HERRAMIENTA SOFTWARE PARA LA IDENTIFICACIÓN DE SISTEMAS DINÁMICOS NO LINEALES BASADO EN MODELOS DE REDES NEURONALES ARTIFICIALES

URIEL CHINCHILLA CHINCHILLA
GUSTAVO PÉREZ ARIAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2007

HERRAMIENTA SOFTWARE PARA LA IDENTIFICACIÓN DE SISTEMAS DINÁMICOS NO LINEALES BASADO EN MODELOS DE REDES NEURONALES ARTIFICIALES

URIEL CHINCHILLA CHINCHILLA
GUSTAVO PÉREZ ARIAS

TRABAJO DE GRADO PRESENTADO PARA ACCEDER AL TÍTULO DE INGENIERO DE SISTEMAS.

DIRECTOR
PhD. JORGE LUÍS CHACÓN VELASCO
Profesor de la Escuela de Ingeniaría Mecánica, UIS

CODIRECTOR
MSc. JUAN CARLOS REYES FIGUEROA
Profesor Escuela de Ingeniería de Sistemas, UIS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2007

A Dios, que bendice mi vida,
A Sebastián y Deyanira mis padres,
Por formar un hombre de bien,
A Tatiana por creer en mí
Y Uriel Sebastián e Iván Leonardo
Por hacerme tan feliz

URIEL

Dedicado a Dios por bendecirme
A Alix Maria mi Madre,
Por formar hombres con principios
A Arturo mi hermano
Por el apoyo económico,
A Vanesa mi hija,
Por traer mi felicidad.

GUSTAVO

Los autores expresan sus agradecimientos a:

PhD. JORGE LUÍS CHACÓN VELASCO. Director. Por la confianza depositada en nosotros para desarrollar esta investigación.

MSc. JUAN CARLOS REYES FIGUEROA. Codirector. Por su valioso apoyo, orientación y por su dedicación a esta investigación.

A nuestras familias, por su apoyo y tolerancia

TABLA DE CONTENIDO

LISTAS DE FIGURAS

LISTAS DE TABLAS

TABLA DE SÍMBOLOS

Símbolo	Descripción	Símbolo	Descripción
@	Arroba, en direcciones de correo	.pas	Extensión de archivos en Delphi
?	Derivada	Radianes	Unidad de ángulo
?	Sumatoria	RBF	Función de Base Radial
=	Menor igual	RBFN	Red neuronal de Función de Base Radial
=	Mayor igual	RNA	Red neuronal artificial
2D	Dos dimensiones	Sen ?	Seno del ángulo
Bar	Unidad de presión	UML	Lenguaje de modelado unificado
C1	Cilindro 1	W_i	Peso de la neurona i
C2	Cilindro 2	X_i	Entrada i
C3	Cilindro 3	Y_i	Salida i
C4	Cilindro 4	p	Pi igual 3.1416
Cos ?	Coseno de ángulo	f_i	Valor de entrada a la neurona i
d_i	Distancia i	?	Exponencial
IEEE	Bases de datos	.txt	Extensión de archivo de datos de texto
MCI	Motores de combustión interna	.xls	Extensión de archivo para Excel
PMC	Perceptrón multicapa	UPV	Universidad Politécnica de Valencia
PC1_90	Presión en el cilindro 1 a 900 RPM	PC1_140	Presión en el cilindro 1 a 1400 RPM
PC1_340	Presión en el cilindro 1 a 3400 RPM	RPM	Revoluciones por minuto

TÍTULO

HERRAMIENTA SOFTWARE PARA LA IDENTIFICACIÓN DE SISTEMAS DINÁMICOS NO LINEALES BASADO EN MODELOS DE REDES NEURONALES ARTIFICIALES[*]

AUTORES: CHINCHILLA CHINCHILLA, Uriel
PEREZ ARIAS, Gustavo[**]

PALABRAS CLAVES: REDES NEURONALES ARTIFICIALES, IDENTIFICACIÓN DE SISTEMAS DINÁMICOS

DESCRIPCIÓN

Las Redes Neuronales Artificiales (RNA) se han convertido en unas de las metodologías mas efectiva para estudiar, desarrollar y describir los sistemas dinámicos no lineales, la variantes que las RNA nos ofrece para investigar el comportamiento de estos sistemas, ha hecho que en actualidad se están aplicando en importantes campos de la economía donde están presente los sistemas dinámicos no lineales.

Precisamente, las RNA han demostrado su competencia en el estudio de este tipo de sistemas, ya que facilitan la generación de modelos que entregan resultados más reales, en comparación con los obtenidos mediante el uso de complejos modelos matemáticos que en su mayoría no alcanzan la suficiente exactitud requerida para resolver estos Sistemas.

Esta investigación esta enfocada, en demostrar que las RNA son aplicables en la identificación de un sistema de estas características; nos obstante se a tomado como ejemplo el estudio del comportamiento de los émbolos dentro un motor de combustión interna, donde se realiza un análisis detallado, buscando reducir el error de diagnostico y soportar la toma de decisiones de los analistas de motores.

Solo se requieren los datos de entrada de este sistema dinámico no lineal correspondientes al ángulo de recorrido del cigüeñal contra presión generada por el émbolo en condiciones especificas de carga y velocidad, para que la Red Neuronal Artificial (RNA) diseñada y previamente entrenada sea capaz de identificar su comportamiento dinámico y determinar el estado en el que se encuentra el motor a partir de un análisis particular de cada émbolo, donde se puede detectar si alguno de estos presenta una falla. Este diagnóstico le permitirá al usuario precisar el procedimiento a seguir, tomar la decisión sobre las reparaciones oportunas para la máquina, buscando aumentar la vida útil, disminuir los costos y el tiempo de inactividad del vehiculo.

---

[*] Trabajo de grado

[**] Facultad de Ciencias Físico-Mecánicas, Ingeniería de Sistemas e Informática, PhD. Jorge Luís Chacón Velasco

TITLE

SOFTWARE TOOL FOR THE IDENTIFICATION OF DYNAMIC NOT LINEAR SYSTEMS BASED ON MODELS OF ARTIFICIAL NEURAL NETWORKS[*]

AUTHORS: CHINCHILLA CHINCHILLA, Uriel
PERES ARIAS, Gustavo[**]

KEY WORDS: ARTIFICIAL NEURAL NETWORKS, IDENTIFICATION OF DYNAMIC SYSTEMS

DESCRIPTION

The Artificial Neural Networks (RNA) has become one of the most effective methodologies to study, develop and describe nonlinear dynamic systems, the number of variants that RNA offers us to investigate the behavior of these systems has made this application has increase to such an extent, that currently are being implemented in key areas of the economy where this is the nonlinear dynamic systems.

Indeed, the RNA have demonstrated their competence in the study of such systems, this facilitate the generation of models that deliver real more results, compared with those obtained by using complex mathematical models that most do not achieve sufficient accuracy required to solve these systems.

This research is focused on demonstrating that the RNA are applicable in identifying such a system, however, we are taken as an example to study the behavior of the piston inside an internal combustion engine, which performs a detailed analysis, seeking to reduce the error diagnostic and support the decision-making of analysts engines.

It only requires the input for this dynamic non-linear system, corresponding of the crankshaft angle against pressure generated by the piston in a position specific load and speed, so that the network Neuronal Artificial (RNA) previously designed and trained, be able identify your dynamic behavior and identify the state in which the find engine based on an analysis of each piston, which can detect if any of these presents a flaw. This diagnosis will allow the user to specify the procedure to be followed, the decision on the repairs necessary for the machine, looking to increase service life, reduce costs and downtime for the car.

---

[*] Work of degree

[**] Faculty of Physical-Mechanics Engineering's, Systems and Informatic Engineering, PhD. Jorge Luís Chacón Velasco

CAPITULO 1

1 INTRODUCCIÓN

Desde hace varios años se ha venido incorporando nuevas tecnologías en la búsqueda de optimizar procesos que permitan el tratamiento de aquellos casos los cuales evidencian un comportamiento no lineal, son los mismos que han puesto en dificultad los métodos tradicionales de estudio por su elevada complejidad, pues dichos métodos ha demostrado su debilidad para abordar y entregar soluciones acordes con la realidad.

Ante este panorama, los especialistas se habían mostrado impotentes para estudiar y resolver este tipo de casos, lo que hizo que surgieran nuevas ideas que demandara una lógica hábil y acertada para la investigación de los mismos, así fue como apareció en escena la definición de los sistemas dinámicos no lineales, pero el siguiente problema por remediar seria desarrollar metodologías consecuentes a este razonamiento que permitieran entregar respuestas satisfactorias a este tipo específico de problemas.

Después de apostarle a varias ideas se llego a la conclusión que la mejor manera de manejar estos casos seria mediante la utilización de redes neuronales artificiales (RNA), las cuales en los estudios preliminares surgieron como metodología eficaz para tratar, estudiar y responder los interrogantes que esta clase de sistemas habían manifestado, aún cuando esta metodología no gozara de la aprobación de mayoría de la comunidad científica.

No obstante, durantes las últimas cinco décadas el desarrollo de Las RNA ha venido evolucionando al punto de colmar las expectativas propuestas inicialmente por sus ideólogos, pues han demostrado su capacidad de otorgar respuestas confrontables, lo que ha hecho que hoy en día sean las más utilizadas para resolver los casos donde se abordar la identificación de sistemas dinámicos no lineales.

Precisamente, las RNA emergen como una metodología importante en el tratamiento de estos casos de sistemas dinámicos no lineales, por el apoyo que pueden suministrar cuando se requiere información detallada del sistema y la eficacia con la que pueden llegar a resolverlo, ya que son especialmente inteligentes para determinar una respuesta rápida a un problemas cuando previamente se le ha entrenado, reduciendo ostensiblemente la complejidad que demanda los sistemas dinámicos no lineales.

1.1 PLANTEAMIENTO DEL PROBLEMA

La creciente demanda de tecnología en nuestra sociedad requiere de nuevos enfoques para los problemas actuales de control e identificación de sistemas. Se conoce de la importancia, pero también de la complejidad en la Identificación de Sistemas Dinámicos No Lineales, no obstante, este tipo de sistemas se han convertido en un problema de difícil solución utilizando metodologías clásicas de identificación.

Este tema es de gran potencial investigativo y aplicabilidad, debido a la amplia gama de comportamientos físicos que describen los Sistemas Dinámicos No lineales. La identificación de sistemas trata la estimación de modelos dinámicos, a partir, de los datos observados, aunque, solo se conoce la información de entrada-salida, con las consecuencias que esto implica: incertidumbres paramétricas, dinámicas no modeladas, perturbaciones externas o incertidumbres mixtas, ruido en mediciones, etc., podemos predecir y controlar su comportamiento.

El ejercicio seleccionado para la Identificación de Sistemas Dinámicos no lineales, es el comportamiento de los cilindros de un motor de combustión interna. Existen diferentes causas por las cuales los cilindros pueden perder presión; las más comunes son:

- Por uso prolongado del motor
- Por fallas de sistema alimentador de corriente
- Por lubricación deficiente
- Por fallas en el sistema de refrigeración.
- Por defectos de manejo, sobrepasando los limites superior e inferior de las revoluciones establecidas para el motor.

Esto repercute directamente en el funcionamiento global del motor; lo óptimo es determinar el momento en que el émbolo comienza fallar y realizar la reparación inmediata, pues así se reduce en costos y tiempos de reparación. El problema surge cuando se debe diagnosticar el estado del motor, para conocer su confiabilidad y no es fácil acceder a tecnologías que permitan comprobar si realmente el motor esta funcionando correctamente, o por el contrario, se debe efectuar reparaciones a su interior para corregir las fallas técnicas.

Actualmente, en nuestro entorno, no se conoce un software que aborde esta problemática, ósea, basarse en cálculos computacionales para determinar si un cilindro esta trabajando adecuadamente al interior de un motor, sólo se puede establecer su funcionamiento empleando la técnica; recurriendo a los servicios de un mecánico automotriz quien desarma la máquina, para examinar el rastro que deja el recorrido del cilindro al interior de la camisa; lo que conlleva a gastos exagerados y la inutilización del vehículo, lo cual puede resultar improductivo.

Lamentablemente, en nuestra sociedad poco se utiliza el mantenimiento preventivo del vehículo, en el caso del motor por lo imperceptible de sus fallas y por los altos costos de la reparación, se deja llegar al peor escenario; que es cuando el motor pierde su fuerza y comienza a consumir aceite, en este instante los costos de reparación se elevan sustancialmente, por que la máquina se ve afectada en conjunto y no particularmente en el cilindro como se quiere diagnosticar con la herramienta software propuesta.

Solo se necesitan los datos de presión en el cilindro al momento de la combustión al interior del motor y la complejidad de diagnosticar su funcionamiento queda sujeto a la solución del sistema, debido a que este se describe como un caso de Identificación de Sistemas Dinámicos No lineales. Por esto, el desarrollo de esta aplicación ofrece un avance significativo en la investigación de este tipo de casos, por la necesidad y utilidad en nuestro entorno.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

Diseño, desarrollo e implementación de una herramienta software fundamentada en modelos de redes neuronales artificiales, que identifica un sistema dinámico no lineal a partir de sus datos experiméntales de entrenamiento.

1.2.2 OBJETIVOS ESPECÍFICOS

- Realizar la revisión del estado del arte, para determinar las metodologías de Identificación de Sistemas Dinámicos no lineales utilizadas en la actualidad.

- Diseñar una herramienta software basada en Redes Neuronales Artificiales para la Identificación de Sistemas Dinámicos no lineales, aplicando la metodología de prototipado y UML.

- Diseñar una interfaz hombre-máquina, que facilite el manejo de la herramienta software.

- Desarrollar e implementar una herramienta software que permita:

    - Introducir datos de entrada, mediante archivos tipo texto
    - Escoger el tipo de red a utilizar
    - Definir la topología de la red
    - Visualizar los resultados obtenidos

- Probar la herramienta software con un problema específico, partiendo de datos experiméntales, comparando la respuesta obtenida con la encontrada por métodos clásicos de identificación.

- Elaborar un manual de instalación, operación y mantenimiento

1.3 JUSTIFICACIÓN

Existen diversos sistemas complejos que no pueden ser descritos y estudiados por la ciencia clásica, como la identificación de Sistemas Dinámicos no lineales. Para abordar este tipo de problemas, una alternativa con bastante aceptación en la actualidad, son las (RNA) Redes Neuronales Artificiales, debido a los excelentes resultados obtenidos en el estudio de este tipo de situaciones, por su habilidad de aprender a partir de la experiencia; de generalizar nuevos casos basados en los anteriores; de abstraer características esenciales a partir de conjuntos de datos entradas-salida; etc. Hace que éstas ofrezcan numerosas ventajas para la investigación de este tipo de sistemas.

Por su utilidad las RNA se han aplicado en diferentes campos del sector productivo y de servicios, en la actualidad se tiene conocimiento de la utilización de este tipo de modelos con gran éxito en el campo militar, la biología, el medio ambiente, las finanzas, la manufacturación, la medicina y empresas de diferente índole

Los resultados obtenidos hasta el momento, permiten comprobar que las RNA realmente poseen la habilidad para representar sistemas no lineales, donde los métodos clásicos no conducen a resultados satisfactorios. La aplicación de estas técnicas de control en Sistemas Dinámicos requieren de una representación que caracterice el proceso físico, ósea, un modelo matemático del mismo. Esta representación matemática puede obtenerse en forma estrictamente teórica, lo que se denomina *Modelado*; así como también en forma empírica, mediante experimentos sobre el sistema real, basándose en datos de entrada y salida del proceso, llamado *Identificación*.

Debido a su auge, la comunidad científica a dirigido su atención hacia las RNA, ya que es una tecnología computacional emergente con gran potencial de investigación, que puede expandir sus aplicaciones a diversos sectores, su utilización conlleva a optimizar los procesos en la realización de tareas concretas, alcanzando mejores resultados que las tecnologías convencionales, debido a que éstas ofrecen apoyo basado en el conocimiento, lo que permite conseguir respuestas efectivas en periodos de tiempos razonables al momento de tomar decisiones.

Esta tesis dirige su enfoque investigativo principalmente a mejorar la resolución de los problemas de identificación y estimar los estados de Sistemas Dinámicos no-lineales y el control adaptativo de los mismos. La tarea es de suma importancia, al integrar esta clase de modelos en el estudio de los Sistemas Dinámicos no lineales y particularmente en el análisis del funcionamiento de los cilindros al interior de un motor. En la actualidad se conoce una amplia gama de aplicaciones de las RNA, pero se tiene pocos referentes sobre su utilización en este tipo de

diagnóstico, aun cuando se entiende que los motores son la parte más costosa del vehículo.

