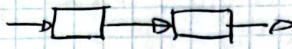


- **Soft computing:** Rama de la IA que engloba diversas técnicas para solucionar problemas que manejan info incompleta

- { - Neural Networks (NNs)
- Fuzzy logic (FL)
- Genetic Algorithms (GA)

- **Sistemas de control abierto**

- Funcionan sin feedback



- **Sistemas de control cerrado**

- El sistema observa el feedback y distintos sensores para ver como actuar



- **Estabilidad del sistema**

- Sabemos que un sistema es estable si de la planta es capaz de volver a su posición inicial después de una perturbación

- El problema de los sistemas es como encontrar una función que podamos implementar para que haga lo que nosotros queremos
- posibles soluciones!

↳ **Análisis:** consiste en recolectar info

↳ **Síntesis:** Construyendo un exitoso sistema de control

- Requiere mucho conocimiento y es solo una aproximación

- **Aproximación clásica:** Define una ley de control plausible a partir de un modelo matemático

- **Métodos actuales:** Es el entrenamiento del modelo sin un modelo matemático.

Esta es la base para la approx neuronal y borrosa

## Tema -2-

- **Sistema o proceso:** Toda realidad en la que interactúan variables de diferentes tipos para producir señales observables



- **Modelo de un sistema:** Descripción matemática del mismo que permite predecir su comportamiento

### Tipos de sistemas:

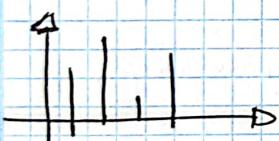
- **Líneales** Aplico  $x_1(t)$  y la salida es  $y_1(t)$   
"  $x_2(t)$  " " "  $y_2(t)$

- **Invariantes** La salida es la misma independiente del momento en que se aplique la entrada

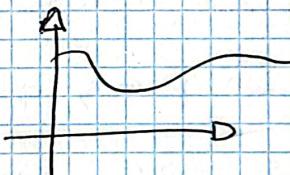
- **Causales** La salida nunca precede a la entrada

- **Estables** Ante entradas acotadas, produce salidas acotadas

### Discretos



### Continuos



#### Estaticos

{ La salida solo depende del valor de la entrada en el tiempo actual

#### Diminicos

" " " del momento en el que se aplicó

En el mundo Real son siempre causales

LTI (Líneales e invariantes en el tiempo)

LD Líneales, invariantes y causales

- Sistemas en tiempo continuo

- Su estudio permite obtener Ecuaciones diferenciales

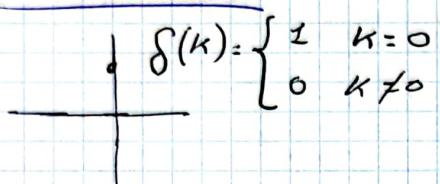
- Podemos usar la transformada de Laplace

- Sistemas en tiempo discreto

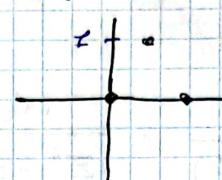
- Se describen mediante Ecuaciones EN diferencias

- Podemos indicando valores o usar la transformada Z

### Función delta

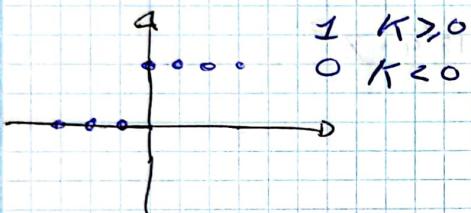


$$\delta(k-1)$$



~~δ(k-3)~~ solo va a ser ≠ en k=3

### Escalón unitario



• Formas de representar los datos

① Figura

②  $x(k) = \{4, 3, 5, 2'5, 2, 3'5, \dots\}$

③ Suma ponderada de secuencias impulso desplazadas

$$x(k) = 4\delta(k) + 3\delta(k-1) + 5\delta(k-2) + 2'5\delta(k-3) + \dots$$

④  $x(0) = 4$

$x(1) = 3$

$x(2) = 5$

### Transformada Z

$x(k)$	$X(z)$
Impulso unitario $\delta(k)$	1

$$X(z)$$

$x(k)$	$X(z)$
Escalón unitario $u(k)$	$\frac{z}{z-1}$

$$\frac{z}{z-1}$$

$x(k)$	$X(z)$
$a^k$	$\frac{z}{(z-a)^2}$

$$\frac{z}{(z-a)^2}$$

$$\frac{z}{z-a}$$

### Teorema del valor inicial

$$x(0) = \lim_{z \rightarrow \infty} X(z)$$

### Teorema del valor final

$$\lim_{k \rightarrow \infty} x(k) = X_{ss} = \lim_{z \rightarrow 1^-} (1 - z^{-1}) \cdot X(z)$$

