PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

COMPARACIÓN DE MLP-R Y MLP-C PARA APROXIMACIÓN DE FUNCIONES NO LINEALES

ROCÍO ALEJANDRA ARAVENA BUSTAMANTE

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO
PROFESIONAL DE INGENIERO
CIVIL EN INFORMÁTICA

NOVIEMBRE 2007

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

ACTA DE APROBACION

ROCÍO ALEJANDRA ARAVENA BUSTAMANTE

NIBALDO RODRIGUEZ AGURTO

PROFESOR GUÍA

NOVIEMBRE 2007

Resumen

Las redes neuronales tienen la propiedad de ser aproximadoras universales de funciones, esto permite que sean utilizadas en muchas aplicaciones de ingeniería, incluyendo problemas de telecomunicaciones tales como el procesamiento de señales. En esta tesis, en particular, se propone un esquema de predistorsión para reducir la distorsión no lineal de amplitud y fase introducida por un amplificador de tubo de onda viajera sobre una señal de 16 y 64-QAM. El compensador está basado en una red neuronal de tipo MLP (Multilayer Perceptron) de coeficientes complejos que aproxima la función inversa del amplificador, con el objeto de predistorsionar la señal. Además, se evalúa el rendimiento de la red neuronal compleja en comparación con su símil de coeficientes reales.

Abstract

Neural Networks have the property of being universal approximators of functions, this makes that they are used in many applications of engineering, including problems of telecommunications such as the signal processing. This paper proposes a predistortion scheme in order to reduce the non-lineal distortion in amplitude and phase introduced by a traveling wave tube amplifier on a signal of both 16 and 64-QAM. The predistorter is based on a neuronal network of type MLP (Multilayer Perceptron) and complex coefficients that approximates the inverse function of the amplifier, with the intention of predistorting the signal. In addition, is evaluated the performance of the complex neural network compared with is resemblance network of real coefficients.

Para la mujer que me apoyó todo estos años,
Por su infinito amor, cariño, comprensión y apoyo.
A mi madre.

Agradecimientos

A mis padres, por su amor y apoyo incondicional, a mi hermana y hermano por todo el tiempo que les robé no estando con ellos. A mis amigos. A mis profesores. A la Pontificia Universidad Católica de Valparaíso, por los mejores años de mi vida hasta ahora.

Índice

GLOSARIO

ART	Adaptative Resonance Theory
ADALINE	Adaptive Linear Neuron
ACPR	Adjacent Channel Power Ratio
AM/AM	Amplitude Modulation/Amplitude Modulation
AM/PM	Amplitude Modulation/Phase Modulation
BP	Backpropagation
BER	Bit Error Rate
BSB	Brain-State-in-a-Box
CBP	Complex-Backpropagation
DQPSK	Differential Quadrature Phase Shift Keying
DVB-S	Digital Video Broadcasting by Satellite
EVM	Error Vector Magnitude
ETSI Hiperlan	European Telecommunications Standards Institute for High Performance Radio LAN
HPA	High Power Amplifier
IEEE	Institute of Electrical and Electronics Engineers
INNS	International Neural Networks Society
LMS	Least Mean Square
MER	Modulation Error Ratio
MLP	Multilayer Perceptron
MADALINE	Multiple Adaline
M-QAM	Multiple Quadrature Amplitude Modulation
NTSC	National Television System(s) Committee.
OFDM	Orthogonal Frecuency-Division Multiplexing
PDP	Parallel Distributed Processing
PAPR	Peak to Average Power Ratio
PAL	Phase Alternating Line
RBF	Radial Basis Functions
RF	Radio frequency
RNA	Redes Neuronales Artificiales
RMS	Root Mean Square
SOM	Self-Organizing Maps

TWTA	Traveller Wave Tube Amplitude
UMTS	Universal Mobile Telecommunications System
VSA	Vector Signal Analyzer
VLSI	Very Large Scale Integration
WCDMA	Wideband-Code Division Multiple Access

Lista de Ilustraciones

Lista de Tablas

Capítulo 1

Introducción

Las redes neuronales constituyen un potente instrumento para la aproximación de funciones no lineales [1]. Su uso resulta especialmente útil en la modelización de aquellos fenómenos complejos donde la presencia de relaciones no lineales entre las variables es

habitual. La aplicación de este tipo de técnicas en el procesamiento de señales [2] y aplicaciones de telecomunicaciones ha proporcionado interesantes resultados [3].

Es importante señalar el explosivo crecimiento experimentado por las telecomunicaciones en Chile en los últimos años, en particular de las telecomunicaciones móviles (teléfonos celulares), desde el momento en que la tecnología permitió que se transformaran en un producto de consumo, ha tenido una demanda que la posiciona entre las tecnologías de más rápido crecimiento de los tiempos modernos. Este crecimiento ha estimulado la investigación en muchos campos, particularmente en aquellos relacionados con el uso eficiente del ancho de banda y eficiencia de potencia, este último un factor de suma importancia en los sistemas donde la autonomía es crucial, por ejemplo en telefonía móvil y satélites, puesto que una mayor eficiencia de potencia se traduce en una larga duración de la batería del móvil [4] y [5].

Ambas exigencias, elevada eficiencia de ancho de banda y elevada eficiencia de potencia, son, convencionalmente, difíciles de conciliar. Este escenario ha dado mayor vigencia al problema de la linealización de los *amplificadores de potencia*, en la búsqueda de una solución que permita una operación eficiente, en términos de potencia, por una parte y una amplificación sin distorsión de los formatos de modulación QAM [6] y OFDM [7], por otra. Vale decir, los amplificadores de potencia son dispositivos altamente no lineales, cuyo alto consumo de energía y su indispensable presencia en los sistemas de comunicaciones, demanda necesariamente un convenio entre eficiencia y linealidad, así pues, los amplificadores de potencia más eficientes desde un punto de vista de rendimiento energético, son los que presentan mayor grado de distorsión especialmente en los formatos de modulación anteriormente mencionados.

Ahora bien, en esta tesis se utilizan las redes de tipo Multilayer Perceptron (MLP) en conjunto con el algoritmo de aprendizaje Backpropagation (BP) para aproximar la función *inversa* de un amplificador de potencia de tubo (TWTA) [8] utilizado en antenas satelitales, de esta manera, la red neuronal actúa como un Predistorsionador que compensa la distorsión no lineal de fase y amplitud introducida por el amplificador en una señal de 16 y 64-QAM.

La Modulación Amplitud Cuadratura Multinivel (M-QAM) [9] ha sido adoptada en varios estándares de comunicaciones inalámbricos, tales como: IEEE 802.11a y difusión de video digital por satélite (DVB-S) debido a su alta eficiencia espectral, sin embargo, es altamente vulnerable a la distorsión no lineal y en particular, a la distorsión no lineal de amplitud y fase introducida por el amplificador de potencia de tubo, distorsión que degrada fuertemente la eficiencia del sistema de transmisión. Existen métodos de compensación para reducir estos efectos negativos: una solución simple para evitar estos efectos es operar el amplificador muy lejos de su punto de operación óptimo y así mantener la linealidad del sistema, pero esta solución ofrece un sistema de comunicación de baja eficiencia de potencia, logrando con ello que las baterías de los equipos móviles tienen una corta duración. Sin embargo, hoy en día los usuarios demandan equipos móviles de bajo consumo de potencia y amplio ancho de banda.

Otro método es la técnica de Predistorsión que ha sido considerada muy efectiva para compensar la distorsión no lineal de un amplificador de potencia. Consiste en pre-distorsionar digitalmente la señal con el objeto de linearizar la salida del amplificador. Por esta razón, la presente tesis plantea un Predistorsionador para compensar la distorsión no lineal introducida por un amplificador TWT en una señal de telecomunicaciones inalámbrica que está basado en una red neuronal con una capa oculta, un nodo de entrada y

un nodo de salida. Los coeficientes de la red son ajustados usando el algoritmo de entrenamiento Backpropagation en su versión compleja [10] y real [11].

Para crear la arquitectura del Predistorsionador se usaron redes neuronales de coeficientes complejos y reales. El propósito central de este trabajo es evaluar y confrontar el rendimiento de ambas redes, porque, si bien no es frecuente utilizar el algoritmo Backpropagation Complejo, por las particularidades que ciertas funciones analíticas presentan en este dominio, dado que la función a aproximar está también definida en el dominio de los números complejos, resulta lógico teorizar que una red neuronal compleja presentará un mejor rendimiento que su símil real a la hora de aproximar una función también compleja.

Para simular el esquema de Predistorsión se utiliza el software Matlab 7.0. [12] Los resultados obtenidos mediante esta simulación son presentados en detalle en las secciones siguientes.

1.1 Objetivos

1.1.1 Objetivo General

- *El objetivo de esta tesis es el desarrollo, implementación y evaluación de un compensador neuronal de coeficientes complejos para reducir la distorsión no lineal de un canal de comunicaciones digital.*

1.1.2 Objetivos Específicos

- *Revisión de la literatura sobre MLP y algoritmos de aprendizaje*
- *Estudio teórico del algoritmo backpropagation en el dominio de los Reales y en el dominio de los Complejos*
- *Evaluación y Comparación del rendimiento de ambos compensadores neuronales propuestos*

1.2 Organización del documento

Se ofrece a continuación un pequeño resumen de lo que se discutirá en cada uno de los capítulos.

Capítulo 2. Se introduce el concepto de red neuronal y los aspectos fundamentales de las mismas: arquitectura, topología, clasificación. Se presenta también la noción de aprendizaje de una red neuronal y algoritmos de entrenamiento utilizando el método de descenso de gradiente.

Capítulo 3. Un repaso al modelo neuronal de tipo Multilayer Perceptron (MLP) y el algoritmo de entrenamiento supervisado Backpropagation (BP) y Complex - Backpropagation (CBP, para el dominio de los números complejos) ambos utilizados a lo largo de la tesis

Capítulo 4. En este capítulo se presentan el problema que se estudia en la tesis y el modelo propuesto para su solución.

Los capítulos que se describen a continuación constituyen la aportación original de esta tesis:

Capítulo 5. Una vez introducido el problema objeto de esta memoria de título, así como el modelo con el que se busca implementar su solución, se hace un estudio comparativo del rendimiento del compensador propuesto y sus resultados obtenidos mediante simulación computacional.

Capítulo 6. Este capítulo recoge las principales conclusiones que se deducen de todo lo anterior, además de presentar posibles trabajos de investigación para el futuro.

Anexo 1. Este anexo muestra el código de los algoritmos de entrenamiento utilizados para implementar el compensador neuronal.

Anexo 2. Muestra los resultados del la ejecución del simulador programado en MatLab para el compensador neuronal complejo y compensador real, con modulación 16 y 64-QAM.

Capítulo 2

Estado del Arte de las Redes Neuronales

2.1 Introducción a las Redes Neuronales Artificiales

El interés por comprender los procesos cognoscitivos del cerebro humano, fue, en definitiva lo que permitió el surgimiento de las Redes Neuronales Artificiales (RNA), debido a que el cerebro es un sistema de procesamiento de la información extremadamente complejo, cuyo modo de funcionamiento es eminentemente paralelo y cuyo comportamiento no puede describirse por medio de modelos sencillos como lo son los lineales se buscó modelarlo, posteriormente, con la esperanza de que se pudieran crear sistemas pensantes que tuvieran mejores resultados en tareas como clasificación, problemas de decisión, predicciones y sistemas de control adaptables que un sistema computacional convencional.

En definitiva, las RNA surgieron de la observación del funcionamiento del cerebro y de su comparación con la forma de trabajar de los computadores. Los elementos estructurales constituyentes del cerebro, las neuronas biológicas [13], son unos seis órdenes de magnitud más lentas que las puertas lógicas de silicio, ofreciendo tiempos de respuesta del orden del milisegundo frente a los vertiginosos tiempos de respuesta del orden del nanosegundo que tienen ciertos dispositivos digitales actuales.

Esta relativa lentitud no impide al cerebro realizar ciertas funciones habituales, como son las tareas de reconocimiento, percepción y control motriz, con una eficacia y rapidez fuera del alcance del mejor computador actual. Esta supremacía le viene dada por la alta complejidad de su estructura, formada por un elevado número de células nerviosas (neuronas) masivamente interconectadas y funcionando en paralelo. Se estima que el número de neuronas en el córtex humano es del orden de diez millares de millones (10^{10}), y que el número de sinapsis o conexiones es del orden de 60 billones ($60*10^{12}$). La alta complejidad de esta estructura biológica no sería operativa si no fuese además eficiente. La eficiencia energética del cerebro es aproximadamente de 10^{-16} julios por operación por segundo, mientras que los mejores ordenadores no superan los 10^{-6} julios por operación por segundo [14].

Las RNA están inspiradas de la estructura del cerebro, y fueron concebidas para resolver cierto tipo de problemas especialmente mal resueltos por las técnicas de programación tradicionales. Formalmente, computación neuronal es la disciplina tecnológica que trata sistemas paralelos y adaptativos de procesamiento de información

distribuida, que desarrollan sus capacidades bajo su exposición a un entorno de información.

A pesar de esto, las RNA son diferentes de su símil biológico, puesto que su arquitectura está fuertemente ligada a las necesidades de las aplicaciones para la que es diseñada, en cambio, el modelo de tipo biológico busca simular, evidentemente, sistemas neuronales biológicos, tal como las funciones auditivas o algunas funciones básicas de la visión. Sin embargo, ambos presentan la noción de neurona como elemento estructural constitutivo del modelo. En la gráfica 2.1 se puede ver un esquema simplificado de la morfología de una neurona típica. Algunos elementos a destacar de su estructura son:

- Las **dendritas**, que son la vía de entradas de las señales que se combinan en el cuerpo de la neurona. De alguna manera, la neurona elabora una señal de salida a partir de ellas.
- El **axón**, que es la vía de salida de la señal generada por la neurona.
- La **sinapsis** o proceso de comunicación entre neuronas.

El concepto de plasticidad esta relacionado con la capacidad del cerebro de responder de forma correcta a los estímulos exteriores, incluso frente a un estímulo nunca antes recibido. Así como la plasticidad parece la clave esencial en el funcionamiento de las neuronas como elemento de proceso de la información, dicho elemento se modeliza e intenta imitar en las redes neuronales artificiales.

En la bibliografía se pueden encontrar diferentes definiciones sobre lo que es una red neuronal, quizá la más acertada es la siguiente [15]:

> "*Una red neuronal es un modelo computacional con un conjunto de propiedades específicas, como son la habilidad de adaptarse o aprender, generalizar u organizar la información, todo ello basado en un procesamiento eminentemente paralelo*."

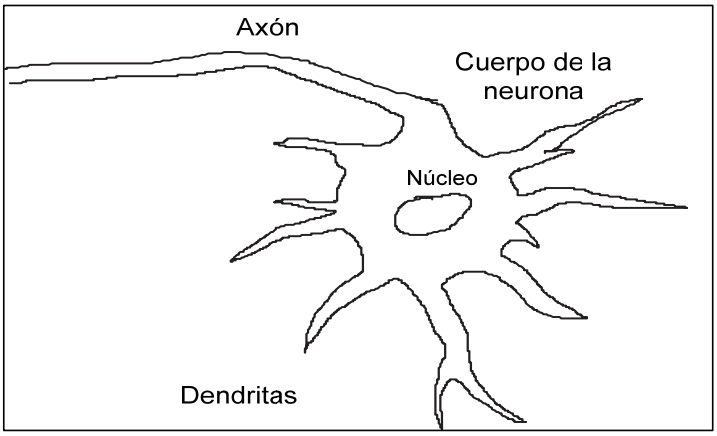


Figura 2.1 Modelo de una Neurona Biológica

2.2 Historia de las redes Neuronales Artificiales

En 1936 Alan Turing fue el primero en estudiar el cerebro como una forma de ver el mundo de la informática [16]. Sin embargo, los primeros teóricos que concibieron los

fundamentos de la *computación* neuronal fueron Warren McCulloch, un neurofisiólogo, y Walter Pitts, un matemático, quienes, en 1943, lanzaron una teoría acerca de la forma de trabajar de las neuronas [17] Ellos modelaron una red neuronal simple mediante circuitos eléctricos.

En 1949 Donald Hebb fue el primero en explicar los procesos del aprendizaje (que es el elemento básico de la inteligencia humana) desde un punto de vista psicológico, desarrollando una regla de como el aprendizaje ocurría. Aún hoy, este es el fundamento de la mayoría de las funciones de aprendizaje que pueden hallarse en una red neuronal. Su idea fue que el aprendizaje ocurría cuando ciertos cambios en una neurona eran activados. También intentó encontrar semejanzas entre el aprendizaje y la actividad nerviosa. Los trabajos de Hebb [18] formaron las bases de la Teoría de las Redes Neuronales.

En 1956 se organizó en Dartmouth la primera conferencia sobre Inteligencia Artificial (IA). Aquí se discutió el uso potencial de los computadores para simular *todos los aspectos del aprendizaje o cualquier otra característica de la inteligencia* y se presentó la primera simulación de una red neuronal, aunque todavía no se sabían interpretar los datos resultantes.

En 1959, Widrow publica una teoría sobre la adaptación neuronal y unos modelos inspirados en esa teoría [19], el Adaline (Adaptative Linear Neuron) y el Madaline (Multiple Adaline). Estos modelos fueron usados en numerosas aplicaciones y permitieron usar, por primera vez, una red neuronal en un problema importante del mundo real: filtros adaptativos para eliminar ecos en las líneas telefónicas.

En 1962, Rosemblatt publica los resultados de un ambicioso proyecto de investigación, el desarrollo del Perceptrón [20], un identificador de patrones ópticos binarios, y salida binaria. Las capacidades del Perceptrón se extendieron al desarrollar la regla de aprendizaje delta, que permitía emplear señales continuas de entrada y salida.

En 1969, Minsky y Papert realizan una seria crítica del Perceptrón [21], revelando serias limitaciones, como su incapacidad para representar la función XOR, debido a su naturaleza lineal. Este trabajo creó serias dudas sobre las capacidades de los modelos conexionistas y provocó una caída en picado de las investigaciones.

En los años 70, a pesar del duro golpe que supuso el trabajo de Minsky y Papert para las investigaciones en el campo de las RNA, un puñado de investigadores siguió trabajando y desarrollando nuevas ideas:

Anderson estudia y desarrolla modelos de memorias asociativas. Destaca el autoasociador lineal conocido como modelo brain-state-in-a-box [22] (BSB).

Kohonen continua el trabajo de Anderson y desarrolla modelos de aprendizaje competitivo basados en el principio de inhibición lateral. Su principal aportación consiste en un procedimiento para conseguir que unidades físicamente adyacentes aprendieran a representar patrones de entrada similares [23].

Grossberg realizó un importante trabajo teórico - matemático tratando de basarse en principios fisiológicos; aportó importantes innovaciones con su modelo ART (Adaptative Resonance Theory) [24] y, junto a Cohen, elabora un importante teorema sobre la estabilidad de las redes recurrentes en términos de una función de energía .

En los años 80: En esta década se produce el renacimiento del interés por el campo gracias sobre todo al trabajo de el grupo PDP (Parallel Distributed Processing) creado por Rumelhart, McClelland & Hinton. Como resultado de los trabajos de este grupo salieron los manuales con más influencia desde la crítica de Minsky y Papert. Destaca el capítulo dedicado al algoritmo de backpropagation [25], que soluciona los problemas planteados

por Minsky y Papert y extiende enormemente el campo de aplicación de los modelos de computación conexionistas.

Hopfield elabora un modelo de red consistente en unidades de proceso interconectadas que alcanzan mínimos energéticos, aplicando los principios de estabilidad desarrollados por Grossberg. El modelo de Hopfield [26] resultó muy ilustrativo sobre los mecanismos de almacenamiento y recuperación de la memoria. Su entusiasmo y claridad de presentación dieron un nuevo impulso al campo y provocaron el incremento de las investigaciones.

Finalmente, en las últimas décadas se mantiene un continuo interés en la investigación de las RNA y sus posibles aplicaciones, lo que ha traído numerosos avances no sólo en informática sino también en diversos campos de la ingeniería, medicina, economía y otras ciencias. Algunos de los hitos que marcaron los últimos años:

En 1987 se celebra la IEEE International Conference on Neural Networks con 1700 participantes en San Diego. Se crea la *International Neural Networks Society* (INNS)

En 1988 se publica la revista *Neural Networks* por el INNS.

En 1989 se publica la revista *Neural Computation*

En 1990 se publica la revista *Transactions on Neural Networks* por el IEEE.

2.3 Un modelo de Neurona. El Nodo.

El elemento constitutivo básico de un sistema neuronal biológico es la neurona, lo mismo sucede con las redes neuronales artificiales, en este caso a la neurona se le denomina Elemento de Proceso o Nodo. En la gráfica 2.2 se puede observar un modelo de neurona con sus tres elementos fundamentales:

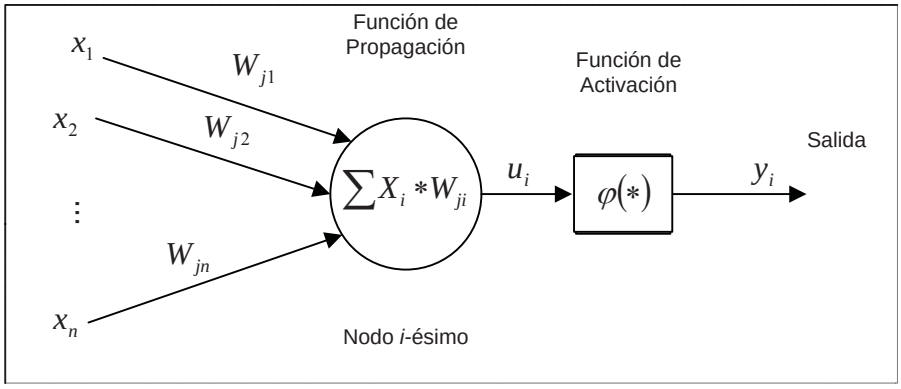


Figura 2.2: Modelo de una neurona artificial o nodo.

- Un conjunto **conexiones**, cada una de ellas caracterizada por su peso. Así, una señal de entrada X_i tras pasar la conexión, se habrá convertido en una señal $X_i * W_{ji}$,donde W_{ji} es el peso o fuerza de la conexión con la entrada i-ésima de la neurona j. De acuerdo con el signo del peso W_{ji} se tienen conexiones excitadoras cuando es positivo, y conexiones inhibidoras cuando es negativo.

- Una **función de propagación** correspondiente a la entrada neta $\sum(X_i * W_{ji})$, que produce la suma ponderada de las entradas de acuerdo a los correspondientes pesos de las conexiones.
- Una **función de activación**, $\phi(*)$, es el proceso algorítmico que transforma el resultado de la función de propagación en la salida real de la neurona. En la mayoría de los casos la función de activación y la de propagación son idénticas. La función de activación $\phi(*)$ determina el nivel de activación de la neurona en términos de la actividad existente en sus entradas. Hay una infinidad de funciones para ser utilizadas como función de activación en una red neuronal artificial, pero se pueden distinguir tres grandes clases: tipo escalón, lineales y sigmoide. Es más frecuente utilizar funciones sigmoidales, puesto que éstas y sus derivadas son continuas. Algunas funciones de activación pueden verse en la gráfica 2.3. Las funciones de activación más utilizadas habitualmente son las siguientes:

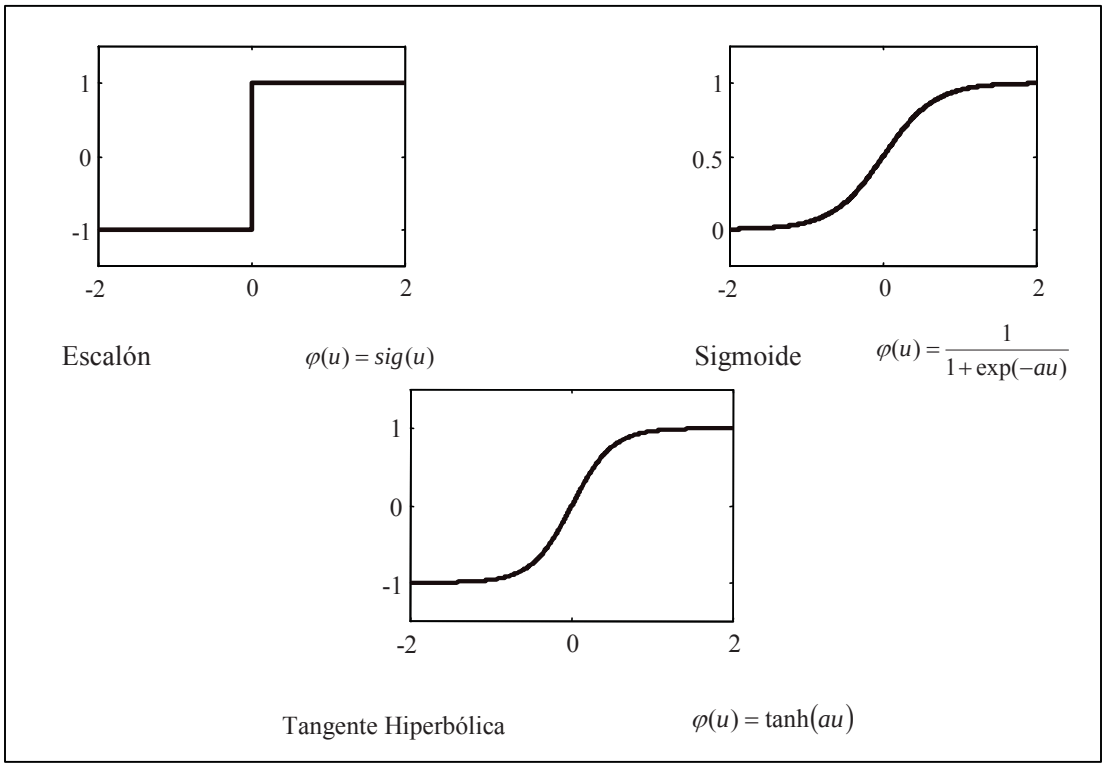


Figura 2.3. Existen varias funciones de activación. La Ilustración muestra las tres más usadas; función escalón, función sigmoide y tangente hiperbólica

En la función de activación el valor de la salida de combinación puede ser comparada con algún valor umbral para determinar la salida de la neurona. Si la suma es mayor que el valor umbral, neurona generará una señal. Si la suma es menor que el valor umbral, ninguna señal será generada.

De acuerdo con este modelo se puede describir el comportamiento de la neurona (o nodo) con las siguientes expresiones:

- X_i : entrada a la neurona *i*

- $\sum (X_i * W_{ji})$: entrada neta o función de propagación de la neurona *i*-ésima
- W_{ji} : peso de la conexión de la neurona *i*-ésima a la neurona *j*-ésima
- $y = \phi(u) = \phi\left(\sum X_i * W_{ji}\right)$: Función de activación de la neurona *i*-ésima

2.4 Modelo de Red Neuronal Artificial

Una red neuronal puede verse un modelo del cerebro humano y que busca simular sus capacidades cognoscitivas. Para lograr este objetivo, una red neuronal está formada por un conjunto de neurona interconectadas entre sí formando capas.