El objetivo principal de esta investigación, es determinar si las RNA funcionan correctamente en este caso especifico y si los resultados que se obtengan, satisfacen nuestras expectativas; pero esto solo lo podemos determinar con la elaboración de una herramienta que permita analizar el funcionamiento del motor a partir de los datos de la presión en los cilindros, y establecer el grado de confiabilidad que este ofrece, o por el contrario ayude a detectar las fallas, aumentado la precisión del dictamen de los expertos sobre la inmediatez de las reparaciones.

CAPITULO 2

2 MARCO TEÓRICO

2.1 IDENTIFICACIÓN DE SISTEMAS DINÁMICOS

La identificación de sistemas es la teoría y el arte de construir modelos matemáticos de Sistemas Dinámicos basándose en las entradas y salidas observadas. Estos modelos necesitan simular el comportamiento real del sistema. Diversos métodos y algoritmos para la identificación de sistemas se han estudiado desde 1960 y muchos procedimientos han sido propuestos y utilizados, sin embargo, su aplicabilidad para la identificación de sistemas no lineales es limitada.

El término *Identificación de sistemas* fue definido por Lofti Zadeh en 1962, como: la determinación, en base a las entradas y la salida, de un sistema, dentro de una clase específica, de sistemas, al cual el sistema probado es equivalente.

La identificación de sistemas dinámicos no lineales se considera un problema complejo. La razón es que identificar un sistema no lineal conlleva dos grandes etapas: la selección de la estructura del modelo con un cierto número de parámetros y la selección de un algoritmo que estime dichos parámetros. Por tanto, el obtener un buen modelo, con una estructura que refleje la información real del sistema es exigente.

Los modelos pueden ser paramétricos, que tienen la ventaja de estar dados por un conjunto pequeño de coeficientes, o bien no paramétricos como Las RNA, que tienen la ventaja de no estar restringidas a un número, posiblemente pequeño, de descripciones posibles del modelo.

Se considera que la identificación de sistemas quedó establecida como un campo de investigación reconocido dentro del área de control automático a mediados de los sesenta, en el tercer congreso de la IFAC en Londres, 1966 en el que fue presentado un artículo de visión general sobre identificación de sistemas[1].

2.1.1 MODELOS NO PARAMÉTRICOS RNA

Los modelos de RNA han sido estudiados durante muchos años con el ánimo de conseguir desempeños parecidos a los humanos. Estos modelos están compuestos por elementos computacionales, trabajando en paralelo y organizados en patrones que recuerdan a las redes neuronales biológicas.

[1] Cita tomada de la tesis, Identificación, Estimación y Control de Sistemas No-lineales mediante RGO, Garrido Santiago, UNIVERSIDAD CARLOS III DE MADRID (1999).

Los elementos computacionales o nodos están conectados mediante pesos que son adaptados a lo largo del proceso para mejorar su rendimiento. Es imposible cubrir todos los tipos de RNA y, solamente, se considerarán las llamadas Feedforward y Recurrentes, que son las más utilizadas en identificación de sistemas.

2.1.1.1 REDES CON CONEXIONES HACIA ADELANTE (FEEDFORWARD)

Las funciones de la base, llamadas nodos o neuronas, son funciones univaluadas que convierten la RNA en un desarrollo en funciones simples. La elección específica de la función de activación, suele ser escogida igual para todos los nodos. El nombre *feed forward* está explicado por la figura; hay una dirección específica en el flujo de los cálculos cuando se calcula la salida Y_n.

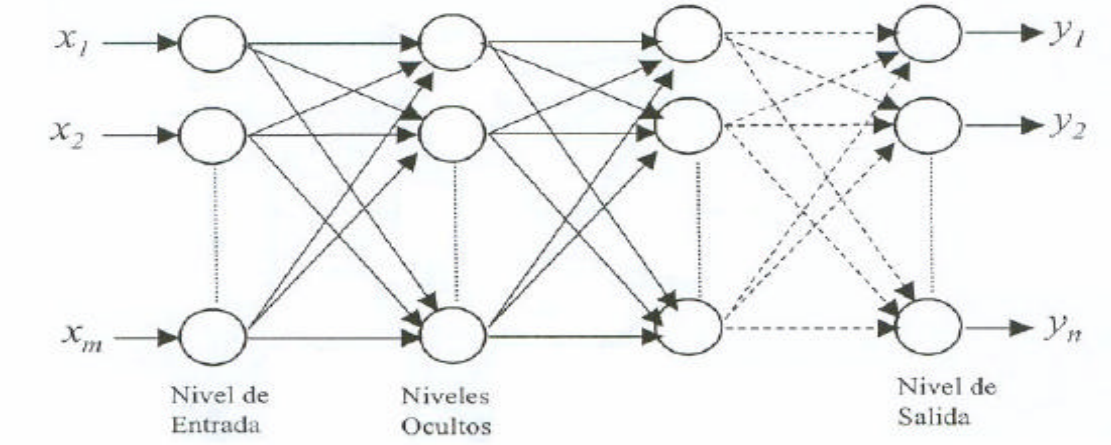


Figura 1. Red feedforward con X_m entradas, dos capas ocultas y Y_n salida[2]

En este tipo de redes, primero se calculan las sumas con peso en la entrada de cada unidad, después estas sumas son pasadas por la función de activación y forman las salidas de las capas ocultas. Para calcular Y_n se forma una suma con pesos de las salidas de los nodos de la capa oculta. Si Y_n es una función vectorial, hay varios nodos de salida formando la capa de salida, la entrada se denomina capa de entrada, los pesos de las diferentes sumas son los parámetros de la RNA y La función de activación tipo *sigmoide* es una función continúa, en la Figura 1, se muestra un esquema general de una red feedforward[3].
Las redes feedforward más conocidas son: Perceptrón, Adaline, Madaline, Linear Adaptive Memory (LAM), Drive-Reinforcement, Backpropagation. Todas ellas son especialmente útiles en aplicaciones de reconocimiento o clasificación de patrones.

[2] Figura 1. tomada de la tesis Estabilidad de entrada-estado (ISS) para identificación con redes neuronales dinámicas, Cruz Sandoval Alejandro, INSTITUTO POLITÉCNICO NACIONAL (2003)
[3] Definición tomada de la tesis, Identificación, Estimación y Control de Sistemas No-lineales mediante RGO, Garrido Santiago, UNIVERSIDAD CARLOS III DE MADRID (1999)

2.1.1.2 REDES RECURRENTES

Si alguna de las entradas de la red *feedforward* consiste en salidas retardadas de la RNA, o bien en algún estado interno retardado, la estructura recibe el nombre de red recurrente o red dinámica Figura.2. Éste tipo de RNA es utilizada en la identificación de sistemas no lineales, con resultados satisfactorios.

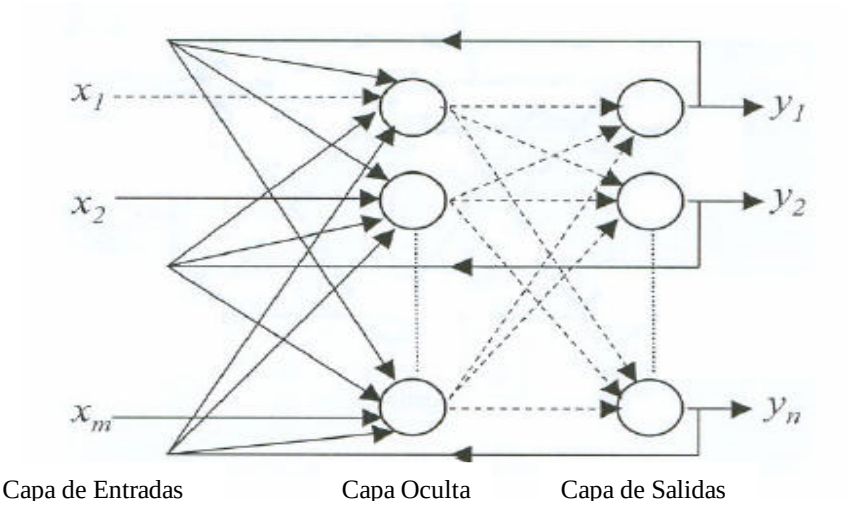


Figura 2. Red Recurrente, X_m entradas, una capa oculta y Y_n salida[4]

Las redes neuronales recurrentes[5] (RNR), son aquellas donde la información circula tanto hacia adelante como hacia atrás durante el funcionamiento de la red.

2.1.2 IDENTIFICACIÓN CON RNA

Los dos esquemas tradicionales de identificación para sistemas dinámicos, llamados serie-paralelo y paralelo, son los representados en las figura 3[6].

El esquema de identificación serie-paralelo utiliza las entradas y salidas del sistema y deja que internamente la RNA reduzca el error existente. Este esquema, también llamado *teacher forcing* puede ser usado con cualquier algoritmo de aprendizaje.

---

[4] Figura 2. tomada de la tesis Estabilidad de entrada-estado (ISS) para identificación con redes neuronales dinámicas, Cruz Sandoval Alejandro, INSTITUTO POLITÉCNICO NACIONAL (2003)
[5] Definición tomada de la tesis Estabilidad de entrada-estado (ISS) para identificación con redes neuronales dinámicas, Cruz Sandoval Alejandro, INSTITUTO POLITÉCNICO NACIONAL (2003)
[6] Figura 3. tomada de la tesis Identificación, Estimación y Control de Sistemas No-lineales mediante RGO, Garrido Santiago, UNIVERSIDAD CARLOS III DE MADRID (1999)

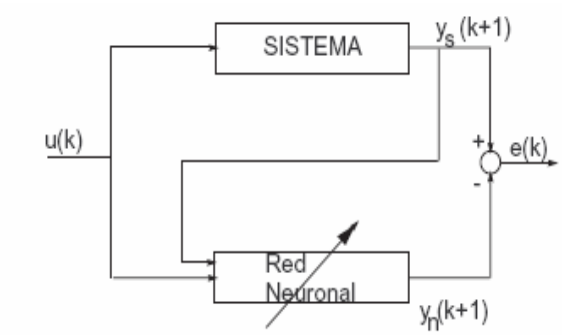


Figura 3. Identificación Serie-Paralelo

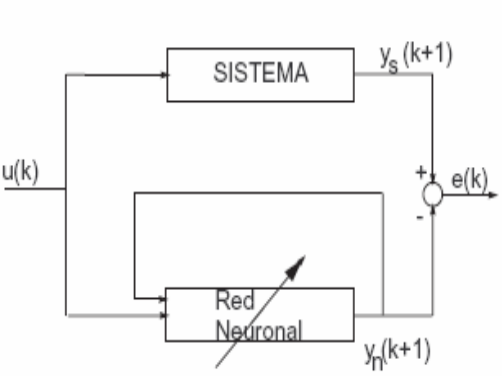


Figura 4. Identificación Paralela

La idea es calcular durante la fase de entrenamiento los nuevos estados de la RNA, utilizando las conexiones de realimentación en las salidas actuales del sistema, en lugar de las salidas de la RNA, su implementación puede ser sencilla con cualquier algoritmo.

Por otro lado, En el esquema de identificación paralela de la Figura 4[7], la RNA no tiene información directa acerca de las salidas del sistema, sólo las entradas y el error entre las salidas del sistema y la RNA.

2.1.3 IDENTIFICACIÓN DE SISTEMAS NO LINEALES POR MEDIO DE RNA

La identificación de un sistema usando RNA es un problema de correspondencia no lineal entre las entradas y salidas del sistema, y exhibe una gran cantidad de analogías con la teoría de identificación clásica. Un modelo "*serie-paralelo*" describe la dinámica del sistema basado en los datos de entrada y de salida del sistema, ver la Figura 5[8].

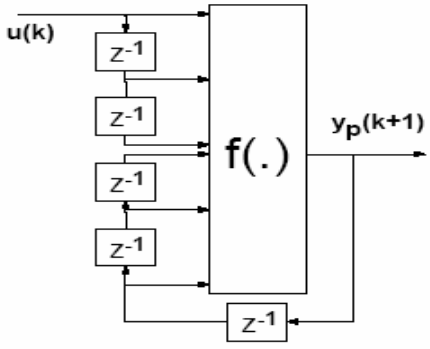


Figura 5. *Modelo Serie-Paralelo*

[7] Figura 4. tomada de la tesis Identificación, Estimación y Control de Sistemas No-lineales mediante RGO, Garrido Santiago, UNIVERSIDAD CARLOS III DE MADRID (1999)
[8] Figura 5. tomada de la tesis Identificación, Estimación y Control de Sistemas No-lineales mediante RGO, Garrido Santiago, UNIVERSIDAD CARLOS III DE MADRID (1999)

2.2. REDES NEURONALES ARTIFICIALES (RNA)

2.2.1 INTRODUCCIÓN

Las RNA son una simulación abstracta de un sistema nervioso real que esta formada por un conjunto de redes neuronales que están conectadas unas con otras. Este tipo de modelos fueron creado bajo los conceptos biológicos del cerebro y la finalidad era tratar ejercicios de control sobre comportamientos complejos de los cuales no obtenían respuestas satisfactorias utilizando métodos de la ciencia clásica. En la figura 6[9] se muestra la analogía entre las RNA y el sistema nervioso real.

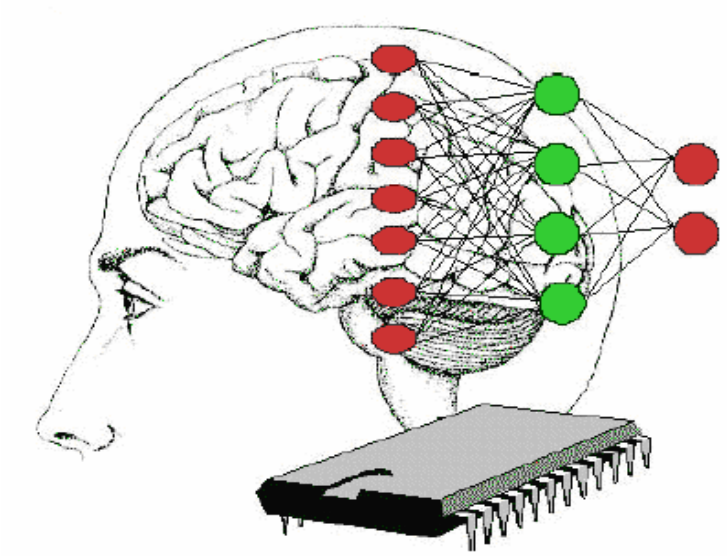


Figura 6. Analogía entre el sistema nervioso real y las RNA

2.2.2 DEFINICIÓN DE RNA

Son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativo) y con una organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nerviosos biológico.

2.2.3 RESEÑA HISTÓRICA DE LAS RNA

En el siguiente cuadro se resumen los principales acontecimientos de la historia de las RNA, ver tabla 1[10]

[9] Figura 6. tomada del texto sistemas conexionistas, González Penedo, Manuel F.
[10] Tabla 1, realiza por los autores

año	Descripción del suceso
1936	Alan Turing, fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación
1943	Warren McCulloch, neurofisiólogo y Walter Pitts, matemático; fueron los primeros teóricos que concibieron los fundamentos de la computación neuronal, lanzaron una teoría acerca de la forma de trabajar de las neuronas, modelando una red neuronal simple mediante circuitos eléctricos.
1949	Se lanza el libro *La organización del comportamiento*, de Donald Hebb, donde se tiene una conexión entre psicología y fisiología.
1957	Se inicia el desarrollo del Perceptrón, por Frank Rosenblatt. El Perceptrón es la red neuronal más antigua, y es muy utilizado hasta hoy en día como reconocedor de patrones.
1959	El modelo ADALINE (ADAptative LINear Elements) es desarrollado por Bernard Widrow y Marcial Hoff. Fue la primera red neuronal aplicada a un problema real, que eran filtros adaptativos para eliminar ecos en las líneas telefónicas, usado durante varias décadas comercialmente.
1967	Stephen Grossberg, de la Universidad de Boston, uno de los investigadores más activos de las redes neuronales desde los años 60 hasta la actualidad realizó una red llamada Avalancha, utilizada para reconocimiento continuo del habla y aprendizaje del movimiento de los brazos de un robot. Además, Grossberg ha escrito muchos libros y desarrollado otros modelos neuronales.
1969	Numerosas críticas frenaron el crecimiento de las investigaciones sobre redes neuronales hasta 1982 debido a un libro publicado por Marvin Minsky y Seymour Papert, del Instituto Tecnológico de Massachusetts (MIT) llamado *Perceptrons*, el cual contenía un análisis matemático del Perceptrón en forma detallada, y en el que consideraban que la extensión a Perceptrones multinivel era completamente esteril.
1982	Resurge el interés por las redes neuronales a consecuencia de varios eventos. John Hopfield presenta la red que lleva su nombre, que es una variación del Asociador lineal. Se celebró la *U.S. – Japan on Cooperative/Competitive NEURAL Networks*.
1985	Instituto Americano de Física comenzó la reunión anual *NEURAL Networks for Computing*
1987	Se formó la *International NEURAL Network Society (INNS)*, con dirección de Grossberg en U.S.A., Kohonen en Finlandia y Amari en Japón.
1990	Se comenzó el fenómeno de crear aplicaciones de las redes neuronales artificiales empleándose en diferentes campos del sector productivo y de servicios.
2005	Se tiene conocimiento de innumerables aplicaciones que han permitido el avance de las investigaciones de la comunidad científica.

