ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Técnicas de compresión de tablas de datos mediante regresiones lineales, redes neuronales y sistemas fuzzy

AUTOR: Ismael Nicolás Nicolás
DIRECTOR: Juan Hinojosa Jimenez

Abril / **2008**

Autor	Ismael Nicolás Nicolás
E-mail del Autor	[vespu18@hotmail.com]
Director	Juan Hinojosa Jimenez
E-mail del Director	[juan.hinojosa@upct.es]
Título del PFC	Técnicas de compresión de tablas de datos mediante regresiones lineales, redes neuronales y sistemas fuzzy

Resúmen

Utilizando simuladores electromagnéticos para diseñar y optimizar circuitos de radiofrecuencias/microondas obtenemos una gran precisión, pero en cambio el manejo de datos es mucho mayor.

Hay un deseo común para encontrar nuevas técnicas de almacenamiento y compresión de datos con un alto índice de precisión. Por ello, dedicaremos este proyecto fin de carrera en desarrollar nuevas técnicas de compresión de datos.

En este proyecto vamos a explicar y exponer los resultados de compresión obtenidos a través de las tres técnicas de compresión de que se compone.

Titulación	Ing. Técnico de Telecomunicación, Esp. Telemática
Departamento	Electrónica, Tecnología de Computadores y Proyectos
Fecha de Presentación	Abril - 2008

Técnicas de compresión de tablas de datos mediante regresiones lineales, redes neuronales y sistemas fuzzy

1. Introducción

En la actualidad, existen muchas herramientas de diseño asistido por ordenador (Advanced Design System, Microwave Office, etc.) que permiten diseñar y optimizar circuitos de radiofrecuencias/microondas con una gran precisión con respecto a medidas experimentales y con una gran velocidad de cómputo. Sin embargo, muchas de estas herramientas no permiten llevar a cabo simulaciones de dispositivos con ciertas dimensiones geométricas, nuevos dispositivos y materiales específicos (anisótropos, ferroeléctricos, etc.). Para solucionar estos problemas, tenemos que utilizar técnicas de simulación y análisis electromagnéticos en los dominios de la frecuencia o del tiempo. Estas técnicas necesitan un elevado tiempo de cómputo con respecto a las técnicas iniciales y el manejo de una gran cantidad de datos. El problema que vamos a tratar en este proyecto es la ingente cantidad de datos involucrados en los simuladores electromagnéticos.

Hay un deseo común por parte de la industria para encontrar nuevas técnicas de almacenamiento y compresión de datos con un alto índice de precisión. Por ello, dedicaremos este proyecto fin de carrera en desarrollar nuevas técnicas de compresión de datos. La compresión de datos es un recurso que en la actualidad se utiliza para todos campos de las telecomunicaciones (señales, imágenes, datos, etc.) y consiste en obtener un archivo comprimido que conserve las propiedades del archivo original, pero que ocupe un menor espacio para almacenarse. Las ventajas de la compresión son evidentes: Obtener un mejor aprovechamiento de los soportes de almacenamiento de datos debido a que podemos tener una mayor cantidad de archivos en el mismo espacio de almacenamiento.

Volviendo al caso de los simuladores electromagnéticos, estos generan grandes tablas de datos en las cuales hay una gran cantidad de entradas y salidas. Nuestro objetivo consiste

en desarrollar técnicas de compresión que se ajusten a lo que actualmente se esta demandando en la industria para este tipo de problemas:

- Alto índice de compresión.
- Excelente precisión.
- Bajo tiempo de ejecución.

Estructura del Proyecto Fin de Carrera:

Este proyecto fin de carrera se descompone en seis capítulo incluyendo la introducción:

- El segundo capítulo describe diferentes métodos de aproximaciones tales como regresiones lineales, sistemas fuzzy y redes neuronales que nos serán útil para el desarrollo de distintas técnicas de compresión implicadas en este proyecto.

- El tercer capítulo presenta las distintas técnicas de compresión de datos que hemos desarrollado. Están basadas en técnicas híbridas con distintas estructuras y combinan regresiones lineales con un sistema fuzzy o una red neuronal. Las más representativas son: La técnica de diferencia de fuente y la técnica con entrada de conocimiento previa.

- El cuarto capítulo es relativo al formato de los datos que utilizaremos para aplicar nuestras técnicas de compresión presentadas en el tercer capítulo. Los

datos se obtuvieron de dos dispositivos mediante simulador electromagnético. Finalmente, definimos una relación para evaluar el índice de compresión de las técnicas.

- El quinto capítulo muestra los resultados de compresión de las distintas técnicas desarrolladas para las dos tablas de datos obtenidas de las simulaciones electromagnéticas de los dos dispositivos. Se realiza un estudio de los resultados para elegir la técnica de compresión que presenta mayores prestaciones (índice de compresión, menor tiempo en comprimir los datos, menor error en la descompresión).

- El sexto y último capítulo presenta nuestras conclusiones y perspectivas.

2. Regresiones lineales, redes neuronales y sistemas fuzzy

2.1 Regresiones Lineales

Las regresiones lineales se suelen utilizar para modelar la respuesta de una función de una o varias variables [1]. Existen varios tipos de regresiones. En este apartado, haremos una breve descripción de las más representativas como la regresión lineal simple y regresión lineal múltiple, siendo la regresión lineal múltiple basada en polinomios la que utilizaremos para la compresión de datos.

2.1.1. Regresión lineal simple.

Debido a su simplicidad analítica, la forma funcional que más se utiliza en la práctica es la relación lineal cuando solo existe una variable independiente. Esto se reduce a una línea recta:

$$y = b_0 + b_1 x \tag{2.1}$$

Disponemos de un conjunto de valores de entrada contenidos en x y un conjunto de valores de salida en y. Los coeficientes b_0 y b_1 son parámetros que definen la posición e inclinación de la recta. El parámetro b_0 (la ordenada en el origen), nos indica el valor de y cuando x vale 0. El parámetro b_1 (la pendiente), nos indica cuanto aumenta y por cada aumento de x. En el análisis de regresión, estas estimaciones se obtienen por medio del método de mínimos cuadrados.

Se conoce como recta de regresión lineal, al procedimiento de encontrar la ecuación de la recta que mejor se ajusta a un conjunto de puntos. A continuación, presentamos un ejemplo de diagrama de dispersión (Figura 2.1) en el que se muestra como a partir de los datos de x e y se obtiene la recta de regresión.

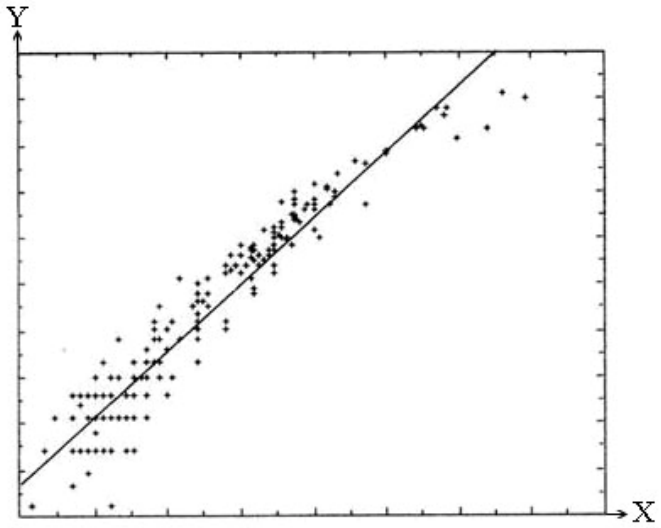


Figura 2.1. Ejemplo de regresión lineal simple.

En este gráfico se aprecia que no existe una relación matemáticamente exacta entre las variables, ya que todos los puntos de ambas variables no varían de una manera relacionada. Si entre estas variables existiera una relación lineal perfecta, entonces todos los puntos caerían a lo largo de la recta de regresión, que también ha sido trazada y que muestra la relación "promedio" que existe entre las dos variables. Como se puede observar, la mayoría de los puntos no caen directamente sobre la recta, sino que están "dispersos" en torno a ella. Esta dispersión representa la variación en *Y* que no puede atribuirse a la variación en *X*.

2.1.2 Regresión lineal múltiple.

En la mayor parte de estudios intervienen una serie de datos agrupados en un conjunto de variables. En algunos casos el análisis de dicha información se lleva a cabo centrando la atención en pequeños subconjuntos de las variables recogidas utilizando para ello análisis sencillos que involucran únicamente técnicas de dos variables (regresiones lineales simples). Sin embargo, un análisis apropiado debe tener en consideración toda la información de interés para el estudio y requiere de técnicas multivariantes más complejas.

En particular, hemos visto como el modelo de regresión lineal simple es un método sencillo para analizar la relación lineal para una variable. En la mayoría de los casos se pretende predecir una respuesta en función de un conjunto más amplio de variables, siendo necesario considerar un modelo de regresión lineal múltiple como una extensión de la recta de regresión que permite la inclusión de un número mayor de variables. La forma generalizada de notación para este tipo de regresión es:

$$(x_1, x_2, \cdots, x_k, y_i) \qquad i = 1, \cdots, n \tag{2.2}$$

donde x_k es el conjunto de datos de la entrada k e y_i es el conjunto de datos de la salida i. El modelo de regresión lineal múltiple con k variables responde a la ecuación:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + E_i \qquad i = 1, \ldots, n \tag{2.3}$$

donde E_i corresponde al error producido en la salida i.

Los coeficientes β_i se estiman siguiendo el criterio de mínimos cuadrados:

$$min \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_1 - \beta_2 x_2 - \cdots - \beta_k x_k) \tag{2.4}$$

Al igual que ocurría en el caso bidimensional, se puede visualizar la relación entre tres variables en un gráfico de dispersión, de modo que la técnica de regresión lineal múltiple proporcionaría el plano que mejor ajusta a la nube de puntos resultante (Figura 2.2):

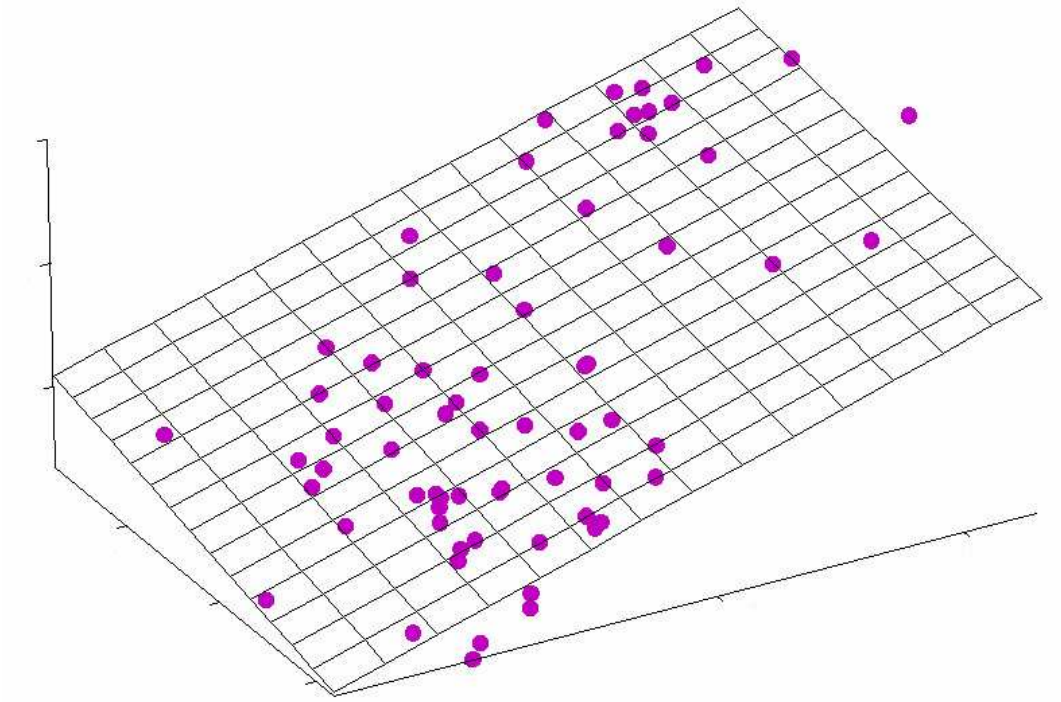


Figura 2.2. Ejemplo de regresión lineal múltiple con 2 entradas

Una variante de las regresiones lineales múltiples es la regresión polinomial, que es la que vamos a utilizar en este PFC. Su formulación general para *k* variables es la siguiente:

$$y(x) = \underbrace{\beta_0 + \sum_{i=1}^{k} \beta_i x_i}_{\text{términos lineales}} + \underbrace{\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \beta_{ij} x_i x_j}_{\textit{productos}\ \text{cruzados}} + \underbrace{\sum_{i=1}^{k} \beta_{ii} x_i^2}_{\text{términos cuadrados}} \qquad i = 1,\ldots,n \tag{2.5}$$

Donde $x = [x_1\ x_2 \cdots x_k]$ es el vector de entrada e *y* corresponde a la salida. El valor de *k* es igual al número total de entradas.

2.2 ANFIS

Antes de explicar el funcionamiento, elementos y estructura de un sistema ANFIS (Adaptive Neuro-Fuzzy Inference System) [2-3], nos centraremos en el funcionamiento de la lógica en la que ANFIS se basa (lógica difusa o fuzzy), y posteriormente, describiremos el funcionamiento de ANFIS.

2.2.1 Lógica Fuzzy

La palabra fuzzy traducida al castellano quiere decir borrosa o difusa, la cual hace alusión a un objeto con los contornos mal definidos. La lógica difusa nos permite manejar y procesar ciertos tipos de información en la que existen términos subjetivos (o inexactos). Al contrario de la lógica tradicional, la lógica difusa trata de emular al cerebro humano realizando un razonamiento basado en reglas imprecisas. Aunque las palabras son mucho menos precisas que los números, su uso esta mas cerca de la intuición humana. La lógica difusa es sinónimo de teoría de conjuntos difusos, la cual identifica a los datos como un tipo de conjunto en los cuales los elementos pertenecen a un subconjunto en un cierto grado. Es decir, un dato pertenece a un conjunto u otro de una manera subjetiva.

Los conjuntos difusos comenzaron a utilizarse en 1965 por L.A.Zadeh para procesar informaciones afectadas de incertidumbre no probabilística [4]. La idea de Zadeh era la de hacer una lógica en la cual un elemento no solo pueda adoptar los valores de 0 y 1 (también equivalentes a verdadero y falso), sino que dentro de ese intervalo pueda adoptar un número infinito de valores. Los conjuntos difusos fueron diseñados para representar la incertidumbre mediante métodos matemáticos y así proporcionar herramientas formalizadas para trabajar con esa imprecisión en los problemas planteados. La clave de esta adaptación al lenguaje, se basa en comprender los cuantificadores de nuestro lenguaje (como por ejemplo "mucho", "muy", "un poco"). A partir de la teoría de conjuntos, Zadeh define la lógica difusa como una

extensión de las lógicas polivaloradas. Los puntos fundamentales en los que Zadeh define la lógica difusa son:

- En la lógica difusa todo es cuestión de grado.
- El razonamiento exacto (el tradicional de verdadero y falso) es un caso límite del razonamiento aproximado.
- En lógica difusa, el conocimiento se interpreta como una colección de restricciones elásticas (difusas) sobre un conjunto de variables.
- En lógica difusa la inferencia puede verse como propagación de un conjunto de restricciones elásticas.
- Un sistema difuso es el resultado de la "fuzzificacion" de un sistema convencional.
- Los sistemas difusos operan con conjuntos difusos en lugar de números.
- La representación de la información en sistemas difusos imita el mecanismo de razonamiento aproximado que realiza la mente humana.

Para cada conjunto difuso, existe asociada una función de pertenencia (FP) para sus elementos, la cual indica en qué medida el elemento forma parte de ese conjunto difuso. Para tratar de entender mejor la lógica difusa, vamos a ilustrarlo con un ejemplo sencillo en el que resulta muy útil la aplicación de esta lógica y explica el uso de las funciones de pertenencia [5]. Vamos a clasificar la temperatura corporal (Tº) de una persona de una manera más próxima a nuestro razonamiento, para ello establecemos tres términos de temperatura corporal: **Normal**, **moderada** y **elevada**. Establecemos que una persona tiene una temperatura corporal **elevada** cuando alcance los 39 ºC. Si tenemos un paciente con una temperatura corporal de 38.9 ºC utilizando nuestro razonamiento sabemos que esa persona está en la misma situación que la persona con 39 ºC.

Supongamos que disponemos de una máquina que continuamente comprueba la temperatura corporal de los pacientes y avisa a las enfermeras cuando la temperatura es elevada. Si esta maquina funcionara con lógica tradicional un paciente con 38.9 ºC está clasificado como persona con temperatura corporal **moderada**, mientras que es una situación igual de grave que el de una persona que tenga 39 ºC. Además, si el paciente estuviera continuamente oscilando entre 38.9 y 39ºC la alarma estaría activándose y desactivándose (porque se estaría cambiando de estado cada vez entre temperatura moderada y elevada).

Lo que se propone implementar con la lógica difusa es definir para cada conjunto (pacientes con temperatura corporal elevada, normal y moderada) una función de pertenencia (FP) tal y como se muestra en la figura 2.3.

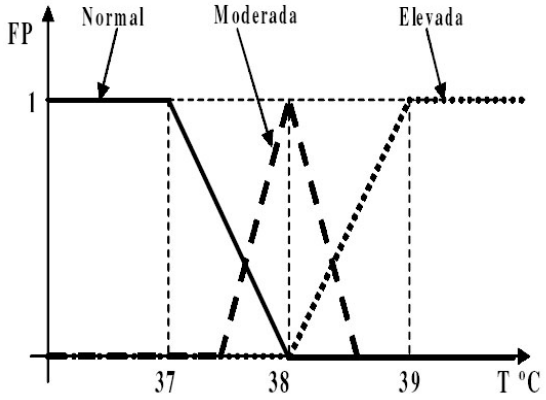


Figura 2.3. Funciones de pertenencia de la Tº.

Mediante el uso de las funciones de pertenencia un paciente con una temperatura corporal de 36.5 ºC pertenece al conjunto **normal** (FP = 1). Otro paciente que tenga una temperatura corporal de 38.2 ºC pertenece a la vez a dos conjuntos (pero con distinto grado de pertenencia), el primer conjunto sería el de temperatura corporal **moderada** (con FP = 0.6) y el segundo conjunto sería el de temperatura corporal **elevada** (con FP = 0.2).

Las funciones de pertenencia se establecen de forma arbitraria (generalmente por la experiencia adquirida del diseñador del sistema). Los principales tipos de funciones de pertenencia empleados en la práctica se detallan en la figura 2.4.

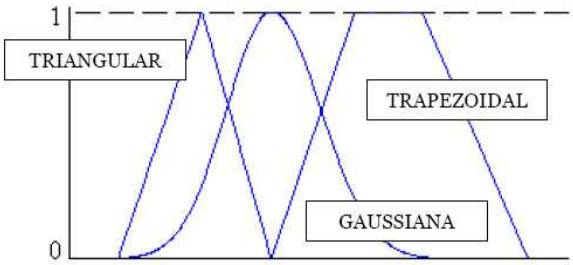


Figura 2.4. Principales tipos de funciones de pertenencia

Para poder operar con conjuntos difusos, es necesario conocer el conjunto de operaciones que se les pueden aplicar (ecuación 2.6). Suponiendo que disponemos de un conjunto denominado X cuya función de pertenencia es FP(X) y además disponemos de otro conjunto Y con función de pertenencia FP(Y), las operaciones que se pueden realizar con estos dos conjuntos se muestran en (2.6).

