Capítulo 2

Redes neuronales para el modelado de sistemas dinámicos: los módulos neuronales

En este capítulo se consideran las redes neuronales como modelo para describir sistemas dinámicos y se introduce una clase particular, los módulos neuronales, con propiedades distintivas que resultan muy útiles en el modelado de sistemas como los descritos en la introducción de esta memoria.

Las redes neuronales gozan de características que las hacen muy interesantes en el modelado de sistemas dinámicos, como la capacidad de aproximar funciones no lineales con precisiones arbitrariamente pequeñas y la posibilidad de permitir el diseño de controladores utilizando diversas metodologías. Sin embargo, los modelos neuronales tienen inconvenientes, entre los que sobresalen los siguientes: son poco explicativos respecto a los principios físicos que rigen la dinámica del sistema identificado, por tratarse de modelos de tipo caja-negra; y la posibilidad de incorporar información previa en el modelo es muy limitada, cuando no nula.

En este estudio se han aprovechado al máximo las ventajas que las redes neuronales brindan al problema de la identificación y control de sistemas dinámicos y se han minimizado sus inconvenientes. Esto se logra con las redes neuronales modulares, compuestas por módulos que aproximan las diferentes partes que integran un sistema. De esta manera, la red resultante es, formalmente, similar a la descripción del sistema mediante diagramas de bloques. Los bloques son, aquí, los módulos neuronales.

En este capítulo se describe el diseño de módulos neuronales que aproximan funciones no lineales frecuentes en el modelado de sistemas mecánicos, eléctricos, etc. Se trata de las llamadas no-linealidades duras (*hard nonlinearities*), que constituyen un conjunto especialmente interesante en el modelado de sistemas dinámicos no lineales.

2.1 El modelo escogido: redes neuronales artificiales

Las redes neuronales artificiales (*Artificial Neural Networks*, ANN) son mecanismos de procesado de la información inspirados en las redes de neuronas biológicas. Su funcionamiento, explicado en numerosos textos (véase para una introducción [Fau-94] o [Hertz-91]), se basa en operaciones sencillas realizadas en paralelo por un gran número de células elementales, las neuronas.

Sin embargo, una red neuronal es un mecanismo que no puede ser definido de forma única. Existe un amplio repertorio de tipos de ANN que difieren en la topología (la forma en que las neuronas están conectadas), en las funciones de activación, en el algoritmo de aprendizaje, etc. El tipo de ANN que se estudiará en este trabajo es el de las redes síncronas y discretas que, a diferencia de las redes continuas o de las de Hopfield, obtienen, para una entrada dada en el instante de tiempo k, la salida correspondiente en el instante de tiempo k. Además, el esquema de conexión entre neuronas es libre, sin seguir una estructura de capas y permitiendo conexiones recurrentes. Esta última consideración permite que las redes exhiban comportamientos dinámicos. El tipo de funciones de activación comprende las funciones derivables, aunque usualmente se utilizarán las funciones lineal y sigmoidal. El mecanismo de aprendizaje no está limitado en ningún aspecto. Cualquier algoritmo, que permita el aprendizaje de estructuras como las propuestas, será potencialmente útil. Por tanto, en este trabajo y de ahora en adelante se considerarán redes neuronales del tipo descrito en este párrafo, si no se especifica lo contrario.

Las características que hacen más interesantes a las ANN para el modelado de sistemas dinámicos son:

- Son modelos no lineales, puesto que la función de activación es no lineal.
- Son dinámicos, si incluyen conexiones recurrentes, o estáticos, si no las incluyen.
- Son modelos paramétricos y los parámetros corresponden a los pesos de las conexiones entre neuronas.
- Son modelos adaptativos, puesto que la aparición de nuevos datos permite el reaprendizaje de los parámetros adaptando los valores anteriores a los datos actuales.
- Son tolerantes a fallos, ya que su comportamiento está distribuido entre todos los parámetros. Así, cuando alguno de ellos es incorrecto, el resultado global se degrada, pero no deja al modelo inoperante, necesariamente.
- Son inherentemente paralelos, cosa que permite una implementación eficiente.

En los siguientes apartados se repasan con más detalle los principales aspectos del modelado y control de sistemas dinámicos con redes neuronales.

2.1.1 Aproximación de funciones con ANN

El motivo que, probablemente, ha dado más impulso a la utilización de las redes neuronales en el campo de la identificación y control de procesos, es su demostrada capacidad de aproximar funciones con un grado arbitrario de precisión. Los resultados a este respecto parten de los trabajos de Kolmogorov (publicados en 1957), de carácter general en la disciplina de *Aproximación de Funciones*. Recientemente han sido adaptados y refinados por diversos autores (Hecht-Nielsen, Kurková, Hornik, Ito, Cybenko, etc.) al campo de la aproximación de funciones basada en redes neuronales. El resultado general más interesante (formulado independientemente en [Cyb-89] y [Hor-89]) es que las redes neuronales de tres capas con funciones de activación sigmoidales son aproximadores universales.

Estos resultados, a pesar de establecer un marco sólido en la aproximación de funciones con redes neuronales, tienen matices que los hacen poco útiles en la práctica. El primero es debido a que se basa en la norma máxima para funciones $f \in L^p(\mu)$. Esta norma no es derivable, y por ello la hace incompatible con los algoritmos de aprendizaje más populares, basados en el descenso de gradiente. El segundo es debido a que da garantías de existencia pero no aporta el método de construcción de la aproximación óptima.

Sin embargo, aportaciones más recientes ([Hush-98] y [Mel-96], por ejemplo) han abierto las puertas a algoritmos constructivos de redes neuronales que, utilizando como fundamento los teoremas precedentes de existencia de soluciones, crean redes neuronales con errores de aproximación arbitrariamente pequeños.

En este trabajo se utilizan los resultados precedentes, especialmente los de Hornik, para formular cotas del error de aproximación de los modelos que se estudiarán de sistemas no lineales. Estos resultados se presentan en el capítulo 3.

2.1.2 Identificación de sistemas dinámicos con ANN

Se considera que la identificación de sistemas dinámicos entró a formar parte de las áreas tratadas por la disciplina del control automático a principios de los años 60, cuando las técnicas estadísticas estándar, métodos como el de los mínimos cuadrados o el de la verosimilitud máxima ya eran bien conocidos. En estos casi cuarenta años, los textos sobre identificación de

sistemas dinámicos han adquirido un carácter fundamental y existen referencias muy sólidas como [Ljung-87] o [Söd-89].

Existen numerosos métodos de identificación pero, en general, todos siguen unas pautas comunes: preprocesado de los dados, selección del modelo, minimización del error de aproximación y validación del modelo.

El proceso de identificación consta de los siguientes componentes:

- *Los datos*: son pares de vectores de entrada y salida $Z^N = \{[y(t), \varphi(t)]\,;\, t = 1 \ldots N\}$ donde $y(t)$ corresponde a las salidas en el instante t, que se supone discreto, y $\varphi(t)$ a las entradas o regresores del modelo.
- *El modelo*: es una función que estima las salidas a partir de las entradas y de la información temporal que almacena, $\hat{y}(t) = \hat{g}_N(t, \varphi(t))$.
- *Los parámetros*: el modelo es una función descrita a partir de un número finito de parámetros, θ, de forma que una expresión más correcta del modelo es $g(t, \theta, \varphi(t))$.
- *La medida del error*: el error, $\varepsilon(t, \theta) = y(t) - g(t, \theta, \varphi(t))$, permite ponderar la bondad de un modelo y su medida se realiza a través de una norma, que suele ser la cuadrática:

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} \left\| y(t) - g(t, \theta, \varphi(t)) \right\| \quad \Rightarrow \quad V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N} \left| \varepsilon(t, \theta) \right|^2$$

- *El algoritmo de minimización* del error, que puede tomar formas muy diversas: directa, iterativa, heurística, analítica, estocástica, etc. En el caso de identificación de sistemas no lineales se suelen utilizar algoritmos iterativos o recursivos que, de forma genérica, se pueden expresar según

$$\theta(t) = \theta(t-1) + \mu_t R_t^{-1} \psi(t) \varepsilon(t)$$

  El escalar μ_t es el tamaño del paso, R_t^{-1} es una matriz definida positiva que permite modificar el criterio de búsqueda (cuando $R_t = \mathrm{I}$ se sigue el gradiente) y $\psi(t)$ corresponde a la derivada de g respecto a los parámetros, θ.

Las redes neuronales, a pesar de surgir a partir de motivaciones diferentes de las de la identificación de sistemas dinámicos, representan un conjunto de modelos que encaja adecuadamente en la metodología genérica descrita anteriormente. Son modelos paramétricos que, basándose en una medida del error y en un algoritmo para minimizarlo, aproximan una función no lineal con errores arbitrariamente pequeños.

Los modelos clásicos de identificación lineal (ARMAX, OE, BJ etc.) y sus versiones no lineales se pueden expresar en forma de ANN, como se apunta en [Sjö-95] y el algoritmo de aprendizaje *backpropagation* se ajusta a la regla de minimización genérica con $R_t = \mathrm{I}$. Por ejemplo, la estructura neuronal de la figura 2.1, es equivalente al modelo ARX (o NARX si las neuronas son sigmoidales en vez de lineales). En esta figura se observa que la red neuronal realiza una función estática de las observaciones que recibe de las variables de entrada y de salida tanto en el instante actual como en instantes anteriores, de forma similar a como se procede en los citados modelos.

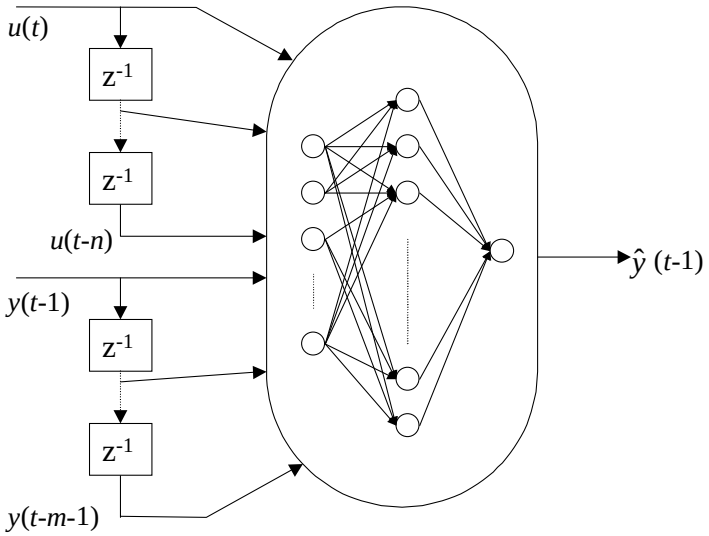


Fig. 2.1: Estructura neuronal tipo ARX o NARX

El hecho interesante es que las redes neuronales constituyen una estructura de modelado con ventajas sobre otras (como las expansiones de polinomios u otras estructuras no lineales tipo caja negra, por ejemplo). Las ventajas más obvias son que una misma estructura comprende diferentes modelos y que, computacionalmente, tienen un coste muy razonable.

Pero tienen dos ventajas más, remarcadas en [Ljung-92], menos evidentes y quizás más atractivas. La gran mayoría de los sistemas que uno se encuentra en la práctica tienen un comportamiento asintótico constante. Este comportamiento se aproxima, en otros tipos de modelos (por ejemplo, las expansiones de polinomios), a fuerza de añadir parámetros que mantengan las salidas limitadas para entradas grandes. En las ANN esto se da de forma intrínseca, debido al comportamiento de las funciones sigmoidales, y así el número de parámetros es más reducido. La otra característica que les confiere un interés especial, sobre todo en problemas mal condicionados, es la redundancia paramétrica. De forma intuitiva, esto quiere decir que el número de parámetros eficientes de la red es mucho menor que el número

real de parámetros. De esta manera, no es tan importante como en otros modelos pasar un exhaustivo proceso de validación para eliminar aquellos parámetros poco influyentes.

