quad 0 \quad 0'096 \quad 0'442)$

4 $x = (0 \quad 0 \quad 0'008 \quad 0'422)$

5 $x = (0 \quad 0 \quad 0 \quad 0'421)$

1

$x = (0 \quad 0'08 \quad 0'32 \quad 0'56)$

2 $x_1(2) = 0 - K = -K \to 0$

$x_2(2) = [0'08 - 0'2(0'32+0'56)] = -0'168 \to 0$

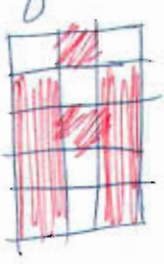
$x_3(2) = (0'32 - 0'2(0+0'08+0'56)) = 0'192$

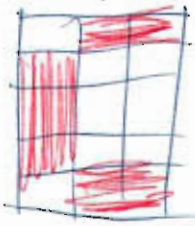
$x_4(2) = (0'56 - 0'2(0+0'08+0'32)) = 0'48$

$x = (0 \quad 0 \quad 0'192 \quad 0'48)$

BAM

Se quieren asociar las letras A y C con los códigos (-1,1) y (1,1), respectivamente





Las letras se codifican bipolares. P.e., ~~la matriz~~ para el vector para A sería:

(-1, 1, -1, 1, -1, 1, 1, 1, 1, 1, -1, 1, 1, -1, 1)

FASE 1: inicializar pesos red:

$$[A]\begin{pmatrix}-1\\1\\-1\\1\\-1\\1\\1\\1\\1\\1\\-1\\1\\1\\-1\\1\end{pmatrix}(-1\ 1)=\begin{pmatrix}1&-1\\-1&1\\1&-1\\-1&1\\1&-1\\-1&1\\-1&1\\-1&1\\-1&1\\-1&1\\1&-1\\-1&1\\-1&1\\1&-1\\-1&1\end{pmatrix}=[W_1]$$

$$[C]\begin{pmatrix}-1\\1\\1\\1\\-1\\-1\\1\\-1\\-1\\-1\\-1\\1\\-1\\-1\\-1\\1\end{pmatrix}(1\ 1)=\begin{pmatrix}-1&-1\\1&1\\1&1\\1&1\\-1&-1\\-1&-1\\1&1\\-1&-1\\-1&-1\\-1&-1\\1&1\\-1&-1\\-1&-1\\-1&-1\\1&1\end{pmatrix}=[W_2]$$

$$W=\frac{1}{2}W_1+W_2=\begin{pmatrix}0&-2\\0&2\\2&0\\0&2\\0&-2\\-2&0\\0&2\\-2&0\\-2&0\\0&2\\0&-2\\-2&0\\-2&0\\2&0\\0&2\end{pmatrix}$$

FASE 2: recuperación info.

(-1,1) → $x_{in}=y$ $W^T=$(-2 2 -2 2 -2 2 2 2 2 2 -2 2 2 -2 2)

Vemos que a partir del código de A se recupera la propia A

(-1, 1, -1, 1, -1, 1, 1, 1, 1, 1, -1, 1, 1, -1, 1)

FASE 1: inicializar pesos red:

$$[A]\begin{pmatrix}-1\\1\\-1\\1\\-1\\1\\1\\1\\1\\1\\-1\\1\\1\\-1\\1\end{pmatrix}(-1\ 1)=\begin{pmatrix}1&-1\\-1&1\\1&-1\\-1&1\\1&-1\\-1&1\\-1&1\\-1&1\\-1&1\\-1&1\\1&-1\\-1&1\\-1&1\\1&-1\\-1&1\end{pmatrix}$$

$$[C]\begin{pmatrix}-1\\1\\1\\1\\-1\\-1\\1\\-1\\-1\\-1\\-1\\1\\-1\\-1\\-1\\1\end{pmatrix}(1\ 1)=\begin{pmatrix}-1&-1\\1&1\\1&1\\1&1\\-1&-1\\-1&-1\\1&1\\-1&-1\\-1&-1\\-1&-1\\1&1\\-1&-1\\-1&-1\\-1&-1\\1&1\end{pmatrix}$$

$$W=\frac{1}{2}W_1+W_2=\begin{pmatrix}0&-2\\0&2\\2&0\\0&2\\0&-2\\-2&0\end{pmatrix}$$