Intersección o AND	FP (X AND Y) = mínimo de (FP (X), FP(Y))
Unión u OR	FP (X OR Y) = máximo de (FP (X), FP (Y))
Complemento o NOT	FP (Complemento X) = 1 – FP (X)

(2.6)

Uno de los elementos básicos para el funcionamiento de un sistema difuso son las llamadas reglas de inferencia. El contenido de una regla de inferencia (de una manera lingüística) es una frase en lenguaje natural, proposición o sentencia en la que se involucran términos definidos. Las reglas de inferencia siguen la estructura lógica de dos elementos SI-ENTONCES (IF-THEN). El primer elemento de una regla expresa una condición (por ejemplo “SI la presión es alta…”) y es conocido como antecedente. El segundo elemento expresa la consecuencia que conlleva la condición anterior (“ENTONCES abrir la válvula al 90 %”) y es conocido como consecuente. Otros ejemplos de reglas de inferencia serían los siguientes:

- SI hace mucho calor ENTONCES bajo drásticamente la temperatura.
- SI voy a llegar un poco tarde ENTONCES aumento ligeramente la velocidad.

Las reglas de inferencia (utilizándolas en conjunto) nos permiten tener una manera de actuar ante determinadas situaciones muy fáciles de entender con razonamiento difuso pero que son extremadamente difíciles de moldear mediante lógica tradicional. Los resultados de dichos métodos son un área final resultante del conjunto de áreas de cada regla de inferencia solapadas. A la hora de utilizar un criterio que se asemeje a la realidad, es recomendable utilizar lógica difusa, pero no siempre es la mejor solución para todos los problemas de clasificación de datos. Para saber si tenemos que utilizar esta lógica tenemos que tener en cuenta si pertenecen a los siguientes casos:

- En modelos complejos (si no existe una solución sencilla).
- Cuando ciertas partes del sistema pueden contener errores.
- Cuando al ajustar una variable se produce el desajuste de otras.
- Reconocimiento de patrones.
- Sistemas de información (bases de datos).

Las reglas de las que dispone el motor de inferencia de un sistema difuso pueden ser formuladas por personas con experiencia en este campo, o bien aprendidas por el sistema, haciendo uso en este caso de redes neuronales para fortalecer las futuras tomas de decisiones. Una vez entendidos los conocimientos necesarios para entender todos los elementos de la lógica difusa, a continuación vamos a explicar detalladamente y siguiendo los pasos como se seguirían realmente, un ejemplo completo de adaptación de un problema hipotético real a lógica difusa.

Disponemos de una maquina industrial y nuestro trabajo es implementar un sistema difuso para mantener su funcionamiento. Tenemos que tener en cuenta las variables de entrada y de salida. Las variables de entrada que nos interesan son la presión **P** y el flujo **F.** Disponemos de una sola variable de salida que es la temperatura de la máquina **T**. Con la

experiencia de alguien que ha trabajado con sistemas difusos, lo primero que tenemos que hacer es definir que términos vamos a utilizar para cada una de las variables de que disponemos (posteriormente los utilizaremos para la elaboración de las reglas de inferencia de nuestra máquina). Vamos a utilizar tres términos para cada variable del problema:

- Para la variable presión utilizamos los términos **alta**, **media** y **baja**.
- Para la variable flujo utilizamos los términos **mucho**, **normal** y **poco**.
- Para la variable temperatura utilizamos los términos **frió**, **tibio** y **caliente**.

El siguiente paso sería definir las funciones de pertenencia de las tres variables. En este paso la elección de los distintos grados de pertenencia son escogidos por nosotros de manera subjetiva según queramos que funcione nuestra máquina. Para las tres variables que estamos utilizando tenemos que establecer los valores limite (FP = 0 y FP = 1), y para este caso (no tiene porque ser así) la región interior entre FP = 0 y FP = 1 toman valores intermedios. Una vez hemos establecido las funciones de pertenencia, el siguiente paso es la definición de las reglas de inferencia. Estas reglas tienen que reflejar la forma de pensar de la persona que diseña el manejo de la máquina. Ejemplos de reglas de inferencia para nuestra máquina podría ser:

- SI **P** es **alta** AND **F** es **mucho** ENTONCES **T** es **frío**.
- SI **P** es **media** AND **F** es **normal** ENTONCES **T** es **tibio**.

En este ejemplo hemos establecido tres términos para cada variable. Al tener dos variables de entrada, el número de combinaciones de reglas de inferencia posibles con las variables de entrada serán nueve tal y como se detalla en la tabla 2.1.

		Presión P		
		Baja	Media	Alta
Flujo F	Mucho	T es tibio	T es frío	T es frío
	Normal	T es caliente	T es tibio	T es frío
	Poco	T es caliente	T es caliente	T es tibio

Tabla 2.1: Combinación de las reglas de entrada

Cuando hemos terminado la definición de las reglas de inferencia nuestro sistema esta listo para funcionar y ser testado, por lo que entra en un bucle cerrado donde se van siguiendo cíclicamente los siguientes pasos.

- Se leen los valores proporcionados por las entradas **P** y **F**.
- Se calculan las funciones de pertenencia de los tres términos de cada una de las entradas (es el proceso explicado anteriormente como fuzzificación).
- Se evalúan los antecedentes de las reglas y se le asigna el valor de operación AND (por ejemplo) a los elementos consecuentes de las reglas.
- Se inicia el proceso en el que los elementos difusos se transforman en una salida real para la salida T (defuzzificación).
- La maquina variara la temperatura según haya sido el resultado del sistema difuso.

Existen diferentes tipos de sistemas difusos que se pueden aplicar a la hora de resolver un problema. En este proyecto vamos a centrarnos en la explicación de los dos tipos más utilizados en la actualidad para los sistemas difusos.

- Sistema difuso tipo Mamdani (contiene fuzzificador y defuzzificador).
- Sistema difuso tipo Takagi-Sugeno.

2.2.1.1 Sistema difuso Mamdani

El tipo Mamdani es el más usado en la metodología fuzzy. Fue uno de los primeros sistemas de control construidos usando los conjuntos difusos. Fue propuesto en 1975 por Ebrahim Mamdani [6] como un intento de controlar una combinación de maquina de vapor y caldera con un conjunto de reglas de control obtenido de la experiencia de los trabajadores de la máquina. Un sistema difuso del tipo Mamdani esta compuesto por los elementos que se detallan en la figura 2.5 y que se explican a continuación.

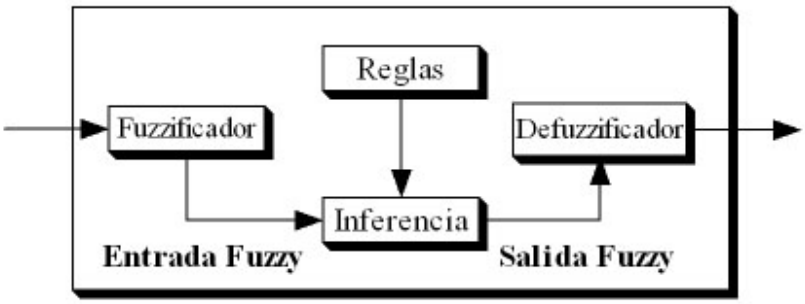


Figura 2.5. Sistema difuso Mamdani

- Fuzzificador: Transforma las entradas normales (por ejemplo un valor numérico proveniente de un sensor de presión, de temperatura) en entradas difusas para que el mecanismo de inferencia pueda utilizarlo. Estos valores difusos son el grado de pertenencia de los valores de entrada a los conjuntos difusos.
- Reglas: Conjunto de reglas difusas que representan la inteligencia del sistema En estas reglas se almacena el contenido lingüístico que le hemos suministrado para la resolución de un problema (son reglas del tipo SI-ENTONCES como las explicadas en ejemplos anteriores).

- Inferencia: Motor inferenciador de las normas fuzzy. El funcionamiento de este motor es la de generar la salida difusa del sistema utilizando el conjunto de reglas y los grados de las funciones de pertenencia de los que dispone.
- Defuzzificador: Recoge la salida que le proporciona el elemento inferencia y transforma esa salida difusa en una salida normal. El sentido de utilizar un defuzzificador es que normalmente los elementos a los que esta conectado un sistema difuso no tienen porque ser difusos (no es lo usual), por lo que si le enviamos una salida difusa no son capaces de entenderla. Tenemos que realizar esta conversión para asegurarnos de que van a entender la salida calculada correctamente.

Para calcular la salida numérica a partir de la salida difusa por medio del defuzzificador se utilizan dos métodos. El primero es el centro de gravedad (2.7) y el segundo es el método de los centros promediados (2.8).

$$y = \frac{\sum_i b_i \int \mu(i)}{\sum_i \int u(i)} \tag{2.7}$$

En este método se cortan las funciones de pertenencia a sus valores máximos (para la salida que se esta calculando), a continuación se superponen las áreas resultantes del corte y se calcula la coordenada del centro de gravedad del área total.

$$y = \frac{\sum_i b_i \mu_{premisa}(i)}{\sum_i \mu_{premisa}(i)} \tag{2.8}$$

En este método no se utilizan las áreas de las funciones de pertenencia, sino que se opera separadamente cada contribución y se calcula un promedio ponderado de los centros de las contribuciones.

Para comprender mejor todo lo explicado sobre el sistema difuso Mamdani, la figura 2.6 ilustra el funcionamiento (independientemente del método de salida) de un sistema difuso Mamdani.

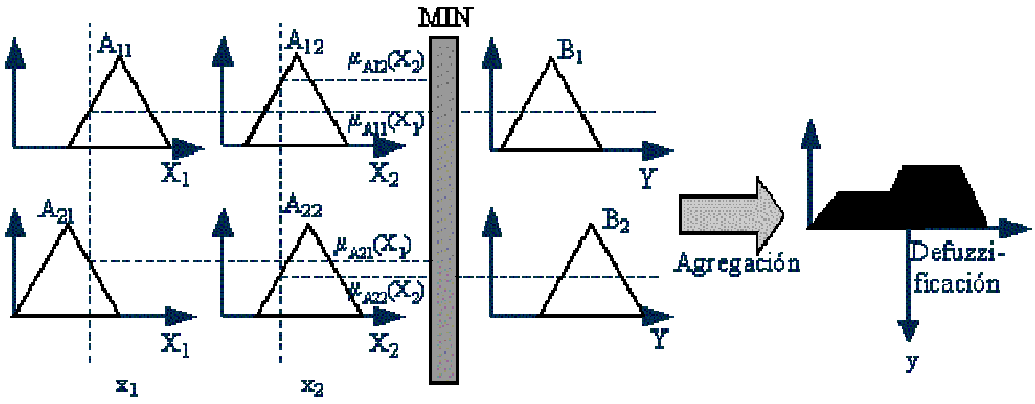


Figura 2.6. Sistema difuso Mamdani, procesamiento detallado

2.2.1.2 Sistema difuso Sugeno

Los sistemas tipo Sugeno se pueden utilizar para cualquier modelo en el que las funciones de pertenencia de salida sean lineales o constantes [7]. Un sistema difuso del tipo Sugeno esta compuesto por los elementos que se detallan en la figura 2.7 y que se explican a continuación.

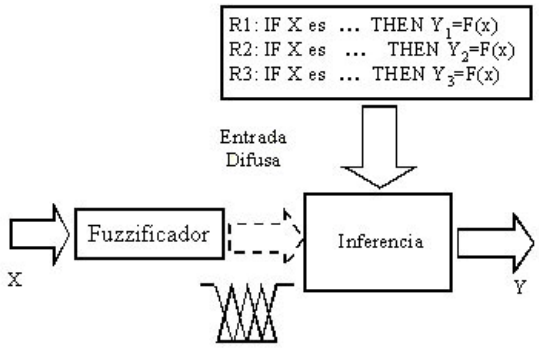
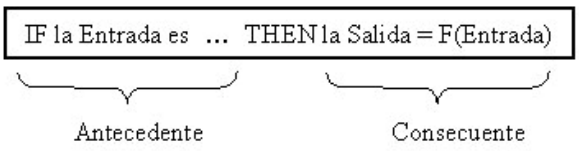


Figura 2.7. Sistema difuso Sugeno

- Fuzzificador: Transforma las entradas normales en entradas difusas entendibles por el sistema Sugeno (el funcionamiento es idéntico que el explicado en el anterior sistema difuso).
- Reglas: El conjunto de reglas que se utilizan en un sistema sugeno son diferentes que las que hemos explicado anteriormente para Mamdani debido que el elemento

consecuente de las reglas (ENTONCES) ya no es una etiqueta lingüística, sino que es una función de entrada que tenga nuestro sistema en un momento dado (la estructura de las reglas se observa en la ecuación (2.9).



(2.9)

- Inferencia: El motor inferenciador de este sistema tiene el mismo funcionamiento que el sistema difuso mamdani que hemos explicado anteriormente.

En este sistema no es necesario tener un elemento defuzzificador que nos convierta nuestra salida difusa en una salida numérica, ya que los valores que nos proporcionan el conjunto de reglas ya son valores numéricos. Para calcular la salida del sistema Sugeno se realiza una ponderación de los diferentes consecuentes (hay que tener en cuenta cual fue el valor que activo el antecedente para cada una de las reglas). Utilizando un ejemplo para entender esto, para un sistema difuso de tipo Sugeno con dos reglas la salida se calcularía con la expresión (2.10). y_1 e y_2 son las funciones que permiten calcular el consecuente de las dos reglas del ejemplo. W_1 y W_2 son los niveles de activación del antecedente de las reglas. W_1 es el producto de los niveles de pertenencia del valor x_1 al conjunto A_{11} y el nivel de pertenencia del valor de x_2 al conjunto A_{12} (las líneas punteadas que llegan a la barra de producto de la figura 2.8). W_2 es el producto de los niveles de pertenencia del valor x_1 al conjunto A_{21} y el nivel de pertenencia del valor de x_2 al conjunto A_{22}.

$$\begin{matrix} y_1 = f_1(x) \\ y_2 = f_2(x) \end{matrix} \qquad y = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2} \tag{2.10}$$

Para comprender mejor todo lo explicado sobre el sistema difuso Sugeno, la figura 2.8 ilustra como seria el funcionamiento de uno de estos sistemas difusos.

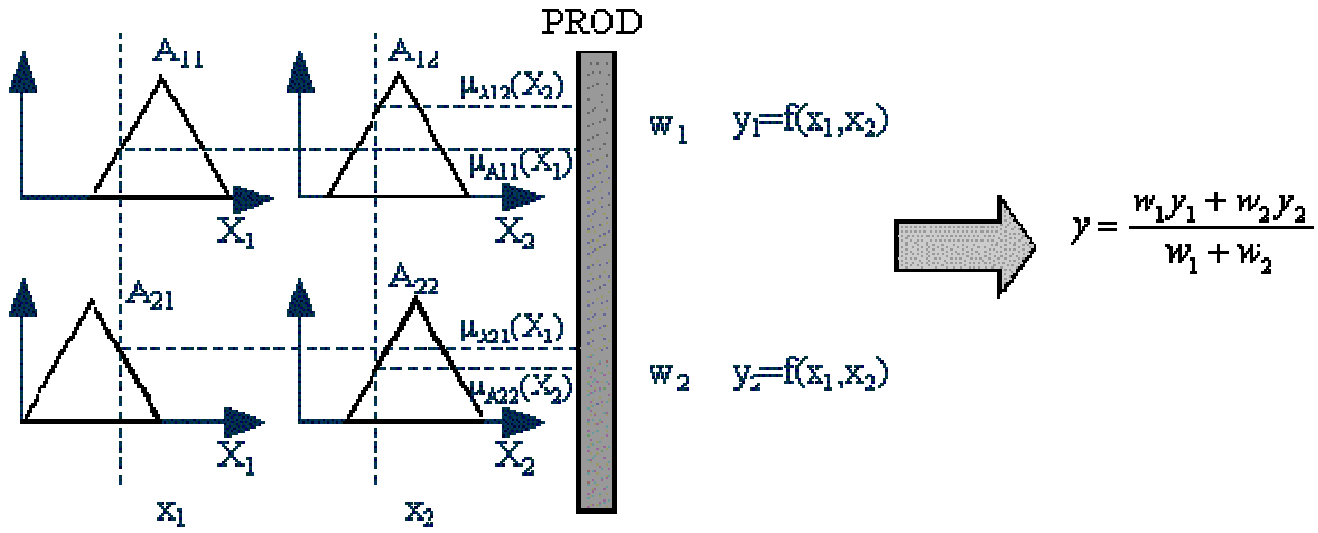


Figura 2.8. Funcionamiento de un sistema difuso Sugeno

A la hora de decidir que método nos conviene más hay que tener en cuenta las ventajas de cada uno:

Ventajas del método Mamdani:

- Es intuitivo.
- Cuenta con una amplia aceptación.
- Se adapta bien a entradas humanas (más reales).

Ventajas del método Sugeno:

- Es eficiente computacionalmente.
- Funciona bien con técnicas lineales.
- Funciona bien con técnicas de optimización y adaptación.
- Es adecuado para el análisis matemático.

Otro ejemplo de cómo funcionaria un sistema difuso con la construcción de sus reglas se observa en la figura 2.9 [3]. Este ejemplo determina que propina tenemos que dar en función de lo que nos ofrecen (las entradas). Disponemos de dos entradas, una salida y tres

reglas. Las dos entradas del ejemplo son, el servicio que recibimos por la comida y la calidad de la comida.

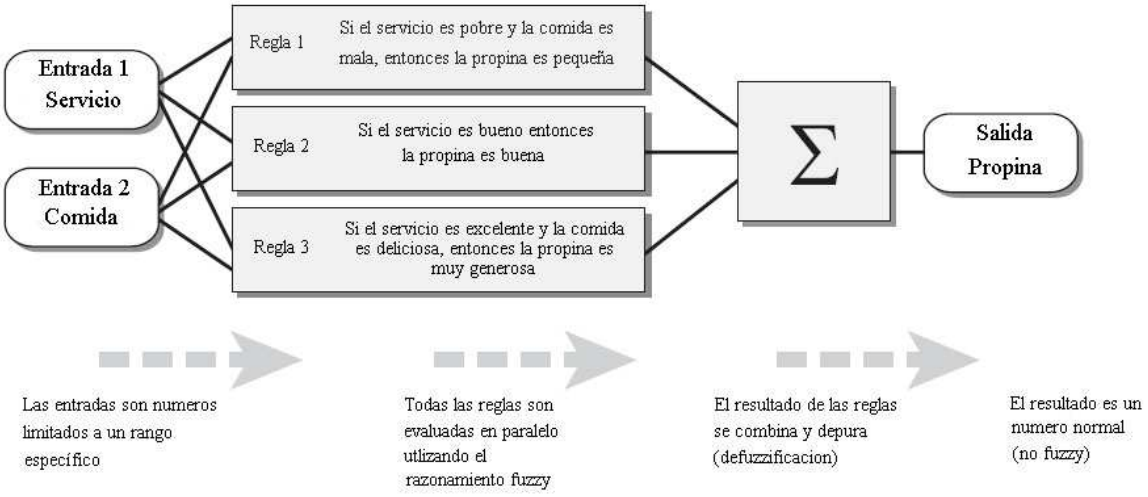


Figura 2.9. Esquema del ejemplo propina FIS

Una vez diseñado el sistema y establecidas las reglas de este sistema difuso, vamos a ir introduciendo entradas e inmediatamente vamos a obtener la salida calculada a partir de esas entradas. En primer lugar establecemos las reglas según nuestro criterio van a clasificar una propina según nuestra manera de pensar. En función del valor de las dos entradas, se aplican las reglas anteriormente creadas (en nuestro ejemplo utilizamos las tres reglas que se muestran en la figura 2.9) y se combinan para obtener un resultado final. Este resultado será un número real calculado por el sistema difuso (no es un numero difuso) y nos dirá cual es la propina exacta que tenemos que dar mas acorde a la comida, servicio recibido. Si variamos alguno de los elementos mencionados (entradas y reglas) el resultado final también cambiará.

