

Inteligencia Artificial (IA)

clase 1

Libros: Freeman
Spikeret.

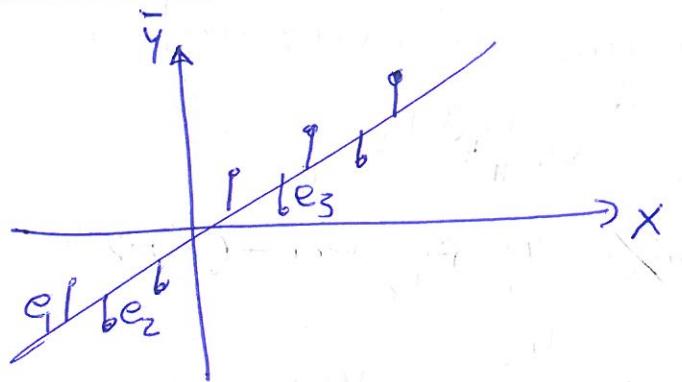
Profesor: Antonio Amorán

amoranUNI@gmail.com

Regressión Lineal

Datos

X	\bar{Y}
x_1	\bar{y}_1
:	:
x_n	\bar{y}_n



$$\text{Función de costo} \quad J = \frac{1}{2} e_1^2 + \dots + \frac{e_n^2}{2}$$

$$= [(ax_1 + b - \bar{y}_1)^2 + \dots] \frac{1}{2}$$

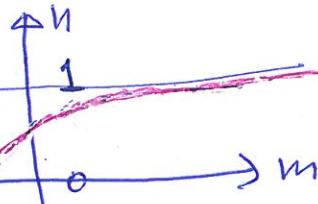
Minimizar J para parámetros a y b

$\frac{\partial J}{\partial a} = 0 = a \sum_{k=1}^n x_k^2 + b \sum_{k=1}^n x_k - \sum_{k=1}^n x_k \bar{y}_k$
$\frac{\partial J}{\partial b} = 0 = a \sum_{k=1}^n x_k + b n - \sum_{k=1}^n \bar{y}_k$

Solución exacta de a y b .

Métodos iterativos: Usan funciones de transformación debido al aprendizaje de neuronas

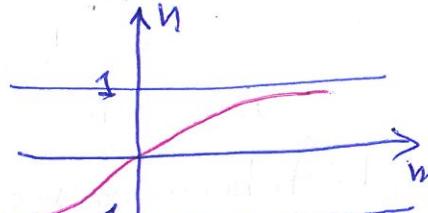
Sigmoides 1



arg8(a)

$$n(n-1) = f'(n) \text{ en función de } n$$

Sigmoides 2

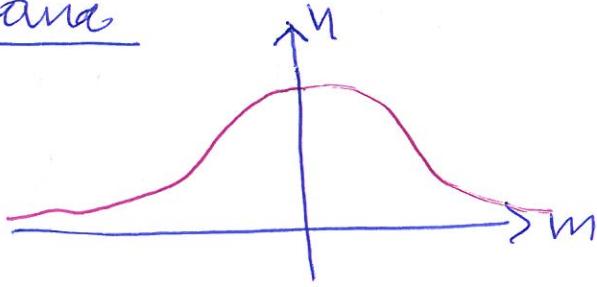


$$n = \frac{2}{1 + e^{-m}} - 1$$

$f(m) = \text{en función de } n?$

$$\frac{1}{1 + e^{-m}} = \frac{1}{2} \Rightarrow m = 0$$

Gaussianas

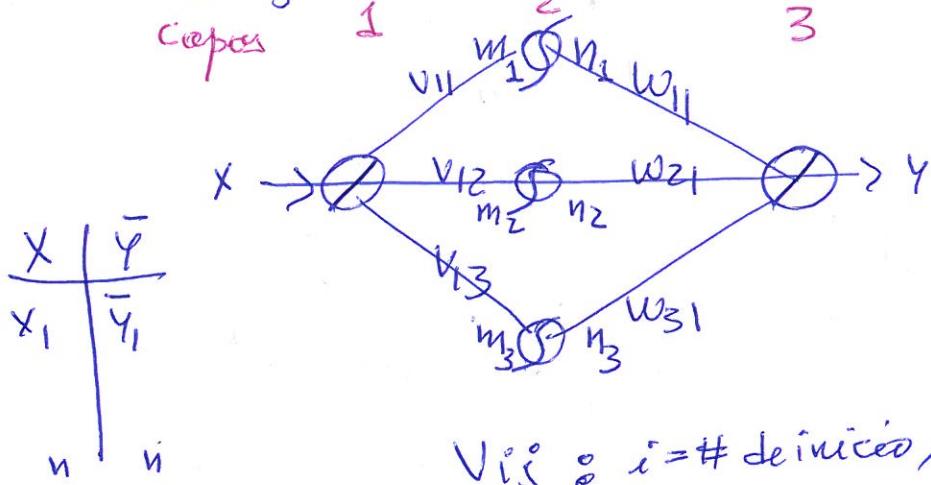


$$n = e^{-m^2}$$

$$\frac{\partial n}{\partial m} = e^{-m^2} (-2m)$$

$$= -2mn$$

Se utiliza una red neuronal de 3 ó más capas



$V_{ij} \Rightarrow i = \# \text{ de inicio}, j = \# \text{ de fin}$

$W_{ij} \Rightarrow 11$

$\phi \Rightarrow$ Transformación lineal

$\phi \Rightarrow$ Transformación sigmoidal o logística

Iteraciones para ~~calcular~~ calcular a y b

Método del Gradiente Descendente

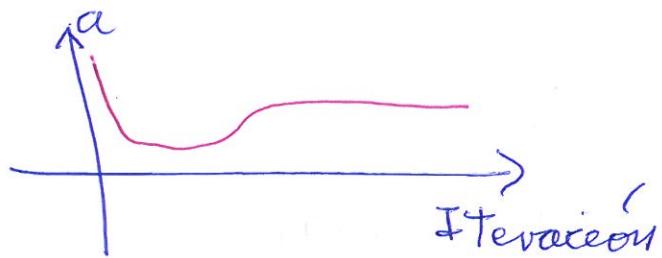
$$a = a - \eta \frac{\partial J}{\partial a}$$

anterior

$$b = b - \eta \frac{\partial J}{\partial b}$$

Leta = paso, se va
probando: a_1, a_2, \dots

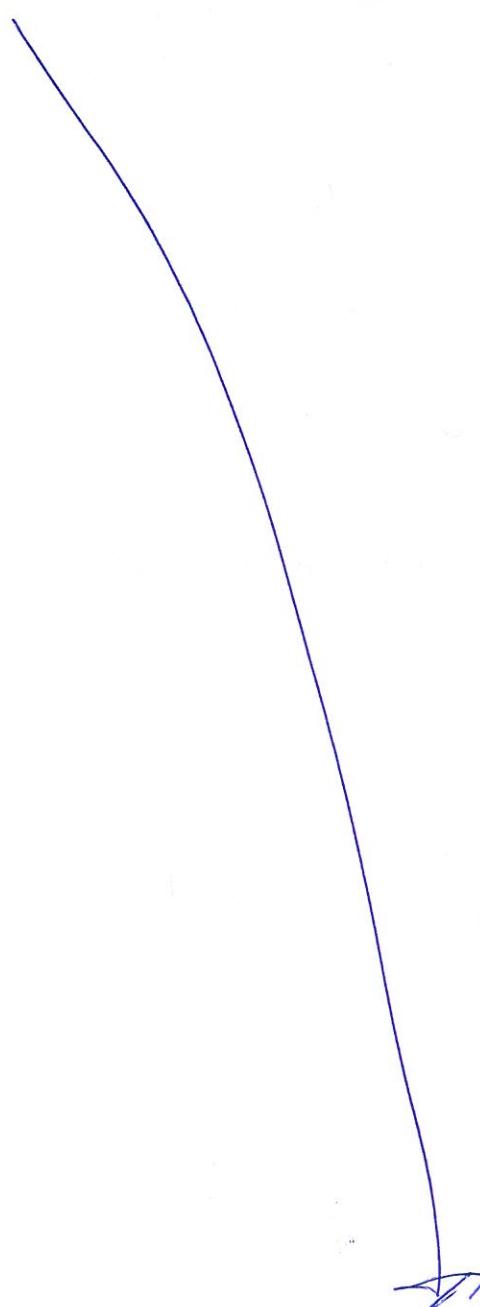
a y b deben converger.



Si no converge se disminuye η

~~Si las iteraciones son suficientes para llegar~~

minimizan a $J = \sum_k (\hat{y}_k - y_k)^2 \frac{1}{2}$



Todos son Regresión ~~lineal~~ reductiva como extensión de la y exacta y iterativa.