Cada neurona recibe como entrada un conjunto de señales discretas o continuas, las pondera e integra, y transmite el resultado a los nodos conectados a él. Cada conexión entre dos neuronas tiene una determinada importancia asociada denominada peso sináptico o, simplemente, peso. En los pesos se suele guardar la mayor parte del conocimiento que la red neuronal tiene sobre la tarea en cuestión. El proceso mediante el cual se ajustan estos pesos para lograr un determinado objetivo se denomina *aprendizaje* o *entrenamiento* y el procedimiento concreto utilizado para ello se conoce como algoritmo de aprendizaje o algoritmo de entrenamiento. El ajuste de pesos es la principal forma de aprendizaje de las redes neuronales, aunque hay otras formas posibles (Por ejemplo, la modificación del número de nodos o de la forma de conectarlos)

En definitiva, las RNA son estructuras adaptativas de procesamiento de información inspiradas en la estructura cerebral, donde el procesamiento se lleva a cabo mediante la interconexión de nuronas. Esta arquitectura da lugar a estructuras altamente paralelizables donde el flujo de información no sigue un camino secuencial, sino que se distribuye a través de las conexiones de los elementos de proceso donde la información es tratada.

Durante la fase de aprendizaje, la RNA se expone a un entorno de información para que pueda adaptar sus pesos y posiblemente también su estructura. Durante la fase de evaluación, los pesos de la red se mantienen fijos y la RNA se limita a tratar la información de entrada que se le suministra.

Existen tres aspectos que caracterizan una red neuronal y que constituyen su arquitectura: el número de capas, el mecanismo de aprendizaje, y por último, el tipo de asociación realizada entre la información de entrada y de salida. Con el objeto de definir adecuadamente una RNA estos aspectos se detallan a continuación.

2.5 Arquitectura de las Redes Neuronales Artificiales

Muchas veces se habla de la arquitectura de una red neuronal. Este concepto se refiere básicamente a la manera en que se interconectan los distintos nodos que forman la red.

Normalmente los nodos se organizan como una secuencia de capas con un determinado patrón de interconexión entre las diferentes neuornas que las forman, y con un patrón de conexión entre las neuronas de las distintas capas. Uno de los rasgos que puede ayudar a definir una capa es el hecho de que todas las neuronas que la forman usan la misma función de propagación.

2.5.1 Arquitectura de capas

En muchas de las arquitecturas de redes neuronales se puede hacer la siguiente distinción entre las capas:

- **Redes monocapa**: son aquellas compuestas por una única capa de neuronas.
- **Redes multicapa** : son aquellas cuyas neuronas se organizan en varias capas.

También se pueden hacer la siguiente distinción (Figura 2.4):

- **Capa de entrada**: está compuesta por neuronas que reciben entradas procedentes del entorno. sus pesos se mantienen constantes, y su misión simplemente es la de distribuir dicha entrada al resto de los elementos de proceso que constituyen la red.
- **Capa oculta**: es aquella que no tiene conexión directa con el contorno, es decir, que no es de entrada ni de salida. Crean una representación interna de los patrones de entrada. Puede haber más de una capa oculta.
- **Capa de salida**: es aquella cuyas neuronas proporcionan la respuesta de la red neuronal.

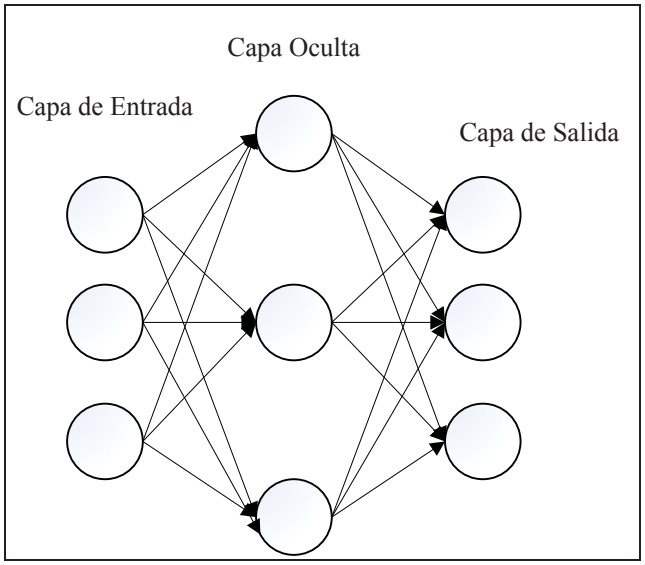


Figura 2.4. Capa de Entrada, Oculta y de Salida de una red

2.5.2 Arquitectura basada en el flujo de señales

Atendiendo al flujo de datos en la red neuronal, se puede hablar de:

- **Redes Feedforward**: la información circula en un único sentido desde las neuronas de entrada a las de salida. Representan sistemas lineales dado que no existen nodos de realimentación (Figura 2.5).

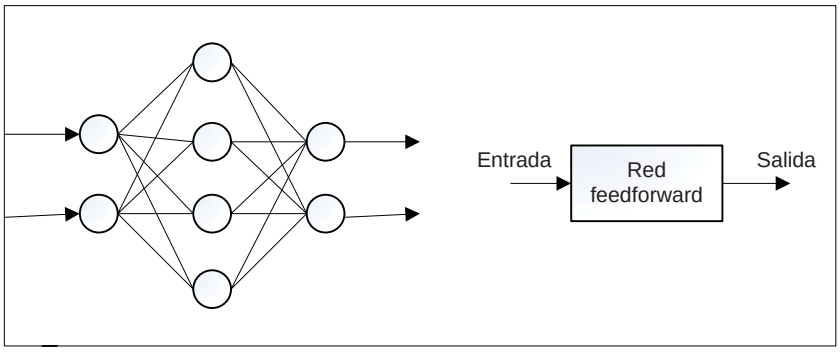


Figura 2.5. Red feedforward

Algunas de las redes Feedforward mas importantes son: las redes Multilayer Perceptron (MLP) [25], las redes RBF (Radial Basis Functions) [26] y las redes SOM (Self-Organizing Maps) [23].

- **Redes Feedback**: la información puede circular entre las capas en cualquier sentido. Representan sistemas no lineales. Son llamadas también recurrentes y poseen unidades de realimentación. (Figura 2.6)

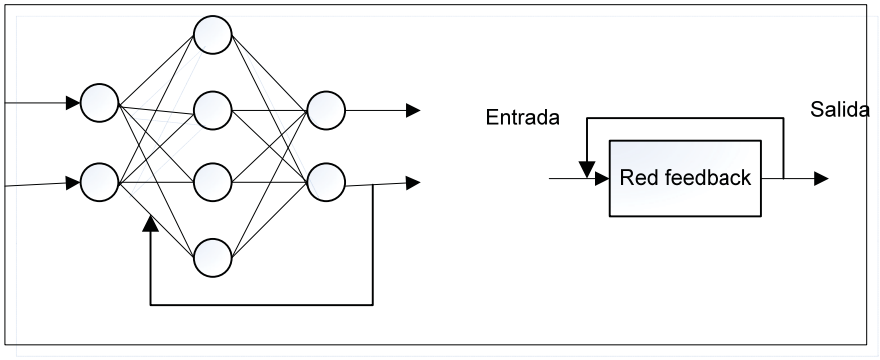


Figura 2.6: Red feedback

2.5.3 Topología de la Asociación entre los datos de entrada y salida

Ya se sabe que las redes neuronales son sistemas que almacenan cierta información aprendida. Esta información se registra de forma distribuida en los pesos asociados a las conexiones entre nodos. Por tanto, puede imaginarse una red como cierto tipo de memoria que almacena datos de forma estable, datos que se grabarán en dicha memoria como consecuencia del aprendizaje de la red y que podrán ser leídos a la salida como respuesta a cierta información de entrada, comportándose entonces la red como lo que habitualmente se conoce por memoria asociativa: cuando se aplica un estímulo (dato de entrada) la red responde con una salida asociada a dicha información de entrada.

Existen dos formas primarias de realizar esta asociación entre entradas/salidas que se corresponden con la naturaleza de la información almacenada en la red:

- Una primera sería la denominada heteroasociación, que se refiere al caso en el que la red aprende parejas de datos $[(X_1, Y_1), (X_2.Y_2), ..., (X_n, Y_n)]$ de tal forma que cuando

se presente cierta información de entrada X_i, deberá responder generando la correspondiente salida asociada Y_i.

- La segunda se conoce como autoasociación, donde la red aprende ciertas informaciones $X_1, X_2, ..., X_n$; de tal forma que cuando se le presenta una información de entrada realizará una autocorrelación, respondiendo con uno de los d atos almacenados, el más parecido al de entrada.

Estos dos mecanismos de asociación dan lugar a dos tipos de redes neuronales: las redes heteroasociativas y las autoasociativas. Una red heteroasociativa podría considerarse como aquella que computa cierta función, que en la mayoría de los casos no podría expresarse analíticamente, entre un conjunto de entradas y un conjunto de salidas, correspondiendo a cada posible entrada una determinada salida. Por otra parte, una red autoasociativa es una red cuya principal misión es reconstruir una determinada información de entrada que se presente incompleta o distorsionada (le asocia el dato almacenado más parecido).

2.5.4 Redes Heteroasociativas

Las redes heteroasociativas, al asociar informaciones de entrada con diferentes informaciones de salida, precisan al menos de dos capas, una para captar y retener la información de entrada y otra para mantener la salida con la información asociada. Si esto no fuese así, se perdería la información inicial al obtenerse el dato asociado, lo cual no debe ocurrir, ya que en el proceso de obtención de la salida se puede necesitar acceder varias veces a esta información que, por tanto, deberá permanecer en la capa de entrada. En cuanto a su conectividad, pueden ser del tipo con conexión hacia delante (o feedforward) o con conexión hacia atrás (feedback) o bien con conexiones laterales.

2.5.5 Redes Autoasociativas

Una red autoasociativa asocia una información de entrada con el ejemplar más parecido de los almacenados conocidos por la red. Estos tipos de redes pueden implementarse con una sola capa de neuronas. Esta capa comenzará reteniendo la información inicial a la entrada, y terminará representando la información autoasociada. Si se quiere mantener la información de entrada y salida, se deberían añadir capas adicionales, sin embargo, la funcionalidad de la red puede conseguirse en una sola capa.

2.5.6 Algoritmos de Aprendizaje

Se denomina paradigma de aprendizaje al modelo del entorno en el que la red neuronal trabaja.

Una de las principales ideas sobre las que se basan las redes neuronales artificiales es la de responder a los estímulos del entorno mediante un proceso de aprendizaje por el cual va adaptando los pesos de las conexiones de sus nodos, de tal forma que memoriza los ejemplos de entrenamiento que se le presentan. El paradigma de aprendizaje indica la forma en que el entorno influye en ese proceso de aprendizaje.

- **Supervisado**: Se presentan los conocimientos en forma de pares de [entrada, salida deseada]. La salida real de la red es comparada con el valor deseado de salida, Los pesos, que normalmente han sido establecidos de manera aleatoria en un principio, son ajustados por la red de manera que en la siguiente iteración, producirá un resultado más cercano entre el valor esperado y la salida real. Hasta ajustar la salida a un margen de error aceptable.
- **No Supervisado**: Durante este proceso de aprendizaje a la red no se la presenta la salida deseada. Sus principales utilidades son, entre otras, descubrir las regularidades presentes en los datos, extraer rasgos o agrupar patrones. Un ejemplo bien conocido son las redes SOM [23].
- **Reforzado**: Se sitúa entre los dos anteriores, de forma que, por una parte se emplea la información del error cometido, pero se sigue sin poseer la salida deseada. Este aprendizaje descansa en la idea de premio-castigo, donde se refuerza toda aquella acción que permita una mejora del modelo.
- **Híbrido**: Coexisten en la red los dos tipos básicos de aprendizaje, el supervisado y el no supervisado, normalmente en distintas capas de neuronas.

Sin embargo, a efectos de esta tesis, es más relevante clasificar los algoritmos de aprendizaje de otra manera, basado en el método de cálculo de la función de error, esto es, basados en la primera derivada o segunda derivada y algoritmos no basados en la derivada.

Ahora bien, Para entrenar la red de forma supervisada se necesita normalmente algún tipo de medida del error $E[t]$ que describa la adecuación de la salida proporcionada por la red al valor deseado. Los parámetros se ajustan intentando minimizar este error. La función de error más habitual es la función de error cuadrático, definida para el instante *t* como:

$$E[t] = \sum_{i=1}^{n} \left(d_i[t] - y_i[t]\right)^2 \qquad (1)$$

donde el subíndice *i* corresponde al *i*-ésimo nodo de la red , *n* es el número total de nodos , $d_i[t]$ es la salida deseada u objetivo para el *i*-ésimo nodo de salida en el instante *t* e $y_i[t]$ es la salida real de la red.

Una posible forma de encontrar la solución que minimice el valor del error es la búsqueda exhaustiva sobre todas las posibles combinaciones de valores de los pesos (o sobre un conjunto finito lo suficientemente significativo de posibles valores). Evidentemente, esta forma de resolución es intratable en la mayoría de los casos. Si el problema a aprender es sencillo, puede que una estrategia basada en generar aleatoriamente conjuntos de valores para los pesos funcione. En general, sin embargo, se hace necesaria la utilización de algún tipo de heurística que recorte el espacio de soluciones a explorar; esta es la labor de los algoritmos de aprendizaje y estos se detallan en el siguiente capítulo.

2.6 Ventajas de las RNA

Las RNA deben su capacidad de procesamiento de información a su estructura distribuida y paralela (la información queda almacenada en los elementos de proceso de la

red de forma no centralizada) y a su capacidad de aprendizaje y por tanto de generalización (en contraposición con la memorización).

Estas dos capacidades de procesamiento hacen que las RNA sean capaces de resolver cierto tipo de problemas muy complejos (tanto por su tamaño como por su estructura) que hasta antes de que aparecieran no habían quedado resueltos de forma satisfactoria.

Hay que dejar claro que las RNA no son una nueva metodología de ingeniería para la resolución global de problemas. Son herramientas de tratamiento de información que pueden integrarse fácilmente en arquitecturas modulares, para resolver de forma muy eficaz aquellas subtareas precisas del problema global que mejor se adaptan a sus capacidades. Entre estas subtareas es posible citar procedimientos de reconocimiento de patrones, de aproximación funcional y de memorias asociativas.

Las propiedades o características de las RNA que suelen ser más útiles son:

- **No linealidad:** las neuronas son elementos de proceso generalmente no lineales. La interconexión de estos elementos genera estructuras de transformación de datos donde este carácter no lineal queda distribuido a lo largo y ancho de la red. Esta característica permite modelar procesos intrínsecamente no lineales (como por ejemplo el reconocimiento del discurso hablado) pero complica también los métodos de análisis de las estructuras resultantes, impidiendo la aplicación de técnicas de análisis bien establecidas como son las de los sistemas lineales.
- **Modelado de relaciones de entrada/salida:** un paradigma de aprendizaje especialmente extendido y útil para los objetivos de esta tesis es el llamado aprendizaje supervisado. En este tipo de aprendizaje se dispone de un conjunto de muestras de la relación entrada/salida a modelar, formado por pares (entradas, salidas deseadas), que permite optimizar los pesos de la red de tal forma que se espera que la relación de entrada/salida generada sea capaz de reproducir casos no representados en el conjunto de datos original. Esta capacidad de generalización se obtiene utilizando estructuras de aproximación funcional con capacidad de representación universal, y estrategias de aprendizaje como las descritas en el Capítulo 3, que cuidan de un modo especial el no sobrepasar el límite del sobre-entrenamiento.
- **Adaptabilidad:** las RNA son por definición estructuras adaptativas capaces de ajustar sus pesos, y por tanto su función de transferencia, a cambios en el entorno. Esta característica las hace particularmente útiles en el tratamiento de procesos no estacionarios, donde pueden diseñarse estrategias de aprendizaje en tiempo real para que el modelo conexionista se vaya adaptando de forma continua a los cambios del proceso en cuestión. En el diseño de sistemas adaptativos hay que tener siempre en cuenta las constantes de tiempo del sistema de adaptación y las del sistema bajo estudio: un sistema adaptativo con constantes de tiempo demasiado bajas puede responder demasiado rápido de tal forma que no sea capaz de ignorar perturbaciones espúreas y se haga inestable. En el caso concreto del diagnóstico basado en modelos conexionistas de funcionamiento normal esta capacidad de adaptación permitirá al sistema de diagnóstico el irse acomodando al lento proceso de envejecimiento de los componentes. Otros campos de aplicación relevantes son el reconocimiento adaptativo de patrones, el procesamiento adaptativo de señales y el control adaptativo.

- **Respuesta evidencial:** en el ámbito de aprendizaje supervisado (para aproximación funcional y clasificación), una RNA puede, además de estimar la salida deseada, dar una medida de la fiabilidad de la estimación. Esta información puede ser utilizada para rechazar patrones de entrada, completando de esta forma el proceso de estimación.
- **Tolerancia frente a fallos:** una red neuronal realizada en *hardware* tiene la capacidad de seguir respondiendo de forma no catastrófica cuando parte de su estructura está dañada. Esto es debido al tratamiento distribuido de la información y a la redundancia implícita en su estructura.
- **Realización en VLSI:** la naturaleza masivamente paralela de las RNA las hace potencialmente eficaces para la realización de ciertas tareas complejas. Esta misma característica, junto con la uniformidad de su estructura, las hace especialmente adecuadas para su construcción en tecnología de integración VLSI (*Very Large Scale Integration*). Esto permite construir estructuras extremadamente complejas (en cuanto a su tamaño) que de otro modo serían inviables.

2.7 Aplicaciones de las RNA

2.7.1 Procesamiento de lenguaje natural

- **Conversión de texto escrito a lenguaje hablado:** NETtalk (Sejnowski T. & Rosemberg.)[27]: toma como entradas textos escritos y como salidas deseadas los códigos elegidos para representar los fonemas correspondientes. Mediante la ayuda de un sintetizador (DECtalk) se transforman los códigos en fonemas. Durante el proceso de aprendizaje se observó como iba mejorando su habilidad desde un nivel de bebé hasta el nivel de un niño de 6 años, aprendiendo a hacer distinciones difíciles como pronunciar una c suave o fuerte según el contexto. Si bien esto se había conseguido antes, la novedad más importante reside en que mediante la red neuronal no es necesario definir y programar un montón de complejas reglas, puestoque la red extrae automáticamente el conocimiento necesario.
- **Aprendizaje de gramáticas**: Rumelhart y McClelland [25] estudiaron la forma en que construimos las reglas sobre el lenguaje, y trataron de enseñar a una red neuronal el pasado de los verbos ingleses. El sistema fue mejorando y al final era capaz de generalizar y conjugar verbos desconocidos.

2.7.2 Reconocimiento de caracteres

- **Reconocimiento de escritura manual:** El Neocognitrón (Kunihiko Fukishima): [28] simula la forma en que la información visual avanza en la corteza cerebral. Consigue un reconocimiento muy avanzado de patrones con gran capacidad de abstracción y generalización, que lo hacen capaz de reconocer patrones con distinta orientación y altos niveles de distorsión.

2.7.3 Reconocimiento de patrones en imágenes

- **Clasificación de objetivos**: en este campo se han desarrollado numerosas aplicaciones como la clasificación de imágenes de sonar y radar, la detección de células cancerosas [29], lesiones neurológicas [30] y cardíacas [31], prospecciones geológicas, etc. Son muy útiles para procesar imágenes de las que no se sabe bien cuales son las características esenciales o diferenciales, ya que las redes no necesitan disponer de reglas explícitas previas para realizar la clasificación, sino que extraen el conocimiento necesario.
- **Visión artificial en robots industriales**: para inspección de etiquetas, clasificación de componentes [32] (minería), etc. Supera a otros sistemas de visión, además minimiza los requerimientos de operadores y facilita el mantenimiento.

2.7.4 Procesamiento de señales

- **Predicción:** se han obtenido mejores resultados a la hora de predecir series "caóticas" usando backpropagation (Lapedes & Farber) [33] que mediante métodos lineales y polinomiales.
- **Modelado de sistemas**: permite modelar funciones de transferencia (Lapedes & Farmer).[34]
- **Filtro de ruido**: las redes neuronales artificiales son mejores preservando la estructura profunda y el detalle que los filtros tradicionales cuando eliminan el ruido. La primera aplicación profesional de las redes neuronales consistió en un filtro para eliminar ruido en las líneas telefónicas (Widrow, 1959)[19]

2.7.5 Otras aplicaciones

- **Modelado y predicción de indicadores económicos**: se obtienen mejores resultados que con cualquier otro método conocido. Se ha aplicado por ejemplo a la predicción de tasas de interés, déficits comerciales, precios de stock, etc.
- **Servocontrol**: compensación adaptativa de variaciones físicas en servomecanismos complicados como el control de ángulos y posiciones de los brazos de un robot.
- **Síntesis funcional**: gracias a la naturaleza interpolativa de las redes neuronales son aptas para la síntesis de funciones multidimensionales a partir de unos pocos ejemplos de entrenamiento. Por ejemplo, para estimar el alcance de un cañón en función de la inclinación del cañón, la velocidad del viento y la cantidad de explosivo.
- **Problemas de combinatoria**: las redes neuronales artificiales están ofreciendo ciertas esperanzas en el área de problemas algorítmicamente tan complejos como los NP-completos; pe el problema del viajante de comercio [35] (Hopfield, J. & Tank, D.)

Capítulo 3

Algoritmo de Aprendizaje Backpropagation

3.1 Introducción

La redes Multilayer Perceptron son redes feedforward multicapa de aprendizaje supervisado que utilizan como algoritmo de entrenamiento el algoritmo Backpropagation. De alta popularidad para la solución de problemas de clasificación y pronóstico, constituyen parte del objeto de estudio de esta tesis.

Las redes MLP han sido empleadas, de manera exitosa, en diversos campos de aplicación, tales como: minería de datos, predicción, optimización, identificación de sistemas, aplicaciones médicas, control y aproximaciones de funciones lineales y no lineales. La alta popularidad de las redes neuronales radica en su alta capacidad de resolver problemas demasiado complejos para las técnicas convencionales, y la popularidad de la red MLP se debe a que es capaz de aproximar cualquier función si se escoge una adecuada configuración para la red y un adecuado número de nodos en la capa oculta, lo que depende de la experiencia del desarrollador para determinar la configuración exacta de la red para cada aplicación.

El MLP es una generalización del Perceptron de Rosenblatt [36] de una capa; El perceptron está limitado a resolver solo problemas linealmente separables (Minsky & Papert) [37] que obviamente, resultan ser casos muy particulares en los problemas reales. El perceptrón multicapa de Rumelhart [15] es el exponente más típico de las redes neuronales artificiales con aprendizaje supervisado. El entrenamiento de estas redes, mediante backpropagation, se basa en la presentación sucesiva y de forma reiterada, de pares de vectores en las capas de entrada y salida (vectores entrada y salida deseada). La red crea un modelo a base de ajustar sus pesos en función de los vectores de entrenamiento, de forma que a medida que se pasan estos patrones, para cada vector de entrada la red producirá un valor de salida más similar al vector de salida esperado. Estas redes también se llaman de *backpropagation*, nombre que viene dado por el tipo de aprendizaje que utilizan.

3.2 Algoritmos de Aprendizaje Basados en la Derivada

Los principales algoritmos de aprendizaje de una red neuronal se basan en el cálculo del gradiente de la función de error, esto es, de la derivada de la función de error con respecto a los distintos parámetros ajustables de la red (en general, los pesos de las conexiones). Se trata de intentar encontrar el mínimo de la función de error mediante la búsqueda de un punto donde el gradiente se anule.

Una de las variantes basadas en el gradiente más utilizadas es el descenso por el gradiente. En el los sucesivos ajustes realizados a los pesos se hacen de forma individual para cada W_i en sentido opuesto al vector de gradiente $\partial E[n]/\partial W_i[n]$**:**

$$W_i[n+1] = W_i[n] - \alpha * \partial E[n]/\partial W_i[n] \qquad (2)$$

Donde α es un parámetro conocido como tasa de aprendizaje, que ha de tomar un valor convenientemente pequeño. Al pasar de la iteración *n* a la *n + 1*, el algoritmo aplica la corrección:

$$\Delta W_i[n] = W_i[n+1] - W_i[n] = -\alpha * \partial E[n]/\partial W_i[n] \qquad (3)$$

Para valores positivos muy pequeños de la tasa de aprendizaje y funciones de error globales, la formulación del algoritmo de descenso por el gradiente permite que la función de error decrezca en cada iteración. La tasa de aprendizaje α tiene, por tanto, una enorme influencia en la convergencia del método de descenso por el gradiente.

Si α es pequeña, el proceso de aprendizaje se desarrolla suavemente, pero la convergencia del sistema a una solución estable puede llevar un tiempo excesivo. Si α es grande, la velocidad de aprendizaje aumenta, pero existe el riesgo de que el proceso de aprendizaje diverja y el sistema se vuelva inestable.

Es habitual añadir un término de momento que en ocasiones puede acelerar el aprendizaje y reducir el riesgo de que el algoritmo se vuelva inestable. La nueva ecuación de actualización del parámetro ajustable W_i tiene la forma:

$$\Delta W_i[n] = W_i[n+1] - W_i[n] = -\alpha * \partial E[n]/\partial W_i[n] - \Delta\gamma * W_i[n+1] \qquad (4)$$

donde α es la tasa de aprendizaje y γ es la constante de momento.

Existen otros algoritmos de aprendizaje más sofisticados (por ejemplo, aquellos que consideran la información suministrada por las derivadas de segundo orden), que, en general, proporcionan mejores resultados que el descenso por el gradiente, a veces simplemente con una leve modificación, pero estos no serán tratados en este capítulo, dado que no son relevantes como objeto de estudio de la tesis.

3.3 Algoritmo de Aprendizaje Backpropagation

3.3.1 Antecedentes

La regla de aprendizaje del Perceptrón de Rosenblatt [36] y el algoritmo LMS de Widrow y Hoff [38] fueron diseñados para entrenar redes de una sola capa. Como se discutió anteriormente, estas redes tienen la desventaja que sólo pueden resolver problemas linealmente separables, fue esto lo que llevó al surgimiento de las redes multicapa para sobrepasar esta dificultad en las redes hasta entonces conocidas.

El primer algoritmo de entrenamiento para redes multicapa fue desarrollado por Paul Werbos en 1974 [39], éste se desarrolló en un contexto general, para cualquier tipo de redes, siendo las redes neuronales una aplicación especial, razón por la cual el algoritmo no fue aceptado dentro de la comunidad de desarrolladores de redes neuronales. Fue sólo hasta mediados de los años 80 cuando el algoritmo Backpropagation fue redescubierto al mismo tiempo por varios investigadores, David Rumelhart, Geoffrey Hinton y Ronal Williams, David Parker y Yann Le Cun. El algoritmo se popularizó cuando fue incluido en el libro Parallel Distributed Processing Group [25] por los sicólogos David Rumelhart y James McClelland. La publicación de éste trajo consigo un auge en las investigaciones con redes neuronales, siendo la Backpropagation una de las redes más ampliamente empleadas, aun actualmente.