Tabla 1. Reseña histórica de las redes neuronales

2.2.4 VENTAJAS DE LAS REDES NEURONALES ARTIFICIALES

A continuación se enuncian algunas de las ventajas RNA:

- *Aprendizaje adaptativo*: Son capaces de aprender de la experiencia inicial o al realizar tareas basadas en un entrenamiento.
- *Generalización:* Los casos anteriores se utilizan en la evaluación de los nuevos casos
- *Autoorganización:* Una red neuronal puede crear su organización o representación de la información que recibe mediante una etapa de aprendizaje.
- *Tolerancia a fallos:* La destrucción parcial de una red conduce a una degradación de su estructura; sin embargo algunas de las capacidades de la red se pueden retener a pesar de los daños.
- *Abstracción:* Extrae las características esenciales a partir de las entradas que representa información irrelevante
- *Operación en tiempo real:* Los cómputos neuronales pueden ser realizados en paralelo, y se diseñan y fabrican hardware especial par obtener esta capacidad
- *Fácil inserción dentro la tecnología existente*: Se pueden obtener chips especializados para mejorar las capacidades de las redes neuronales en la realización de ciertas tareas, ellos facilita la inserción de módulos en los sistemas existentes.

2.2.5 ELEMENTOS DE LAS REDES NEURONALES ARTIFICIALES

Las RNA son modelos que intentan reproducir el comportamiento del cerebro. Este modelo realiza una simplificación, averiguando cuales son los elementos relevantes del sistema, bien por la cantidad de información de que dispone o bien porque esta es redundante, cualquier modelo de RNA consta de dispositivos elementales de proceso:

2.2.5.1 NEURONA

En las neuronas de las RNA pueden generar representaciones específicas de tal forma que un estado conjunto de ellas puede significar una letra numero o cualquier otro objeto, generalmente se pueden encontrar tres tipos de neuronas.

- Aquellas que reciben estímulos externos, relacionado con el aparato sensorial, que toma la información de entrada.

- Dicha información se tramite a ciertos elementos internos que se ocupan de su procesado. Es la sinapsis y las neuronas correspondientes a este segundo nivel donde se genera cualquier tipo de representación interna de

la información. Puesto que no tiene relación directa con la información de entrada, ni de salida, estas unidades se denominan unidades ocultas.

- Una vez finalizado el periodo de procesado, la información llega las unidades de salida, cuya misión es dar respuesta al sistema.

En la figura 7[11], se muestra los elementos de una neurona artificial utilizadas por las RNA en su proceso de aprendizaje.

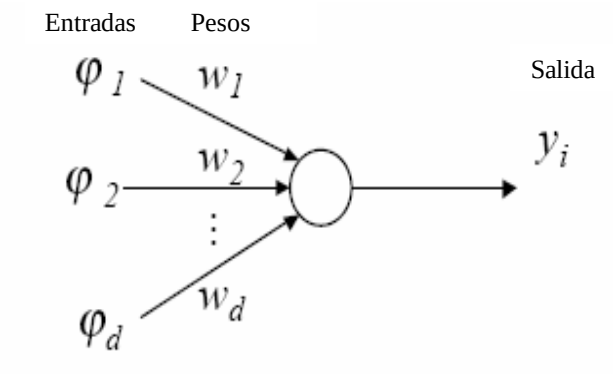


Figura 7. Representación de una neurona

Las neuronas artificiales pretenden mimetizar las características más importantes de las neuronas biológicas. Cada neurona i-ésima está caracterizada en cualquier instante por un valor numérico denominado valor o estado de activación asociado a cada unidad, existe una salida, que transforma el estado actual de activación en una señal de salida, dicha señal es enviada a través de los canales de comunicación unidireccionales a otras unidades de la red; en estos canales la señal se modifica de acuerdo con la sinapsis (peso), asociada a cada uno de ellos según una determinada regla. Las señales moduladas que han llegado a la unidad j-ésima se combina entre ellas, generando así una entrada total.

La función de activación, determina el nuevo estado de la activación de la neurona, teniendo en cuenta la entrada total calculada y el anterior estado de activación, la dinámica que rige la actualización de los estados de las unidades (evolución de red natural) puede ser de dos tipos.

Modo Asincrónico. Las neuronas evalúan sus estado continuamente, según les valla llegando información, y lo hacen de forma independiente.

Modo Sincrónico. La información también llega de forma continua pero los cambios se realizan simultáneamente, como si existiera un reloj interno que decidiera cuando debería cambiar de estado.

[11] Figura 7. Tomada del curso de Identificación de sistemas capitulo 4, Escobet Teresa y Morcego Bernardo

2.2.5.2 UNIDADES DE PROCESAMIENTO: NEURONA ARTIFICIAL

Si se tiene N unidades (neuronas), se ordenan, arbitrariamente, designando la j-ésima unidad. Su trabajo es simple y único, y consiste en recibir las entradas de las células vecinas y calcular un valor de salida, que es enviado a las demás células restantes. En cualquier sistema que se esté modelando, es útil caracterizar tres tipos de unidades, en la figura 8[12] se muestra las unidades de procesamiento.

Unidades de Entrada. Reciben señales desde el entorno; estas entradas que a la vez son entradas de la red, puede ser señales proveniente de sensores o de otros sectores del sistema.

Unidades Ocultas. Son aquellas cuyas entradas y salidas ser encuentran dentro del sistema; decir, no tiene contacto con el exterior.

Unidades de Salida. Envía la señal fuera del sistema (salida de la red); estas señales pueden controlar o potenciar a otros sistemas, también se conoce como capa o nivel un conjunto de neuronas cuyas entradas provienen de la misma fuente que puede ser de otra capa de neuronas, o cuya salida se dirige al mismo destino que puede ser otra capa de neuronas.

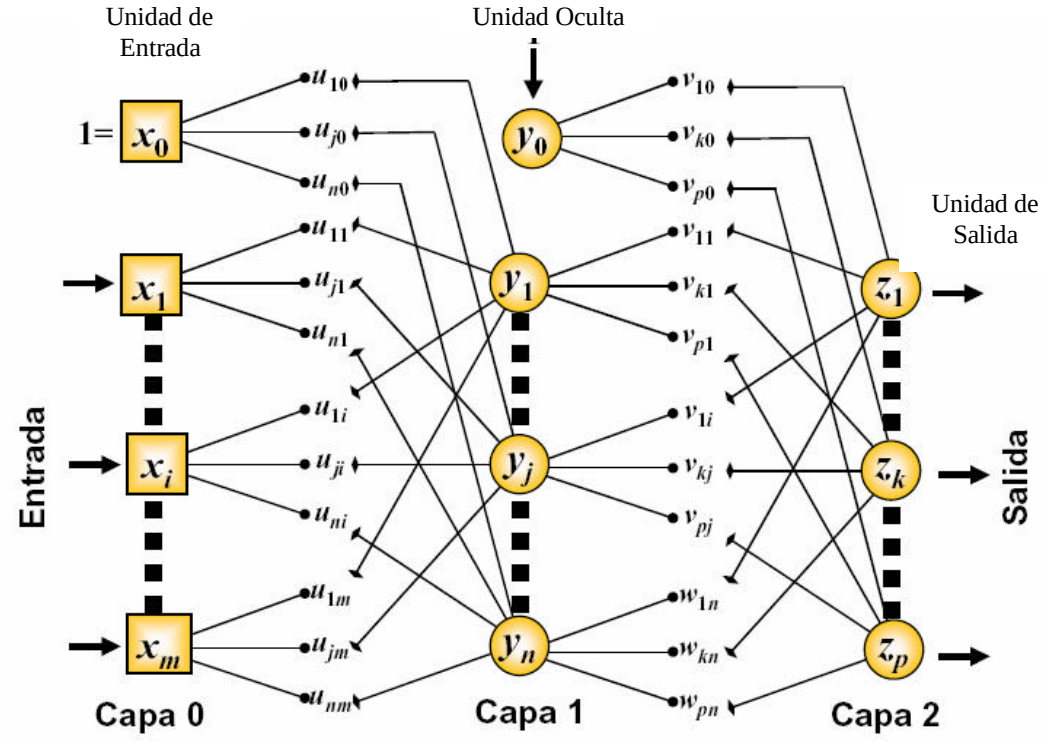


Figura 8. Unidades de procesamiento de una RNA

[12] Figura 8. Tomada de documento Redes Neuronales Artificiales. IZAURIETA, Fernando y SAAVEDRA, Carlos, Departamento de física, UNIVERSIDAD DE CONCEPCIÓN, Chile.

2.2.5.3 ESTADOS DE ACTIVACIÓN

Adicionalmente, al conjunto de unidades, la representación necesita los estados del sistema en un tiempo t. esto significa por un vector de N números reales, que representa el estado de activación del conjunto de unidades de procesamiento. Cada elemento del vector representa la activación de una unidad en un tiempo t.

Todas las neuronas que componen la RNA se hallan en cierto estado, aunque podemos decir que existe dos estados, reposo y excitado a los que llamaremos globalmente estado de activación, y a cada uno de ellos se le asigna un valor. Estos valores pueden ser continuos o discretos, además, pueden ser limitados o ilimitados, si son discretos suelen tomar un conjunto de pequeños valores o bien valores binarios. En notación binaria, un estado activo se indica por un 1, y se caracteriza por la emisión de un impulso por parte de la neurona, mientras que un estado pasivo se indica por un 0, y significa que la neurona esta en reposo. En otros modelos se considera un conjunto continuo de estados de activación, en lugar de uno solo, dos estados, en cuyos casos se le asigna un valor ente [0,1] o en el intervalo [-1,1], generalmente siendo una función sigmoidal, lineal, etc.

2.5.5.4 FUNCIÓN DE SALIDA O DE TRANSFERENCIA

Entre las unidades o neuronas que forman una red neuronal artificial, existen conjunto de conexiones que unen las unas con las otras. Cada neurona transmite señales a aquellas que están conectadas con su salida.

Existen cuatro funciones de transferencia típicas que determinan los distintos tipos de neuronas

- Función binaria
- Función lineal
- Función sigmoidal
- Función gaussiana

2.5.5.4.1 FUNCIÓN BINARIA O ESCALÓN

La forma más fácil de definir la activación de una neurona es considerar que es binaria. La función de transferencia escalón se asocia a neuronas binarias en las cuales, cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es 1; si es menor, la activación es 0 (o -1). Por otro lado, las RNA formadas por este tipo de neuronas son fáciles de implementar en hardware, pero sus capacidades son limitadas

2.5.5.4.2 FUNCIÓN LINEAL Y MIXTA

La función lineal o identidad responde a la expresión $f(x) ? x$. En las neuronas con función mixta, si la suma de las señales de entrada es menor que un límite inferior, la activación se define como cero. Si dicha suma es mayor o igual que el límite superior, entonces la activación es 1. Si la suma de la entrada está comprendida entre ambos limites, superior e inferior, entonces la activación se define como una función lineal de la suma de las señales de entrada.

2.5.5.4.3 NEURONAS DE FUNCIÓN CONTINUA

La función sigmoidal, para la mayoría de los valores del estímulo de entrada definidos como variables independientes, el valor dado por la función es cercano a uno de los valores asintóticos. Razón para que en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja de la función sigmoidal. De hecho, cuando la pendiente es elevada, esta función tiende a la función escalón.

Sin embargo, la importancia de la función sigmoidal es que su derivada es positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando x es 0. Esto permite usar reglas de aprendizaje definidas para la función escalón, con la ventaja, respecto a esta función, que la derivada está definida en todo el intervalo. La función escalón no podía definir la derivada en el punto de transición y esto no ayuda a los métodos de aprendizaje en los cuales se usan derivadas

2.5.5.4.4 FUNCIÓN DE TRANSFERENCIA GAUSSIANA

Los centros y el ancho de estas funciones pueden ser adaptados, hecho que las hace más adaptativas que la función sigmoidal. La función gaussiana suelen requerir dos niveles ocultos, donde utiliza neuronas con funciones de transferencia sigmoidal; aunque algunas veces se pueden realizar con un solo nivel en redes con neuronas de función gaussiana

En al Figura 9[13], se muestran las funciones de activación o transferencia anteriormente enunciadas.

[13] Figura 9. Tomada del curso de Identificación de sistemas capitulo 4, Escobet Teresa y Morcego Bernardo

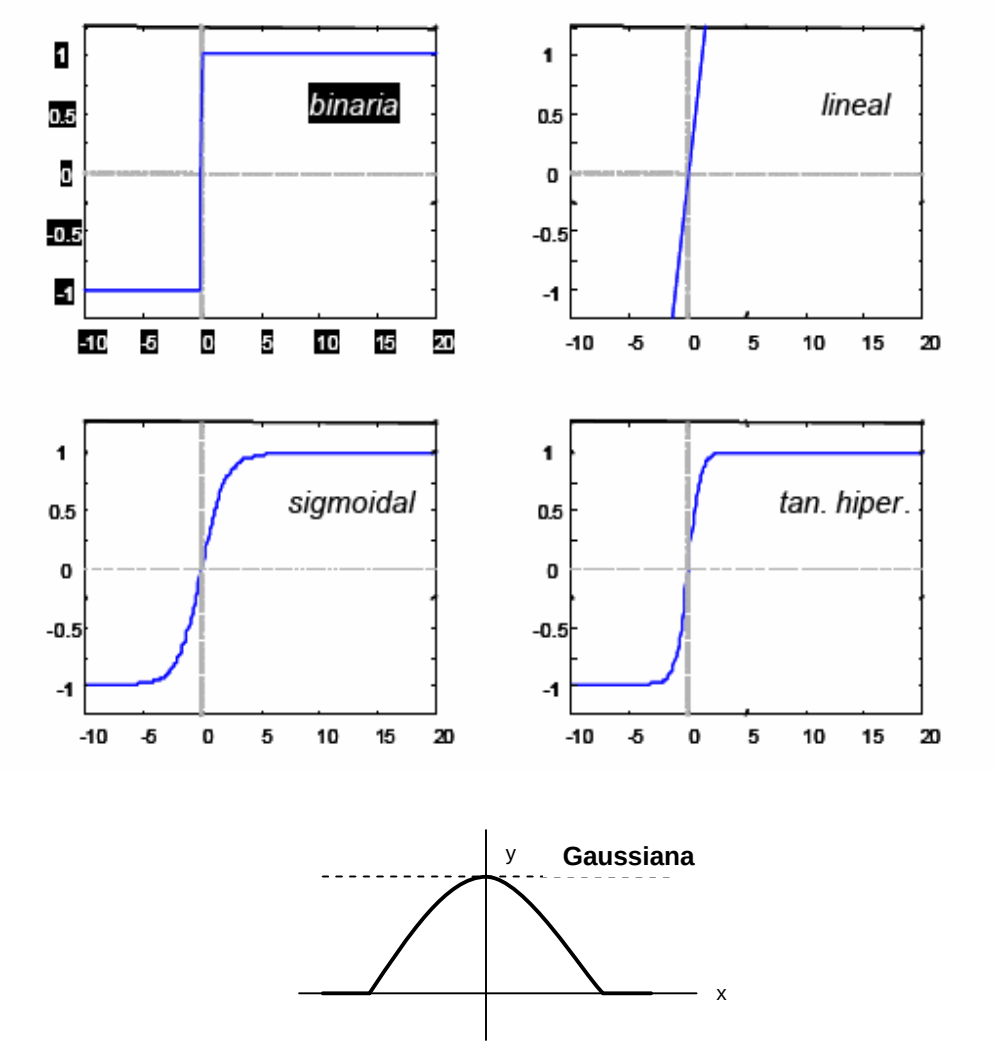


Figura 9. Función activación: binaria F(x)= Sgn(x), lineal F(x)=x, Sigmoidal F(x)= $(1+e^{-x})^{-1}$, Tangencial Hiperbólica F(x)= $(1+e^{-x})^{-1} * (1+e^{-x})$ y Gaussiana

2.2.5.5 CONEXIONES ENTRE NEURONAS

Las conexiones que unen a las neuronas que forman las RNA tienen asociado un peso, que es el que hace que la red adquiera conocimiento. Una neurona recibe un conjunto de señales que le dan información del estado de activación de todas las neuronas con las que se encuentra conectada. Cada conexión (sinapsis) entre la neurona i y la neurona j esta ponderada por un peso. Normalmente, como simplificación, se considera que el efecto de cada señal es aditivo, de tal forma que la entrada neta que recibe la neurona (potencial postsináptico) es la suma del producto de cada señal individual por el valor sinapsis que conecta ambas neuronas

2.2.5.6 FUNCIÓN O REGLA DE ACTIVACIÓN

Así como es necesaria una regla que combine las entradas a una neurona con los pesos de las conexiones, también se requiere una regla que combine las entradas con el estado actual de la neurona para producir un nuevo estado de activación.