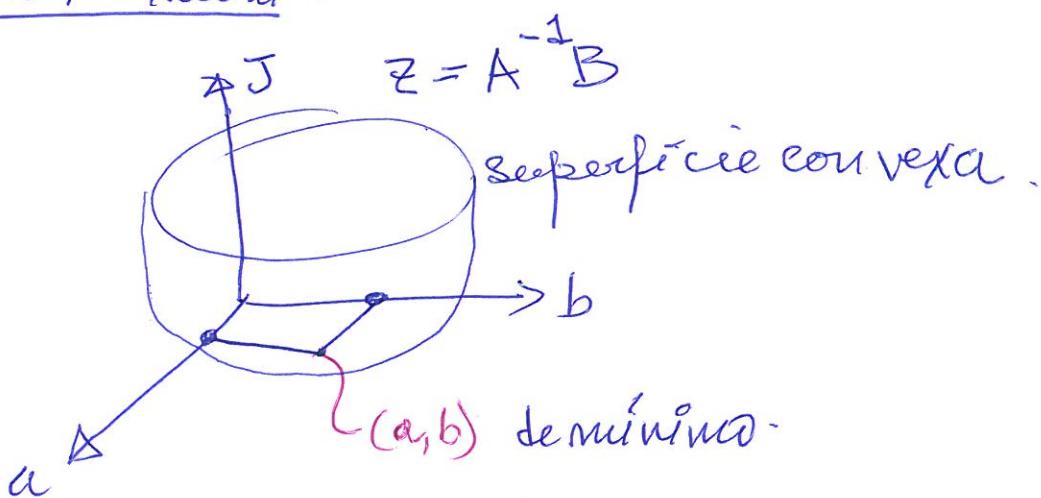
Regresión Línea (Notación matricial)

X	Y
x_1	y_1

$$J = \frac{1}{2} \sum_{k=1}^n \text{error}_k^2$$

$$\begin{bmatrix} \frac{\partial J}{\partial a} \\ \frac{\partial J}{\partial b} \end{bmatrix} = \begin{bmatrix} \sum x_k^2 & \sum x_k \\ \sum x_k & n \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} - \begin{bmatrix} \sum x_k \bar{y}_k \\ \bar{y}_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solución exacta: A Z B

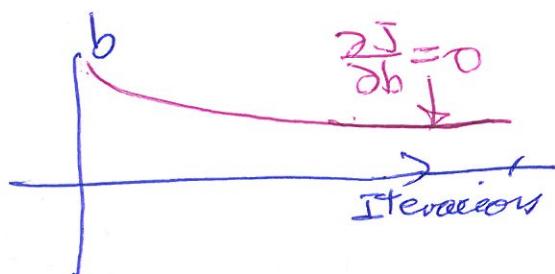
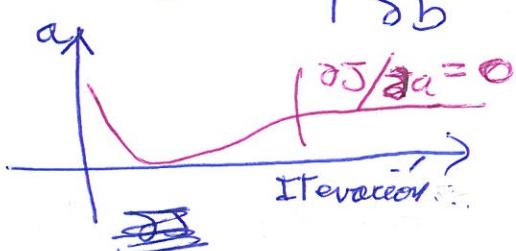


Solución iterativa: método de Gradiente descendente

η = ratio de aprendizaje = varía $0.1 \rightarrow 0.2$
 a, b iniciales, se toman aleatorios

$$a = a - \eta \frac{\partial J}{\partial a}$$

$$b = b - \eta \frac{\partial J}{\partial b}$$



Comentarios al programa: el Método Lineal Exacto.

% comentario

clear; %

clc; % limpia la pantalla de comandos.

close all; % cierra ventanas de figuras.

$x = -10 : 0.1 : 10;$ % Vector horizontal:

$$[-10, -9.9, -9.8, \dots, 9.8, 9.9, 10]$$

$y = y';$ % transpuesta

$y_b = \underbrace{a * x + b + 0.5 * \text{random}(mx, 1)}_x;$ % u
random uniforme $\rightarrow [-1, 1]$

hold on; % Mantiene la ventana activa en espera.

disp('texto'); % escribe texto

pause; % pausa el programa

$c_s = \sum(y_b * x);$ % $\sum_{k=1}^n y_b_k * x_k = \text{producto escalar}$

$[a \quad a_n]$ operador

$b \quad c];$ % sin; despliega los valores en matriz

ctrl+c % aborta el programa.

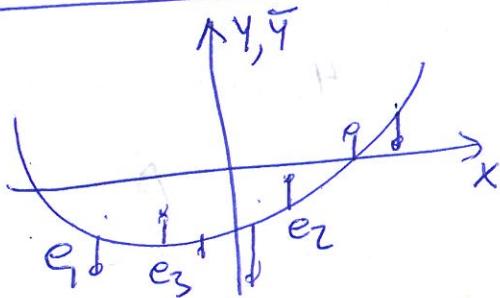
Regresión Cuadrática

Datos

$x | \bar{y}$

$x_1 | y_1$

$x_n | \bar{y}_n$



$$\bar{J} = \frac{1}{2} (e_1^2 + e_2^2 + \dots)$$

$$y = ax^2 + bx + c$$

$$\frac{\partial \bar{J}}{\partial a} = \frac{\partial \bar{J}}{\partial b} = \frac{\partial \bar{J}}{\partial c} = 0$$

para minimizar error de \bar{J}

Solución exacta matricial.

$$\begin{bmatrix} \frac{\partial J}{\partial a} \\ \frac{\partial J}{\partial b} \\ \frac{\partial J}{\partial c} \end{bmatrix} = \begin{bmatrix} \sum x_k^4 & \sum x_k^3 & \sum x_k^2 \\ \sum x_k^3 & \sum x_k^2 & \sum x_k \\ \sum x_k^2 & \sum x_k & n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} - \begin{bmatrix} \sum x_k^2 \bar{y}_k \\ \sum x_k \bar{y}_k \\ \sum \bar{y}_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

A Z B

$$A Z - B = 0$$

$$Z = A^{-1} B.$$

Tarea 1: Programar la regresión cuadrática
condiciones iniciales:

$$-3 \leq x \leq 3, \text{ paso } \gamma = 1 = 0.1 \quad a = L$$

$$b = Z$$

$$c = -5$$

solución iterativa de Regresión cuadrática.

$$\begin{aligned} a &= a - \eta \frac{\partial J}{\partial a} \\ b &= b - \eta \frac{\partial J}{\partial b} \\ c &= c - \eta \frac{\partial J}{\partial c} \end{aligned}$$

condición de fín

$$J(K, 1) = 0.5 * \text{sum}(e_r * e_r)$$

- 1) Se toma valores iniciales de a, b, c
- 2) se calcula $\frac{\partial J}{\partial a}, \frac{\partial J}{\partial b}, \frac{\partial J}{\partial c}$:

calcular

$$d_j d a = [\text{sum}(x^4) \cdot a + \text{sum}(x^3) \cdot b +$$

$$\text{sum}(x^2) \cdot c - \text{sum}(x_k^2 \cdot \bar{y}_k)] / n$$

$$d_j d b = [\text{sum}(x^3) \cdot a + \text{sum}(x^2) \cdot b +$$

$$\text{sum}(x) \cdot c - \text{sum}(x_k \cdot \bar{y}_k)] / n$$

$$\Delta \hat{d}c = [\text{seum}(x^2) a + \text{seum}(x) b \\ + nc - \text{seum}(\bar{x}_k)] / n$$

3) Se calcula $a = a_{\text{anterior}} - \gamma a_{\text{anterior}}$;
 $b =$
 $c =$

4) Se calcula: variaciones porcentuales.

$$\Delta a_{\text{por}} = \frac{\text{abs}(a - a_{\text{anterior}})}{a} * 100$$

$$\Delta b_{\text{por}}$$

$$\Delta c_{\text{por}}$$

5) Criterio de fin

epsilon para a, b y c

$$\text{if } [(\Delta a_{\text{por}} < \text{epsabc}) \& (\Delta b_{\text{por}} < \text{epsabc}) \& (\Delta c_{\text{por}} < \text{epsabc})] \text{ break;}$$

Hacer pruebas para.

$$\gamma = 0.5, 0.2, 0.1, 0.05, 0.01, 0.005, 0.001$$

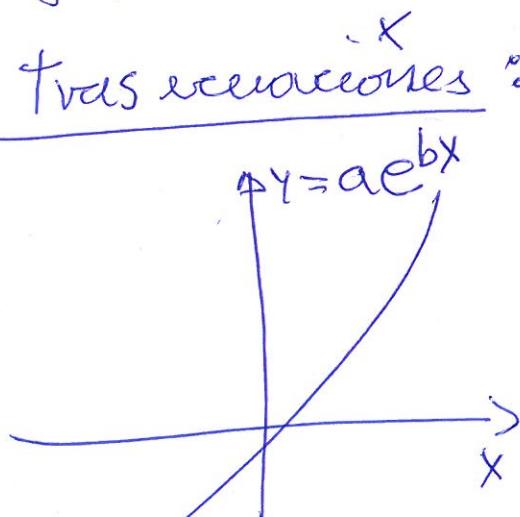
Valores iniciales para a, b y c cerca y lejos de la solución.