### 2 Métodos

#### División directa

$$X(z) = \frac{10z + 5}{(z-1)(z-0.5)} = \frac{10z + 5}{z^2 - 1.5z + 0.5}$$

$$\begin{array}{r} 10z + 5 \\ \underline{-10z + 12 - 2 \cdot z^{-1}} \\ \hline 17 - 2 \cdot z^{-1} \end{array} \quad \begin{array}{r} |z^2 - 1.5z + 0.5| \\ 10 \cdot z^{-2} + 17 \cdot z^{-2} + 18'4 \cdot z^{-3} \\ \hline -17 + 20'4 \cdot z^{-1} - 3'4 \cdot z^{-2} \\ \hline |18'4 \cdot z^{-1} - 3'4 \cdot z^{-2}| \\ -18'4 \cdot z^{-2} + 22'08 \cdot z^{-3} - 3'68 \cdot z^{-4} \\ \hline |18'68 \cdot z^{-2} - 3'68 \cdot z^{-3} \dots| \end{array}$$

$$10 \cdot z^{-1} + 17 \cdot z^{-2} + 18'4 \cdot z^{-3} \dots$$

$$x(k) = 10 \cdot \delta(k-1) + 17 \delta(k-2) + 18'4 \delta(k-3)$$

a)  $\frac{P(z)}{(z-a) Q(z)} = \frac{A}{(z-a)} + \frac{R(z)}{Q(z)} \Rightarrow A = \frac{P(a)}{Q(a)}$

b)  $\frac{P(z)}{(z-a)^2 Q(z)} = \frac{A}{(z-a)^2} + \frac{B}{(z-a)} + \frac{R(z)}{Q(z)} \Rightarrow A = \frac{P(a)}{Q(a)}$   
 $B = \frac{P'(a) - A \cdot Q'(a)}{Q(a)}$

c)  $\frac{P(z)}{(z-a)^3 \cdot Q(z)} = \frac{A}{(z-a)^3} + \frac{B}{(z-a)^2} + \frac{C}{(z-a)} + \frac{R(z)}{Q(z)}$

$$A = \frac{P(a)}{Q(a)} \quad B = \frac{P'(a) - A \cdot Q'(a)}{Q(a)}$$

$$C = \frac{P''(a) - A \cdot Q''(a) - 2B \cdot Q'(a)}{2! \cdot Q(a)}$$

## Fracciones Parciales

$$X(z) = \frac{10z + 5}{(z - 1)(z - 0'2)} = \frac{A_1}{(z - 1)} + \frac{A_2}{(z - 0'2)} =$$

$$\alpha_1 = 1 \rightarrow A_1 = \frac{P(\alpha_1)}{Q(\alpha_1)} = \frac{10 \cdot 1 + 5}{(z - 0'2)} = \frac{15}{0'8} = 18'75$$

$$\alpha_2 = 0'2 \rightarrow A_2 = \frac{P(\alpha_2)}{Q(\alpha_2)} = \frac{7}{-0'8} = -8'75$$

$$= \frac{18'75}{(z - 1)} - \frac{8'75}{(z - 0'2)} \xrightarrow[z^k]{} (0'2)^k$$

$$\underline{X_1(z)} = z \cdot X(z) = 18'75 \cdot \frac{z}{(z - 1)} - 8'75 \cdot \frac{z}{(z - 0'2)}$$

$\downarrow T z^{-1}$

$$X_1(k) = (18'75 \cdot (1)^{k-1} - 8'75 (0'2)^{k-1}) \cdot 0(k)$$

$$\underline{X(z) = z^{-1} X_1(z)}$$

$\downarrow T z^{-1}$

$$X(k) = X_1(k-1) = (18'75 \cdot (1)^{k-1} - 8'75 (0'2)^{k-1}) \cup (k-1)$$

$$ejm \quad Y(k) = 3x(k) + 2x(k-1) - Y(k-2)$$

-  $x(-k) = y(-k) = 0$  (composición de sistemas multos)

- Aplicar transformada Z

$$x(k) = u(k-1)$$

$$X(z) = Z\{x(k)\}$$

$$Y(z) = Z\{y(k)\}$$

$$Z\{x(k-1)\} = z^{-1}x(k)$$

$$Z\{y(k-2)\} = z^{-2}Y(k)$$

↓ Tz

$$Y(z) = 3x(z) + 2z^{-1}x(z) - z^{-2}Y(z)$$

$$Y(z) = 3 \cdot \frac{1}{(z-1)} + 2 \cdot \frac{z^{-1}}{(z-1)} \quad \text{z } z^{-2}Y(z)$$

$$Y(z)(1 + z^{-1}) = \frac{3 + 2z^{-1}}{(z-1)}$$

$$Y(z) = \frac{3 + 2z^{-1}}{(z-1)(1+z^{-1})} \Rightarrow \text{todo } \cdot z$$

$$Y(z) = \frac{3z + 2}{(z-1)(z+1)} \quad \frac{(z-1)(1+z^{-1}) \cdot z}{(z-1) \cdot (z(1+z^{-1}))}$$

$$Y(z) = \frac{A_1}{(z-1)} + \frac{A_2}{(z+1)}$$

$$\alpha_1 = 1 \rightarrow A_1 = \frac{P(\alpha_1)}{Q(\alpha_1)} = \frac{5}{2}$$

$$\alpha_2 = -1 \rightarrow A_2 = \frac{P(\alpha_2)}{Q(\alpha_2)} = \frac{-1}{-2} = \frac{1}{2}$$

$$Y(z) = \frac{5/2}{(z-1)} + \frac{1/2}{(z+1)}$$

$$\frac{5}{2} \cdot \frac{z}{(z-1)} + \frac{1}{2} \frac{1}{(z+1)}$$

• Me faltan una  $z \Rightarrow$  cambio var

$$Y_1(z) = z \cdot Y(z)$$

$$\downarrow +z^{-1}$$

$$Y_1(k) = \left( \frac{5}{2} \cdot (z)^k + \frac{1}{2} (-z)^k \right) \cdot u(k)$$

$$Y(z) = z^{-1} \cdot Y_1(z)$$

$$\downarrow +z^{-1}$$

$$Y(k) = Y_1(k-z) = \left( \frac{5}{2} \cdot (1)^{k-1} + \frac{1}{2} (-1)^{k-1} \right) u(k-z)$$