Sombrero mexicano

$$f(x)=\begin{cases}0 & x<0\\ x & 0\le x\le 2\\ 2 & 2<x\end{cases}$$

$R_1=1,\ C_1=0'6$

$R_2=2,\ C_2=-0'4$

(0) (0'5) (0'8) (1) (0'8) (0'5) (0)

$t=0$ $X=(0,\ 0'5,\ 0'8,\ 1,\ 0'8,\ 0'5,\ 0)$

$t=1$ $x_i(1)$

$x_{in1}(1)=0'6(0+0'5)-0'4\cdot 0'8=-0'02\longrightarrow 0$

$x_{in2}(1)=0'6(0+0'5+0'8)-0'4\cdot 1=0'38\longrightarrow 0'38$

$x_{in3}(1)=0'6(0'5+0'8+1)-0'4\cdot(0+0'8)=1'06\to 1'06$

$x_{in4}(1)=0'6(0'8+1+0'8)-0'4(0'5+0'5)=1'16\to 1'16$

$x_{in5}(1)=0'6(1+0'8+0'5)-0'4(0'8+0)=1'06\to 1'06$

$x_{in6}(1)=0'6(0'8+0'5+0)-0'4\cdot 1=0'38\longrightarrow 0'38$

$x_{in7}(1)=0'6(0'5+0)-0'4\cdot 0'8=-0'02\longrightarrow 0$

$X=(0\ \ 0'38\ \ 1'06\ \ 1'16\ \ 1'06\ \ 0'38\ \ 0)$

$t=2$

$x_{in1}(2)=0'6(0+0'38)-0'4\cdot 1'06=-0'196\longrightarrow 0$

$x_{in2}(2)=0'6(0+0'38+1'06)-0'4(1'16)=0'39\longrightarrow 0'39$

$x_{in3}(2)=0'6(0'38+1'06+1'16)-0'4(0+1'06)=1'14\longrightarrow 1'14$

$x_{in4}(2)=0'6(1'06+1'16+1'06)-0'4(0'38+0'38)=1'136\longrightarrow 1'14$

$x_{in5}(2)=$ $0'39\to 0'39$

$x_{in6}(2)=$ $-0'196\longrightarrow 0$

$x_{in7}(2)=$

$t=1$

$x_{in1}(1)=0'6(0+0'5)-0'4\cdot 0'8=-0'02\longrightarrow 0$

$x_{in2}(1)=0'6(0+0'5+0'8)-0'4\cdot 1=0'38\longrightarrow 0'38$

$x_{in3}(1)=0'6(0'5+0'8+1)-0'4\cdot(0+0'8)=1'06\to 1'06$

$x_{in4}(1)=0'6(0'8+1+0'8)-0'4(0'5+0'5)=1'16\to 1'16$

$x_{in5}(1)=0'6(1+0'8+0'5)-0'4(0'8+0)=1'06\to 1'06$

$x_{in6}(1)=0'6(0'8+0'5+0)-0'4\cdot 1=0'38\longrightarrow 0'38$

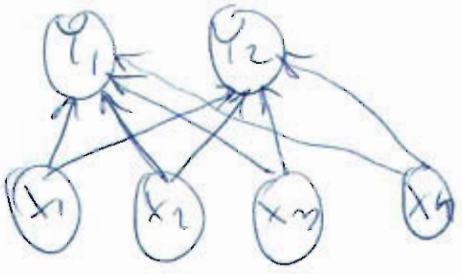
$x_{in7}(1)=0'6(0'5+0)-0'4\cdot 0'8=-0'02\longrightarrow 0$