Varias iteraciones de convergencia con γ (no con a, b y c)?

Otras ecuaciones:

$$y = ae^{bx}$$

$$y = a \sin(bx)$$

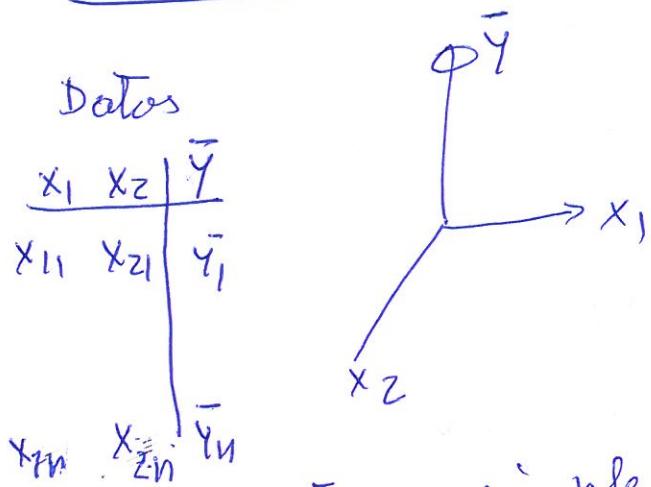


? a, b?

? a, b?

Entrenamiento de Redes Neuronales -

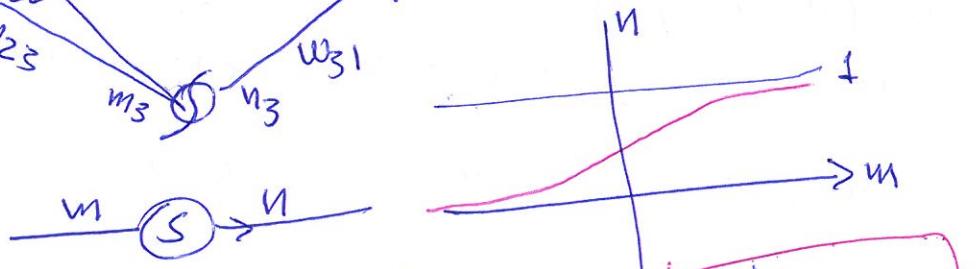
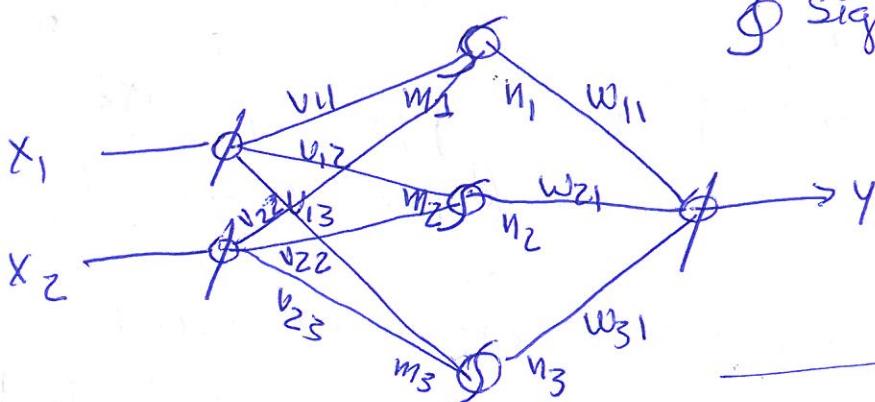
8



Forma simple:

ϕ lineal

ϕ sigmoidal



$$n = f(m) = \frac{1}{1 + e^{-m}} \Rightarrow e^{-m} = \frac{1-n}{n}$$

$$\begin{aligned} \frac{\partial n}{\partial m} &= (1+e^{-m})^{-2} e^{-m} (-1) \\ &= -\frac{e^{-m}}{(1+e^{-m})^2} = -\left(\frac{1-n}{n}\right) n^2 \\ &\boxed{\frac{\partial n}{\partial m} = n(n-1)} \end{aligned}$$

Soluciones:

APRENDIZAJE PATRON

$$J = \frac{1}{2} (y_k - \bar{y}_k)^2$$

Deriva $\frac{\partial J}{\partial w_{ij}}$ epoca $k=1 \rightarrow$ actualiza w_{ij}, w_{ik}
Deriva para $k=2 \rightarrow$ actualiza w_{ij}, w_{ik}
Deriva para $k=n \rightarrow$ actualiza w_{ij}, w_{ik}

APRENDIZAJE BATCH

$$J = \frac{1}{2} \sum (y_k - \bar{y}_k)^2$$

Deriva para $k=1 \rightarrow$ Guardar
 $k=2 \rightarrow$ J
 $k=n \rightarrow$ J

Promedio de derivadas
actualiza v_{ij}, w_{ij}

$$J = \frac{1}{2} \sum (y - \bar{y})^2$$

Hallar v_{ij} w_{ij} que minimice

Sólo para el método ITERATIVO \Rightarrow NO PARA EL EXACTO

$$m_i = \sum_{k=1}^{z=n} v_{ki} x_k$$

$$y = \sum_{k=0}^{3=n} w_{kj} v_k$$

$$\begin{aligned} m_1 &= v_{11}x_1 + v_{21}x_2 \rightarrow u_1 = f(m_1) \\ m_2 &= v_{12}x_1 + v_{22}x_2 \rightarrow u_2 = f(m_2) \\ m_3 &= v_{13}x_1 + v_{23}x_2 \rightarrow u_3 = f(m_3) \end{aligned}$$

$$Y = w_{11}u_1 + w_{21}u_2 + w_{31}u_3$$

$$\frac{\partial J}{\partial w_{11}} = (y - \bar{y}) \frac{\partial Y}{\partial w_{11}} = (y - \bar{y}) u_1$$

$$\frac{\partial J}{\partial w_{21}} = (y - \bar{y}) u_2$$

$$\frac{\partial J}{\partial w_{31}} = (y - \bar{y}) u_3$$

$$\frac{\partial J}{\partial v_{11}} = (y - \bar{y}) \frac{\partial Y}{\partial v_{11}} = (y - \bar{y}) \frac{\partial}{\partial v_{11}} (w_{11}u_1 + w_{21}u_2 + w_{31}u_3)$$

$$= (y - \bar{y}) w_{11} \frac{\partial u_1}{\partial v_{11}} = (y - \bar{y}) w_{11} \left(\frac{\partial u_1}{\partial m_1} \frac{\partial m_1}{\partial v_{11}} \right)$$

$$= (y - \bar{y}) w_{11} f'(m_1) \cdot X_1$$

$$\frac{\partial J}{\partial v_{12}} = (y - \bar{y}) w_{21} f'(m_2) X_1$$

$$\frac{\partial J}{\partial v_{13}} = (y - \bar{y}) w_{31} f'(m_3) X_1$$

$$\frac{\partial J}{\partial v_{21}} = (y - \bar{y}) w_{11} f'(m_1) X_2$$

$$\frac{\partial J}{\partial v_{22}} = (y - \bar{y}) w_{21} f'(m_2) X_2$$

$$\frac{\partial J}{\partial v_{23}} = (y - \bar{y}) w_{31} f'(m_3) X_2$$

$$\frac{\partial J}{\partial v_{ij}} = (y - \bar{y}) w_{ij} f'(m_j) X_i$$

∞ uno

Dados v_{ij} , w_{ij} iniciales se calcula $m_i \rightarrow n_i \rightarrow \hat{y} = \text{estimación de } y$

$$\hat{y} = w_{11}n_1 + w_{21}n_2 + w_{31}n_3$$

Para calcular los siguientes v_{ij} , w_{ij} se procede al revés

$$\delta_{w_{ij}} := \frac{\partial J}{\partial w_{ij}} = (\hat{y} - \bar{y}) n_i$$

$$w_{ij}^* = w_{ij} - \eta \frac{\partial J}{\partial w_{ij}}$$

$$\frac{\partial J}{\partial v_{ij}} = (\hat{y} - \bar{y}) w_{ji} f(m_j) x_i$$

$$v_{ij} = v_{ij} - \eta \frac{\partial J}{\partial v_{ij}}$$

$x_1 x_2 \bar{y}$
$\bar{x}_1 \bar{x}_2 \bar{y}_1$ patrón
$\bar{x}_1 \bar{x}_2 \bar{y}_2$

