REDES NEURONALES ARTIFICIALES

María Jesús de la Fuente
Dpto. Ingeniería de Sistemas y Automática
Universidad de Valladolid

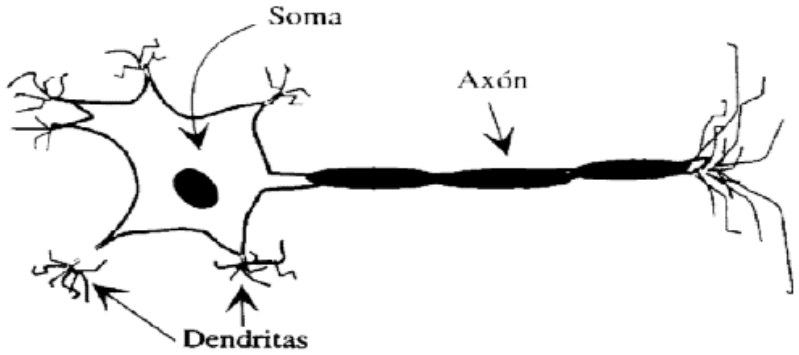
ÍNDICE

- Introducción
- Clasificación de redes neuronales:
  - Estructura
  - Entrenamiento
- Aplicación de las redes neuronales a la identificación de sistemas
- Las redes neuronales en el control

REDES NEURONALES

- Neurona: base del funcionamiento del cerebro.
- Sistema de procesamiento cerebral de la información:
  - Complejo, No lineal y Paralelo.

Fisiología de una neurona elemental

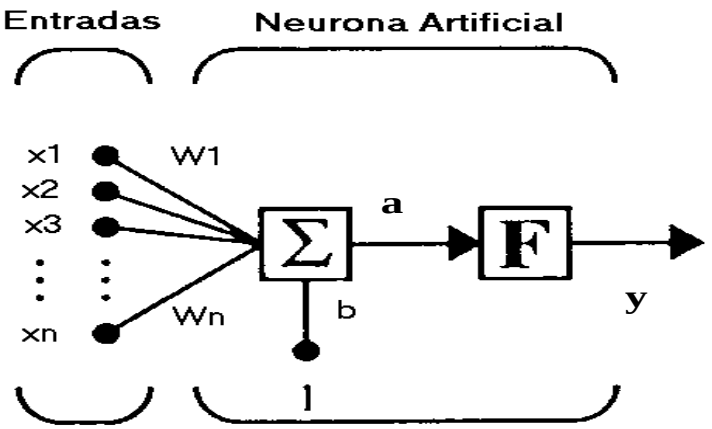


- Elementos de que consta: sinapsis, axón, dentritas y soma o cuerpo

NEURONA ARTIFICIAL

- Neurona artificial: unidad de procesamiento de la información, es un dispositivo simple de cálculo que ante un vector de entradas proporciona una única salida.

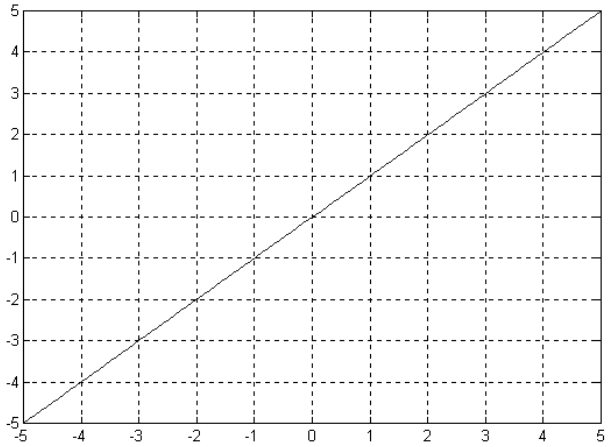
- Elementos:
  - Conjunto de entradas, *xj*
  - Pesos sinápticos, *wi*
  - Función de activación:
    $w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n = a$
  - Función de transferencia:
    $y = F(w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n)$

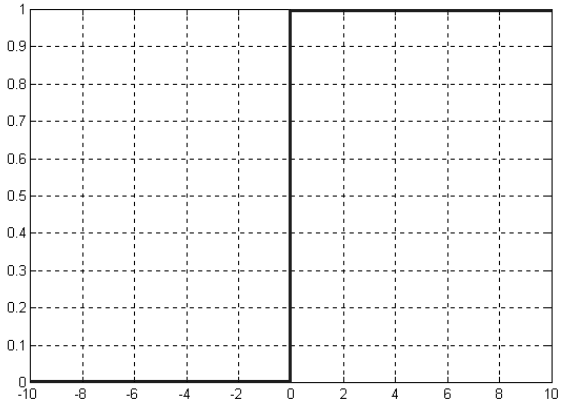


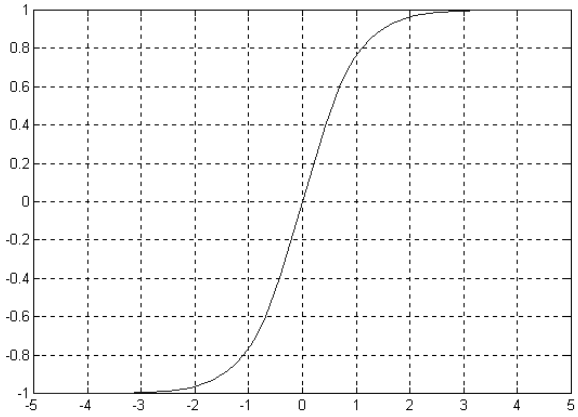
  - Bias o polarización: entrada constate de magnitud 1, y peso b que se introduce en el sumador

NEURONA ARTIFICIAL

- Principales funciones de transferencia:
  - Lineal: *y=ka*
  - Escalón: *y = 0 si a<0; y=1 si a>=0*
  - Sigmoide
  - Gaussiana.