2.2.2 ANFIS

Un sistema ANFIS (Adaptive Neuro-Fuzzy Inference System) [2-3] combina los beneficios de una red neuronal artificial y los de un sistema de inferencia difusa (FIS) en un

solo modelo. Este sistema se ha hecho muy popular en estos últimos años debido a unas buenas características tales como un aprendizaje rápido y preciso, capacidad para alojar datos y conocimientos existentes para un problema dado. Debido a sus múltiples virtudes, ANFIS puede considerarse como una red adaptativa que tiene las mismas funcionalidades que un sistema difuso.

Cuando intentamos convertir la forma de razonamiento del cerebro humano en un sistema difuso (reglas y funciones de pertenencia) no recibimos una respuesta tan exacta como desearíamos. El principal objetivo de ANFIS es optimizar los parámetros del sistema difuso (para obtener una respuesta mas exacta a nuestro problema) mediante el uso de un algoritmo de aprendizaje y un conjunto de entradas-salidas del que queremos que aprenda. En un sistema ANFIS, utilizando un conjunto de datos de entrada y salida, se construye un sistema de inferencia difusa en el que los parámetros de las funciones de pertenencia están ajustados utilizando el algoritmo de backpropagation (se puede utilizar este algoritmo sólo o combinándolo con el método de los mínimos cuadrados). Este ajuste permite al sistema difuso aprender del conjunto de datos que le estamos proporcionando.

El método de neuro-aprendizaje adaptativo funciona de manera similar al de una red neuronal. La técnica de aprendizaje neuro-adaptativa proporciona un procedimiento de modelado difuso para aprender información de un conjunto de datos. La forma de las funciones de pertenencia dependen de los parámetros, y el cambio de estos parámetros, cambia la forma de las funciones de pertenencia.

En algunas situaciones (por ejemplo si ya tenemos un conjunto de entrada y salida de datos), no podemos distinguir que número de funciones de pertenencia debemos de utilizar simplemente mirando los datos. En este tipo de casos debemos elegir los parámetros (por lo general se eligen arbitrariamente si no tenemos experiencia en este campo), para adaptar las

funciones de pertenencia al grupo de datos y poder percibir las variaciones que se produzcan en los valores de los datos resultantes.

Los parámetros asociados a las funciones de pertenencia cambian durante el proceso de aprendizaje. El cálculo de estas variaciones esta ayudado por un vector denominado vector gradiente. El vector gradiente se utiliza para conocer como de bien se están aproximando los resultados de aplicar ANFIS (valores resultantes de salida) con los valores reales de salida que tenemos. Una vez obtenido el vector gradiente, con el fin de ajustar los parámetros para minimizar el error cometido, se pueden aplicar varias rutinas de optimización.

El error normalmente esta delimitado por la suma de la diferencia de los cuadrados entre los datos de salida iniciales y los datos de salida que se obtienen por medio de ANFIS. Para que ANFIS funcione correctamente y no aumente el error cometido mas de lo normal, el conjunto de datos que le suministramos debe estar bien construido, y no tener errores anteriores, porque que si el conjunto de datos ya contiene errores, si le sumamos el error cometido por ANFIS aunque sea bajo, se puede llegar en algunos casos a no parecerse a los datos que le suministramos originalmente.

A la hora de poder comprobar que los datos obtenidos se asemejan con los datos iniciales que teníamos, tenemos que crearnos dos conjuntos de datos diferentes. El primer conjunto será el conjunto de datos de test y el segundo conjunto será el conjunto de datos de entrenamiento que le suministramos a ANFIS para que aprenda sobre nuestros datos. El conjunto de datos de test será el que vamos a utilizar posteriormente a la ejecución de ANFIS para comprobar que los resultados obtenidos por ANFIS (con el conjunto de entrenamiento) concuerdan con el conjunto de datos de entrenamiento.

Es importante que ambos conjuntos no tengan errores (por ejemplo por ruido que se ha metido a la hora de elaborar los datos) ya que si es afectado un grupo de los dos por algún tipo de error, va a ser imposible que ambos conjuntos concuerden.

ANFIS es mucho más complejo que un sistema difuso por lo que todas las opciones que presenta un sistema difuso no son compatibles con ANFIS. En concreto ANFIS solo admite los sistemas de tipo Sugeno y deben de tener las siguientes propiedades:

- Sistema Sugeno de primer orden.
- Tener una sola salida.
- Todas las funciones de pertenencia de salida deben de ser del mismo tipo y además ser lineales o constantes.
- No se pueden compartir las reglas. No pueden tener la misma función de pertenencia de salida dos reglas distintas (de lo que se deduce que el número de reglas es igual al número de funciones de pertenencia de salida).
- Tener un único peso para cada regla.

En la siguiente figura se detalla la estructura general de un sistema ANFIS (figura 2.10) y a continuación, se explica en detalle cada una de las capas que componen un sistema ANFIS [2].

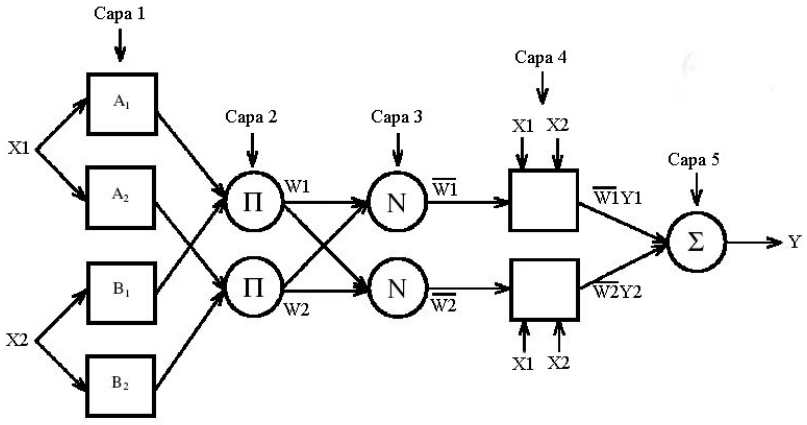


Figura 2.10. Estructura de un sistema ANFIS

De las cinco capas, dos de ellas contienen pesos ajustables (las de forma de cuadrado), las cuales van a ir variando para que ANFIS aprenda y minimice el error. Las

otras tres capas (con forma circular) son para realizar operaciones como productos y sumas. Las funcionalidades de las 5 capas son:

- Capa 1: Los nodos de esta capa implementan una regla de decisión difusa a través de una función de pertenencia. Las FP mas usuales son:

- Triangular

$$FP_T(x;a,b,c) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \tag{2.11}$$

- Campana generalizada

$$FP_C(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{2.12}$$

- Gaussiana

$$FP_G(x;c,\sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \tag{2.13}$$

En estos ejemplos, *a*, *b*, *c* y σ corresponden al conjunto de parámetros que varían la configuración de las funciones de pertenencia y se conocen como parámetros de premisa.

- Capa 2: Los nodos de esta capa (tienen el símbolo de producto) multiplican las entradas que reciben y envían el resultado a los nodos de las capas siguientes como se puede ver en la figura 2.10.
- Capa 3: Los nodos de esta capa (tienen como nomenclatura la letra N mayúscula) normalizan las funciones de pertenencia para actualizar las entradas.

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}, \qquad i = 1,2 \tag{2.14}$$

- Capa 4: Los nodos de esta capa asocian cada FP normalizada con una salida. Los parámetros $a_{0,i}$ $a_{1,i}$ y $a_{2,i}$ de esta capa son conocidos como parámetros de consecuencia y se van ajustando durante el proceso de entrenamiento.

  $\overline{w}_i y_i = \overline{w}_i (a_{2,i} x_1 + a_{1,i} x_2 + a_{0,i})$ siendo $y_i = a_{2,i} x_1 + a_{1,i} x_2 + a_{0,i}$ e $i = 1,2$ (2.15)

- Capa 5: Esta última capa realiza un sumatorio de todas las salidas de la capa 4 y así dar la salida final del programa (Y) que es un número real (se aprecia en la figura 2.10).

$$Y = \sum_i y_i \tag{2.16}$$

Mientras estamos en el proceso de aprendizaje, ANFIS va a ir ajustándose para ir en cada iteración asemejándose a la salida real y así minimizar el error (para esto se van ajustando los pesos de los parámetros de antecedencia y los parámetros de consecuencia). Para finalizar ANFIS, resumimos cuales son sus ventajas y desventajas generales:

- Ventajas

  - Para describir el comportamiento de un sistema complejo se depuran las reglas difusas SI-ENTONCES.

  - No requiere experiencia previa (utiliza los datos que le proporcionamos para aproximar el resultado).

  - Rápido tiempo de convergencia.

  - No hay problema con los nodos ocultos.

- Desventajas

  - Con un sistema ANFIS sólo podemos tener una salida.

  - La forma que tiene una función de pertenencia no cambia durante el entrenamiento (puede influir a la hora de elaborar modelos con precisión del problema en el que se esté ejecutando).

  - Solo se puede ejecutar una regla de aprendizaje.

2.2.3 MANFIS

Generalmente los problemas que se nos presentan tienen mas de una salida (los problemas más comunes tienen múltiples entradas y múltiples salidas). El principal inconveniente que presenta ANFIS es que posee una sola salida [8].

Para solventar este problema, vamos a utilizar en este PFC una variante de ANFIS llamada MANFIS (figura 2.11) que permite solucionar problemas con múltiples salidas (MANFIS es una extensión del sistema ANFIS). MANFIS ofrece las mismas propiedades que un sistema ANFIS pero con múltiples salidas. La estructura de MANFIS presenta varias ANFIS (a las cuales se le introducen las mismas entradas) en paralelo para obtener diferentes salidas. La interconexión de varios ANFIS para formar una estructura MANFIS se muestra en la figura 2.11.

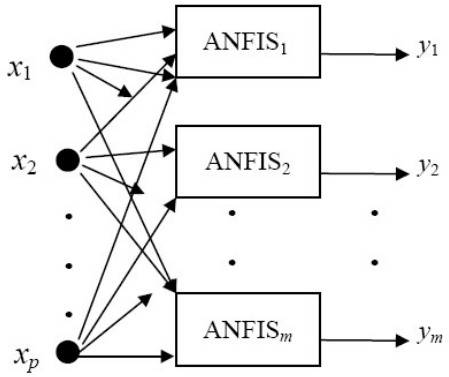


Figura 2.11. Estructura de MANFIS

2.3 Redes Neuronales

Resulta difícil pensar que los ordenadores actuales sean capaces de realizar cien millones de operaciones en coma flotante por segundo pero que sean totalmente incapaces de distinguir entre distintas clases de objetos. Son altamente eficientes para el manejo de bases de datos, procesamiento de textos, gráficos, comunicaciones electrónicas pero definitivamente no pueden ayudarnos a solventar problemas más cerca del mundo real (problemas que el cerebro humano realiza con suma facilidad).

Esta incapacidad para resolución de problemas simples (vistos desde el punto de vista del razonamiento humano) ha hecho que se realicen muchas investigaciones para ver como poder solventar problemas cotidianos a los que se enfrenta el ser humano. Estas investigaciones al final se han decantado por una unidad de proceso y almacenamiento de datos capaz de solucionar estos problemas y aprender de ellos. Esta unidad de proceso es el cerebro humano. El cerebro cuenta con unas buenas características para cualquier sistema de procesamiento, tales como:

- Es altamente paralelo (10^{11} elementos con 10^4 conexiones por elemento).
- Puede utilizar información difusa o inconsistente.
- Es muy robusto y tolerante a fallos.
- Es compacto y consume poca energía.
- Se ajusta a nuevos problemas por medio del aprendizaje (no se tiene que volver a programar el cerebro entero para esto).

El cerebro es un gran procesador de información, puede procesar casi instantáneamente una gran cantidad de información que le llega de distintas fuentes (los sentidos). También puede comparar estas informaciones con lo que tenia almacenado con anterioridad (para saber si la respuesta que tiene que dar ya la ha tenido que dar en otra

ocasión) e incluso utilizarlas a la vez para dar la respuesta mas adecuada (aunque sea una situación que nunca había visto).

A lo largo de la historia siempre ha habido un intento por entender e imitar el funcionamiento del cerebro (procesado de información, reconocimiento de patrones, almacenamiento de la información, etc.) y estos intentos han ido evolucionando conforme evolucionaba la tecnología para poder construir una red neuronal artificial [11]. Para hacerse una idea de cómo fue evolucionando la idea de las redes neuronales artificiales, tenemos que hacer un breve resumen de su historia:

- 1920-1930: Se intenta asemejar la conmutación cerebral con la conmutación telefónica de la época.
- 1943: Walter Pitts, Bertran Russell y Warren McCulloch intenta explicar el funcionamiento del cerebro humano como un dispositivo (tipo binario) con varias entradas y salidas. Seria un conjunto de células interconectadas entre si pero que solo se comunican mediante impulsos simples (del tipo binario). También definen la memoria como un conjunto de ondas que se repiten en un conjunto cerrado de neuronas.
- 1949: Donald O.Hebb introduce un nuevo concepto en el campo de las redes neuronales basado en investigaciones psicofisiológicas. Es la primera persona en decir que las redes neuronales pueden aprender. En base a sus investigaciones, propone que las conexiones entre dos neuronas se refuerzan si se activan repetidamente y que al aumentar esta conexión entre ellas hacia aumentar su conductividad.
- 1951: Marvin Minsky y Dean Edmons montan la primera “red neuronal” que consistía en 300 tubos de vacío en un piloto automático de un bombardero B-24. Se

trataba de imitar el cerebro de una rata con una red compuesta por 40 neuronas artificiales.

- 1959: Widrow publica una teoría (adaptación neuronal) que permite por primera ver utilizar las redes neuronales en un problema real, filtros adaptativos para eliminar ecos en las líneas telefónicas.
- 1962: Rosemblatt publica un proyecto para un identificador de patrones ópticos binarios y salida binaria mediante una red neuronal artificial denominada Perceptrón.
- 1969: Minsky y Papert sacan a la luz errores serios en el funcionamiento del Perceptrón (como no poder representar la función XOR) y supone un gran golpe en contra para estas teorías. Tras esto, muchos investigadores abandonan sus estudios sobre este campo.
- 1982: Hopfield publica un modelo que resulto de ayuda para los mecanismo de almacenamiento y recuperación de la memoria. Esto provoca un incremento en este tipo de investigaciones.
- 1984: Kohonen siguió investigando en este campo y introduce un procedimiento para conseguir que elementos que están juntos entre si físicamente aprendan a representar patrones de entradas similares (a estas redes se les llama redes de Kohonen).
- Se crea el grupo PDP (Parallel Distributed Processing) fundado por Rumelhart, McClelland & Hinton. Publican trabajos relacionados con el algoritmo de retropropagación (backpropagation) que soluciona los problemas que expusieron Minsky y Papert. Esto hace que se extienda en gran medida el campo de aplicación de los modelos de computación conexionistas (las redes neuronales).

En la actualidad, gracias a diversos grupos de investigación repartidos por todas las universidades del mundo, las redes neuronales han alcanzado una madurez muy elevada y se utilizan para todo tipo de aplicaciones.

Las redes neuronales constan de distintos elementos para su funcionamiento. El elemento principal es la neurona (su diagrama de funcionamiento se detalla en la figura 2.12) encapsulada en una unidad de proceso [10]. Cada unidad de proceso esta compuesta de los siguientes elementos:

- Función de Red (o propagación): Calcula el valor de entrada. Generalmente se utiliza un sumatorio de las entradas multiplicadas por el peso de las conexiones.
- Función de activación (o transferencia): Es la función que define el comportamiento de la neurona. Posteriormente se explicaran los principales tipos.
- Conexiones Ponderadas: El valor que tenga el peso sobre la entrada (signo y valor del peso) definen el tipo de influencia (excitación/inhibición).
- Salida: Se calcula por regla en general con el valor de la función de activación (aunque no siempre es así).

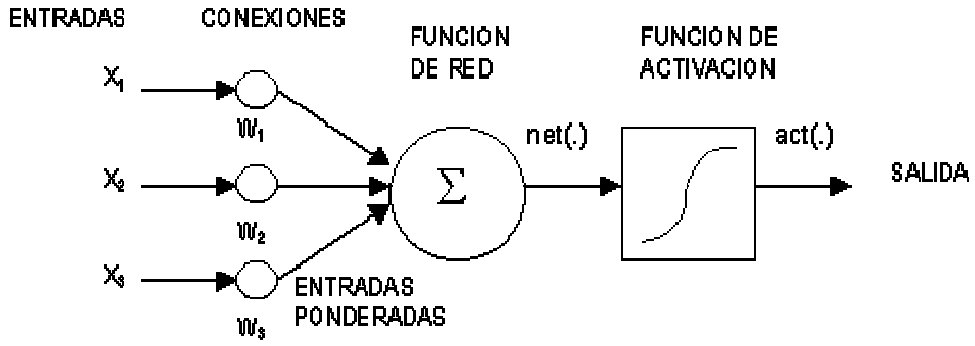


Figura 2.12. Unidad de proceso

Para saber cual va a ser el comportamiento de nuestra red neuronal, debemos de conocer y elegir cual va a ser la función de transferencia que vamos a elegir. Las 3 funciones de transferencia principales que podemos elegir son:

- Función Umbral (hardlim): Supone que las neuronas permanecen inactivas y solo se activan si la excitación total alcanzan un cierto valor umbral. Para modelar este caso se utiliza normalmente la función escalón y más concretamente el escalón unitario, en la cual la función devuelve 0 por debajo del valor umbral y 1 si lo supera. El grafico de esta función se detalla en la figura 2.13.

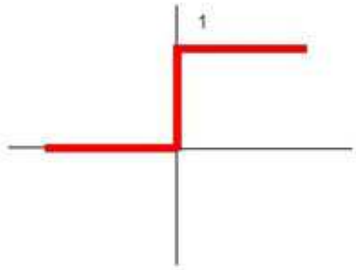


Figura 2.13. Función escalón unitario

También puede utilizarse una variación del escalón unitario en la que en lugar de estar comprendida entre 0 y 1 está entre -1 y 1 (figura 2.14). Esta función se denomina hardlims.

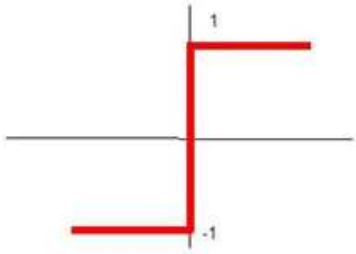


Figura 2.14. Variación de la función escalón

- Función Lineal (purelin): El valor de entrada es igual al valor de salida F(x)= x. La forma general es la presentada en la figura 2.15.