Uno de los grandes avances logrados con la Backpropagation es que esta red aprovecha la naturaleza paralela de las redes neuronales para reducir el tiempo requerido por un procesador secuencial para determinar la correspondencia entre unos patrones dados. Además el tiempo de desarrollo de cualquier sistema que se esté tratando de analizar se puede reducir como consecuencia de que la red puede aprender el algoritmo correcto sin que alguien tenga que deducir por anticipado el algoritmo en cuestión.

La mayoría de los sistemas actuales de cómputo se han diseñado para llevar a cabo funciones matemáticas y lógicas a una velocidad que resulta asombrosamente alta para el ser humano. Sin embargo la destreza matemática no es lo que se necesita para solucionar problemas de reconocimiento de patrones en entornos ruidosos, característica que incluso dentro de un espacio de entrada relativamente pequeño, puede llegar a consumir mucho tiempo. El problema es la naturaleza secuencial del propio computador; el ciclo tomar – ejecutar de la naturaleza Von Neumann sólo permite que la máquina realice una operación a la vez. En la mayoría de los casos, el tiempo que necesita la máquina para llevar a cabo cada instrucción es tan breve (típicamente una millonésima de segundo) que el tiempo necesario para un programa, así sea muy grande, es insignificante para los usuarios. Sin embargo, para aquellas aplicaciones que deban explorar un gran espacio de entrada o que intentan correlacionar todas las permutaciones posibles de un conjunto de patrones muy complejo, el tiempo de computación necesario se hace bastante grande. Lo que se necesita es un nuevo sistema de procesamiento que sea capaz de examinar todos los patrones en paralelo. Idealmente ese sistema no tendría que ser programado explícitamente, lo que haría es adaptarse a sí mismo para aprender la relación entre un conjunto de patrones dado como ejemplo y ser capaz de aplicar la misma relación a nuevos patrones de entrada. Este sistema debe estar en capacidad de concentrarse en las características de una entrada arbitraria que se asemeje a otros patrones vistos previamente, sin que ninguna señal de ruido lo afecte. Este sistema fue el gran aporte de la red de propagación inversa, Backpropagation.

La Backpropagation es un tipo de red de aprendizaje supervisado, que emplea un ciclo propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, éste se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta sólo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que

describa su contribución relativa al error total. Basándose en la señal de error percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento.

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento. Y a la inversa, las unidades de las capas ocultas tienen una tendencia a inhibir su salida si el patrón de entrada no contiene la característica para reconocer, para la cual han sido entrenadas.

Varias investigaciones han demostrado que, durante el proceso de entrenamiento, la red Backpropagation tiende a desarrollar relaciones internas entre neuronas con el fin de organizar los datos de entrenamiento en clases. Esta tendencia se puede extrapolar, para llegar a la hipótesis consistente en que todas las unidades de la capa oculta de una Backpropagation son asociadas de alguna manera a características específicas del patrón de entrada como consecuencia del entrenamiento. Lo que sea o no exactamente la asociación puede no resultar evidente para el observador humano, lo importante es que la red ha encontrado una representación interna que le permite generar las salidas deseadas cuando se le dan las entradas, en el proceso de entrenamiento. Esta misma representación interna se puede aplicar a entradas que la red no haya visto antes, y la red clasificará estas entradas según las características que compartan con los ejemplos de entrenamiento.

3.3.2 Estructura de la Red Multilayer Perceptron.

Este tipo de redes tienen una capa de entrada, una o más ocultas y otra de salida. La gráfica 3.1 muestra un diagrama con su topología típica. La información se propaga de capa en capa (de abajo hacia arriba), por medio de las conexiones entre los nodos de cada capa.

Los nodos de las capas intermedias calculan la suma de los productos de los valores de los nodos de entrada, y los valores de los pesos asociados a las conexiones entre ambas capas de neuronas. Cada una de los nodos de la capa intermedia, utiliza esta suma para calcular el valor de una función sigmoidal.

- $X_p = \left(X_{1p}, X_{2p}, ..., X_{np}\right)$: *P*-ésimo vector de entrada del conjunto de entrenamiento
- W_{kj} : peso de la conexión del nodo oculto *j* al nodo de salida *k*
- W_{ji} : peso de la conexión del nodo de entrada *i* al nodo oculto *j*.

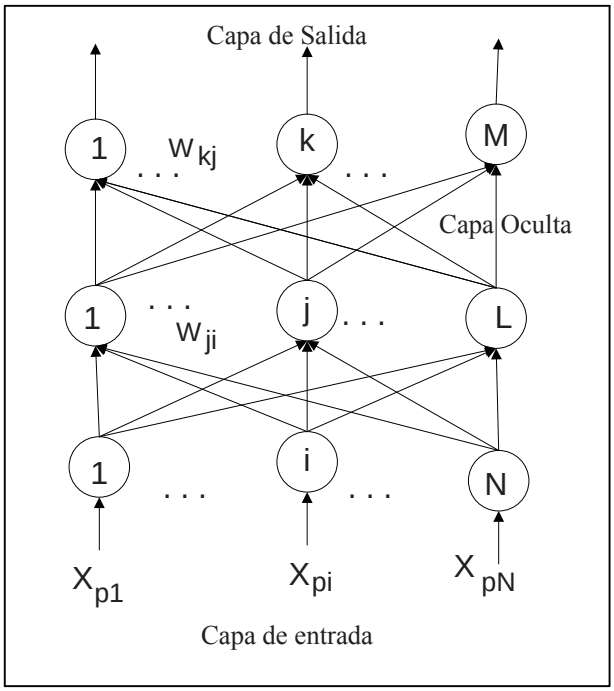


Figura 3.1: Arquitectura de una red MLP

El algoritmo de aprendizaje del MLP es por excelencia el backpropagation y algunas de sus variantes, como los algoritmos de segundo orden Newton-Raphson [40] y Levenberg-Marquardt [41] y [42], que se utilizan para mejorar su velocidad de convergencia, aunque tienen el inconveniente de que ocupan mucha más memoria.

Como ya se ha indicado, las redes MLP son las más ampliamente usadas, principalmente porque tiene la capacidad de actuar como un aproximador universal de funciones. Esta propiedad convierte a las redes MLP en ideales para ser utilizadas en una gran variedad de problemas de clasificación de alta complejidad, pronóstico y control, por ejemplo, procesamiento de imágenes [29], filtraje de ruido de electrocardiogramas [31], el control de maquinaria industrial, previsión en series de tiempo y muchos otros.

3.3.3 Teorema de Aproximación Universal

La gran ventaja que presentan las MLP radica principalmente a que es capaz de actuar como un aproximador universal de funciones [1]. Más concretamente, una red conteniendo al menos una capa oculta con suficientes nodos no lineales puede aprender cualquier tipo de función o relación continua entre un grupo de variables de entrada y salida. Esta propiedad convierte a los MLP en herramientas de propósito general flexibles y no lineales. De cualquier forma el teorema de aproximación universal dice así:

Dada una función $f:[0,1]^n \rightarrow R^n$, y cualquier $\varepsilon > 0$, existe una red multicapa feedforward de tres capas (la de entrada, la de salida y la capa oculta) que puede aproximar f con error cuadrático medio menor o igual a ε, siendo el error cuadrático medio:

$$E = \sum_{\mu i}^{p} [fi(\varepsilon^{\mu}) - O_i(\varepsilon^{\mu}, W, w)]^2 \qquad (5)$$

Donde ε^{μ} es el vector que contiene los valores para las unidades de entrada del ejemplo (o patrón) μ, y la función que se desea aproximar es conocida en finitos puntos $f(\varepsilon^{\mu})$.

Sin embargo, este resultado no permite predecir cual debe ser la arquitectura de la red (cantidad de nodos en la capa oculta) y en general no hay recetas.

Se debe tener en cuenta que el número de ejemplos que se le presenta a la red durante su entrenamiento, debe ser por lo menos *diez veces* mayor que el número de parámetros libres de la misma.

Si bien una única capa oculta es suficiente, puede resultar en algunos casos, que el número de nodos requerido sea demasiado grande; entonces es conveniente diseñar redes con más de una capa oculta y tal que la suma de nodos de todas las capas resulte menor, de manera de disminuir el volumen de los datos procesados.

3.3.4 Regla de Aprendizaje

El algoritmo Backpropagation para redes multicapa es una generalización del algoritmo LMS, ambos algoritmos realizan su labor de actualización de pesos y ganancias con base en el error medio cuadrático. La red Backpropagation trabaja bajo aprendizaje supervisado y por tanto necesita un set de entrenamiento que le describa cada salida y su valor de salida esperado de la siguiente forma:

$$\{X_1, Y_1\}, \{X_2, Y_2\} \ldots \{X_n, Y_n\} \qquad (6)$$

Donde X_n es una entrada a la red y Y_n es la correspondiente salida deseada para el patrón n-ésimo. El algoritmo debe ajustar los parámetros de la red para minimizar el error medio cuadrático. El entrenamiento de una red neuronal multicapa se realiza mediante un proceso de aprendizaje, para realizar este proceso se debe inicialmente tener definida la topología de la red esto es: número de neuronas en la capa de entrada el cual depende del número de componentes del vector de entrada, cantidad de capas ocultas y número de neuronas de cada una de ellas, número de neuronas en la capa de la salida el cual depende del número de componentes del vector de salida o patrones objetivo y funciones de transferencia requeridas en cada capa, con base en la topología escogida se asignan valores iniciales a cada uno de los parámetros que conforma la red.

Es importante recalcar que no existe una técnica para determinar el número de capas ocultas, ni el número de neuronas que debe contener cada una de ellas para un problema específico, esta elección es determinada por la experiencia del diseñador, el cual debe cumplir con las limitaciones de tipo computacional.

Cada patrón de entrenamiento se propaga a través de la red y sus parámetros para producir una respuesta en la capa de salida, la cual se compara con los patrones objetivo o salidas deseadas para calcular el error en el aprendizaje, este error marca el camino mas adecuado para la actualización de los pesos y ganancias que al final del entrenamiento producirán una respuesta satisfactoria a todos los patrones de entrenamiento, esto se logra minimizando el error medio cuadrático en cada iteración del proceso de aprendizaje.

La deducción matemática de este procedimiento se realizará para una red con una capa de entrada, una capa oculta y una capa de salida y luego se generalizará para redes que tengan más de una capa oculta.

Es importante aclarar que en la figura 3.2:

- N : el número de patrones aplicados a la capa de entrada de la red. También puede verse como el número de neuronas de la capa de entrada.
- p : subíndice de la tupla del patrón de entrenamiento.
- L : Número de neuronas de la capa oculta de la red.
- M : Número de neuronas de la capa de salida de la red.
- $X_p = \left(X_{1p}, X_{2p}, ..., X_{Np}\right)$: *P*-ésimo vector de *entrada* del conjunto de entrenamiento.
- $Y_p = \left(Y_{1p}, Y_{2p}, ..., Y_{Np}\right)$: P-ésimo vector de *salida* del conjunto de entrenamiento.
- W_{kj} : peso de la conexión del nodo oculto *j* al nodo de salida *k.*
- W_{ji} el peso de la conexión del nodo de entrada *i* al nodo oculto *j*
- Los símbolos θ son los términos de tendencia o bias; las unidades de tendencia proporcionan un valor de entrada ficticio igual a 1 en una conexión del peso de tendencia, como cualquier otro peso.

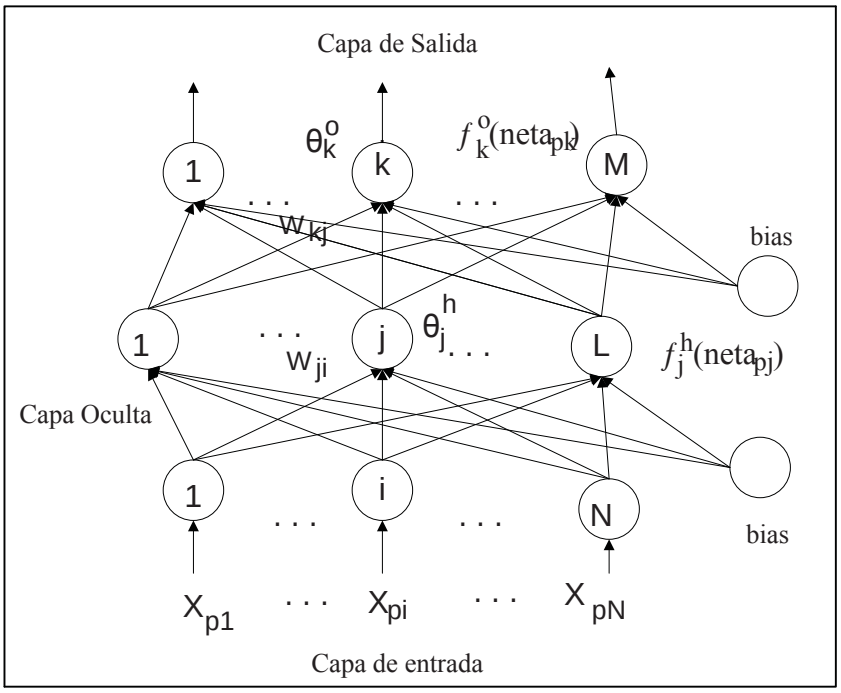


Figura 3.2 Arquitectura de una red MLP de tres capas

Cuando se le presenta a la red una patrón de entrenamiento, este se propaga a través de las conexiones existentes produciendo una entrada *neta* en cada una las neuronas de la siguiente capa, la entrada neta a la neurona *j* de la siguiente capa debido a la presencia de un patrón de entrenamiento en la entrada esta dada por la ecuación (7), nótese que la entrada neta es el valor justo antes de pasar por la función de transferencia.

La entrada neta de la *j*-ésima unidad oculta es:

$$neta_{pj}^{h} = \sum_{i=1}^{n} W_{ji}^{h} X_{pi} + \theta_{j}^{h} \qquad (7)$$

Se calculan las salidas de la capa oculta, donde *f* hace referencia a la función de transferencia. Nótese que el índice *h* hace referencia a la capa oculta.

$$Z_{pj} = f_{j}^{h}\left(neta_{pj}^{h}\right) \qquad (8)$$

Las salidas Z_{pj} de las neuronas de la capa oculta (de L componentes) son las entradas a los pesos de conexión de la capa de salida, comportamiento que es descrito por la ecuación. (9)

$$neta_{pk}^{h} = \sum_{j=1}^{L} W_{kj}^{o} Z_{pj} + \theta_{k}^{o} \qquad (9)$$

Se calcula otra vez la salida aplicando una función de transferencia *f*:

$$O_{pk} = f\left(neta_{pk}^{o}\right) \qquad (10)$$

Se calculan los términos de error para las unidades de salida comparando la salida obtenida con la salida deseada.

$$\delta_{pk}^{o} = \left(Y_{pk} - O_{pk}\right) f'\left(neta_{pk}^{o}\right) \qquad (11)$$

Donde: Y_{pk} es la salida deseada y O_{pk} es la salida real de la red

El error debido a cada patrón *p* propagado está dado por (11):

E_{p}^{2}: Error medio cuadrático para cada patrón de entrada *p*

$\delta^{o}{}_{pk}$: Error en la neurona *k* de la capa de salida con *M* neuronas

$$E_{p}^{2} = \tfrac{1}{2} \sum_{k=1}^{M} \left(\delta_{k}\right)^{2} \qquad (12)$$

Este proceso se repite para el número total de patrones de entrenamiento (*N*). Para un proceso de aprendizaje exitoso el objetivo del algoritmo es actualizar todos los pesos y ganancias de la red minimizando el error medio cuadrático total descrito en (13):

$$E_{p} = \sum^{N} E_{p}^{2} \qquad (13)$$

Error total en el proceso de aprendizaje en una iteración luego de haber presentado a la red los *N* patrones de entrenamiento

El error que genera una red neuronal en función de sus pesos, genera un espacio de *n* dimensiones, donde *n* es el número de pesos de conexión de la red, al evaluar el gradiente del error en un punto de esta superficie se obtendrá la dirección en la cual la función del error tendrá

un mayor crecimiento, como el objetivo del proceso de aprendizaje es minimizar el error debe tomarse la dirección negativa del gradiente para obtener el mayor decremento del error y de esta forma su minimización, condición requerida para realizar la actualización de la matriz de pesos en el algoritmo Backpropagation:

$$W_{k+1} = W_k - \alpha \nabla E_p{}^2 \tag{14}$$

El gradiente negativo de $E_p{}^2$ se denotará como $\nabla E_p{}^2$ y se calcula como la derivada del error respecto a todos los pesos de la red.

En la capa de salida el gradiente negativo del error con respecto a los pesos es:

$$-\frac{\partial E_p{}^2}{\partial W_{kj}{}^o} = -\frac{\partial}{\partial W_{kj}{}^o}\left(1/2 \sum_{k=1}^{L} \left((Y_k - O_k) \right)^2 \right) = (Y_k - O_k)\frac{\partial O_k{}^o}{\partial W_{kj}{}^o} \tag{15}$$

$-\frac{\partial E_p{}^2}{\partial W_{kj}{}^o}$: Componente del gradiente $\nabla E_p{}^2$ respecto al peso de la conexión de la neurona de la capa de salida y la neurona *j* de la capa oculta $W_{kj}{}^o$

$\frac{\partial O_p{}^M}{\partial W_{kj}{}^M}$: Derivada de la salida de la neurona *k* de la capa de salida respecto al peso $W_{kj}{}^o$

Para calcular $\frac{\partial O_k{}^o}{\partial W_{kj}{}^o}$ se debe utilizar la regla de la cadena, pues el error no es una función explícita de los pesos de la red, de la ecuación (10) puede verse que la salida de la red O_{pk} esta explícitamente en función de $neta_{pk}^h$ y de la ecuación (9) puede verse que $neta_{pk}^h$ esta explícitamente en función de $W_{kj}{}^o$, considerando esto se genera la ecuación:

$$\frac{\partial O_k{}^o}{\partial W_{kj}{}^o} = \frac{\partial O_k{}^o}{\partial W_k{}^o} \times \frac{\partial neta_k{}^o}{\partial W_{kj}{}^o} \tag{16}$$

Tomando la ecuación (16) y reemplazándola en la ecuación (15) se obtiene:

$$-\frac{\partial E_p{}^2}{\partial W_{kj}{}^o} = (Y_k - O_k)\frac{\partial O_k{}^o}{\partial W_k{}^o} \times \frac{\partial neta_k{}^o}{\partial W_{kj}{}^o} \tag{17}$$

$\frac{\partial neta_k{}^o}{\partial W_{kj}{}^o}$: Derivada de la entrada neta a la neurona *k* de la capa de salida respecto a los pesos de la conexión entre las neuronas de la última capa oculta y la capa de salida

$\frac{\partial O_k{}^o}{\partial W_k{}^o}$: Derivada de la salida de la neurona *k* de la capa de salida respecto a su entrada neta.

Reemplazando en la ecuación (17) las derivadas de las ecuaciones (10) y (11) se obtiene:

$$-\frac{\partial E_p{}^2}{\partial W_{kj}{}^o} = \left(Y_k - O_k\right) \times f'\left(neta_k{}^o\right) \times Z^h{}_{pj} \qquad (18)$$

Como se observa en la ecuación (17) las funciones de transferencia utilizadas en este tipo de red deben ser continuas para que su derivada exista en todo el intervalo, ya que el término $f'\left(neta_k{}^o\right)$ es requerido para el cálculo del error.

Las funciones de transferencia *f* más utilizadas y sus respectivas derivadas son las siguientes:

$$\textit{Logsig}: \; f(n) = \frac{1}{1+e^{-n}} \qquad f'(n) = f(n)(1 - f(n)) \qquad (19)$$

$$\textit{Tansig}: \; f(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}} \qquad f'(n) = 1 - \left(f(n)^2\right) \qquad (20)$$

$$\textit{Purelin}: \quad f(n) = n \qquad f'(n) = 1 \qquad (21)$$

De la ecuación (18), los términos del error para las neuronas de la capa de salida están dados por la ecuación (22), la cual se le denomina comúnmente sensitividad de la capa de salida.

$$\delta_k{}^o = \left(Y_k - O^o{}_k\right) \times f'\left(neta^o{}_k\right) \qquad (22)$$

Este algoritmo se denomina Backpropagation o de propagación inversa debido a que el error se propaga de manera inversa al funcionamiento normal de la red, de esta forma, el algoritmo encuentra el error en el proceso de aprendizaje desde las capas más internas hasta llegar a la entrada; con base en el cálculo de este error se actualizan los pesos y ganancias de cada capa.

Después de conocer (22) se procede a encontrar el error en la capa oculta el cual esta dado por:

$$\frac{-\partial E^2}{\partial W_{kj}{}^h} = -\frac{\partial}{\partial W_{kj}} \times \left(1 \Big/ 2\sum_{k=1}^{L}\left(Y_k - O^o{}_k\right)^2\right) = \sum_{k=1}^{L}\left(Y_k - O^o{}_k\right) \times \frac{\partial O^o{}_k}{\partial W_{ji}} \qquad (23)$$

Para calcular el último término de la ecuación (23) se debe aplicar la regla de la cadena en varias ocasiones como se observa en la ecuación (24) puesto que la salida de la red no es una función explícita de los pesos de la conexión entre la capa de entrada y la capa oculta

$$\frac{\partial O^{o}{}_{k}}{\partial W^{o}{}_{ji}} = \frac{\partial O_{k}{}^{o}}{\partial neta_{k}{}^{o}} \times \frac{\partial neta_{k}{}^{o}}{\partial O_{k}{}^{h}} \times \frac{\partial O_{k}{}^{h}}{\partial neta_{j}{}^{h}} \times \frac{\partial neta_{j}{}^{h}}{\partial W_{ji}} \qquad (24)$$

Todas los términos de la ecuación (25) son derivados respecto a variables de las que dependan explícitamente, reemplazando (22) en (21) se tiene:

$$-\frac{\partial E^{2}}{\partial W_{ji}{}^{h}} = \sum_{k=1}^{L}\left(Y_{k} - O^{o}{}_{k}\right) \times \frac{\partial O_{k}{}^{o}}{\partial neta_{k}{}^{o}} \times \frac{\partial neta_{k}{}^{o}}{\partial O_{k}{}^{h}} \times \frac{\partial O_{k}{}^{h}}{\partial neta_{j}{}^{h}} \times \frac{\partial neta_{j}{}^{h}}{\partial W_{ji}} \qquad (25)$$

Tomando las derivadas de las ecuaciones (7) (8) (9) y reemplazándolas en la ecuación (24) se obtiene la expresión del gradiente del error en la capa oculta

$$-\frac{\partial E^{2}}{\partial W_{ji}{}^{h}} = \sum_{k=1}^{L}\left(Y_{k} - O_{k}{}^{o}\right) \times f'^{o}\left(neta_{k}{}^{o}\right) \times W_{kj}{}^{o} \times f'^{h}\left(neta_{j}{}^{h}\right) \times Z_{pj} \qquad (26)$$

Reemplazando la ecuación (22) en (25) se tiene:

$$-\frac{\partial E^{2}}{\partial W_{ji}{}^{h}} = \sum_{k=1}^{L}\delta_{k}{}^{o} \times W_{kj}{}^{o} \times f'^{h}\left(neta_{j}{}^{h}\right) \times Z_{pj} \qquad (27)$$

Los términos del error para cada neurona de la capa oculta están dados por la ecuación (28), este término también se denomina sensitividad de la capa oculta.

$$\delta_{pj}^{h} = f'\left(neta_{pj}^{h}\right) * \sum_{k=1}^{M}\delta_{pk}^{o} W_{kj}^{o} \qquad (28)$$

Los términos de error de los nodos ocultos se calculan antes de que hayan sido actualizados los pesos de conexión en la capa de salida.

Se actualizan los pesos de la capa de salida:

$$W_{pk}^{o}\left(t+1\right) = W_{kj}^{o}\left(t\right) + \alpha * \delta_{pk}^{o} Z_{pj} \qquad (29)$$

Es decir, la ecuación del ajuste de pesos queda:

$$W_{kj}^{o}\left(t\right) = \alpha * \delta_{pk}^{o} Z_{pj} \qquad (30)$$

Como se dijo anteriormente α es la tasa de aprendizaje, es positivo y suele ser menor que uno. Luego de encontrar el valor del gradiente del error se procede a actualizar los pesos de todas las capas empezando por la de salida, para la capa de salida la actualización de pesos está dada por (31) y (32).

$$W_{kj}^{o}\left(t+1\right) = W_{kj}^{o}\left(t\right) + \alpha * \delta_{k}^{o} \qquad (31)$$

Que queda:

$$\Delta W_{ji}^{t}(t) = \alpha * \delta_{ji}^{h} X_i \qquad (32)$$

Luego de actualizar los pesos y ganancias de la capa de salida se procede a actualizar los pesos y ganancias de la capa oculta mediante las ecuaciones:

$$W_{ji}^{h}(t+1) = W_{ji}^{h}(t) + \alpha * \delta_{pj}^{h} X_i \qquad (33)$$

Como se ve el algoritmo Backpropagation utiliza la misma técnica de aproximación en pasos descendientes que emplea el algoritmo LMS, la única complicación está en el cálculo del gradiente, el cual es un término indispensable para realizar la propagación de la sensitividad.

En las técnicas de gradiente descendiente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos; esto se debe a que se tiene una información local de la superficie y no se sabe lo lejos o lo cerca que se está del punto mínimo, con incrementos grandes, se corre el riesgo de pasar por encima del punto mínimo, con incrementos pequeños, aunque se tarde más en llegar, se evita que esto ocurra. El elegir un incremento adecuado influye en la velocidad de convergencia del algoritmo, esta velocidad se controla a través de la rata de aprendizaje α , la que por lo general se escoge como un número pequeño, para asegurar que la red encuentre una solución. Un valor pequeño de α significa que la red tendrá que hacer un gran número de iteraciones, si se toma un valor muy grande, los cambios en los pesos serán muy grandes, avanzando muy rápidamente por la superficie de error, con el riesgo de saltar el valor mínimo del error y estar oscilando alrededor de él, pero sin poder alcanzarlo.

Es recomendable aumentar el valor de α a medida que disminuye el error de la red durante la fase de entrenamiento, para garantizar así una rápida convergencia, teniendo la precaución de no tomar valores demasiado grandes que hagan que la red oscile alejándose demasiado del valor mínimo. Algo importante que debe tenerse en cuenta, es la posibilidad de convergencia hacia alguno de los mínimos locales que pueden existir en la superficie del error del espacio de pesos como se ve en la gráfica 3.3.