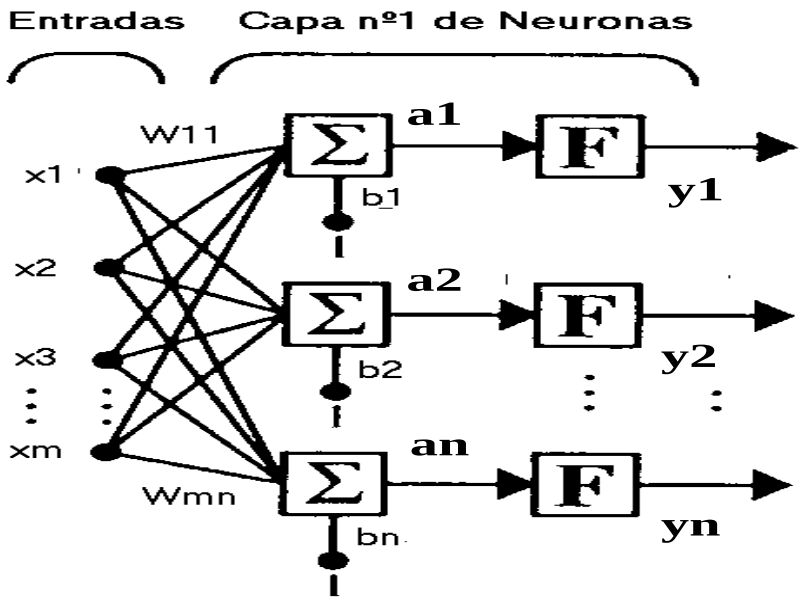






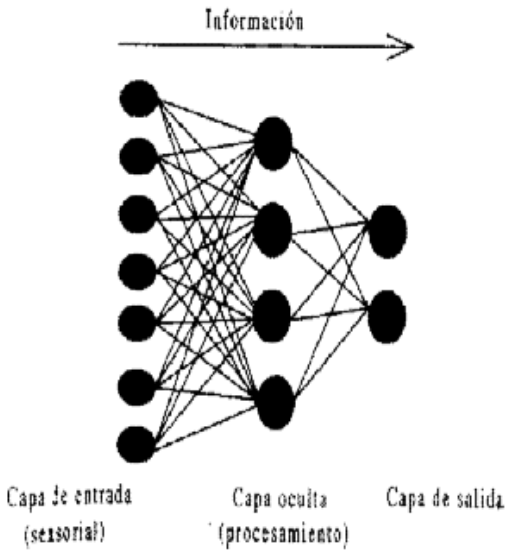
RNA de una capa

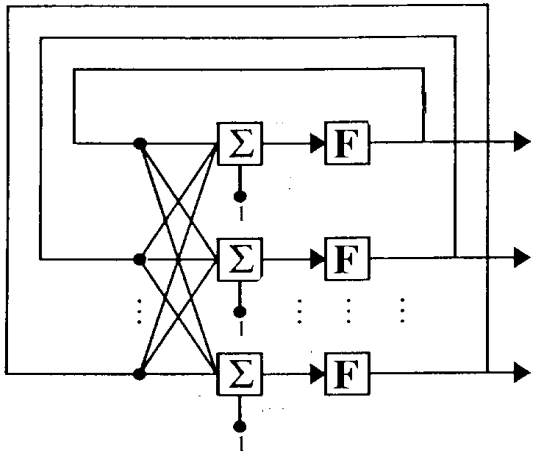
- Una neurona aislada dispone de poca potencia de cálculo.
- Los nodos se conectan mediante la sinapsis
- Las neuronas se agrupan formando una estructura llamada capa.
- Los pesos pasan a ser matrices W (n x m)
- La salida de la red es un vector: $Y=(y_1, y_2, \ldots, y_n)^T$
- *Y=F(W·X+b)*



RNA Multicapa

- Redes multicapa: capas en cascada.
- Tipos de capas:
  - Entrada
  - Salida
  - Oculta
- No hay realimentación => *red feedforward*
  - Salida depende de entradas y pesos.
- Si hay realimentación => *red recurrente*
  - Efecto memoria
  - Salida depende también de la historia pasada.
- Una RNA es un aproximador general de funciones no lineales.



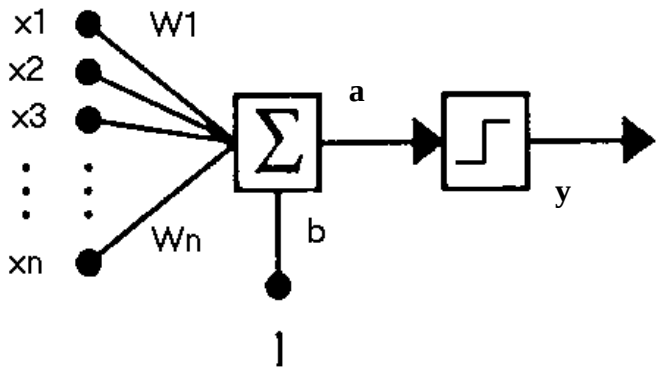


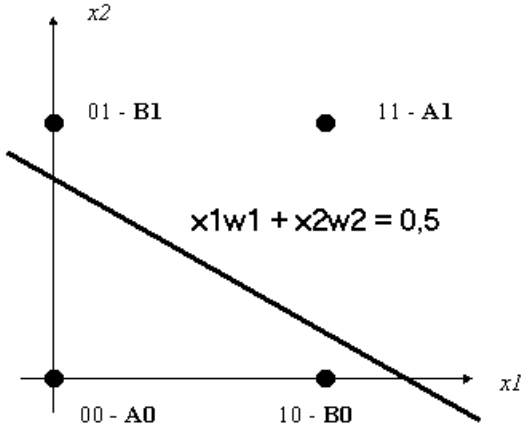
Entrenamiento I

- Entrenamiento: proceso de aprendizaje de la red.
- Objetivo: tener un comportamiento deseado.
- Método:
  - Uso de un algoritmo para el ajuste de los parámetros libres de la red: los pesos y las bias.
  - Convergencia: salidas de la red = salidas deseadas.
- Tipos de entrenamiento:
  - Supervisado.
    - Pares de entrenamiento: entrada - salida deseada.
    - Error por cada par que se utiliza para ajustar parámetros
  - No-supervisado.
    - Solamente conjunto de entradas.
    - Salidas: la agrupación o clasificación por clases
  - Reforzado.

Perceptrones

- McCulloch y Pitts, en 1943, publicaron el primer estudio sobre RNA.
- El elemento central: perceptrón.





- Solo permite discriminar entre dos clases linealmente separables: XOR.
  - $0.5 = a = w_1 \cdot x_1 + w_2 \cdot x_2$
  - No hay combinación de x1 y x2 que resuelva este problema.
- Solución: más capas o funciones de transferencia no lineales.

Aprendizaje del Perceptrón.