A pesar de todo esto, los modelos neuronales tienen inconvenientes que no se han podido resolver aún. Uno de ellos, observado en [Nar-90], es la dificultad de comprobar las condiciones de estabilidad en modelos recursivos (como las redes que identifican estructuras NOE, NBJ, y NARMAX). Este problema se ha superado parcialmente con una técnica denominada *teacher forcing*, que consiste en sustituir, en la realimentación, las salidas de la red neuronal por las salidas reales del sistema a identificar. Otros inconvenientes son la dificultad de incorporar conocimiento previo que permita mejorar el modelo o acelerar el proceso de identificación y, sobre todo, la impracticabilidad de interpretar físicamente los parámetros resultantes del modelo.

Estos problemas se tratan explícitamente en este trabajo y se propone una metodología que permite resolverlos.

2.1.3 Control de sistemas dinámicos con ANN

Así como la bibliografía en identificación de sistemas dinámicos con ANN es muy extensa y cubre aspectos teóricos y prácticos, la bibliografía en temas de control con ANN es más reducida. En este apartado se describen los dos esquemas clásicos de control con redes neuronales (el directo y el indirecto) y los resultados más recientes.

El tema del control con ANN se ha tratado desde los años 70, siendo Albus (con el *Cerebellar Model Articulation Controller*, CMAC) uno de los precursores más destacados. Los algoritmos de control que se han ido imponiendo desde entonces en el campo de las redes neuronales están muy relacionados con los que se han ido desarrollando desde la teoría del control adaptativo. Los principales enfoques se basan en control directo e indirecto [Hunt-92].

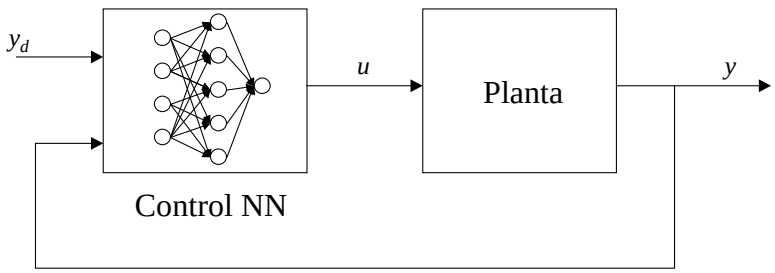


Fig. 2.2: Esquema de control directo con redes neuronales

El esquema de control directo (figura 2.2) es muy sencillo estructuralmente y más complicado en aspectos funcionales. En este caso, la señal con la que la red aprende corresponde a la diferencia entre la señal de control idónea y la señal de control calculada por la red neuronal. El problema está en que la señal de control idónea no se conoce y, por tanto, el error no se puede calcular. Una solución a este problema la dan los algoritmos de aprendizaje por refuerzo (*reinforcement learning*), que permiten el aprendizaje con señales cualitativas del error en vez de cuantitativas, aunque también se han adaptado algoritmos de aprendizaje supervisado al caso de señales cualitativas (ver [Mor-95]). Estos métodos, a pesar de gozar de gran popularidad en la comunidad de la inteligencia artificial, están poco adaptados al problema de control de procesos por falta de pruebas formales que puedan asegurar la convergencia en el aprendizaje y la estabilidad de la solución.

Los esquemas de control indirecto (figura 2.3) toman su nombre de la necesidad de un modelo neuronal de la planta para obtener el controlador. Este modelo puede servir para propagar los errores hacia atrás y permitir el aprendizaje del controlador o para generar un modelo invertido de la planta. El método de propagación de los errores hacia atrás fue introducido por Jordan y Rumelhart, [Jor-91], y se ha aplicado con éxito en numerosas ocasiones. Los esquemas de control en los que se utiliza el modelo para obtener su inversa (o una aproximación) están directamente relacionados con esquemas de control conocidos, como el control con modelo de referencia (*Model Reference Control*), el control mediante modelo interno (*Internal Model Control*) o el control predictivo.

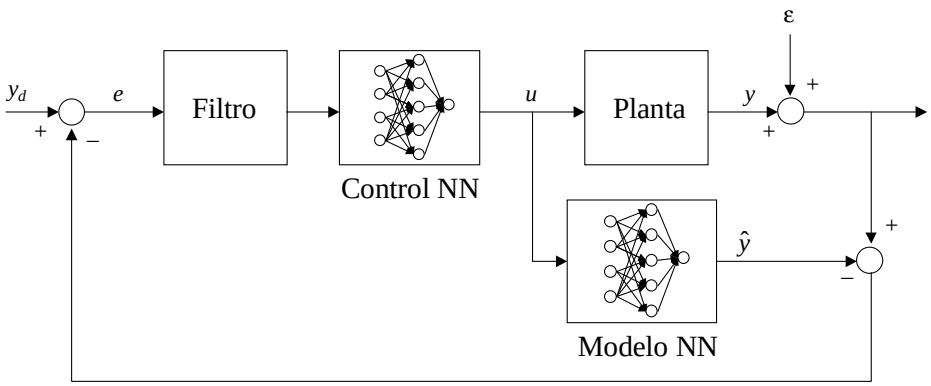


Fig. 2.3: Esquema de control indirecto con redes neuronales

Recientemente los esquemas de control indirecto han sido objeto de estudios formales que han dado, como consecuencia, resultados interesantes. Por ejemplo, en [Lev-93] se describe el diseño de controladores neuronales que aseguran la estabilización de un sistema no lineal, siempre y cuando sea de estado acotado, diferenciable (Lipshitz, en realidad) y accesible. En

[Lev-96] se extienden los anteriores resultados a sistemas en los que el estado no es accesible (y necesita ser estimado) y a problemas de seguimiento. En [Suy-96] se utiliza un modelo neuronal (llamado NL_q) que permite la síntesis de controladores óptimos y se demuestra también la estabilidad de los sistemas controlados y la robustez de los controladores obtenidos.

2.2 Módulos Neuronales

El módulo neuronal (*Neural Module,* NM) es un concepto nuevo en al ámbito de las ANN desarrollado en este trabajo. Definimos un módulo neuronal como una red neuronal que aprovecha el uso de restricciones estructurales para forzar un tipo de comportamiento en el modelo.

2.2.1 Motivación

La mayoría de las topologías populares de ANN (redes de tres capas, redes recurrentes totalmente conectadas, etc.) son modelos muy flexibles que pueden adaptarse a un rango amplio de características de entrada-salida. Esta es su principal ventaja, pero también puede representar su principal inconveniente por las siguientes razones:

- cuando una ANN no converge en el aprendizaje, no es obvio discernir si la causa es que el error mínimo es difícil de alcanzar o que la configuración de la red no es la correcta, y
- se pueden observar comportamientos muy similares en estructuras y topologías neuronales radicalmente diferentes, complicando enormemente la búsqueda de modelos con menor número de parámetros

Solla demuestra, en [Solla-89], que la probabilidad de que una ANN muestre un determinado comportamiento depende no sólo del algoritmo de aprendizaje de la red, sino también de la arquitectura. El procedimiento para incrementar la probabilidad de la correspondencia correcta entre entradas y salidas suele recaer en el aprendizaje pero, tal y como se argumenta en [Happ-94], un método potencialmente mucho más versátil para cambiar la distribución de probabilidades de comportamientos en una red neuronal es imponer restricciones sobre su topología.

Precisamente, este último argumento es el que da pie a la creación de los módulos neuronales.

2.2.2 Estructura del modelo que se utiliza en este trabajo

Los modelos que se estudian en este trabajo son redes neuronales modulares. Están compuestas por redes más sencillas que se comportan como bloques funcionales y éstos son los módulos neuronales, [Mor-96b].

Una red de módulos neuronales es una red neuronal clásica en el sentido de que está compuesta por neuronas de activación sigmoidal o lineal, es discreta y se entrena con algoritmos típicos de aprendizaje (*backpropagation* para redes estáticas y *backpropagation through time*, BPTT, para redes dinámicas). Lo que la diferencia de otros modelos neuronales es que, en cada módulo, el patrón de conectividad no está limitado a capas de neuronas sino que es totalmente libre. La segunda diferencia importante es que los parámetros de los NM pueden tomar valores constantes, que no varían a lo largo del aprendizaje. De esta manera es posible forzar un comportamiento específico en cada bloque funcional y, por consiguiente, en el modelo.

Una vez aplicado con éxito el proceso de identificación, el modelo resultante es una red neuronal compuesta por módulos, cada uno de los cuales representa un bloque funcional del sistema con un significado fácilmente interpretable.

El propósito de este modelo es aprovechar al máximo las ventajas que las redes neuronales brindan al problema de identificación y control de sistemas dinámicos, minimizando a su vez sus inconvenientes.

2.2.3 Definición de módulo neuronal

Dada una familia de funciones, *F*, que supondremos definida sobre un compacto de los reales:

$$F = \left\{ f_\theta : [a,b] \to [a,b] \;\; | \, \theta \in \Theta \subset \mathbb{R}^{\mathrm{N}} \right\} \tag{2.1}$$

definimos un módulo neuronal como una red neuronal que aproxima adecuadamente el comportamiento de *F*:

$$F_{NM} = \left\{ f_{\theta,\omega} : [a,b] \to [a,b] \;\; | \, \theta \in \Theta \subset \mathbb{R}^{\mathrm{N}},\ \omega \in \mathbb{R}^{\mathrm{M}} \right\} \quad \text{t.q.} \quad d(F, F_{NM}) < \varepsilon(\omega) \tag{2.2}$$

El conjunto [a,b] corresponde al espacio de salida y llegada de una instancia cualquiera, f_θ, de la familia de funciones *F*. Se puede observar que las instancias de F_{NM} dependen de más parámetros que las de *F*. Este exceso de parámetros corresponde a los propios de la red neuronal

con que se realiza el NM, los pesos. La manera en que el NM aproxima la familia de funciones viene determinada por la distancia $d(F,F_{NM})$, que se discutirá en las siguientes secciones.

Las condiciones que hacen posible que una ANN como las indicadas pueda aproximar una familia dada de funciones, ajustándose a unos determinados criterios de distancia son dos:

- la estructura topológica de la red, las conexiones entre neuronas, está fijada de antemano
- parte del vector de parámetros de la red, ω, no es modificada por el algoritmo de aprendizaje

Así pues, un NM es una red neuronal que, debido a una serie de restricciones impuestas sobre su estructura y parámetros, se comporta inherentemente como una familia de funciones. El procedimiento para sintonizar una función concreta dentro de la familia F es el mecanismo de aprendizaje, que sólo afecta a un subconjunto de los pesos de la red, mientras que el resto no varía. Por ejemplo, se verá posteriormente que hay un NM que realiza la función saturación y que tiene dos pesos que corresponden, respectivamente, al valor de la salida saturada y a la pendiente del intervalo no saturado, mientras que el resto de los pesos (no modificables durante el aprendizaje) tienen la misión de forzar un comportamiento 'de tipo saturación'.

Lo que resta de este capítulo está dedicado a describir el proceso de diseño de los módulos neuronales y en el capítulo 3 se justifica, mediante desarrollos formales, su adecuación en el modelado de sistemas dinámicos no lineales.