2.2.5.7 REGLA DE APRENDIZAJE

Una definición para la regla de aprendizaje seria la "modificación del comportamiento inducido por la interacción con el entorno y como resultado de la experiencia conducente al establecimientos de nuevos modelos de respuesta a estímulos externos". En las redes neuronales artificiales se pude decir que el conocimiento se encuentra representado en los pesos de las conexiones entre neuronas. Todo proceso de aprendizaje implica cierto número de cambios en estas conexiones. En realidad se puede decir que se aprende modificando los valores de los pesos de la red.

2.2.5.7 ESTRUCTURA DE UNA RNA.

Las RNA están compuestas por niveles o capas de neuronales, estos niveles pueden ser de tres tipos:

- Entrada: Es la capa que recibe la información proveniente de fuente externa a la red.
- Ocultas: Son internas en la red y no tiene contacto con el entorno exterior
- Salida: Transfiere la información de la red al exterior.

Según la forma de conexión La RNA puede tener propagación hacia delante cuando ninguna salida de la neurona es la entrada de neuronas del mismo nivel o niveles precedentes y propagación hacia atrás, cuando las salidas pueden ser conectadas como entradas de las neuronas de niveles previos o del mismo nivel, excluyéndose ellas mismas.

2.2.6 MECANISMO DE APRENDIZAJE DE LAS RNA

El entrenamiento de una red se lleva a cabo mediante una regla o algoritmo de aprendizaje. Este algoritmo es un procedimiento para modificar los pesos de una red y su propósito es entrenar la red para ejecutar alguna tarea[14].
Según el algoritmo de aprendizaje, las RNA se dividen en redes de entrenamiento supervisado y no supervisado

[14] Hagan, Dermuth y Beale: Neural network design. Boston, USA: PWS Publishing Company, 1996.

2.2.6.1 RNA SUPERVISADAS

Las RNA de entrenamiento **supervisado** constituyen la línea fundamental de desarrollo en este campo. Algunos ejemplos bien conocidos de las primeras redes es el PERCEPTRÓN, adaline /madaline y varias redes multicapa. En el entrenamiento supervisado hay dos fases a realizar: fase de prueba y fase de entrenamiento. En el entrenamiento supervisado, los patrones de entrenamiento se dan en forma de pares de entrada/muestro. Dependiendo de la naturaleza de la información del muestro, hay dos aproximaciones al entrenamiento supervisado. Uno se basa en la corrección a partir de una decisión y la otra se basa en la optimización de un criterio de validación. De la última, la aproximación del error cuadrático medio es el más importante.

2.2.6.2 RNA NO SUPERVISADAS

Para los modelos de entrenamiento **No Supervisado**, el conjunto de datos de entrenamiento consiste sólo en los patrones de entrada. Por lo tanto, la red es entrenada sin el beneficio de un muestro. La red aprende a adaptarse basada en las experiencias recogidas de los patrones de entrenamiento anteriores. Este es un esquema típico de un sistema "**No Supervisado**".
Una clase de modelos de entrenamiento no supervisado son las redes de pesos fijos un ejemplo son las redes de Memoria Asociativa, que se usan para obtener patrones originales libres de ruido a partir de señales incompletas o distorsionadas. La principal característica de las redes asociativas de pesos fijos es que sus pesos son preestablecidos y precalculados.

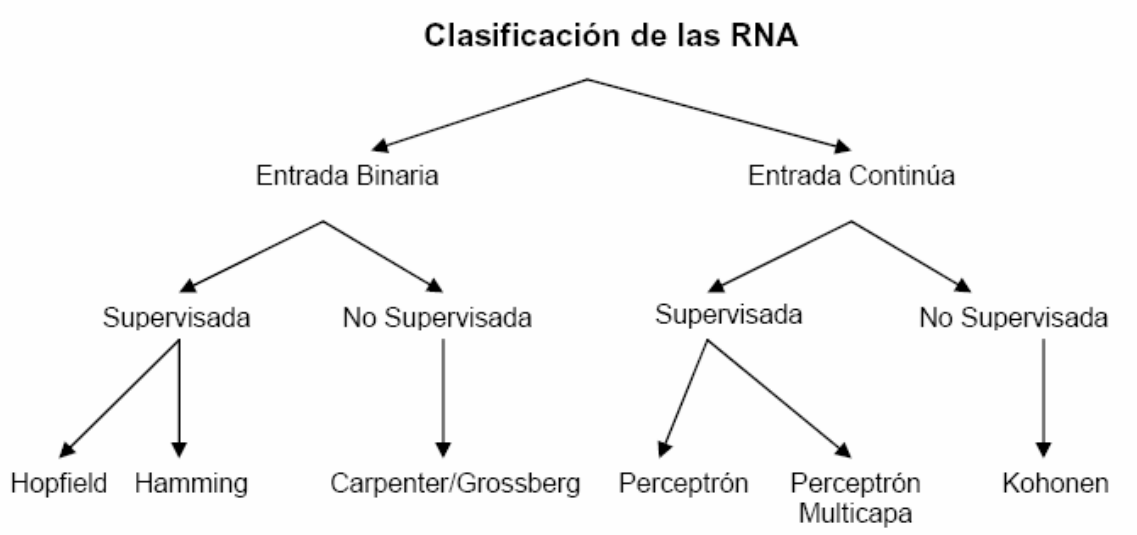


Figura 10. Clasificación de lasa redes neuronales

Los modelos de pesos fijos tienen aplicaciones limitadas ya que no se pueden adaptar a "ambientes cambiantes". Hay otra variedad de redes no supervisadas, llamadas redes de aprendizaje competitivo, cuyos pesos se adaptan de acuerdo

con reglas de aprendizaje no supervisadas. Estas redes pueden aprender en ausencia de un muestro. En otras palabras, el entrenamiento se basa únicamente en la información de los patrones de entrada. La clase de redes de aprendizaje competitivo se componen, por ejemplo, de la red de autoorganización.
Para profundizar los conceptos de esta metodología en la figura 10[15] se muestra la clasificación de las redes neuronales artificiales.

2.2.7 TIPOS DE REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales ofrecen la posibilidad de diseñar su arquitectura de acuerdo a los requerimientos de la aplicación, consiguiendo una adaptación a los ejercicios que difícilmente se pueden hallar en otras metodologías.

2.2.7.1 PRIMEROS MODELOS COMPUTACIONALES

2.2.7.1.1 CÉLULAS DE MCCULLOCH-PITTS

El primer ejemplo que puede considerarse como una RNA, al menos estructuralmente, son las células de McCulloch-Pitts. Este primer modelo de neurona fue propuesto por Warren McCulloch y Walter Pitts en 1943[16]. En él se modelaba una estructura y un funcionamiento simplificado de las neuronas del cerebro, considerándolas como dispositivos con sólo dos estados posibles: apagado (0) y encendido (1) (ver Figura 11[17]).

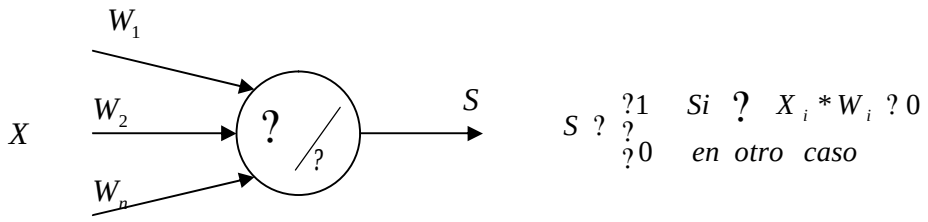


Figura 11. Esquema de una célula McCulloch-Pitts

La célula de McCulloch-Pitts recibe como entrada un conjunto de n valores binarios, $X = (x_1, x_2, ..., x_n)$ procedente de las salidas de otras células, o de la

[15] Figura 10. Tomada de la tesis Determinación del Patrón de Flujo Multifásico en Tuberías De Recolección de Petróleo Emulsionado, A partir de los Datos Históricos de Producción Apoyado en una Aplicación con Redes Neuronales Artificiales, Florez Sánchez, Jorge Eduardo, Porras Mejia, Freddy Alberto, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)
[16] W. S. McCulloch y W. A. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Boulletin of Mathematics and Biophysics*, 5, págs, 115-133, 1943.
[17] Figura 11. Tomada de la tesis Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Reyes Figueroa, Juan Carlos, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)

entrada a la red; y produce una única salida también binaria s. Cada célula se caracteriza por $n?1$ valores reales, de los cuales n son los pesos de las conexiones (w_i) correspondientes a las entradas x_i, y el otro es un valor de umbral $?$, que puede ser distinto para cada célula. La célula opera en lapsos discretos. La forma de procesar la entrada es la siguiente: la célula se activará y, por lo tanto, producirá un valor 1, si la suma de las entradas multiplicadas por los pesos supera al umbral $?$.

$$S(t?1)?\begin{cases}1 & ?_i w_i x_i(t)??\\ 0 & en\ otro\ caso\end{cases} \quad (1)$$

A partir del modelo de neurona de McCulloch-Pitts se define el primer modelo de RNA. Una RNA es una colección de neuronas de McCulloch-Pitts, todas con las mismas escalas de tiempos, donde sus salidas están conectadas a las entradas de otras neuronas.
De este modo, una salida puede actuar sobre varias entradas, pero una entrada viene a lo sumo de una salida. La RNA tiene contacto con el exterior a través de las líneas de entrada y de salida. Las líneas de entrada de la RNA formarán parte de la entrada de alguna o de todas las neuronas de la RNA. Asimismo, las líneas de salida procederán de alguna o de todas las neuronas de la RNA.

Esta formalización matemática de la RNA no es y no pretendió ser un modelo del cerebro, pero si un punto de partida para iniciar los estudios sobre el mismo.

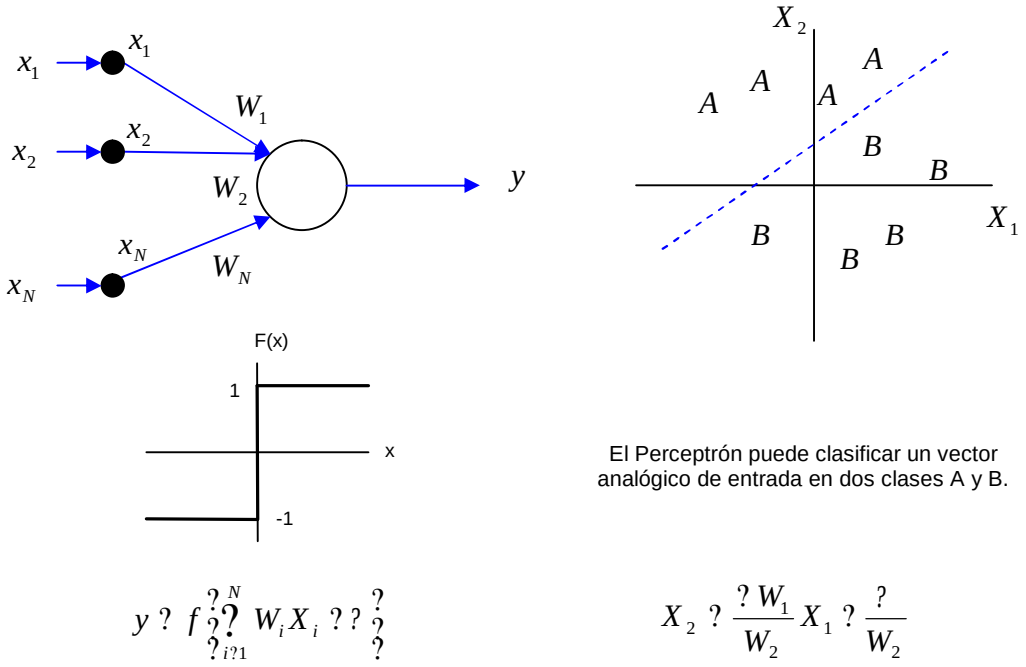
2.2.7.1.2 EL PERCEPTRÓN

El dispositivo conocido con el nombre de **Perceptrón** fue inventado por el Psicólogo Frank Rosenblatt a finales de los años cincuenta. En un intento de ilustrar algunas de las propiedades fundamentales de los sistemas inteligentes en general, sin entrar en detalles, en ciertas condiciones especiales, y muchas veces desconocidas, que son válidas para organismos biológicos concretos, Rosenblatt creía que la conectividad que se desarrolla en las redes biológicas tiene un elevado porcentaje de aleatoriedad. Por tanto, desaprobaba los análisis anteriores tales como el modelo de McCulloch-Pitts, en los cuales se empleaba lógica simbólica para analizar unas estructuras bastante idealizadas.
La única neurona de salida del Perceptrón realiza la suma ponderada de las entradas, resta el umbral y pasa el resultado a una función de transferencia de tipo escalón. La regla de decisión es responder $?1$ si el patrón presentado pertenece a la clase A, o $?1$ si el patrón pertenece a la clase B (ver Figura 12[18]). La salida

[18] Figura 12. Tomada de la tesis Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Reyes Figueroa, Juan Carlos, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)

dependerá de la entrada neta (suma de las entradas x_i ponderadas) y del valor del umbral ? .

Sin embargo, el perceptrón, al constar sólo de una capa de entrada y otra de salida con una única neurona, tiene una capacidad de representación bastante limitada. Este modelo sólo es capaz de discriminar patrones muy sencillos, linealmente separables. El caso más conocido es la imposibilidad del Perceptrón de representar la función Or-Exclusiva.



$$y ? f\left(\sum_{i=1}^{N} W_i X_i ? ?\right)$$

$$X_2 ? \frac{? W_1}{W_2} X_1 ? \frac{?}{W_2}$$

Figura 12. El Perceptrón

La separabilidad lineal limita a las RNA con sólo dos capas a la resolución de problemas en los cuales el conjunto de puntos (correspondientes a los valores de entrada) sean separables geométricamente. En el caso de dos entradas, la separación se lleva a cabo mediante una línea recta. Para tres entradas, la separación se realiza mediante un plano en el espacio tridimensional, y así sucesivamente hasta el caso de N entradas, en el cual el espacio N-dimensional es dividido en un hiperplano.

2.2.7.1.2.1 REGLA DE APRENDIZAJE DEL PERCEPTRÓN

El algoritmo de aprendizaje del Perceptrón es de tipo supervisado, lo cual requiere que sus resultados sean evaluados y se realicen las oportunas modificaciones del sistema si fuera necesario. Desgraciadamente, sólo se pueden aprender clasificaciones fáciles (problemas de orden 1 en la terminología de Minsky y Papert[19]).

Para dar más claridad sobre el funcionamiento de la RNA tipo *Perceptrón*, es importante comprender el algoritmo de convergencia para ajustar los pesos, el cual realiza el aprendizaje para esta RNA, específicamente, el aprendizaje por corrección de error) con N elementos procesales de entrada y un único elemento procesal de salida. A continuación se describe el algoritmo de aprendizaje que consta de:

1. Inicialización de los pesos y del umbral

Inicialmente se asignan valores aleatorios a cada uno de los pesos (w_i) de las conexiones y al umbral $? w_0 ? ?$.

2. Presentación de un nuevo par (Entrada, Salida esperada)

Presentar un nuevo patrón de entrada $X_P ? (x_1, x_2, ..., x_N)$ junto con la salida esperada $d(t)$.

3. Cálculo de la salida actual

$$y(t) ? f\left(? \sum_i w_i(t)x_i(t) ? ?\right) \quad (2)$$

Siendo $f(x)$ la función de transferencia escalón.

4. Adaptación de los pesos

$$w_i(t ? 1) ? w_i(t) ? ? [d(t) ? y(t)]x_i(t) \qquad (0 ? i ? N) \quad (3)$$

Donde $d(t)$ representa la salida deseada, y será 1 si el patrón pertenece a la clase A, y -1 si es de la clase B. En estas ecuaciones, $?$ es un factor de ganancia en el rango de 0.0 a 1.0. Este factor debe ser ajustado de forma que satisfaga tanto los

[19] M Minsky y S. Papert, “Perceptrons”; Ed. MIT Press, 1969.

requerimientos de aprendizaje rápido como la estabilidad de las estimaciones de los pesos. Este proceso se repite hasta que el error que se produce para cada uno de los patrones (diferencia entre el valor de salida deseado y obtenido) es cero o bien menor que un valor preestablecido. Hay que observar que los pesos no se cambian si la red ha tomado la decisión correcta.

5. Volver al paso 2

Este algoritmo es extensible al caso de múltiples neuronas en la capa de salida. El perceptrón será capaz de aprender a clasificar todas sus entradas, en un número finito de pasos, siempre y cuando el conjunto de los patrones de entrada sea linealmente separable.