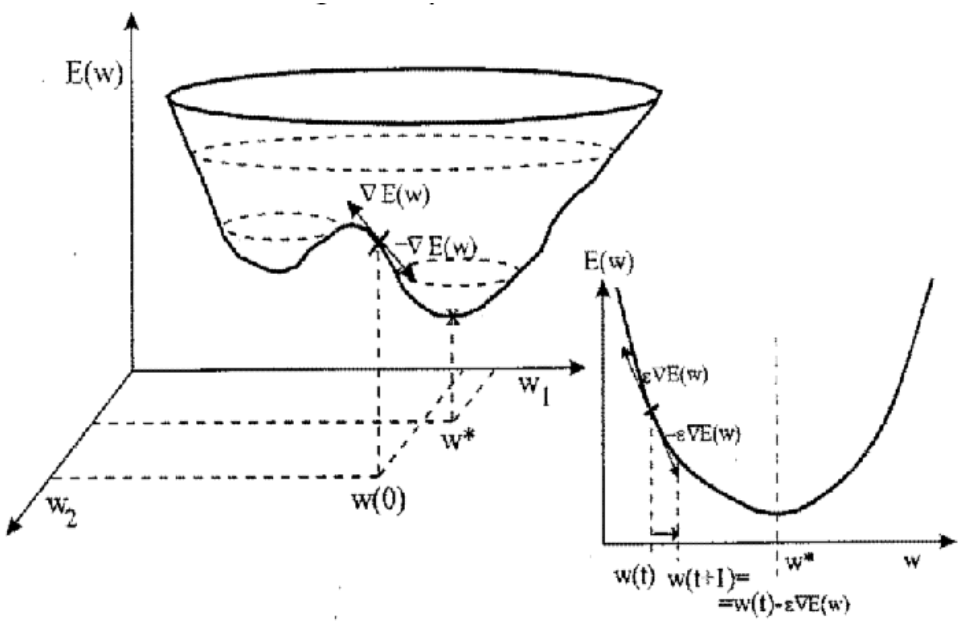
- Algoritmo supervisado:
  - Aplicar patrón de entrada y calcular salida de la red
  - Si salida correcta, volver a 1
  - Si salida incorrecta
    - 0 → sumar a cada peso su entrada
    - 1 → restar a cada peso su entrada
  - Volver a 1
- Proceso iterativo, si el problema es linealmente separable este algoritmo converge en un tiempo finito.
- Nos da los pesos y las bias de la red que resuelve el problema.

Regla delta

- Generalización del algoritmo del perceptrón para sistemas con entradas y salidas continuas.
- Se define: δ=**T-A=** $e_k(n)$ (salidas deseadas - salidas de la red).
- Minimiza una función de coste basada en ese vector de error:

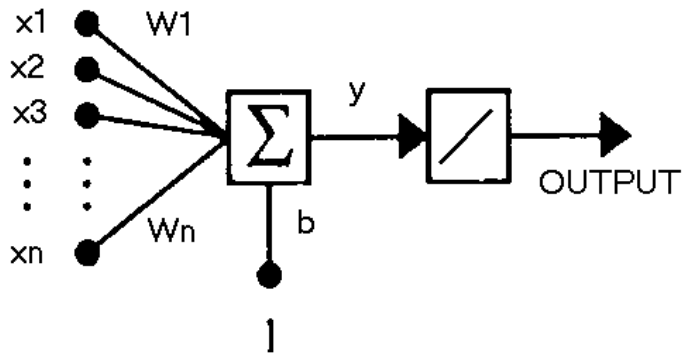
$$J = E\left[\frac{1}{2}\sum_k e_k^2(n)\right]$$

- $\Delta_i = \delta\, l_r\, x_i$
- $W_i(n+1) = W_i(n) + \Delta_i$
- Razón de aprendizaje l_r
- Si las neuronas son lineales=> un único mínimo



Redes Neuronales Lineales.

- Función de transferencia lineal.
- Algoritmo de entrenamiento de Widrow-Hoff o Delta, tiene en cuenta la magnitud del error.
- Entrenamiento:
  - Suma de los cuadrados de los errores sea mínima.
  - Superficie de error con mínimo único.
  - Algoritmo tipo gradiente.

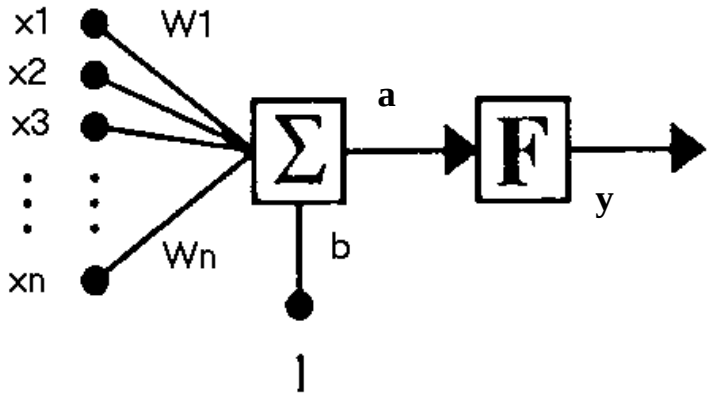


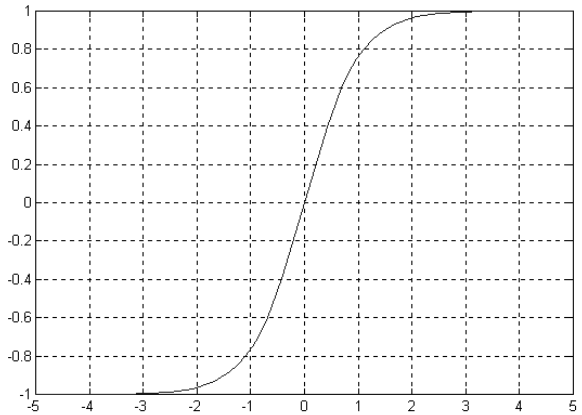
- Aproximan funciones lineales.

Backpropagation

- Clave en el resurgimiento de las redes neuronales.
- Primera descripción del algoritmo fue dada por Werbos en 1974
- Generalización del algoritmo de Widrow-Hoff para redes multicapa con funciones de transferencia no-lineales y diferenciables.
- 1989 Hornik, Stinchcombe y White
  - Una red neuronal con una capa de sigmoides es capaz de aproximar cualquier función con un número finito de discontinuidades
- Propiedad de la generalización.
- La función de transferencia es no-lineal, la superficie de error tiene varios mínimos locales.