2.3 Diseño de módulos neuronales

El diseño de un NM puede ser una tarea compleja, dependiendo de la familia de funciones que se desee aproximar. Por ejemplo, se verá (apartado 2.4.5) que es relativamente sencillo realizar la función valor absoluto con un módulo neuronal de cinco neuronas. Sin embargo, cuesta imaginar cómo sería el NM correspondiente a un comportamiento complicado, como el de un sistema caótico.

El conjunto de NMs que se describe en la próxima sección está basado en las no-linealidades típicas de los sistemas electromecánicos, las no-linealidades duras. Los textos clásicos de análisis y control de sistemas dinámicos, como ya se ha indicado en la introducción de esta memoria, suelen distinguir entre las no-linealidades analíticas y las no-linealidades duras. El primer grupo, por pertenecer generalmente al grupo de las funciones continuas, derivables, de integral acotada, etc., puede tratarse con algunas técnicas ya existentes como, por ejemplo, la linealización mediante expansiones de la serie de Taylor alrededor de un punto de operación. El

segundo grupo, por otro lado, suele tratarse poco aunque se dan excepciones, como [Tao-96], íntegramente dedicado al control de sistemas con no-linealidades duras. El conjunto de no-linealidades que suele considerarse de interés está formado por: el relé ideal (característica todo-nada), la zona muerta, la saturación, el relé con histéresis y el juego de engranajes.

En este trabajo se obtendrán los NMs que aproximan las no-linealidades duras mencionadas y también otras que pueden ser de utilidad como: el valor absoluto, la fricción viscosa y de Coulomb, el limitador de velocidad y el tope mecánico.

Previamente, no obstante, es necesario explicar con detalle cómo, con redes neuronales, es posible aproximar dos tipos de comportamientos que servirán después para el diseño de las demás funciones no lineales. Estos dos comportamientos surgen a partir de analizar las funciones lineales a trozos. Una función lineal a trozos se define así:

$$f(x)=\begin{cases} f_1(x) & \alpha_0 < x \le \alpha_1 \\ \dots & \\ f_n(x) & \alpha_{n-1} < x \le \alpha_n \end{cases} \qquad f_i(x)=a_i x+b_i\ , \qquad a_i, b_i \in \mathbb{R} \qquad (2.3)$$

Nótese que para poder reproducir el comportamiento de una función lineal a trozos es necesario, en primer lugar, poder determinar si x pertenece a un intervalo (α_{i-1}, α_i). En segundo lugar, hay que poder calcular $f_i(x)$ de forma que valga 0 (u otro valor constante) para $x \notin (\alpha_{i-1}, \alpha_i)$.

2.3.1 Pertenencia a intervalos

La activación clásica de las neuronas, activación sigmoidal, ofrece una forma natural de detectar la pertenencia a intervalos puesto que su salida es esencialmente diferente para entradas mayores o menores que cero. El esquema básico de detección es:

$$umbral_\alpha(x) = sigs(w_d(x-\alpha)) \qquad (2.4)$$

donde α es el valor a partir del cual la función $umbral_\alpha$ cambia de signo y el peso w_d determina la pendiente del cambio (recordemos que *sigs* es una función continua y que justamente la utilizamos como función de activación, preferentemente a funciones de activación definidas a trozos, por ser continua).

Un factor de estudio interesante es la pendiente de la función $umbral_\alpha$ cuando cambia de signo. Se define franja de incertidumbre (notada a partir de ahora con γ) como la distancia entre

valores de entrada correspondientes a las salidas -1+*p* y 1-*p*. La franja de incertidumbre de $umbral_\alpha$ se puede calcular fácilmente a partir de la definición de *sigs*:

$$\gamma(w_d, p) = 2 \cdot \frac{k}{-w_d} \qquad \text{y} \qquad k = \ln\left(\frac{p}{2-p}\right) \tag{2.5}$$

Se observa en la ecuación (2.5) que γ es inversamente proporcional al peso w_d. Es decir, que a mayor valor del peso menor es la franja de incertidumbre y mejor se realizará la detección de pertenencia al intervalo. Si se quiere diseñar una neurona que realice la función $umbral_\alpha$ solamente es necesario fijar los valores de *p* y γ para despejar el correspondiente peso, w_d, que cumple las condiciones especificadas. Por ejemplo, si queremos una franja de incertidumbre $\gamma = 0.1$ y consideramos que un valor de salida de ± 0.75 ($p = 0.25$) es suficiente para indicar cuándo las entradas son mayores o menores que α, entonces el peso resultante vale

$$w_d = -2 \cdot \frac{\ln\left(\frac{p}{2-p}\right)}{\gamma} = -2 \cdot \frac{\ln\left(\frac{0.25}{1.75}\right)}{0.1} = 38.92$$

Es posible definir otras funciones $umbral_\alpha$ con neuronas no simétricas (utilizando la función de activación *sig*) o con el peso w_d negativo, de forma que para $x < \alpha$ la salida sea positiva y para $x > \alpha$ sea negativa. De este modo habrá cuatro procedimientos para realizar la detección (con sus franjas de incertidumbre asociadas):

$$\left.\begin{array}{l} umbral1_\alpha(x) = sigs(w_d(x-\alpha)), \quad \text{con } w_d > 0 \\ umbral2_\alpha(x) = sigs(w_d(x-\alpha)), \quad \text{con } w_d < 0 \end{array}\right\} \qquad \gamma(w_d, p) = 2 \cdot \frac{k}{-w_d},\ k = \ln\left(\frac{p}{2-p}\right)$$

$$\left.\begin{array}{l} umbral3_\alpha(x) = sig(w_d(x-\alpha)), \quad \text{con } w_d > 0 \\ umbral4_\alpha(x) = sig(w_d(x-\alpha)), \quad \text{con } w_d < 0 \end{array}\right\} \qquad \gamma(w_d, p) = 2 \cdot \frac{k}{-w_d},\ k = \ln\left(\frac{p}{1-p}\right)$$

cada uno de los cuales tiene un comportamiento diferente. En la figura 2.4 están detalladas las redes neuronales que realizan las cuatro funciones $umbral_\alpha$ y una gráfica asintótica (con w_d tendiendo a ∞) de las mismas.

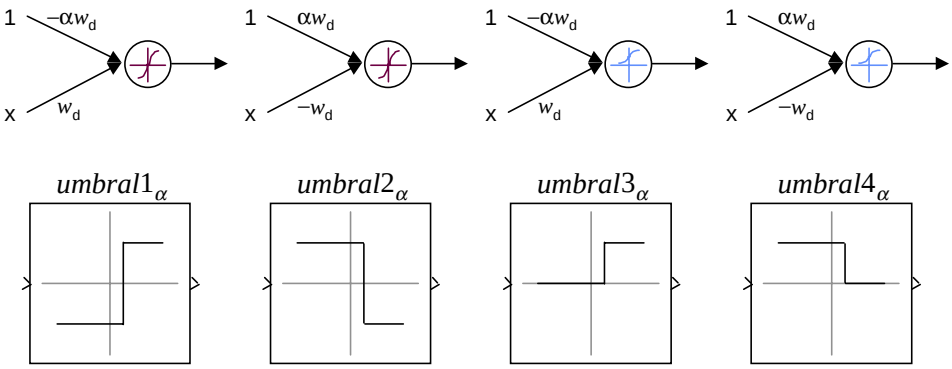


Fig 2.4: redes neuronales y gráficas de las cuatro funciones $umbral_\alpha$

Combinando adecuadamente las funciones $umbral_\alpha$ es posible detectar la pertenencia a intervalos arbitrarios de diferentes maneras. Por ejemplo: si se desea realizar la siguiente detección:

$$x \in (\alpha_1, \alpha_2) = \begin{cases} 1 & \alpha_1 \le x \le \alpha_2 \\ 0 & x < \alpha_1, x > \alpha_2 \end{cases} \qquad (2.7)$$

basta con diseñar una red de dos módulos $umbral_\alpha$ cuya salida sea la suma de ambos. Un módulo detecta si $x > \alpha_1$ y el otro detecta si $x < \alpha_2$. La suma de ambos, debidamente ponderada, aproxima la función que se expresa en la ecuación (2.7). La precisión de la detección la podemos calcular previamente y ajustar al grado que más nos interese. La red neuronal resultante se puede expresar mediante

$$x \in (\alpha_1, \alpha_2) = lin(0.5 \cdot umbral3_{\alpha 1}(x) + 0.5 \cdot umbral4_{\alpha 2}(x))$$

2.3.2 Función de pendiente constante en un rango y nula fuera de él

Recordemos que el objetivo de esta sección es permitir el diseño de módulos neuronales que aproximen el tipo de funciones no lineales ya indicado. Para ello nos basamos en la ecuación (2.3) y el propósito de este apartado es describir la obtención de $f_i(x)$ lineal con $x = 0 \ \forall \ x \notin (\alpha_{i-1}, \alpha_i)$.

La comprobación de que *sigs* es aproximadamente lineal en un pequeño entorno de $x = 0$ es trivial. Este comportamiento se degrada (alejándose de la recta) a medida que x se aleja de cero. Por ello, si se desea obtener una respuesta aproximadamente lineal para entradas pertenecientes a un intervalo genérico (α_1, α_2) y utilizando la función *sigs*(x), hay que escalarlas de forma que

se transformen en un pequeño intervalo simétrico entorno a cero, (-δ, δ). El cálculo para obtener la entrada escalada, dados α_1, α_2 y ε (el máximo error permitido en la aproximación) es:

$$x_e = \frac{\delta \cdot \left(2x - \alpha_2 - \alpha_1\right)}{\alpha_2 - \alpha_1} \quad \text{con } \varepsilon = \frac{1 - e^{-\delta}}{1 + e^{-\delta}} - \frac{\delta}{2} \tag{2.8}$$

donde x_e es el valor de x escalado y δ no se puede calcular analíticamente pero se puede obtener a partir de tablas o con ayuda de una gráfica de la función $\varepsilon(\delta)$, como la de la figura 2.5.

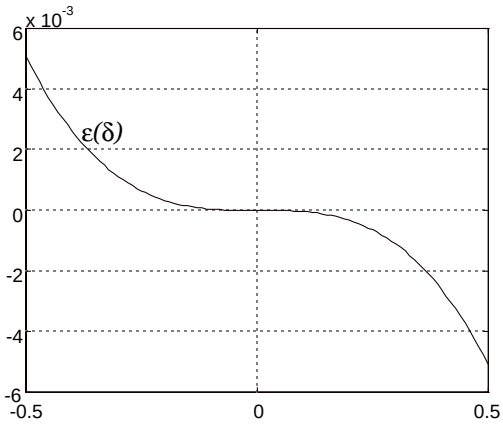


Fig 2.5: gráfica de $\varepsilon(\delta)$ para δ entre -0.5 y 0.5

Conocidos, por tanto, los extremos del intervalo al que pertenece la entrada y el error máximo permitido, la red neuronal que lleva a cabo la aproximación lineal es:

$$lineal(x) = \beta \cdot w_r \cdot sigs\left(\frac{x}{w_r} - bias\right) \quad \text{donde } w_r = \frac{\alpha_2 - \alpha_1}{2 \cdot \delta} \quad \text{y } bias = \delta \frac{\alpha_2 + \alpha_1}{\alpha_2 - \alpha_1} \tag{2.9}$$

El parámetro β controla la pendiente de la recta que genera *lineal*(x) y se puede comprobar que si el intervalo está centrado en cero ($\alpha_2 = -\alpha_1$) el parámetro *bias* se anula. Para lograr la pendiente deseada, a_i, es necesario que $\beta = 2 \cdot a_i$ dado que *sigs* no tiene pendiente unitaria en el origen.