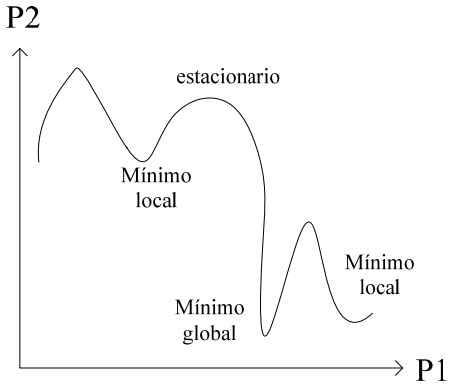


Figura 3.3 Superficie típica de error

En el desarrollo matemático que se ha realizado para llegar al algoritmo Backpropagation, no se asegura en ningún momento que el mínimo que se encuentre sea global, una vez la red se asiente en un mínimo sea local o global cesa el aprendizaje, aunque el error siga siendo alto. En

todo caso, si la solución es admisible desde el punto de vista del error, no importa si el mínimo es local o global o si se ha detenido en algún momento previo a alcanzar un verdadero mínimo.

3.4 Algoritmo de Entrenamiento Complex - Backpropagation

El Algoritmo Backpropagation es extendido al dominio complejo [43] (Complex-Backpropagation, ahora en adelante CBP) y puede ser usado para entrenar redes neuronales en las cuales las entradas, pesos, funciones de activación y salidas son valuadas de forma compleja [44]. Una circunstancia especial la constituyen las funciones de activación complejas, que tienen singularidades altamente indeseables tal como se indica en [45] y [46].

Es sabido que las señales transmitidas por un enlace inalámbrico se representan comúnmente en el plano complejo, ahora bien, el algoritmo (real) BP puede ser usado para entrenar una red neuronal convencional que utiliza, en teoría, la mismas señales entrada-salida sin envolver a los números complejos y que obtiene una performance aceptable. Por tanto, resulta interesante crear una red neuronal compleja, como indica Koutsougeras [47], para compensar la distorsión no lineal en canales inalámbricos y observar su comportamiento en comparación a la red neuronal real.

Las ecuaciones de Cauchy-Riemann constituyen una condición necesaria para la derivabilidad de funciones complejas, aunque no suficiente, pero si sirve para comprobar que una función compleja sea analítica en un punto $z \in C$ donde $f(z) = u(x,y) + iv(x,y)$, (ya que se puede descomponer en la suma de dos funciones reales *u* y *v)* y siendo $z = x + iy$, se tiene que:

$$f'(z) = u_x + iv_x = v_y - iu_y \qquad (34)$$

Igualando las partes reales e imaginarias en (17), se obtienen las ecuaciones de Cauchy-Riemann: $u_x = v_y, v_x = -u_y$, nótese que la ecuación también puede ser expresada como:

$$f'(z) = f_x = if_y \qquad (35)$$

Para la función de activación compleja $f(z)$, el error cuadrático medio en la capa de salida (ECM de la red) puede ser escrito como:

$$E = 1/2 \sum_k |e_k|^2 \quad , \quad e_k = d_k - o_k \qquad (36)$$

$$O_k \equiv f(z_k) \equiv u_k + iv_k \quad , \quad z_k \equiv X_k + iy_k \equiv \sum_k W_{kj} X_{pj}$$
$$\equiv \sum_k (W_{kjR} + iW_{kjI})(X_{kjR} + iX_{pjI}) \qquad (37)$$

d_k es la salida deseada del k-ésimo nodo y O_k es la salida efectivamente obtenida del k-ésimo nodo. Los subíndices R e I indican el componente real e imaginario, respectivamente, el subíndice i corresponde al i-ésimo nodo de la capa oculta conectado con el k-ésimo nodo de la capa de salida, y el índice p, indica la p-ésimo patrón de entrada.

La regla de adaptación del algoritmo BP requiere el cálculo de gradiente $\partial E / \partial W_{kj}$. El gradiente de la función de error con respecto a los componentes reales e imaginarios de W_{kj} pueden ser escritos como:

$$\partial E / \partial W_{kj} \equiv \Delta_{w_{kj}} E \equiv \partial E / \partial W_{kjR} + i\partial E / \partial W_{kjI} \qquad (38)$$

Y usando la regla de la cadena:

$$\frac{\partial E}{\partial W_{kjR}} = \frac{\partial E}{\partial u_k}\left(\frac{\partial u_k}{\partial x_k}\frac{\partial x_k}{\partial W_{kjR}} + \frac{\partial u_k}{\partial y_k}\frac{\partial y_k}{\partial W_{kjR}}\right) + \frac{\partial E}{\partial v_k}\left(\frac{\partial v_k}{\partial x_k}\frac{\partial x_k}{\partial W_{kjR}} + \frac{\partial v_k}{\partial y_k}\frac{\partial y_k}{\partial W_{kjR}}\right) \qquad (39)$$

$$\frac{\partial E}{\partial W_{kjI}} = \frac{\partial E}{\partial u_k}\left(\frac{\partial u_k}{\partial x_k}\frac{\partial x_k}{\partial W_{kjI}} + \frac{\partial u_k}{\partial y_k}\frac{\partial y_k}{\partial W_{kjI}}\right) + \frac{\partial E}{\partial v_k}\left(\frac{\partial v_k}{\partial x_k}\frac{\partial x_k}{\partial W_{kjI}} + \frac{\partial v_k}{\partial y_k}\frac{\partial y_k}{\partial W_{jkI}}\right) \qquad (40)$$

Definiendo $\delta_k \equiv -\partial E / \partial u_k - i\partial E / \partial v_k$, $\delta_{kR} \equiv -\partial E / \partial u_k$ y $\partial_{kI} \equiv -\partial E / \partial v_k$, ahora, usando las derivadas parciales de (38):

$$\frac{\partial x_k}{\partial W_{kjR}} = X_{kjR} \quad , \quad \frac{\partial y_k}{\partial W_{kjR}} = X_{kjI} \quad , \quad \frac{\partial x_k}{\partial W_{kjI}} = -X_{kjI} \, , \quad \frac{\partial y_k}{\partial W_{kjI}} = X_{kjR} \qquad (41)$$

Así (40) y (41) pueden simplificarse como:

$$\frac{\partial E}{\partial W_{kjR}} = -\delta_{kR}\left(\frac{\partial u_k}{\partial x} X_{kjR} + \frac{\partial u_k}{\partial y} X_{kjI}\right) - \delta_{kI}\left(\frac{\partial v_k}{\partial x} X_{kjR} + \frac{\partial v_k}{\partial y} X_{kjI}\right) \qquad (42)$$

$$\frac{\partial E}{\partial W_{kjI}} = -\delta_{kR}\left(\frac{\partial u_k}{\partial x}\left(-X_{kjI}\right) + \frac{\partial u_k}{\partial y} X_{kjR}\right) - \delta_{kI}\left(\frac{\partial v_k}{\partial x}\left(-X_{kjI}\right) + \frac{\partial v_k}{\partial y} X_{kjR}\right) \qquad (43)$$

Combinando (42) y (43) el gradiente de la función de error queda:

$$\Delta_{W_{kj}} E = -\overline{X}_{kj}\left\{\left(\frac{\partial u_k}{\partial x} + i\frac{\partial u_k}{\partial y}\right)\delta_{kR} + \left(\frac{\partial v_k}{\partial x} + i\frac{\partial v_k}{\partial y}\right)\delta_{kI}\right\} \qquad (44)$$

Una forma más compacta de representar la función de gradiente del error, puede obtenerse aplicando las ecuaciones de Cauchy-Riemann:

$$\Delta_{W_{kj}} E = -\overline{X}_{kj}\left(\frac{\partial \overline{f}}{\partial x}\partial_{kR} - i\left(-i\frac{\partial \overline{f}}{\delta y}\right)\delta_{kI}\right) = \overline{X}_{kj}\,\overline{f}'(z)\overline{\delta}_k \qquad (45)$$

Esto no tiene mayor importancia salvo por el conjugado de cada término, que es idéntico a la función de gradiente del error en la versión real del algoritmo BP tal como podría esperarse.

La actualización de los pesos complejos ΔW_{kj} es proporcional al gradiente negativo:

$$\Delta W_{kj} = \alpha \overline{X}_{kj} \overline{f}'(z) \overline{\delta_k} \tag{46}$$

Donde α es la tasa de aprendizaje real y positivo. Cuando el peso complejo pertenece a un nodo de salida:

$$\delta_k = e_k = d_k - O_k \tag{47}$$

Y cuando se trata de W_{ji} , es decir, pesos de la conexión al j-ésimo nodo de la capa oculta, la salida de la red Z_j del nodo j es lo mismo que en la ecuación (38):
$Z_j = x_j + iy_j = \sum_j (u_j + iv_j)(W_{jiR} + iW_{jiI})$, donde el índice i pertenece a la entrada i-ésima que alimenta al nodo j-ésimo. Ahora, usando la regla de la cadena:

$$\partial_{jR} = -\frac{\partial E}{\partial u_j} = -\sum_k \frac{\partial E}{\partial u_k}\left(\frac{\partial u_k}{\partial x_k}\frac{\partial x_k}{\partial u_j} + \frac{\partial u_k}{\partial y_k}\frac{\partial y_k}{\partial u_j}\right) - \sum_k \frac{\partial E}{\partial v_k}\left(\frac{\partial v_k}{\partial x_k}\frac{\partial x_k}{\partial u_j} + \frac{\partial v_k}{\partial y_k}\frac{\partial y_k}{\partial u_j}\right)$$
$$= \sum_k \delta_{kR}\left(\frac{\partial u_j}{\partial x_k}W_{jiR} + \frac{\partial u_j}{\partial y_k}W_{jiI}\right) + \sum_k \delta_{kI}\left(\frac{\partial v_j}{\partial x_k}W_{jiR} + \frac{\partial v_j}{\partial y_k}W_{jiI}\right) \tag{48}$$

Análogamente:

$$\partial_{jI} = -\frac{\partial E}{\partial v_j} = -\sum_k \frac{\partial E}{\partial u_k}\left(\frac{\partial u_k}{\partial x_k}\frac{\partial x_k}{\partial v_j} + \frac{\partial u_k}{\partial y_k}\frac{\partial y_k}{\partial v_j}\right) - \sum_k \frac{\partial E}{\partial v_k}\left(\frac{\partial v_k}{\partial x_k}\frac{\partial x_k}{\partial v_j} + \frac{\partial v_k}{\partial y_k}\frac{\partial y_k}{\partial v_j}\right)$$
$$= \sum_k \delta_{kR}\left(\frac{\partial u_k}{\partial x_k}(-W_{jiI}) + \frac{\partial u_k}{\partial y_k}W_{jiR}\right) + \sum_k \delta_{kI}\left(\frac{\partial v_k}{\partial x_k}(-W_{jiI}) + \frac{\partial v_k}{\partial y_k}W_{jiR}\right) \tag{49}$$

Usando las mismas derivadas parciales que ayudaron a establecer (38) y (39) y combinando (42) y (43) la siguiente expresión es obtenida similarmente para la función de actualización de peso (24) usando las ecuaciones de Cauchy Riemann:

$$\delta_j = \delta_{jR} + i\delta_{jI} = \sum_k \overline{W}_{kj} \overline{f}'(z_j) \overline{\delta_k} \tag{50}$$

Considerando la función de activación compleja $f(z) = u(x) + iv(y)$ y esto quiere decir $u_y = v_x = 0$ para el algoritmo backpropagation complejo. Removiendo estos términos ceros de (25) se obtienen las siguientes actualizaciones de pesos complejos:

$$\Delta W_{kj} = \alpha \overline{X}_{jk}\left(\frac{\partial u_k}{\partial x}\delta_{kR} + i\frac{\partial v_k}{\partial y}\delta_{kI}\right) \tag{51}$$

Como anteriormente, para el nodo de la capa de salida, $\delta_k = d_k - O_k$, y para la entrada y la capa oculta:

$$\delta_j = \sum_k \overline{W}_{jI}\left(\frac{\partial u_j}{\partial x_k}\delta_{kR} + i\frac{\partial v_j}{\partial y_k}\delta_{kI}\right) \qquad (52)$$

De esta manera, se tienen todas las ecuaciones analíticas que conforman el algoritmo Backpropagation en el dominio complejo: se comprende que los pesos, entradas y salidas se definen en los complejos. Ahora bien, el entrenamiento de una red feed-forward con el algoritmo CBP debería hacerse de manera similar al (real) BP. Primero los pesos deberían ser inicializados a errores randómicos complejos. Cada vector de entrada es propagado a través de la red y es calculado el correspondiente error cuadrático medio (el forward del algoritmo), entonces el error es retropropagado (Backward) por cada nodo de la red y los pesos son ajustados buscando reducir ese error.

3.5 Tasa de Aprendizaje Adaptativa

Se entiende comúnmente por entrenamiento eficiente de la red neuronal la minimización de la función del error y se sabe que ésta depende del valor de los pesos de la red. Esta perspectiva resulta ventajosa para el desarrollo de algoritmos de entrenamiento efectivo, porque el problema de aproximar una función es bien conocido en el campo del análisis numérico, sin embargo, a veces esto no es suficiente para asegurar una rápida convergencia de la red. Para esto, existen métodos para ajustar uno de los parámetros más significativos de la red: la tasa de aprendizaje.

En [48] (Descenso Estocástico con Aprendizaje Adaptativo) se propone un esquema de adaptación que explota la información relacionada al gradiente actual tanto como al valor de los gradientes previos, lo cual, por cierto , provee a la red de estabilización en el valor de la tasa de aprendizaje adaptativa y ayuda al gradiente descendente a exhibir rápida convergencia y un alto porcentaje de éxito.

El algoritmo es el siguiente:

$$\alpha^{k+1} = \alpha^k + \gamma_1\left\langle \nabla \mathrm{E}_{p-1}\left(w^{k-1}\right), \nabla \mathrm{E}_p\left(w^k\right)\right\rangle + \gamma_2\left\langle \nabla \mathrm{E}_{p-2}\left(w^{k-2}\right), \nabla \mathrm{E}_{p-1}\left(w^{k-1}\right)\right\rangle \quad (53)$$

Donde α es el valor de la tasa de aprendizaje, γ_1 e γ_2 son constantes positivas, p indica el número de patrón de la red neuronal, y $\nabla \mathrm{E}_p\left(w^k\right)$ el error cuadrático medio de la red en determinada iteración del proceso de aprendizaje.

3.5.1 Pseudocódigo de Descenso Estocástico con Aprendizaje Adaptativo

Para clarificar más el algoritmo, se presenta su descripción en pseudocódigo:

0: Inicializar los pesos w, y los valores $\alpha, \gamma_1, \gamma_2$

1: **Repetir**

2: sea $k = k + 1$

3: Escoger un patrón p del set de patrones de entrenamiento.
4: Usando este patrón, calcular $\mathrm{E}_p\left(w^k\right)$ y entonces $\nabla \mathrm{E}_p\left(w^k\right)$
5: Calcular los nuevos pesos usando:
$w^{k+1} = w^k - \alpha^k \nabla \mathrm{E}_p\left(w^k\right)$
6: Calcular la nueva tasa de aprendizaje usando:
$\alpha^{k+1} = \alpha^k + \gamma_1 \left\langle \nabla \mathrm{E}_{p-1}\left(w^{k-1}\right), \nabla \mathrm{E}_p\left(w^k\right) \right\rangle + \gamma_2 \left\langle \nabla \mathrm{E}_{p-2}\left(w^{k-2}\right), \nabla \mathrm{E}_{p-1}\left(w^{k-1}\right) \right\rangle$
7: **Hasta** que se cumpla la condición de terminación.
8: **Retornar** los pesos finales w^{k+1}

3.6 Variables significativas del entrenamiento

A continuación se describen las variables más importantes que actúan en el algoritmo de aprendizaje. La variación de dichas variables puede alterar de manera considerable los resultados del entrenamiento, y por tanto el funcionamiento de la red neuronal.

- **Número de neuronas de la capa oculta:** Generalmente un aumento de neuronas ocultas aumenta la eficacia del aprendizaje. Sin embargo, hay que tener en cuenta que existe un margen óptimo en el número de neuronas ocultas, es decir, fuera de ese intervalo los resultados pueden empeorar. Con un error elevado en la salida, un aumento del número de neuronas ocultas puede hacer que disminuya dicho error. De todas formas la tendencia es tener el menor número posible de ellas ya que el proceso de aprendizaje es más rápido.
- **Número de capas ocultas**: Un aumento del número de capas de neuronas ocultas se traduce en un cambio en la estructura de la red, pudiéndose obtener resultados diferentes. Generalmente con una capa oculta es suficiente. En una red neuronal sin capas ocultas normalmente solo se pueden tener relaciones lineales entre la entrada y la salida.
- **Condiciones iniciales**: Una mejora en las condiciones iniciales de pesos y umbrales da la posibilidad de que se alcance un mínimo global y no uno local. Diferentes condiciones iniciales pueden hacer que se alcancen diferentes mínimos, y por tanto diferentes resultados.
- **Número de iteraciones:** Un número elevado de ellas significa un entrenamiento lento. Mediante control de los otros parámetros se puede conseguir que la red entrene con un número más reducido de iteraciones. Recuérdese que el objetivo del entrenamiento de una red neuronal es adaptar los pesos y los umbrales de las neuronas para así minimizar el error entre un conjunto de patrones dados como ejemplo y sus salidas estimadas.
- **Patrones de entrada y salida**: Un aumento del número de patrones de entrada y salida hace que la red tenga un mejor aprendizaje, y que por tanto cuando se muestren a la red datos desconocidos (diferentes de los utilizados en el entrenamiento) el error en el cálculo de las salidas sea más pequeño. Sin embargo, cuanto mayor sea el número de datos más largo es el proceso de aprendizaje de la red neuronal. Es muy importante hacer un estudio previo de los datos para evitar confusiones a la red neuronal. Cuanto más definidos estén los datos más rápido aprenderá la red.
- **Tasa de aprendizaje**: La tasa de aprendizaje es la encargada de acelerar el proceso de aprendizaje. Se suelen escoger valores pequeños, empezando con una tasa constante. Lógicamente es deseable que, hasta alcanzar el mínimo, el entrenamiento sea rápido

mientras que en el entorno del mínimo sea lento para poder alcanzarlo plenamente. Por este motivo se utiliza una tasa de aprendizaje variable, adaptativa a lo largo del proceso de entrenamiento.

- **Momento:** Contrarresta las posibles inestabilidades que se crean en la variación de los pesos, y es importante porque reduce la posibilidad de caer en un mínimo local, además puede acelerar enormemente el proceso.
- **Función de activación**: La función de transferencia o función de activación es la encargada de representar las salidas en función de las entradas a cada neurona. Para el modelo de red Backpropagation dicha función debe ser derivable.

Capítulo 4

Descripción del Sistema Propuesto

4.1 Descripción del problema

4.1.1 Perspectiva general del problema

El ritmo actual de desarrollo de las telecomunicaciones está imponiendo la necesidad de tasas de transmisión de información cada vez más altas. Ya que el espectro radioeléctrico es un recurso finito este requerimiento no puede ser satisfecho mediante un simple incremento del ancho de banda de los sistemas. Esta circunstancia está conduciendo al uso más generalizado de formatos de modulación multinivel espectralmente más eficientes, como por ejemplo MQAM.

Los formatos de modulación con mayor eficiencia de ancho de banda, los denominados esquemas de modulación lineal, requieren, por otra parte, de amplificación lineal. La solución convencional para amplificación lineal consiste en usar amplificadores en clase A. Sin embargo, la enorme relación valor pico a valor medio que presentan los formatos M-arios, impone que estos amplificadores sean operados con un elevado *back-off*, mermando aun más la intrínsecamente baja eficiencia de esta clase de amplificadores. Por otra parte, la eficiencia de potencia es, también, un factor de suma importancia en los sistemas de comunicaciones modernos, en particular en sistemas donde la autonomía es crucial, como por ejemplo en móviles y satélites. Ambas exigencias, elevada eficiencia de ancho de banda y elevada eficiencia de potencia, son, convencionalmente, difíciles de conciliar. Este escenario ha dado mayor vigencia al problema de la linealización de los *amplificadores de potencia*.

Los amplificadores de potencia o AP [49,50] son dispositivos altamente no lineales, cuyo alto consumo (aproximadamente un 70% de la energía disponible) y su indispensable presencia en los sistemas de comunicaciones, los convierten en continuo objeto de investigación [51]. Además, como es bien sabido, existe un compromiso entre eficiencia y linealidad, así pues, los AP's más eficientes desde un punto de vista de rendimiento energético (clase AB, clase C y los conmutados clase D, E, F), son los que presentan mayor grado de no linealidades. Los efectos de las no linealidades en los AP's son dobles: por un lado causan recrecimiento espectral, dando lugar a interferencias en los canales adyacentes (distorsión fuera de banda), mientras que por otro lado, causan distorsión dentro de la propia banda de transmisión, degradando por tanto el *bit error rate* (BER).

Los organismos reguladores fijan, a través de los diferentes estándares de comunicaciones, los niveles máximos de emisión fuera de banda permitidos (máscaras de emisión de potencia que delimitan el *adjacent channel power ratio*, ACPR), así como de la propia distorsión en banda, por ejemplo especificando porcentajes máximos de error en las constelaciones (*error vector magnitude*, EVM).

Por otro lado, los actuales estándares de comunicaciones (IEEE 802.11g, IEEE 802.16, ETSI HiperLAN-2, UMTS) apuestan por una alta eficiencia espectral en canales con anchos de banda considerables, utilizando modulaciones multinivel, multiportadora o ambas (M-QAM, $\pi/4$ DQPSK, WCDMA, OFDM). Dichas modulaciones son muy sensibles a las no linealidades de los AP's y puesto que presentan envolventes con gran relación potencia de pico a potencia media (*peak to average power ratio*, PAPR), requieren altos niveles de back-off para poder operar en una región lo más lineal posible, penalizando de este modo la eficiencia del AP.

Las diferentes estructuras linealizadoras propuestas en la literatura [52], tienen como objetivo minimizar los efectos de las no linealidades a la par que maximizar la eficiencia de los AP's. La presente investigación está enmarcada dentro de este contexto, el de la linealización de amplificadores de potencia mediante el uso de redes neuronales

4.1.2 Técnicas de Linealización

Diferentes técnicas de linealización han sido actualmente propuestas, y en menor escala también desarrolladas. Un análisis sistemático y detallado de todas las técnicas de linealización utilizadas hoy día se puede leer en las referencias [53] y [54].

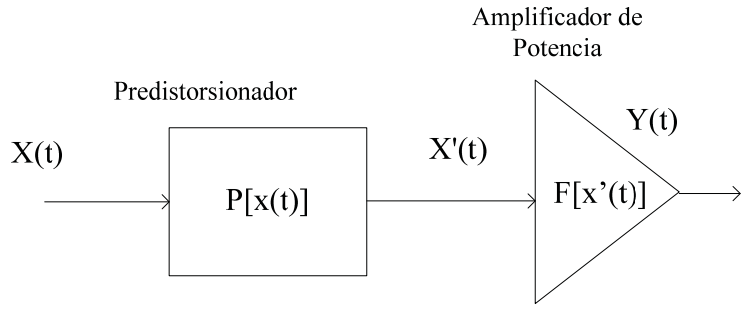
Los métodos de linealización pueden ser divididos en dos grandes grupos:

- **Aquellos en los que se reduce la distorsión**. A este grupo pertenecen aquellos sistemas en los que mediante un mecanismo adecuado (la linealización) se elimina o compensa, propiamente, la distorsión introducida por el amplificador. Esta compensación puede ser efectuada tomando una muestra de los productos de intermodulación generados por el AP e inyectándolos apropiadamente desfasados en la salida (*feedforward*), o tomando una muestra de la señal de salida e inyectándola a la entrada (*feedback*), o bien mediante una alteración apropiada de la forma de la envolvente de la señal de entrada (predistorsión), que es el método utilizado en esta tesis.
- **Aquellos en los que se evita la distorsión**. El segundo grupo lo constituyen aquellos métodos en los que la señal original con envolvente variable en el tiempo es transformada (reversiblemente) en dos señales con envolvente constante. Las señales que resultan son amplificadas por separado y sin distorsión, y posteriormente son recombinadas produciendo una réplica amplificada de la señal original. Los esquemas de modulación que **estimulan** muy poco la característica no lineal de los amplificadores de potencia pueden verse, también, como esquemas de linealización en sí mismos.

4.1.3 Predistorsión

Esta técnica es empleada extensivamente en la linealización de amplificadores TWTA [8] y SSPA [55], y es una de las técnicas de linealización (junto con *feedforward*) que más atención recibe actualmente en el contexto de las comunicaciones inalámbricas de próxima generación [56].

En esta técnica de linealización la señal de entrada del AP es modificada mediante cierta ley de predistorsión, haciendo preceder el amplificador de potencia por un dispositivo alineal cuya relación entrada-salida es la inversa de la relación entrada-salida del AP (Figura 4.1).



$$Y(t) = F\{ P[x(t)] \} = k_o x(t)$$

Figura 4.1 Arquitectura básica de predistorsión

Las técnicas de predistorsión pueden aplicarse sobre la señal o los datos. Ésta última, la que se aplica sobre los datos, se propuso en los años 80 y su interés creció a principios de los 90. Es una técnica que se realiza a medida para los formatos concretos de modulación digital. La función de predistorsión pretende compensar el espacio vectorial de los datos o constelación. Los coeficientes se optimizan minimizando el parámetro EVM. Sin embargo, con esta técnica la distorsión fuera de banda no se compensa y esto puede ser un problema para cumplir determinadas especificaciones técnicas de los estándares. Además, es preferible considerar una técnica de predistorsión que pueda utilizarse más en general como la que se aplica sobre la señal.

La predistorsión digital de señal genera una señal predistorsionada que al pasar a través del amplificador de potencia debería no tener distorsión a la salida, es decir, convertiría al amplificador de potencia en un dispositivo ideal lineal hasta el punto de saturación. El objetivo de esta técnica es compensar la distorsión in-band y out-of-band mientras el nivel de saturación del amplificador lo permita. Es una técnica que presenta ventajas desde el momento en que no depende del tipo de amplificador (clase A, AB, C o de la tecnología), y de la modulación.

La predistorsión digital es una técnica de linealización vigente que seguirá siendo objeto de intensa investigación en el futuro próximo.