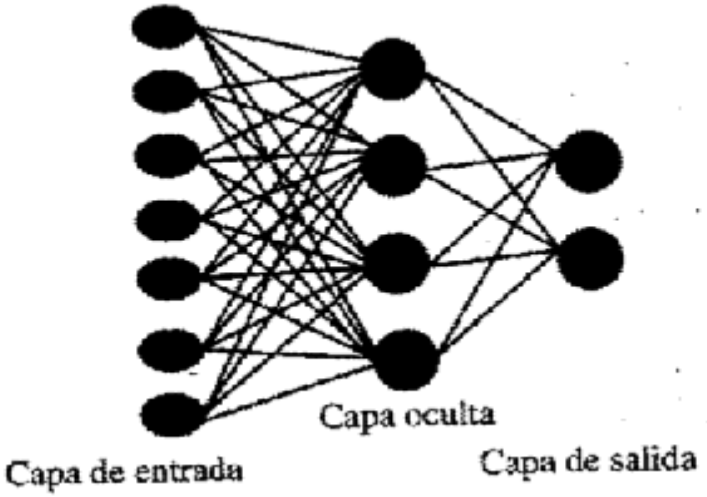
Red Perceptron Multicapa (MLP)





- Función acotada, monótona creciente y diferenciable.

- Red de tipo feedforward.
- Suficiente con dos capas.



Algoritmo backpropagation I

- Descripción:

Adelante

- Tras inicializar los pesos de forma aleatoria y con valores pequeños, seleccionamos el primer par de entrenamiento.
- Calculamos la salida de la red
- Calculamos la diferencia entre la salida real de la red y la salida deseada, con lo que obtenemos el vector de error

Atrás

- Ajustamos los pesos de la red de forma que se minimice el error
- Repetimos los tres pasos anteriores para cada par de entrenamiento hasta que el error para todos los conjuntos de entrenamiento sea aceptable.

- Descenso por la superficie del error
- Cálculo de derivadas del error respecto de los pesos y de las bias.

Algoritmo backpropagation II

- Detalles:
  - SSE: $E=\Sigma E_p=\Sigma (y_{pk}-o_{pk})^2$
  - $\Delta w_{ij}=-\eta \partial E/ \partial w_{ij}$
- Pasos:
  - Inicialización:
    - Construcción de la red.
    - Inicialización aleatoria de pesos y umbrales (-0.5, 0.5)
    - Criterio de terminación (número máximo de iteraciones,…).
    - Contador de iteraciones n=0.
  - Fase hacia delante:
    - Calcular la salida de la red para cada patrón de entrada.
    - Calcular el error total cometido (SSE)
    - Si la condición de terminación se satisface, parar
  - Fase hacia atrás:

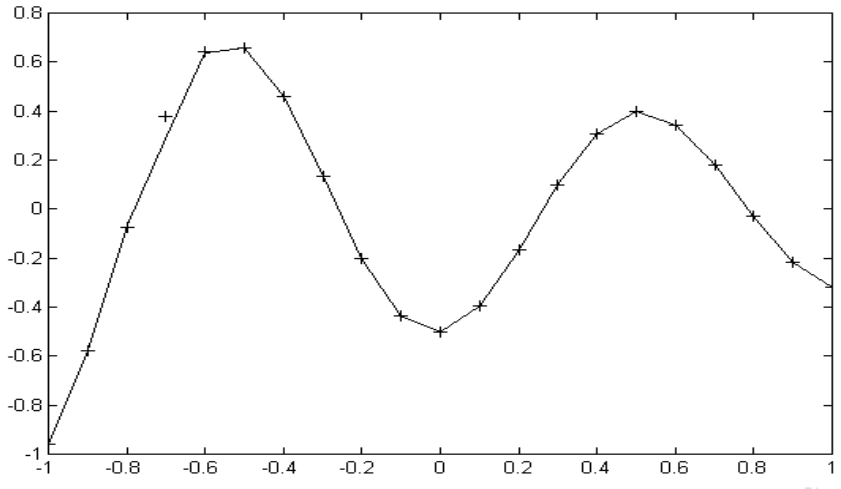
Algoritmo backpropagation III

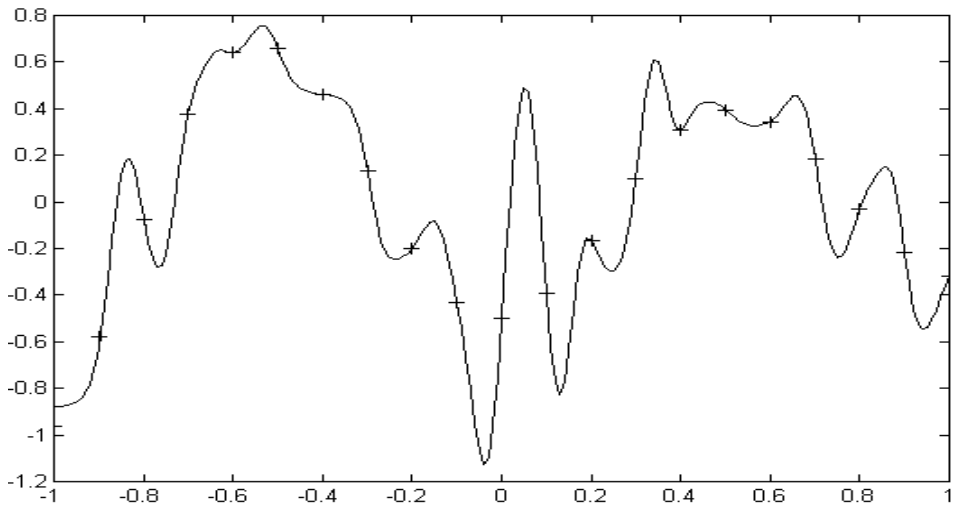
- Fase hacia atrás:
  - Incrementar el contador n=n+1.
  - Paca cada neurona de salida calcualr: $\delta_k=(o_k-y_k)f'(net_k)$ donde $net_j=\Sigma_i w_{ij}x_i+b_j$
  - Para cada unidad oculta, calcular $\delta_j=f'(net_j)\Sigma_k\delta_k w_{jk}$
  - Actualizar pesos: $\Delta w_{ij}(n+1)=\eta\delta_j o_i+\alpha\Delta w_{ij}(n)$
  - Volver a la fase hacia delante.

- Inconvenientes del algoritmo backpropagation:
  - Tiempo de entrenamiento no acotado.
  - Dependiente de las condiciones iniciales:
    - Parálisis de la red.
    - Mínimos locales.

Algoritmo Backpropagation IV

- Underfitting.
- Memorización o Sobreaprendizaje.
- Caracterización de la red. ¿Cuantas capas, cuantas neuronas en cada capa,...?