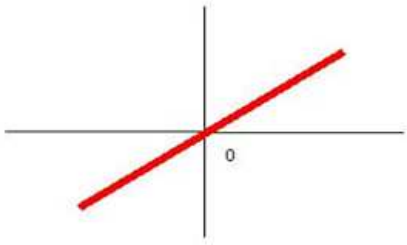


Figura 2.15. Función Lineal

Existe una variante en la que si el valor de la suma de las entradas esta dentro del rango establecido, se sigue usando la función F(x) = x, pero si la suma de las entradas es inferior o superior al rango, se va a utilizar el mismo valor. Como ejemplo aclaratorio véase la figura 2.16.

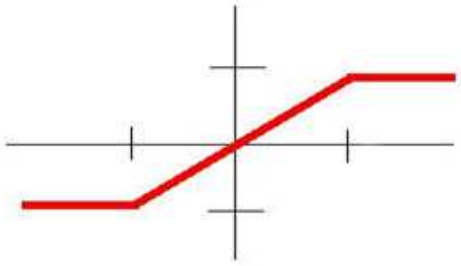


Figura 2.16. Función Mixta

- Función Sigmoidal (tansig): Es una función continua no lineal y la más utilizada en la actualidad. Esta acotada al rango [0,1] sea cual sea la entrada. Como se observa en la figura 2.17 si la entrada es 0 el valor de salida es 0,5. Esta función tiene la particularidad de no dar valores negativos. La expresión general para esta función es:

$$y_i(t+1) = \frac{1}{(1+e^{-(Net_i-\theta_i)})} \tag{2.17}$$

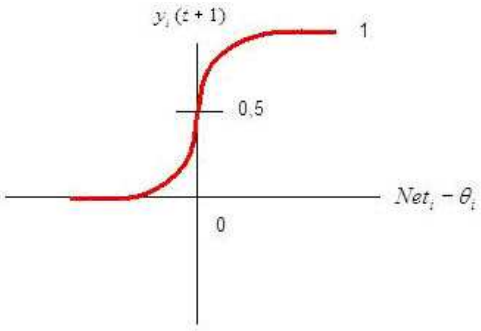


Figura 2.17. Función Sigmoidal

Los elementos que componen una red neuronal biológica y una red neuronal artificial tienen distinta nomenclatura. Para saber cómo se llaman los elementos en ambas redes, proponemos un resumen en la tabla 2.2.

Redes Neuronales Biológicas	**Redes Neuronales Artificiales**
Neuronas	Unidades de proceso
Conexiones sinápticas	Conexiones ponderadas
Efectividad de las sinápsis	Peso de las conexiones
Efecto excitatorio o inhibitorio de una conexión	Signo del peso de una conexión
Efecto combinado de las sinápsis	Función de propagación o de red
Activación -> tasa de disparo	Función de activación -> Salida

Tabla 2.2: Comparación entre red neuronal biológica y artificial

A la hora de diseñar una red neuronal, hay que tener en cuenta cuantas neuronas necesitamos para realizar correctamente nuestro programa y de que forma van a estar dispuestas. El criterio mas general para agrupar neuronas es el concepto de capa, en el que pueden estar dispuesta en una o varias capas.

En un principio el uso de las redes neuronales era el de agrupar todas las neuronas en una capa, pero en la actualidad lo mas usual es disponer de dos o tres capas de neuronas. La primera capa que pongamos será la capa de entrada que es la encargada de almacenar la información que obtiene por las entradas (como es la primera capa, es la que esta conectada a las entradas) y realizar un pre-procesado de la información. La otra capa mas usual es la capa de salida, que es la encargada de almacenar la respuesta de la red para poder ser posteriormente leída. Opcionalmente (si queremos que nuestro diseño este compuesto por mas capas) pueden existir un número de capas intermedias entre las dos capas explicadas anteriormente que se denominan capas ocultas. En estas capas se extrae, procesa y almacena la información para posteriormente enviarlo a la capa de salida. A continuación se puede ver en la figura 2.18 un esquema de una red neuronal con los 3 tipos de capas descritos anteriormente.

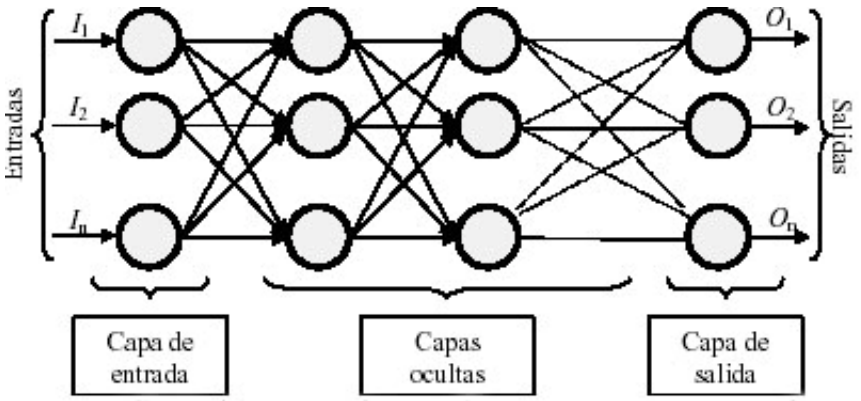


Figura 2.18. Esquema red neuronal

Para explicar un poco más el funcionamiento de una red neuronal, hay que entender como aprende una red neuronal. El proceso de aprendizaje consiste en la modificación de los pesos de las conexiones siguiendo alguna regla de aprendizaje (que se van a describir a continuación) para optimizar la respuesta. Los 3 tipos principales de aprendizaje que se utilizan en las redes neuronales son:

- Aprendizaje supervisado: En este tipo de aprendizaje (el más sencillo), se le suministra a la red un conjunto de entradas junto con otro conjunto de salidas que han producido estas entradas. Este aprendizaje va modificando los pesos de las conexiones con el fin de disminuir el error que se produce entre la salida que le hemos proporcionado y la salida calculada.
- Aprendizaje no supervisado: En este tipo de aprendizaje no se le presentan los patrones de salida deseados, sino que se le deja seguir alguna regla de auto-organización. Este proceso de aprendizaje consigue extraer ciertas propiedades estadísticas y agrupar en patrones. Estos patrones producen salidas consistentes.
- Aprendizaje reforzado: Sigue la filosofía del aprendizaje supervisado pero no le damos toda la información que suministrábamos en el aprendizaje supervisado. Lo

que se hace aquí con la supervisión es decirle si la salida que ha producido es correcta o no.

Las principales ventajas que ofrece una red neuronal (debido a su parecido con el cerebro humano) son las siguientes:

- Aprendizaje: Las red neuronal tiene la habilidad de aprender a partir de los datos que le damos y así obtener una salida similar a la que le hemos suministrado.
- Auto organización: La red neuronal crea su propia representación de la información (por lo que el usuario ya no tiene que hacer este trabajo).
- Tolerancia a fallos: Como la información en una red neuronal se almacena de forma redundante si fallara parcialmente la red podría seguir funcionando sin problemas.
- Flexibilidad: La red neuronal puede manejar cambios no importantes por ruido en los datos de entrada.
- Tiempo real: La red neuronal opera en paralelo, por lo que si se implementa en dispositivos adecuados, se pueden obtener respuestas en tiempo real.

Hasta ahora, no hay un criterio establecido para determinar la configuración de una red, se deja la elección a la experiencia del diseñador. A partir de esto hay que preguntarse que numero de neuronas hay que poner para las capas ocultas, tiene que ser suficiente para formar una región lo suficientemente compleja como para resolver el problema y también no demasiado grande porque la estimación de los pesos puede ser no confiable.

Existen diferentes modelos de Redes neuronales en función de la configuración que vamos a elegir para nuestra red (adaline, backpropagation, perceptron, memorias asociativas, etc.). Se va a explicar el funcionamiento de la red perceptron y posteriormente la que se emplea para el desarrollo de este PFC, el modelo **Backpropagation.**

2.3.1 Perceptron

El perceptron fue inventado por Frank Rosenblatt en 1957. El primer perceptron fue creado para imitar el funcionamiento del ojo humano (se le llamo fotoperceptron) y respondía a señales ópticas. Mediante las primeras investigaciones se pudo demostrar que el perceptron clasificaba patrones pero su precisión iba disminuyendo conforme se le pedía que aprendiera un número mayor de patrones. Una gran desventaja que presentaba el perceptron era la incapacidad de solucionar problemas que no sean linealmente separables.

Para explicar una red de tipo perceptron es recomendable empezar detallando como es una red perceptron simple (Figura 2.19) en la que se observa que esta formado por una única neurona [9]. Para el calculo de la salida se realiza la suma ponderada de las entradas y se resta el resultado al umbral y se pasa este resultado final a la función de transferencia, para esta red es de tipo hardlims (W_{1n} van a ser los pesos de la red).

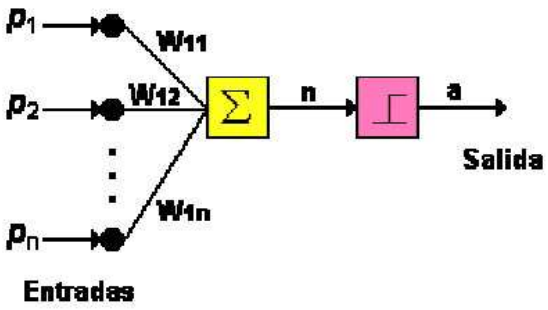
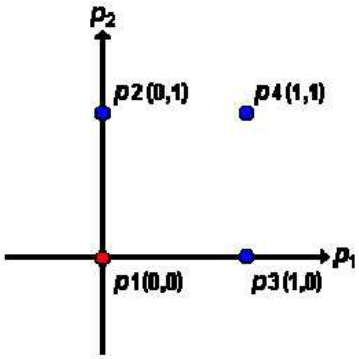


Figura 2.19. Perceptron simple

Generalmente una red de tipo perceptron emplea dos funciones de transferencia por regla general (ambas descritas anteriormente). La primera es la función hardlim con salidas [0,1] y la segunda es hardlims con salidas [-1,1]. Se suele utilizar más la función hardlims sobre hardlim. Es debido a que al tener en harlim algunos valores multiplicándose por el valor 0 hace que el aprendizaje sea mas lento (los pesos no se actualizan al multiplicarse por 0).

El tipo de aprendizaje que se utiliza en una red perceptron es el aprendizaje supervisado (descrito anteriormente) en el que se le suministran unos valores de entrada y las salidas deseadas y los pesos de la red se ajustan de forma que se obtengan una salida a la red lo más aproximadas a la salida real. Aparte de los pesos utilizados en la red, hay otro elemento que también se debe de ser ajustado para obtener una salida de la red mas aproximada a la real. Este elemento se llama ganancia y para entender el porque debe ser ajustado vamos a explicarlo mediante dos ejemplos [12].

Vamos a hacer que una red perceptron simple aprenda a diferenciar las entradas de una tabla OR. En la figura 2.20 se muestra la tabla de una función OR y como se deben de disponer los patrones.



p_1	p_2	a
0	0	0
0	1	1
1	0	1
1	1	1

Figura 2.20. Función OR

Utilizando la forma de expresión de la figura 2.19, si $w_1p_1 + w_2p_2$ es mayor que 0, la salida será 1, mientras que si no se cumple la salida será -1 (función hardlims). Durante el proceso de aprendizaje se varían los pesos para así obtener una recta que divida el plano en dos espacios para dos posibles valores de entrada. Para nuestro ejemplo de la función OR tenemos que separar los valores que tienen como salida 1 (01,10 y 11) del valor que tiene como salida -1 (00). En la figura 2.21 se observa que este problema de separación se puede resolver con la recta en el origen que divide ambos espacios. También se puede ver como seria la estructura de este ejemplo.

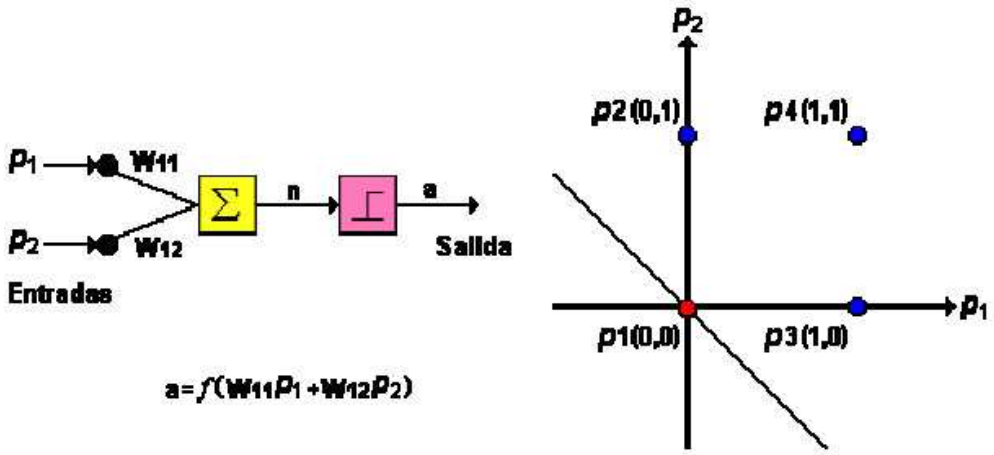


Figura 2.21. Perceptron aplicado a la función OR

Ahora vamos a intentar lo mismo que en el ejemplo anterior pero para que la red aprenda la función AND. Si queremos resolver este ejemplo y no tuviéramos el elemento ganancia seria imposible poder resolverlo (porque una recta que pase por el origen no es capaz de separar este ejemplo). La ganancia (se nombra por la letra b) es un elemento que permite desplazar la recta del origen de coordenadas para así poder solucionar problemas que no es posible resolverlos con la recta en el origen. Inicialmente se le establece su valor a 1 y se ajusta durante el aprendizaje (como los pesos W). La solución para este ejemplo utilizando las entradas que le introducimos y la ganancia esta representada en la figura 2.22.

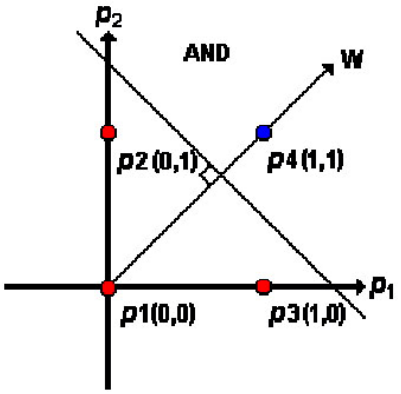


Figura 2.22. Solución a la función AND utilizando la ganancia

Como se ha podido comprender con los ejemplos anteriores, es necesario el uso de la ganancia para todos aquellos problemas en los que la recta que separa los espacios no puede estar en el origen de coordenadas. Podemos resumir el algoritmo de aprendizaje del perceptron siguiendo los siguientes puntos:

- Se establecen los valores por defecto de la ganancia y la matriz de pesos.
- Se presenta el primer patrón a la red: Los valores de entrada y salida que deseamos que aprenda.
- Se calcula la salida de la red realizando la suma ponderada de las entradas con sus pesos más la ganancia. La fórmula que lo expresa es (siendo *f* la función escalón unitario o su variación):

$$a = f(w_1 p_1 + w_2 p_2 + b) \tag{2.18}$$

- Si la salida es incorrecta hay que variar los pesos para que se asemeje al valor de la entrada. Lo que se hace normalmente es sumar *p* a *w* haciendo que el vector *w* apunte en la dirección de *p*, y así cuando se hayan realizado unas cuantas iteraciones *w* se aproximará a *p*.

Como hemos comentado anteriormente, la red perceptron es incapaz de resolver problemas que no sean linealmente separables. Para explicar porque no puede resolver este tipo de problemas vamos a poner como ejemplo una función que no puede resolverse. La función XOR, la cual vemos su tabla y esquema de funcionamiento (de la red perceptron) para este caso en la figura 2.23 (utiliza el mismo esquema que las funciones OR y AND):

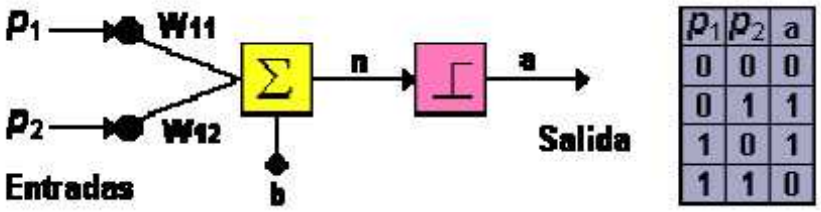


Figura 2.23. Tabla y esquema de función XOR

Queremos conseguir separar los dos pares de entradas [00,11] en un grupo y [01,10] en otro. Al llevar esta tabla al plano (figura 2.24) se puede observar perfectamente como con una red perceptron simple es imposible separar ambos grupos.

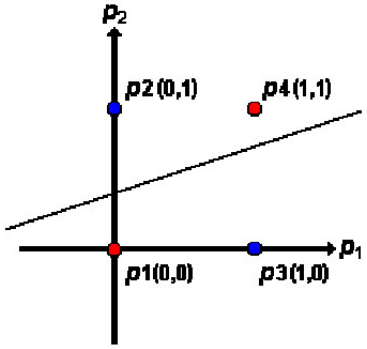


Figura 2.24. Plano de la función XOR

Como se observa, no hay ninguna línea posible que pueda separar el plano como deseamos. Esta limitación es muy importante a la hora de usar redes del tipo perceptron porque no puede abordar todos los problemas. Una solución para tratar de resolver este problema con redes perceptron seria utilizar una red perceptron multicapa.

2.3.1.2 Perceptron multicapa

Para resolver un problema que no puede ser solucionado por una red perceptron simple tenemos que descomponer el espacio en tres regiones y no en dos como se hace habitualmente (figura 2.25). Para llevar esto a la práctica, es necesario utilizar dos neuronas en vez de una como en el perceptron simple, así quedaría una región delimitada por las dos neuronas como se describe en la siguiente figura.

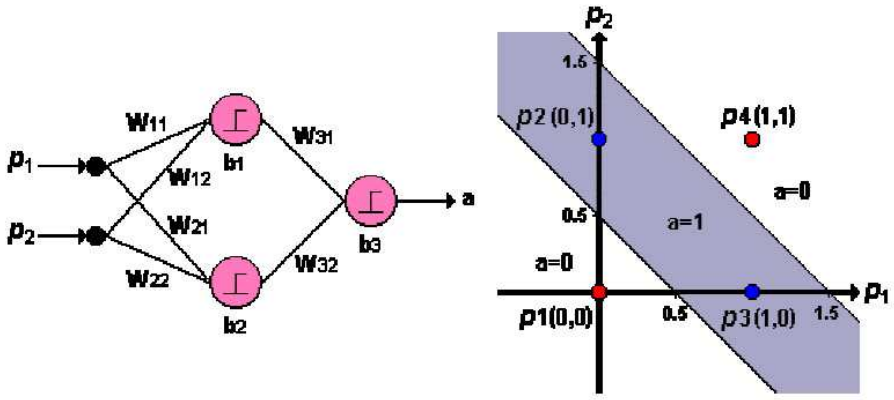


Figura 2.25. Perceptron con dos neuronas para la función XOR

Una multicapa perceptron permite resolver problemas más complejos que el perceptron simple, debido a que puede establecer regiones de decisión más complejas (como se ve en la figura 2.25) y no solo dividir el plano en dos espacios como hace el perceptron simple.

En la tabla 2.3 se puede observar la forma de separación de subespacios según el número de capas que tengamos en nuestra red.