4.1.4 Modulación M-QAM

La Modulación de Amplitud en Cuadratura o M-QAM [57] es una modulación digital en la que el mensaje está contenido tanto en la amplitud como en la fase de la señal transmitida. Se basa en la transmisión de dos mensajes independientes por un único camino. Esto se consigue modulando una misma portadora, desfasada 90º entre uno y otro mensaje. Esto supone la formación de dos canales ortogonales en el mismo ancho de banda, con lo cual se mejora en la eficiencia de ancho de banda, donde $M = 2^n$ bits transmitidos, de modo que en una constelación 16-QAM, $M = 2^4$ bits por símbolo transmitido como se puede apreciar en la Figura 4.2

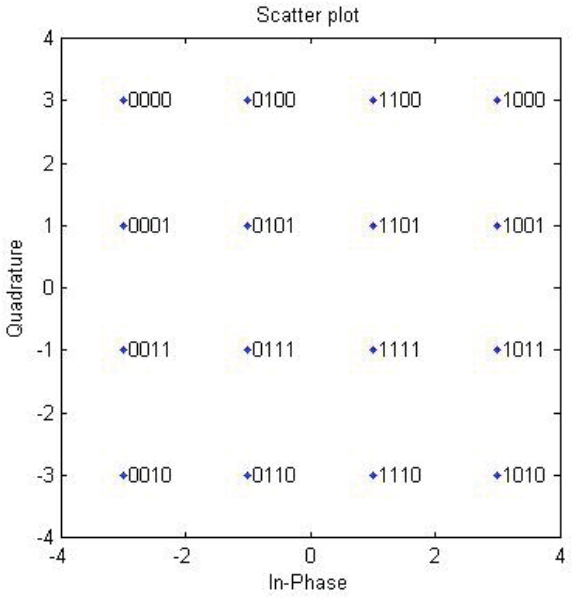


Figura 4.2 Constelación 16-QAM

La importancia de este sistema de modulación se debe a la gran cantidad de aplicaciones asociadas a ella [58]:

- **Sistemas analógicos que utilizan la modulación M-QAM**: la modulación de amplitud en cuadratura es utilizada en los sistemas PAL y NTSC de televisión analógica para transmitir las dos señales de crominancia.
- **Sistemas digitales que utilizan la modulación M-QAM**: la modulación de amplitud en cuadratura es utilizada en sistemas digitales de telecomunicación, como los módems. Según el número de símbolos existentes combinando las distintas amplitudes posibles de las dos señales que se transmiten, la modulación es denominada 4-QAM, 16-QAM, 64-QAM, etc.

El comportamiento alineal de un amplificador de potencia altera la forma original de la señal de entrada produciendo una señal de salida distorsionada. La respuesta alineal de un amplificador de potencia se suele caracterizar mediante las funciones de conversión AM/AM (*Amplitude Modulation/Amplitude Modulation*) y AM/PM (*Amplitude Modulation/Phase Modulation*) según las cuales, la modulación de amplitud de la señal de entrada produce, a su vez, una modulación modificada (alineal) de amplitud y una modulación no lineal de la fase, respectivamente, en la señal de salida. El efecto conjunto de ambas funciones produce distorsión dentro y fuera de la banda de frecuencias de la señal original. Mientras la distorsión dentro de banda puede corromper la señal transmitida haciendo imposible la extracción de la información, dado que deforma la constelación, tal como se muestra en la Figura 4.3.

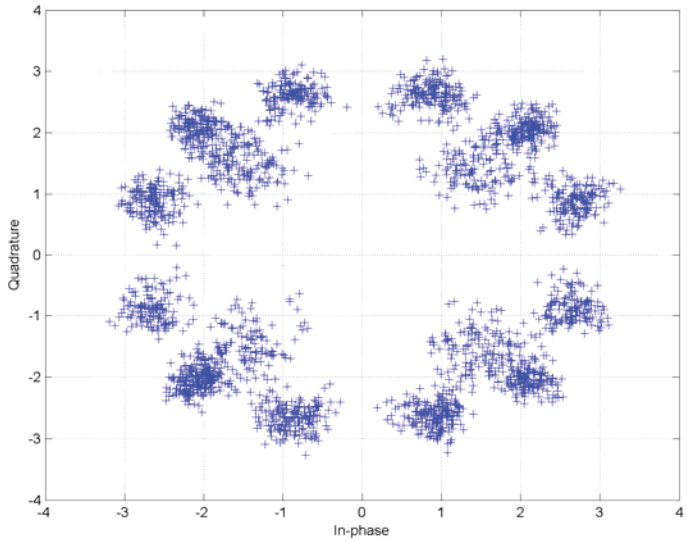


Figura 4.3 Aspecto de la constelación 16-QAM después de la amplificación alineal**.**

La calidad de la modulación se puede verificar de varias maneras:

- Inspección visual de la constelación de la modulación. Se realiza mediante una representación gráfica de los puntos de datos recibidos en un diagrama de ejes I y Q. Desviaciones respecto al diagrama de constelación ideal permite detectar problemas como la saturación en amplificadores.
- Métodos numéricos, como la medida de la desviación de los símbolos recibidos (en términos de valores de I y Q) respecto de la posición ideal. Estas medidas se denominan porcentaje de error de modulación (MER) y magnitud del vector de error (EVM).
- Medidas tradicionales de RF como relación señal a ruido y potencia de la señal.

4.1.5 EVM (Error Vector Magnitude)

La magnitud del vector de error (EVM) es una medida de la calidad de la modulación [59]. Esta métrica ha venido a ser un estándar industrial para una gran variedad de aplicaciones, desde móviles hasta televisión por cable. La idea básica detrás de EVM es que cualquier señal perturbada (normalmente señales complejas) puede representarse como una suma de señales ideales y una señal de error. Como la señal de error no se puede medir directamente, la instrumentación de medida determina el error en la señal reconstruyendo la señal ideal basándose en los datos detectados y restándola de la señal real.

El parámetro EVM se determina a la salida del demodulador a partir de la comparación entre el valor de símbolo recibido y el valor esperado idealmente. Se puede definir como el error RMS (Root Mean Square o raíz cuadrada del error cuadrático medio) de la diferencia de los valores de símbolos recibidos respecto a los esperados.

El EVM se define en los estándares de comunicación como la ecuación (54), donde I y Q son las componentes en fase y cuadratura de los símbolos. ΔI es la diferencia entre la componente I de la señal recibida y esperada. N es el número de símbolos del conjunto de medida. S es la potencia media de los símbolos esperados o de referencia.

$$EVM = \sqrt{\frac{1/N \sum_{1}^{N} \left(\Delta I_j^{\,2} + \Delta Q_j^{\,2} \right)}{S^{2}_{media}}} [\%] \qquad (54)$$

Es una medida de cómo la no linealidad afecta al proceso de detección. Para calcular este parámetro a menudo se utilizan analizadores vectoriales de señal o VSA (Vector Signal Analyzer) [60] u otros instrumentos que incorporan funciones de software específicas. Se expresa en porcentaje y normalizado para permitir la comparación entre las diferentes modulaciones.

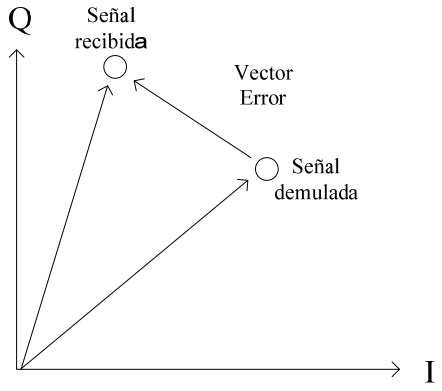


Figura 4.3. Definición del Error Vector Magnitud

4.2 Diseño de la Solución

4.2.1 Modelo de Distorsión No Lineal

Con todo lo expuesto, el comportamiento no lineal de los amplificadores de alta potencia, constituye uno de los principales obstáculos para el funcionamiento de los sistemas de comunicación digital basados en M-QAM. Afortunadamente, este nocivo efecto puede ser compensado mediante diversas técnicas clásicas de linealización cuyas variantes ad hoc han sido propuestas e investigadas durante años. Entre dichas técnicas, la predistorsión digital ofrece óptimas condiciones para el diseño de linealizadores adaptativos, ya que puede ser fácilmente implementada sobre la información discreta de las señales de banda base. El objetivo general que se persigue es alcanzar el nivel de linealidad necesario para explotar al máximo las capacidades inherentes a la modulación QAM y, al mismo tiempo, alcanzar un máximo aprovechamiento de la potencia disponible, para esto se propone un modelo de predistorsión basado en redes neuronales cuyas características se presentan a continuación.

En la Ilustración 4.4 se presenta el diagrama de bloque del sistema de transmisión para un canal satelital. La entrada al sistema es una secuencia de bits $\{x(t)\}$ de información, la cual es modulada como un símbolo complejo M-QAM, la señal M-QAM $y(t)$ que es amplificada usando un amplificador de potencia de basado en tubos de ondas progresivas (TWT) que genera la señal

distorsionada de salida $z(t)$. Tal como se indicó anteriormente, esta señal $z(t)$ presenta distorsión alineal de amplitud y fase, cono lo cual la eficacia del sistema de comunicación se degrada.

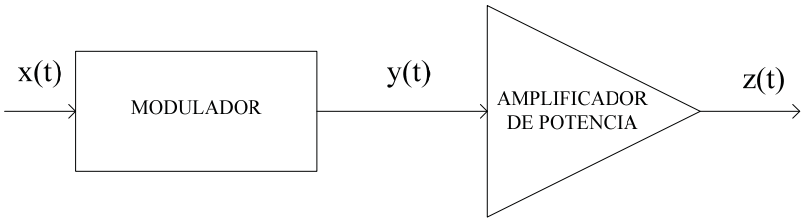


Figura 4.4. Sistema de transmisión Satelital

Los Amplificadores TWT (Traveling-Wave Tubes) se encuentran en una posición predominante en aplicaciones de alta frecuencia, especialmente en sistemas de microondas. Así, entre los modelos disponibles para caracterizar estos Amplificadores de Potencia, se opta por el modelo de A.M. Saleh [8] para TWT sin memoria. Este es de hecho el modelo más universalmente referido en la literatura, y normalmente es formulado a través de las siguientes expresiones:

$$z(t) = A(|y(t)|) * \exp\left(j\{\angle y(t) + P(|y(t)|)\}\right) \tag{55}$$

Donde $|y(t)|$ y $<y(t)$ representan la amplitud y fase de la señal predistorsionada $y(t)$. Las funciones $A(\cdot)$ y $P(\cdot)$ representan la distorsión no lineal de amplitud y fase respectivamente y son obtenidas usando el modelo propuesto por Saleh :

$$A(|y(t)|) = \frac{2|y(t)|}{1+|y(t)|^2} \tag{56}$$

$$P(|y(t)|) = \frac{2|y(t)|^2}{1+|y(t)|^2} \times \frac{\pi}{6} \tag{57}$$

4.2.2 Técnica Analítica de Predistorsión

La idea básica del Predistorsionador que aquí se propone es reducir la distorsión no lineal introducida por el Amplificador de tubo. Para lograr este propósito la estructura del Predistorsionador se representa usando la siguiente ecuación:

$$G(t) = M(|x(t)|) * \exp\left(j\{\angle x(t) + N(|x(t)|)\}\right) \tag{58}$$

donde $M(\cdot)$ y $N(\cdot)$ denotan las funciones *inversas* de amplitud y fase del amplificador; respectivamente . $x(t)$ representa la señal banda base filtrada previamente distorsionada por la red neuronal. Estas funciones son obtenidas usando las siguientes ecuaciones [58]:

$$M\left(|x(t)|\right) = A^{-1}\left(|x(t)|\right) \tag{59}$$

$$N\left(|x(t)|\right) = -P\left(|x(t)|\right) \tag{60}$$

Se ve claramente que para lograr la función de predistorsión ideal $G(t)$, sólo basta hallar la *función inversa de amplitud* del Amplificador de Potencia. En la Ilustración 4.5 se puede apreciar la función ideal *directa* de la amplitud, la *inversa* de la misma y la función *compuesta* por ambas, esta última indica la linealidad del sistema.

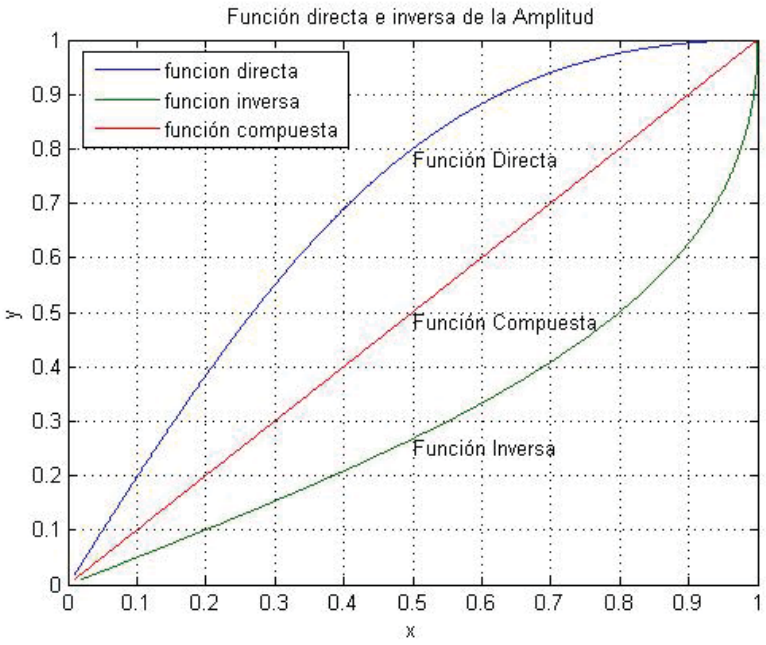


Figura 4.5. Función directa, compuesta e inversa ideal de la amplitud

La función de predistorsión $G(t)$ es aproximada usando una red neuronal de tipo MLP cuyos coeficientes son ajustados usando el algoritmo de aprendizaje Backpropagation.

El sistema de transmisión satelital con predistorsión queda de la siguiente manera:

Figura 4.6. Sistema de transmisión con Predistorsión

En la Figura 4.6 se presenta el diagrama de bloque simplificado del sistema de transmisión con predistorsión. La entrada al sistema es una secuencia de bits $\{x(t)\}$ de información, la cual es mapeada sobre un símbolo complejo 16QAM. La señal banda base filtrada $y(t)$ es amplificada

usando un amplificador de potencia de tubo TWT, pero cuando el sistema es implementado con técnica de compensación; la señal filtrada *y*(*t*) es previamente predistorsionada, insertándose el compensador neuronal entre el modulador y el AP de manera de predistorsionar la señal *y*(t) en una *y´*(t) ideal inversa que al pasar por el AP produzca una señal *z*(t) sin distorsión de amplitud (AM/AM) y fase (AM/PM).

4.2.3 Modelo de Predistorsión basado en Redes Neuronales

La técnica analítica de predistorsión descrita en la sección previa precisa que para obtener la función de predistorsión ideal solo basta hallar la función inversa de la *Amplitud* del amplificador de potencia. Para conocerla se desarrolla un algoritmo de predistorsión que estima esta función de predistorsión requerida (58) directamente de los datos obtenidos del amplificador y no requiere conocimiento directo de esta misma función inversa.

La función del predistorsionador es no lineal, desconocida y, como se ha expuesto anteriormente, una técnica excelente para aproximar funciones no lineales es el uso de las redes neuronales, el particular el modelo conocido como Multilayer Perceptron [1]. Resultados obtenidos por Cybenko [61] demuestran que la utilización de funciones sigmoidales en una red multilayer perceptron es suficiente para modelar cualquier curva continua a un grado arbitrario de precisión bajo ciertas condiciones. Por lo tanto un predistorsionador basado en una red neuronal puede ser implementada usando un par de redes MLP y el algoritmo de entrenamiento Backpropagation para aproximar las funciones de predistorsión deseadas.

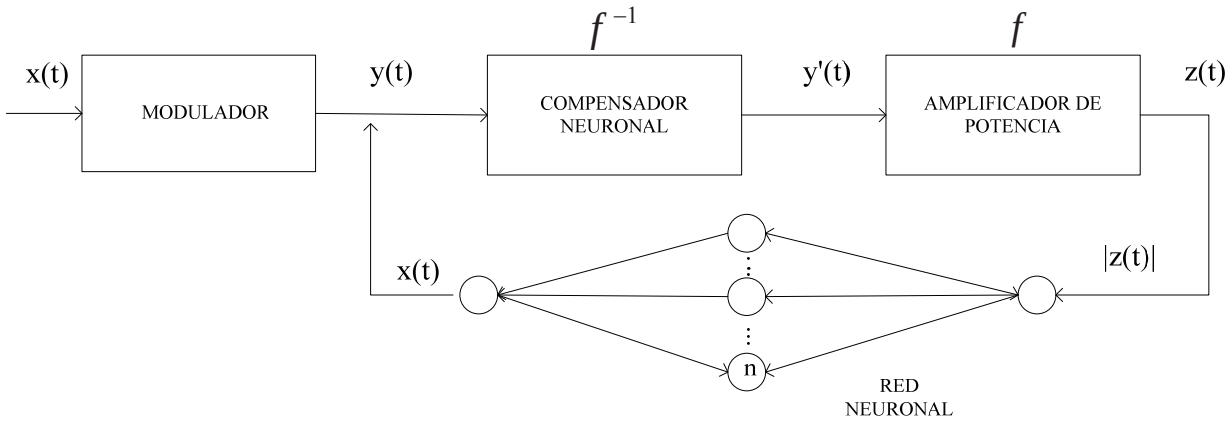


Figura 4.7 Funcionamiento de un Predistorsionador en un sistema de transmisión**.**

Como se ilustra en la Figura 4.7 el predistorsionador basado en redes neuronales muestrea la data del amplificador y recupera la amplitud y fase de las señales de entrada y salida. El predistorsionador de la función inversa de Amplitud A^{-1} es obtenido aplicando la salida del amplificador como la entrada (*input*) de la red neuronal, mientras la amplitud de entrada del amplificador es utilizada como data de entrenamiento. Cuando la red neuronal converge su respuesta será la inversa de la función de Amplitud del Amplificador, con lo que se obtendrá la función de predistorsión. La salida *z*(t) debe ser equivalente o al menos proporcional a *x*(t).

En esta sección, es presentado un esquema de predistorsión basado en redes neuronales de tipo MLP con algoritmo aprendizaje Backpropagation. Este esquema de predistorsión se simulará y evaluará su rendimiento en su versión Real y Compleja. A continuación, se procede a pormenorizar las características de la implementación de este compensador neuronal: el software en que se desarrolla la simulación, formato de los datos de entrada y salida, diagrama de flujo del programa principal y entrenamiento de la red neuronal.

4.3 Implementación del Modelo Propuesto

4.3.1 Software de Simulación

Para realizar la simulación del compensador neuronal se utiliza un entorno comercial y propietario llamado MatLab en su versión 7.014. Matlab es un software técnico para computación y visualización numérica y una herramienta elemental para cursos introductorios y avanzados en matemáticas, ingenierías y ciencias. En la industria MatLab es la herramienta por excelencia para proyectos de investigación que impliquen el desarrollo de algoritmos computacionales y matemáticos, para modelado, simulación y elaboración de prototipos, y análisis, exploración y manipulación de datos. MatLab incluye un toolbox (conjunto de aplicaciones específicas) de, entre otros, redes neuronales (el Neural Network Toolbox). Para la simulación del compensador neuronal no se utiliza este toolbox, sino que se programa la red neuronal y su algoritmo de entrenamiento usando estructuras de datos básicas, para aprovechar el potencial de MatLab.

4.3.2 Formato de los Datos

Para realizar el entrenamiento del compensador neuronal, se utilizarán muestras proporcionadas en formato de Matlab (archivo .mat). Corresponden en este caso, a una modulación 16 y 64-QAM a la entrada y salida de un amplificador de potencia. Una vez realizado el ajuste de los coeficientes de la red mediante el algoritmo Bakpropagation, los pesos ajustados se guardan también en un archivo de extensión .mat que podrá ser luego manipulado cuando se requiera presentar la salida del sistema de transmisión con predistorsión.

- **Patrones de Entrada de la red**: muestra de amplitud del amplificador de tubo de onda viajera TWT en 16 y 64-QAM contenidos en el archivo datatr.mat entregado por el profesor.
- **Patrones de Salida de la red:** constituye la salida deseada de la red y corresponde a la amplitud inversa del amplificador de potencia.

4.3.3 Diagrama de Flujo del Programa Principal

La figura 4.8 muestra la primera parte del diagrama de flujo del programa principal.

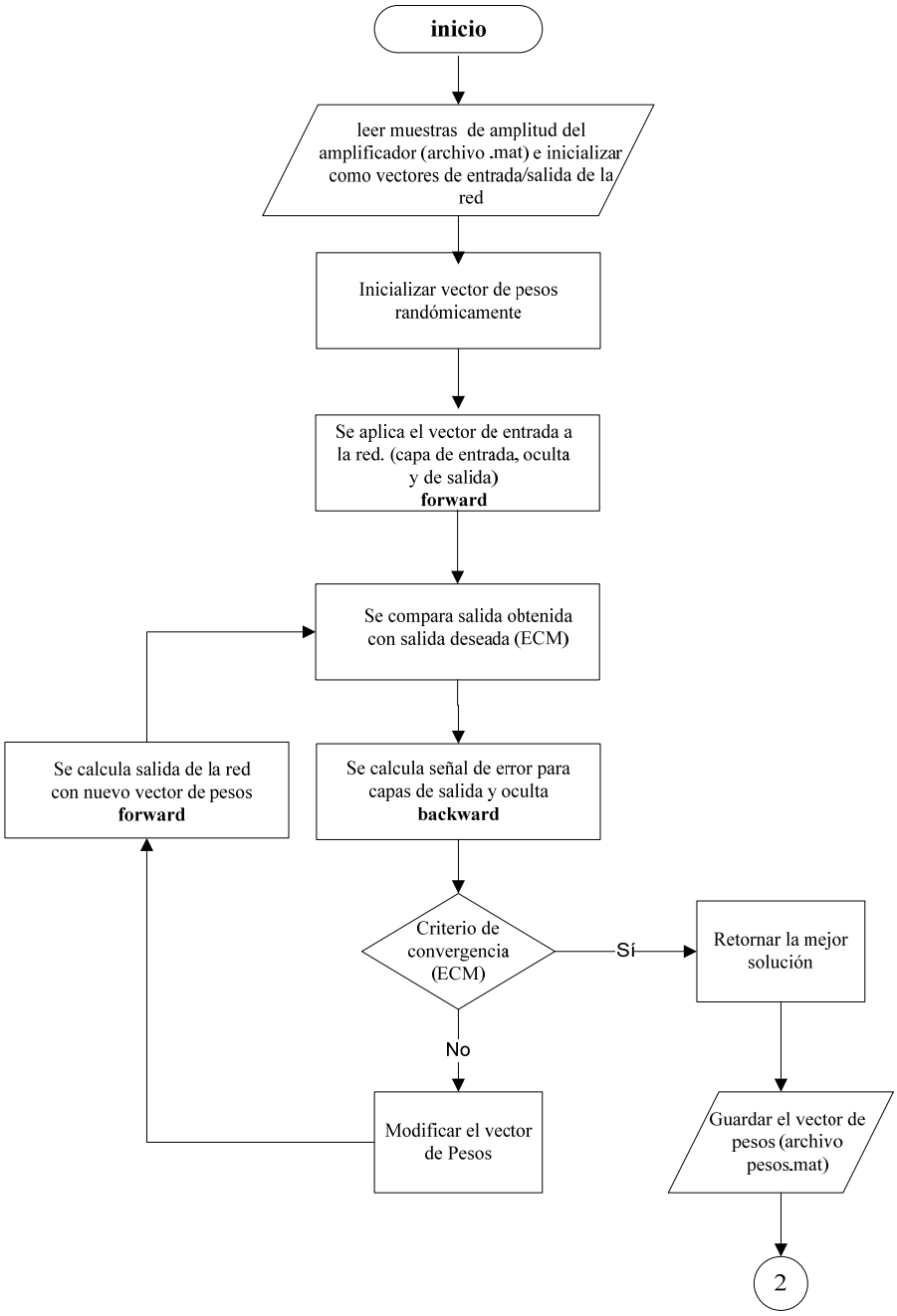


Figura 4.8 Diagrama de Flujo del programa principal

Las funciones principales son:

- Leer los datos de muestra de la amplitud amplificador desde el archivo .mat e inicializarlos normalizados como vectores de entrada (la función amplitud A) y salida (la función inversa A-1 deseada de la amplitud) del compensador neuronal.

- Se define la configuración de la red (neuronas y matriz vectorial de pesos) utilizando estructuras de datos estáticas y se inicializa el vector de pesos randómicamente con valores pequeños entre 0 y 1.
- Se aplican el vector de entrada a la primera capa de la red como estímulo y éste se propaga a través de las capas superiores de la red, hasta generar una salida.
- La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas, el ECM (Error Cuadrático Medio).
- Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta sólo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total.
- Basándose en la señal de error percibida, se actualiza el vector de pesos de conexión de cada neurona, para hacer que la red converja.
- Se calcula la salida de la red, con el nuevo vector de pesos y se calcula nuevamente el ECM, este proceso se repite hasta que se cumple el criterio de convergencia (un ECM mínimo) o se realizan cierto número de iteraciones.
- Una vez cumplido el criterio de convergencia se guarda el vector de pesos que arrojo el mejor ECM de la red en un archivo .mat

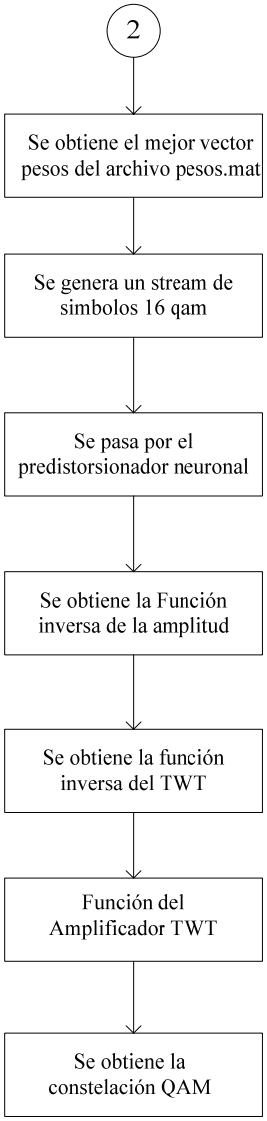


Figura 4.9 Diagrama de flujo del programa principal (parte 2)

La segunda parte del programa, una vez terminado el entrenamiento de la red neuronal y obtenido el mejor vector de pesos, es insertar el compensador neuronal en el sistema de transmisión y evaluar su rendimiento. La mejor prueba de la eficiencia del predistorsionador es la constelación QAM, puesto que esta muestra directamente la posición de los símbolos en el cuadrante IQ.

- Una vez obtenido el mejor vector de pesos con el cual converge la red, este se utiliza para implementar el compensador neuronal: los pesos de conexión de las neuronas se dejan fijos y se aplica una nueva entrada a la red, la cual genera una salida que constituye la función inversa de la amplitud del amplificador.
- Se genera un stream de símbolos de 16QAM y se aplica como vector de entrada del compensador neuronal.