2.2.7.2 EL PERCEPTRÓN MULTICAPA

El perceptrón multinivel o multicapa es una RNA de tipo feedforward compuesta de varias capas de neuronas entre la entrada y la salida de la misma (ver Figura 13[20]). Esta RNA permite establecer regiones de decisión mucho más complejas que las de dos semiplanos, como hacía el Perceptrón de un solo nivel.

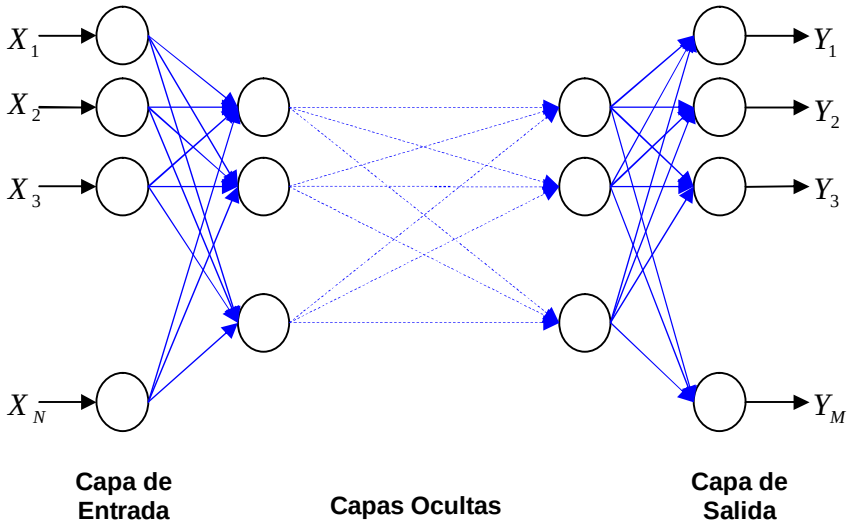


Figura 13. Perceptrón Multinivel

[20] Figura 13. Tomada de la tesis Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Reyes Figueroa, Juan Carlos, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)

Las capacidades del Perceptrón con dos, tres, y cuatro niveles o capas y con una única neurona en el nivel de salida, se muestran en la figura 14[21], en la segunda columna se exhibe el tipo de región de decisión que se puede formar con cada una de las configuraciones. En la siguiente columna se indica el tipo de región de decisión que se formaría para el problema de XOR. En las dos últimas columnas se muestran las regiones formadas para resolver el problema de clases con regiones mezcladas y las formas de regiones más generales para cada uno de los casos.

El Perceptrón básico de dos capas (la de entrada con neuronas lineales y la de salida con función de activación de tipo escalón) sólo puede establecer dos regiones separadas por una frontera lineal en el espacio de patrones de entrada. Un perceptrón con tres o más niveles de neuronas puede formar cualquier región convexa en este espacio.

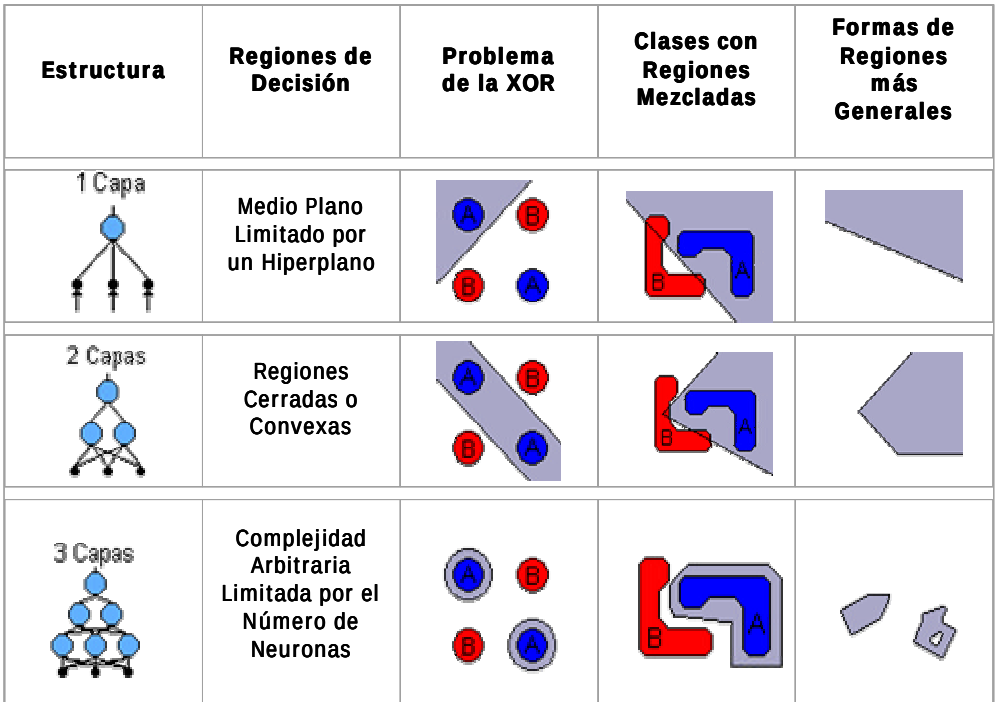


Figura 14. Formas de regiones generadas por un *Perceptrón* multinivel

[21] Figura 14. Tomada de la tesis Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Reyes Figueroa, Juan Carlos, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)

2.2.7.2.1 ARQUITECTURA DEL PERCEPTRÓN MULTICAPA

La arquitectura del Perceptrón multicapa se caracteriza por que tiene sus neuronas agrupadas en capas de diferentes niveles. Cada una de las capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas diferentes: la capa de entrada, las capas ocultas y la capa de salida.
Las neuronas de la capa de entrada no actúan como neuronas propiamente dichas, sino que se encargan de recibir las señales o patrones que proceden del exterior y propagar dichas señales a todas las neuronas de la siguiente capa. La última capa actúa como salida de la red, proporcionando al exterior la respuesta de la red para cada uno de los patrones de entrada. Las neuronas de las capas ocultas realizan un procesamiento no lineal de los patrones recibidos.
Las conexiones del Perceptrón multicapa siempre están dirigidas hacia delante, es decir, las neuronas de una capa se conectan con las neuronas de la siguiente capa, de ahí que reciban el nombre de redes alimentadas hacia delante o redes "feedforward". Las conexiones entre las neuronas llevan asociado un número real, llamado peso de la conexión. Todas las neuronas de la red llevan también asociado un umbral, que en el caso del Perceptrón multicapa suele tratarse como una conexión más a la neurona, cuya entrada es constante e igual a 1.
Generalmente, todas las neuronas de una capa están conectadas a todas las neuronas de la siguiente capa. De este modo, las neuronas de la capa de entrada están conectadas a todas las neuronas de la primera capa oculta; las neuronas de la primera capa oculta se conectan a las neuronas de la siguiente capa, etc. Se dice entonces que existe conectividad total o que la red está totalmente conectada.

2.2.7.2.2 SELECCIÓN DE LA ARQUITECTURA DEL PERCEPTRÓN MULTICAPA

Cuando se aborda un problema utilizando el ***Perceptrón*** multicapa, uno de los primeros pasos a realizar es el diseño de la arquitectura de la RNA. Este diseño implica la determinación de la función de activación a emplear, el número de neuronas y el número de capas de la RNA.

La elección de la función de activación se suele hacer basándose en el recorrido deseado, y el hecho de elegir una u otra, no influye en la capacidad de la red para resolver el problema. En lo que respecta al número de neuronas y capas, algunos de estos parámetros vienen dados por el problema y otros deben ser elegidos por el diseñador. Así, por ejemplo, tanto el número de neuronas en la capa de entrada, como el número de neuronas de la capa de salida, vienen dados por las variables que definen el problema.

El número de capas ocultas y el número de neuronas en estas capas deben ser elegidos por el diseñador. No existe un método o regla que determine el número óptimo de neuronas ocultas para resolver un problema dado.

2.2.7.2.3 ALGORITMO BACKPROPAGATION

La regla de aprendizaje del Perceptrón de Rosenblatt y el algoritmo LMS de Widrow y Hoff fueron diseñados para entrenar redes de una sola capa. Estas redes tienen la desventaja que solo pueden resolver problemas linealmente separables, fue esto lo que llevó al surgimiento de las RNA multicapa para sobrepasar esta dificultad de las RNA hasta entonces conocidas.

En 1986, Rumelhart, Hinton y Willians [Rumelhart 86], basándose en los trabajos de investigadores como: [Werbos 74] y [Parker 82], formalizaron un método para que una RNA aprendiera la asociación que existe entre los patrones de entrada a la misma y las clases correspondientes, utilizando más niveles de neuronas que los que utilizó Rosenblatt para desarrollar el Perceptrón. Este método, conocido como Backpropagation (propagación del error hacia atrás), está basado en la generalización de la regla delta y, a pesar de sus propias limitaciones, ha ampliado de forma considerable el rango de aplicaciones de las RNA.

2.2.7.2.4 LA REGLA DELTA GENERALIZADA

La regla propuesta por Widrow en 1960 (regla delta) ha sido extendida a redes con capas intermedias (regla delta generalizada) con conexiones hacia delante (feedforward) y cuyas células tienen funciones de activación continuas (lineales o sigmoidales), dando lugar al algoritmo de retropropagación (Backpropagation). Estas funciones continuas son no decrecientes y derivables. La función sigmoidal pertenece a este tipo de funciones, a diferencia de la función escalón que se utiliza en el perceptrón, porque esta última no es derivable en el punto en el que se encuentra la discontinuidad

2.2.7.2.5 FUNCIONAMIENTO DEL ALGORITMO

El método que sigue la regla delta generalizada para ajustar los pesos es el mismo que el de la regla delta utilizada en la red Perceptrón y en la red Adaline, es decir, los pesos se actualizan de forma proporcional a la delta, o diferencia entre la salida deseada y la obtenida ($?$ = sal. Deseada - sal. Obtenida).

Dada una neurona (unidad U_i) y la salida que produce y_i (ver Figura 15[22]), el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad $U_j\,(w_{ji})$ para un patrón de aprendizaje p determinado es:

[22] Figura 15. Tomada de la tesis Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Reyes Figueroa, Juan Carlos, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)

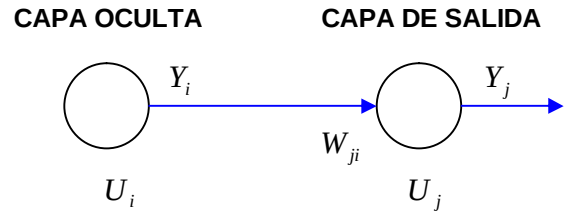


Figura 15. Conexión Entre una Neurona de una Capa Oculta con una Neurona de Salida

$$? w_{ji}(t ? 1) ? ? ? _{pj} y_{pi} \quad (4)$$

En donde el subíndice p se refiere al patrón de aprendizaje concreto y $?$ es la constante o tasa de aprendizaje.

El punto en el que difieren la regla delta generalizada de la regla delta es en el valor concreto de $? _{pj}$. Por otro lado, en las redes multinivel, a diferencia de las redes sin neuronas ocultas, en principio no se puede conocer la salida deseada de las neuronas de las capas ocultas para determinar los pesos en función del error cometido. Sin embargo, inicialmente se puede conocer la salida deseada de las neuronas de salida. Según esto, si se considera la unidad U_j (ver Figura 16[23]), entonces se define:

$$? _{pj} ? (d_{pj} ? y_{pj}) * f ?(net_j) \quad (5)$$

Donde d_{pj} es la salida deseada de la neurona j para el patrón p y net_j es la entrada neta que recibe la neurona j.

Esta formula es como la de la regla delta, excepto en lo que se refiere a la derivada de la función de transferencia. Este término representa la modificación que hay que realizar en la entrada que recibe la neurona j. En el caso que dicha neurona no sea de salida, el error que se produce estará en función del error que se cometa en las neuronas que reciban como entrada la salida de dicha neurona. Esto es lo que se denomina procedimiento de propagación del error hacia atrás.

[23] Figura 16. Tomada de la tesis Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Reyes Figueroa, Juan Carlos, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)

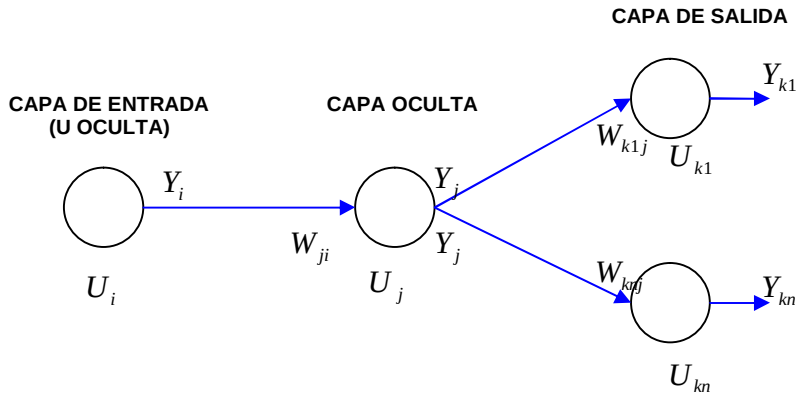


Figura 16. Conexiones Entre Neuronas de la Capa Oculta con la Capa de Salida

Según lo anterior, en el caso que U_j no sea una neurona de salida (ver Figura 16), el error que se produce está en función del error que se comete en las neuronas que reciben como entrada la salida de U_j :

$$? _{pj} ? ? ? ? _{pk} w_{kj} ? * f ?(net_j) \qquad (6)$$

Donde el rango de k cubre todas aquellas neuronas a las que está conectada la salida de U_j . De esta forma, el error que se produce en una neurona oculta es la suma de los errores que se producen en las neuronas a las que está conectada la salida de ésta, multiplicando cada uno de ellos por el peso de la conexión.

2.2.7.2.6 ADICIÓN DE UN MOMENTO EN LA REGLA DELTA GENERALIZADA

El método de retropropagación del error, también conocido como del gradiente descendente, requiere un gran número de cálculos para lograr el ajuste de los pesos de La RNA. En la implementación del algoritmo, se toma una amplitud de paso que viene dada por la tasa de aprendizaje ? . A mayor tasa de aprendizaje, mayor es la modificación de los pesos en cada iteración, con lo que el aprendizaje será más rápido, pero por otro lado, puede dar lugar a oscilaciones. Rumelhart, Hinton Y Willians (1986), sugirieron que para filtrar estas oscilaciones se añada en la expresión del incremento de los pesos un término momento ? , de manera que dicha expresión quede:

$$w_{ji}(t ? 1) ? w_{ji}(t) ? ? ? _{pj} y_{pi} ? ? (w_{ji}(t) ? w_{ji}(t ? 1)) ? \\ ? w_{ji}(t ? 1) ? ? ? _{pj} y_{pi} ? ? ? w_{ji}(t) \qquad (7)$$

Donde $?$ es una constante o momento que determina el efecto en $t ? 1$ del cambio de los pesos en el instante t.

Con este momento se consigue la convergencia de la red en menor número de iteraciones, porque si en t el incremento de un peso era positivo y en $t ? 1$ también, entonces el descenso por la superficie de error en $t ? 1$ es mayor. Si embargo, si en t el incremento era positivo y en $t ? 1$ es negativo, el paso que se da en $t ? 1$ es más pequeño, lo cual es adecuado, debido a que eso significa que se ha pasado por un mínimo y que los paso deben ser menores para alcanzarlo. Resumiendo el Algoritmo de Backpropagation queda finalmente:

Si U_j es una neurona de salida y

$$w_{ji}(t ? 1) ? w_{ji}(t) ? [? w_{ji}(t ? 1)] \quad (8)$$

$$w_{ji}(t ? 1) ? w_{ji}(t) ? [? ? _{pj} y_{pi} ? ? ? w_{ji}(t)] \quad (9)$$

Donde:

$$? _{pj} ? (d_{pj} ? y_{pj}) f ?(net_j) \quad (10)$$

$$? _{pj} ? (? _k ? _{pk} w_{kj}) f ?(net_j) \quad (11)$$

Si U_j no es una neurona de salida.

2.2.7.2.7 ESTRUCTURA Y APRENDIZAJE DE UNA RNA CON EL ALGORITMO BACKPROPAGATION

En una RNA multinivel existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (excepto la de entrada) recibe entradas, de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las salidas). No hay conexiones hacia atrás feedback ni laterales entre neuronas de la misma capa.

La aplicación del algoritmo Backpropagation tiene dos fases, una hacia delante y otra hacia atrás. Durante la primera fase el patrón de entrada es presentado a la RNA y propagado a través de las capas hasta llegar a la capa de salida. Obtenidos los valores de salida, se inicia la segunda fase, comparándose estos valores con la salida esperada para obtener el error. Se ajustan los pesos de la última capa proporcionalmente al error. Se pasa a la capa anterior con una retropropagación del error (Backpropagation), ajustando de manera conveniente los pesos y se continúa con este proceso hasta llegar a la primera capa.