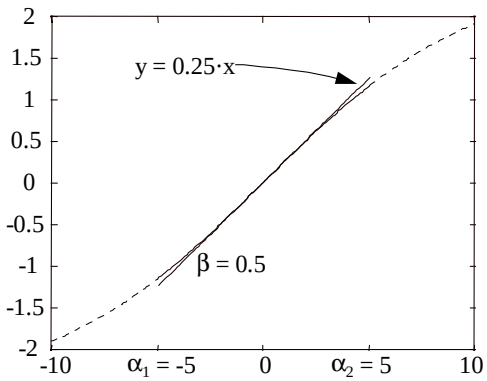


Fig. 2.6: gráfica de lineal(x) para $\alpha_1 = -\alpha_2 = -5$ y $\beta = 0.5$

A modo de ejemplo, si se sabe que la entrada pertenece al rango (-10, 10) y no se quiere cometer un error mayor de 0.001 en la aproximación lineal, la ecuación (2.9) proporciona un valor del peso $w_r \geq 33$. En la gráfica de la figura 2.6 se pueden comparar las representaciones de $y = 0.25 \cdot x$ y su aproximación mediante la función *lineal*(*x*), teniendo en cuenta que se ha utilizado un valor del peso w_r pequeño para hacer así más evidentes las diferencias entre ambas funciones.

	$(-\infty, \alpha_1)$	(α_1, α_2)	(α_2, ∞)
$x1 = umbral4_{\alpha 1}(x)$	1	0	0
$x2 = umbral3_{\alpha 2}(x)$	0	0	1

Tabla 2.1: activaciones ideales para dos neuronas que detectan α_1 y α_2

Llegados a este punto, nos interesa conseguir que el comportamiento lineal se dé únicamente para valores pertenecientes al intervalo (α_1, α_2). Una opción posible para lograrlo consiste en utilizar módulos de tipo $umbral_\alpha$ para forzar que la salida de la red esté completamente saturada con entradas $x \notin (\alpha_1, \alpha_2)$. Por ejemplo, si se utilizan dos neuronas cuyas salidas *x1* y *x2* cumplen las especificaciones ideales de la tabla 2.1 y éstas se agregan a la entrada de *lineal*(*x*) se obtiene una función lineal modificada tal como

$$linealMod(x) = \beta \cdot w_r \cdot sigs(\frac{x}{w_r} - bias - w_s x1 + w_s x2) \qquad (2.10)$$

El valor del peso w_s no es crítico; únicamente interesa que sea suficientemente grande como para saturar la salida de *sigs* cuando *x1* o *x2* sea 1. Se ha comprobado empíricamente que, para ello, basta con que w_s se encuentre entre 10 y 20. La gráfica de la figura 2.7 muestra la

representación de *linealMod*(*x*) y ya se observa claramente que para valores de *x* fuera del intervalo de interés la salida tiene un comportamiento cualitativamente diferente.

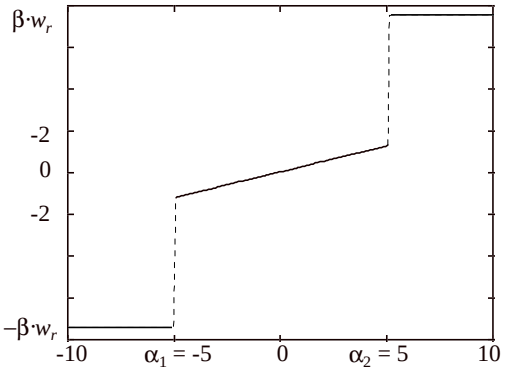


Fig. 2.7: gráfica linealMod(*x*) *para* $\alpha_1 = -\alpha_2 = -5$ *y* $\beta = 0.5$

La función *linealMod*(*x*) (recordemos que no es más que la salida de una red neuronal compuesta por tres neuronas) vale $\beta \cdot w_r$ para $x > \alpha_2$ y $-\beta \cdot w_r$ para $x < \alpha_1$, pero esto se puede corregir utilizando *x1* y *x2* como entradas de unas funciones de compensación, que se definen según:

$$compens1(x) = \beta \cdot w_r \cdot sigs\left(w_s x1 + \frac{d1}{w_r} \right)$$

$$compens2(x) = \beta \cdot w_r \cdot sigs\left(-w_s x2 - \frac{d2}{w_r} \right) \qquad (2.11)$$

donde w_r y w_s son los mismos valores empleados en las ecuaciones (2.9) y (2.10) y *d1* y *d2* son unos parámetros que se utilizarán más adelante; por el momento interesa $d1 = d2 = 0$. La representación conjunta de *compens1*(*x*) y *compens2*(*x*) se da en la gráfica de la figura 2.8.

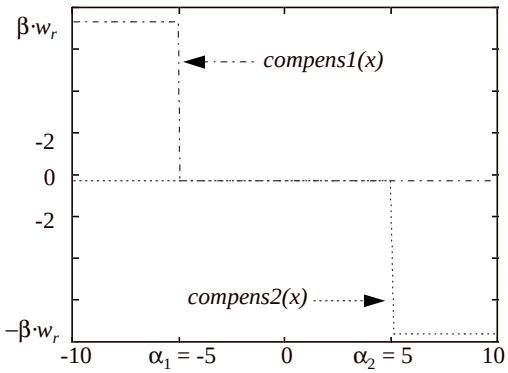


Fig. 2.8: gráfica conjunta de compens1(*x*) *compens2*(*x*)

Recordemos de nuevo que la función que queremos aproximar es $f_i(x) = a_i \cdot x + b_i$ y por el momento estamos en disposición de aproximar $a_i \cdot x - a_i(\alpha_2 + \alpha_1)/2$ mediante:

$$\tilde{f}_i(x) = linealMod(x) + compens1(x) + compens2(x) \qquad (2.12)$$

En la gráfica de la figura 2.9 está la representación correspondiente a $\tilde{f}_i(x)$. La forma de incorporar el parámetro b_i en esta aproximación es utilizando *d1*, *d2* y $\bar{b}_i$, un nuevo parámetro que se suma a $\tilde{f}_i(x)$. Al sumar $\bar{b}_i$ a $\tilde{f}_i(x)$, ésta se desplaza verticalmente al punto que uno desee y los parámetros *d1* y *d2* permiten que $\tilde{f}_i(x) = 0$ para $x \notin (\alpha_1, \alpha_2)$. Sin embargo, existe un problema al hacerlo que se verá seguidamente.

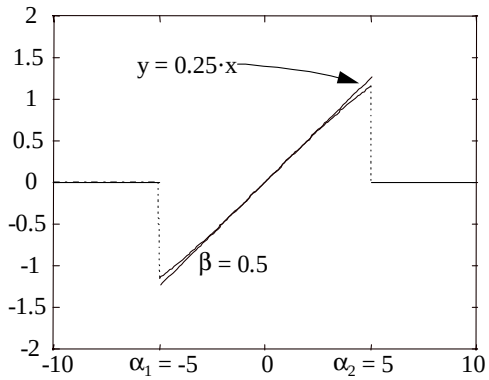


Fig. 2.9: gráfica de $\tilde{f}_i(x)$

$$\tilde{f}_i(x) \approx \begin{cases} -\dfrac{\beta \cdot d2}{2} + \bar{b}_i = -a_i \cdot d2 + \bar{b}_i & x < \alpha_1 \\ a_i \cdot x - a_i \cdot \dfrac{\alpha_1 + \alpha_2}{2} + a_i (d1 - d2) + \bar{b}_i & \alpha_1 \le x \le \alpha_2 \\ \dfrac{\beta \cdot d1}{2} + \bar{b}_i = a_i \cdot d1 + \bar{b}_i & x > \alpha_2 \end{cases} \qquad (2.13)$$

Al igualar (2.13) a sus valores deseados se obtiene el siguiente sistema compatible y determinado de ecuaciones lineales:

$$\begin{bmatrix} 0 & -a_i & 1 \\ a_i & -a_i & 1 \\ a_i & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} d1 \\ d2 \\ \bar{b}_i \end{bmatrix} = \begin{bmatrix} 0 \\ b_i + a_i \dfrac{\alpha_1 + \alpha_2}{2} \\ 0 \end{bmatrix} \qquad (2.14)$$

cuyas soluciones son: $d1 = \dfrac{b_i}{a_i} + \dfrac{\alpha_1 + \alpha_2}{2}$, $d2 = -\dfrac{b_i}{a_i} - \dfrac{\alpha_1 + \alpha_2}{2}$ y $\bar{b}_i = -b_i - a_i \dfrac{\alpha_1 + \alpha_2}{2}$. El problema que plantean los valores de estos parámetros es que, puesto que b_i y a_i no tienen restricciones de rango, tanto *d1* como *d2* pueden estar fuera del intervalo (α_1, α_2) y, de esta forma, las aproximaciones que realizan las funciones *compens1*(x) y *compens2*(x) no seguirían las especificaciones de precisión dadas en los prerrequisitos a través de ε y δ.

Este problema no es del todo insalvable porque se pueden rehacer las especificaciones del error máximo teniendo en cuenta también los parámetros a_i y b_i. Sin embargo, dada la naturaleza del tipo de funciones que en la práctica se aproximan, parece más razonable imponer alguna restricción sobre $f(x)$, definida en la ecuación (2.3), para poder continuar con los supuestos ya enunciados.

La función $f(x)$ es muy genérica pues permite la realización de cualquier función lineal a trozos. En la práctica, no obstante, muchas de las características no lineales observadas (la saturación, la zona muerta, algunos modelos de la fricción y otras) son funciones continuas de la entrada. Si, por tanto, restringimos $f(x)$ a las funciones continuas lineales a trozos seguiremos abarcando un conjunto interesante y, ahora sí, con la posibilidad de diseñar módulos neuronales que las aproximen cumpliendo unas especificaciones previas de error máximo.

La nueva definición de $f(x)$, incluyendo la restricción de continuidad queda:

$$f(x)=\begin{cases} f_1(x) & \alpha_0 < x \le \alpha_1 \\ \dots & \\ f_n(x) & \alpha_{n-1} < x \le \alpha_n \end{cases} \qquad f_i(x) = a_i x + b_i \ , \qquad \left| \begin{array}{l} a_i \in \ \mathbb{R} \\ b_i = b_n + \sum_{j=i}^{n-1} \alpha_j (a_{j+1} - a_j) \end{array} \right. \tag{2.15}$$

que se diferencia de (2.3) en que ahora solamente hay un parámetro b libre (aquí es b_n, pero podría ser cualquier otro) y los demás quedan en función de él.

$f(x)$ representa una familia de funciones que viene caracterizada por $2 \cdot n+2$ parámetros para n tramos lineales. Estos parámetros son: a_i (n valores), α_i (n+1 valores) y b_n (1 valor). Si se define

$$\tilde{f}(x) = \bar{b} + \sum_{i=1}^{n} \tilde{f}_i(x) \tag{2.16}$$

que es la aproximación de $f(x)$, se pueden relacionar los parámetros de una y otra familia de funciones según:

$$\begin{array}{ll} w_s \text{ y } w_d \quad \text{libres} & \beta_i = 2 \cdot a_i \\ w_{ri} = \dfrac{\alpha_i - \alpha_{i-1}}{2 \cdot \delta} & bias_i = \delta \dfrac{\alpha_i + \alpha_{i-1}}{\alpha_i - \alpha_{i-1}} = \dfrac{\alpha_i + \alpha_{i-1}}{2 \cdot w_r} \\ d1_i = d2_i = \dfrac{\alpha_i - \alpha_{i-1}}{2} & \bar{b} = a_n \cdot \alpha_n + b_n - \sum_{i=1}^{n} a_i \dfrac{\alpha_i - \alpha_{i-1}}{2} \end{array} \tag{2.17}$$

Por ahora, los parámetros w_d y w_s pueden definirse independientemente de $f(x)$ siempre y cuando se cumplan las restricciones impuestas sobre ellos anteriormente. El error máximo permitido en los tramos lineales limita el valor de δ y fija el valor de w_{ri}. El resto de los parámetros se puede calcular a partir de a_i, α_i y b_n.