- La salida del compensador neuronal es la función inversa aproximada de la amplitud del amplificador de potencia TWT.
- Una vez obtenida la función inversa de la amplitud, se calcula la función inversa del amplificador TWT, mediante las ecuaciones de predistorsión detalladas anteriormente.
- Se obtiene la función directa del amplificador TWT, simulada en matlab por el módulo twta.m, que implementa las funciones de cálculo de Saleh [8] que definen el amplificador de potencia.
- Se demodula y dibuja la constelación QAM resultante (matchet.m), después de pasar la señal por el amplificador de potencia, esta constelación no debiera verse afectada por la distorsión no lineal de amplitud y fase introducida por el AP en la señal transmitida en virtud del compensador neuronal utilizado para su predistorsión.

4.3.4 Compensador Neuronal Real

- La red neuronal real fue configurada con un nodo de entrada, diecinueve nodos ocultos y un nodo en la capa de salida.
- Tiene una función de transferencia tangencial hiperbólica en la capa oculta y sigmoidal en la capa de salida.
- Para su entrenamiento utiliza el algoritmo Backpropagation.
- Tiene un momento μ=0.9
- Tasa de aprendizaje α=0.02.
- Completa su aprendizaje en 500 iteraciones
- Converge con ECM (Error Cuadrático Medio) de 0,00444.

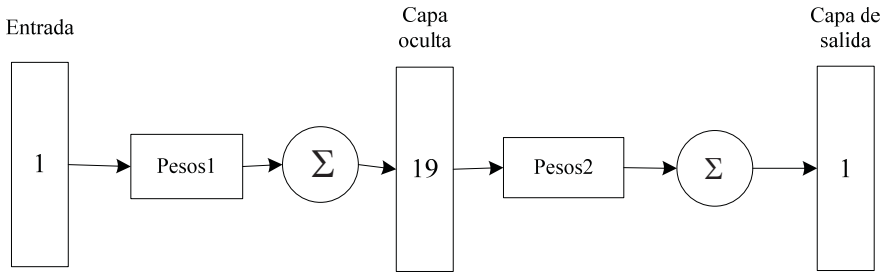


Figura 4.10. Arquitectura de la red real

4.3.5 Compensador Neuronal Complejo

- La red neuronal compleja fue configurada con un nodo de entrada, quince nodos ocultos y un nodo de salida.
- En la capa oculta tiene una función de transferencia especial [46]: una sigmoidal modificada y en la capa de salida una función tangencial hiperbólica, ambas funciones se definen en el dominio de los complejos.
- Los pesos son complejos y son inicializados de manera randómica.
- Tiene un momento μ=0.9
- Tasa de aprendizaje adaptativa [48].

- Completa su aprendizaje en 140 iteraciones.
- Converge con un ECM final de 0.000145; su error es mucho menor que su símil real.
- Los pesos fueron estimados usando una modificación del algoritmo Backpropagation para el dominio complejo, detallado anteriormente.

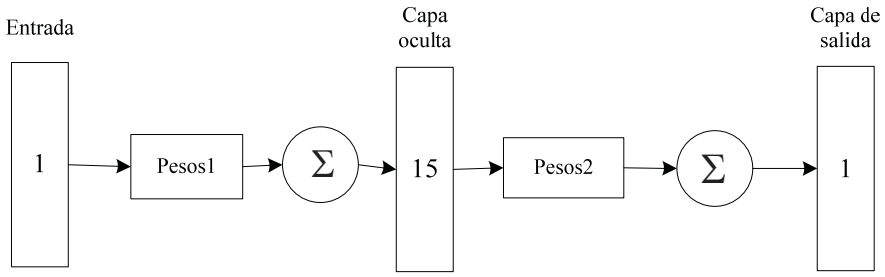


Figura 4.11. Arquitectura de la red compleja

Una diferencia muy importante respecto al compensador estudiado en el apartado anterior son las funciones de transferencia utilizadas con esta red neuronal, ambas complejas, acotadas en el dominio de entrada de la red y que hacen posible la convergencia de la red neuronal compleja.

4.4 Calibración de los Compensadores Neuronales

La calibración de los parámetros de aprendizaje y estructura de la red siguen siendo actualmente más bien mediante ensayo y error y hasta el momento no existe una heurística definida. Sin embargo, cuando las redes son correctamente adaptadas a un problema, permiten alcanzar a menudo tasas superiores de aprendizaje y generalización que métodos de IA o estadísticos.

La elección de la mayoría de los parámetros que intervienen en la definición y ajuste de los compensadores bajo estudio se ha llevado a cabo partiendo de lo que se denomina experimentos piloto. De una manera totalmente arbitraria, y orientado por la experiencia del investigador, se elige un conjunto de parámetros de partida, y con ellos se crea y evalúa la habilidad de aprendizaje del tipo de modelo bajo estudio. Repitiendo este procedimiento para otros valores de los parámetros, se puede acotar un valor supuestamente óptimo, que será utilizado en los experimentos posteriores. Conviene aclarar que esta búsqueda no es totalmente exhaustiva, y que por lo tanto, es posible que otra elección de valores para los parámetros pudiera haber producido resultados mejores.

En esta sección se presenta los resultados de los experimentos piloto o método de ajuste de parámetros de las redes, para ello, se varía el valor del parámetro a ajustar y se mantienen fijos el resto de los parámetros. Considérese que los patrones de entrenamiento son siempre los mismos para ambas redes.

4.4.1 Calibración del Compensador de coeficientes Reales

Como se indicó arriba, esta red tiene una arquitectura de 1:19:1, con pesos reales inicializados randómicamente. Los datos que constituyen los patrones de entrada y salida de la

red son los mismos señalados en secciones anteriores. La tabla 1 se muestra las diferentes funciones de activación usadas en la capa de salida de la red y el correspondiente ECM arrojado por la misma; en cambio en la capa oculta se utiliza la función de activación tangencial hiperbólica, que ofrece un mejor rendimiento a la red, con diferencia, del resto de las funciones de activación

Función de Activación		ECM
TANH	$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	0.013817
TAN	$\tan(x) = \frac{\cos(x)}{\sin(x)}$	>1
SIGMOIDAL	$\log sig(x) = \frac{1}{1 + e^{-x}}$	0.004347

Tabla 1. Performance de la red con diferentes funciones de activación

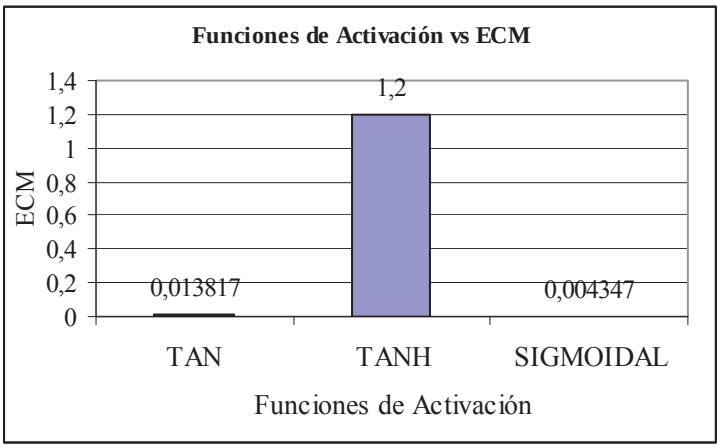


Figura 4.12 ECM para diferentes funciones de activación

Se puede observar que la función de activación que permite mejor rendimiento a la red es la función sigmoidal con un ECM=0.004347 que es más de tres veces menor que función la tangencial (TAN). También resulta evidente que la función tangencial hiperbólica (TANH) es altamente inadecuada para aproximar la función inversa de la Amplitud. El siguiente parámetro de la red a configurar es el Momento, para ello, se dejan fijos el resto de los parámetros.

Momento	ECM
ρ =0.5	0.030799
ρ =0.7	0.034476
ρ =0.8	0.009394
ρ =0.9	0.004347

Tabla 2. Momento versus Error Cuadrático Medio

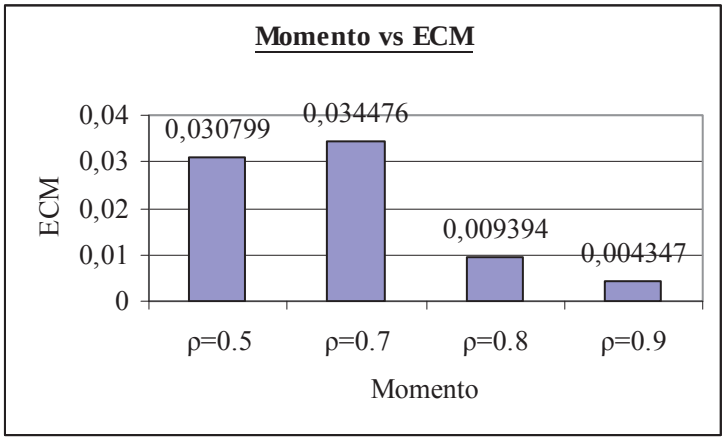


Figura 4.13 ECM para distintos valores de Momento

En la tabla 2 se puede observar claramente que el mejor valor de momento para la red real es 0,9. Para valores menores que 0,9 la convergencia de la red resulta lenta. Cabe señalar, que para valores mayores que 0,9 la red *diverge* y el ECM resultante es mayor a uno.

Epoch	ECM
1	0,132265
101	0,035824
201	0,03419
301	0,017823
401	0,00596
501	0,004972
601	0,00478
701	0,004727
801	0,004708

Tabla 3. Número de iteraciones versus ECM

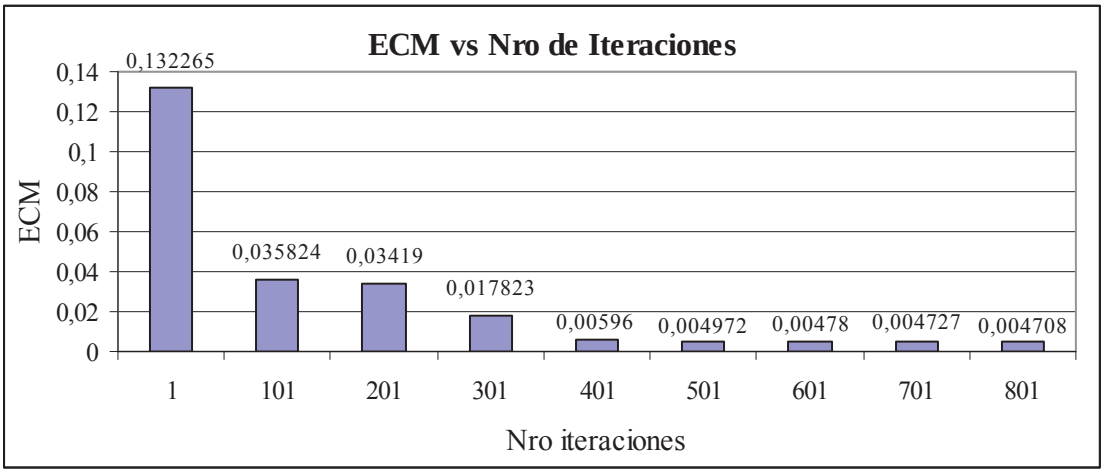


Figura 4.14 ECM versus número de iteraciones

En la tabla 3 se muestran los ECM obtenidos para cada 100 ciclos de la red, como se puede apreciar, después de la iteración número 500 el decremento del ECM es despreciable, llegando casi al tope de la capacidad aproximativa de la red, por eso es ahí donde se fijó el límite de ciclos. El mayor decremento en el ECM es en los primeros 100 ciclos, hecho que se explica porque es en los primeros ciclos donde más fuertemente varía el vector de pesos de conexión de la red, por el contrario en los 300 últimas iteraciones el vector de pesos varía solo en una fracción de su propio valor y esta variación no resulta lo suficientemente relevante para provocar un cambio en la salida de la red y por ende, en el ECM.

4.4.2 Calibración del compensador de coeficientes Complejos

Como se indicó con anterioridad, esta red tiene una arquitectura de 1:15:1, con pesos complejos inicializados randómicamente. En la tabla 4 se muestran las diferentes funciones de activación *complejas* usadas en la capa de salida de la red y su correspondiente ECM, en contraste, en la capa oculta de la red se utiliza la función de activación casi-sigmoidal [46], que ofrece un mejor rendimiento, con gran diferencia, del resto de las funciones de activación complejas probadas.

Función de Activación		ECM
ARCSINH	Arcsinh(z)	0.005752
TANH	$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	0.011969
TAN	$\tan(z) = \frac{e^{iz} - e^{-iz}}{i(e^{iz} + e^{-iz})}$	0.000145
CASI-SIGMOIDAL	$\log sig = \frac{2c_1}{1 + e^{-c_2 z}} - c1$	0.072182

Tabla 4. Funciones de Activación Complejas y su correspondiente ECM

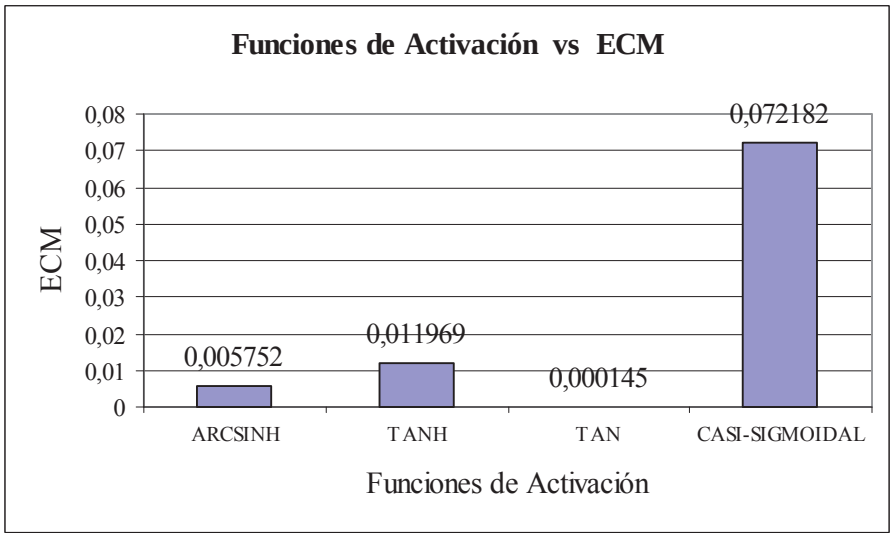


Figura 4.15 ECM para diferentes funciones de activación

Se puede observar que la función de activación compleja que proporciona mejor rendimiento a la red es la función tangencial (TAN) con un ECM=0.000145. Esta función es también la que presenta un mejor ECM en la red real: considerando esto, se puede concluir que es altamente adecuada para aproximar la función de amplitud inversa, a diferencia de la función sigmoidal o su símil compleja, la función casi-sigmoidal (acotada en el dominio de los complejos) que resulta altamente inconveniente, con un ECM cercano al 1.

El siguiente parámetro a configurar es el Momento, para ello se fijan el resto de los parámetros de la red y se procede a experimentar con diferentes valores de momento, en la tabla 5 se muestran los valores más significativos resultantes de las experiencias realizadas.

Momento	ECM
ρ =0.5	0.005890
ρ =0.7	0.005878
ρ =0.8	0.005081
ρ =0.9	0.000574

Tabla 5. ECM para diferentes valores de Momento

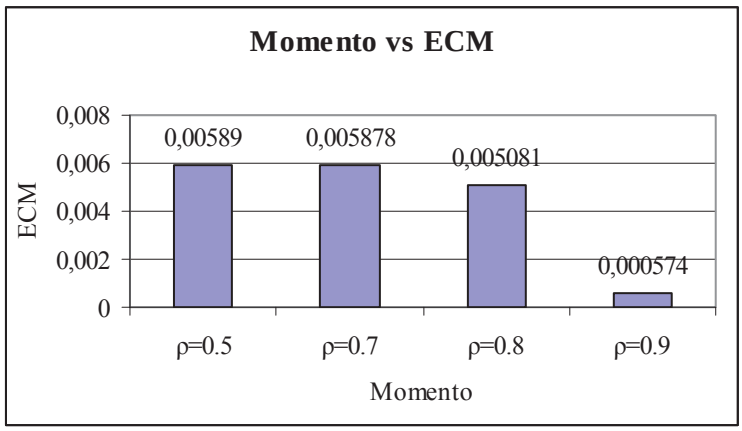


Figura 4.16. ECM para distintos valores de Momento

Resulta evidente que el mayor valor de momento (momento igual a 0,9) es el óptimo para la red, esto sucede porque mientras mayor sea el valor del momento la red tiende a converger con más rapidez, sin embargo, para valores mayores de 0.9 sucede que la red comienza a divergir y el ECM tiende a 1. Los valores de momento hasta 0,8 resultan pequeños para ejercer gran influencia en el ECM final de la red.

En la tabla 6 se muestran los ECM obtenidos para cada 10 iteraciones de la red, como se puede apreciar, después de la iteración número 140 el decremento del ECM es despreciable, por eso es ahí donde se fijó el límite de ciclos y donde alcanza su límite la capacidad aproximativa de la red. Obsérvese que el número necesario de ciclos para que esta red (compleja) converga a un ECM satisfactorio (fijado en 10^{-5}) es muchísimo menor que los necesarios para su símil real (500 iteraciones) que por cierto, no logra un ECM tan cercano a cero.

Epoch	ECM	Epoch	ECM
1	2,188	100	0,000233
10	0,328	110	0,000191
20	0,178	120	0,000158
30	0,095	130	0,000147
40	0,024	140	0,000145
50	0,022	150	0,000143
60	0,0067	160	0,000142
70	0,00191	170	0,000142
80	0,001388		
90	0,000544		

Tabla 6. Número de iteraciones (epochs) y ECM

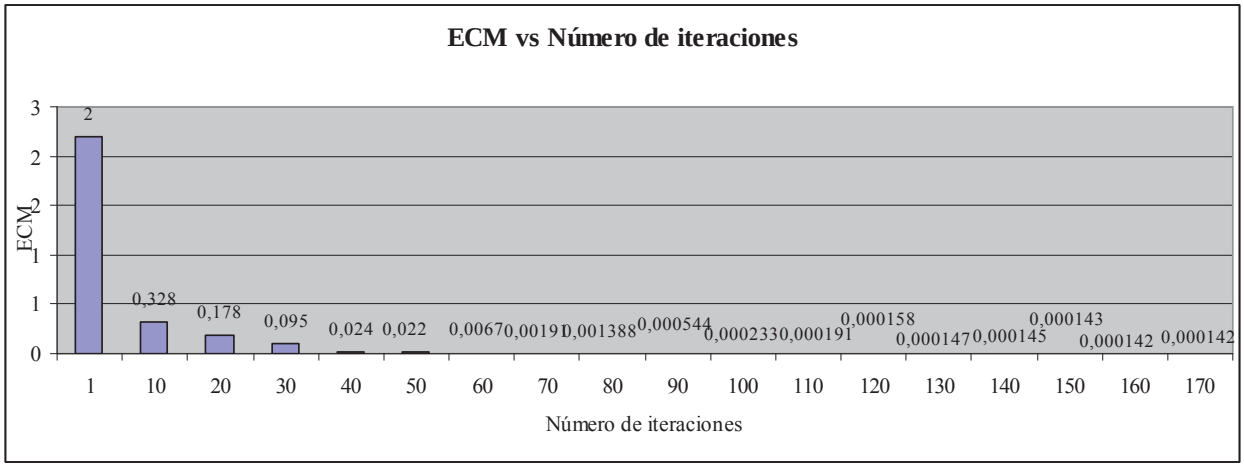


Figura 4.17. Numero de iteraciones de la red vs Error Cuadrático Medio

4.4.3 Tasa de aprendizaje fija y tasa de aprendizaje adaptativa

Se ha presentado el proceso de configuración de los parámetros del compensador complejo y se ha expuesto que éste utiliza para su aprendizaje una tasa de aprendizaje o alfa adaptativa, sin embargo, en un principio se utilizó una tasa fija (α=0,0014). Al experimentar con un alfa adaptativo [48] se llegó a la conclusión de que la red convergía a un ECM menor y de una manera mas rápida, es decir, el entrenamiento de la red resultaba más efectivo.

El esquema de ajuste en que se basa el método de adaptación de la tasa de aprendizaje explota la información relacionada al gradiente actual tanto como al valor de los gradientes previos, lo cual, provee a la red de estabilización en el valor de la tasa de aprendizaje adaptativa y ayuda al gradiente descendente a exhibir rápida convergencia. En la Tabla 7 se compara el rendimiento de la red compleja con tasa de aprendizaje fija y red compleja con tasa de aprendizaje adaptativa, en términos de ECM y número de iteraciones. Considérese que ambas

redes tienen el mismo número de capas y número de nodos e iguales funciones de activación y se entrenan con el mismo algoritmo de aprendizaje.

	Tasa fija	Tasa adaptativa
ECM	0,0005	0,000145
Nro iteraciones	140	130
α	0,0014	Oscila entre 0,001 y 0,009

Tabla 7. Rendimiento de una red con tasa de aprendizaje adaptativa y fija.

El ECM de la red con tasa adaptativa disminuye en aproximadamente 29% y 10 iteraciones, lo que es bastante progreso si se considera que es el resultado del ajuste de un solo parámetro de la red.

4.4.4 Red Real versus Red Compleja.

Como se pudo apreciar en las secciones anteriores, existe una gran disparidad entre los parámetros de la red compleja y la red real, contraste que se extiende también a su rendimiento, en términos de ECM y capacidad aproximativa de la función buscada, todo esto, a pesar de que los patrones de entrada y salida usados para el entrenamiento de la red son exactamente los mismos.

En la Figura 4.18 se muestra el gráfico de curvas de convergencia de las redes real y compleja durante su entrenamiento. Las curvas de convergencia muestra el ECM de la red para cada iteración del entrenamiento. Se consideraron solamente 200 iteraciones, como un número adecuado para contrastar la disímil convergencia de ambas redes.

De las curvas se desprende inmediatamente que la red compleja converge más rápido y mejor que su símil real. En el gráfico se puede apreciar, que en las 60 primeras iteraciones la curva de aprendizaje de la red compleja oscila fuertemente, esto se debe a la naturaleza del dominio en el que se encuentra embebida: en el dominio de los complejos las funciones de activación de los nodos, aunque sean derivables y acotadas al dominio de la función, tienen un comportamiento inestable debido a singularidades altamente indeseables [44] y [45], sin embargo debido a la alta variación del vector de pesos y por ende de los valores del ECM, que provoca la oscilación de la curva, es que la red converge rápidamente.

Por su parte la red real converge suavemente pero de forma mucho más lenta y su convergencia tiende más bien a estabilizarse pasadas las 100 iteraciones del algoritmo de aprendizaje disminuyendo cada vez menos el ECM por cada nuevo ciclo, lo que hace la convergencia de la red sumamente lenta.

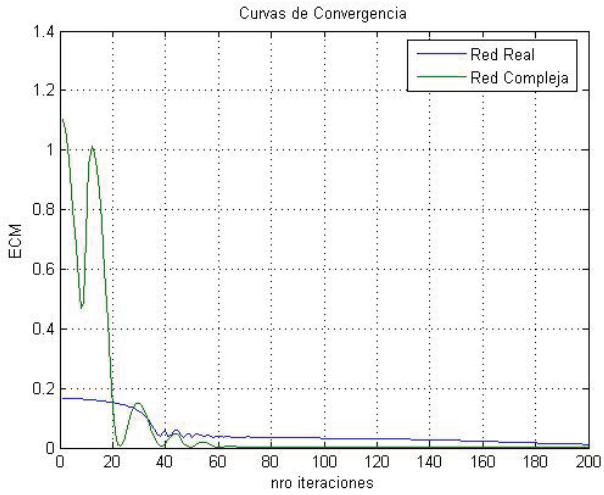


Figura 4.18 Curva de convergencia de la red real contrastada con la red compleja

En la tabla 8 se muestran comparativamente los parámetros más importantes de las redes compleja y real, para su contraste. La diferencia primordial entre ambas, es el dominio en que se desarrollan las ecuaciones del algoritmo de entrenamiento y en el que, por tanto, se encuentra embebida toda la red. Otra diferencia sumamente importante es el número de nodos en la capa oculta. Generalmente un aumento de nodos ocultos aumenta la eficacia del aprendizaje. Sin embargo, no es el caso en la red neuronal real donde se presenta un mayor número de nodos y su eficacia es menor. De todas formas la tendencia es tener el menor número posible de nodos ya que el proceso de aprendizaje es más rápido y a la vez menos costoso, en términos de hardware, lo que significa que la red compleja no solo es más eficaz, sino también más eficiente.

Uno de los mayores inconvenientes para la implementación en hardware de redes neuronales es la cantidad de lógica necesaria para realizar la multiplicación de cada entrada por su correspondiente peso y las subsecuentes adiciones, por eso es tan importante reducir el número de neuronas al mínimo.

	Red Compleja	Red Real
Dominio	*Dominio de los complejos*	*Dominio de los reales*
ECM	*0,000145*	*0,004972*
Nro de iteraciones	*140*	*500*
Nro de capas ocultas	*1*	*1*
Nro de neuronas ocultas	*15*	*19*
Tasa de aprendizaje	*adaptativa*	*0,002*
Momento	*0,9*	*0,9*
Tiempo de entrenamiento	*13.0174 segundos*	*36.1634 segundos*
Pesos	*Complejos, inicializados randomicamente*	*Reales,inicializados randómicamente*
Algoritmo de Entrenamiento	*BP – complejo*	*BP- real*
Función de activación capa oculta	*casi-sigmoidal* $\log sig = \frac{2c_1}{1+e^{-c_2 z}} - c1$	*tangencial hiperbólica* $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Función de activación capa de salida	*Tangencial* $\tan(z) = \frac{e^{iz} - e^{-iz}}{i(e^{iz} + e^{-iz})}$	*Sigmoidal* $\log sig(x) = \frac{1}{1+e^{-x}}$

Tabla 8. Red real versus Red Compleja

Capítulo 5
Simulación y Discusión de Resultados

Luego de haber presentado la definición del problema, la técnica de predistorsión a utilizar y el diseño e implementación de esta solución, es hora de verificar si los resultados son los esperados, es decir, si se ha logrado compensar eficientemente la distorsión introducida por el AP en el sistema de transmisión a un nivel aceptable. Para ello se realiza la simulación de ambos compensadores neuronales, real y complejo, una vez optimizados y fijos todos los parámetros del mismo.

Las simulaciones se realizarán mediante MatLab y los resultados se presentarán de forma de extraer el mayor conocimiento posible acerca del comportamiento de las redes neuronales y especialmente del algoritmo de aprendizaje utilizado en su entrenamiento, su eficiencia, eficacia y rendimiento para este tipo de problema (la aproximación de una función compleja) en particular.

Para el justo análisis de los resultados arrojados por la simulación se examina la calidad de la modulación del sistema de transmisión del cual forma parte el compensador neuronal.

Existen parámetros de medición y evaluación que permiten discriminar cuando la calidad de una señal es buena o mala y que son ampliamente utilizados en el campo de las telecomunicaciones hoy en día, entre ellos se encuentra el EVM y el análisis de los diagramas de constelación.