$X=(0\ \ 0'38\ \ 1'06\ \ 1'16\ \ 1'06\ \ 0'38\ \ 0)$

Hamming

Vectores ejemplares: $e(1) = (1, -1, -1, -1)$
$e(2) = (-1, -1, -1, 1)$

Almacenar los vectores en la red



$w_{ij} = \frac{e_i(j)}{2}$

$$W = \begin{pmatrix} 0'5 & -0'5 \\ -0'5 & -0'5 \\ -0'5 & -0'5 \\ -0'5 & 0'5 \end{pmatrix}$$

Bias → $b_1 = b_2 = \frac{4}{2} = 2$

Vectores de entrada

$x = (1, 1, -1, -1)$ → $y_{in1} = b_1 + \sum_1^4 x_i w_{i1} = 2 + (0'5 - 0'5 + 0'5 + 0'5)$
$d = (0, 3)$ $= 3$

$y_{in2} = 2 + (-0'5 - 0'5 + 0'5 - 0'5) = 1$

~~Aplicar Maxnet~~ $y = y_{in} = (3\ 1)$

Aplicar Maxnet → Y_1 → x se parece + a Y_1

$x = (-1, -1, 1, 1)$ → $y_{in1} = 2 + (-0'5 + 0'5 - 0'5 - 0'5) = 1$
$d = (3, 0)$ $y_{in2} = 2 + (0'5 + 0'5 - 0'5 + 0'5) = 3$

$y = y_{in} = (1\ 3)$

Maxnet → Y_2 → x se parece más a Y_2

$$W = \begin{pmatrix} -0'5 & -0'5 \\ -0'5 & 0'5 \end{pmatrix}$$

Bias → $b_1 = b_2 = \frac{4}{2} = 2$

Vectores de entrada

$x = (1, 1, -1, -1)$ → $y_{in1} = b_1 + \sum_1^4 x_i w_{i1} = 2 + (0'5 - 0'5 + 0'5 + 0'5)$
$d = (0, 3)$ $= 3$

$y_{in2} = 2 + (-0'5 - 0'5 + 0'5 - 0'5) = 1$

Hopfield discreta

Red q. almacena el vector $(1,1,1,0)$. $\theta_i = 0$

Aprendizaje del vector (no supervisado). Se pasa el vector a bipolar $(1\ 1\ 1\ -1)$

$$\begin{pmatrix}1\\1\\1\\-1\end{pmatrix}(1\ 1\ 1\ -1) = \begin{pmatrix} x_0 & 1 & 1 & -1 \\ 1 & x_0 & 1 & -1 \\ 1 & 1 & x_0 & -1 \\ -1 & -1 & -1 & x_0 \end{pmatrix} = W$$

← diagonal a 0

(si hubiera + patrones, habría q. sumar las respectivas matrices)

Vector entrada → $x = (0, 0, 1, 0)$ ← 2 valores desconocidos ~~errores~~

(0) Elección aleatoria de unid. → y_1

$y_{in1} = 0 + (0 + 0 + 1 + 0) = 1 \rightarrow 1$

$y = (1, 0, 1, 0)$

(1) Elección → y_4

$y_{in4} = 0 + (-1 + 0 - 1 + 0) = -2 \rightarrow 0$

$y = (1, 0, 1, 0)$

(2) Elección → y_3

$y_{in3} = 1 + (1 + 0 + 0 + 0) = 2 \rightarrow 1$

$y = (1, 0, 1, 0)$

(2) Elección → y_2

$$\begin{pmatrix} x_0 & 1 & 1 & -1 \\ 1 & x_0 & 1 & -1 \\ 1 & 1 & x_0 & -1 \\ -1 & -1 & -1 & x_0 \end{pmatrix}$$

← diagonal a 0

(si hubiera + patrones, habría q. sumar las respectivas matrices)