2 Formas de Entrenamiento

PATRON

$i = \text{iteraciones}$

$K=1, N$

calcula $\frac{\partial J}{\partial w_{ij}}$, $\frac{\partial J}{\partial v_{ij}}$

actualiza w_{ij} , v_{ij}
va al siguiente patrón ó ala
sigue iteración

Aprende más para x_i local

BATCH

$i = \text{iteraciones}$

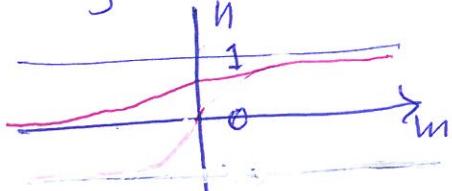
$i = \text{iteraciones}$

calcula $\frac{\partial J}{\partial w_{ij}}$, $\frac{\partial J}{\partial v_{ij}}$

para los K patrones
promedio de los $\frac{\partial J}{\partial w_{ij}}$, $\frac{\partial J}{\partial v_{ij}}$
calcula w_{ij} , v_{ij}

Resumen de funciones f de transformación de neurona

Sigmoidal t

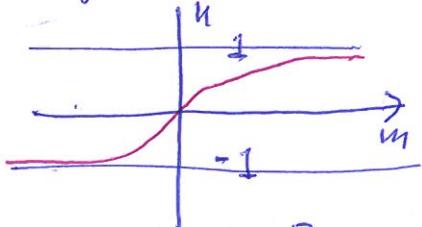


$$n = f(m) = \frac{1}{1 + e^{-m}}$$

$$f'(m) = n(n-1)$$

pag. 8

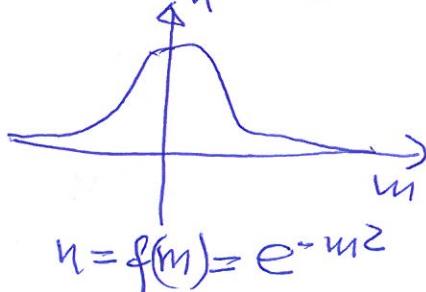
Sigmoidal z



$$n = f(m) = \frac{z}{1 + e^{-m}} - 1$$

$f'(m)$ en función de n

Gaussiana



$$n = f(m) = e^{-m^2}$$

$$f'(m) = -2mn$$

Tercera clase

Entrenamiento Batch. para red neuronal.

Datos	
x_1	x_2
\bar{y}	
$x_{11} \ x_{21}$	y_1

PATRON

$$S = \frac{1}{2}(y - \bar{y})^2 \quad J = \frac{\frac{1}{2} \sum_k (y_k - \bar{y})^2}{N}$$

BATCH

$$\boxed{\frac{\partial J}{\partial v_{23}}} = (y - \bar{y}) w_{31} f(m_3) x_2$$

$$v_{23} = v_{23} - \eta \boxed{\frac{\partial J}{\partial v_{23}}}$$

Patron

\bar{y}	
x_1	x_2
$x_{11} \ x_{21}$	\bar{y}_1

\rightarrow deriva y actualiza los $v_{23} =$

Batch.

Deriva =

$$\text{derivat } \frac{\partial J}{\partial V_{23}}$$

Deriva

Deriva

Procedimiento de derivadas] \rightarrow actualiza derivadas

Los dos métodos son 2 bucles exteriores
 Externo $\approx k = 1, \rightarrow 10.000$ iteraciones
 Interno :

Ambas métodos son 2 bucles.

Externo: $i = 1, \rightarrow 10.000$ iteraciones.

Interno: N componentes de la muestra

Patrón

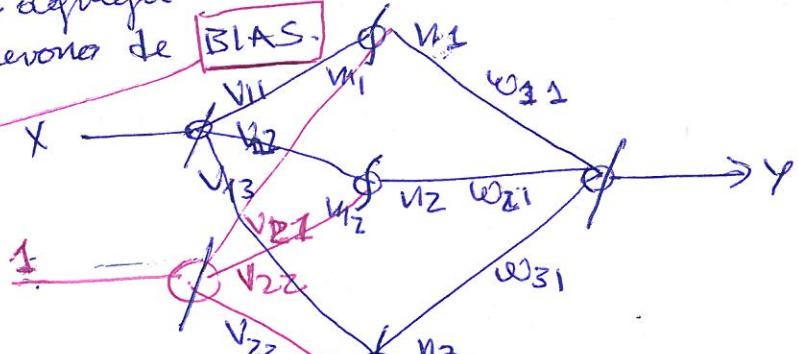
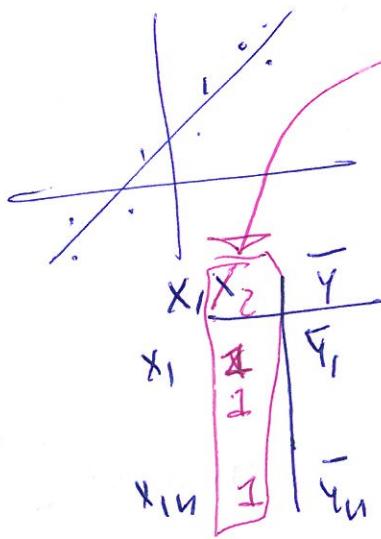
i
 $k = 1, N$
 derivada
 actualiza
 10 000

Batch.

$i =$
 $k = 1, N$
 derivada = derivada + $\frac{\partial J}{\partial V}$
 1000
 derivada = $\frac{\text{derivada}}{N}$
 actualiza

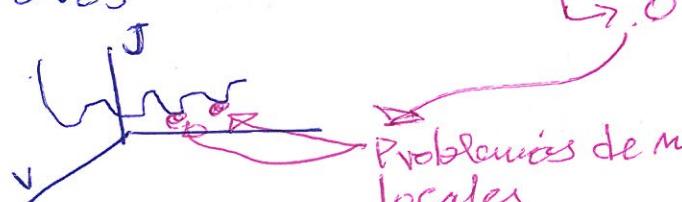
Programa.

Se agrega
neuronas de BIAS.

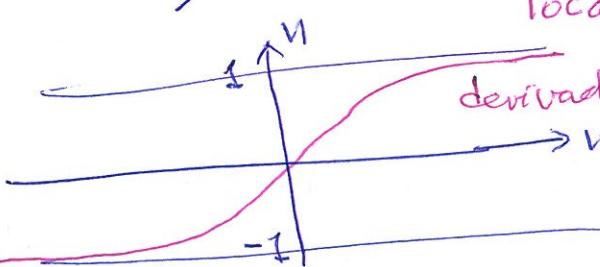


$$v_{23} = v_{23} - \eta \frac{\partial J}{\partial v_{23}}$$

$$\frac{\partial J}{\partial v_{23}} = (y - \bar{y}) w_{31} f'(u_3) x_2$$



Sigmoidal



derivada $\rightarrow 0$
 Para evitar que $f'(u) = 0 \Rightarrow u$ debe ser pequeña

$$m_1 = v_{11} x_1 + v_{21} x_2$$

↓ ↓
pequeños

si x es muy grande se escalan los valores.

Tarea: Entrenar una red neuronal

Patrón

Batch

- Analizar la variación del RATIO DE APRENDIZAJE
 η (eta) = 5, 2, 1, 0.5, 0.1
- Tipo de función de activación:
Sigmoidal, 2 o gáesiana.
- Valores iniciales de v_{ij} ~~y~~ w_{ij} en vectores: 10, 5, 1, 0.5
 - Con y sin BIAS
 - % del criterio de convergencia: 10%, 1%
0.75%

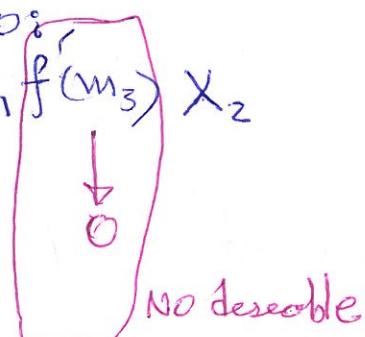
clase 04

Recomendaciones

se tiene las derivadas del tipo:

$$\frac{\partial J}{\partial v_{23}} = (\hat{y} - \bar{y}) w_{31} f'(m_3) x_2$$

↓ ↓
0 0
Deseable



Se prefiere superar:

- 1) Agregando neurona BIAS.
Adicionar $x = [1, 1, \dots, 1]$ al otro valor.
- 2) Escalamiento de valores de x