5.1 EVM del Compensador Real versus EVM del Compensador Complejo

Para *medir* la calidad de la señal transmitida se utiliza el parámetro cuantitativo EVM o Error Vector Magnitud que se define como la relación, en porcentaje, entre la magnitud de error promedio de los símbolos recibidos y la magnitud del símbolo original. El EVM permite conocer la calidad de la modulación, y cuantifica la desviación de los símbolos recibidos a partir de su posición ideal, sin tomar en cuenta la naturaleza de las afecciones que provocan la distorsión. Como dato de referencia, es conocido que niveles de EVM de hasta 2% son considerados convenientes para señales QAM en sistemas americanos y europeos. La tabla 9 muestra los valores de EVM para la modulación en 16 y 64-QAM en un sistema de transmisión con y sin predistorsionador. El predistorsionador neuronal es real y su entrenamiento se realizó utilizando el algoritmo Backpropagation en su versión tradicional (real). La ganancia obtenida por el sistema de transmisión se calcula restando los coeficientes obtenidos del sistema de transmisión sin y con predistorsión.

EVM [%]	Sin PD-R	Con PD-R	ganancia
64-QAM	78,04	66,34	11,7
16-QAM	126,55	78,04	48,51

Tabla 9. EVM en porcentaje sin/con predistorsion red real

La tabla 10 muestra los valores de EVM para la modulación en 16 y 64-QAM en un sistema de transmisión con y sin predistorsionador Complejo. La red se entrenó utilizando el algoritmo Backpropagation Complejo. Se puede apreciar que el EVM de la red compleja es significativamente menor que el EVM de la red real, es decir, la desviación de los símbolos desde su posición original es menor y por ende, la calidad de la señal es mayor

EVM [%]	sin PD-C	Con PD-C	ganancia
64-QAM	54,09	1,55	52,54
16-QAM	36,36	1,45	34,91

Tabla 10. EVM en porcentaje sin/con predistorsion red compleja

En las tablas 11 y 12 se muestran los valores del parámetro EVM en decibelios (dB). Esto se calcula:

$$EVM\left[dB\right] = 10 * \log_{10}(EVM)$$

Cuando se trata de decibeles, mientras mejor sea la calidad de la señal menor es el valor del EVM: una buena calidad de transmisión de la señal en EVM expresado en decibeles, deberá ser un número negativo más grande que el EVM resultante de una señal de pobre calidad.

EVM [dB]	Red Real	Red Compleja
64-QAM	-1,150	-18,09
16-QAM	-1,65	-18,38

Tabla 11. EVM en decibeles (dB)

La ganancia en del sistema se calcula restando el EVM sin compensador menos el EVM con compensador en [dB]. Se observa que la ganancia obtenida por el sistema de transmisión es claramente mayor cuando éste es compensado por el Predistorsionador Complejo, del orden de 22 veces en el caso de modulación 64-QAM y 7 veces para 16-QAM.

Ganancia [dB]	Red Real	Red Compleja
64-QAM	0,70	15,42
16-QAM	2,09	13,99

Tabla 12. Ganancia en decibeles (dB)

5.2 Análisis de la Constelación

Para *evaluar* la calidad de la señal transmitida se analizan los diagramas de constelaciones, debido a que se puede identificar el problema que aqueja a una señal digital en los diagramas de constelación a simple vista, pues las constelaciones muestran un patrón de símbolos modulados digitalmente e identificar el problema consiste en reconocer el patrón que provoca una desviación respecto a la posición ideal de los símbolos: si cada punto en la constelación corresponde a la posición ideal del símbolo correspondiente, entonces la señal es perfecta y no contendrá errores, por el contrario, si la posición de los símbolos en la constelación se ve alterada, significa que la señal presenta ruido o algún tipo de distorsión. A continuación se analizan los diagramas de constelación arrojados por la simulación del compensador real y del compensador complejo, respectivamente.

5.2.1 Simulación del Compensador Real

Como se ha mencionado en los capítulos anteriores, el predistorsionador real fue configurado con un nodo de entrada, un nodo de salida y diecinueve centros para la función de transferencia tangencial hiperbólica, los cuales fueron inicializados con valores randómicos. El proceso de de entrenamiento de la red fue realizado utilizando 100 muestras de la función amplitud $A(\cdot)$. La red finalmente convergió a un nivel de ECM de 0,004972 después de 500 iteraciones.

En las Ilustraciones 5.1 y 5.2 se presentan sólo los efectos no lineales del Amplificador de tubo sobre los símbolos 16QAM sin y con la técnica de predistorsión propuesta.

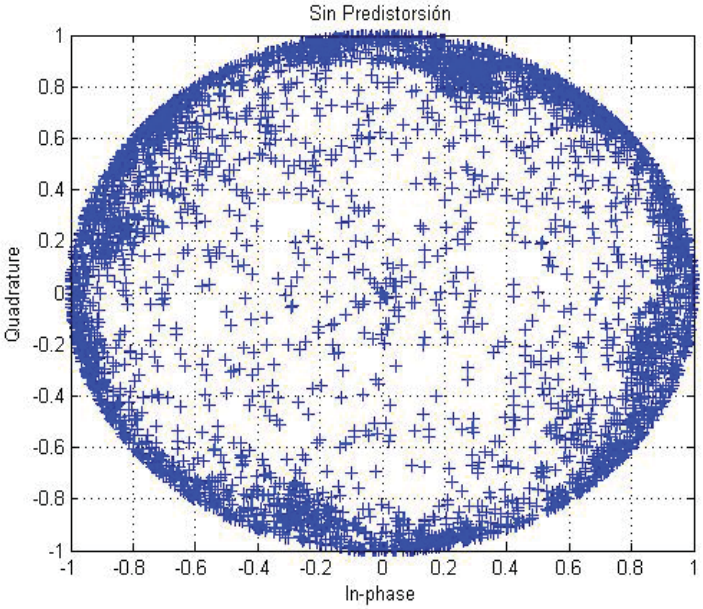


Figura 5.1. Constelación 16-QAM con distorsión no lineal introducida por el Amplificador

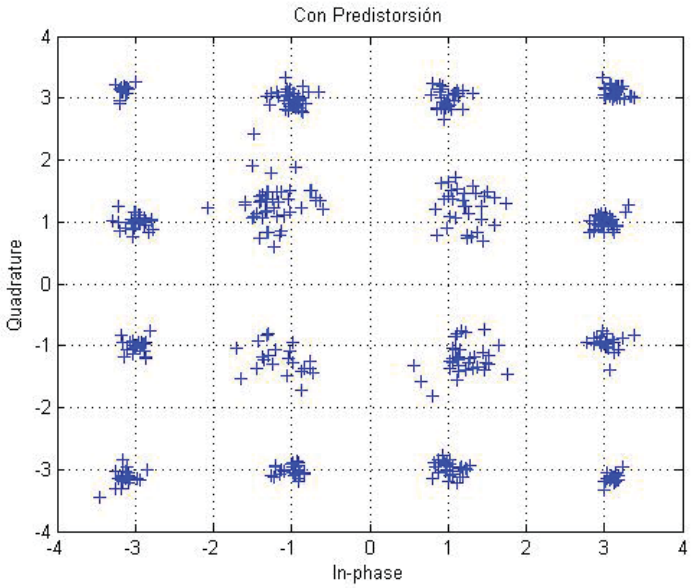


Figura 5.2. Constelación 16-QAM compensada por el Predistorsionador real

En la Figura 5.1 se observa el efecto significativo de distorsión (rotación en amplitud y fase de los símbolos) del Amplificador de Potencia sobre una constelación de símbolos QAM donde 4000 símbolos fueron graficados, la constelación se muestra rotada, producto de un desbalanceo de los ejes IQ.

La figura 5.2 muestra el resultado arrojado por la simulación del compensador neuronal real en un sistema de transmisión con modulación de 16-QAM. Se observa la baja efectividad del predistorsionador real para compensar la deformación de la constelación 16QAM: se observa a simple que vista no logra compensar eficientemente el desbalanceo de los ejes IQ y devolver cada símbolo a su posición correspondiente, sino que aún existe cierto nivel de distorsión en éstos, evidenciado por la posición de los símbolos en la constelación, que aún se ve alterada.

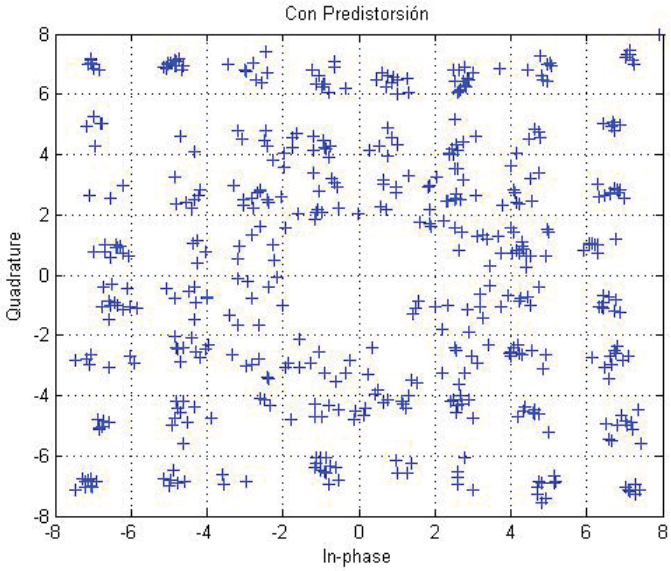


Figura 5.3. Constelación 64-QAM compensada por el Predistorsionador real

La figura 5.3 muestra la constelación 64-QAM resultante de la simulación del compensador real. Puede observarse que en esta constelación los símbolos resultan aún más distorsionados que en su equivalente de 16-QAM, y que en el centro del cuadrante se desdibuja definitivamente la constelación, rotándose todo los símbolos en torno a un eje. Esto se debe a que el espacio de salida de la red (función inversa del amplificador con modulación 64-QAM) es mayor (que modulación 16-QAM) y por tanto resulta más difícil de aproximar.

5.2.2 Simulación del Compensador Complejo

El Predistorsionador complejo, como se mencionó en secciones anteriores, tiene un nodo de entrada, quince nodos ocultos y un nodo de salida no lineal, su ECM de convergencia es considerablemente menor que el predistorsionador real (del orden de 0.000145) y ciertamente necesita menos iteraciones para completar su entrenamiento. Los pesos son complejos e inicializados de manera randómica, El proceso de aprendizaje fue realizado utilizando 100 muestras de la función amplitud $A(\cdot)$ y la constelación fue graficada usando aproximadamente 4000 símbolos transmitidos.

Las curvas de la Figura 5.4 muestran la función directa, inversa y compuesta de la función Amplitud lograda por la red neuronal compleja. Se observa que la función inversa resulta suficientemente aproximada a la curva ideal propuesta (Ilustración 4.5) y la función compuesta es lineal hasta valores de $x = 0{,}92$. Nótese que el amplificador tiene su punto óptimo de eficiencia desde $x = 0{,}8$ a $x = 1$, y es ahí donde la curva resulta más difícil de aproximar.

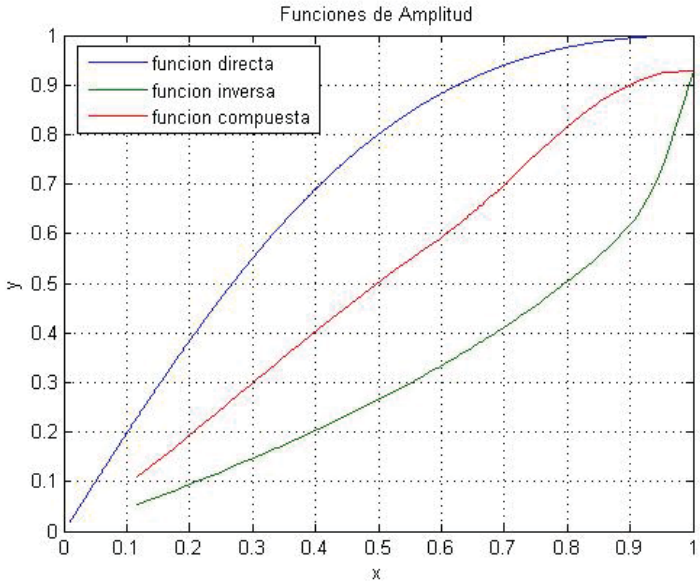


Figura 5.4. Función inversa y compuesta de Amplitud lograda por el Predistorsionador complejo

En la Gráfica 5.5 se muestra la constelación 16QAM resultante de la simulación del Predistorsionador Complejo en MatLab. Es posible observar que es mucho más efectiva en compensar la distorsión no lineal de amplitud y fase que introduce el amplificador de potencia en el sistema de transmisión. Los símbolos prácticamente no se desvían de su posición original y la constelación resultante es muy similar a la constelación ideal (Figura 5.5)

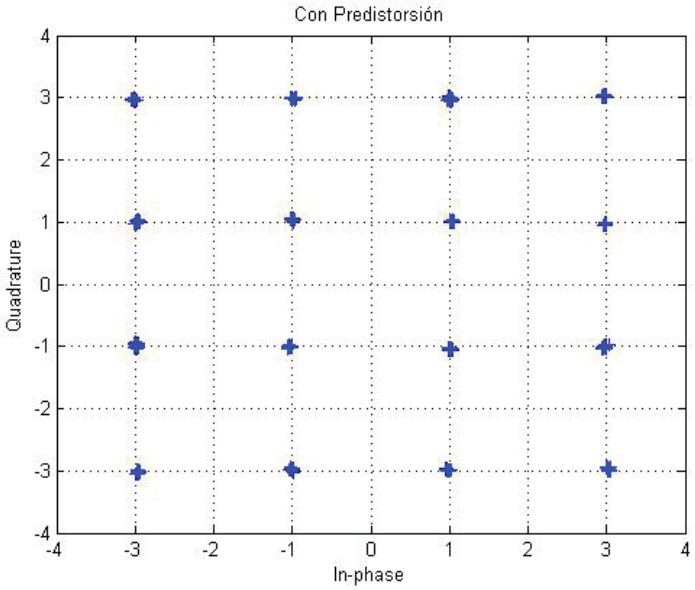


Figura 5.5. Constelación 16-QAM compensada con el Predistorsionador Complejo

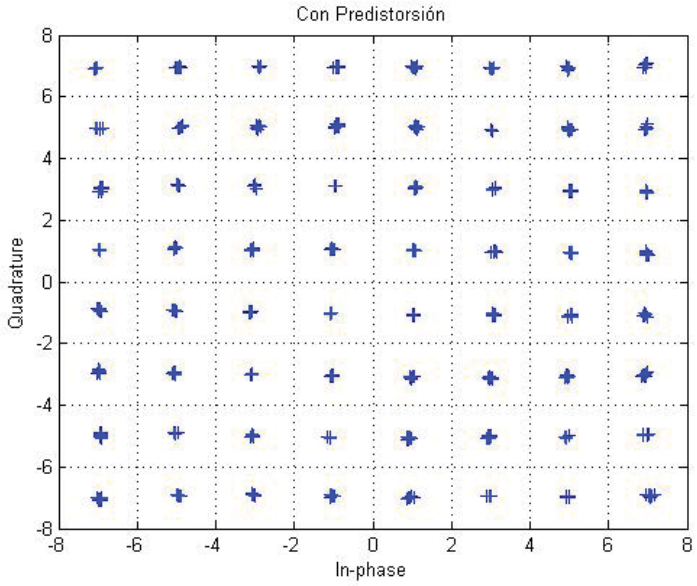


Figura 5.6. Constelación 64-QAM compensada con el Predistorsionador complejo

La constelación de la figura 5.6 es el resultado de la simulación del predistorsionador Complejo para una modulación 64-QAM. Se observa que la red compensa de manera efectiva la distorsión de los símbolos y que estos son perfectamente identificables a simple vista en sus posiciones ideales, sin embargo, se puede apreciar una ligerísima rotación que tiene como eje el centro del cuadrante en la constelación como un conjunto. Al examinar el parámetro EVM se confirma que la predistorsión, aunque sigue siendo eficaz (menor que 2%), es menos eficiente que la predistorsión lograda para una modulación 16-QAM, como se señaló anteriormente, es más complicado compensar una constelación mientras mayor sea el número de símbolos mapeados en ella, es decir, para la red neuronal, compleja o no, es más difícil aproximar una función mientras mayor sea el dominio del espacio de salida.

Capítulo 6

Conclusiones y Trabajo Futuro

Este trabajo de tesis de pre-grado se ha planteado evaluar el rendimiento de dos esquemas de predistorsión basados en redes neuronales para reducir la distorsión no lineal introducida por un amplificador de potencia TWT en un sistema de transmisión con modulación de 16 y 64-QAM.

Cada Predistorsionador se basa en una red neuronal de tipo Multilayer Perceptron cuyo fin último es aproximar la función inversa del amplificador de potencia. El objetivo central de esta tesis radica en comparar el rendimiento de ambas redes neuronales (real y compleja) y su capacidad aproximativa de la función compleja no lineal planteada.

En este informe se entregó un resumen del marco teórico de las Redes Neuronales, específicamente de las redes MLP y su algoritmo de aprendizaje más conocido: el algoritmo Backpropagation, del cual se derivaron sus ecuaciones particulares en el dominio de los números reales y complejos. De este marco se han obtenido conclusiones que han sido tenidas en cuenta en el diseño de la solución.

Se realizó una segunda etapa donde se describe el problema en particular que esta tesis ha examinado (sistema de comunicaciones con distorsión no lineal introducido por un AP) y se diseña la solución con base en la técnica de linealización de predistorsión por medio de redes neuronales. Finalmente se implementó un simulador en MatLab de ambos compensadores neuronales real y complejo y se realizaron pruebas con el fin de configurar correctamente los parámetros más significativos para su óptimo rendimiento.

De la implementación del algoritmo se pueden sacar conclusiones que ayudan a comprender la forma en que se comporta el algoritmo Backpropagation cuando se define en el dominio complejo:

1. Las funciones de activación deben ser complejas y acotadas en el dominio, por eso, se utilizan las funciones especiales llamadas trascendentales elementales, propias del dominio complejo, de otro modo, la red se muestra incapaz de aproximar la función y diverge.

2. La curva de aprendizaje de la red compleja oscila fuertemente en las primeras iteraciones, producto del gran cambio del vector de pesos en el entrenamiento de la red, debido a las particularidades especiales de las ecuaciones complejas que definen algoritmo de aprendizaje Complex-Backpropagation.

De los resultados arrojados por la simulación computacional efectuada de ambos compensadores neuronales, real y complejo, y su posterior evaluación y comparación, se puede concluir:

1. La red neuronal Compleja (Predistorsionador complejo) realiza una aproximación de tipo global de la transformación entrada/salida, lo que favorece su capacidad de generalización y permite una mayor eficacia en la capacidad aproximativa de la función compleja buscada. En cambio, la red neuronal real (Predistorsionador real) realiza una aproximación de tipo local y esto limita su capacidad de generalización, lo que convierte esta red en una mala aproximadora de la función compleja buscada.

2. El Compensador complejo converge más rápido: 130 iteraciones del algoritmo de entrenamiento versus 500 iteraciones de la red real, y converge mejor: logra un ECM de 10^{-5} versus un ECM de 10^{-3} de la red real, esto permite compensar de manera notoriamente más eficiente la distorsión de los símbolos en la constelación 16 y 64-QAM, lo cual se aprecia a simple vista, mediante el análisis del diagrama de la constelación QAM, diagrama que es proporcionado por el simulador y el parámetro EVM de la constelación. El EVM del sistema de transmisión con predistorsionador complejo es menor que 2%, completamente aceptable para los estándares europeos y americanos.

3. Se concluye que la superior capacidad aproximativa de la red neuronal compleja por sobre la red neuronal real, se debe a que la función que aproxima la red es compleja, y una función compleja se aproxima mejor embebida en un medio complejo como lo es el compensador neuronal donde los pesos, funciones de activación, entradas y salidas de la red son complejas

4. Es importante destacar que a pesar del menor número de nodos en la capa oculta de la red compleja (15 nodos ocultos) versus la red real (19 nodos ocultos) , es en realidad la red compleja la que presenta un mayor número de parámetros, debido precisamente a sus coeficientes complejos: 15 nodos ocultos con parte real e imaginaria es cuasi equivalente a considerar 15*2=30 nodos, con el resultado de que la red compleja presenta un mayor número de parámetros que la red real, en consecuencia resulta más costosa la implementación microelectrónica VLSI de la red compleja que la red real.

5. Finalmente, la ganancia en decibeles del sistema de transmisión es mayor en el caso del compensador neuronal complejo; del orden de 22 veces en el caso de modulación 64-QAM y 7 veces para 16-QAM. Esto prueba que tan potente es el compensador de predistorsionador complejo comparado con el predistorsionador real.

6.1 Contribuciones de la Tesis

Dos son las contribuciones que esta tesis aporta al campo de las redes neuronales y, en general, a la discusión:

1. Si bien habitualmente se han utilizado redes neuronales en las técnicas de linearización de amplificadores de potencia, la utilización de una red neuronal compleja con el algoritmo de aprendizaje Complex-Backpropagation en la creación

de un predistorsionador complejo para el amplificador de potencia TWT, utilizando el modelo de M. Saleh [4], es una innovación.

2. Para acelerar la convergencia de la red compleja se utilizó un método de adaptación de parámetros llamado tasa de aprendizaje adaptativa [18], lo que optimizó el rendimiento de la red, disminuyendo considerablemente el ECM (en un 29%), así como el número de iteraciones en el entrenamiento (siendo necesarias sólo 130 iteraciones para converger en vez de 140). Cabe destacar que es la primera vez que este método de adaptación se utiliza en conjunto con el algoritmo Complex-Backpropagation.

6.2 Trabajo futuro

Algunos aspectos que merecen un estudio más detallado en próximos trabajos son los siguientes:

1. Dada la incertidumbre existente en torno al valor que debe asignarse a los parámetros que definen la red neuronal (número de nodos ocultos, tasa de aprendizaje, momento, entre otros) se podría estudiar la posibilidad de optimizar dichos valores mediante el uso de algoritmos genéticos.

2. Mejorar aspectos claves tales como velocidad, carga computacional y precisión de la convergencia en la red neuronal, asimismo minimizar el número de parámetros de la red, de manera de que su implementación microelectrónica no resulte desfavorablemente costosa.

3. Utilizar el algoritmo Complex-Backpropagation combinado con alguna de las muchas técnicas existentes para mejorar la convergencia de la red y reducir el riesgo de los mínimos locales asociado a los métodos de entrenamiento basados en gradiente.

Referencias

[1] Hornik K., *Multilayer feedforward networks are universal approximators*, Neural Networks, Vol 2. pag 359-366, 1989.

[2] Yu Hen Hu, Jenq-Neng Hwang, *Handbook of Neural Network Signal Processing*, Editorial CRC Press, 1999.

[3] Unbehauen Rolf, Luo Fa-Long, *Applied Neural Networks for Signal Processing*, Cambridge University Press, 1999.

[4] Geist T., Kamath H., Foster Porter S., May-Ostendorp P., *Designing Battery Charger Systems for Improved Energy Efficiency*. Preparado para California Energy Commission, 28 de Septiembre, 2006.

[5] Croce Walter, *Cell Phones Demand Better Battery Life*. Julio-Agosto 2004. http://www.wsdmag.com/Articles/Print.cfm?ArticleID=8629

[6] http://en.wikipedia.org/wiki/Quadrature_amplitude_modulation

[7] http://en.wikipedia.org/wiki/OFDM

[8] Saleh M. , Adel A, *Frecuency-Independent and Frecuency-Dependent Nolinear Models TWT Amplifiers*, IEEE Trans. on Comm, Vol. 29, pag. 1715-1719, Noviembre 1981.

[9] Anderson John B., *Digital Transmission Engineering*, publicado 2005, Wiley-IEEE

[10] Benvenuto N., Piazza F., *On the complex Backpropagation Algorithm*, IEE transactions on signal Processing, Vol 40, Nro 4, pag. 967-969, Abril 1992.

[11] Freeman James A, Skapura David M. *Redes neuronales: Algoritmos, aplicaciones y técnicas de programación*. Addison-Wesley, iberoamericana, 1991.

[12] Learning Matlab 7.0 Release 14, COPYRIGHT 1984 - 2004 by The MathWorks, Inc. http://www.mathworks.com

[13] Ramón y Cajal, *Histología del sistema nervioso del hombre y de los vertebrados* Santiago Ramón y Cajal Consejo Superior de Investigaciones Científicas, Madrid, 1911.

[14] Faggin F., *VLSI implementation of neural networks*, International Joint Conference on Neural Networks. Seattle, WA, 1991.

[15**]** Rummelhart D., McClelland J. *Explorations in Parallel Distributed Processi*ng , Editorial MIT Press, Cambridge, 1986.

[16]Turing Alan, *Los números computables, con una aplicación al Entscheidungsproblem*, Proceedings of the London Mathematical Society, 1936.

[17] McCulloch, W.S., Pitts, W. *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, Vol. 5 pag. 115-133, 1943.

[18] Hebb, D.O. *Organization of behavior*. New York, Editorial Science Editions, 1949.

[19] Widrow, B. *Adaptative sampled-data systems, a statistical theory of adaptation*. IRE WESCON Convention Record, parte 4. New York, Institute of Radio Engineers, 1959.

[20] Rosemblatt, R. *Principles of neurodynamics*. New York, Editorial Spartan Books, 1959.

[21] Minsky, M.L., Papert, S. *Perceptrons*. Cambridge MA, Editorial MIT Press, 1969.

[22] Anderson J.A, Silverstein J.W., Ritz S.A., Jomnes R.S. *Distinctive features, categorical perception and probability learning: some aplications of a neural model.* Psychological Review, Nro 84, 1977.

[23] Kohonen, T. *Self-organization and Associative Memory*, Series in Information Sciences, Editorial Springer-Verlag, Vol. 8, Berlín,1984.

[24] Grossberg, S. *Competitive learning: from interactive activation to adaptative resonance*. Cognitive Science, Nro 11, pag. 23-63, 1987.

[25] Rumelhart D.E., McClelland, J.L. & Group, *Parallel Distributed Processing. Explorations in the Microstructure of Cognition.* Cambridge, MA: MIT Press, 1986.

[26] Hoplfield, J.L. *Neural networks and physical systems with emergent collective computational abilities*. Proceedings of the National Academy of Science USA, Nro 79, pag. 2554-2558, 1982.

[27] Sejnowski, T. J., Rosenberg, C. R. *NETtalk: A parallel network that learns to read aloud*, Johns Hopkins University Department of Electrical Engineering and Computer Science Technical Report, 1986.

[28] Fukushima K. *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*. BiologicalCybernnetics, Nro 36, Vol 4, pag. 93-202, 1980.

[29] Armoni A. *Use of neural networks in medical diagnosis*, Medical Computing, Vol 15, pag. 100-1044, 1988.