A diferencia de la regla delta en el caso del Perceptrón, la técnica Backpropagation o generalización de la regla delta, requiere el uso de neuronas cuya función de activación sea continua, y por tanto, diferenciable. La función a utilizar será del tipo sigmoidal.

A continuación se presentan, a modo de síntesis, los pasos y fórmulas a utilizar para aplicar el algoritmo de entrenamiento.

Paso 1

Inicializar los pesos de La RNA con valores pequeños aleatorios.

Paso 2

Presentar un patrón de entrada, $X_P : x_{p1}, x_{p2}, ..., x_{pN}$ y especificar la salida deseada que debe generar La RNA: $d_1, d_2, ..., d_M$ (si la red se utiliza como un clasificador, todas las salidas deseadas serán cero, salvo una, que será la de la clase a la que pertenece el patrón de entrada).

Paso 3

Calcular la salida actual de la red, para ello se presentan las entradas a la red y se calcula la salida que presenta cada capa hasta llegar a la capa de salida ésta será la salida de la red $y_1, y_2, ..., y_M$. Los pasos son los siguientes:

- Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada.

  Para una neurona j oculta:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h} \tag{12}$$

  En donde el índice h se refiere a magnitudes de la capa oculta (hidden); el subíndice p, al p-ésimo vector de entrenamiento, y j a la j-ésima neurona oculta. El término θ puede ser opcional, pues actúa como una entrada más.

- Se calculan las salidas de las neuronas ocultas:

$$y_{pj} = f_j^h(net_{pj}^h) \tag{13}$$

? Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida (capa o: output).

$$net_{pk}^{o} ? \sum_{j?i}^{L} w_{kj}^{o} y_{pj} ? ? _{k}^{o} \quad (14)$$

$$y_{pk} ? f_{k}^{o}(net_{pk}^{o}) \quad (15)$$

Paso 4

Si la neurona k es una neurona de la capa de salida, el valor del delta es:

$$? _{pk}^{o} ? (d_{pk} ? y_{pk}) f_{k}^{o?}(net_{pk}^{o}) \quad (16)$$

La función f debe cumplir el requisito de ser derivable, lo que implica la imposibilidad de utilizar una función escalón. En general se disponen de dos formas de función de salida para cumplir el requisito: la función lineal de salida $(f_k(net_{jk}) ? net_{jk})$ y la función sigmoidal definida por la expresión:

$$f_k(net_{jk}) ? \frac{1}{1 ? e^{? net_{jk}}} \quad (17)$$

La selección de la función de salida depende de la forma en que se decida representar los datos de salida: si se desea que las neuronas de salida sean binarias, se utiliza la función sigmoidal, puesto que esta función es casi biestable y, además, derivable. En otros casos es tan aplicable un función como otra.

Para la función lineal, se tiene $f_k^{o?} ? 1$, mientras que la derivada de una función f sigmoidal es:

$$f_k^{o?} ? f_k^{o}(1 ? f_k^{o}) ? y_{pk}(1 ? y_{pk}) \quad (18)$$

Por lo que los términos de error para las neuronas de salida quedarían:

$$? _{pk}^{o} ? (d_{pk} ? y_{pk}) \quad (19)$$

Para la salida lineal, y

$$? _{pk}^{o} ? (d_{pk} ? y_{pk}) y_{pk}(1 ? y_{pk}) \quad (20)$$

Para la salida sigmoidal.

Si la neurona j no es de salida, entonces la derivada parcial del error no puede ser evaluada directamente. Por tanto, se obtiene el desarrollo a partir de valores que son conocidos y otros que pueden ser evaluados.

La expresión obtenida en este caso es:

$$? _{pj}^{h} ? f_j^{h?}(net_{pj}^{h}) \sum_{k}^{?} ? _{pk}^{o} w_{kj}^{o} \quad (21)$$

Donde se observa que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término de propagación hacia atrás. En particular, para la función sigmoidal:

$$? \, {}^{h}_{pj} \; ? \; x_{pj}(1 ? \; x_{pj}) \sum_{k} ? \, {}^{o}_{pk} w^{o}_{kj} \tag{22}$$

Donde k se refiere a todas las neuronas de la capa superior a la de la neurona j. Así, el error que se produce en una neurona oculta es proporcional a la suma de los errores conocidos que se producen en las neuronas a las que está conectada la salida de está, multiplicando cada uno de ellos por el peso de la conexión. Los umbrales internos de las neuronas se adaptan de forma similar, considerando que están conectados con pesos desde entradas auxiliares de valor constante.

Paso 5

Actualización de los pesos

Para ello, se utiliza el algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustado los pesos de la siguiente forma:
Para los pesos de las neuronas de la capa de salida:

$$w^{o}_{kj}(t ? 1) ? \; w^{o}_{kj}(t) ? \; ? \, w^{o}_{kj}(t ? 1) ; \tag{23}$$

$$? \, w^{o}_{kj}(t ? 1) ? \; ? \; ? \, {}^{o}_{pk} y_{pj} \tag{24}$$

Y para los pesos de las neuronas de la capa oculta:

$$w^{h}_{ji}(t ? 1) ? \; w^{h}_{ji}(t) ? \; ? \, w^{h}_{ji}(t ? 1) ; \tag{25}$$

$$? \, w^{h}_{ji}(t ? 1) ? \; ? \; ? \, {}^{h}_{pj} x_{pi} \tag{26}$$

En ambos casos, para acelerar el proceso de aprendizaje, se puede añadir un término momento de valor: $? \, (w^{o}_{kj}(t) ? \; w^{o}_{kj}(t ? 1))$ en el caso de la neurona de salida y $? \, (w^{h}_{ji}(t) ? \; w^{h}_{ji}(t ? 1))$ cuando se trata de una neurona oculta.

Paso 6

El proceso se repite hasta que el término de error

$$E_{p} \; ? \; \frac{1}{2} \sum_{k?1}^{M} ? \, {}^{2}_{pk} \tag{27}$$

Resulta pequeño para cada uno de los patrones aprendidos.

2.2.7.2.8 CONTROL DE CONVERGENCIA

En las técnicas de gradiente decreciente es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos. Esto se debe a que tenemos 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 sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarde más en llegar, se evita que ocurra esto.

El elegir un incremento paso adecuado influye en la velocidad con la que converge el algoritmo. Esta velocidad se controla a través de la constante de proporcionalidad o tasa de aprendizaje ? , normalmente, ? debe ser un número pequeño (del orden de 0.05 a 0.25), para asegurar que la red llegue a asentarse en una solución. Un valor pequeño de ? significa que la red tendrá que hacer un gran número de iteraciones. Si esa constante es muy grande, los cambios de pesos son muy grandes, avanzando rápido por la superficie de error, con el riesgo de saltar el mínimo y estar oscilando alrededor de él, pero sin poder alcanzarlo.

2.2.7.2.9 DIMENSIONAMIENTO DE LA RNA, NÚMERO DE NEURONAS OCULTAS

No se pueden dar reglas concretas para determinar el número de neuronas o el número de capas de una red para resolver un problema concreto. Lo mismo ocurre a la hora de seleccionar el conjunto de vectores de entrenamiento. En todos estos casos, lo único que se puede dar son ideas generales deducidas de la experiencia de numerosos autores [Freeman 91].

Respecto al número de capas de la red, en general tres capas son suficientes (entrad-oculta-salida). Sin embargo, hay veces en que un problema es más fácil de resolver (la red aprende más de prisa) con más de una capa oculta. El tamaño de las capas, tanto de entrada como de salida, suele venir determinado por la naturaleza de la aplicación. En cambio, decidir cuántas neuronas debe tener la capa oculta no suele ser tan evidente.

El número de neuronas ocultas interviene en la eficacia del aprendizaje y en la generalización de la red. No hay ninguna regla que indique el número óptimo, en cada problema se debe ensayar con distintos números de neuronas para organizar la representación interna y escoger el mejor. Un comienzo de construcción puede ser el de asumir que el número de neuronas en la capa oculta serian menos de la mitad de la suma del número de entradas más el de las salidas[24].

[24] Cita tomada de la tesis Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Reyes Figueroa, Juan Carlos, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)

2.2.7.2.10 APLICACIONES DE LAS RNA CON ALGORITMO BACKPROPAGATION

En la actualidad este tipo de RNA se aplica de manera sistemática en distintas clases de problemas, y por ello es difícil realizar una selección de las aplicaciones más relevantes. La versatilidad de la redes Backpropagation se debe a la naturaleza general de su proceso de aprendizaje, debido a que sólo se necesitan dos ecuaciones para propagar las señales de error hacia atrás.

Entre los campos de aplicación se encuentran:

- Codificación de la Información
- Traducción de Texto en Lenguaje Hablado
- Reconocimiento de Lenguaje Hablado
- Reconocimiento Óptico de Caracteres (OCR)

2.2.7.3 RNA DE BASE RADIAL (RBFN)

Son RNA multicapa con conexiones hacia delante, al igual que el Perceptrón multicapa, las RBFN se caracterizan porque están formadas por una única capa oculta y cada neurona de esta capa posee un carácter local, en el sentido que cada neurona oculta de la RNA se activa en una región diferente del espacio de patrones de entrada. Este carácter local viene dado por el uso de las llamadas funciones de base radial, generalmente, la función gaussiana, como funciones de activación. Las neuronas de la capa de salida de las redes de base radial realizan una combinación lineal de las activaciones de las neuronas ocultas.
La mayor contribución a la teoría, diseño y aplicaciones de las RNA de base radial se debe a Moody y Darken [Moody and Darken 1989], Renals [Renals 1989] y a Poggio y Girossi [Poggio and Girossi 1990]. Uno de los objetivos iniciales de los autores era construir una red de neuronas que requiriera un menor tiempo de aprendizaje que el necesario por el Perceptrón multicapa y, de este modo, disponer de una red de neuronas que pudiera ser apropiada para aplicaciones en tiempo real. Esto se consiguió incorporando funciones de activación locales en las neuronas ocultas de la red, lo cual permitía que sólo unas pocas neuronas ocultas tuvieran que ser procesadas para nuevos patrones de entrada.
Al igual que el Perceptrón multicapa, las redes de neuronas de base radial son aproximadores universales, en el sentido que pueden aproximar cualquier función continua sobre un compacto de R^N .
Las funciones de base radial definen hiperesferas o hiperelipses que dividen el espacio de entrada. Por lo tanto, cada neurona oculta de la red de base radial construye una aproximación local y no lineal en una determinada región de dicho espacio. Puesto que la salida de la red es combinación lineal de las funciones de

base radial. Las aproximaciones que construyen las RBFN son combinaciones lineales de múltiples funciones locales y no lineales. De este modo, se suele decir que las RBFN aproximan relaciones complejas mediante una colección de aproximaciones locales menos complejas, dividiendo el problema en varios subproblemas menos complejos.

2.2.7.3.1 ARQUITECTURA DE LAS RNA DE BASE RADIAL

Las RBFN están formadas por tres capas de neuronas: la capa de entrada, una única capa oculta y la capa de salida, como se ilustra en la Figura 17[25]. La capa de entrada la componen un conjunto de neuronas que reciben las señales del exterior, transmitiéndolas a la siguiente capa sin realizar ningún procesado sobre dichas señales. Las neuronas de la capa oculta reciben las señales de la capa de entrada y realizan una transformación local y no lineal sobre dichas señales. Este carácter local es lo que las diferencia del Perceptrón multicapa, no sólo en cuanto a arquitectura, sino también en cuanto a comportamiento. Esta capa es la única que incluye componentes no lineales en las redes de base radial. Y, finalmente, la capa de salida que realiza una combinación lineal de las actividades de las neuronas ocultas, que actúa además como salida de la RNA.

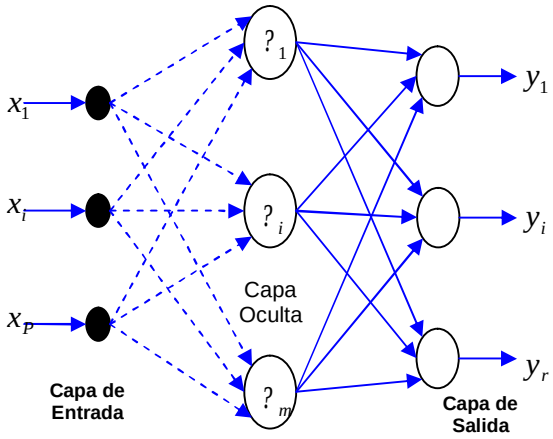


Figura 17. Arquitectura de las RBF

Las RBF son redes con conexiones hacia delante, estas conexiones se dirigen siempre de una capa a la siguiente capa. La red se caracteriza porque las conexiones de la capa de entrada a la capa oculta no llevan asociado ningún peso, mientras que, y como es habitual en el contexto de RNA, las conexiones de la capa oculta a la capa de salida si llevan asociado un número real peso de la conexión. En lo referente a los umbrales de las neuronas, en las RBF únicamente

[25] Figura 17. Tomada de la tesis Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Reyes Figueroa, Juan Carlos, UNIVERSIDAD INDUSTRIAL DE SANTANDER (2007)

las neuronas de salida poseen un umbral, que también se suele tratar, al igual que el Perceptrón multicapa, como una conexión más de la neurona cuya entrada es constante e igual a 1.

2.2.7.3.2 FUNCIONES DE ACTIVACIÓN DE LAS NEURONAS EN LAS REDES RBF

Dada una RBFN con p neuronas en la capa de entrada, m neuronas en la capa oculta y r neuronas en la capa de salida, la activación de las neuronas de salida para el vector de entrada $n, X(n) ? (x_1(n), x_2(n),..., x_p(n))$, descritas como $y_k(n)$, vienen dadas por la siguiente expresión:

$$y_k(n) ? \sum_{i?1}^{m} w_{ik} ?_i(n) ? u_k \qquad para\ k ? 1,2,...,r \tag{28}$$

Donde, w_{ik} representa el peso de la conexión de la neurona oculta i a la neurona de salida k, w_k es el umbral de la neurona de salida k y $?_i(n)$ son las funciones de activación de las neuronas ocultas para el vector de entrada $X(n)$, en la anterior ecuación, las neuronas de salida de la red utilizan la función de activación identidad, realizando una transformación lineal en la activación de todas las neuronas ocultas.

La función $?_i$, también conocidas como función de base radial, determina la actividad de las neuronas ocultas de la red en función del vector de entrada a la red $X(n)$ y vienen dadas por la siguiente expresión:

$$?_i(n) ? ? \left(\frac{\| X(n) ? C_i \|}{d_i} \right) \qquad para\ i ? 1,2,...,m \tag{29}$$

Donde $?$ es una función de base radial; $C_i ? (c_{i1},...,c_{ip})$ son vectores que representan los centros de la función de base radial; d_i son números reales que representa la desviación, anchura o dilatación de la función de base radial; y ||X(n) – C_i || es la distancia euclidiana del vector de entrada $X(n)$ al centro C_i, definida como:

$$\| X(n) ? C_i \| ? \left(\sum_{j?1}^{P} (x_j(n) ? c_{ij})^2 \right)^{\frac{1}{2}} \tag{30}$$

Por lo tanto, la activación de una neurona oculta en las RBFN depende de la distancia del vector de entrada $X(n)$ al centro C_i de la función de base radial. Esta función de base radial ? posee un carácter local, pues es una función que alcanza un nivel cercano al máximo de su recorrido cuando el vector de entrada $X(n)$ está próximo al centro de la neurona; a medida que el vector se aleja del centro, el valor de función tiende al valor mínimo de su recorrido.

La función de base radial ? puede adoptar diferentes formas y expresiones, entre ellas se encuentran las siguientes:

? Función Gaussiana

$$?(r)?e^{?\frac{?r^2?}{?2?}} \tag{31}$$

? Función Inversa Cuadrática

$$?(r)?\frac{1}{1?r^2} \tag{32}$$

? Función Inversa Multicuadrática

$$?(r)?\frac{1}{\sqrt{1?r^2}} \tag{33}$$

En el concepto de RBFN la más utilizada es la función gaussiana [Moody and Darken 89]. Por lo tanto, la activación de las neuronas ocultas de las RBFN viene dada, generalmente, por la siguiente expresión:

$$?_i(n)?\exp^{\frac{\|X(n)?C_i\|^2}{2d_i^2}}?\exp^{\frac{?\sum_{j?1}^{p}(x_j(n)?c_{ij})^2}{2d_i^2}} \qquad para\ i?1,2,..,m \tag{34}$$

Las salidas de las RBFN (28) son una combinación lineal de gaussianas, cada una de las cuales se activa para una determinada porción del espacio definido por los patrones de entrada.

2.2.7.3.3 DISEÑO Y SELECCIÓN DE LA ARQUITECTURA DE LAS REDES TIPO RBFN

El número de entradas y salidas en una RBFN viene dado por el número de variables que definen el problema. Como ocurría cuando se utilizaba un Perceptrón multicapa, en algunas aplicaciones no hay lugar a duda sobre dichas variables. Sin embargo, existen aplicaciones en las que pudiera ser necesario llevar a cabo un análisis de las variables más relevantes y significativas que definen el problema.