Veamos con un ejemplo cómo es la red neuronal $\tilde{f}_i(x)$ que aproxima un tramo lineal cualquiera $f_i(x) = a_i \cdot x + b_i \ \forall \ x \in (\alpha_{i-1}, \alpha_i)$. Previamente, para dejar bien clara la utilización en forma de red neuronal de las funciones *compens1*(x), *linealMod*(x), etc., es conveniente comprender el diagrama de la figura 2.10, que esquematiza la interconexión entre las diferentes funciones. El diagrama correspondiente a la ANN que aproxima $f_i(x)$, con los detalles de los pesos, se puede observar en la figura 2.11.

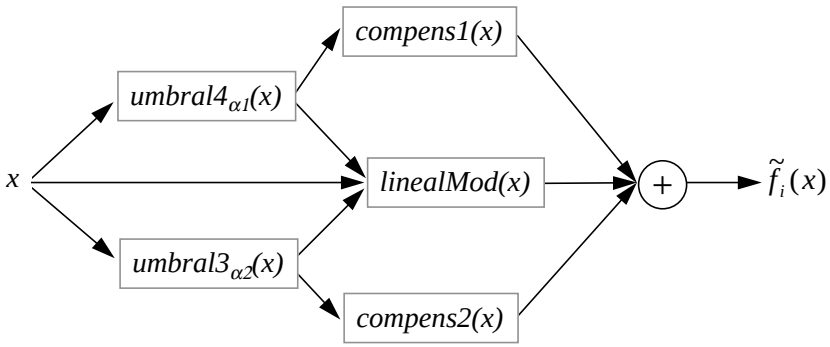


Fig. 2.10: esquema de la red neuronal que aproxima $f_i(x)$

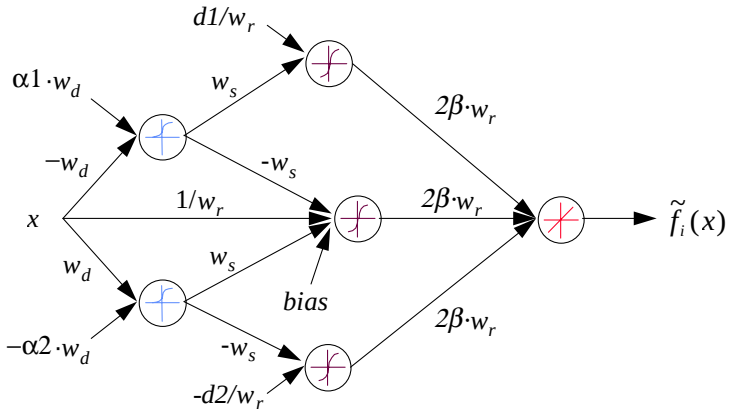


Fig. 2.11: esquema de la red neuronal que aproxima $f_i(x)$

En este caso, partimos de unas especificaciones de error máximo a partir de las que es sencillo obtener el valor de δ mediante una gráfica similar a la de la figura 2.5. Llegados a este punto, basta con aplicar las igualdades (2.17) y obtenemos los parámetros w_r, β_i, $bias_i$, $d1_i$ y $d2_i$ que definen la red neuronal. La respuesta, $\tilde{f}_i(x)$, de la red construida a partir de estos parámetros se puede observar en la figura 2.12.

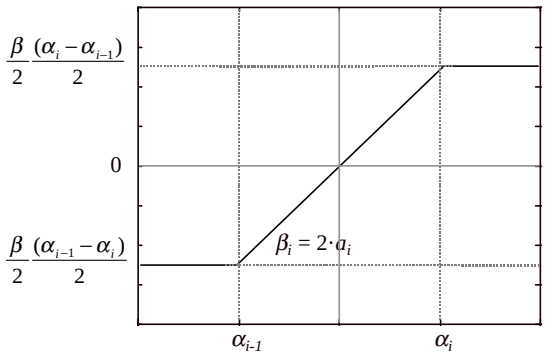


Fig. 2.12: gráfica de la respuesta $\tilde{f}_i(x)$

2.3.3 Aproximación de funciones lineales a tramos

A la vista de cómo se diseña la red neuronal que aproxima un tramo lineal en un intervalo veamos cómo se diseñaría, finalmente, el módulo neuronal capaz de aproximar $f(x)$ según la definición (2.15). Esta aproximación se basa en (2.16) y en las especificaciones de error máximo.

Volviendo a la definición de módulo neuronal (2.2) los conjuntos de parámetros θ y ω para la familia de funciones lineales a tramos (2.15) corresponden a:

$$\theta = \{a_i, \alpha_j, b_n \quad | \quad i = 1\ldots n,\ j = 0\ldots n\} \tag{2.18}$$

$$\omega = \{w_{si}, w_{di}, w_{ri}, \beta_i, bias_i, d1_i, d2_i \quad | \quad i = 1\ldots n\} \tag{2.19}$$

Ya se ha apuntado anteriormente que los pesos w_{si} pueden fijarse a valores entre 10 y 20 directamente. Los pesos w_{di} son determinantes en la precisión de la aproximación y es necesario fijarlos a valores grandes (del orden de 100). Los pesos w_{ri} dependen del mínimo error permitido y de las longitudes de los tramos lineales. No es posible calcularlos a priori porque las longitudes varían entre instancias de la familia de funciones. Sin embargo, dado el error máximo y el rango total de la entrada siempre es posible calcular, de forma conservadora, un peso w_r común a partir de (2.17). El resto de los parámetros (β_i, $bias_i$, $d1_i$ y $d2_i$) se pueden obtener a partir de los únicos pesos modificables del NM, que corresponden a a_i, α_j, y b_n.

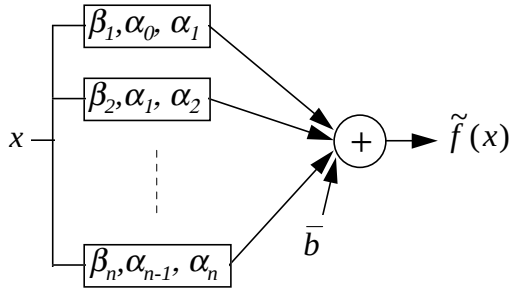


Fig. 2.13: esquema del módulo neuronal que aproxima f(x)

En la figura 2.13 se presenta un diagrama del módulo neuronal que aproxima $f(x)$ resaltando los únicos parámetros que se modificarán a través del algoritmo de aprendizaje de la red neuronal. Para comprender mejor cómo es posible obtener los restantes parámetros de la aproximación (β_i, $bias_i$, $d1_i$ y $d2_i$) mediante conexiones entre neuronas, la figura 2.14 detalla la arquitectura neuronal de un tramo lineal y destaca estas conexiones.

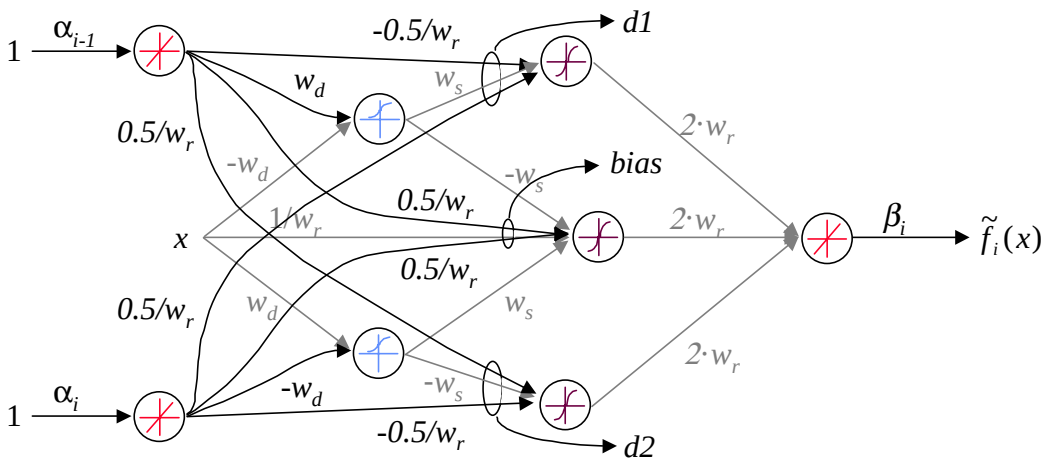


Fig. 2.14: detalle de la realización de un tramo lineal con las conexiones de los únicos parámetros que modifica el algoritmo de aprendizaje (α_{i-1}, α_i, β_i) destacados

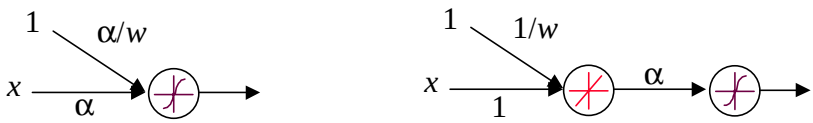
2.3.4 Proceso de diseño de un módulo neuronal

Dada una familia de funciones, lineales o no, parametrizada a través de un conjunto, θ, como en la ecuación (2.1), el proceso de diseño de un módulo neuronal que aproxime dicha familia tiene la finalidad de obtener una red neuronal cuyos pesos pueden agruparse en pesos fijos y modificables. Los pesos fijos fuerzan el comportamiento deseado en el modelo, mientras que los modificables deberían tener el mismo significado que los parámetros del conjunto θ.

Los comportamientos descritos en los apartados anteriores, la pertenencia a un intervalo y las funciones lineales a tramos, son comportamientos que, combinados adecuadamente, permiten generar una gran cantidad de funciones no lineales. La clave para poder crear módulos neuronales de diferentes familias de funciones está en poder representarlas como combinaciones de los anteriores comportamientos.

Existe, por otro lado, una serie de pautas, obtenida a través de la experiencia, que puede resultar útil a la hora de diseñar el módulo neuronal correspondiente a una función concreta. Estas pautas son las siguientes:

- Las funciones $umbral_\alpha$ permiten clasificar las entradas según el intervalo al que éstas pertenecen. Esto suele ser útil cuando la función está definida a tramos.
- Cualquier función estática y lineal a tramos se puede generar a partir de una combinación adecuada de funciones definidas como en la ecuación (2.12), aunque siempre será más conveniente, como ya se ha razonado, aproximar funciones lineales a tramos y continuas.
- Existen transformaciones topológicas en las redes neuronales que, sin modificar su comportamiento, permiten reducir el número de parámetros variables y aumentar el número de parámetros fijos, cosa que, generalmente, será deseable.

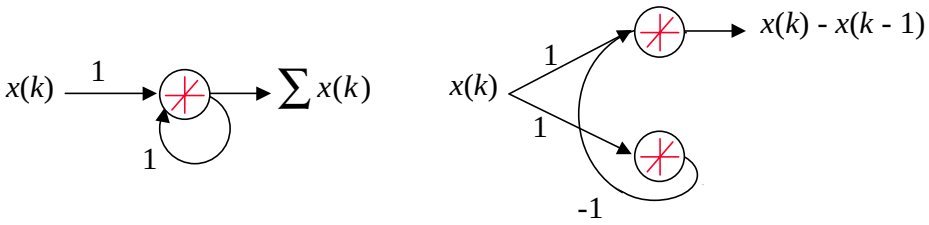


a) red original b) red transformada

Fig. 2.15: ejemplo de transformación topológica

Por ejemplo, la red de la figura 2.15a tiene dos pesos y, suponiendo que α es uno de los parámetros de la función que se quiere aproximar, ambos son modificables. La red de la figura 2.15b realiza una función equivalente, tiene una neurona más pero ahora sólo tiene un peso modificable y dos fijos. De esta manera, añadiendo una neurona lineal, se ha añadido una restricción topológica que facilita el comportamiento deseado.