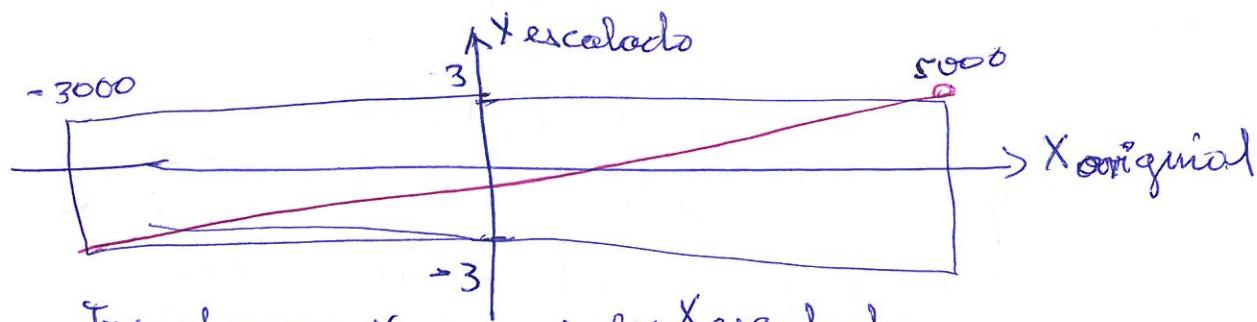
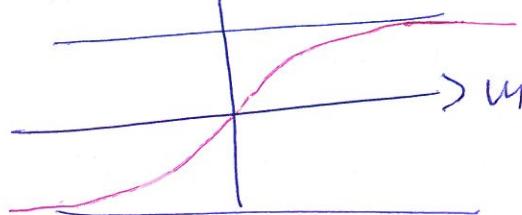
Datos fuera de escala

X_1	X_2	\bar{Y}
5000	0.1	1000
⋮	⋮	⋮
-4000	1.2	-1000
↓		
Rango: $\langle -3, 3 \rangle$		

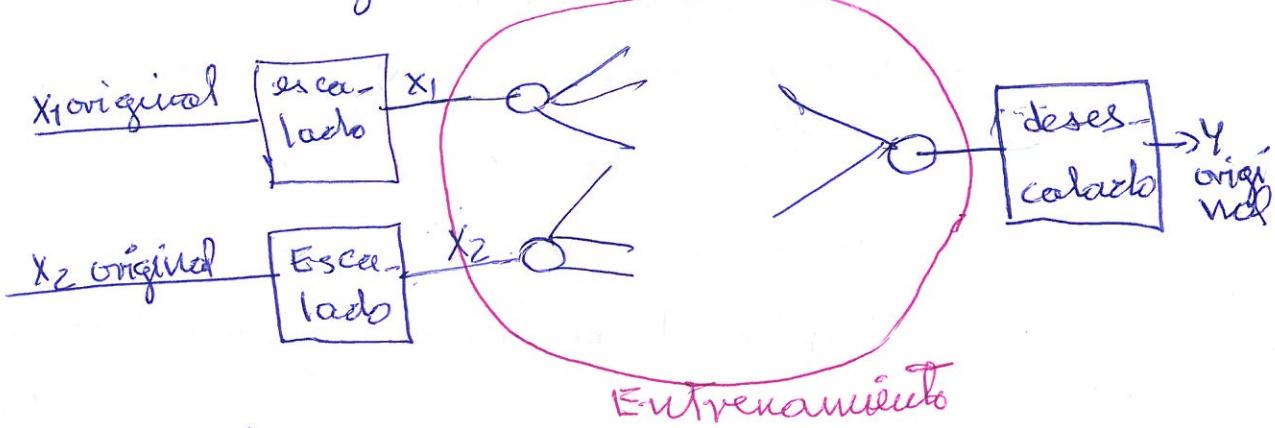
Si los x_i son pequeños; los w_i lo serán¹⁴

$$w_i = v_{1i}x_1 + v_{2i}x_2 + v_{3i}x_3$$

$f(w_i)$ no será ≈ 0

$$f^n = f(w)$$


Transformar X_{original} en X_{escalado} .



3) Valores iniciales pequeños para v_{ij} w_{ij}

Aprendizaje Patrón

$$J = \frac{1}{2} (\bar{Y} - \hat{Y})^2$$

$$\frac{\partial J}{\partial w_{23}} = (\bar{Y} - \hat{Y}) w_{31} f'(w_3) x_2$$

Sí hay MAS SALIDAS $\therefore Y_1, Y_2, \dots$

X_1	X_2	\bar{Y}_1	\bar{Y}_2	$J_{\text{patrón}} = \frac{1}{2} (\bar{Y} - \hat{Y})^T (\bar{Y} - \hat{Y})$
⋮	⋮	⋮	⋮	$= \frac{1}{2} \left[(\bar{Y}_{11} - \hat{Y}_{11})^2 + (\bar{Y}_{12} - \hat{Y}_{12})^2 \right]$
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	

Se hace muy complicado los cálculos de

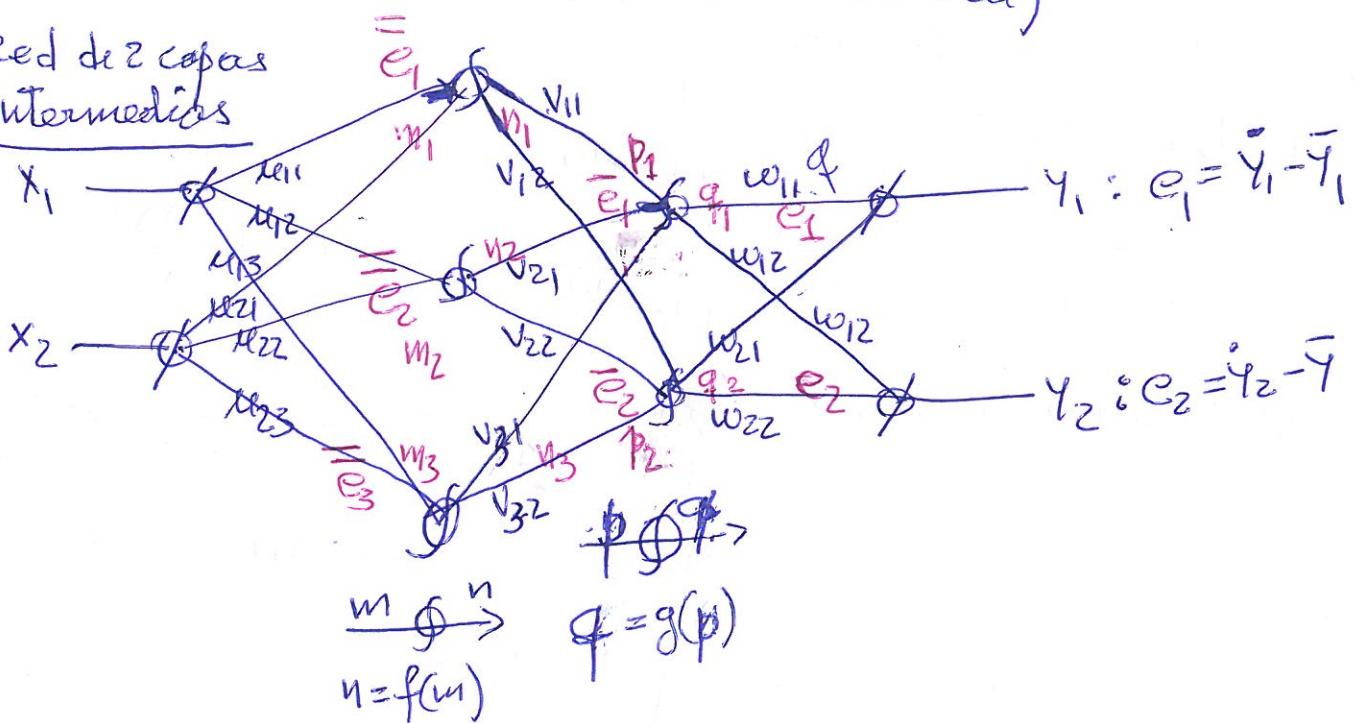
$$\frac{\partial J}{\partial w_{ij}}$$

$$\frac{\partial J}{\partial v_{ij}}$$

Algoritmo de Retropropagación de errores <https://www.cs.uis.es/cursos/fiaz/temos15>
 Solo parece calcular las derivadas $\frac{\partial J}{\partial v_{ij}}$, $\frac{\partial J}{\partial w_{ij}}$, ...

$$\text{error} = e = y - \bar{y} \quad (\text{a la salida})$$

Red de 2 capas
intermedias



$$\left[\begin{array}{l} \frac{\partial J}{\partial w_{11}} \\ \frac{\partial J}{\partial v_{11}} \end{array} \right] = (\text{error retropropagado}) \times (\text{salida de memoria de capa anterior})$$

$$\frac{\partial J}{\partial w_{21}} = e_1 q_2$$

$$\frac{\partial J}{\partial v_{21}} = \overline{e}_1 q_2$$

$$\frac{\partial J}{\partial w_{32}} = e_2 q_3$$

Comentarios del programa:

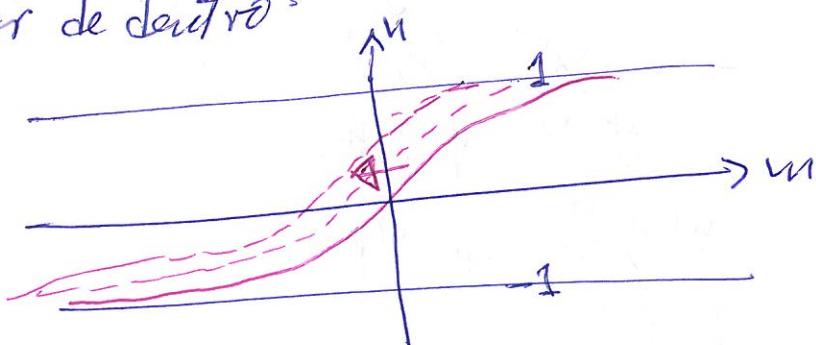
2 entradas, 1 capa intermedia con 50 neuronas
2 salidas

Cambios en las funciones

1) de pendiente $u = \frac{1}{1+e^{-m}}$

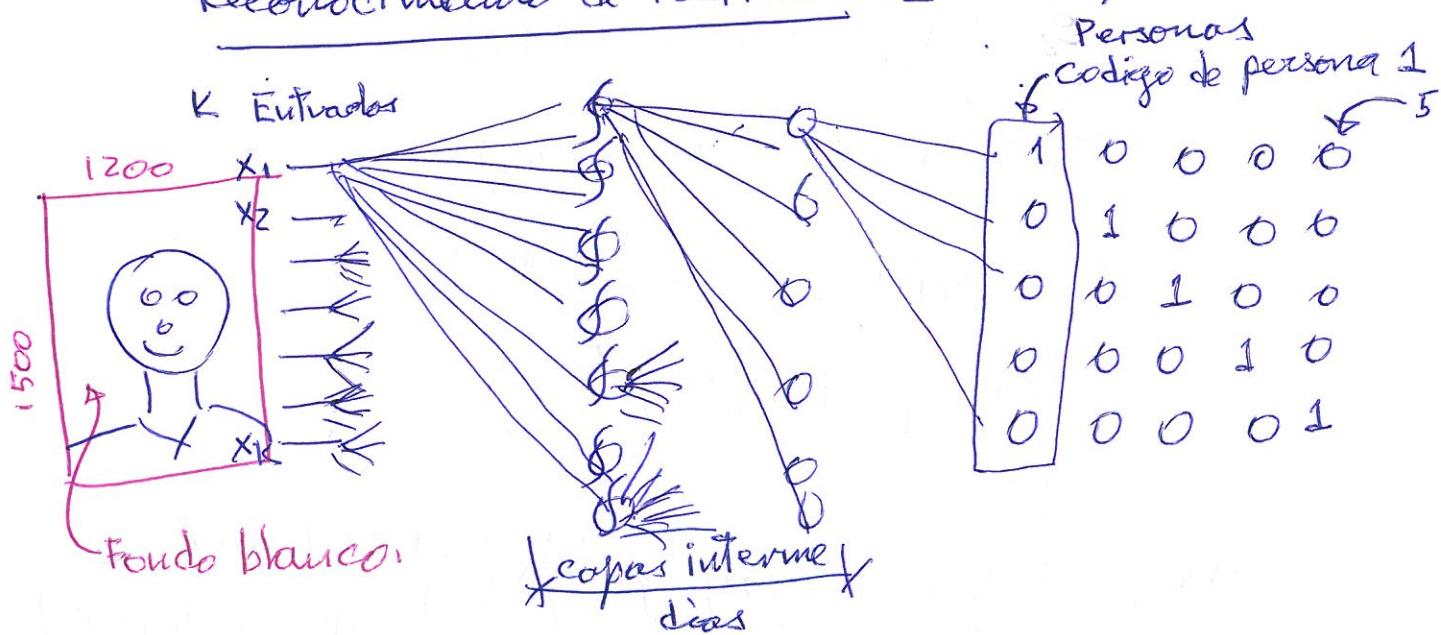
Ejemplo. $u = \frac{1}{1+e^{-\frac{w}{d}}}$ $a = a - \eta \frac{\partial J}{\partial a}$

2) Cambiar de dentro:



FIN DE LA TEORIA

Reconocimiento de Rostros. (5 rostros)

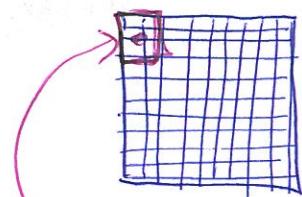


Tratamientos de fotos

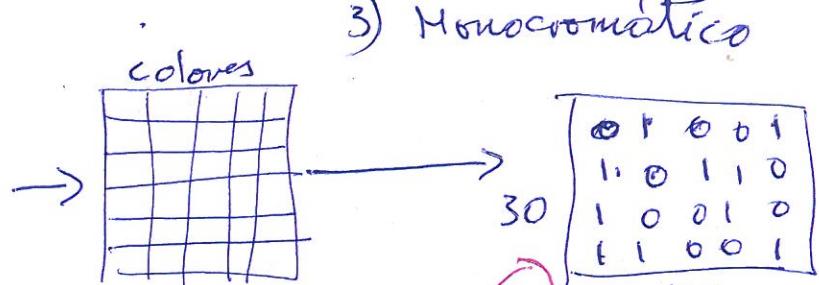
1) Recorte



3) Pixelamiento



promedio.



3) Monocromático

0	1	0	0	1
1	0	1	1	0
1	0	0	1	0
1	1	0	0	1

30

40

1200 puntos

Valor de pixel: [0, 250]

blanco → 1

negro → 0

[0, 125] → 0

[126, 250] → 1

4) Convertir matriz de 30×40 en vector de 1200 (5 caras)

X	y_1	y_2	y_3	y_4	y_5
$x_{1,1}$ → $x_{1,1200}$	1	0	0	0	0
$x_{2,1}$ → $x_{2,1200}$	0	1	0	0	0
$x_{3,1}$	0	0	1	0	0
$x_{4,1}$	0	0	0	1	0
$x_{5,1}$ → $x_{5,1200}$	0	0	0	0	1

Proceso:

1) Se entrena la red.

2) Se valida $\| \cdot \|_1$: cambios pequeños en las personas \Rightarrow reconoce la misma persona.

Todas las imágenes de la red neuronal tienen el mismo tamaño.

Clase 05Reconocimiento de carasImagen a colores: [R][G][B] ~~Monocromático~~.~~Matriz~~~~escala de grises~~~~pixeles~~(grupos de 10×10)

promedio

De gris a blanco

Monocromático

 40×30

Para 5 caras: 40×30

$$\frac{100}{11} \approx 60 \times 50?$$

↳ Se equivoca más. Aumentar la precisión

Manejo de patrones.

	x_1 — x_{1200}	\bar{y}_1	\bar{y}_5
1	[sam, patrones]		
2			
5	[tau5 patrones]		

Programas que entrega el profesor:

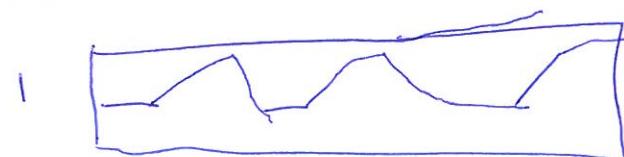
- 1) NeuroCarasReconoce.m = Para entrenar
 - 2) NeuroCarasReconoce.m = Para validar
- salida: pesosCaras.mat

Trabajo 4 :- Trabajar con 12 fotos

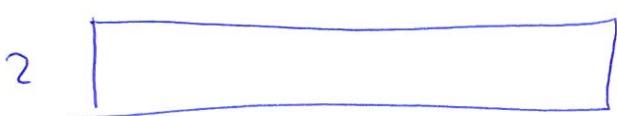
- Matriz de 50×50 ?