Vector entrada → $x = (0, 0, 1, 0)$ ← 2 valores desconocidos ~~errores~~

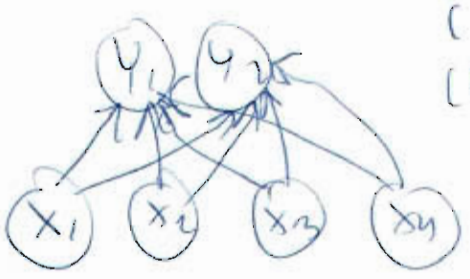
(0) Elección aleatoria de unid. → y_1

$y_{in1} = 0 + (0 + 0 + 1 + 0) = 1 \rightarrow 1$

SOM

Vectores de entrada: (1,1,0,0)
(0,0,0,1)
(1,0,0,0)
(0,0,1,1)

$\alpha(0) = 0'6$
$\alpha(t+1) = 0'5 \cdot \alpha(t)$
Radio 0
Sin vecindad
Nº máx. clusters = 2



1) Inicializ. matriz W (aleatorio)

$$W = \begin{pmatrix} 0'2 & 0'8 \\ 0'6 & 0'4 \\ 0'5 & 0'7 \\ 0'9 & 0'3 \end{pmatrix}$$

$x = (1,1,0,0)$ Buscar neurona + próxima

$\alpha = 0'6$

$D(1) = (0'2-1)^2 + (0'6-1)^2 + 0'5^2 + 0'9^2 = 1'86$

$\rightarrow D(2) = (0'8-1)^2 + (0'4-1)^2 + 0'7^2 + 0'3^2 = 0'98$

Actualizar pesos de Y_2

$\Delta w_{i2} = \alpha(x_i - w_{i2}(old))$

$\Delta w_{12} = 0'6(1-0'8) = 0'12$ $\quad \Delta w_{22} = 0'6(1-0'4) = 0'36$

$\Delta w_{32} = 0'6(0-0'7) = -0'42$ $\quad \Delta w_{42} = 0'6 \cdot (0-0'3) = -0'18$

$$W^T = \begin{pmatrix} 0'2 & 0'6 & 0'5 & 0'9 \\ 0'12 & 0'36 & 0'28 & 0'12 \end{pmatrix}$$

$x = (0,0,0,1)$ $\rightarrow D(1) = (0'2-0)^2 + 0'6^2 + 0'5^2 + (0'9-1)^2 = 0'66$

$\alpha = 0'6$ $\quad D(2) = 0'92^2 + 0'76^2 + 0'28^2 + (0'12-1)^2 = 2'27$

$\Delta w_{11} = 0'6(0-0'2) = -0'12$ $\quad \Delta w_{21} = 0'6(0-0'6) = -0'36$

$\Delta w_{31} = 0'6(0-0'5) = 0'30$ $\quad \Delta w_{41} = 0'6(1-0'9) = 0'06$

$$W^T = \begin{pmatrix} 0'08 & 0'24 & 0'82 & 0'36 \\ 0'42 & 0'28 & 0'18 & 0'12 \end{pmatrix}$$

1) Inicializ. matriz W (aleatorio)

$$W = \begin{pmatrix} 0'2 & 0'8 \\ 0'6 & 0'4 \\ 0'5 & 0'7 \\ 0'9 & 0'3 \end{pmatrix}$$

$x = (1,1,0,0)$ Buscar neurona + próxima

$\alpha = 0'6$

$D(1) = (0'2-1)^2 + (0'6-1)^2 + 0'5^2 + 0'9^2 = 1'86$

$\rightarrow D(2) = (0'8-1)^2 + (0'4-1)^2 + 0'7^2 + 0'3^2 = 0'98$

Actualizar pesos de Y_2

$\Delta w_{i2} = \alpha(x_i - w_{i2}(old))$

$X = (0, 0, 1, 1) \rightarrow D(1) = (0'08)^2 + (0'24)^2 + (0'2-1)^2 + (0'96-1)^2 =$
$= 0'0064 + \quad = 0'70$

$D(2) = 0'97^2 + 0'3^2 + (0'12-1)^2 + (0'05-1)^2 =$
$= 0'94 + 0'09 + 0'77 + 0'9 = 2'70$

$\Delta w_{11} = 0'6(0 - 0'08) = -0'048 \quad \Delta w_{21} = 0'6(0 - 0'24) = 0'144$