[30] Clermont G, Angus DC, DiRusso SM, Griffin M, Linde-Zwirble WT. *Predicting hospital mortality for patients in the intensive care unit: A comparison of artificial neural networks with logistic regression models*. Crit Care Med, Vol 29, pag 291-296, 2001.

[31] Ohlsson M, Ohlin H, Wallerstedt SM, Edembrandt L. *Usefulness of serial electrocardiograms for diagnosis of acute myocardial infarction*. Am J Cardiol, Vol 88, pag. 478-481, 2001.

[32] Wigeson D, *Fragmentation Analysis using computer vision techniques*, Master Thesis, University of Witwatersrand, 1987.

[33] Lapedes A., Farber, R. *NonLinear signal processing using neural networks*. IEEE Conference on Neural Information Processing System - Natural and Synthetic, 1987.

[34] Lapedes A., Farber R., *How neural nets work*, in Evolution, learning and cognition, World scientific, pag. 231-346, 1988.

[35] Hopfield, J. J. , Tank, D. W. *Neural computation of decisions in. optimization problems, Biological Cybernetics*, Nro. 52, pag. 141-152, 1985.

[36] Rosenblatt, F., *The Perceptron: A probabilistic Model .for information storage and organization in the brain*, Psycological Review, 1958.

[37] Minsky, M., Papert, S., *Perceptron*: *An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1969.

[38] Widrow B., Hoff M.E., *Adaptive switching circuits*. En 1960 IRE WESCON Convention Record, pag. 96-104, New York, 1960.

[39] Werbos Paul. *Beyond Regresión: New tools for prediction and Anayisis in the behavioral Sciences*. Harvard, Cambridge, 1974.

[40] Deuflhard P., *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms.* Springer Series in Computational Mathematics, Springer, Vol. 35, Berlin, 2004.

[41] Levenberg, *K. A Method for the Solution of Certain Problems in Least Squares*.Quart. Appl. Math. Vol 2, pag. 164-168, 1944.

[42] Marquardt, D. *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, SIAM J. Appl. Math. Vol 11, pag. 431-441, 1963.

[43] Ivorra Castillo Carlos, *Funciones de Variable Compleja de Variable Compleja con Aplicaciones a la Teoría de Números.* ebook, Facultad de Economía, Universidad de Valencia. http://www.uv.es/~ivorra/Libros/Libros.htm

[44] Benvenuto N., Piazza F.*, On the complex Backpropagation Algorithm*, IEE transactions on signal Processing, Vol 40, Nro 4, pag. 967-969, Abril 1992.

[45] Taehwan Kim, Tulay Adali, *Fully Complex Multilayer Perceptron Network for Nonlinear Signal Processing*, the MITRE Corporation & University of Maryland Baltimore County, U.S.A.,1998

[46] Taehwan Kim, Tulay Adali, *Complex Backpropagation Neural Network using elementary transcendental activation function*, the MITRE Corporation & University of Maryland Baltimore County, U.S.A., 1996.

[47] Georgiou G., Koutsougeras C., *Complex Domain Backpropagation*, IEEE Trans on Circuits and systems II, Vol 39, Nro 5, pag. 330-334, 1992.

[48] Plagianakos V.P., Magoulas G.D., y Vrahatis M.N., *Learning Rate Adaptation in Stochastic Gradient Descent,* IEEE transactions on neural networks, Vol 13, Nro. 3, Mayo 2002.

[49] http://es.wikipedia.org/wiki/Etapa_de_potencia

[50] James F. *Amplificadores operacionales y circuitos integrados lineales*, Editorial Paraninfo, Primera Edición, 2002.

[51] O'Droma M. S., Portilla J., Bertran E. *Linearisation Issues in Microwave Amplifiers*, *GAAS Symp*. Amsterdam, pag.199-202, Octubre 2004.

[52] Kenington Peter B., *High-linearity RF Amplifier Design*, Artech House Publishers, Boston, 2000.

[53] Sundstrom L. *Digital RF Power Amplifier Linearisers. Analysis and Design*. PhD thesis, Department of Applied Electronics, Lund University, Agosto 1995.

[54] AmpliX Inc. TWT Linearizers, SSPA Linearizers. www.amplix.com, 2000.

[55] Berglund B. , Nygren T., Sahlman K. *RF multicarrier amplifier for third-generation systems*. Ericsson Online Review. http://www.ericsson.com, April 2001.

[56] IEEE Standards: http://standards.ieee.org/

[57] Tamgnoue V., Bette S., Moeyaert V., Mégret P., *Statistical analysis of M-QAM signal impaired by clipping noise in optical CATV systems,* Proceedings Symposium IEEE/LEOS Benelux Chapter, 2005.

[58] Erkan Acar, Sule Ozev, Redmond Kevin B, *Enhanced Error Vector Magnitude (EVM) Measurements for Testing WLAN Transceivers,* ICCAD'06, Nro 9, Noviembre 5, 2006, San Jose, CA.

[59] Georgiadis A., *Gain, phase imbalance, and phase noise effects on error vector magnitude*. IEEE Transactions on Vehicular Technology, pag. 443–449, Marzo 2004.

[60] Rodríguez Agurto Nibaldo, *Predistorsión de Amplificadores de Potencia usando un Algoritmo de Aprendizaje Híbrido*, XIII Congreso Internacional de Telecomunicaciones, 2006.

[61] Cybenko G., *Approximation by Superpositions of a Sigmoid Function*, Mathematics of control, Signals and Systems, Nro 2, pag. 303-314, 1989

Anexo 1: Código de los Algoritmos de Entrenamiento

```
function red_compleja()
    % prueba.m
    % Se utiliza el algoritmo Complex - Backpropagation con tasa de
    % aprendizaje adaptativa para aproximar la inversa de una funcion compleja.
    % la variable NOCULTAS se refiere a la cantidad de nodos ocultos
    % Este script inicializa todas las variables y llama al archivo.m
    % que realiza en entrenamiento. La red es COMPLEJA

  tic
    clear;
    close all
    echo off;
    load datatr

    LL=100;
    ss=ss(1:LL);
    N = LL;
    rs=abs(ss);
    wo=max(abs(rs));
    r=rs./wo;
    r=sort(r);
    NOCULTAS=15;
    r2=r.^2;
    A=2*r./(1+r2);
    P=(2.*r2)./(1+r2);
    P=P.*(pi/6);
     Zt=A.*exp(j.*(angle(r)+P));
    Z=abs(Zt);
    Patrones=Z;
    Salida=r;

    [NINPUTS, NPAT]  = size(Patrones);
    [NOUTPUTS, NPAT] = size(Salida);
    Inputs1 = Patrones;

    Momentum = 0.903;
    alfa =  .00215;
    ECM_Limite = 0.00001;
```

```
  Pesos1 = complex(InitPesos1,x);
  Pesos2 = complex(InitPesos2,y);

  deltaP1 = 0;
  deltaP2 = 0;

  clf reset, whitebg(gcf,[0 0 0]), hold on, box on
 xlabel('x'), ylabel('f(x)^-1')

  E2(1,15)=0;
  E1(1,15)=0;
  E3(1,15)=0;
  epoch = 0;

  for i = 1:140

   epoch = epoch+1;
   bp_ciclo_inv_ta

 if rem(epoch-1,1) == 0
   figure(proyecto)
   PlotApprox(Patrones,Result1,Salida,Result2);
   title('Aproximando la funcion')
   xlabel('x'), ylabel('f(x)^-1')
   fprintf('%f\n',ECM);
   T2(i)=ECM;
   drawnow
 end

  if ECM < ECM_Limite, break, end

  end

save pesos  Pesos1  Pesos2
  figure(figura)
  plot(Z,Result2); title('Funcion Inversa Aproximada');
  grid; grid on
  xlabel('x'); ylabel('f(x)^[-1]');

  toc
  t=toc
```

```
% bp_ciclo_inv_ta.m
      %
      % ciclo interno del algoritmo Complex-backpropagation, con una capa oculta de 15
      % nodos. en la capa oculta la funcion de transferencia es sigmoidal y
      % tangente en la capa de salida, ambas funciones complejas. Los
      % pesos son también complejos al igual que la entrada y salida de
      % la red
      %
      % Se usan los siguientes parametros de entrada/salida
      %    Inputs1   -  Patrones de entrada
      %    Salida    -  Salida de Patrones
      %    alfa      -  tasa de aprendizaje adaptativa
      %    Momentum  -  Momento
      %    Pesos1    -  pesos de la primera capa (actualizada en esta rutina)
      %    Pesos2    -  pesos de la segunda capa (actualizada en esta rutina)
      %    deltaP1   -  delta de los pesos inicializado en cero
      %    deltaP2   -  delta de los pesos de la segunda capa, inicializado en cero
      %    ECM       -  Error Cuadratico Medio

       c1=0.45;c2=0.82;
       y1=0.000135; y2=0.00022;

      % Forward del algoritmo: propagación hacia adelante

      Neta1 = Pesos1 * Inputs1;
      Result1 =2*c1./(1+exp(-c2.*Neta1))-c1;
      Inputs2 =Result1;
      Neta2 = Pesos2 * Inputs2;
      Result2 =tan(Neta2);

      % Backward : Propagacion de errores hacia atras

      Result2Error = (Result2 - Salida);
      ECM = sum(abs(Result2Error).^2)./NPAT;
      In2Error = Result2Error .* (( c2/(2*c1)*(c1^2-(2*c1./(1+exp(-c2.*Neta2))-c1).^2) ) );
      Result1Error = Pesos2' * In2Error;
      In1Error=Result1Error.* (sec(Neta1).^2) ;

      dP2 = In2Error * Inputs2';
      dP1 = In1Error * Inputs1';

      % Adaptación de la tasa de aprendizaje

      if epoch >2
         E3=E1;
         E1=E2;
         E2=dP2;
```

```
end

% parámetros: Momentum=0.903 alfa=0.00215  y1=0.00013;
% y2=0.00022 ECM =.000155 epoch=141

alfa= alfa +y1.*(E1.'*dP2)+ y2.*(E3.'*E2); %0.000147 ECM disminuyó en un 74%
alfa=mean(abs(alfa));
alfa=mean(alfa);

% Actualización de Pesos

deltaP2 = -alfa * dP2 + Momentum * deltaP2;
deltaP1 = -alfa * dP1 + Momentum * deltaP1;

Pesos2 = Pesos2 + deltaP2;
Pesos1 = Pesos1 + deltaP1;
```

```
function red_real()
    % prueba.m
    % Se utiliza el algoritmo Complex - Backpropagation con tasa de
    % aprendizaje adaptativa para aproximar la inversa de una funcion compleja.
    % la variable NOCULTAS se refiere a la cantidad de nodos ocultos
    % Este script inicializa todas las variables y llama al archivo.m
    % que realiza en entrenamiento. La red es COMPLEJA

    tic
    clear;
    close all
    echo off;
    load datatr

    LL=100;
    ss=ss(1:LL);
    N = LL;                .
    rs=abs(ss);
    wo=max(abs(rs));
    r=rs./wo;
    r=sort(r);
    NOCULTAS=19;
    r2=r.^2;
    A=2*r./(1+r2);
    P=(2.*r2)./(1+r2);
    P=P.*(pi/6);

    Zt=A.*exp(j.*(angle(ss)+P));
    Z=abs(Zt);
    Patrones=Z;
    Salida=r;

    [NINPUTS, NPAT]  = size(Patrones);
    [NOUTPUTS, NPAT] = size(Salida);
    Inputs1 = Patrones;

    Momentum = 0.9;
    alfa =  .002;
    ECM_Limite = 0.00001;

    InitPesos1=rand(NOCULTAS,1);
    InitPesos2=rand(1,NOCULTAS);

    Pesos1 = InitPesos1;
    Pesos2 = InitPesos2;

    deltaP1 = 0;
```

```
  deltaP2 = 0;

  clf reset, whitebg(gcf,[0 0 0]), hold on, box on
  xlabel('x'), ylabel('f(x)^-1')

 E2(1,NOCULTAS)=0;
 E1(1,NOCULTAS)=0;
  E3(1,NOCULTAS)=0;
  epoch = 0;

  for i = 1:500

  if epoch >80
     alfa=0.0025;
 end
   epoch = epoch+1;
   bp_ciclo_red_real

if rem(epoch-1,1) == 0
   figure(proyecto)
   PlotApprox(Patrones,Result1,Salida,Result2);
   title('Aproximando la funcion')
   xlabel('x'), ylabel('f(x)^-1')
   drawnow
end

  if ECM < ECM_Limite, break, end

  end

save pesosreal  Pesos1  Pesos2

  figure(figura)
  plot(Z,Result2); title('Funcion Inversa Aproximada');
  grid; grid on
  xlabel('x'); ylabel('f(x)^[-1]');

   toc
   t=toc
```

```
% bp_ciclo_red_real.m
      %
      % ciclo interno del algoritmo backpropagation, con una capa
      % oculta de 19 nodos. en la capa oculta la funcion de transferencia tangencial
hiperbolica y
      % sigmoidal en la capa de salida, ambas funciones reales. Los
      % pesos son también complejos al igual que la entrada y salida de
      % la red
      %
      % Se usan los siguientes parametros de entrada/salida
      %    Inputs1   -  Patrones de entrada
      %    Salida    -  Salida de Patrones
      %    alfa      -  tasa de aprendizaje adaptativa
      %    Momentum  -  Momento
      %    Pesos1    -  pesos de la primera capa (actualizada en esta rutina)
      %    Pesos2    -  pesos de la segunda capa (actualizada en esta rutina)
      %    deltaP1   -  delta de los pesos inicializado en cero
      %    deltaP2   -  delta de los pesos de la segunda capa, inicializado en cero
      %    ECM       -  Error Cuadratico Medio


      y1=0.0001; y2=0.0002;

      % Forward del algoritmo: propagación hacia adelante

      Neta1 = Pesos1 * Inputs1;
      Result1 =tanh(Neta1);
      Inputs2 =Result1;
      Neta2 = Pesos2 * Inputs2;
      Result2 = logsig(Neta2);

      % Backward : Propagacion de errores hacia atras

      Result2Error = (Result2 - Salida);
      ECM = sum(abs(Result2Error).^2)./NPAT;
      In2Error = Result2Error .* (cosh(Neta2).^(-2));
      Result1Error = Pesos2' * In2Error;
      In1Error=Result1Error.* (exp(-Neta1)./(1+(exp(-Neta1).^2)) ) ;

      dP2 = In2Error * Inputs2';
      dP1 = In1Error * Inputs1';

      % Adaptación de la tasa de aprendizaje

      if epoch >2
        E3=E1;
        E1=E2;
```

```
    E2=dP2;
end

  alfa= alfa +y1.*(E1.'*dP2)+ y2.*(E3.'*E2);
  alfa=mean(abs(alfa));
  alfa=mean(alfa);

  % Actualización de Pesos

  deltaP2 = -alfa * dP2 + Momentum * deltaP2;
  deltaP1 = -alfa * dP1 + Momentum * deltaP1;

  Pesos2 = Pesos2 + deltaP2;
  Pesos1 = Pesos1 + deltaP1;
```

Anexo 2: Resultados de la Ejecución del Simulador

A.2.1 Simulación del Predistorsionador Real 16-QAM

El simulador del esquema de predistorsión programado en MatLab calcula los coeficientes del compensador neuronal, posteriormente, a partir de nuevas muestras estima la salida del predistorsionador y a continuación, las muestras pasan por el modelo del amplificador TWT.

En la gráfica siguiente, se muestran los resultados de esta simulación para el compensador neuronal real inserto en un sistema de transmisión de 16-QAM: se puede apreciar la constelación ideal 16-QAM, los símbolos transmitidos (4000 símbolos) y finalmente la constelación resultante de la salida del sistema sin predistorsión y la constelación que origina el mismo sistema con predistorsión.

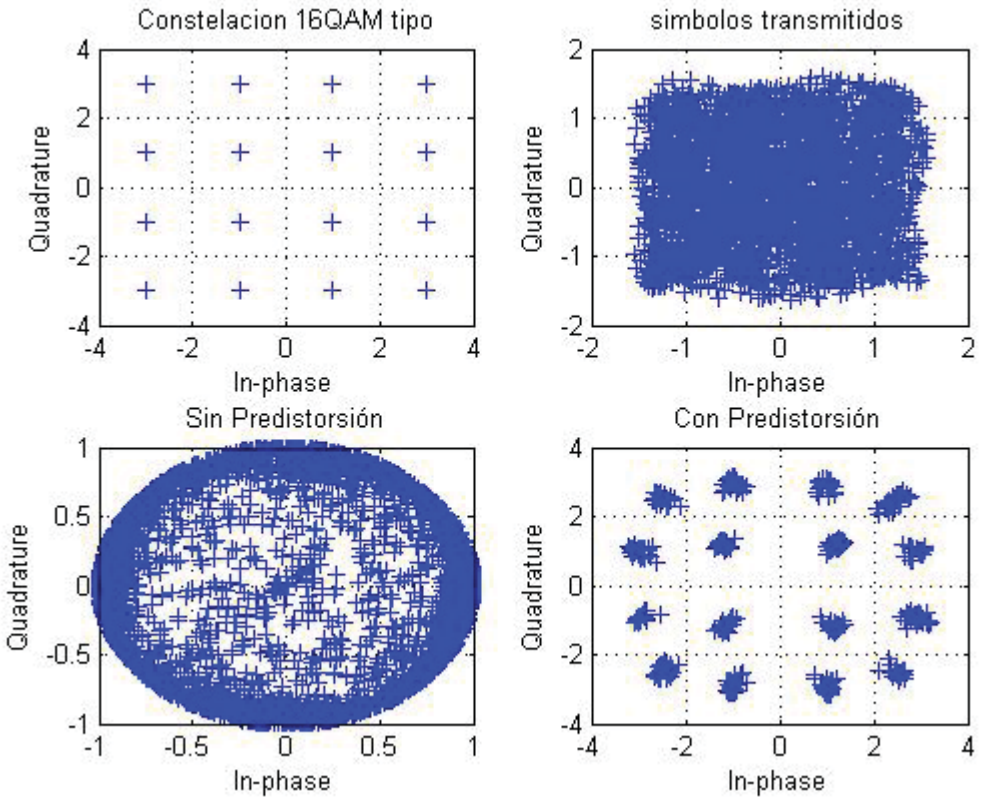


Figura A.1 Resultados para la simulación del Compensador Real en 16-QAM

A.2.2 Simulación del Predistorsionador Real 64-QAM

La figura muestra el rendimiento del predistorsionador real para un sistema de transmisión de 64-QAM, también se puede apreciar la constelación ideal 64-QAM, los símbolos transmitidos y la constelación resultante sin predistorsión.

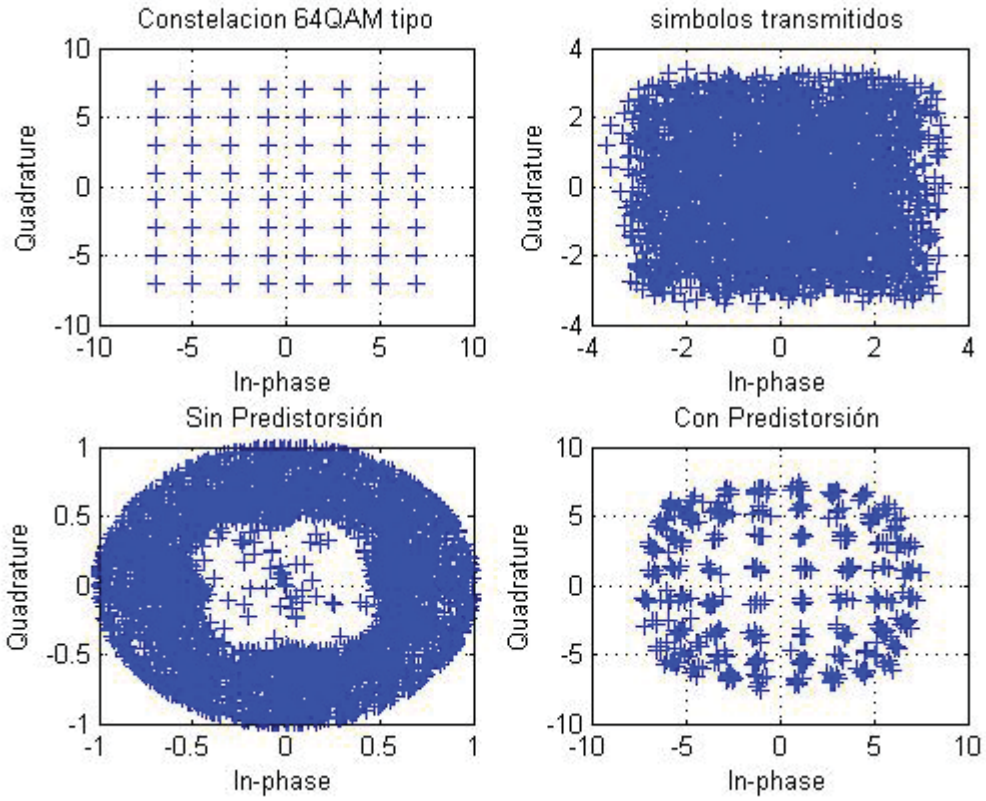


Figura A.1 Resultados para la simulación del Compensador Real en 64-QAM

A.2.3 Simulación del Predistorsionador Complejo 16-QAM

En la ilustración se muestran los resultados de la simulación del compensador neuronal complejo en un sistema de transmisión de 16-QAM: se puede apreciar la constelación ideal QAM, los símbolos y por último, la constelación resultante de la salida del sistema sin predistorsión y la constelación que origina el mismo sistema, pero con predistorsión.

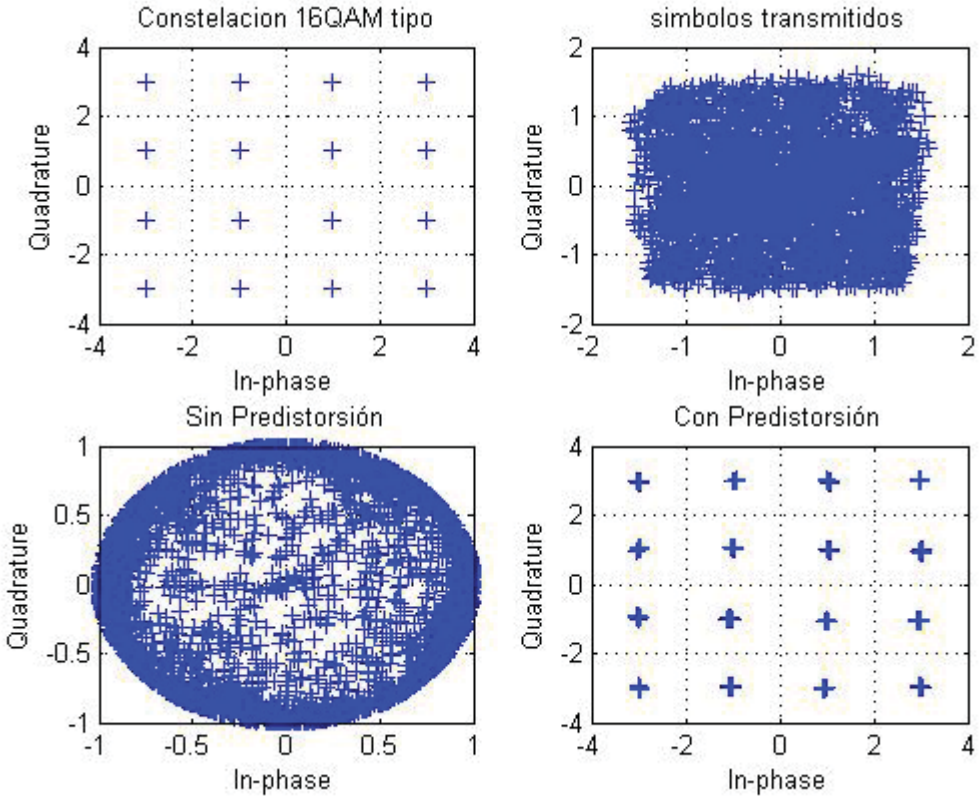


Figura A.1 Resultados para la simulación del Compensador Complejo en 16-QAM

A.2.4 Simulación del Predistorsionador Complejo 64-QAM

Se muestra los resultados arrojados por la simulación del compensador complejo para un sistema de transmisión 64-QAM. Como en los gráficos anteriores, se aprecia la constelación ideal, los símbolos transmitidos y la constelación resultante con y sin esquema de predistorsión.

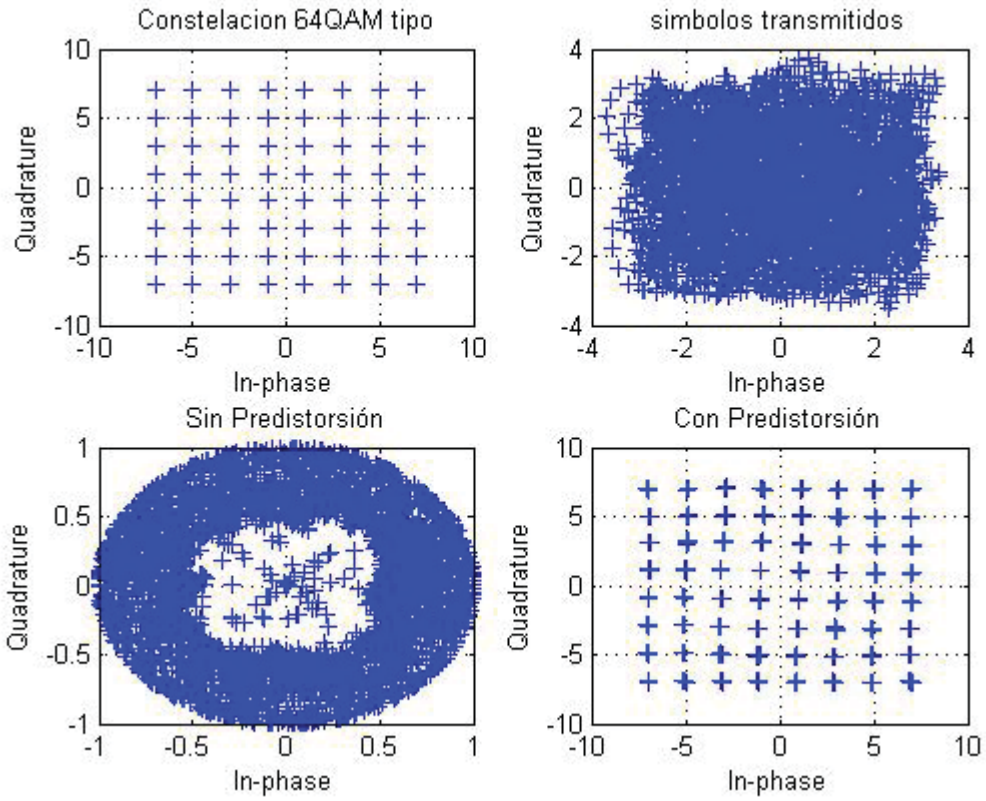


Figura A.1 Resultados para la simulación del Compensador Complejo en 64-QAM