- Hasta ahora solamente se han considerado funciones estáticas, pero las redes neuronales tienen capacidad para mostrar comportamientos dinámicos. Para ello se utilizan las conexiones ‘hacia atrás’, que introducen un retardo temporal de una muestra.



a) suma en el tiempo b) resta en el tiempo

Fig. 2.16: ejemplos de redes dinámicas sencillas

Unas funciones sencillas de realizar y que, nuevamente, dan juego para incorporar comportamientos dinámicos más complejos en otros módulos son las aproximaciones discretas de la integral y la derivada. La figura 2.16 muestra estas dos redes dinámicas.

– Finalmente, la interconexión de módulos neuronales es una fuente inagotable de funciones. Por ejemplo, se verá en la siguiente sección que el NM que aproxima la zona muerta se puede generar a partir del NM que aproxima la saturación, el del juego de engranajes se puede generar a partir del módulo de la zona muerta y otros ejemplos.

Con la ayuda de estas pautas es posible diseñar módulos neuronales que aproximan las funciones que se describen en la siguiente sección, que son lineales a tramos, continuas y estáticas o dinámicas. Pero también se pueden diseñar funciones de características diferentes, como por ejemplo funciones de varias entradas (ver [Mor-96a]).

2.4 Diseño de un conjunto de NMs útil para el modelado de sistemas no lineales

El conjunto básico de no-linealidades para el modelado de sistemas dinámicos que se ha considerado de interés en este estudio consta de 9 tipos de no-linealidades y para cada uno de ellos se ha diseñado el correspondiente módulo neuronal, explicado con detalle en [Mor-96a]. Además, también ha sido necesario diseñar NMs para modelar el comportamiento de sistemas lineales. En esta sección se da una descripción de cada uno de estos NMs, indicando las relaciones entre sus parámetros y los de la familia de funciones que aproximan.

Se notan con mayúsculas las familias de funciones no lineales y sus aproximaciones se notan añadiendo al nombre de la familia las letras NM. Por ejemplo SAT representa la familia de funciones de tipo saturación y SATNM su correspondiente módulo neuronal.

2.4.1 No-linealidad tipo umbral

La no-linealidad tipo umbral puede dar lugar a múltiples funciones de interés en el modelado de sistemas no lineales: desde funciones sencillas como las características ON-OFF hasta otras más complejas como comparadores de señales, etc.

La familia de funciones umbral se define según:

$$\text{UMBRAL} = \left\{ f_\theta(x) = \begin{cases} k1 & x \le \alpha \\ k2 & x > \alpha \end{cases} \quad | \quad \theta = \{\alpha, k1, k2 \in [a,b]\} \right\} \qquad (2.20)$$

y su realización neuronal se puede llevar a cabo con 2 neuronas y 4 pesos. La relación entre los parámetros de UMBRAL y UMBRALNM es la siguiente:

$$\text{UMBRALNM} = \left\{ \tilde{f}_{\theta,\omega} : [a,b] \to [k1, k2] \quad | \quad \theta = \{\alpha, w1, w2\}, \omega = \{w_d\} \right\} \qquad (2.21)$$

con $\tilde{f}(x) = w1 \cdot umbral1_\alpha(x) + w2$ y $w1 = \dfrac{k1 - k2}{2}$, $w2 = \dfrac{k1 + k2}{2}$.

La figura 2.17 muestra el diagrama correspondiente a la red neuronal que aproxima la familia de funciones UMBRAL. Nótese que este módulo neuronal puede simplificarse para casos concretos del grupo de funciones. Por ejemplo, si $k1 = -k2 = 1$ entonces $w1 = 1$ y $w2 = 0$, con lo cual se puede omitir la neurona con función de activación lineal sin modificar el comportamiento del módulo neuronal. En este caso, utilizar solamente una neurona con dos pesos (uno de los cuales es fijo) será más eficiente en cuanto a velocidad de aprendizaje.

Fig. 2.17: esquema y red neuronal correspondiente a UMBRALNM

2.4.2 No-linealidad tipo relé con histéresis

El efecto de histéresis aparece con frecuencia en componentes ferromagnéticos aunque también se da en elementos plásticos y piezoeléctricos y se suele incluir en los modelos de sistemas como relés y solenoides. A pesar de que se han estudiado modelos muy completos del efecto de

la histéresis (véase para un ejemplo [Tao-96]) en este trabajo se utiliza la más habitual, que contempla principalmente el carácter de desdoblamiento en la respuesta de un sistema. No se consideran pendientes diferentes para los tramos que conforman la respuesta ni lazos menores de histéresis dentro del lazo principal.

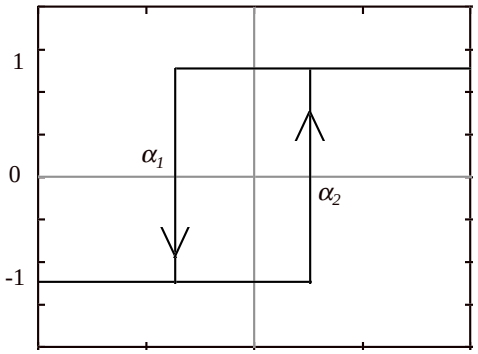


Fig. 2.18: gráfica de la función relé de dos posiciones con histéresis ideal

De hecho, más que estudiar el efecto de la histéresis, en este trabajo se obtiene un módulo neuronal que aproxima el comportamiento ideal de un relé de dos posiciones con histéresis. La figura 2.18 representa la gráfica de dicho elemento y su definición matemática, como función de tiempo discreto es

$$f(x(k)) = \begin{cases} -1 & x(k) \le \alpha_1 \\ f(x(k-1)) & \alpha_1 < x(k) \le \alpha_2 \\ 1 & x(k) > \alpha_2 \end{cases} \qquad (2.22)$$

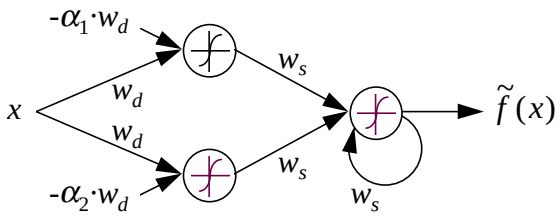


Fig. 2.19: módulo neuronal correspondiente al relé con histéresis (nótese que la conexión hacia atrás lleva implícita un retardo temporal)

La representación gráfica del NM correspondiente al relé con histéresis se da en la figura 2.19 y la relación entre la familia de funciones RELE y RELENM es la siguiente:

$$\text{RELE} = \left\{ f_\theta(x) \quad | \quad \theta = \{\alpha_1, \alpha_2\} \right\}$$

$$\text{RELENM} = \left\{ \tilde{f}_{\theta,\omega} : [a,b] \to [-1,1] \quad | \quad \theta = \{\alpha_1, \alpha_2\}, \omega = \{w_d, w_s\} \right\} \tag{2.23}$$

La última consideración que se realizará acerca de este NM es que el peso w_d permite controlar la sensibilidad de discriminación entre los intervalos de entrada y que w_s permite controlar la pendiente de cambio en la salida.

2.4.3 No-linealidad tipo saturación

La saturación es un efecto causado, generalmente, por limitaciones en el tamaño de los componentes y por limitaciones en las condiciones de operación, como la potencia que puede proporcionar o recibir un elemento. Por ejemplo, la mayoría de los amplificadores se diseñan para comportarse linealmente, pero su rango de operación es finito, por lo que cuando reciben entradas fuera de rango son incapaces de trabajar correctamente, dando lugar al fenómeno de la saturación.

De nuevo, en el contexto de este trabajo se utiliza la definición de la saturación más habitual en la literatura, aunque no es ni la más completa ni la única válida. La saturación ideal y simétrica se define como:

$$f(x) = \begin{cases} -\beta\alpha & x < -\alpha \\ \beta x & -\alpha \le x \le \alpha \\ \beta\alpha & x > \alpha \end{cases} \tag{2.24}$$

que forma parte de la familia de funciones descrita por:

$$\text{SAT} = \left\{ f_\theta(x) \quad | \quad \theta = \{\alpha, \beta\} \right\} \tag{2.25}$$

El módulo neuronal que aproxima el conjunto de funciones SAT queda descrito según:

$$\text{SATNM} = \left\{ \tilde{f}_{\theta,\omega}(x) \quad | \quad \theta = \{\alpha, \beta\}, \omega = \{w_d, w_r, w_s\} \right\} \tag{2.26}$$

La figura 2.20 muestra el diagrama de la red neuronal que aproxima la familia de funciones SAT. Nótese que este módulo neuronal tiene un total de 16 pesos, de los cuales solamente 2 serán modificables en el proceso de aprendizaje. Estos pesos son α y β, que corresponden a los parámetros de definición de la función.

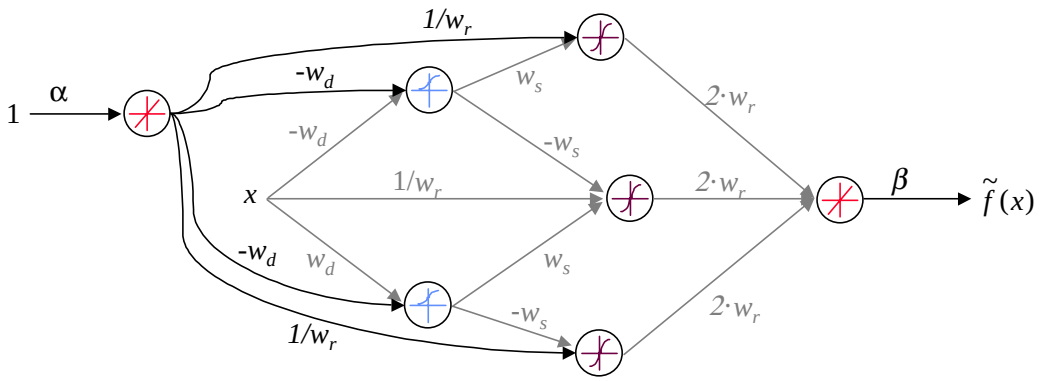


Fig. 2.20: esquema de la red neuronal que aproxima la familia de funciones SAT

2.4.4 No-linealidad tipo zona muerta

La zona muerta es una función estática no lineal caracterizada por dar salida nula para un determinado rango de entrada y por una relación lineal de pendiente constante entre la salida y la entrada fuera de este rango. En la figura 2.21 se da una gráfica de ella.