$\Delta w_{31} = 0'6(1 - 0'12) = 0'528 \quad \Delta w_{41} = 0'6(1 - 0'96) = 0'024$

$$W^+ = \begin{pmatrix} 0'32 & 0'09 & 0'68 & 0'98 \\ 0'97 & 0'3 & 0'12 & 0'05 \end{pmatrix}$$

Después de esta época se actualiza $\alpha = 0'3$

Después de la 2ª época, la matriz es

$$W(2) = \begin{pmatrix} 0'016 & 0'98 \\ 0'047 & 0'36 \\ 0'630 & 0'05 \\ 0'991 & 0'024 \end{pmatrix} \qquad W(100) = \begin{pmatrix} 0 & 1 \\ 0 & 0'49 \\ 0'51 & 0 \\ 1 & 0 \end{pmatrix}$$

x_1, x_3

x_2, x_4

LVQ

Vectores de referencia

Vectores	Clase
(1, 1, 0, 0)	1
(0, 0, 0, 1)	2
(0, 0, 1, 1)	2
(1, 0, 0, 0)	1

$$W = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$x = (0, 0, 1, 1)$

$D(1) = \| \cdot 1^2 + 1^2 - 1^2 - 1^2 \| = 4$

$D(2) = \| 0 - 0, 1^2 - 0 \| = 1 \leftarrow$

Clase esperada = clase obtenida → sumar (reforzar)

$$W^+ = \begin{pmatrix} 0'32 & 0'09 & 0'68 & 0'98 \\ 0'97 & 0'3 & 0'12 & 0'05 \end{pmatrix}$$

Después de esta época se actualiza $\alpha = 0'3$

Después de la 2ª época, la matriz es

$$W(2) = \begin{pmatrix} 0'016 & 0'98 \\ 0'047 & 0'36 \\ 0'630 & 0'05 \\ 0'991 & 0'024 \end{pmatrix} \qquad W(100) = \begin{pmatrix} 0 & 1 \\ 0 & 0'49 \\ 0'51 & 0 \\ 1 & 0 \end{pmatrix}$$

x_1, x_3

clase esperada = clase obtenida

$w_1(new) = (1\ 1\ 0\ 0) + 0'1\,(0\ -1\ 0\ 0) = (1\ 0'9\ 0\ 0)$

$$w = \begin{pmatrix} 1 & 0'9 & 0 & 0 \\ 0 & 0 & 0'1 & 1 \end{pmatrix}$$

$x = (0\ 1\ 1\ 0)$

$D(1) = \|1\ -0'1\ -1\ 0\| \simeq \sqrt{2}$ ←

$D(2) = \|0\ -1\ -0'9\ 1\| \simeq \sqrt{3}$

clase esperada ≠ clase obtenida

$w_1(new) = (1\ 0'9\ 0\ 0) - 0'1\,(-1\ 0'1\ 1\ 0) = (1'1\ 0'89\ -0'1\ 0)$

$$w = \begin{pmatrix} 1'1 & 0'89 & -0'1 & 0 \\ 0 & 0 & 0'1 & 1 \end{pmatrix}$$

Hasta aquí la 1ª época.

$D(2) = \|0\ -1\ -0'9\ 1\| \simeq \sqrt{3}$

clase esperada ≠ clase obtenida

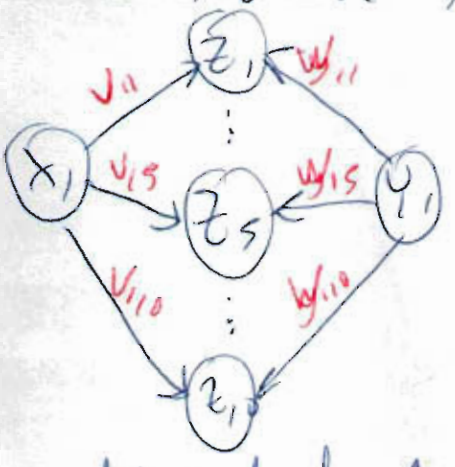
$w_1(new) = (1\ 0'9\ 0\ 0) - 0'1\,(-1\ 0'1\ 1\ 0) = (1'1\ 0'89\ -0'1\ 0)$