Redes Neuronales no supervisadas I

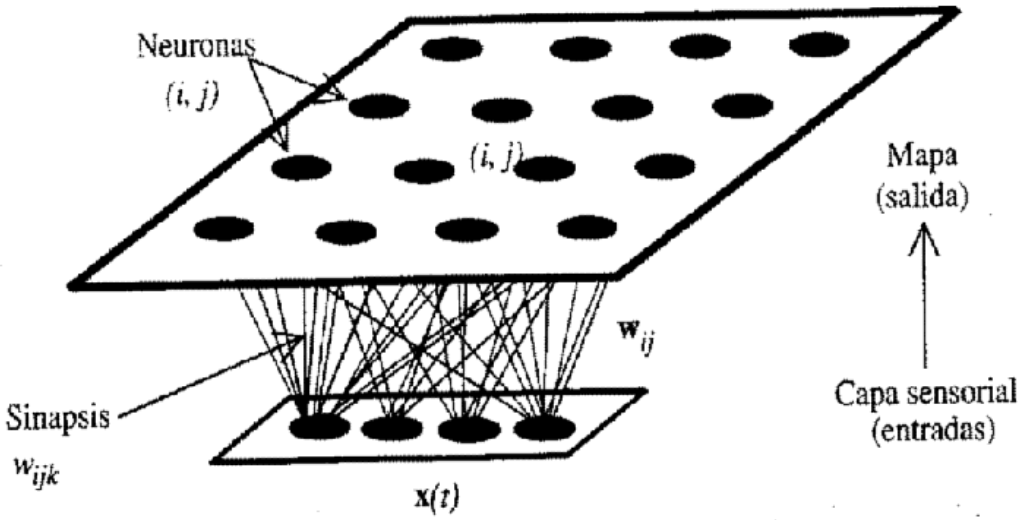
- Autoorganizativas: durante el proceso de aprendizaje la red debe descubrir por si misma regularidades o categorías => la red debe autoorganizarse en función de las señales procedentes del entorno.
- Mapa de Rasgos Autoorganizados, SOM (Kohonen, 80)
- Características:
  - Red competitiva
  - Arquitectura unidireccional de dos capas:
    - Capa de entrada: m neuronas una por cada vector de entrada.
    - Capa segunda se realiza el procesamiento, formando el mapa de rasgos. Tiene nx X ny neuronas operando en paralelo.
    - Todas las neuronas de entrada están conectadas a las neuronas de la segunda capa, a través de los pesos wij

Redes Neuronales No-Supervisadas II

- Cada neurona (i,j) calcula la similitud entre el vector de entradas y su vector de pesos
- Vence la neurona cuyo vector de pesos es más similar al vector de entrada.

$$d(w_g, x) = min_{ij} d(w_{ij}, x)$$

- Cada neurona sirva para detectar alguna característica del vector de entrada.
- Función de vecindad: relación entre neuronas próximas en el mapa.



RNA no supervisadas III

- Aprendizaje:
  - Inicialización de los pesos wij
  - Presentación de las entradas x(t)
  - Cada neurona calcula, la similitud entre su vector de pesos wij y el vector de entrada x, usando la distancia Euclídea

$$d^2(w_{ij}, x) = \sum_{k=1}^{n} (w_{ijk} - x_k)^2$$

  - Determinación de la neurona ganadora: $Ganadora = min_j \quad d_j^2$
  - Actualización de los pesos de la neurona ganadora y sus vecinas

$$w_{ijk}(t+1) = w_{ijk}(t)\alpha(t)h(|i-g|, t)(x_k(t) - w_{ijk}(t))$$

  - Las demás neuronas no actualizan su peso
  - Si se ha alcanzado el número de iteraciones parar, si no volver al paso 2.

VENTAJAS

- Ventajas de las RNA:
  - Aprendizaje adaptativo: lo necesario es aplicar un buen algoritmo y disponer de patrones (pares) de entrenamiento.
  - Auto-organización => conduce a la generalización
  - Tolerancia a fallos: las redes pueden aprender patrones que contienen ruido, distorsión o que están incompletos.
  - Operación en tiempo real: procesan gran cantidad de datos en poco tiempo.
  - Facilidad de inserción en tecnología ya existente.

APLICACIONES

- Detección de patrones.
- Filtrado de señales
- Segmentación de datos
- Control
- Identificación.

Redes Neuronales en identificación de sistemas

Identificación de sistemas

- La identificación consiste en calcular un modelo del sistema en base a datos experimentales.

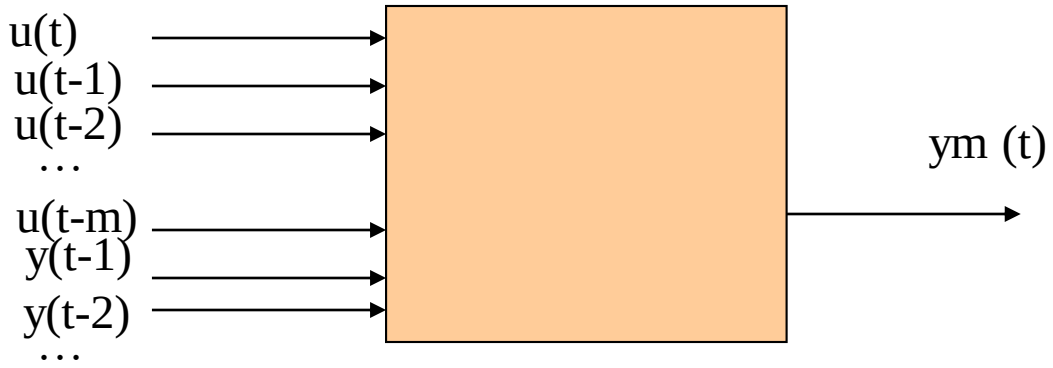
- Pasos:
  - Seleccionar una clase de modelos (CARIMA, Box-Jenkis,...)
  - Obtener un conjunto de datos experimentales
  - Seleccionar un modelo de la clase elegida
  - Estimar los parámetros (método de Identificación: LS,RLS,IV,...)
  - Validación (exactitud, adecuación de uso)

RNA que representan el tiempo

- Representación del tiempo.
  - Tratamiento de señales.
  - Identificación de modelos dinámicos
  - Control de sistemas.
- Redes dinámicas:
  - Respuesta a:
    - Las entradas actuales
    - La historia pasada del sistema.
  - Dotar de memoria a la red:
    - Introduciendo directamente en la red tanto la señal actual como los valores pasados.
    - Mediante conexiones recurrentes.