- Validar x cambiar pixeles hasta que no se reconozca
- x Imágenes giradas, rotadas

Reconocimiento de Anomalías Cardíacas



1: Normal



2: Anomalía 1



4 11 4

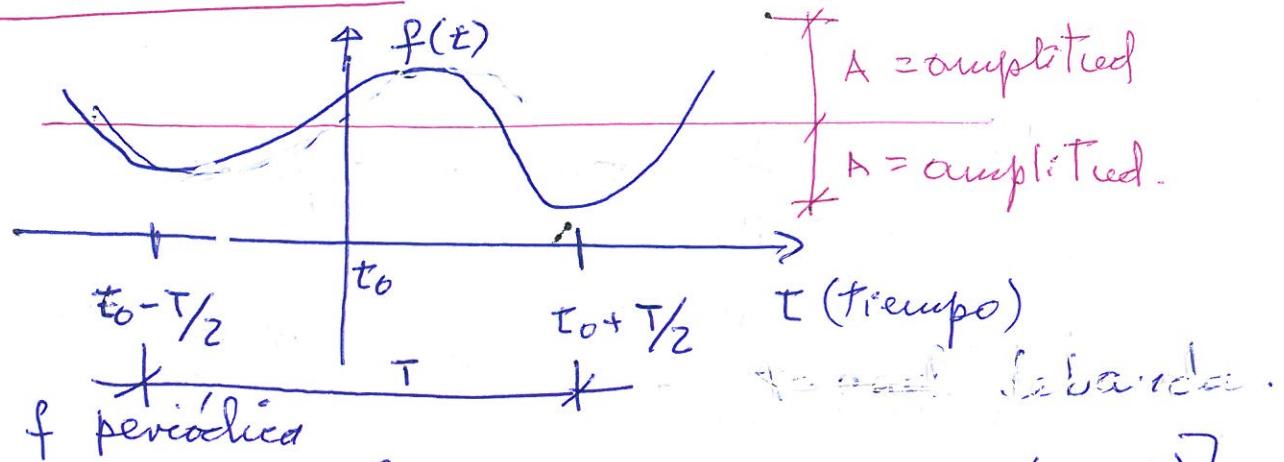
\bar{y}_1

\bar{y}_5

\bar{y}_1

\bar{y}_5

Serie de Fourier -



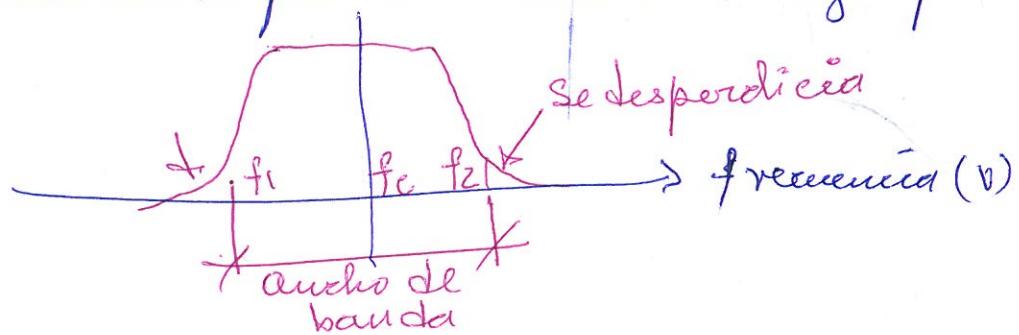
$$f(t) \approx \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(\omega_n t) + b_n \sin(\omega_n t)]$$

$$\omega_n = \left(\frac{2\pi}{T}\right)n \quad a_0 = \frac{2}{T} \int_{t_0 - T/2}^{t_0 + T/2} f(t) dt.$$

$$a_n = \frac{2}{T} \int_{t_0 - T/2}^{t_0 + T/2} f(t) \cos(\omega_n t) dt; \quad b_n = \frac{2}{T} \int_{t_0 - T/2}^{t_0 + T/2} f(t) \sin(\omega_n t) dt$$

$$c_n = \sqrt{a_n^2 + b_n^2}$$

- T = periodo = tiempo de una oscilación completa
- A = Amplitud = Separación máxima de la posición de equilibrio.
- Frecuencia = v = # oscilaciones en 1 seg. (Hz)
- Frecuencia angular o pechazos (ω) = $2\pi v$
- Ancho de banda (señales analógicas) H_3 = extensión de la frecuencia en la que se concentra la mayor parte de una señal



Teorema de muestreo

Frecuencia de muestreo ≥ 2 (ancho de banda de señal f_{muestreo})

Ancho de banda ~~de los pulsos~~ de los pulsos del corazón = 150 Hz.

[Frecuencia de latido del corazón = 1/seg = 1 Hz.]
lati 60 veces/muestra

$$f_{\text{muestreo}} \geq 2(150) = 300 \text{ Hz}.$$

Materiales Recursos

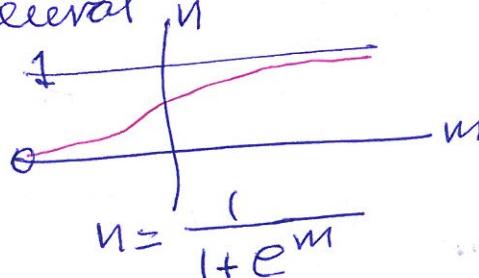
Archivo:	Descripción
1 mneuroECG.m	Programa de entrenamiento (señal limpia)
2 ValidaECG.m	" para validación (" ruidosa)
3 VillaECG.mot	saldade 1 ?
4 ecgred.mot	" 2 ?

Cantidad mínima de neuronas intermedias $\geq \# \text{ de salidas } / 4$

Trabajo: Entrenar red neuronal

$$v = v - \eta \frac{\partial J}{\partial v}$$

$$w = w - \eta \frac{\partial J}{\partial w}$$



Para aumentar la velocidad de cálculo se procede:

1) cambiar la pendiente

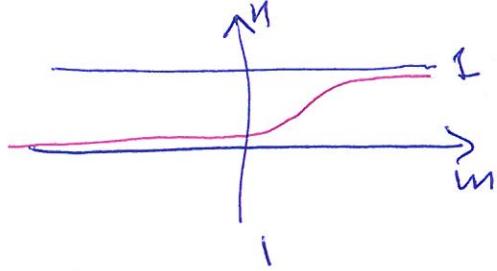


$$n = \frac{a}{1 + e^{-m/a}}$$

$$a_{\text{inicial}} = 1$$

$$a = a - \eta_a \frac{\partial J}{\partial a}$$

2) Cambiar el centro



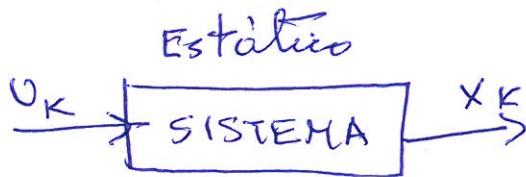
$$u = \frac{1}{1 + e^{-(m-c)/a}}$$

$$c = c - \gamma \frac{\partial J}{\partial c}$$

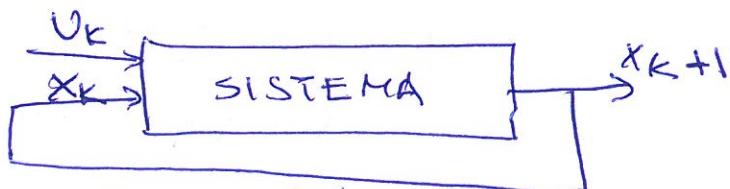
$$c_{\text{initial}} = 0$$

Clase 06 - Redes Neuronales Dinámicas.

Aprendizaje \rightarrow tiempo

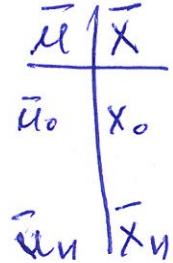
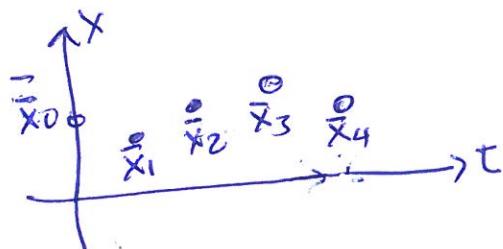
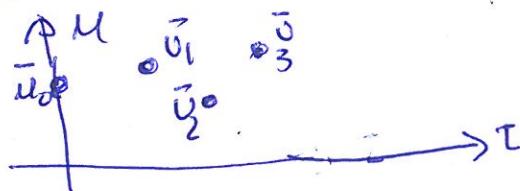


$$x_k = \phi(u_k)$$



$$x_{k+1} = \phi(x_k, u_k)$$

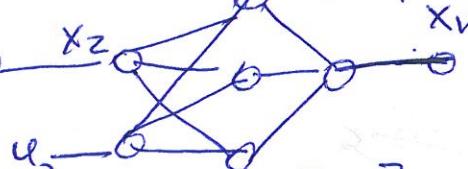
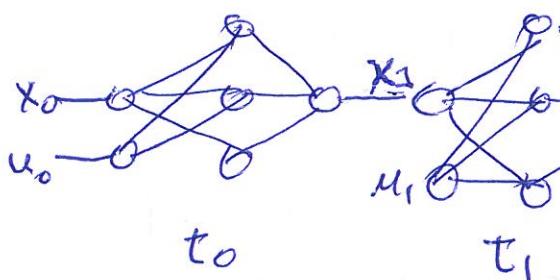
Aplicaciones = Robótica.