$$w = \begin{pmatrix} 1'1 & 0'89 & -0'1 & 0 \\ 0 & 0 & 0'1 & 1 \end{pmatrix}$$

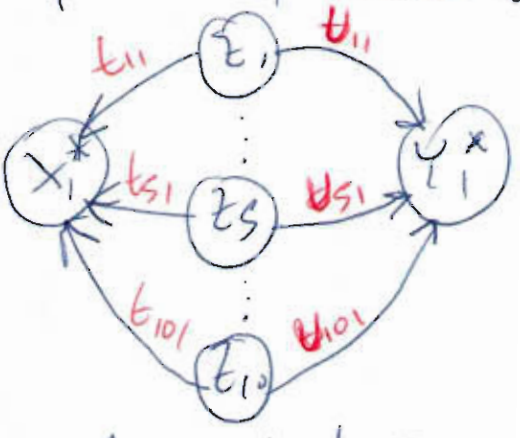
Hasta aquí la 2ª época.

Contrapropagación

Se desea un aproximador de la func. $f(x) = \frac{1}{x}$
Se tienen 1000 ptos. en $[0'1, 10]$ para entrenar la red. Se estima q. con 10 neuronas en la capa de cluster es suficiente. Para la capa de entrada y salida sólo se necesita 1 ~~cluster~~ neurona.



Red en la fase 1



Red en la fase 2

$X = 10 \quad y = f(x) = 0'1$

FASE 1

1) Inicializ. pesos a valores aleatorios dentro de $[0'1, 1]$

a=b=α=β=0'5 $v = (0'8 \; 3'1 \; 5'2 \; 6'4 \; 2'2 \; 8'1 \; 1'5 \; 9'1 \; 4'4 \; 9'9)$

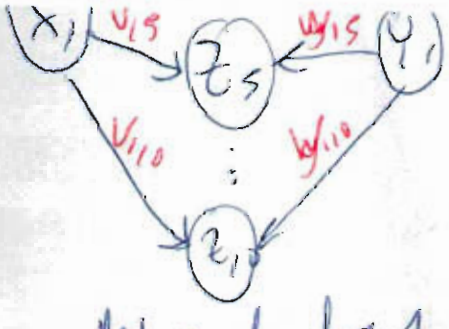
$w = (9'9 \; 4'4 \; 9'1 \; 1'5 \; 8'1 \; 2'2 \; 6'4 \; 5'2 \; 3'1 \; 0'2)$

$t = (0'3 \; \ldots \; 8) \quad u = (7 \; \ldots \; 2)$

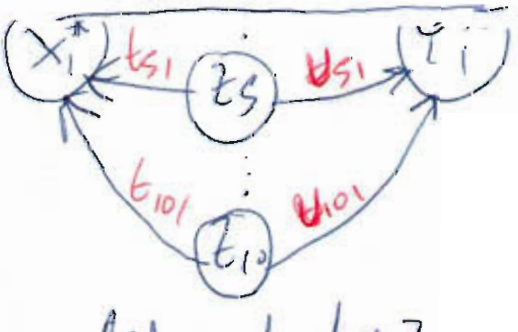
2) Para cada pareja de entrada ajustar los pesos

$X = 10 \quad y = 0'1 \quad D_1 = \sqrt{(10 - 0'2)^2 + (0'1 - 9'9)^2} \simeq 15$

...



Red en la fase 1



Red en la fase 2

de cluster es suficiente. Para la capa de entrada y salida sólo se necesita 1 ~~cluster~~ neurona.

Unid. del cluster z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8 z_9 z_{10}

v [0'11 0'14 0'2 0'3 0'6 1'6 3'3 5 7 9)

w (9 7 5 3'3 1'6 0'6 0'3 0'2 0'14 0'11)

FASE 2 Se pasan de nuevo los vectores de entrenamiento

$x=1$ $y=1$ $D(1) = \sqrt{(1-0'11)^2 + (1-9)^2} \simeq 8$

...