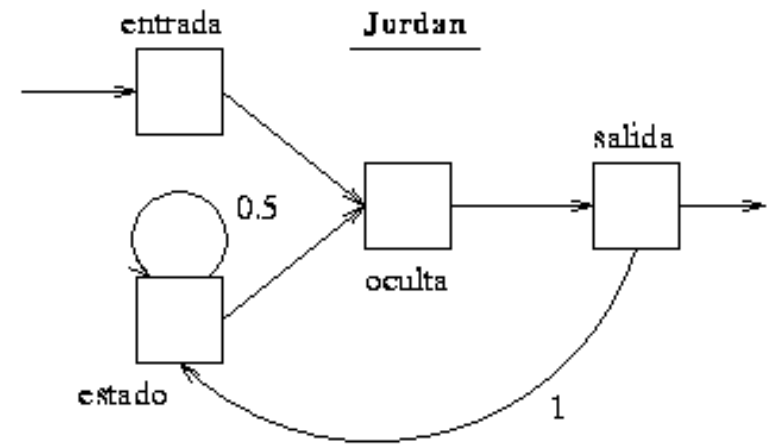
Red PML con ventana temporal

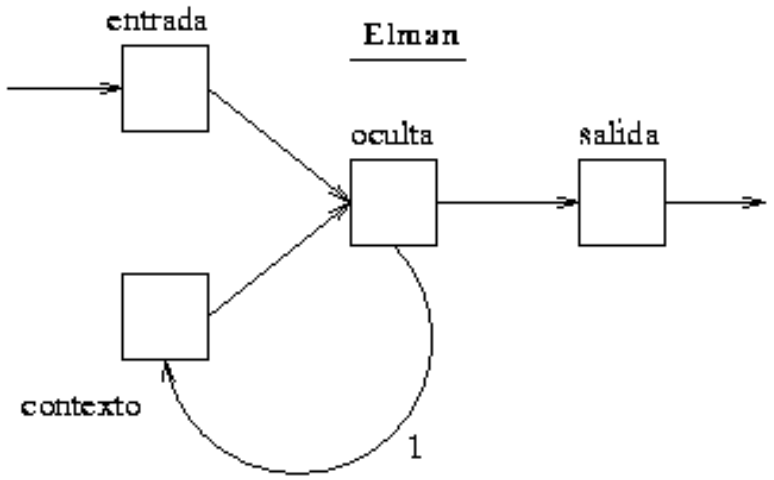
- Ventanas de datos pasadas de las entradas y de las salidas.
- Ventajas:
  - Algoritmo simple es suficiente
  - No problemas de realimentación
- Desventajas
  - Información útil debe "caber" en la ventana temporal
  - Muchas entradas ⇒ Sobreparametrización



Redes neuronales recurrentes

- Redes neuronales parcialmente recurrentes:
  - Conexiones recurrentes con valores fijos
  - Algoritmo de aprendizaje "ve" una red perceptrón multicapa
  - Ejemplos:
    - Jordan
    - Elman
- Redes neuronales recurrentes:
  - Todas las neuronas interconectadas
  - Computacionalmente costoso





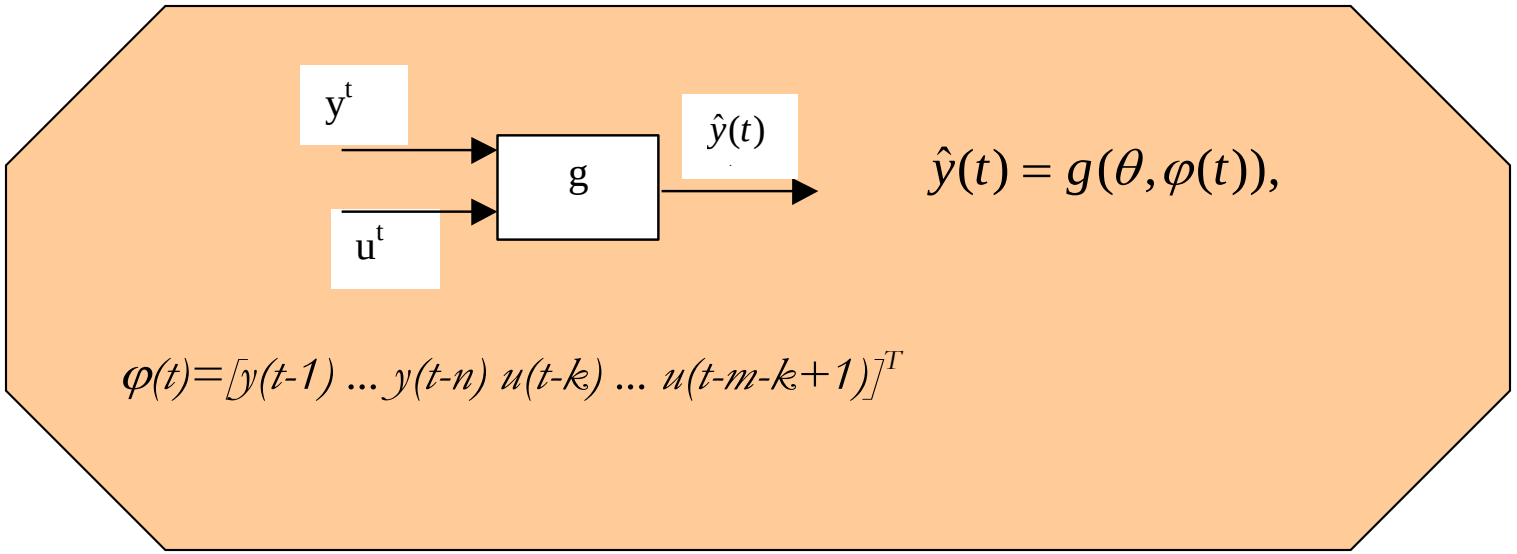
Capas contexto y oculta del mismo tamaño y conexion uno a uno.

Estructuras neuronales para la identificación

- Determinación o elección de la estructura del modelo.
- ¿Es necesario un modelo neuronal?
- Nos basamos en modelos establecidos en el caso lineal
- Diseño:
  - Variables que forman parte del regresor $\varphi(t)$
  - Función no-lineal $g(\cdot,\cdot)$ desde el espacio de regresiones al espacio de salida NO en modelos lineales
  - $y(t)=g(\theta,\varphi(t))+e(t)$
- Estructura de caja negra: modelo de entrada- salida.
- Elementos del regresor:
  - Entradas pasadas *u(t-k)*
  - Salidas pasadas medidas: *y(t-k)*
  - Salidas pasadas calculadas por el modelo: $\ddot{y}_u(t-k|\theta)$
  - Residuos pasados calculados: $\varepsilon_u(t-k)=y(t-k)-\ddot{y}_u(t-k|\theta)$