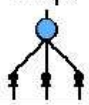
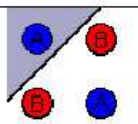
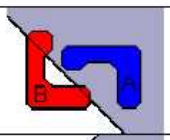
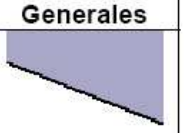
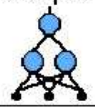
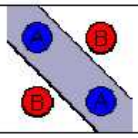
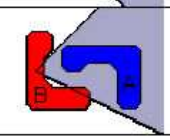
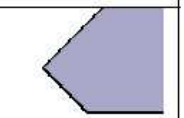
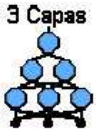
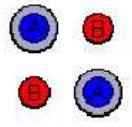
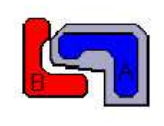
Estructura	Regiones de Decisión	Problema de la XOR	Clases con Regiones Mezcladas	Formas de Regiones más Generales
1 Capa 	Medio Plano Limitado por un Hiperplano			
2 Capas 	Regiones Cerradas o Convexas			
3 Capas 	Complejidad Arbitraria Limitada por el Número de Neuronas			

Tabla 2.3: Regiones delimitadas por un perceptron

2.3.2 Backpropagation

Como hemos podido apreciar en el apartado anterior, las redes de una sola capa solo pueden resolver problemas que sean linealmente separables. Esto motivó la investigación de redes multicapa que vino a solventar esta desventaja [9].

A mediados de los años 80, surge el algoritmo de propagación inversa (backpropagation). La publicación de este algoritmo incremento las investigaciones en el campo de las redes neuronales, convirtiendo el algoritmo de propagación inversa en una de las principales redes que se utilizan en la actualidad.

Un gran punto a favor de backpropagation es que aprovecha la característica de trabajo en paralelo de las redes neuronales, con lo que el tiempo de ejecución de un programa se ve reducido. Es un tipo de red con aprendizaje supervisado. En la primera capa, se recibe el patrón de entrada y este se propaga por el resto de capas para posteriormente dar un resultado en la capa de salida. Este resultado se comprueba con la salida deseada y se calcula el error cometido. A partir de este momento se produce una propagación inversa. La capa de salida es la que propaga ahora hacia atrás (hacia las capas ocultas) el resultado del error cometido.

Las neuronas de la capa oculta que reciben el error no lo reciben completo sino una porción del error total. Es decir, la parte aproximada en que haya contribuido cada neurona a la anterior salida. De este modo, el error se va propagando capa por capa (hacia atrás) hasta que todas las neuronas han recibido su parte, y según su contribución al error cometido, se recalculan los pesos de manera que se minimice el error en la siguiente iteración. Conforme se entrena la red, las neuronas de la capa intermedia aprenden a reconocer características de las entradas (patrones de entrada). La estructura general que presenta una red backpropagation de tres capas es la presentada en la figura 2.26 (equivale a 3 redes perceptron en cascada).

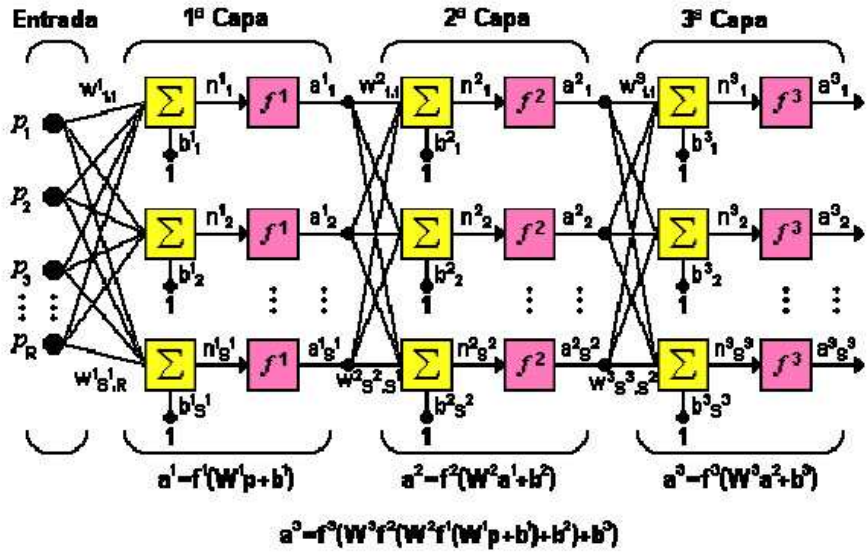


Figura 2.26. Red backpropagation con tres capas

Como se observa la salida de la primera capa es la que utiliza la segunda capa como entradas (y así sucesivamente). La ventaja de esta estructura es que cada capa puede disponer de un número de entradas distinto al de las otras y a su vez disponer también de distinta función de transferencia entre ellas. En la figura 2.26, la matriz de pesos para cada capa esta definido por W y el número que lleva como superíndice es la capa a la que pertenece esa matriz. R es el número total de entradas y el número de neuronas por capa es S (siendo el superíndice el número de capa al que se refiere). La letra b se refiere a la ganancia (siendo el superíndice el número de la capa a la que pertenece).

A continuación, vamos a explicar el funcionamiento interno de una red backpropagation mediante una red simple como la de la figura 2.27. Esta red posee una capa de entrada, una capa oculta y una capa de salida

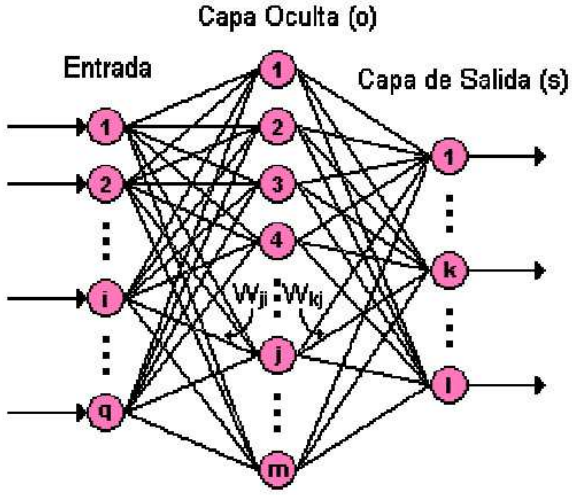


Figura 2.27. Red Backpropagation de 3 capas

La nomenclatura que vamos a utilizar en las expresiones es:

- *q*: Número de elementos del vector de entrada.
- *m*: Número de neuronas de la capa oculta.
- *l*: Número de neuronas de la capa de salida.
- *o*: Superíndice que se utiliza para referirse a la capa oculta.
- *s*: Superíndice que se utiliza para referirse a la capa de salida.
- *n*: Entrada neta.
- W^{o}_{ji}: Peso que une el elemento i de entrada con la neurona j de la capa oculta.
- b^{o}_{j}: Ganancia de la neurona j de la capa oculta.
- f^{0}: Función de transferencia de las neuronas de la capa oculta.
- *t*: Salida deseada (la salida que le introducimos junto con la entrada).
- ep^2: Error medio cuadrático para el patrón de entrada p.
- δ_k: Error en la neurona k.

Para empezar el proceso de aprendizaje, introducimos en la red un vector de entrada que dispone de *q* elementos (P_q) [10]. Este patrón se propaga por las conexiones produciendo una entrada neta *n* en cada una de las neuronas de la siguiente capa (la entrada neta es el

valor que resulta justo antes de pasar por la función de transferencia). El cálculo de la entrada neta a la neurona *j* de la siguiente capa viene definido por la fórmula (2.19).

$$n_j^o = \sum_{i=1}^{q} W_{ji}^o p_i + b_j^o \tag{2.19}$$

La salida de las neuronas de la capa oculta (la salida la definimos como *a*) se calcula por medio de la fórmula (2.20).

$$a_j^o = f^o(\sum_{i=1}^{q} W_{ji}^o p_i + b_j^o) \tag{2.20}$$

Como se ve en la formula anterior, la salida se calcula aplicando la función de transferencia al sumatorio del patrón de entrada por el valor del peso más el valor de la ganancia de la capa oculta. Ahora estas salidas sirven como entradas para los pesos de conexión de la siguiente capa que es la capa de salida ($a^o{}_k = n^s{}_k$). Las entradas netas de la capa de salida se calculan con la fórmula (2.21).

$$n_k^s = \sum_{j=1}^{m} W_{kj}^s a_j^o + b_k^s \tag{2.21}$$

La salida final de la red se calcula aplicando la función de transferencia de la capa de salida como se observa en la fórmula (2.22).

$$a_k^s = f^s(n_k^s) \tag{2.22}$$

Al final, la salida que hemos calculado con la red la comparamos con la salida deseada (t_k) de manera que así sabemos el error cometido en cada neurona. Para ellos se utiliza la relación (2.23).

$$\delta_k = (t_k - a_k^s) \tag{2.23}$$

El error cometido por cada patrón se calcula mediante la siguiente relación.

$$ep^2 = \frac{1}{2} \sum_{k=1}^{s} (\delta_k)^2 \tag{2.24}$$

Este proceso se repite para todos los elementos del vector de entrada. En resumen, para que el proceso de aprendizaje sea lo más ajustado posible a los valores que queremos que aprenda, lo que se debe de hacer es actualizar durante todo el proceso, los pesos y ganancias de las distintas capas para minimizar el error medio cuadrático cometido.

REFERENCIAS

[1] M. H. DeGroot, *Probability and Statistics,* 2nd ed. Reading, MA: Addison-Wesley, 1986.

[2] J-S, R. Jang: “ANFIS: Adaptative-Network-Based Fuzzy Inference System”, IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, May/June 1993, pp. 665-685.

[3] MATLAB. Fuzzy Logic Toolbox 2.

[4] L.A.Zadeh: “Fuzzy Sets”, Information and Control, vol. 8, 1965, pp. 338-353.

[5] Manuel Hernández Calviño, “Aclarando la lógica borrosa (fuzzy logic)”, Vol. 20, No. 2, 2003.

[6] Método de Mamdani: MATLAB “Fuzzy Inference Systems”.

[7] Método de Sugeno: MATLAB “Sugeno-Type Fuzzy Inference”.

[8] Chi-Bin Cheng: “Process optimization via a Neuro-Fuzzy System”

[9] MATLAB. Neural Network Toolbox.

[10] HILERA José R., MARTINEZ Víctor J. "Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones”. Ra-ma Editorial. Madrid. 1995

[11] Kevin Gurney, “An Introduction to Neural Networks”, 1-85728-503-4 June 1996

[12] Fernando Izaurieta y Carlos Saavedra, “Redes Neuronales Artificiales”

3. Técnicas de compresión de datos

En este apartado, vamos a presentar cuales van a ser las tres técnicas de compresión que hemos utilizado para comprimir las tablas de datos de nuestros dispositivos. Para poder aplicar estas técnicas correctamente, tenemos que decir cuales van a ser los dos modelos que vamos a utilizar con estas técnicas para optimizar su funcionamiento. El primer modelo se denomina modelo de aproximación, el cual es muy eficiente en tiempo de procesado pero contiene un error considerable (lo denominaremos modelo grueso "MG"). El segundo modelos que utilizaremos se denomina modelo fino ("MF"), el cual es muy fiable en cuanto al error cometido pero necesita mucho tiempo de procesado.

Lo que se está buscando en la actualidad es encontrar modelos híbridos que combinen los beneficios de ambos modelos, un modelo que sea tan fiable como un MF y que consuma poco tiempo de procesado (como un MG). Los modelos gruesos, generalmente tienen un rango limitado de sus parámetros y su simulación suele ser imprecisa. Los modelos finos suelen venir de medidas que se han hecho directamente sobre el dispositivo o mediante un simulador electromagnético (como es el caso en nuestro PFC). En este PFC, utilizaremos para todas las técnicas que vamos a presentar, como modelo fino los datos de entrada de unas tablas que hemos obtenido mediante un simulador electromagnético. Como modelo grueso utilizaremos los datos conseguidos mediante la aplicación de la regresión lineal sobre los datos de entrada. Para todas las técnicas descritas en este apartado, utilizaremos dos tablas de datos para dos dispositivos (en el apartado 4 se detallan los dispositivos de los cuales se sacan las tablas de datos). Para realizar el entrenamiento de las técnicas y posteriormente tener datos para comprobar si se generan correctamente, vamos a dividir las tablas de datos en dos grupos. El primer conjunto de datos se utilizará para el entrenamiento de la técnica y el segundo para comprobarlo con los datos reales y saber el error cometido.

3.1 Técnica clásica (TC)

La estructura de entrenamiento de la técnica clásica está representada en la figura 3.1 [1-2]. Consiste en proporcionarle un conjunto de entradas y salidas a los distintos métodos de aproximación (véase apartado 2 del PFC): Regresión lineal, MANFIS y Redes Neuronales. Obtenida la salida de esta técnica para cada método de aproximación, comprobaremos cual ha sido el error que hemos cometido.

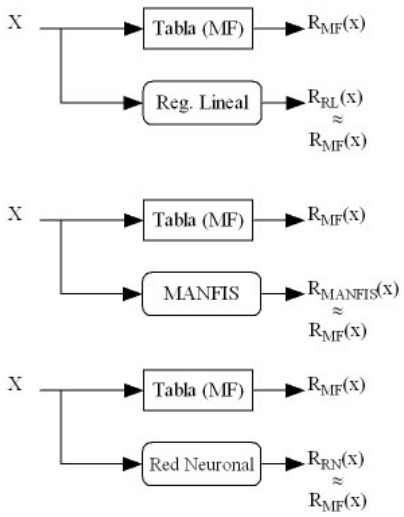


Figura 3.1. Estructura de entrenamiento en la técnica clásica

La figura 3.2 presenta la estructura de la técnica clásica una vez entrenados los distintos métodos.

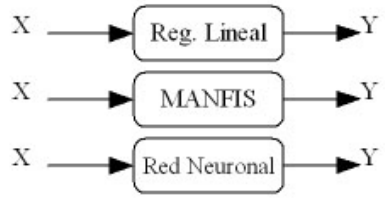


Figura 3.2. Estructura de test en la técnica clásica

Como se puede comprobar, esta técnica es simple en cuanto a estructura.

3.2 Técnicas avanzadas

3.2.1 Técnica de diferencia de fuente (TDF)

Con respecto a la técnica clásica, la técnica de diferencia de fuente fue desarrollada para utilizar menos cantidad de datos de entrenamiento [1-2]. La estructura general que utiliza la técnica TDF para su aprendizaje se muestra en la figura 3.3. Los tres elementos que integra esta estructura son:

- Modelo fino (tabla de datos obtenida mediante simulador electromagnético).
- Modelo grueso. Se obtiene mediante regresiones lineales con la técnica clásica.
- Red neuronal o MANFIS. Entrenadas para aproximar la diferencia entre el modelo fino y el modelo grueso.

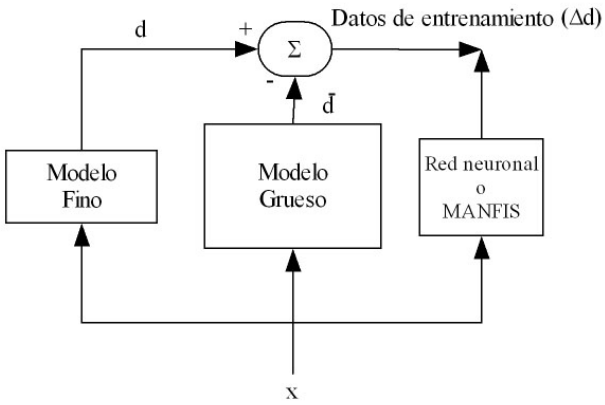


Figura 3.3. Estructura de entrenamiento para el método de la diferencia de fuente

El funcionamiento que rige la forma en que trabaja la técnica de diferencia de fuente para su aprendizaje es el siguiente:

- Inicialmente tenemos los valores de entrada (x_k) que nos proporciona el modelo fino (MF) y sus correspondientes salidas (d_k).
- Se calcula la salida $\overline{d}_k$ utilizando el modelo grueso (MG).

- La diferencia entre el valor de salida del simulador electromagnético y el modelo grueso se calcula para todos los elementos de entrada mediante la ecuación (3.1).

$$\Delta d_k = d_k - \bar{d}_k \tag{3.1}$$

- El conjunto de valores de entrada, salida ($x_k, \Delta d_k$) se utiliza para entrenar la red neuronal y MANFIS.

Al emplear Δd_k como la salida de la red neuronal o MANFIS, se produce un menor rango de salidas Δy en comparación con una relación normal de pares entrada-salida. El entrenamiento de la red neuronal requiere menos valores de la simulación electromagnética para obtener una buena precisión. Cuando finaliza el entrenamiento, el modelo neuronal puede predecir la diferencia entre valores del simulador electromagnético y el modelo de aproximación.

La figura 3.4 presenta la estructura de la técnica TDF una vez entrenados los distintos métodos de aproximación. El primer elemento es el modelo de aproximación, el cual aproxima el valor de salida real. El segundo elemento es la red neuronal o MANFIS (ya entrenadas) que se encargan de predecir la diferencia.

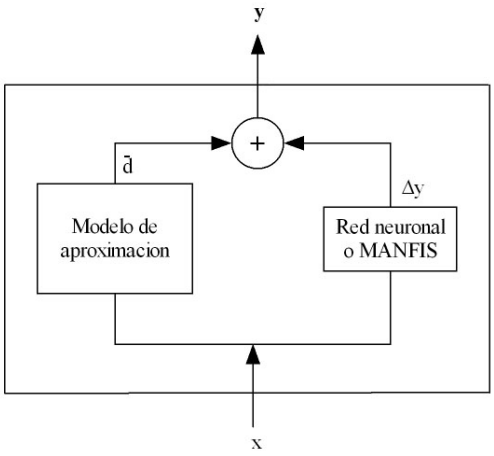


Figura 3.4. Estructura de test para el método de la diferencia de fuente.

Para adaptar esta técnica a nuestro PFC, hemos utilizado como modelo grueso la regresión lineal para ayudar a aprender al sistema MANFIS y la red neuronal [1-2]. La diferencia (que llamaremos ΔR) corresponde al cálculo entre los valores de salida que proporciona la regresión lineal y los datos de salida del modelo fino obtenidos a través del simulador electromagnético. Esta diferencia se utilizará como datos de salida para el sistema MANFIS y la red neuronal. La estructura que hemos diseñado para utilizar esta técnica con ambos métodos se detalla en la figura 3.5.

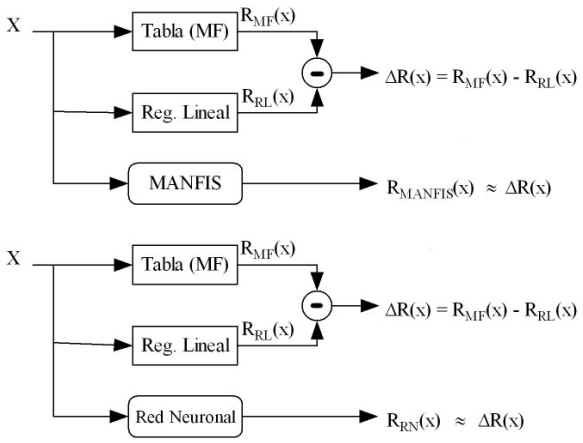


Figura 3.5. Estructura de entrenamiento de la técnica TDF para nuestro PFC

El modelo obtenido para cada estructura se detalla en la siguiente figura.