$\rightarrow D(5) = \sqrt{(1-0'6)^2 + (1-1'6)^2} \simeq 0'2$

$D(6) = \simeq 0'2$

...

$D(10) = \sqrt{(1-9)^2 + (1-0'11)^2} \simeq 8$

La unid. ganadora es z_5. Se actualizan los pesos:

$\Delta v_{15} = 0'005 \cdot (1-0'6) = 0'002$ $v_{15} = 0'602$

$\Delta w_{15} = 0'005(1-1'6) = 0'003$ $w_{15} = 1'603$

$\Delta t_{51} = 0'5(1-2) = -0'5$ $t_{51} = 1'5$

$\Delta u_{51} = 0'5(1-0'5) = 0'25$ $u_{51} = 0'25$

Actualizar tasa aprendizaje: $\alpha = 0'004 = \beta$; $a = 0'49 = b$

Tras pasar los 1000 patrones, los pesos quedan así:

$$\begin{matrix} v \\ w \\ t \\ u \end{matrix} \begin{pmatrix} 0'11 & 0'14 & 0'2 & 0'3 & 0'6 & 1'6 & 3'3 & 5 & 7 & 9 \\ 9 & 7 & 5 & 3'3 & 1'6 & 0'6 & 0'3 & 0'2 & 0'14 & 0'11 \\ 0'11 & 0'14 & 0'2 & 0'3 & 0'6 & 1'6 & 3'3 & 5 & 7 & 9 \\ 9 & 7 & 5 & 3 & 1'6 & 0'6 & 0'3 & 0'2 & 0'14 & 0'11 \end{pmatrix}$$

...

$\rightarrow D(5)' = \sqrt{(1-0'6)^2 + (1-1'6)^2} \simeq 0'2$

$D(6) = \simeq 0'2$

...

$D(10) = \sqrt{(1-9)^2 + (1-0'11)^2} \simeq 8$

La unid. ganadora es z_5. Se actualizan los pesos:

$\Delta v_{15} = 0'005 \cdot (1-0'6) = 0'002$ $v_{15} = 0'602$

$\Delta w_{15} = 0'005(1-1'6) = 0'003$ $w_{15} = 1'603$

$\Delta t_{51} = 0'5(1-2) = -0'5$ $t_{51} = 1'5$

1) Buscar la unid. del cluster + próxima

$P_1 = \sqrt{(0'12 - 0'11)^2 + (0 - 9)^2} \simeq 9$

$\vdots$

$\rightarrow P_6 = \sqrt{(0'12 - 1'6)^2 + (0 - 0'6)^2} \simeq 1'6$

$\vdots$

$P_{10} = \sqrt{(0'12 - 9)^2 + (0 - 0'11)^2} \simeq 8'9$

2) Calcular la aprox.:

$x^* = t_6 = 1'6 \qquad y^* = 0'6$

La aprox. es burda, pq. hemos usado un valor desconocido de "y" para la búsqueda. Si buscamos en el cluster sólo con x,

$\rightarrow P_1 = \sqrt{(0'12 - 0'11)^2} = 0'01$

$\vdots$

$P_6 = \sqrt{(0'12 - 1'6)^2} = 1'48$

$\vdots$

$P_{10} = \sqrt{(0'12 - 9)^2} = 8'88$

Calculando la aprox. con z_1,

$x^* = t_1 = 0'11$

$y^* = u_1 = 9$

2) Calcular la aprox.:

$x^* = t_6 = 1'6 \qquad y^* = 0'6$

La aprox. es burda, pq. hemos usado un valor desconocido de "y" para la búsqueda. Si buscamos en el cluster sólo con x,

$\rightarrow P_1 = \sqrt{(0'12 - 0'11)^2} = 0'01$

$\vdots$

$P_6 = \sqrt{(0'12 - 1'6)^2} = 1'48$