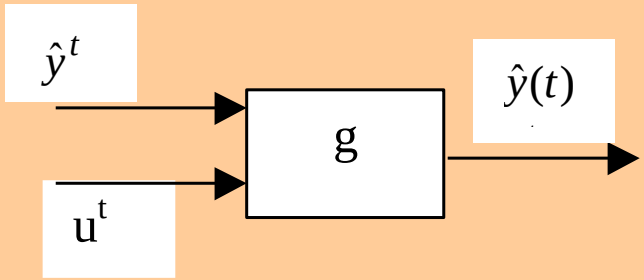
Modelo NARX

- Ventajas:
  - Puede aproximar cualquier sistema no-lineal arbitrariamente bien
  - No recurrente.
- Desventajas:
  - Vector de regresión puede ser grande
  - No se modela el ruido



Modelo NOE

- Corresponde a una red recurrente, ya que parte de las entradas constituye la salida de la propia red.
  - Comprobación difícil para modelo de predicción estable
  - Entrenamiento laborioso por cálculo correcto de gradientes



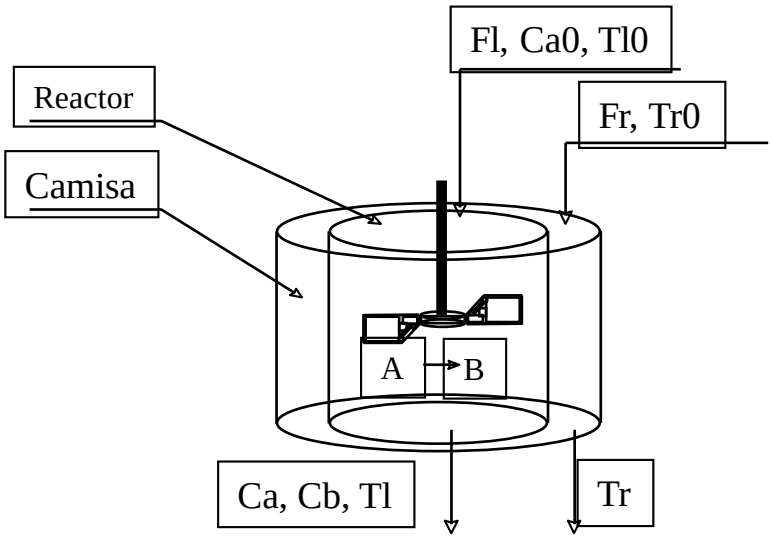
$$\varphi(t) = [\hat{y}(t-1|\theta) \ldots \hat{y}(t-n|\theta)\, u(t-k) \ldots u(t-m-k+1)]^T$$

$$\hat{y}(t|\theta) = g(\varphi(t), \theta)$$

Validación

- Validación: es el proceso de comprobación de la utilidad de modelo obtenido:
  - Si el modelo concuerda con los datos observados
  - Si servirá al propósito para el que fue creado
  - Si describe el sistema real
- Enfoque neuronal:
  - Conjunto de datos de entrenamiento
  - Conjunto de datos de test.
  - Conjunto de datos de validación.
- Enfoque basado en correlaciones:
  - Test de blancura de los residuos
- …

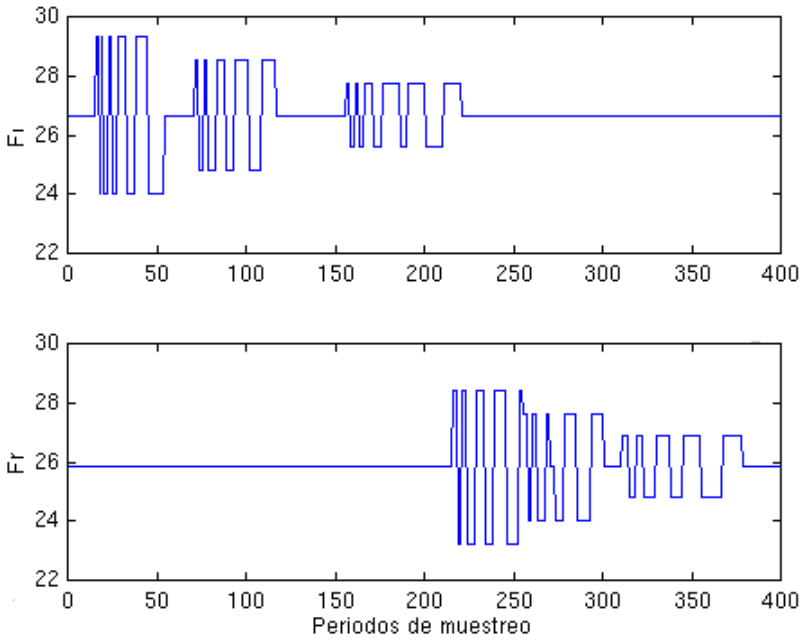
Ejemplo I

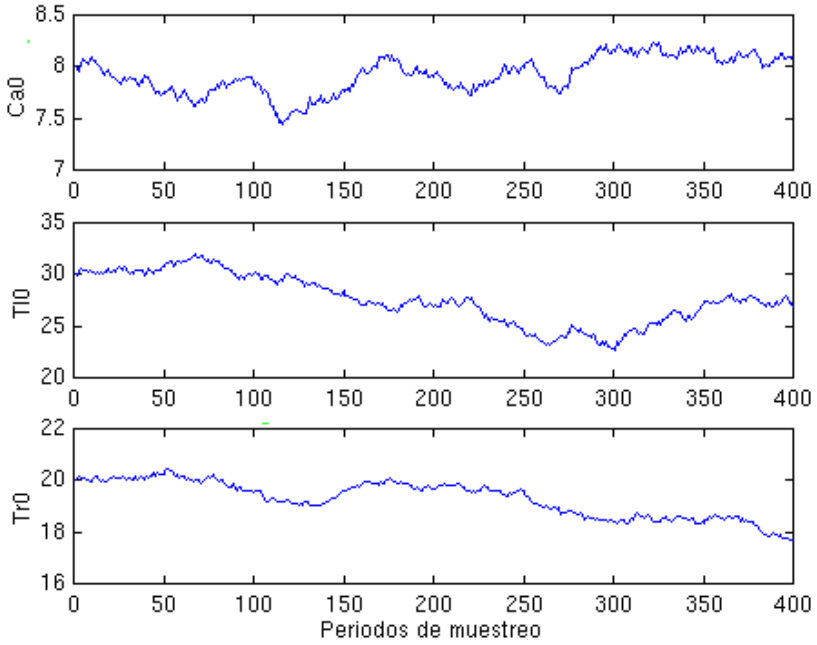


- Transforma un producto A en otro B
- Reacción química exotérmica
- Se controla la temperatura mediante una camisa por la que circula un refrigerante
- Salidas:
  - Ca Cb Tl Tr
- Entradas:
  - Manipulables: Fl Fr
  - Perturbaciones medibles: Ca0 Tl0 Tr0