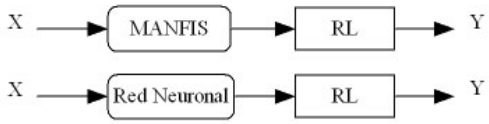


Figura 3.6. Estructura de test de la técnica TDF para nuestro PFC

La salida *Y* de esta técnica se aproxima a los resultados del modelo fino (FM) como se puede observar desarrollando la ecuación (3.2) y sustituyendo la salida de los métodos MANFIS y red neuronal por ΔR.

$$Y(x) = \Delta R(x) + R_{RL}(x) = R_{RL}(x) + R_{FM}(x) - R_{RL}(x) \approx R_{FM}(x) \quad (3.2)$$

3.2.2 Técnica con entrada de conocimiento previa (TECP)

Los métodos existentes, por lo general, pueden no ofrecer toda la precisión que nosotros deseamos que implemente nuestro modelo [1-2]. Esto puede suponer un gran inconveniente para algunos tipos de problemas en los que se exija la máxima precisión. La técnica con entrada de conocimiento previa (TECP) aporta una precisión mayor que el resto de métodos gracias a que utiliza un mayor número de entradas para aprender. La estructura general que utiliza la técnica TECP para su aprendizaje se muestra en la figura 3.7. Los tres elementos que integra esta estructura son:

- Modelo fino (tabla de datos obtenida mediante simulador electromagnético).
- Modelo grueso. Se obtiene mediante regresiones lineales con la técnica clásica.
- Red neuronal o MANFIS. Entrenadas para aproximar la diferencia entre el modelo fino y el modelo grueso.

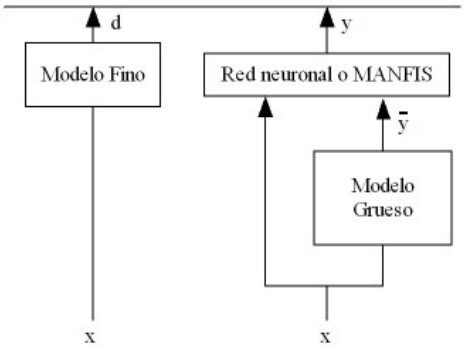


Figura 3.7. Estructura de entrenamiento para la técnica TECP

Las entradas extra que se introducen después de las entradas que ya se aportan al problema son las salidas calculadas a través del modelo grueso. Aparte de aprender como modelos anteriores de las entradas que le proporcionamos, al tener un número mayor de entradas en el sistema, se aprende de una manera más precisa que al no disponer de ellas. El funcionamiento que rige la forma en que trabaja la técnica TECP para su aprendizaje es el siguiente:

- Para cada valor de x_k que nos proporciona el simulador electromagnético, calculamos su salida $\overline{y}_k$ por medio del modelo de aproximación.
- La salida calculada $\overline{y}_k$ se utiliza junto con los datos de entrada suministrados por el simulador electromagnético $(x_k, \overline{y}_k)$ para proporcionarlos como entrada a la red neuronal y realice el aprendizaje para obtener la salida d_k.

La figura 3.8 presenta la estructura de la técnica TECP una vez entrenada el sistema MANFIS y la red neuronal. Los elementos que la componen son:

- Modelo grueso. Se obtiene mediante regresiones lineales con la técnica clásica.
- Método de aproximación mediante red neuronal o MANFIS.

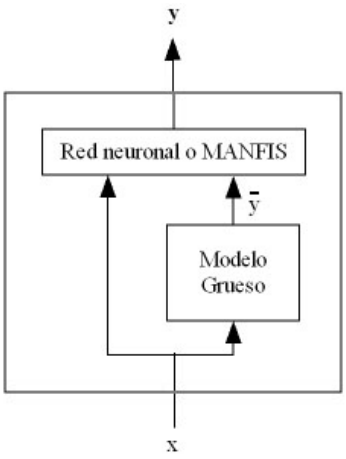


Figura 3.8. Estructura de test para técnica TECP

La evaluación de la red neuronal parte de los valores del modelo aproximado. El resultado global de esta evaluación será la respuesta del método (llegando a coincidir con la precisión del simulador electromagnético). La técnica TECP mantiene la velocidad del modelo grueso y los métodos de aproximación.

Para adaptar esta técnica a nuestro PFC [1-2], hemos utilizado como modelo grueso la regresión lineal para ayudar a aprender al sistema MANFIS y la red neuronal. Como modelo fino utilizaremos la tabla de datos proporcionada por el simulador electromagnético. La estructura que hemos diseñado para utilizar esta técnica con ambos métodos se detalla en la figura 3.9.

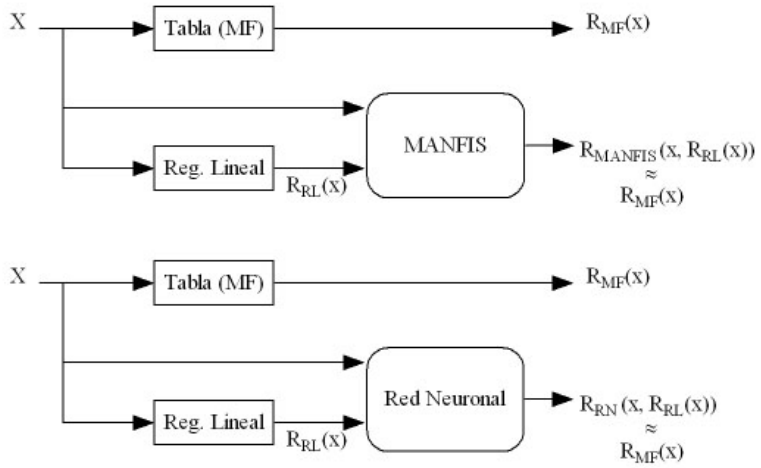


Figura 3.9. Estructura de entrenamiento de la técnica TECP para nuestro PFC

La estructura que se utiliza una vez entrenada la técnica, se muestra en la figura 3.10.

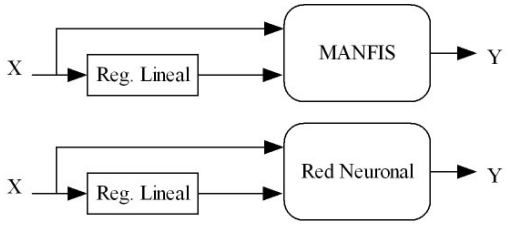


Figura 3.10. Estructura de test de la técnica TECP para nuestro PFC

Incluyendo las nuevas entradas que le proporcionamos a MANFIS (3.3) y a la red neuronal (3.4), las salidas que se calculen tienen que ser muy aproximadas a la salida real que nos proporciona la tabla de datos (modelo fino).

$$Y(x) = R_{MANFIS}(x, R_{RL}(x)) \approx R_{FM}(x) \tag{3.3}$$

$$Y(x) = R_{RN}(x, R_{RL}(x)) \approx R_{FM}(x) \tag{3.4}$$

REFERENCIAS

[1] Q.J.Zhang & K.C.Gupta: “Neural Networks for RF and Microwave design”, Artech House, 2000.

[2] Ginés Doménech-Asensi, Juan Hinojosa, Juan Martinez-Alajarín and Javier Garrigós-Guerrero: “Empirical Model Generation Techniques for Planar Microwave Components Using Electromagnetic Linear Regression Models”, IEEE Transactions on Microwave Theory and Techniques, vol. 53, no. 11, November 2005.

4. Ejemplos ilustrativos de dispositivos y formato de los datos

Como hemos visto en apartados anteriores, para poder comprimir datos mediante las técnicas descritas vamos a necesitar estos datos estructurados en tablas. Las tablas de datos se obtienen a partir de dos dispositivos que vamos a presentar en este apartado. Estos dispositivos se denominan Microstrip y Coplanar. A partir de estos dispositivos obtendremos dos tablas con una serie de entradas y salidas en columnas que utilizaremos para formar los conjuntos de entrenamiento y de test.

4.1 Dispositivo 1: Línea de transmisión de tipo Microstrip encapsulado

El primer dispositivo que vamos a utilizar se denomina Microstrip encapsulado. La estructura que forma este dispositivo se detalla en la figura 4.1. Por medio de este dispositivo, vamos a obtener una tabla con un conjunto de entradas y salidas.

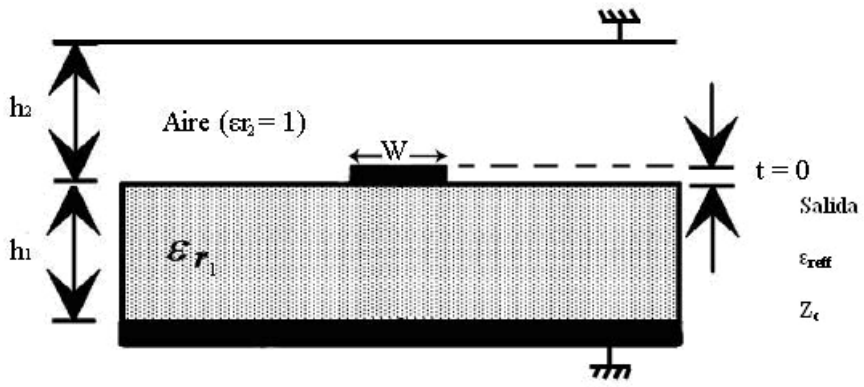


Figura 4.1. Estructura del dispositivo Microstrip

Los elementos que utilizaremos de esta tabla son:

- 4 Entradas: h_2 / h_1, W / h_1, f (frecuencia) y ε_r.
- 2 Salidas: ε_{reff} y Z_c.

Para entender mejor el dispositivo Microstrip, explicamos a continuación las variables de entradas y salidas que lo componen (figura 4.1).

- ε_{r1}: Permitividad relativa del dieléctrico 1.
- ε_{r2}: Permitividad relativa del dieléctrico 2 (aire $\varepsilon_{r2} = 1$).
- h_1: Grosor del dieléctrico 1.
- *h2*: Grosor del dieléctrico 2.
- *W*: Ancho de la tira conductora.
- *F*: Frecuencia.
- *t*: Espesor de metal (fijado a 0 μm).
- ε_{reff}: Permitividad relativa efectiva.
- Z_c: Impedancia característica.

4.2 Dispositivo 2: Línea de transmisión de tipo Coplanar encapsulado

El segundo dispositivo que vamos a utilizar se denomina Coplanar encapsulado. La estructura que forma este dispositivo se detalla en la figura 4.2. Por medio de este dispositivo, vamos a obtener una tabla con un conjunto de entradas y salidas.

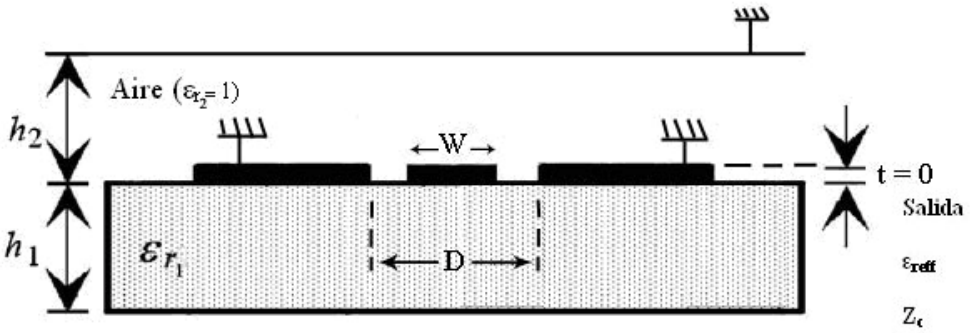


Figura 4.2. Estructura del dispositivo Coplanar

Los elementos que utilizaremos de esta tabla son:

- 4 Entradas: h_1 / D, h_2 / h_1, W / D y ε_r.
- 2 Salidas: ε_{reff} y Z_c.

Para entender el dispositivo Coplanar, explicamos a continuación las variables de entradas y salidas que lo componen (figura 4.2).

- ε_{r1}: Permitividad relativa del dieléctrico 1.
- ε_{r2}: Permitividad relativa del dieléctrico 2 (aire $\varepsilon_{r2} = 1$).
- h_1: Grosor del dieléctrico 1.
- *h2*: Grosor del dieléctrico 2.
- *f*: Frecuencia (fija *f* = 1GHz).
- *W*: Ancho del conductor central.
- *D*: Longitud de separación entre las dos masas.
- *t*: Espesor de los conductores (*t* = 0 μm).
- ε_{reff}: Permitividad relativa efectiva.
- Z_c: Impedancia característica.

4.3 Formato de los datos

En este apartado, vamos a detallar como se estructuran las tablas de los dos dispositivos que disponemos y cual va a ser el orden en que vamos a seleccionar las entradas y salidas para nuestro PFC. El formato de datos obtenido a través del dispositivo Microstrip encapsulado se detalla en la tabla 4.1. Las cuatro primeras columnas que se presentan en la tabla 4.1 se utilizarán como entradas y las columnas número cinco y seis se explotarán como salidas.

Las entradas vienen nombradas como X_i $(i = 1,2,3,4)$ y las salidas como Y_j $(j = 1,2)$.

Col.Nº 1	Col.Nº 2	Col.Nº 3	Col.Nº 4	Col.Nº 5	Col.Nº 6
h_2/h_1	W/h_1	f	ε_r	ε_{reff}	Z_c
X_1	X_2	X_3	X_4	Y_1	Y_2
·	·	·	·	·	·
·	·	·	·	·	·
·	·	·	·	·	·

Tabla 4.1: Disposición de los datos en la tabla Microstrip

El formato de datos obtenido a través del dispositivo Coplanar encapsulado se detalla en la tabla 4.2. Las columnas 1, 2, 3 y 5 se utilizarán como entradas y las columnas 6 y 7 se emplearan como salidas. La columna 4, relativa a la frecuencia, no se tendrá en cuenta puesto que es fija a 1 GHz para todos los datos de entrada. Las entradas vienen nombradas como X_i $(i = 1,2,3,4)$ y las salidas como Y_j $(j = 1,2)$.

Col.Nº 1	Col.Nº 2	Col.Nº 3	Col.Nº 4	Col.Nº 5	Col.Nº 6	Col.Nº 7
h_1/D	h_2/h_1	W/D	f	ε_r	ε_{reff}	Z_c
X_1	X_2	X_3	-	X_4	Y_1	Y_2
·	·	·	·	·	·	·
·	·	·	·	·	·	·
·	·	·	·	·	·	·

Tabla 4.2: Disposición de los datos en la tabla Coplanar

4.4 Índice de compresión

El objetivo de este PFC es la compresión de las tablas de datos descritas en el apartado anterior. Para poder valorar las prestaciones de las distintas técnicas de compresión de datos presentadas en el apartado 2, utilizaremos un elemento importante como el índice de compresión (IC). El índice de compresión es un término utilizado para cuantificar la reducción en el tamaño de una serie de datos producidos por un algoritmo de compresión de datos [1].

La compresión sin pérdidas de datos digitalizados (video, cine digital) conserva toda la información, pero rara vez se puede hacer una compresión mayor a 2:1. Sin embargo, con la compresión con pérdida de datos (por ejemplo MP3, JPEG), se puede lograr un índice de compresión mucho mayor a costa de una disminución de la calidad. La ecuación que utilizaremos para representar el índice de compresión se muestra a continuación:

$$IC = \frac{Número\ de\ puntos\ de\ datos}{Número\ de\ parámetros\ utilizados\ en\ las\ tecnicas} \tag{4.1}$$

En el siguiente apartado del PFC, incluiremos a las tablas de resultados de cada técnica, una columna adicional en la que se muestra el valor del índice de compresión obtenido (el número de parámetros utilizados en las técnicas también se calcula en el siguiente apartado).

REFERENCIAS

[1] W. R. Anis Ibrahim and Medhat M.Morcos: “Novel Data Compression Technique for Power Waveforms Using Adaptative Fuzzy Logic”, IEEE Transactions on power delivery, vol.20, no.3, July 2005.

5. Resultados

En este apartado vamos a presentar las tablas de datos de las simulaciones realizadas con las técnicas descritas en este PFC para nuestros dos dispositivos (Microstrip y Coplanar). Antes de exponer las tablas con las distintas simulaciones de las técnicas, debemos de saber como varían las entradas de los dispositivos en la tabla y cual es su valor de paso (cantidad en la que cada entrada va a ir variando su valor de un dato a otro) para conocer el número total de datos de salida que vamos a tener.

5.1 Intervalos de Variación

Los datos de las tablas del modelo fino para cada dispositivo se han obtenido a partir de un simulador electromagnético (Spectral Domain Approach [1]). Los intervalos de variación de las variables de entrada del dispositivo Microstrip encapsulado se detallan en la tabla 5.1.

Parámetro	Entrada X_i	Valor Mínimo	Valor Máximo	Valor de paso
h_2/h_1	X_1	1	3	1
W/ h_1	X_2	0,3	3	0,1 y 1
ε_r	X_3	8	15	1
f	X_4	1 GHz	40 GHz	1

Tabla 5.1: Intervalos de variación para el dispositivo Microstrip

Los intervalos de variación de las variables de entrada del dispositivo Coplanar encapsulado se detallan en la tabla 5.2. Para este dispositivo, no hemos considerado la frecuencia como variable debido a que es fija: f= 1 GHz.

Parámetro	Entrada X_i	Valor Mínimo	Valor Máximo	Valor de paso
h_1/D	X_1	1	4	1
h_2/ h_1	X_2	1	4	1
W/D	X_3	0,1	0,9	0,1
ε_r	X_4	10	70	1

Tabla 5.2: Intervalos de variación para el dispositivo Coplanar

Basándonos en las variaciones de los datos de entradas y su valor de paso, calculamos el número total de datos de salida para cada dispositivo. El número total de datos de salida para los dos dispositivos son:

- *Microstrip*: $3x10x8x40 = 9600\ datos\ de\ salida$

- *Coplanar*: $4x4x9x61 = 8784\ datos\ de\ salida$

5.2 Resultados de las técnicas

A continuación, vamos a exponer mediante tablas los errores relativos cometidos, tiempo de ejecución, parámetros e índice de compresión para las distintas técnicas de compresión utilizadas en este PFC.

5.2.1 Técnica clásica

5.2.1.1 Técnica clásica con regresión lineal

- Técnica clásica con regresión lineal aplicado al dispositivo Microstrip encapsulado

Ereff	**Ereff**	**Zc**	**Zc**	**Tpo.Ejecución**	**Parámetros**	**IC**
Error Med.	**Error Máx.**	**Error Med.**	**Error Máx.**			
0,9130%	12,9197%	3,6181%	19,5834%	11 seg	30	320

Tabla 5.3: Resultados de la Técnica clásica con Microstrip

- Técnica clásica con regresión lineal aplicado al dispositivo Coplanar encapsulado

Ereff	Ereff	Zc	Zc	Tpo.Ejecución	Parámetros	IC
Error med.	**Error máx.**	**Error med.**	**Error máx.**			
0,4522%	3,7960%	5,0108%	49,4011%	9 seg	30	292,80

Tabla 5.4: Resultados de la Técnica clásica con Coplanar

Esta técnica destaca por su mínimo tiempo de ejecución (el menor de todas las técnicas del PFC) y su alto índice de compresión (también es el más alto porque sólo utiliza 30 parámetros). Pero pese a estas características no es recomendable para utilizarse como técnica para la compresión de las tablas de datos de nuestro PFC, debido a su alto porcentaje de error (muy baja precisión). Unos errores tan altos como los que presentan ambos dispositivos, no son recomendables porque no se aproximan a los resultados reales (por lo que a la hora de recuperar la tabla de datos no se parecería a la original).