---

$H(z)$  función de transferencia

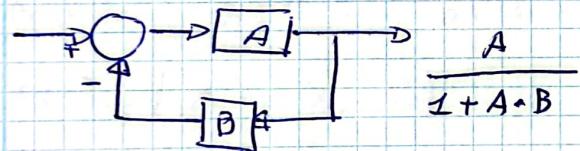
$$\rightarrow [A] \rightarrow [B] \rightarrow A \cdot B$$



$$A + B$$

$$\begin{array}{c} + \\ - \end{array} \xrightarrow{\hspace{1cm}} \text{estable}$$
  

$$\begin{array}{c} + \\ + \end{array} \xrightarrow{\hspace{1cm}} \text{inestable}$$



$$\frac{A}{1+A \cdot B}$$

~~OER 3~~

a)  $f(k) = 2 + 5k + k^2 + \delta(k) =$

$$= 2 \cdot \frac{z}{1-z^{-1}} + 5 \cdot \frac{z^{-1}}{(1-z^{-1})^2} + \frac{z^{-1}(1+z^{-1})}{(1-z^{-1})^3} + 1$$

e)  $k^2 \cdot e^{-3k} =$

$$= \frac{e^{-3} z^{-1} (1 + e^{-3} z^{-1})}{(1 - e^{-3} z^{-1})^3}$$

2)  
a) Teorema del valor final

$$x(z) = \frac{6}{(z-0.5)}$$

$$x_{ss} = \lim_{z \rightarrow \infty} (1 - z^{-1}) \cdot x(z) = \lim_{z \rightarrow \infty} \frac{z-1}{z} \cdot \frac{6}{(z-0.5)} = 0$$

b)  $x(z) = \frac{6}{(z-0.5)(z-1)}$

$$x_{ss} = \lim_{z \rightarrow \infty} \frac{z-1}{z} \cdot \frac{6}{(z-0.5)(z-1)} = \frac{6}{0.5} = 12$$

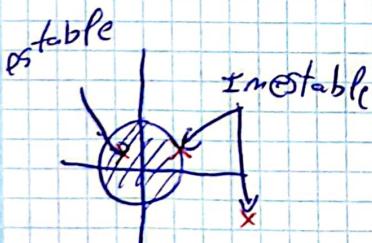
c)  $x(z) = \frac{c}{(z-0.5)(z-1)^2}$

$$x_{ss} = \lim_{z \rightarrow \infty} \frac{z-1}{z} \cdot \underbrace{\frac{c}{(z-0.5)(z-1)^2}}_0$$

↓  
polo en  $z=1$

En el cincelado

sistema inestable  $\Rightarrow x_{ss} \rightarrow \infty$



3) B/ obtener  $x(k)$

$$X(z) = \frac{z+2}{(z-2)z^2} = \underbrace{\frac{A}{z^2}}_{a=0} + \frac{B}{z} + \frac{C}{(z-2)}$$

$$\left. \begin{array}{l} a=0 \\ A = \frac{P(a)}{Q(a)} = \frac{2}{-2} = -1 \\ B = \frac{P'(a) - A \cdot Q'(a)}{Q(a)} = \frac{1 - (-1)}{2} = -1 \\ C = 2 \end{array} \right\}$$

$\overline{P(z) = (z-2) // Q(z) = (z-2)}$   
 $P'(z) = 1 // Q'(z) = 1$

$$C = 2 \Rightarrow C = \frac{P(c)}{Q(c)} = \frac{4}{4} = 1$$

$$X(z) = -\frac{1}{z^2} - \frac{1}{z} + \frac{1}{(z-2)}$$

$$x_z(z) = z \cdot x(z) = -\underbrace{\frac{1}{z}}_{-z^{-1}} - 1 + \frac{z}{z-2}$$

$$x_z(k) = (-\delta(k-1) - \delta(k) + 2^k) u(k)$$

$$x(z) = (z^{-1} \circ x_z(z))$$

$$x(k) = (-\delta(k-z) - \delta(k-1) + 2^{k-1}) u(k-1)$$

$$\{0, 0, 1, 4, 8\}$$

$$4) y(k) - 3y(k-1) - 4y(k-2) = x(k)$$

• Para una entrada  $x(k) = z^k u(k)$  //  $y(-1) = y(-2) = 0$

$$y(k) - 3y(k-1) - 4y(k-2) = z^k u(k)$$

$\downarrow Tz$

$$Y(z) - 3z^{-1} \cdot Y(z) - 4z^{-2} \cdot Y(z) = \frac{1}{1 - 2z^{-1}}$$

• saco factor común  $Y(z)$

$$Y(z) \underbrace{\left( 1 - 3z^{-1} - 4z^{-2} \right)}_{\rightarrow} = \frac{1}{1 - 2z^{-1}}$$

$$Y(z) = \frac{z}{(z - 3z^{-1} - 4z^{-2})(1 - 2z^{-1})}$$

• Numerador y denominador  $\propto z^3$

$$Y(z) = \frac{z^3}{(z^2 - 3z - 4)(z - 2)}$$

$$Y(z) = \frac{z^3}{(z-4)(z+1)(z-2)}$$

$$Y_1(z) = \frac{Y(z)}{(z)} = \text{Reservyo } \neq z$$

$$= \frac{z^2}{(z-4)(z+1)(z-2)} = \frac{A}{(z-4)} + \frac{B}{(z+1)} + \frac{C}{(z-2)}$$

a .. .