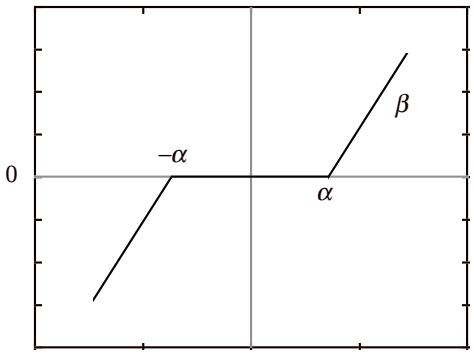


Fig. 2.21: gráfica de la característica de la zona muerta

Esta no-linealidad se da, generalmente, en sistemas físicos en los que es necesaria una cierta energía inicial para ponerlos en movimiento, como muchos motores de corriente continua, ciertos actuadores, sensores, etc. Su definición matemática, para el caso de una zona muerta simétrica, es:

$$f(x)=\begin{cases}\beta(x+\alpha) & x<-\alpha\\ 0 & -\alpha\le x\le\alpha\\ \beta(x-\alpha) & x>\alpha\end{cases} \qquad (2.28)$$

La manera más sencilla de generar la zona muerta es a partir del diseño de la saturación. Es fácil comprobar que si $f_\theta(x) \in$ SAT y $g_\theta(x) \in$ ZONAMUERTA, $g_\theta(x) = x - f_\theta(x)$. De esta manera, la red neuronal que lleva a cabo la zona muerta es la de la figura 2.22, parecida a la de la saturación pero con los pesos que llegan a la neurona de salida cambiados de signo y con un peso adicional que lleva la entrada a la salida.

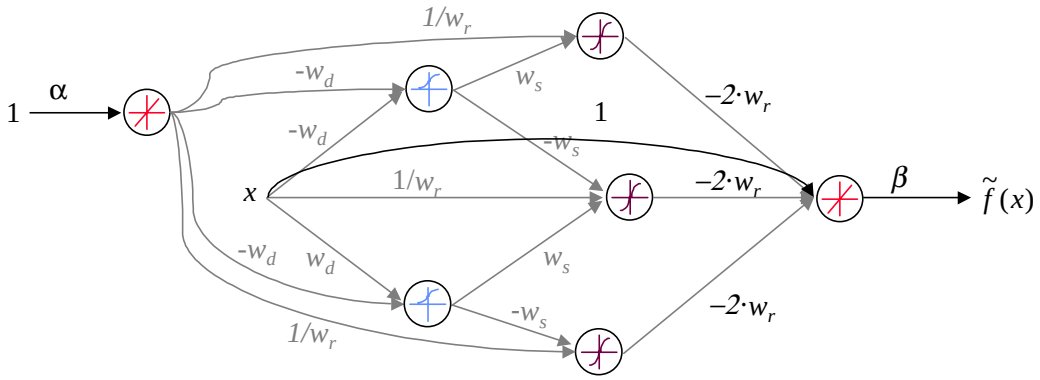


Fig. 2.22: módulo neuronal de la zona muerta

2.4.5 No-linealidad tipo valor absoluto

La característica del valor absoluto se da escasamente en la naturaleza, sin embargo, su realización con redes neuronales puede ser muy útil puesto que, además de aparecer en sistemas electrónicos y mecánicos con cierta frecuencia, permite la construcción de otras no-linealidades más complejas.

$$f(x) = \begin{cases} -x & x < 0 \\ x & x \geq 0 \end{cases} \qquad x \in [a,b] \tag{2.30}$$

La definición matemática del valor absoluto (2.30) permite advertir que no se trata de una familia de funciones, como en los casos anteriores, sino de una única función estática que carece de parámetros. Por esto, el módulo neuronal asociado a ella no tiene pesos modificables, cosa que se refleja seguidamente:

$$\text{ABS} = \left\{ f_\theta(x) \quad | \quad \theta = \varnothing \right\} \tag{2.31}$$

$$\text{ABSNM} = \left\{ \tilde{f}_{\theta,\omega}(x) \quad | \quad \theta = \varnothing, \omega = \{w_r, w_d, w_s\} \right\} \tag{2.32}$$

El diseño del NM asociado al valor absoluto parte del resultado en (2.17) cuando $(\alpha_{i-1}, \alpha_i) = (a, 0)$, siendo $[a, b]$ el rango máximo de la entrada. La figura 2.23 muestra la representación gráfica de este módulo, que carece de pesos modificables, a pesar de la presencia del parámetro α, que en realidad tiene que especificarse a priori o, si se desconoce, dársele un valor conservador.

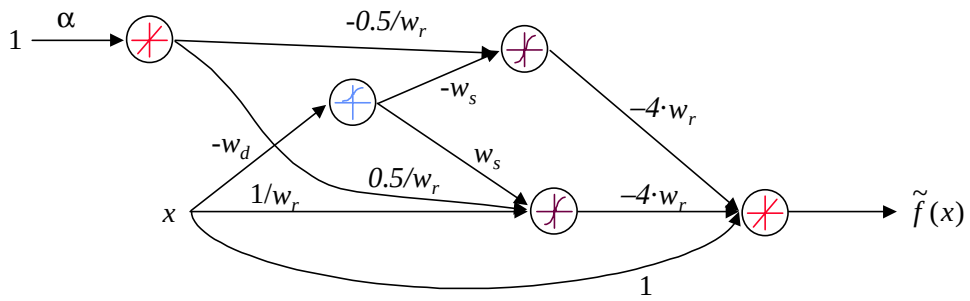
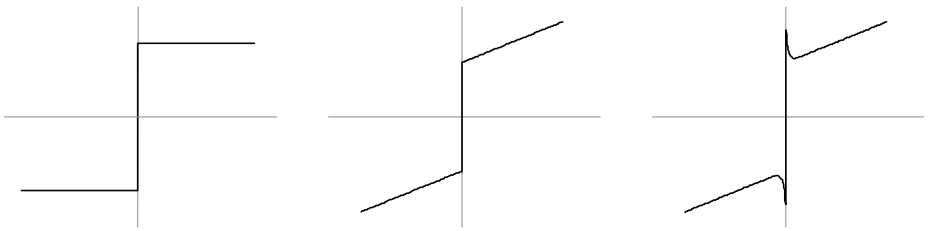


Fig. 2.23: módulo neuronal del valor absoluto

2.4.6 No-linealidad tipo fricción

La fricción produce una fuerza proporcional a la velocidad y opuesta al movimiento entre cuerpos (o fluidos) que están en contacto. Es un efecto que se da en multitud de sistemas y, especialmente, en cualquiera que contenga partes móviles.



a) fricción de Coulomb

b) fricción de Coulomb con efectos viscosos

c) fricción estática con efectos viscosos

Fig. 2.24: gráficas de tres modelos del efecto no lineal de la fricción

Su modelo matemático más completo (cuya gráfica está en la figura 2.24c) incluye la fricción estática, que refleja el hecho de que la fuerza necesaria para iniciar el movimiento suele ser mayor que la que se le opone cuando éste ha comenzado. La figura 2.24a representa la gráfica de la fricción de Coulomb y la figura 2.24b representa la gráfica de la fricción de Coulomb con efectos viscosos, cuya relación matemática es:

$$f(x) = \begin{cases} \beta x - \alpha & x < 0 \\ \beta x + \alpha & x > 0 \end{cases} \tag{2.33}$$

Los parámetros α y β corresponden, respectivamente, a la altura de la discontinuidad en el cero y a la pendiente de la parte lineal (la fricción viscosa clásica). La familia de funciones que describe la fricción es:

$$\text{FRICCIÓN} = \{f_\theta(x) \quad | \quad \theta = \{\alpha, \beta\}\} \tag{2.34}$$

El módulo neuronal, FRICCIÓNNM, que aproxima la anterior familia es:

$$\text{FRICCIÓNNM} = \{\tilde{f}_{\theta,\omega}(x) \quad | \quad \theta = \{\alpha, \beta\}, \omega = \{w_d\}\} \tag{2.35}$$

Este módulo neuronal tiene un peso fijo, de valor w_d, y dos pesos modificables, de valores α y β, respectivamente. Su diseño se realiza con una función $umbral1_0$ y se da en la figura 2.25.

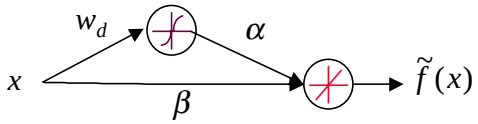


Fig. 2.25: esquema del módulo neuronal FRICCIÓNNM

2.4.7 No-linealidad tipo juego de engranajes

El juego de engranajes (*backlash* en inglés) es una no-linealidad multivaluada que a menudo se considera como un relé con infinitos niveles de salida. Se da en muchos mecanismos de transmisión y, en especial, en los engranajes. La causa del juego de engranajes es el hueco que inevitablemente queda entre los dientes de dos ruedas engranadas. Cuando la rueda impulsora cambia el sentido de giro siempre hay un tiempo en que la rueda impulsada no está en contacto con ella y, por tanto, no se mueve hasta que no se restablezca el contacto. En la figura 2.26a se da una representación gráfica de la característica del juego de engranajes y su definición, en tiempo discreto es:

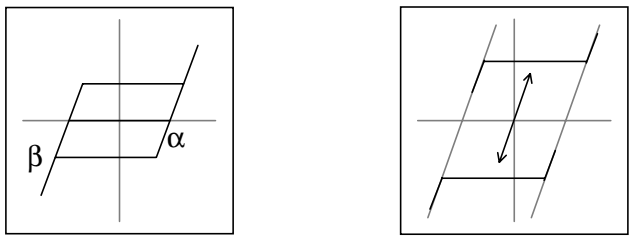
$$f(x(k)) = \begin{cases} \beta(x(k)+\alpha) & x(k) < \dfrac{f(x(k-1))}{\beta} - \alpha \\ f(x(k-1)) & \dfrac{f(x(k-1))}{\beta} - \alpha \le x(k) \le \dfrac{f(x(k-1))}{\beta} + \alpha \\ \beta(x(k)-\alpha) & x(k) > \dfrac{f(x(k-1))}{\beta} + \alpha \end{cases} \quad (2.36)$$

El módulo neuronal JUEGONM que aproxima la familia de funciones JUEGO se define según:

$$\text{JUEGONM} = \left\{ \tilde{f}_{\theta,\omega}(x) \quad | \quad \theta = \{\alpha, \beta\}, \omega = \{w_d, w_r, w_s\} \right\} \quad (2.37)$$

con JUEGO definido como SAT en (2.27)

$$\text{JUEGO} = \left\{ f_\theta(x) \quad | \quad \theta = \{\alpha, \beta\} \right\} \quad (2.38)$$



a) juego b) interpretación

Fig. 2.26: representación gráfica del juego de engranajes

La red neuronal JUEGONM se diseña a partir de observar que el juego de engranajes se puede considerar como una zona muerta que cambia dinámicamente de punto de trabajo. En la figura 2.26b se da una representación gráfica de esta interpretación, en la que el punto de trabajo corresponde a una recta de pendiente β que pasa por el origen de coordenadas. En la figura 2.27 se representa esta red neuronal y se destacan las conexiones que la diferencian de la zona muerta, que son las que permiten cambiar dinámicamente su punto de operación.