5.2.1.2 Técnica clásica con MANFIS

Para la implementación de esta técnica, hemos utilizado un sistema MANFIS compuesto por dos ANFIS (un ANFIS para cada salida). Para poder observar la capacidad de aprendizaje de MANFIS (ver como se aproxima el resultado calculado al resultado real) con la técnica clásica, hemos variado el número de funciones de pertenencia (FP) y el número de iteraciones. En resumen, las simulaciones que vamos a realizar con MANFIS para los dos dispositivos son:

- 4 funciones de pertenencia con 30 y 3 iteraciones.

- 3 funciones de pertenencia con 30 y 3 iteraciones.

- 2 funciones de pertenencia con 30 y 3 iteraciones.

- Técnica clásica con MANFIS aplicado al dispositivo Microstrip encapsulado

Número	Número	Ereff	Ereff	Zc	Zc	Tiempo	Parám	Parám	Parámetros	Compresión
MFs	Epoch	Error med	Error máx	Error med	Error máx	Ejecución	Lineales	No lineales	Totales	Índice
4	3	0,0415%	0,3514%	0,0755%	1,6790%	46m 30s	1280	32	1312	7,32
4	30	0,0272%	0,3471%	0,0533%	1,7858%	7h48m33s	1280	32	1312	7,32
3	3	0,0800%	0,5518%	0,2340%	1,8083%	4m 21s	405	24	429	22,38
3	30	0,0630%	0,4616%	0,1114%	1,7070%	46m38s	405	24	429	22,38
2	3	0,3401%	1.8461%	1.3077%	4,5602%	10s	80	16	96	100
2	30	0,2380%	1,3490%	0,8712%	3,0664%	1m 26s	80	16	96	100

Tabla 5.5: Resultados de la técnica clásica utilizando Microstrip y MANFIS

- Técnica clásica con MANFIS aplicado al dispositivo Coplanar encapsulado

Número MFs	Número Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Parám Lineales	Parám No lineales	Parámetros Totales	Índice Compresión
4	3	0,0011%	0,0254%	0,4963%	2,0077%	41m 21s	1280	32	1312	6,70
4	30	0,0004%	0,0252%	0,4634%	1,8918%	6h 54m 59s	1280	32	1312	6,70
3	3	0,0053%	0,0403%	0,8926%	3,4975%	3m 56s	405	24	429	20,48
3	30	0,0022%	0,0257%	0,7220%	2,6669%	39m 16s	405	24	429	20,48
2	3	0,1031%	0,8681%	2,0969%	8,0717%	9s	80	16	96	91,50
2	30	0,0172%	0,1622%	1,9459%	6,8186%	1m 19s	80	16	96	91,50

Tabla 5.6: Resultados de la técnica clásica utilizando Coplanar y MANFIS

A continuación, vamos a explicar los resultados obtenidos con esta técnica utilizando el modelo de aproximación MANFIS con los dispositivos Microstrip y Coplanar. Observando los resultados obtenidos en precisión, tiempo de ejecución e índice de compresión, vamos a proponer la configuración óptima para cada tabla.

- Técnica clásica utilizando MANFIS y dispositivo Microstrip encapsulado (Tabla 5.5):

Utilizando 4 funciones de pertenencia y 30 iteraciones, obtenemos la mejor precisión posible (en media) para esta técnica. Como puntos opuestos a esta configuración, tenemos un tiempo de ejecución excesivo (más de 7 horas) y un índice de compresión mas bajo. Por otro lado, si utilizamos 2 funciones de pertenencia y 30 iteraciones, obtenemos el mejor índice de compresión y el mejor tiempo de ejecución, pero contiene unos porcentajes de errores demasiados elevados y no permisibles para tenerlo en cuenta. La mejor opción a elegir para esta técnica, teniendo en cuenta las tres variables importantes, es utilizar **3 funciones de pertenencia y 30 iteraciones.** Con esta configuración obtenemos el mínimo error máximo de Zc de todas las configuraciones y el resto de errores son semejantes a utilizar mayor número de funciones de pertenencia. Además obtenemos un tiempo de ejecución normal (menos de una hora) y un índice de compresión tres veces mayor que usando 4 funciones de pertenencia (el índice de compresión es 22).

- Técnica clásica utilizando MANFIS y dispositivo Coplanar encapsulado (Tabla 5.6):

Utilizando 2 funciones de pertenencia (ya sea con 3 o 30 iteraciones), obtenemos un gran índice de compresión y un tiempo mínimo de ejecución., pero utilizando esta opción, se cometen grandes errores (muy baja precisión), por lo que no nos sirve para posteriormente recuperar datos y que sean semejantes a los originales. Si utilizamos 3 funciones de

pertenencia, con ambos números de iteraciones que antes, aumentamos la precisión con respecto al uso de 2 funciones de pertenencia. Sin embargo este aumento de precisión no es suficiente como para poderlo admitir (el error máximo en Zc sigue siendo muy alto). La mejor opción a elegir para esta técnica, teniendo en cuenta las tres variables importantes, es utilizar **4 funciones de pertenencia y 3 iteraciones.** Con esta configuración obtenemos error máximo de Zc bajo y el resto de errores son también bajos. Además reducimos el elevado tiempo de ejecución que se obtendría si escogiéramos 30 iteraciones. El uso de esta técnica correctamente nos proporciona un bajo índice de compresión (6).

5.2.1.3 Técnica clásica con red neuronal

Para implementar esta técnica, hemos utilizado una red neuronal de tipo backpropagation (véase apartado 2) con dos salidas. Para poder observar la capacidad de aprendizaje de la red neuronal con la técnica clásica, hemos variado el número de iteraciones, el número de capas y el número de neuronas en cada capa. Las simulaciones que vamos a realizar con la red neuronal para los dos dispositivos son para:

- Una capa de 50 neuronas.
- Dos capas de 20 neuronas cada una.
- Tres capas de 20 neuronas cada una.

En cada modalidad, vamos a variar el número de iteraciones (epoch) entre 100, 1000 y 2000. El tipo de entrenamiento será de tipo aprendizaje supervisado, en el que le vamos a pasar a la red un conjunto de entradas y sus correspondientes salidas. Como funciones de transferencia, vamos a utilizar la función sigmoidal en la primera capa y la función lineal en el resto.

Tabla 5.7: Resultados de la técnica clásica utilizando Microstrip y Red Neuronal

Núm. Capas	Neur Capa	Número Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Número Pesos	Número Ganancias	Parám Totales	Índice compresión
1	**50**	**100**	0,3037%	2,5739%	0,0790%	1,6727%	2m 24s	(50x4)	50x1	250	38,40
1	**50**	**1000**	0,1740%	1,7138%	0,0347%	1,7481%	23m 17s	(50x4)	50x1	250	38,40
1	**50**	**2000**	0,1926%	2,2110%	0,0533%	1,8053%	46m 29s	(50x4)	50x1	250	38,40
2	**20**	**100**	0,6478%	8,8365%	0,1424%	1,6493%	4m 17s	(20x4)+(20x20)	20x2	520	18,46
2	**20**	**1000**	0,4926%	6,2980%	0,1295%	1,7554%	42m 25s	(20x4)+(20x20)	20x2	520	18,46
2	**20**	**2000**	0,4307%	5,5601%	0,1430%	1,6639%	1h 24m 39s	(20x4)+(20x20)	20x2	520	18,46
3	**20**	**100**	0,6005%	4,3419%	0,2148%	1,9257%	10m 6s	(20x4)+(20x20)+(20x20)	20x3	940	10,21
3	**20**	**1000**	0,2673%	4,6791%	0,0706%	1,6763%	1h 40m 12s	(20x4)+(20x20)+(20x20)	20x3	940	10,21
3	**20**	**2000**	0,3687%	3,3922%	0,1064%	1,6847%	3h 15m 14s	(20x4)+(20x20)+(20x20)	20x3	940	10,21

-

Tabla 5.8: Resultados de la técnica clásica utilizando Coplanar y Red Neuronal

Número Capas	Neur Capa	Número Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Número Pesos	Número Ganancias	Parám Totales	Índice compresión
1	**50**	**100**	0,3148%	5,1903%	0,2748%	2,1655%	2m 7s	(50x4)	50x1	250	35,14
1	**50**	**1000**	0,0247%	0,6523%	0,01478%	0,7375%	21m 27s	(50x4)	50x1	250	35,14
1	**50**	**2000**	0,0090%	0,1388%	0,0079%	0,7283%	42m 34s	(50x4)	50x1	250	35,14
2	**20**	**100**	0,4276%	4,4485%	0,5428%	2,9834%	4m 1s	(20x4)+(20x20)	20x2	520	16,89
2	**20**	**1000**	0,0719%	1,7075%	0,0538%	0,7077%	39m 23s	(20x4)+(20x20)	20x2	520	16,89
2	**20**	**2000**	0,0731%	1,4072%	0,0577%	0,7528%	1h 19m 18s	(20x4)+(20x20)	20x2	520	16,89
3	**20**	**100**	0,2646%	4,5802%	0,2022%	1,1382%	8m 50s	(20x4)+(20x20)+(20x20)	20x3	940	9,34
3	**20**	**1000**	0,1078%	1,9592%	0,1082%	0,8797%	1h 26m 45s	(20x4)+(20x20)+(20x20)	20x3	940	9,34
3	**20**	**2000**	0,0698%	1,7742%	0,0472%	0,7445%	2h 55m 5s	(20x4)+(20x20)+(20x20)	20x3	940	9,34

A continuación, vamos a explicar los resultados obtenidos con esta técnica utilizando el modelo de aproximación Red Neuronal con los dispositivos Microstrip y Coplanar. Observando los resultados obtenidos en precisión, tiempo de ejecución e índice de compresión, vamos a proponer la configuración óptima para cada tabla.

- Técnica clásica utilizando Red Neuronal y dispositivo Microstrip encapsulado (Tabla 5.7):

Si observamos los resultados de precisión obtenidos en esta tabla, podemos apreciar que no es recomendable utilizar para este dispositivo 2 o 3 capas de neuronas. Si utilizamos estas configuraciones, la precisión obtenida en media es mas baja, porque se sobre-entrena la red neuronal y se producen errores mayores a los normales. Para obtener la mejor configuración para esta tabla, debemos de fijarnos en las configuraciones con **1 capa con 50 neuronas** y ver con cuantas iteraciones se produce una mejor precisión. Como queremos obtener el mejor compromiso posible entre las variables que evaluamos en este PFC (precisión, tiempo de simulación e índice de compresión), la mejor opción es utilizar **1000 iteraciones**, puesto que esta configuración tiene el mejor índice de compresión (38) y una precisión mejor que con el resto de iteraciones. Además el tiempo de ejecución (23 minutos) es bueno con respecto a las otras configuraciones con mayor número de capas.

- Técnica clásica utilizando Red Neuronal y dispositivo Coplanar encapsulado (Tabla 5.8):

Los resultados muestran que todas las opciones presentan un buena precisión (todos los errores están por debajo del 1%). Buscaremos la mejor configuración que presente un buen tiempo de procesado y un índice de compresión alto. Si analizamos las posibles configuraciones optimas que presenta esta técnica, es fácil decantarnos por la configuración

con **1 capa de 50 neuronas.** Esta configuración es sin duda la más recomendable, debido a que tiene las mejores precisiones de la tabla y el mejor índice de compresión posibles para esta técnica (35). Ahora, tenemos que elegir entre utilizar 1000 o 2000 iteraciones con esta técnica. Si escogemos la opción de **2000 iteraciones**, la precisión es mejor que aquella con 1000 iteraciones, pero sin embargo el tiempo de ejecución (42 minutos) es el doble que para la configuración con 1000 iteraciones.

5.2.2 Técnica de diferencia de fuente (TDF)

5.2.2.1 Técnica de diferencia de fuente con regresión lineal y MANFIS

- Técnica de diferencia de fuente utilizando Microstrip encapsulado

Número	MFs	Número Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Parám Lineales	Parám No lineales	Parámetros Totales	Índice Compresión
2	4	30	0,1954%	1,9739%	0,3576%	2,2450%	1m 41s	80	16	96+30(RL)	76,19
2	4	3	0,3246%	1,8531%	1,0634%	3.6017%	20s	80	16	96+30(RL)	76,19
3	4	30	0,0662%	0,5071%	0,1068%	1,7001%	43m 55s	405	24	429+30(RL)	20,92
3	4	3	0,0826%	0,5620%	0,2220%	1.6976%	4m 35s	405	24	429+30(RL)	20,92
4	4	30	0,0301%	0,3542%	0,0491%	1,7666%	7h42m46s	1280	32	1312+30(RL)	7,15
4	4	3	0,0430%	0,3570%	0,0712%	1.6687%	48m 2s	1280	32	1312+30(RL)	7,15

Tabla 5.9: Resultados de la técnica TDF utilizando Microstrip y MANFIS

- Técnica de diferencia de fuente utilizando Coplanar encapsulado

Número	Número	Ereff	Ereff	Zc	Zc	Tiempo	Parám	Parám	Parámetros	Índice
MFs	Epoch	Error med	Error máx	Error med	Error máx	Ejecución	Lineales	No lineales	Totales	Compresión
4	3	0,0010%	0,0252%	0,2896%	1,8427%	41m 17s	1280	32	1312+30(RL)	6,55
3	3	0,0058%	0,0616%	0,0546%	2,8035%	4m 2s	405	24	429+30(RL)	19,14
4	30	0,0003%	0,0249%	0,2378%	1,5757%	6h 56m 58s	1280	32	1312+30(RL)	6,55
3	30	0,0021%	0,0257%	0,4718%	2,5371%	40m 7s	405	24	429+30(RL)	19,14
2	3	0,1039%	0,8805%	1,2828%	7,0006%	20s	80	16	96+30(RL)	69,71
2	30	0,0453%	0,5999%	1,2223%	5,2650%	1m 41s	80	16	96+30(RL)	69,71

Tabla 5.10: Resultados de la técnica TDF utilizando Coplanar y MANFIS

- Técnica TDF utilizando MANFIS y dispositivo Microstrip encapsulado (Tabla 5.9):

Utilizando 4 funciones de pertenencia, podemos escoger entre utilizar 3 iteraciones o 30 iteraciones. Si observamos los resultados obtenidos en estas configuraciones, podemos ver que no es recomendable utilizar 30 iteraciones, porque en ambas obtenemos los mismos índices de compresión y una precisión muy semejante. La diferencia estriba en que utilizando 30 iteraciones tenemos un tiempo de ejecución muy elevado (casi ocho horas) en comparación con 3 iteraciones (menos de una hora). Esta opción es recomendable si anteponemos la precisión en cuanto a los errores cometidos al índice de compresión (es el mas bajo de la técnica). Si utilizamos 2 funciones de pertenencia (ya sea con 3 o 30 iteraciones) obtenemos un gran índice de compresión y un mínimo tiempo de ejecución, pero los errores que se cometen no son admisibles para posteriormente recuperar correctamente la tabla de datos. La mejor opción a elegir para esta técnica, teniendo en cuenta las tres variables importantes, es utilizar **3 funciones de pertenencia y 30 iteraciones.** Con esta configuración obtenemos una precisión semejante a utilizar un mayor número de funciones de pertenencia (el error que se comete es admisible para posteriormente poder recuperar la tabla). Además obtenemos un tiempo de ejecución normal (menos de una hora) y un índice de compresión aproximadamente tres veces mayor que utilizando 4 funciones de pertenencia (el índice de compresión es 20).

- Técnica TDF utilizando MANFIS y dispositivo Coplanar encapsulado (Tabla 5.10):

Utilizando 2 funciones de pertenencia (con 3 o 30 iteraciones) obtenemos un buen índice de compresión y un tiempo mínimo de ejecución, pero se cometen grandes errores por lo que no nos sirven para posteriormente recuperar datos. Si utilizamos 3 funciones de pertenencia, indiferentemente de utilizar 3 o 30 iteraciones, aumentamos la precisión con respecto al uso de 2 funciones de pertenencia. Sin embargo este aumento de precisión no es

suficiente como para poderlo admitirlo. La mejor opción a elegir para esta técnica, teniendo en cuenta las tres variables importantes, es utilizar **4 funciones de pertenencia y 30 iteraciones.** Debemos de utilizar esta configuración para conseguir una buena precisión (así conseguimos el error máximo en Zc menor). La desventaja que se obtiene al utilizar esta configuración es un pobre índice de compresión (6) y un elevado tiempo de ejecución (casi siete horas).

5.2.2.2 *Técnica de diferencia de fuente con regresión lineal y red neuronal*

- Técnica de diferencia de fuente utilizando Microstrip encapsulado

Núm. Capas	Neur Capa	Núm. Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Número Pesos	Número Ganancias	Parám Totales	Índice Comp.
1	**50**	**100**	0,1571%	1,5196%	0,0479%	1,7812%	2m 35s	(50x4) +30(RL)	50x1	280	34,29
1	**50**	**1000**	0,0709%	0,9203%	0,0246%	1,7796%	25m 19s	(50x4) +30(RL)	50x1	280	34,29
1	**50**	**2000**	0,0696%	0,7985%	0,0181%	1,7676%	46m 28s	(50x4) +30(RL)	50x1	280	34,29
2	**20**	**100**	0,5432%	6,5452%	0,1276%	1,7645%	4m 32s	(20x4)+(20x20) +30(RL)	20x2	550	17,45
2	**20**	**1000**	0,3370%	3,0894%	0,0956%	1,7534%	42m 24s	(20x4)+(20x20) +30(RL)	20x2	550	17,45
2	**20**	**2000**	0,2921%	3,8040%	0,0757%	1,7726%	1h 24m 44s	(20x4)+(20x20) +30(RL)	20x2	550	17,45
3	**20**	**100**	0,6685%	9,5688%	0,2012%	1,7461%	9m 35s	(20x4)+(20x20)+(20x20) +30(RL)	20x3	970	9,90
3	**20**	**1000**	0,2495%	3,5547%	0,0952%	1,8102%	1h 33m 11s	(20x4)+(20x20)+(20x20) +30(RL)	20x3	970	9,90
3	**20**	**2000**	0,2860%	2,8387%	0,0706%	1,7580%	3h 6m 13s	(20x4)+(20x20)+(20x20) +30(RL)	20x3	970	9,90

Tabla 5.11: Resultados de la técnica TDF utilizando Microstrip y Red Neuronal

Tabla 5.12: Resultados de la técnica TDF utilizando Coplanar y Red Neuronal

- Técnica de diferencia de fuente utilizando Coplanar encapsulado

Númer. Capas	Neur Capa	Número Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Número Pesos	Número Ganancias	Parám Totales	Índice compresión
1	**50**	**100**	0,0387%	0,8519%	0,0302%	0,7246%	2m 20s	(50x4) +30(RL)	50x1	280	31,37
1	**50**	**1000**	0,0136%	0,2471%	0,0150%	0,7285%	20m 52s	(50x4) +30(RL)	50x1	280	31,37
1	**50**	**2000**	0,0091%	0,2183%	0,0099%	0,7260%	41m 42s	(50x4) +30(RL)	50x1	280	31,37
2	**20**	**100**	0,1788%	3,0996%	0,1348%	0,9290%	4m 4s	(20x4)+(20x20) +30(RL)	20x2	550	15,97
2	**20**	**1000**	0,1035%	2,2922%	0,6090%	0,7305%	38m 46s	(20x4)+(20x20) +30(RL)	20x2	550	15,97
2	**20**	**2000**	0,0992%	1,6515%	0,0756%	0,7293%	1h 15m 44s	(20x4)+(20x20) +30(RL)	20x2	550	15,97
3	**20**	**100**	0,2204%	2,6905%	0,0835%	0,7913%	8m 37s	(20x4)+(20x20)+(20x20) +30(RL)	20x3	970	9,06
3	**20**	**1000**	0,0776%	1,3382%	0,0730%	0,7866%	1h 23m 46s	(20x4)+(20x20)+(20x20) +30(RL)	20x3	970	9,06
3	**20**	**2000**	0,0806%	1,3192%	0,0523%	0,7226%	2h 49m 37s	(20x4)+(20x20)+(20x20) +30(RL)	20x3	970	9,06

- Técnica TDF utilizando Red Neuronal y dispositivo Microstrip encapsulado (Tabla 5.11):

Si observamos los resultados de precisión obtenidos en esta tabla, podemos apreciar que no es recomendable utilizar para este dispositivo 2 o 3 capas de neuronas. Si utilizamos estas configuraciones, el error máximo para Zc aumenta considerablemente (indiferente del número de iteraciones), siendo no recomendable su uso para una posterior recuperación de los datos originales. La mejor configuración para esta técnica con el dispositivo Microstrip, es utilizar **1 capa con 50 neuronas y 2000 iteraciones.** Utilizando esta configuración, obtenemos la misma compresión que con menos iteraciones (34) y una mejor precisión (la mejor de la tabla), pero con un tiempo de ejecución mayor (46 minutos) que utilizando menos iteraciones.