b .. —

c .. .

$$Y_1(z) = \frac{Y(z)}{(z)} = \frac{16}{(z-4)} + \frac{1/15}{(z+1)} + \frac{-2/3}{(z-2)}$$

$$y(z) = \frac{16z}{(z-4)} + \frac{1/15 z}{(z+1)} + \frac{-2/3 z}{(z-2)}$$

+  $z^{-1}$

$$x(k) = \left( 16 \cdot (4)^k + \frac{1}{15} (-1)^k - \frac{2}{3} (2)^k \right) \cdot o(k)$$

$$k=0 \Rightarrow y(0) = 1$$

$$k=1$$

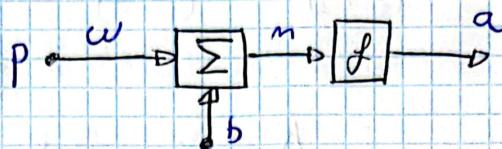
:

:

:

### Tema - 3 -

- escalares  $a, b$
- vectores  $\mathbf{a}, \mathbf{b}$
- Matrices  $A, B$



$$a = f(\omega p + b)$$

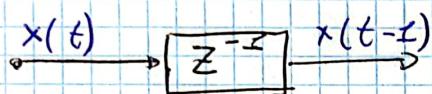
$$P = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$$

$$\omega = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$$

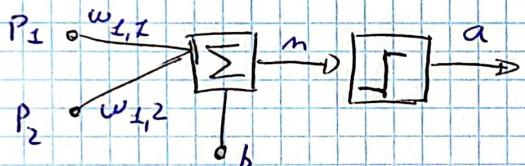
$f$  = funciones de activación

\* En múltiples dimensiones

- Elemento de Retardo =  $D = Z^{-1}$



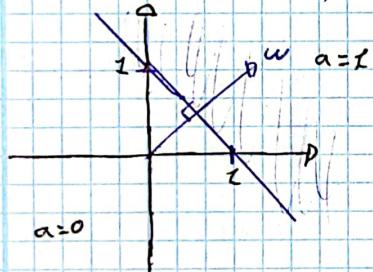
### Perceptrón



$b = b^{\circ}a = \text{desplazamiento}$

$\omega = \text{pendiente de la recta}$

$$a = \text{hardlim}(m_1) = \text{hardlim}(\mathbf{w}^T \mathbf{p} + b) = \text{hardlim}(\omega_{1,1} p_1 + \omega_{1,2} p_2 + b)$$



### - Proceso de entrenamiento

- Prueba con un  $\mathbf{w}$  Random

$$e(k) = (t(k) - a(k))$$

$$\text{só esperado } t \text{ y Real } \rightarrow e(k) = -\omega(k+1) = \omega(k)$$

$$\begin{matrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{matrix}$$

$$k = \underbrace{n^{\text{de pasos}}}_{\text{ }} \quad \begin{matrix} 0 \\ 1 \\ 2 \\ \vdots \end{matrix}$$

$$\begin{matrix} 0 \\ 1 \\ 2 \\ \vdots \end{matrix} \quad \begin{matrix} \text{época } 0 \\ \text{época } 1 \\ \text{época } 2 \\ \vdots \end{matrix}$$

$$= \omega(k) + P(k)$$

$$= \omega(k)$$

$$= \omega(k) - P(k)$$

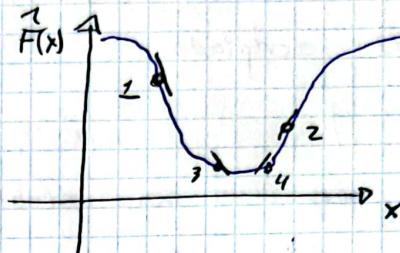
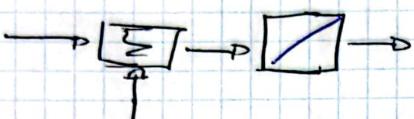
$$= \omega(k)$$

2

$$\omega(k+1) = \omega(k) + e(k) \cdot p(k)$$

## • ADALINE (Adaptive Linear Neuron)

- Parecido al perceptron pero con función lineal (1)
- Mismas limitaciones (problema de separación lineal)
- Widrow-Hoff learning
  - cálcula LMS (Error cuadrático medio) de forma recursiva