Ejemplo II

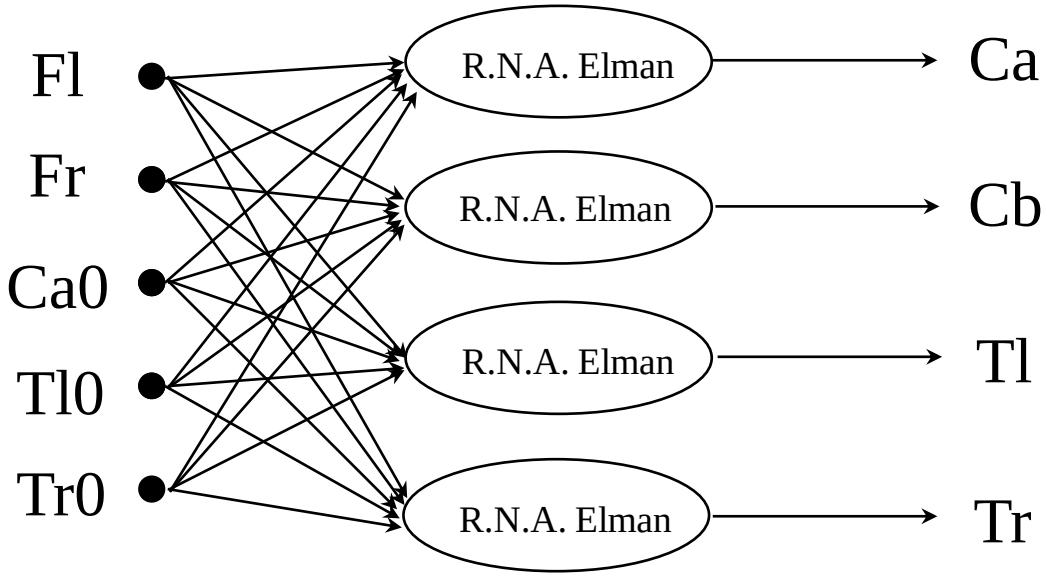
- El periodo de muestreo es $T = 0.2$ horas
- Las entradas han de ser tales que provoquen todas las salidas de interés





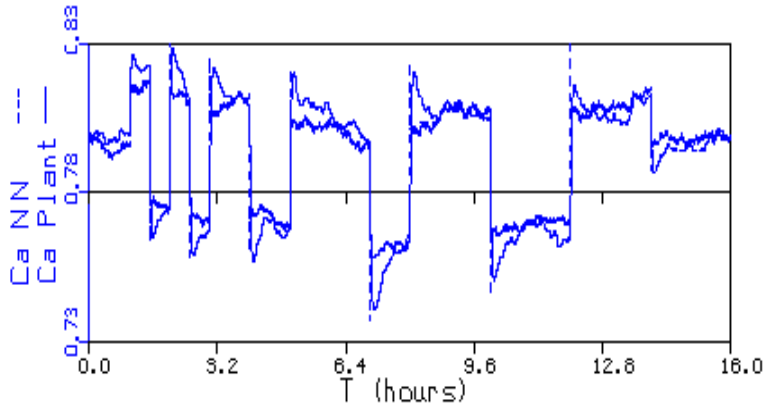
Ejemplo III

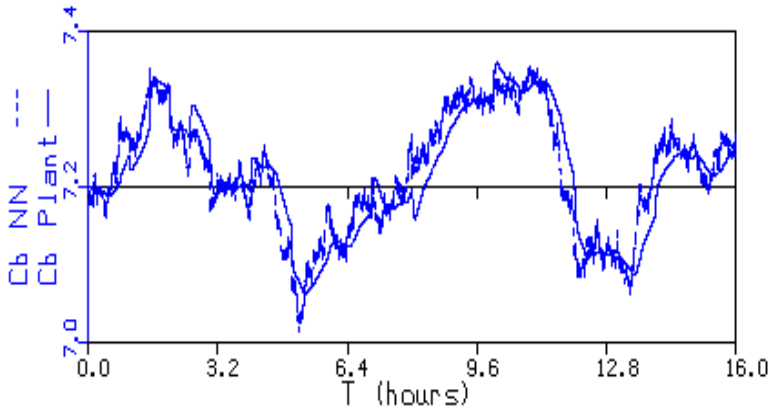
- Se normalizan los datos de entrada y salida
- Se entrenan cuatro redes, cada una modela una salida
- Se usa el algoritmo *backpropagation*

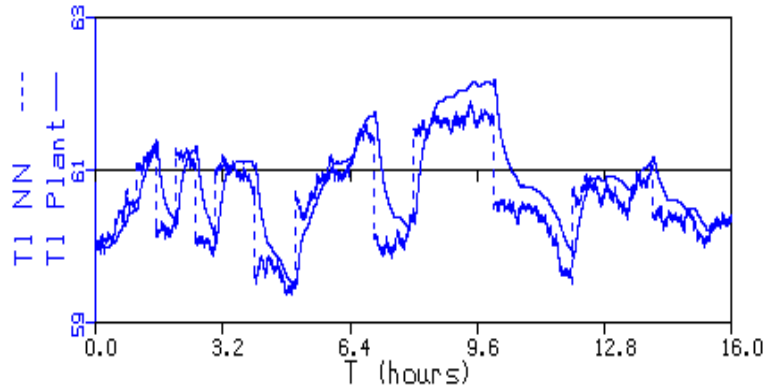


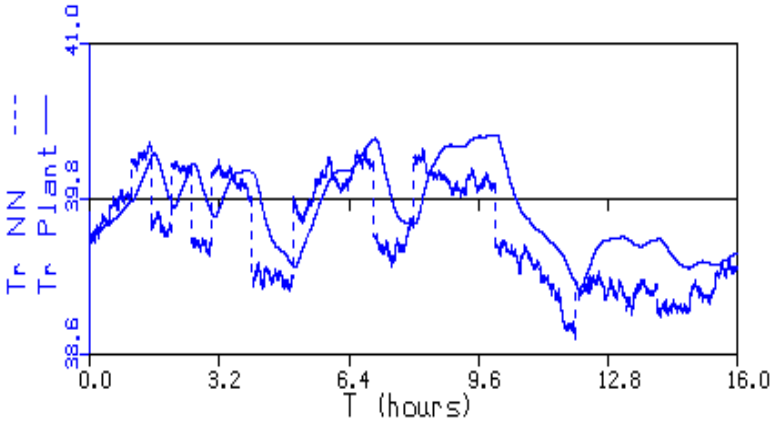
Ejemplo IV

- Validación del modelo.
  - De forma neuronal: test son saltos en Fr









Ejemplo V

- Correlaciones: blancura de los residuos