- Técnica TDF utilizando Red Neuronal y dispositivo Coplanar encapsulado (tabla 5.12):

En esta técnica, es evidente que la configuración óptima a utilizar va a ser **1 capa con 50 neuronas.** La razón que excluye a utilizar 2 o 3 capas de neuronas es la precisión que se obtiene con estas configuraciones. Utilizando 1 capa de 50 neuronas, obtenemos todos los errores inferiores al 1%, indistintamente del número de iteraciones que estemos utilizando. Además, se consigue el mejor índice de compresión de todas las configuraciones de esta tabla (31). A la hora de elegir el número de iteraciones vamos a decantarnos por utilizar **2000 iteraciones** por su precisión con respecto a la de 1000 iteraciones. El tiempo de ejecución es mayor que para aquella con menos iteraciones (1000), pero con esta configuración tenemos un tiempo de procesado dentro de la media (por lo que es un tiempo admisible) y una precisión que es la mejor de toda las configuraciones.

5.2.3 Técnica con entrada de conocimiento previa (TECP)

5.2.3.1 Técnica con entrada de conocimiento previa con regresión lineal y MANFIS

- Técnica con entrada de conocimiento previa utilizando Microstrip encapsulado

Número	MFs	Número	Ereff	Ereff	Zc	Zc	Tiempo	Parám	Parám	Parámetros	Índice
		Epoch	Error med	Error máx	Error med	Error máx	Ejecución	Lineales	No lineales	Totales	Compresión
2	3	3	0,2730%	0,3410%	0,0316%	1,7725%	1h 3m 31s	1458	30	1488+30(RL)	6,32
2	3	30	0,0116%	0,3085%	0,0200%	1,7884%	10h 5m 16s	1458	30	1488+30(RL)	6,32
2	2	3	0,1231%	0,6405%	0,1704%	1,6762%	1m 1s	192	20	212+30(RL)	39,67
2	2	30	0,0718%	0,7212%	0,1230%	1,7712%	7m 52s	192	20	212+30(RL)	39,67

Tabla 5.13: Resultados de la técnica TECP utilizando Microstrip y MANFIS

- Técnica con entrada de conocimiento previa utilizando Coplanar encapsulado

Número MFs	Número Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Parám Lineales	Parám No lineales	Parámetros Totales	Índice Compresión
3	3	0,0030%	0,0257%	0,0315%	0,7396%	56m 23s	1458	30	1488+30(RL)	5,79
3	30	0,0006%	0,0253%	0,0275%	0,7216%	8h 51m 19s	1458	30	1488+30(RL)	5,79
2	3	0,0280%	0,6616%	0,1809%	1,1065%	53s	192	20	212+30(RL)	36,30
2	30	0,0096%	0,1870%	0,1794%	1,1300%	6m 56s	192	20	212+30(RL)	36,30

Tabla 5.14: Resultados de la técnica TECP utilizando Coplanar y MANFIS

- Técnica TECP utilizando MANFIS y dispositivo Microstrip encapsulado (Tabla 5.13):

Utilizando 3 funciones de pertenencia y 30 iteraciones (con 3 iteraciones se obtiene menos precisión), obtenemos el mayor tiempo de ejecución (más de diez horas), el error máximo en Zc más elevado de toda la técnica y un índice de compresión muy bajo entre las distintas configuraciones. A la hora de decantarnos por la mejor opción para esta técnica, vamos a elegir la configuración de **2 funciones de pertenencia y 3 iteraciones** (utilizando 30 iteraciones el sistema se sobre entrena y aumenta el error máximo). Utilizando esta configuración, conseguimos el menor error máximo en Zc y el resto de errores cometidos es muy bajo y admisible para la recuperación de datos. Además obtenemos el mínimo tiempo de ejecución (1 minuto) y un índice de compresión (IC = 39) seis veces mayor que utilizando 3 funciones de pertenencia.

- Técnica TECP utilizando MANFIS y dispositivo Coplanar encapsulado (tabla 5.14):

Utilizando 3 funciones de pertenencia y 30 iteraciones (usando 3 iteraciones se obtiene en media menos precisión), obtenemos el mayor tiempo de ejecución (mas de ocho horas) y un índice de compresión muy bajo. A la hora de decantarnos por la mejor opción para esta técnica vamos a elegir la configuración de **2 funciones de pertenencia y 30 iteraciones**. En media, la precisión con esta configuración es mayor (el error máximo es prácticamente el mismo que con 3 iteraciones). Utilizando esta configuración conseguimos una buena precisión, un bajo tiempo de ejecución (seis minutos) y un índice de compresión (IC = 36) seis veces mayor que utilizando 3 funciones de pertenencia.

5.2.3.2 Técnica con entrada de conocimiento previa con regresión lineal y red neuronal

- Técnica con entrada de conocimiento previa utilizando Microstrip encapsulado

Núm. Capas	Neur Capa	Núm. Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Número Pesos	Número Ganancias	Parám Totales	Índice Compr.
1	**50**	**100**	0,3527%	4,8032%	0,0989%	1,8399%	3m 33s	(50x6)+30(RL)	50x1	380	25,26
1	**50**	**1000**	0,1526%	1,5424%	0,0494%	1,8311%	38m 27s	(50x6)+30(RL)	50x1	380	25,26
1	**50**	**2000**	0,1078%	1,1117%	0,0304%	1,7990%	1h 5m 15s	(50x6)+30(RL)	50x1	380	25,26
2	**20**	**100**	0,6149%	8,9381%	0,2344%	2,5781%	4m 50s	(20x6)+(20x20)+30(RL)	20x2	590	16,27
2	**20**	**1000**	0,2359%	2,7088%	0,0895%	1,7295%	46m 27c	(20x6)+(20x20)+30(RL)	20x2	590	16,27
2	**20**	**2000**	0,4743%	5,4146%	0,1057%	1,7402%	1h 32m 50s	(20x6)+(20x20)+30(RL)	20x2	590	16,27
3	**20**	**100**	0,6161%	9,2365%	0,2144%	2,8265%	10m 17s	(20x6)+(20x20)+(20x20)+30(RL)	20x3	1010	9,50
3	**20**	**1000**	0,2474%	2,5389%	0,0708%	1,6958%	1h 40m 36s	(20x6)+(20x20)+(20x20)+30(RL)	20x3	1010	9,50
3	**20**	**2000**	0,2189%	1,8421%	0,0820%	1,8060%	3h 19m 19s	(20x6)+(20x20)+(20x20)+30(RL)	20x3	1010	9,50

Tabla 5.15: Resultados de la técnica TECP utilizando Microstrip y Red Neuronal

-

Tabla 5.16: Resultados de la técnica TECP utilizando Coplanar y Red Neuronal

Núm. Capas	Neur Capa	Núm. Epoch	Ereff Error med	Ereff Error máx	Zc Error med	Zc Error máx	Tiempo Ejecución	Número Pesos	Número Ganan.	Parám Totales	Índice comp.
1	**50**	**100**	0,1816%	3,6404%	0,1124%	0,9969%	3m 59s	(50x6)+30(RL)	50x1	380	23,12
1	**50**	**1000**	0,0232%	0,4815%	0,0233%	0,7215%	28m 28s	(50x6)+30(RL)	50x1	380	23,12
1	**50**	**2000**	0,0086%	0,1404%	0,0073%	0,7198%	56m 48s	(50x6)+30(RL)	50x1	380	23,12
2	**20**	**100**	0,1969%	2,8912%	0,1216%	1,2285%	4m 34s	(20x6)+(20x20)+30(RL)	20x2	590	14,89
2	**20**	**1000**	0,1641%	2,8474%	0,1076%	1,1315%	41m 32s	(20x6)+(20x20)+30(RL)	20x2	590	14,89
2	**20**	**2000**	0,0518%	0,7922%	0,0550%	0,7340%	1h 23m 43s	(20x6)+(20x20)+30(RL)	20x2	590	14,89
3	**20**	**100**	0,4016%	6,4139%	0,2483%	1,7550%	9m 14s	(20x6)+(20x20)+(20x20)+30(RL)	20x3	1010	8,70
3	**20**	**1000**	0,0937%	2,0862%	0,0658%	0,7968%	1h 29m 36s	(20x6)+(20x20)+(20x20)+30(RL)	20x3	1010	8,70
3	**20**	**2000**	0,0376%	0,6027%	0,0435%	0,7717%	2h 58m 52s	(20x6)+(20x20)+(20x20)+30(RL)	20x3	1010	8,70

- Técnica TECP utilizando Red Neuronal y dispositivo Microstrip encapsulado (Tabla 5.15):

Si observamos los resultados de precisión obtenidos en esta tabla, podemos apreciar que no es recomendable utilizar para este dispositivo 2 o 3 capas de neuronas. Si utilizamos estas configuraciones, el error aumenta considerablemente (indiferente del número de iteraciones) y, también, el tiempo de ejecución. En algunos casos, el uso de un número mayor de iteraciones ocasiona un sobre entrenamiento de la red neuronal y esto influye negativamente en la precisión de la técnica. La mejor configuración con esta técnica utilizando el dispositivo Microstrip, es utilizar **1 capa con 50 neuronas y 2000 iteraciones.** Para esta configuración, obtenemos la misma compresión que con menos iteraciones (IC = 25) y una mejor precisión (la mejor de la tabla). El punto en contra de esta configuración, es el tiempo de ejecución (1 hora y 5 minutos), que es más elevado que utilizando un número menor de iteraciones, pero sigue siendo un tiempo de ejecución admisible.

- Técnica TECP utilizando Red Neuronal y dispositivo Coplanar encapsulado (Tabla 5.16):

Si observamos los resultados de precisión obtenidos en esta tabla, podemos apreciar que no es recomendable utilizar para este dispositivo 2 o 3 capas de neuronas. Si utilizamos estas configuraciones obtenemos una buena precisión (no es la mejor de la tabla), pero el índice de compresión disminuye considerablemente y el tiempo de ejecución resulta excesivo. La configuración óptima de esta técnica es utilizar **1 capa de 50 neuronas**. Queda por analizar el número de iteraciones más conveniente. Al tener las tres opciones posibles el mismo índice de compresión (IC = 23), tenemos que decidir que configuración es la mejor fijándonos en el tiempo de ejecución y la precisión. Estudiando ambos casos, vamos a decantarnos por utilizar **1000 iteraciones**. Se ha escogido esta configuración porque tiene

una elevada precisión, muy similar a la de 2000 iteraciones. En cambio, esta precisión se alcanza en la mitad de tiempo de procesado (28 minutos) que utilizando 2000 iteraciones (56 minutos).

5.3 Conclusiones

Una vez escogidas las configuraciones óptimas para todas las técnicas de este PFC, vamos a escoger las dos mejores técnicas para nuestros dos dispositivos (Microstrip y Coplanar) y analizarlas para extraer la mejor. Para analizar estas técnicas y escoger la mejor, tenemos que elegir cual es la que tiene un mejor compromiso entre tiempo de ejecución, precisión e índice de compresión. A continuación, se muestran las tablas con las dos mejores técnicas para cada dispositivo.

- Configuraciones optimas para Microstrip y Coplanar

MICROSTRIP						
	Ereff		Zc		Tiempo	Índice
	Error med	Error máx	Error med	Error máx	Ejecución	compresión
MANFIS TECP	0,1231%	0,6405%	0,1704%	1,6762%	1m 1s	39,67
Red Neuronal TDF	0,0696%	0,7985%	0,0181%	1,7676%	46m 28s	34,29

Tabla 5.17: Técnicas óptimas Microstrip

COPLANAR						
	Ereff		Zc		Tiempo	Índice
	Error med	Error máx	Error med	Error máx	Ejecución	compresión
MANFIS TECP	0,0096%	0,1870%	0,1794%	1,1300%	6m 56s	36,30
Red Neuronal Clásica	0,0090%	0,1388%	0,0079%	0,7283%	42m 34s	35,14

Tabla 5.18: Técnicas óptimas Coplanar

- Mejor técnica para dispositivo Microstrip (tabla T.17):

Analizando ambas técnicas para este dispositivo, resulta evidente indicar que la mejor técnica para utilizar con nuestro dispositivo Microstrip es la **técnica de entrada de conocimiento previo (TECP) con MANFIS**. Esta técnica proporciona el mejor índice de compresión (39) y un tiempo de ejecución muy por debajo de la media (1 minuto). La precisión que se obtiene con esta técnica es bastante buena (muy similar a la otra técnica) y además obtenemos el error máximo de Zc más bajo.

- Mejor técnica para dispositivo Coplanar (Tabla 5.18):

Estudiando estas técnicas para el dispositivo Coplanar, debemos de tener en cuenta el tiempo de ejecución y la precisión que se obtienen, ya que tienen un índice de compresión muy similar (IC = 35). En cuanto a precisión, podemos observar que los errores sobre la permitividad efectiva son muy similares para las dos técnicas. Sin embargo, la técnica de conocimiento previo (TECA) con MANFIS tiene menor precisión para la impedancia Zc que la técnica clásica con red neuronal. Si comparamos los errores sobre la impedancia entre el dispositivo microstrip y Coplanar con la técnica de conocimiento previo, podemos observar que los errores medios son similares (0.17%), mientras que los errores máximos para el dispositivo Coplanar (1.13%) son inferiores a los errores para el dispositivo Microstrip (1.67 %). Teniendo en cuanta estas constataciones y que el tiempo de ejecución para la técnica de conocimiento previo (TECP) es aproximadamente 7 veces menor (7 minutos) que la técnica clásica con red neuronal, podemos elegir como mejor técnica para nuestro dispositivo Coplanar, la **técnica de entrada de conocimiento previo (TECP) con MANFIS**. Además, el índice de compresión es ligeramente superior con esta técnica (IC = 36) que con la técnica clásica con red neuronal (IC = 35).

REFERENCIAS

[1] T. Itoh and R. Mittra: “Spectral domain approach for calculating the dispersion characteristic of microstrip lines”, IEEE Transaction Microwave Theory and Techniques, vol. 21, pp. 496-499, 1973.

6. Conclusiones y perspectivas

Este Proyecto Fin de Carrera presenta nuevas técnicas para almacenar y comprimir datos. Estas nuevas técnicas son muy útiles para comprimir tablas de datos y pueden ser aplicadas a herramientas de diseño asistido por ordenador. Las técnicas están basadas en técnicas híbridas con distintas estructuras denominadas: Técnica de diferencia de fuente y técnica con entrada de conocimiento previa. Estos métodos combinan regresiones lineales con un sistema fuzzy o una red neuronal para representar de manera precisa tablas de datos. Estas técnicas establecen un alto índice de compresión con un error de descompresión razonable. El principal inconveniente de estos métodos es que requieren un cierto tiempo de entrenamiento (compresión) que varía según la estructura de la técnica utilizada (diferencia de fuente o entrada de conocimiento previa), el número de variables, el método de interpolación utilizado (red neuronal o sistema fuzzy) y los parámetros internos de estos (número de capas de la red neuronal, número de iteraciones, etc.). Sin embargo una vez implementadas, estas técnicas permiten descomprimir los resultados en tiempo real.

Estas técnicas fueron implementadas y testeadas con éxito para dos tablas de datos obtenidas de las simulaciones electromagnéticas de dos líneas de transmisión. El estudio de estas técnicas para las dos tablas de datos en función de sus parámetros internos (número de capas de red neuronal, número de iteraciones), del índice de compresión, el tiempo requerido para la compresión y el error cometido en la descompresión, ha permitido elegir la técnica híbrida con estructura de entrada de conocimiento previa (TECP) como el método con mayores prestaciones. Para las dos tablas de datos, esta técnica proporciona un índice de compresión mayor que 35, un tiempo de compresión inferior a 7 minutos y un error relativo de los datos descomprimidos con respecto a las tablas de datos inferior a 1.7%.

Los siguientes puntos resumen las tareas que hemos llevado a cabo en este proyecto fin de carrera:

1. Hemos estudiado tres métodos de aproximación: Regresión lineal, redes neuronales y sistemas fuzzy (ANFIS).
2. Hemos realizado una selección de las posibles técnicas híbridas de compresión que nos permitan proporcionar las mejores prestaciones.
3. Hemos buscado la fuente de obtención de nuestras tablas de datos. Como se quiere utilizar este proyecto para la compresión de tablas reales de datos, hemos escogidos dos dispositivos que mediante simulaciones electromagnéticas nos han proporcionado las tablas de datos que vamos a utilizar en este proyecto.
4. Para comparar las diferentes técnicas, aparte de los errores cometidos por estas, hemos incluido dos nuevos parámetros de calidad que nos permitió extraer la técnica que ofrece mayores prestaciones (índice de compresión y tiempo requerido para la compresión).
5. Hemos aplicado las técnicas híbridas de compresión a las dos tablas de datos.
6. Una vez finalizado el proceso de entrenamiento (compresión) de todas las técnicas, hemos comprobado éstas y almacenado todos los parámetros necesarios para poder compararlas entre si: Número de iteraciones, número de capas (solo para la red neuronal), pesos, el tiempo requerido para la compresión y los errores cometidos con respecto a los datos originales.
7. Hemos realizado un estudio sobre los resultados obtenidos de las distintas técnicas en función de los parámetros de calidad para decantarnos por el método que presenta mayores prestaciones.

8. Una vez terminado el estudio, concluimos proponiendo la técnica que mejor compromiso obtenga para nuestros dispositivos. La escogida para ambos dispositivos ha sido la técnica de entrada de conocimiento previo con el método de interpolación MANFIS.

Aunque hemos escogido la técnica TECP como técnica optima para la compresión de nuestras tablas, podemos decir que cualquiera de las técnicas presentadas puede tener un uso óptimo según la aplicación a llevar a cabo y el (o los) parámetro(s) de calidad deseado(s) (mínimo error, mayor índice de compresión, etc.).

Finalizaremos esta conclusión insistiendo sobre el interés cada vez mayor de las técnicas de compresión de datos no solo al nivel industrial, sino también en todas las facetas de nuestra vida cotidiana. En efecto, estas técnicas tienen su utilidad en las mayorías de las aplicaciones de tecnologías de la información y las comunicaciones como: Telefonía móvil, televisión digital, etc.