$\alpha \rightarrow$  Ratio de aprendizaje

$$\omega(k+1) = \omega(k) + 2\alpha e(k) p^T(k)$$

$$b(k+1) = b(k) + 2\alpha e(k)$$

constante  $\Rightarrow$  despreciable  $\Rightarrow$  percepción

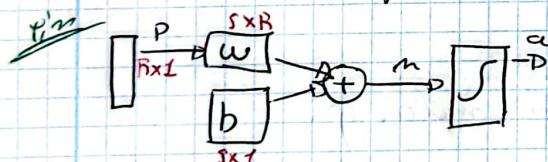
## • 2 tipos de aprendizaje

- Aprendizaje incremental
- Aprendizaje Batch

Metes todas las entradas a la vez

obtienes un vector de errores

## • Adaline con función sigmoidal ( $\sigma$ )



$$a = \text{log-sig} (w_p + b)$$

$$F = \sum_{i=1}^S (t_i - a_i)^2 = \sum_{i=1}^S e_i^2$$

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

$$m_i = \sum_{j=1}^R w_{ij} p_j + b_i$$

$$\omega_{i,j}(k+1) = \omega_{i,j}(k) - \alpha \frac{\partial F}{\partial w_{i,j}} = (-2e_i)(a_i(1-a_i))(p_j)$$

$$b(k+1) = b_i(k) - \alpha \frac{\partial F}{\partial b_i} = (-2e_i)(a_i(1-a_i))(1)$$

Si  $\sigma$

$$\begin{cases} \omega(k+1) = \omega(k) + 2\alpha e(k) (\alpha(k)(1-\alpha(k))) p(k) \\ b(k+1) = b(k) + 2\alpha e(k) (\alpha(k)(1-\alpha(k))) \end{cases}$$

- Las paradas se hacen por epoch (en cualquiera de los 2 tipos)

$\left\{ \begin{array}{l} \text{• El error cuadrático Medio (LMS) deja de caer} \\ \text{• Máximo número de iteraciones} \\ \text{• Máximo LMS aceptado} \end{array} \right.$

- Estabilidad

Para que un algoritmo sea estable:

$$0 < \alpha < \frac{1}{\lambda_{\max}}$$

$\lambda_{\max}$  es el Mayor valor de la matriz de entradas R

$$A=2R$$

### Madapime

- Igual que adalpine pero con múltiples capas

$\left[ \begin{array}{l} \text{- N° neuronas de la 1° capa depende del vector de entrada} \\ \text{- El resto de capas, prueba y error} \end{array} \right]$

(\* Para capas grandes, requerirán muchos datos para aprender \*)

- Con 3 capas es suficiente para un clasificador

### Backpropagation:

- Método de aprendizaje para redes multicapa

#### Proceso

- calculo la salida normal
- La comparo con la esperada
- Voy hacia atrás calculando el error de cada una

\* Estamos viendo aprendizaje supervisado \*

$$w_{i,j}^m(k+1) = w^m(k) - \alpha s_i^m a_j^{m-1}$$

$$b_{i,j}^m(k+1) = b^m(k) - \alpha s_i^m$$

$$s_i^M = -2(t-a) f^M(m_i^M) \quad / \quad s^M = -2 F^M(m^M)(t-a)$$

### Resumen

1º Programar la entrada hacia adelante

$$a^o = p$$

$$a^{m+1} = f^{m+1}(W^{m+1} a^m + b^{m+1})$$

$$a = a^M$$

2º Programar las sensibilidades hacia atrás

$$s^M = -2 F^M(m^M)(t-a)$$

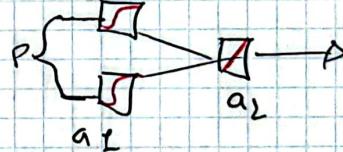
$$s^m = f^m(m^m)(W^{m+1})^T s^{m+1}$$

3º Actualizar los pesos y bias

$$W^{m+1}(k+1) = W^m(k) - \alpha s^m (a^{m-1})^T$$

$$b^{m+1}(k+1) = b^m(k) - \alpha s^m$$

$$g(p) = 1 + \sin\left(\frac{\pi}{4}p\right) \quad / \quad 2 \text{ de entrada y } 1 \text{ oculta}$$



#### • Random

$$w^1(0) = \begin{bmatrix} -0.22 \\ -0.41 \end{bmatrix}, \quad w^2(0) = \begin{bmatrix} 0.09 & -0.17 \end{bmatrix}$$

$$a^o = p_{1,0} = 1$$

$$b^1(0) = \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix} \quad | \quad b^2(0) = [0.48]$$

#### • Salida 1<sup>o</sup> capa

$$a^1 = \text{logsig}\left(\begin{bmatrix} -0.22 \\ -0.41 \end{bmatrix} \cdot I + \begin{bmatrix} -0.48 \\ -0.13 \end{bmatrix}\right) = \begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix}$$

#### • Salida 2<sup>o</sup> capa

$$a^2 = \text{purelin}\left(\begin{bmatrix} 0.09 & -0.17 \end{bmatrix} \cdot \begin{bmatrix} 0.321 \\ 0.368 \end{bmatrix} + [0.48]\right) = [0.445]$$

#### • Calculo el error

$$e = t - a = 1.261$$

### • Backpropagate

$$f'(m) = \frac{\partial}{\partial m} \left( \frac{1}{1+e^{-m}} \right) = (1-a^2)(a')$$

$$f''(m) = \frac{\partial}{\partial m} (m) = 1$$

### • Propagated sensitivity

$$S^2 = -2 F^2(m^2)(t-a) = -2 [f^2(m^2)](-2'264) = -2[1] \cdot (-2'264) = -2'522$$

$$S^2 = F^2(m^2)(W^2)^T S^2 = \begin{bmatrix} (1-a_1^2)(a_1') & 0 \\ 0 & (1-a_2^2)(a_2') \end{bmatrix} \begin{bmatrix} 0'09 \\ -0'12 \end{bmatrix} [-2'522] = \begin{bmatrix} -0'0475 \\ 0'0997 \end{bmatrix}$$