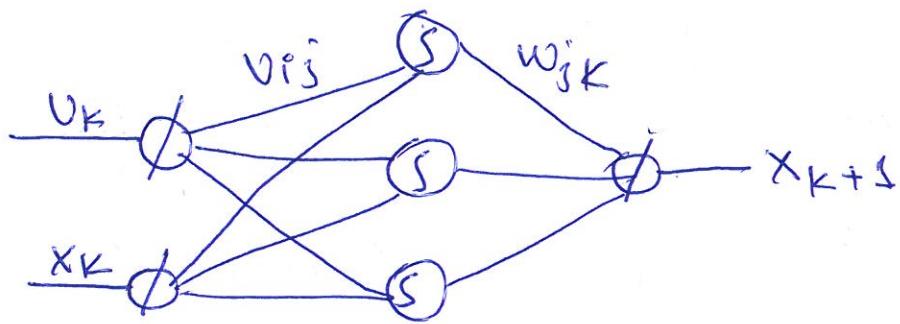
Entrenamiento : Solo se necesita aplicar BATCH
(Training) algoritmo : gradiente descendente

La dinámica se descarta: SE USAN LOS MISMOS

v_i , w_{jk} PARA TODO EL CICLO



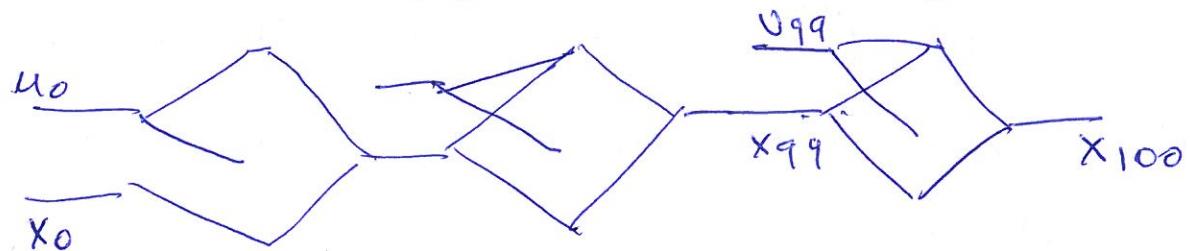
\bar{v}_i , x_i } son conocidos.



Algoritmos utilizados:

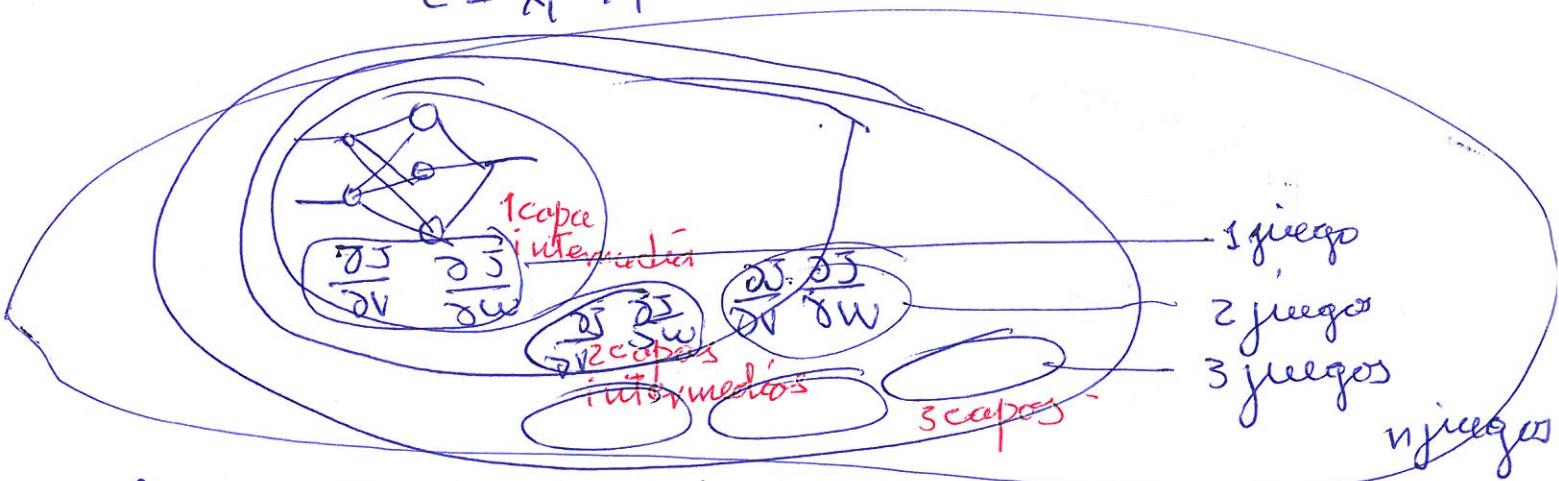
- 1) Back Propagation through Time (BPTT) Paul Werbos 1972
- Fácil de entender; pero requiere mucha memoria para almacenar v_{ij} , w_{jk} , $\frac{\partial J}{\partial v_{ij}}$, $\frac{\partial J}{\partial w_{jk}}$, etc.

$$v = v - \eta \frac{\partial J}{\partial v} \quad w = w - \eta \frac{\partial J}{\partial w}$$



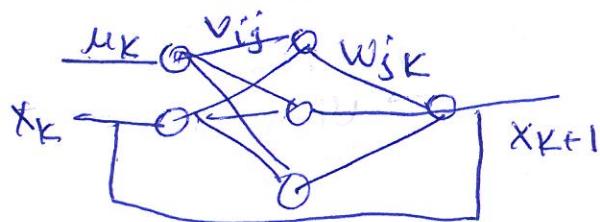
Se tiene \bar{x}_1 , se calcula x_1

$$e = x_i - \bar{x}_i$$



finalmente

$\frac{\partial J}{\partial v}$, $\frac{\partial J}{\partial w}$ son el promedio de todas las $\frac{\partial J}{\partial v}$, $\frac{\partial J}{\partial w}$



Fácil de entender, pero toma mucha memoria

2) Algoritmo Dinamic Back Propagation (DBP)

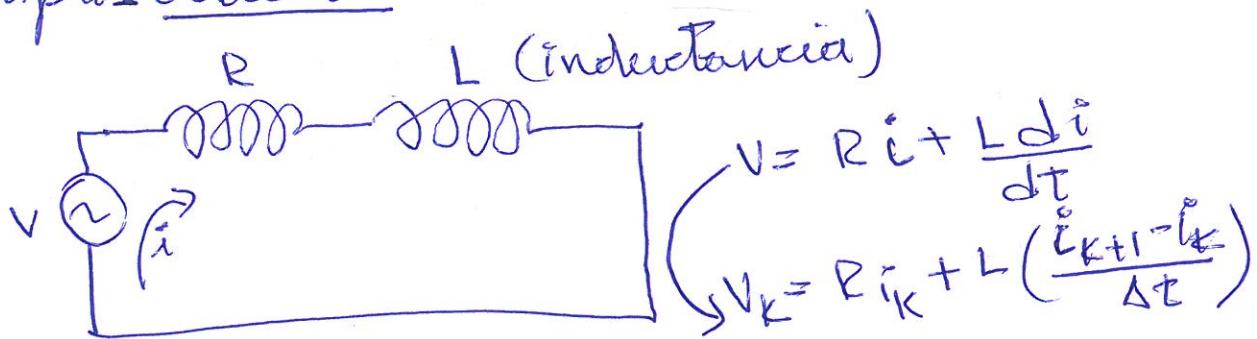
23

(Memorias asociativas)

Kempinski Novedosa
1989

Usa fórmula recursiva que no requiere guardar información anterior.

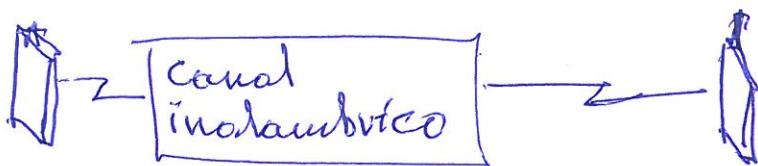
Ejemplo 1: circuitos



Puede cambiar la función de R y no cambia el algoritmo, ejemplo $V = Y_i^3 + \frac{L d\dot{i}}{dt} = f(i_k) V_k$

$$\dot{i}_{k+1} = f(i_k) V_k$$

Ejemplo 2: celulares



Buscador en internet: algoritmos BPPT y DBP.

Cálculo de los V_{ij} W_{ij}

$$V_{ij} = V_{ij} - \eta \frac{\partial J}{\partial V_{ij}}$$

$$W_{jk} = W_{jk} - \eta \frac{\partial J}{\partial W_{jk}}$$

Derivadas parciales Totales
Total Partial derivatives

Ejemplo.

$$z = 3y + 2x$$

$$y = 4x + 5z$$

$$x = 2z + 6y$$

Derivadas parciales sencillas

$$\frac{\partial z}{\partial x} = 2$$

Derivadas parciales totales

$$\begin{aligned}\frac{\partial z}{\partial x} &= \frac{\partial z}{\partial x} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \frac{\partial x}{\partial x} \\ &= 2 + 3 \cdot 4 + 3 \cdot 5 \cdot 2 \\ &= 2 + 12 + 30 = 44\end{aligned}$$