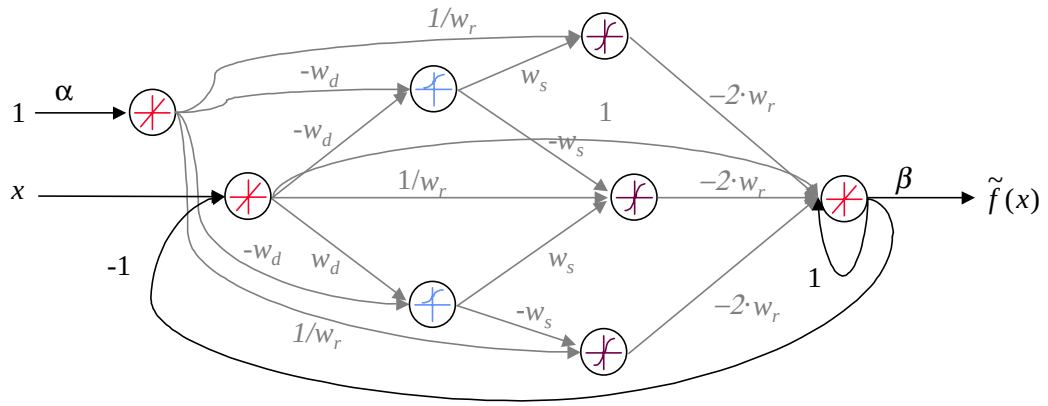


Fig. 2.27: diagrama del módulo neuronal del juego de engranajes (nótese que las conexiones hacia atrás llevan implícitas un retardo temporal)

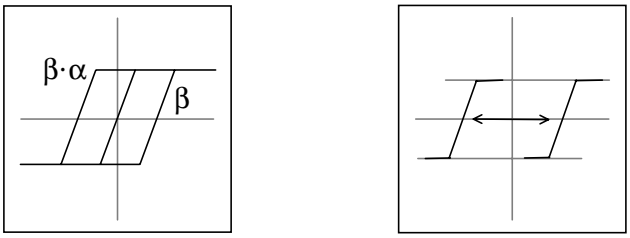
2.4.8 No-linealidad tipo tope mecánico

El tope mecánico es una no-linealidad algo menos corriente que el juego de engranajes y es, respecto a la saturación, lo que el juego es respecto a la zona muerta. En diferentes sistemas mecánicos el desplazamiento de una de las piezas del mecanismo puede estar limitado en dos sentidos permitiendo, a pesar de ello, que la parte conductora se traslade ilimitadamente. Este comportamiento no lineal corresponde al tope mecánico.

La definición, en tiempo discreto es:

$$f(x(k)) = \begin{cases} -\beta\alpha & x(k)-x(k-1) < \dfrac{f(x(k-1))}{\beta} - \alpha \\ \beta(x(k)-x(k-1)) + f(x(k-1)) & \dfrac{f(x(k-1))}{\beta} - \alpha \le x(k)-x(k-1) \le \dfrac{f(x(k-1))}{\beta} + \alpha \\ \beta\alpha & x(k)-x(k-1) > \dfrac{f(x(k-1))}{\beta} + \alpha \end{cases} \quad (2.39)$$

El módulo neuronal TOPENM, que aproxima la familia de funciones TOPE, se obtiene a partir de la observación de que la característica de $f(x(k))$, representada en la figura 2.28a se puede considerar como la de una saturación que cambia el punto de operación dinámicamente, de forma similar a como lo hace la zona muerta en el juego de engranajes. El modo concreto en que esta operación tiene lugar se aprecia en la figura 2.28b.



a) tope b) interpretación

Fig. 2.28: representación gráfica del tope mecánico

Puesto que existe tal paralelismo entre el tope y el juego y la saturación y la zona muerta, la definición de las familias de funciones TOPENM y TOPE es idéntica a la de JUEGONM y JUEGO, respectivamente, y la red neuronal que lleva a cabo la aproximación del juego mecánico se representa en la figura 2.29.

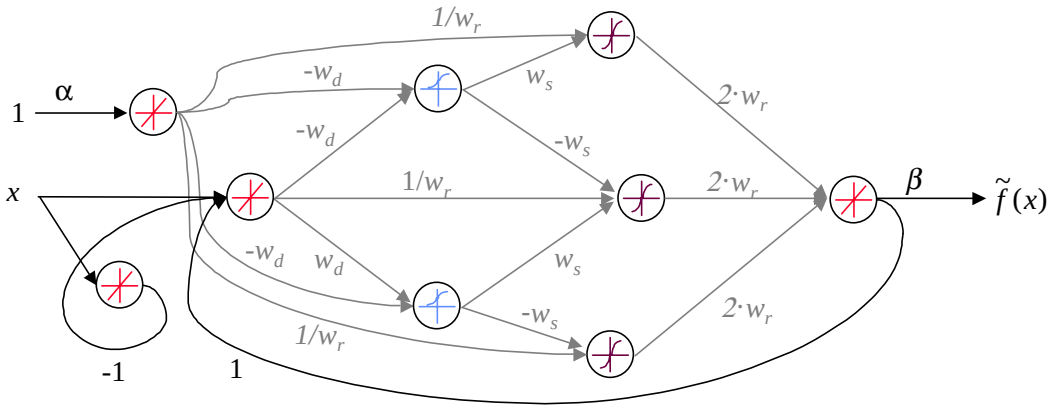


Fig. 2.29: diagrama del NM del juego mecánico (nótese que las conexiones hacia atrás llevan implícitas un retardo temporal)

2.4.9 No-linealidad tipo limitador de velocidad

La característica del limitador de velocidad (*rate limiter* en inglés) se aprecia a menudo en sistemas electrónicos y es una propiedad que consiste en la imposibilidad de un sistema de responder a cambios en su entrada tan rápido como se le supone. Por ejemplo, en los dispositivos lógicos el paso de un estado a otro no se produce instantáneamente (como sería de desear) sino que transcurre un pequeño intervalo de tiempo, asociado a una constante que se llama *slew rate*. El modelo de uno de estos dispositivos se puede confeccionar a base de un modelo de la función lógica que lleva a cabo, junto con la característica del limitador de velocidad.

La expresión, en tiempo discreto, del modelo matemático de limitador de velocidad, ecuación (2.40), desvela que la operación que éste realiza es la de limitar la derivada de la entrada.

$$f(x) = f(x(k-1)) + \begin{cases} -T_M\beta & x(k) - f(x(k-1)) < -T_M\beta \\ x(k) - f(x(k-1)) & -T_M\beta \le x(k) - f(x(k-1)) \le T_M\beta \\ T_M\beta & T_M\beta < x(k) - f(x(k-1)) \end{cases} \quad (2.40)$$

Es decir, se trata de una saturación impuesta sobre la velocidad en la entrada. De esta manera, la salida es idéntica a la entrada siempre y cuando ésta no varíe más rápidamente que $T_M{\cdot}\beta$, el producto entre el tiempo de muestreo y el *slew rate*. Esta consideración permite realizar el diseño del correspondiente módulo neuronal de forma sencilla a partir del NM correspondiente a la saturación, como se muestra en la figura 2.30.

$$\text{LIMITADOR} = \{f_\theta(x) \quad | \quad \theta = \{\beta\}\} \quad (2.41)$$

$$\text{LIMITADORNM} = \left\{\tilde{f}_{\theta,\omega}(x) \quad | \quad \theta = \{\beta'\}, \beta' = T_M\beta, \quad \omega = \{w_d, w_r, w_s\}\right\} \quad (2.42)$$

En la definición de LIMITADORNM se observa que el conjunto de parámetros que conforman la familia de funciones, θ, no es idéntico al correspondiente conjunto en la familia LIMITADOR. Sin embargo, esto no es un problema puesto que el tiempo de muestreo de los datos será, en general, conocido y así se podrá saber el valor identificado de β.

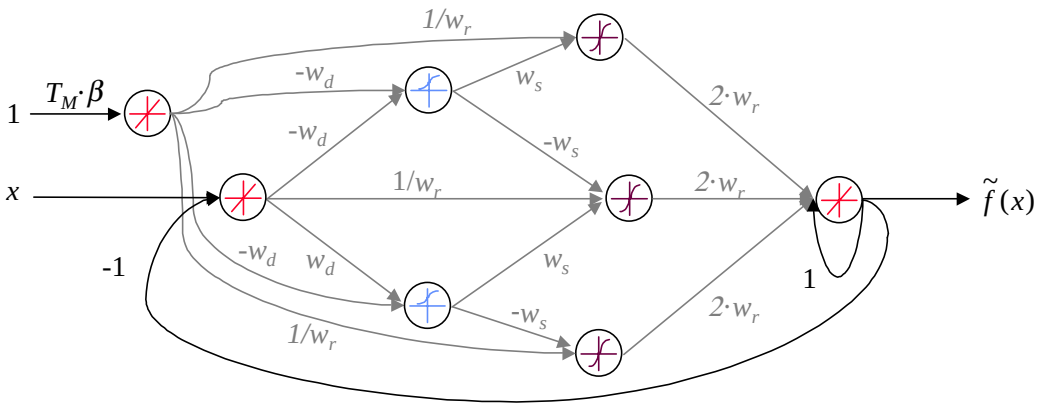


Fig. 2.30: diagrama del NM del limitador de velocidad (nótese que las conexiones hacia atrás llevan implícitas un retardo temporal)

2.4.10 Sistemas lineales

Los sistemas lineales pueden realizarse mediante redes de neuronas lineales. Su realización se basa en las dos formas de representación clásicas, la interna y la externa.

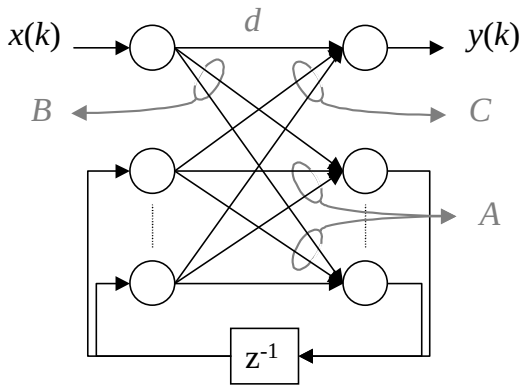


Fig. 2.31: representación en el espacio de estados de un sistema lineal en forma de red neuronal

En la estructura que corresponde a la representación en el espacio de estados, figura 2.31, los pesos son las matrices de estado, entradas, salidas y directa del sistema. De esta manera, la identificación de un sistema lineal consiste en el aprendizaje de estos valores y su interpretación es inmediata.

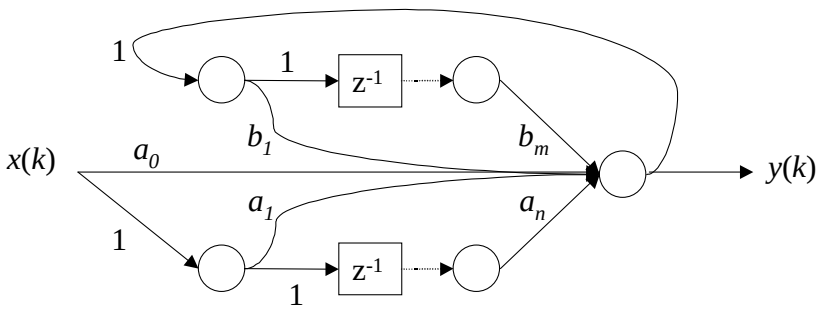


Fig. 2.32: representación de entradas y salidas de un sistema lineal en forma de red neuronal

En la estructura correspondiente a la representación de entradas y salidas, figura 2.32, los pesos son los coeficientes del numerador y del denominador de la función de transferencia del sistema. Así, igual que en el caso anterior, la identificación consiste en el aprendizaje de estos valores y su interpretación es, del mismo modo, inmediata.

A pesar de que ambas representaciones parecen igualmente útiles se ha tomado la segunda por varias razones: la representación interna puede utilizar más parámetros de los necesarios. Es decir, cuando un sistema de orden n pueda caracterizarse con menos de $n \times n + 2n + 1$ parámetros la representación externa será más eficiente y requerirá menor cantidad de pesos a adaptar; el aprendizaje de las matrices de estado es más complejo que el de los coeficientes de la función de transferencia. Se ha podido comprobar empíricamente que el aprendizaje de redes neuronales en representación externa genera, ante condiciones iniciales aleatorias, mayor cantidad de modelos correctos que la interna; y, finalmente, esto facilita el aprendizaje en redes de módulos neuronales entre los cuales hay módulos lineales.