### • Actualization pass

$$\begin{aligned} W^2(z) &= W^2(0) - \alpha S^2(a^2)^T = [0'69 \quad -0'12] - 0'2 \cdot [-2'522] [0'321 \quad 0'368] = \\ &= [0'171 \quad -0'0272] \end{aligned}$$

$$b^2(z) = b^2(0) - \alpha S^2 = [0'48] - 0'2 \cdot [-2'522] = [0'732]$$

$$w^1(z) = w^1(0) - \alpha S^1(a^0)^T$$

$$b^1(z) = b^1(0) - \alpha S^1$$

- Se recomienda tener menos de un 15% de neuronas
  - 100 valores de entrada  $\Rightarrow$  15 neuronas

### Variaciones de Backpropagation

#### Método heurístico

##### - Momentum

Añadir un momento de inercia

##### - Ratio de aprendizaje variable

$\alpha$  no es fijo  $\Rightarrow \alpha(k)$

' $\alpha$ ' Dependerá del momento en el que esté

- Si la pendiente es alta  $\Rightarrow \alpha$  es pequeño

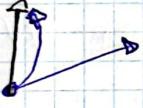
#### Método entrenamiento de una red

##### - Gradiente conjugado



Entre 2 pasos busca en  $L$

##### \* - Levenberg - Marquardt Algoritmo



Pro: Rápido entrenamiento

Cont: Requiere espacio

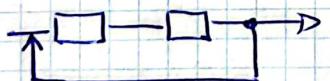
### Redes Dinámicas

- Redes que tienen retardos (Memoria)

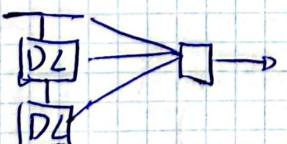
$\boxed{TDZ} \rightarrow$  Si veo esto  $\Rightarrow$  Red dinámica

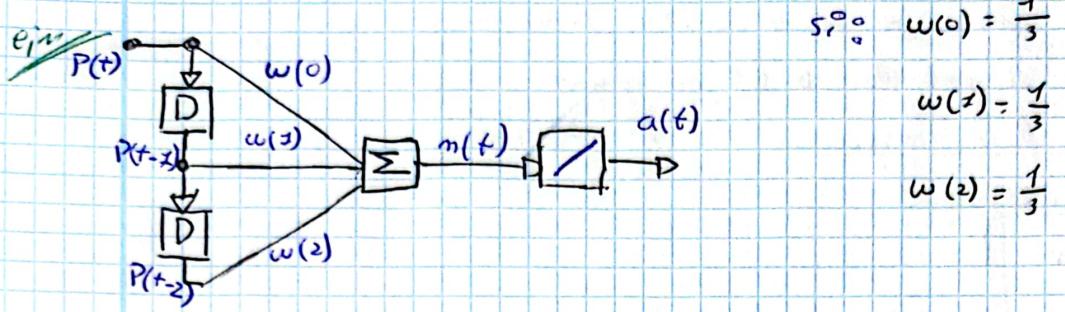
- 2 tipos

#### BPTT (Backpropagation-through time)



#### RTRL (Real-time recurrent learning)





$$s_{r,0} = \frac{1}{3}$$

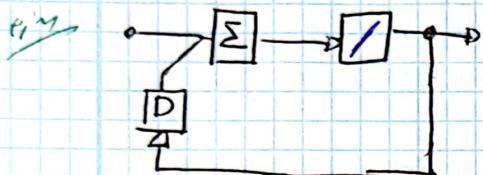
$$w(0) = \frac{1}{3}$$

$$w(1) = \frac{1}{3}$$

$$w(2) = \frac{1}{3}$$

$$a(t) = m(t) = \sum w(d) \cdot p(t-d) =$$

$$= w(0) \cdot p(t) + w(1) \cdot p(t-1) + w(2) \cdot p(t-2)$$



$$\frac{\partial a(t)}{\partial w_{ii}} = p(t) + \rho w_{ii}(t) \underbrace{\frac{\partial a(t-z)}{\partial w_{ii}}}_{\text{Parte dinâmica}}$$

### Principios del aprendizaje dinámico

RTRL

$$\frac{\partial a(t)}{\partial x^T} = \frac{\partial a(t)}{\partial x^T} + \frac{\partial a(t)}{\partial a^T(t-1)} \cdot \frac{\partial a(t-1)}{\partial x^T}$$

BPTT

$$\frac{\partial F}{\partial a(t)} = \frac{\partial F}{\partial a(t)} + \frac{\partial a(t+1)}{\partial a^T(t)} \cdot \frac{\partial F}{\partial a(t+1)}$$

- Generalización
- { • training 70%
- Validation 15%
- Test 15%

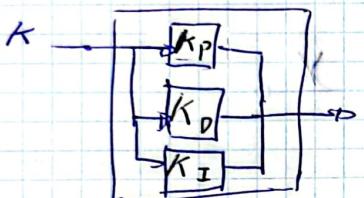
- Prueba parada

cross-validation method

## Tema - 4 - Control neuronal

### • Control básico

- { - Todo o nada "bang-bang"
- PID "Proporcional-Integral-Derivativo"