En lo que respecta al número de neuronas ocultas de la red, generalmente, se determina por prueba y error, variando el número de neuronas hasta conseguir una red capaz de resolver el problema, como en el caso del Perceptrón Multicapa (MLP) para determinar el número de capas ocultas y el número de neuronas en cada capa. No obstante, y a diferencia del MLP, para las RBFN añadir o eliminar unas pocas neuronas ocultas podría influir significativamente en los resultados obtenidos por la red. Esto se debe a que cada neurona oculta representa una determinada región del espacio de entrada, pudiendo no estar adecuadamente representado dicho espacio, bien por la presencia de pocas clases, bien por la presencia de muchas neuronas ocultas en una misma zona, con su posterior consecuencia negativa en la aproximación de la red.

2.2.7.3.4 APRENDIZAJE DE LAS REDES TIPO BASE RADIAL

El proceso de aprendizaje implica la determinación de todos los parámetros que intervine en la red RBFN. Estos son: los centros y las desviaciones de las neuronas ocultas y los pesos de la capa oculta a la capa de salida, así como los umbrales de las neuronas de salida.

Debido a que las capas de neuronas en una red de base radial realizan tareas diferentes, es razonable separar el proceso de optimización de los parámetros de la capa oculta y los de la capa de salida mediante la utilización de diferentes técnicas.

Por tanto, uno de los mecanismos más usados para el aprendizaje de las RBFN es el llamado ***método hibrido***, que combina dos fases: una fase no supervisada para la determinación de los centros y otra supervisada para la determinación de los pesos y los umbrales. Además del método anteriormente mencionado también se encuentra el método de aprendizaje totalmente supervisado, a continuación se describirá en detalle cada uno de estos métodos de aprendizaje, así como sus propiedades y características.

2.2.7.3.4.1 MÉTODO DE APRENDIZAJE HÍBRIDO

El método híbrido realiza el aprendizaje de las RBFN en dos fases:

- Fase no supervisada: determinación de los centros y amplitudes de las neuronas de la capa oculta.
- Fase supervisada: determinación de pesos y umbrales de la capa de salida.

El proceso de optimización en cada una de las fases se formula con un objetivo diferente y las técnicas para su resolución serán también, por lo tanto diferentes.

2.2.7.3.4.2 FASE NO SUPERVISADA

Puesto que las neuronas ocultas de las RBFN se caracterizan porque representan zonas diferentes del espacio de los patrones de entrada, los centros y las desviaciones de las funciones de base radial deben ser determinados con este objetivo, es decir, con el objetivo de clasificar el espacio de entrada en diferentes clases. El representante de cada clase será el centro de la función de base radial y la desviación vendrá dada por la amplitud de cada clase.

2.2.7.3.4.3 DETERMINACIÓN DE LOS CENTROS: ALGORITMO K-MEDIAS

Los centros de las funciones de base radial se determinan, por tanto, mediante un algoritmo de clasificación no supervisado que permita dividir el espacio de patrones de entrada en clases. El número de clases es el número de neuronas ocultas en la red de base radial. El método más utilizado es el algoritmo de K-medias, aunque es necesario destacar que cualquier algoritmo de clasificación no supervisado podría ser utilizado, como, por ejemplo, los mapas autoorganizados de Kohonen.

El algoritmo de K-medias [Lloyd 1982], [MacQueen 1967] es un algoritmo de clasificación no supervisado mediante el cual el espacio de patrones de entrada se divide en K clases o regiones. El representante de cada una de estas clases C_i, será el centro de la neurona oculta i. Dichos centros se determinan con el objetivo de minimizar las distancias euclidianas entre los patrones de entrada y el centro más cercano, es decir:

$$J = \sum_{i=1}^{K} \sum_{n=1}^{N} M_{in} \left\| X(n) - C_i \right\| \qquad (35)$$

Donde, N es el número de patrones, ||X(n)-C_i|| es la distancia euclidiana, $X(n)$ es el patrón de entrada n y Min es la función de pertenencia, que vale 1 si el centro C_i es el más cercano al patrón $X(n)$, y 0 en otro caso, es decir:

$$M_{in} = \begin{cases} 1 & si \left\| X(n) - C_i \right\| = \left\| X(n) - C_i \right\| \; s = i, s = 1,2,...,K \\ 0 & En\ otro\ caso \end{cases} \quad (36)$$

Dado K como el número de clases, $\{X(n) = (x_1(n), x_2(n),..., x_p(n))\}_{n=1...N}$ el conjunto de patrones de entrada y $\{C_i = (c_{i1}, c_{i2},..., x_{ip})\}_{i=1...K}$ los centros de las clases, los pasos para la aplicación del algoritmo son los siguientes:

Paso 1 Se inicializan los centros de las K clases. Pueden inicializarse a K patrones aleatorios del conjunto de patrones disponibles, o bien, puede realizarse aleatoriamente, en cuyo caso es conveniente que se inicialicen dentro del rango de valores de los patrones de entrada.

Paso 2 Se asignan N_i patrones de entrada a cada clase i del siguiente modo: el patrón $X(n)$ pertenece a la clase i si $\left\| X(n) - C_i \right\| = \left\| X(n) - C_s \right\| \; s = i \; con \; s = 1,2,...,K$. Por tanto, cada clase tendrá asociado un determinado número de patrones de entrada, aquellos más cercanos al centro de la clase.

Paso 3 Se calcula la nueva posición de los centros de las clases como la media de todos los patrones que pertenecen a su clase, es decir:

$$c_{ij} = \frac{1}{N_i} \sum_{n=1}^{N} M_{in} x_j(n) \; para \; j = 1,2,..., p \; , i = 1,2,..., K \quad (37)$$

Paso 4 Se repiten los pasos 2 y 3 hasta que las nuevas posiciones de los centros no se modifiquen respecto a si posición anterior, es decir, hasta que:

$$\left\| C_i^{nuevo} - C_i^{anterior} \right\| < \varepsilon \; \forall \; i = 1,2,..., K \quad (38)$$

Siendo, ε un número real positivo próximo a cero que marca la finalización del algoritmo.

El algoritmo de K-medias es un método fácil de implementar y usar: suele ser un algoritmo bastante eficiente en problemas de clasificación, pues converge en pocas iteraciones hacia un mínimo de la función J dada por la ecuación (33), aunque podría tratarse de un mínimo local. Uno de los inconvenientes o desventajas que se le atribuyen al algoritmo de K-medias es su dependencia de los valores iniciales asignados a cada centro, lo cual hace que muchas ocasiones, y siempre dependiendo del problema, se obtengan soluciones locales

2.2.7.3.4.4 DETERMINACIÓN DE LAS AMPLITUDES

Una vez determinados los centros de las funciones de base radial, las amplitudes o desviaciones de dichas funciones deben calcularse de manera que cada neurona oculta se active en una región del espacio de entrada y de manera que el solapamiento de las zonas de activación de una neurona a otra sea lo más ligero posible, para suavizar así la interpolación.

Las amplitudes de cada función de base radial se pueden determinar usando heurísticas basadas en los vecinos más cercanos, tal y como lo propuso Moody [Moody and darken 1989], los cuales permiten que el solapamiento entre las neuronas ocultas sea lo más suave posible. Se pueden utilizar diferentes heurísticas, como, por ejemplo:

- Media uniforme de las distancias euclidianas del centro C_i a los p centros más cercanos:

$$d_i = \frac{1}{p} \sum_{p} \left\| C_i - C_p \right\| \tag{39}$$

- Otra opción bastante efectiva es determinar la amplitud de la función de base radial como la media geométrica de la distancia del centro a sus dos vecinos más cercanos:

$$d_i \sqrt{\left\| C_i - C_t \right\| \left\| C_i - C_s \right\|} \tag{40}$$

Siendo C_t y C_s los dos centros más cercanos al centro C_i

2.2.7.3.4.5 FASE SUPERVISADA

En esta fase se calculan los pesos y los umbrales de las neuronas de salida de la red tipo RBFN. En este caso, el objetivo es minimizar las diferencias entre las salidas de la red y las salidas deseadas. Por tanto, el proceso de aprendizaje está guiado por la minimización de una función error computada en la salida de la red mediante:

$$E = \frac{1}{N} \sum_{n=1}^{N} e(n) \tag{41}$$

Donde N es el número de patrones o muestras y $e(n)$ es el error cometido por la red para el patrón $X(n)$, que viene dado generalmente por:

$$e(n) ? \frac{1}{2} ? \sum_{k?1}^{r} (s_k(n) ? y_k(n))^2 \quad (42)$$

Siendo $Y(n) ? (y_1(n),...,y_r(n))$ y $S(n) ? (s_1(n),...,s_r(n))$ los vectores de salida de la red y salida deseada para el patrón de entrada $X(n)$, respectivamente.

2.2.7.3.4.6 METODO DE MÍNIMOS CUADRADOS

Para resolver este problema de optimización se suele utilizar una técnica basada en la corrección del error. En la ecuación (28) se observa que las salidas de La RBFN dependen linealmente de los pesos y umbrales, por lo que un método bastante simple y eficiente es el algoritmo de los mínimos cuadrados. De este modo, los pesos y umbrales de la red se determinan mediante un proceso iterativo gobernado por la siguiente ley:

$$w_{ik}(n) ? w_{ik}(n ? 1) ? ? _1 \frac{?e(n)}{?w_{ik}} \quad (43)$$

$$u_k(n) ? u_k(n ? 1) ? ? _1 \frac{?e(n)}{?u_k} \quad (44)$$

$$Para\ k ? 1,2,...,r \quad y \quad para\ i ? 1,..,m$$

Donde, $e(n)$ es el error dado por la ecuación (41) y $? _1$ es la razón o tasa de aprendizaje.

Teniendo en cuenta la expresión del error y que el peso w_{ik} y el umbral u_k únicamente afectan a la neurona de salida k se obtiene que:

$$\frac{?e(n)}{?w_{ik}} ? ?(s_k(n) ? y_k(n)) \frac{?y_k(n)}{?w_{ik}} \quad (45)$$

$$\frac{?e(n)}{?u_k} ? ?(s_k(n) ? y_k(n)) \frac{?y_k(n)}{?u_k} \quad (46)$$

Derivando la salida $y_k(n)$ de la RBFN dada en la ecuación (28) respecto a los pesos y umbrales, se obtiene que:

$$\frac{?y_k(n)}{?w_{ik}} ? ? _i(n) \quad (47)$$

Donde $?_i(n)$ es función de la activación de la neurona oculta i para el patrón de entrada $X(n)$, y por lo tanto:

$$\frac{?y_k(n)}{?u_k} ? 1 \tag{48}$$

Por consiguiente, las leyes dadas por las ecuaciones (43) y (44) para adaptar los pesos y los umbrales de la capa de salida de la red de base radial se pueden escribir de la siguiente forma:

$$w_{ik}(n) ? w_{ik}(n ? 1) ? ?_1(s_k(n) ? y_k(n))?_i(n) \tag{49}$$

$$u_k(n) ? u_k(n ? 1) ? ?_1(s_k(n) ? y_k(n)) \tag{50}$$

$$Para\ k ? 1,2,...,r \quad y\ para\ i ? 1,..,m$$

Cuando se calculan los pesos mediante la ley de aprendizaje dada por las ecuaciones (49) y (50), la convergencia es bastante rápida, consiguiendo una solución en un número pequeño de iteraciones o ciclos de aprendizaje.

2.2.7.3.5 MÉTODO DE APRENDIZAJE TOTALMENTE SUPERVISADO

A diferencia del método de aprendizaje híbrido descrito con antelación, el método de aprendizaje totalmente supervisado no conserva, en principio, las propiedades o características locales de las RBFN (centros, amplitudes, pesos y umbrales) se determinan de manera supervisada y con el objetivo de minimizar el error cuadrático medio, es decir, las diferencias entre las salidas de la red y las salidas esperadas.

Al utilizar este método en ningún momento el proceso de aprendizaje se guía para que las amplitudes alcancen valores tales que el solapamiento de las activaciones de las neuronas ocultas sea lo más suave posible, sino que se determinan para minimizar el error cometido por la red en la capa de salida. Por lo tanto, no es posible esperar que la red siga conservando sus características locales.

En las redes tipo RBFN las salidas de la red dependen linealmente respecto de los pesos y umbrales. Sin embargo, la dependencia pasa a ser no lineal en lo que respecta a los parámetros de las neuronas ocultas (centros y desviaciones) debido al uso de las funciones de base radial, las cuales son funciones no lineales. Por tanto, al utilizar el método de aprendizaje totalmente supervisado, los parámetros de la red se determinan empleando una técnica de optimización no lineal, concretamente el método de descenso del gradiente. Mediante un proceso iterativo y siguiendo la dirección negativa del gradiente del error, el método proporciona un mínimo, que puede ser local, de la función error. De este modo, los centros, amplitudes, pesos y umbrales de la red se modifican para cada patrón $X(n)$ de acuerdo con las siguientes leyes de aprendizaje.

$$w_{ik}(n) ? w_{ik}(n ? 1) ? ? _1 \frac{?e(n)}{?w_{ik}} \quad (51)$$

$$u_k(n) ? u_k(n ? 1) ? ? _1 \frac{?e(n)}{?u_k} \quad (52)$$

$$c_{ij}(n) ? c_{ij}(n ? 1) ? ? _2 \frac{?e(n)}{?c_{ij}} \quad (53)$$

$$d_i(n) ? d_i(n ? 1) ? ? _3 \frac{?e(n)}{?d_i} \quad (54)$$

$$Para\ j ? 1,2,..., p, Para\ k ? 1,2,..., r \quad y\ para\ i ? 1,.., m$$

Donde $? _1, ? _2$ y $? _3$ son las razones o tasas de aprendizaje para los pesos, centros y amplitudes, respectivamente. Dichas tasas no tienen por qué tomar los mismos valores.

En el caso de las redes tipo RBFN, el uso del método de descenso del gradiente no implica una retropropagación del error, como ocurría en la regla delta generalizada para el aprendizaje del Perceptrón multicapa. En el contexto de las redes de base radial, la aplicación del método de descenso del gradiente implica el cálculo de la derivada del error $e(n)$ respecto a cada uno de los parámetros (centros, amplitudes o desviaciones, pesos y umbrales). Dichas derivadas poseen expresiones diferentes, porque cada uno de estos parámetros interviene de manera distinta en la salida de la red. A continuación, se desarrollan las leyes del aprendizaje para los parámetros de las redes tipo RBFN.

2.2.7.3.5.1 PESOS Y UMBRALES

Las derivadas del error $e(n)$ respecto a los pesos y umbrales de la red han sido obtenidas en el apartado anterior (fase supervisada del método híbrido), obteniendo las leyes de aprendizaje dadas por las ecuaciones (47) y (48).

2.2.7.3.5.2 CENTROS

Teniendo en cuenta la expresión del error $e(n)$ y aplicando la regla de la cadena para derivar, la derivada de dicho error respecto al parámetro c_{ij} viene dada por:

$$\frac{?e(n)}{?c_{ij}} ? ? \sum_{k?1}^{r} (s_k(n) ? y_k(n)) \frac{?y_k(n)}{?c_{ij}} \quad (55)$$

El parámetro c_{ij} (coordenada j del centro i) sólo interviene en la activación de la neurona oculta i, por lo que al derivar la salida k de la red sólo en necesario derivar el término i de la sumatoria, cuyo resultado quedaría como:

$$\frac{?e(n)}{?c_{ij}} ? ? \sum_{k?1}^{r} (s_k(n) ? y_k(n)) w_{ik} \frac{?? _i}{?c_{ij}} \quad (56)$$

Aplicando de nuevo la regla de la cadena para derivar la función $? _i$ respecto a c_{ij} se obtiene que:

$$\frac{?? _i(n)}{?c_{ij}} ? ? _i(n) \frac{(x_j ? c_{ij})}{d_i^2} \quad (57)$$

Sustituyendo la ecuación (55) en la (54), la ley para modificar los centros de las funciones de base radial dada por la ecuación (53) adopta la siguiente expresión:

$$c_{ij}(n) ? c_{ij}(n ? 1) ? ? _2 (\sum_{k?1}^{r} (s_k(n) ? y_k(n)) w_{ik}) ? _i(n) \frac{(x_j ? c_{ij})}{d_i^2} \quad (58)$$

$$Para\ j ? 1,2,..., p, y\ para\ i ? 1,.., m$$

2.2.7.3.5.3 AMPLITUDES

Al igual que en el caso anterior, para obtener la derivada del error $e(n)$ respecto al parámetro d_i es necesario derivar las salidas de la red respecto a dicho parámetro:

$$\frac{?e(n)}{?d_i} ? ? \sum_{k?1}^{r} (s_k(n) ? y_k(n)) w_{ik} \frac{?? _i(n)}{?d_i} \quad (59)$$

La derivada de la función $? _i$ respecto al parámetro d_i es:

$$\frac{??_i(n)}{?d_i} \; ? \; ?_i(n)\frac{\|X(n) ? C_i\|^2}{d_i^3} \qquad (60)$$

Por tanto, la ley para modificar las amplitudes de las funciones de base radial es:

$$d_i(n) \; ? \; d_i(n ? 1) ? ? _3(\sum_{k?1}^{r}(s_k(n) ? y_k(n))w_{ik})?_i(n)\frac{\|X(n) ? C_i\|^2}{d_i^3} \qquad (61)$$

$$Para\ i ? 1,..,m$$

2.2.7.3.6 APLICACIONES DE LAS RNA

Desde la década de los 50' se comenzó a utilizar el modelo de redes neuronales artificial, para diferentes usos. En 1959 se obtuvo la primera aplicación comercial, esta se baso en el modelo ADALINE (ADAptative LINear Elements) que fue desarrollado por Bernard Widrow y Marcial Hoff. Con esta aplicación se demostró que estos modelos eran adaptables a soluciones de problemas reales, pues se logro eliminar el eco en las líneas telefónicas.