### • Control moderno

- { - control óptimo
- control inteligente
- { control neuronal  
control bonoso

### • Control Neuronal

- Sistema de control que usan las redes neuronales
- Sirve para sistemas complejos que se desconoce su modelo de comportamiento
- 2 procesos de ajustes

#### • Aprendizaje off-line

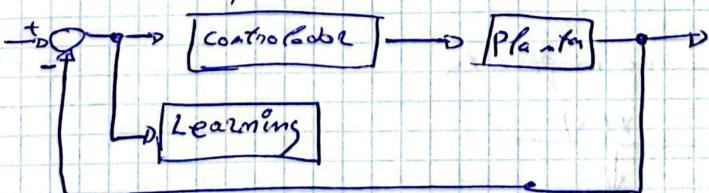
Adquisición previa de conocimiento

#### • Aprendizaje on-line

Adquisición dinámica

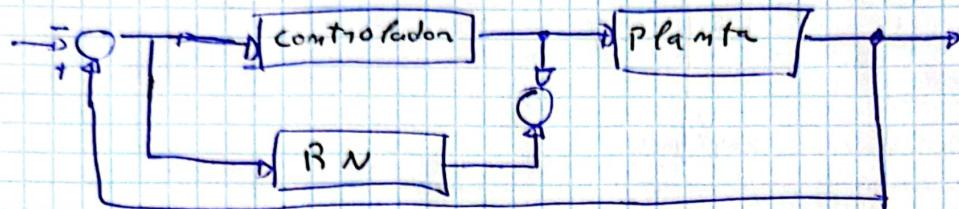
### • Control directo

- La RNN implementa directamente el controlador



### • Control supervisado

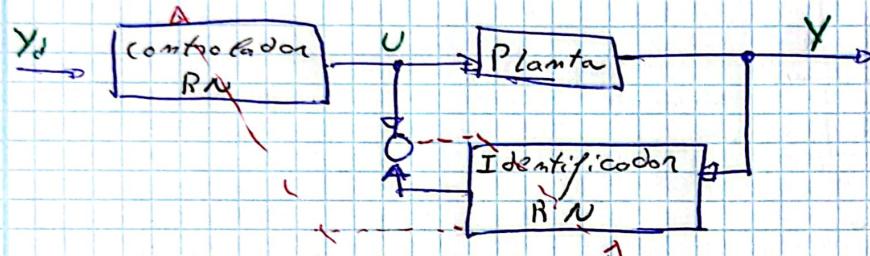
- La RN actua como aproximación universal del comportamiento del controlador



- Una vez entrenada la RN, se sustituye por el controlador

### • Control inverso

- ① Identifica la planta mediante RN
- ② Diseña un controlador a partir de la RN



$$J_{\text{planta}} = \frac{\partial y}{\partial u} \quad \text{Jacobiiano}$$

Derivada de la salida con respecto a la derivada de la entrada.

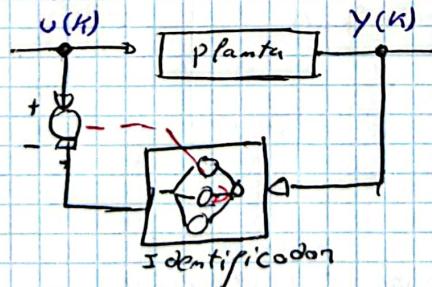
### \* • Identificaciones con RN

- Usar RN para modelar plantas desconocidas

#### - Configuraciones

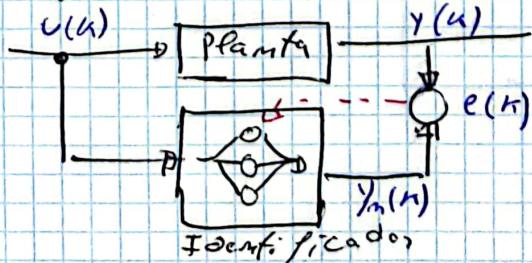
##### • Serie

Identifica la inversa de la función de la planta



##### • Paralelo

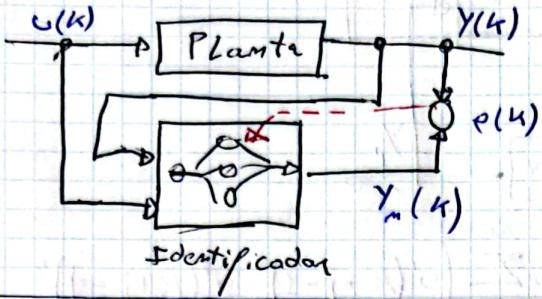
Identifica la función de la planta



## • Serie / Paralelo

Implementa la función F

+ utilizada



- \* • Para entrenar el controlador necesito saber la planta
- \* • Para calcular los pesos del controlador, necesito los pesos del identificador
  - si conocemos parte de la planta, debemos aadirle pesos bajos la complejidad de la RN

## • Sistemas SISO con RN's

Arquitecturas

- Controlador predictivo
- Controlador NARIMA-L2
- Controlador con modelo de referencia

### → Controlador predictivo

- Identificación del sistema (off-line)

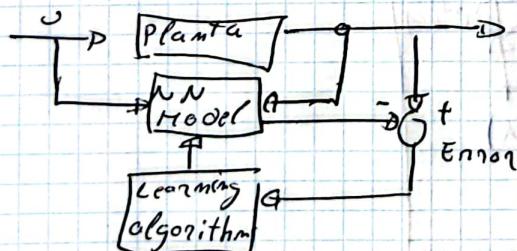
- Usar RN para predecir salida de planta

- Diseño del controlador (on-line)

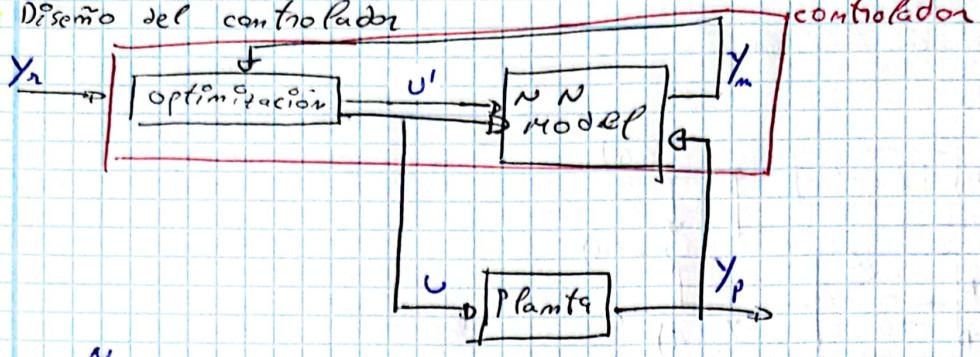
- Utiliza el modelo neuronal de la planta anterior

- Calcula la entrada de control óptima

- Identificación planta



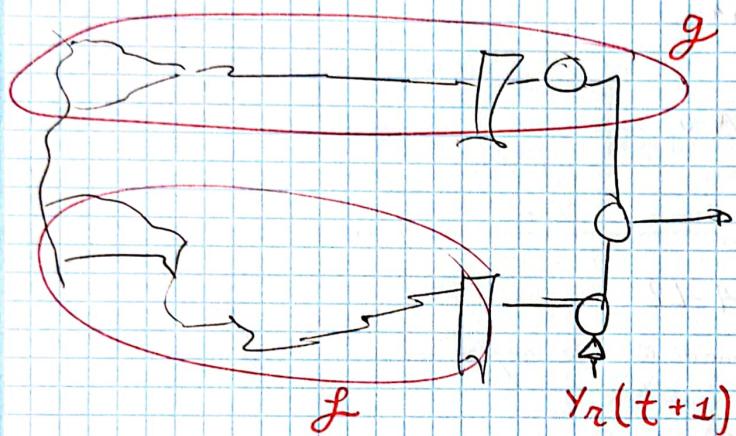
• Diseño del controlador



$$J = \sum_{j=N_x}^{N_2} (y_n(k+j) - y_m(k+j))^2 + p \sum_{j=1}^{N_c} (u'(k+j-1) - u'(k+j-2))^2$$

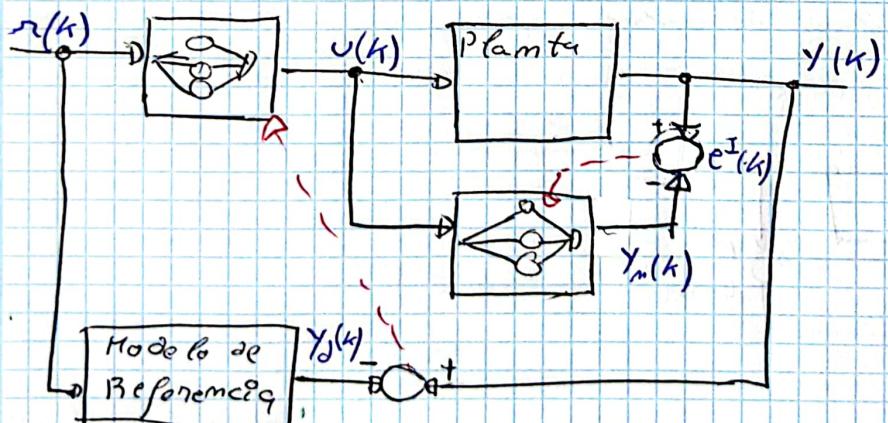
→ controlador NARIMA-L2

- Identificación de la planta
- modelo



→ Controlador con un modelo de Referencia

- tenemos un neuroidentificador y un neurocontrolador
- Intentamos que la planta sea como un modelo lineal



$$E^I(k) = \frac{1}{2} (y(k) - y_m(k))^2 = \frac{1}{2} (e^I(k))^2$$

$$J(k) \approx \frac{\partial y_m(k)}{\partial u(k)}$$

Ajustes de pesos

- 2 pasos de aprendizaje distintos

$L_1$  identificador

$L_2$  controlador

$$\Delta w_i^I(k) = -\alpha_1 \cdot (-e^I(k)) \cdot \frac{\partial y_m(k)}{\partial w_i^I(k)}$$

Depende de la  
topología de la red

$$\Delta w_i^C(k) = -\alpha_2 \cdot e^C(k) \cdot J(k) \cdot \frac{\partial u(k)}{\partial w_i^C(k)}$$

- Sistemas MIMO con RN's