Actualmente, se tiene conocimiento de la utilización de este tipo de modelos con gran éxito en el campo militar, la biología, el medio ambiente, las finanzas, la manufacturación, la medicina y empresas de diferente índole

2.3 LA TRANSFORMADA DE FOURIER[26]

2.3.1 INTRODUCCIÓN

La transformada de Fourier es una de las primeras herramientas que se proponen inicialmente en el tratamiento de cualquier señal. Es ampliamente conocida su funcionalidad y permite en este caso realizar una primera aproximación a la caracterización de señales. Ya que realiza una descomposición de las señales en sus componentes frecuenciales.

Si no se tienes nociones previas, puede ser complicado comprender el concepto de "representación en frecuencia de una señal". Básicamente la Transformada de Fourier se encarga de transformar una señal del dominio del tiempo, al dominio de la frecuencia, de donde se puede realizar su Transformada inversa y volver al dominio temporal

[26] Cita tomada de http://www.arrakis.es/~ppriego/fourier/transf_f.htm

2.3.1 SERIE DE FOURIER

Sea una función f(t) una función periódica de periodo T, la cual se puede representar por la serie trigonométrica.

$$f(t) ? \frac{1}{2}a_0 ? a_1 \cos ? _{0t} ? a_2 \cos 2? _{0t} ? ... ? b_1 sen ? _0 t ? b_2 sen 2? _0 t ? ...$$

$$? \frac{1}{2}a_0 ? \sum_{n?1}^{?} (a_n \cos n? _0 t ? b_2 sen n? _0 t) \quad (62)$$

Donde $? _0 = 2? /T$.

Una serie como la representada se llama serie trigonometrica de Fourier. Esta serie también se puede representar así:

$$f(t) ? C_0 ? \sum_{n?1}^{?} C_n \cos(n? _0 t ? ? _n) \quad (63)$$

Por lo tanto, se obtiene que:

$$f(t) ? \frac{1}{2}a_0 ? \sum_{n?1}^{?} (a_n \cos n? _0 t ? b_2 sen n? _0 t) ? C_0 ? \sum_{n?1}^{?} C_n \cos(n? _0 t ? ? _n) \quad (64)$$

Es obvio que la representación de Fourier de una función periódica, representa la función como la suma de componentes sinusoides que tienen diferentes frecuencias. La componente senosoidal de frecuencia $? _n ? n? _0$ se denomina la enésima armónica de la función periódica. La primera armónica comúnmente se conoce como la componente fundamental porque tiene el mismo período de la función y $? _0 ? 2? f_0 ? 2? /T$ se conoce como la frecuencia angular fundamental. Los coeficientes C_n y los ángulos $? _n$ se conocen como amplitudes armónicas y ángulos de fase, respectivamente.

2.3.2.1 DEFINICIÓN TRANSFORMADA DE FOURIER

La transformada de Fourier se emplea con señales periódicas a diferencia de la serie de Fourier. Las condiciones para poder obtener la transformada de Fourier son las condiciones de Dirichlet:

- Que la señal sea absolutamente integrable
- Que tenga un grado de oscilación finito.

- Que tenga un número máximo de discontinuidades.

La transformada de Fourier es una particularización de la transformada de Laplace cuando S=jw, siendo w=2*pi*f, y se define como:

$$x(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega \cdot t} dt \quad (65)$$

Y su Transformada inversa se define como:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} x(\omega) \cdot e^{j\omega \cdot t} d\omega \quad (66)$$

He mencionado al principio que la transformada de Fourier se usa con señales aperiódicas. Con la invención de la función delta(t) a principios de este siglo es posible calcular la transformada de Fourier de una señal periódica:

Entonces se tiene que para una X(t) periódica se cumple que:

$$x(t)_{periodica} \xleftrightarrow{F} \sum_{k=-\infty}^{\infty} a_k 2\pi\delta(\omega - k\omega_0) \quad (68)$$

La transformada de Fourier se utiliza para pasar al dominio de la frecuencia una señal de entrada, así se puede obtener información que no es evidente en el dominio del tiempo. Matemáticamente se demuestra que una señal periódica se puedo descomponer en una suma de senos y cósenos formando una base ortogonal, de esta forma, la señal se pueden descomponer en varias señales trigonometricas. El conjunto de constantes que multiplican a cada frecuencia forman el espectro de frecuencias.

2.3.2.2 PROPIEDADES DE LA TRANSFORMADA DE FOURIER

A continuación en la tabla 2 se describen las propiedades mas relevante de la transformada de fourier[27], se omitieron otras por la complejidad de su explicación

[27] Para profundizar en propiedades de la transformada de fourier diríjase a http://cnx.org/content/m12957/latest/

Propiedad	Descripción
Linealidad	La combinación de las propiedades de la adición y de la multiplicación escalar se obtiene la linealidad. Si se toma la Transformada de Fourier de una combinación lineal de señales entonces esta será la misma que la combinación lineal de la transformada de Fourier de cada señal individual. Esto es ideal en el uso de la tabla de las transformadas para encontrar la transformada de una señal más complicada.
Simetría	Para calcular la magnitud de los puntos de un periodo completo, tan sólo necesitamos calcular los N/2+1 primeros puntos, siempre y cuando el origen de la transformada este centrado en el punto (N/2,N/2).
Escalamiento en el Tiempo	Esta propiedad trata con el efecto de la representación del dominio de frecuencia de una señal si la variable del tiempo es alterada. El concepto más importante para entender la propiedad de escalamiento es que las señales que son estrechas en el tiempo son amplias en la frecuencia y vice versa.
Desplazamiento en el tiempo	El desplazamiento en el tiempo muestra que un desplazo en el tiempo es equivalente a un desplazo de fase lineal en la frecuencia. Ya que el contenido de la frecuencia depende solamente de la forma de la señal, el cual es invariable en el desplazo en el tiempo, entonces solamente la fase del espectro será alterada.
Modulación (Desplazo de la Frecuencia)	La modulación es absolutamente imprescindible para las aplicaciones de comunicaciones. Siendo capaces de desplazar una señal a diferentes frecuencias sin que estas se alteren:
Convolución	Convolución es una de las grandes razones para convertir señales en dominios de frecuencia ya que la convolución en el tiempo se convierte en multiplicación en frecuencia. Esta propiedad es también otro buen ejemplo de la simetría entre el tiempo y la frecuencia.
Diferenciación en tiempo	Ya que los sistemas LTI pueden ser representados en términos de ecuaciones diferenciales, es evidente que con esta propiedad que convirtiendo al dominio de frecuencia nos permitirá convertir esta complicada ecuación diferencial a una ecuación más sencilla que involucre multiplicación y adición..

Tabla 2. Propiedades de la Transformada de fourier

CAPITULO 3

HERRAMIENTA SOFTWARE PARA LA IDENTIFICACIÓN DE SISTEMAS DINÁMICOS NO LINEALES MEDIANTE RNA, NEURALMOTOR 1.0

3.1. ORIGEN DE LA IDE*A*

La propuesta de investigación nació como respuesta a una inquietud de comprobar si las RNA realmente se pueden aplicar en la identificación de sistemas dinámicos no lineales. Y si los resultados obtenidos de estas aplicaciones pueden comprobar que las RNA funcionan correctamente en la preedición de respuestas acertadas para cada caso en estudio. No obstante, esta investigación plantea una dificultad mayor, ya que no se tienen antecedentes en nuestro ambiente local de trabajos realizados con RNA, enfocados a la identificación de sistemas dinámicos No lineales, específicamente en el análisis del funcionamiento de un motor utilizando los datos reales de entrada y salida del sistema, en donde se presenten resultados concretos que avalen la utilización de la RNA en este campo, aún cuando se conoce que las RNA están siendo utilizadas exitosamente en varios campos de investigación,.

La idea de diseñar y construir una herramienta software capaz de predecir el estado de un motor a partir de sus datos de presión en el cilindro y ángulo de giro del cigüeñal, fue formalizándose a medida que se indagaba sobre el estado del arte, y se concreto cuando se logro identificar los requerimientos del sistema y las potencialidades del producto final que inicialmente se dirigen a la comunidad educativa y científica de la Universidad Industrial de Santander, pero en la etapa de construcción y prueba de la herramienta se evidencio la aplicabilidad comercial del producto.

3.2 ANÁLISIS

El análisis se inicio con la recolección de información referente a identificación de sistemas, RNA y filtrado de señales, pero el panorama encontrado no fue benévolo, en referencia a fuentes de información, investigaciones y desarrollo de aplicaciones en el diagnostico de motores de combustión interna, no obstante, se dio inicio a la recopilación de información en libros, tesis, Web, en la base de datos de la IEEE y en consultas a expertos, logrando profundizar los conceptos y aclarando las dudas, hasta el punto de convérsenos de la viabilidad de la propuesta de construir una herramienta software para la identificación de sistemas dinámicos basado en RNA..

Después de organizar y estudiar toda la información recopilada hasta el momento, y a pesar de los buenos resultados que se han obtenido con la Inteligencia Artificial, donde se incluye las RNA, y el excelente desempeño en varios campos de la industria, la ciencia, la manufacturas y las empresas prestadoras de servicios, no estaba demás indagar sobre la utilización de las RNA en la identificación de sistemas dinámicos no lineales, y específicamente en la predicción del estado de un motor de combustión interna empleando la presión generada en sus cilindros.

Por otra parte, la documentación requerida para soportar los conceptos manejados en esta investigación, se encuentra en el capitulo 1 y 2, donde esta recopilada la información acerca de identificación de sistemas y se recalca la de los sistemas dinámicos No lineales, también se encuentra información valiosa de las topologías de RNA, el perceptròn multicapa y su respectivo algoritmo de aprendizaje Backpropagation y la red de Función de Base Radial (RBF), las cuales fueron utilizadas en la construcción de la herramienta ***NEURALMOTOR 1.0,*** Igualmente en el capitulo 2 se documenta el procedimiento de preprocesamiento de los datos de entrada del motor, con la aplicación de la transformada de Fourier y su respectiva transformada inversa.

Para la realización de la herramienta software, se propone un modelo de desarrollo en cascada documentado bajo estándares UML en las fases de análisis, diseño y construcción de NEURALMOTOR 1.0, donde sobresalen las actividades de definición de requisito y requerimientos del sistema, el posterior análisis de cada uno de estos, paso seguido el diseño de la aplicación, su construcción y se termina con la de implementación, aclaramos que elegimos Delphi®[28], en su versión 7.0, como el lenguaje de programación, debido a las ventajas y facilidades de su parte gráfica, componentes gratuitos que ayudan a la construcción de software.

Para la recolección de requerimientos se buscó apoyo del PhD. Jorge Luís Chacón Velasco, docente de Ingeniera Mecánica UIS, quien nos brindo suficiente orientación concerniente al funcionamiento de los motores de combustión interna en su parte mecánica, la definición de conceptos referentes a las RNA, su utilización en la identificación de sistemas dinámicos no lineales y el tratamiento de los datos de la señal de entrada. En la realización del aplicativo, se contó con la ayuda del MSc Juan Carlos Reyes codirector del proyecto, que cuenta con amplia experiencia en el área de la inteligencia artificial, especialmente en redes neuronales artificiales y lógica difusa, lo cual nos facilito el proceso de preprocesamiento de los datos de entrada y el posterior entrenamiento de las RNA.

[28] Lenguaje de programación orientado a objetos

3.3. DISEÑO

Se partió de la idea que la herramienta software debía contener por los menos dos tipos de RNA, para dilucidar que el caso de estudio se podía abordar con esta metodología, esclareciendo cual de los dos tipos de RNA proveería resultados mas precisos; de antemano se conocía la utilización de las red Perceptrón Multicapa (PMC) en el estudio de este caso, pero no del uso de la RNA de Base Radial (RBF), esto era un desafió mas para la investigación.

Desde el inicio, se había propuesto diseñar un software que pudiese brindar apoyo al proceso de diagnóstico de un motor de un automóvil, a partir de los datos suministrado por el banco de pruebas, donde suministran los valores de presión en los cilindros.

Inicialmente, se simuló el proceso en la toolbox de Matlab, donde están implementados varios tipos de RNA, entre ellas las redes Backpropagation y las redes de Función de Base Radial (RBF), elegidas y posteriormente implementadas en esta investigación, también se procedió a esclarecer las funcionalidades que debía contener la herramienta software, así fue que se realizo el diseño de cada red y se llego a la conclusión final de ayudar mediante un editor de RNA, el análisis de los datos que se utilizan en el entrenamiento de una red previamente definida, los cuales son generalizados con el fin de lograr la clasificación y predicción de los mismos.

Además se diseñó una herramienta de procesamiento de los datos, donde se clasifican de tal forma que estos queden reducidos al número de cilindros del motor, que equivale al número de entradas de de la RNA sin importar sus Topología, además se incluyo la generación de reportes y la creación de imágenes 2D. Igualmente se incluyo la opción de normalización, donde el usuario puede normalizar los datos de entrenamiento y simulación, lo que permite un entrenamiento mas preciso de la RNA.

La aplicación desarrollada deberá ser útil en la construcción, administración, entrenamiento y simulación de RNA de aprendizaje supervisado e hibrido, por esta razón se implementan dos de los algoritmos más utilizados en la industria para la determinación de resultados en base a un entrenamiento con datos de muestra.

3.4. DESARROLLO

La construcción de la herramienta contó con varias etapas, inicialmente se programo la función de preprocesamientos de datos, pero en esta etapa se acumularon tantas dudas sobre este procedimiento, que se debió iniciar el desarrollo de las redes. En esta segunda etapa, la programación del modulo del algoritmo básico de aprendizaje Backpropagation, en donde también surgieron problemas, puesto que la RNA no aprendía correctamente o se quedaba en

mínimos locales, por ello fue necesario implementar un algoritmo de mayor complejidad, que incluye un mometum en la actualización de los pesos, así como la posibilidad de elegir variadas funciones de activación en las diferentes capas.

En el desarrollo del algoritmo de aprendizaje y creación de la red de base radial, se decidió por el uso del algoritmo K-Medias para encontrar los centroides de los grupos iniciales, así como la regla delta generalizada para hallar los pesos ideales de conexión entre la capa de salida y la capa oculta.

Cuando se logro la implementación de estas dos redes se regreso a la etapa de preprocesamiento para corregir los errores que detuvieron su desarrollo, a medida que se fueron probando las redes con cierto número de datos, se fueron aclarando las dudas y se logro optimización del proceso de preprocesamiento.

Se implemento un editor, que permite cargar una cantidad de datos N y al aplicarle el correspondiente preprocesamiento, donde se reduce el número de datos a la cantidad de cilindros del motor, además se puede crear una RNA con una serie de parámetros propuestos por el usuario, o generar varias redes que posteriormente serán probadas. NEURALMOTOR 1.0, también permite cargar, guardar, renombrar, inicializar los pesos, clonar, eliminar y revisar los detalles de una RNA creada. Así mismo permite, cargar datos de entrenamiento, simulación y de la señal de entrada del motor desde un archivos de texto externos.

La documentación del desarrollo de la herramienta NEURALMOTOR 1.0, se hizo bajo estándares del Lenguaje de Modelamiento Unificado (UML - Unified Modeling Language) que es el lenguaje indicado para modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

A continuación se muestran los principales diagramas de desarrollo de la herramienta empleando el lenguaje de modelamiento unificado (UML), con estos diagramas se describe las funcionalidades de NEURALMOTOR 1.0, así, usted podrá forma una idea clara de la estructura y funcionamiento de la herramienta. Prescindimos de otros diagramas de modelamiento por que creemos que con estos es suficiente para explicar puntualmente las fases de diseño y construccion de NEURALMOTOR 1.0; los diagramas seleccionados son los siguientes:

- Diagramas de casos
- Diagramas de secuencia
- Diagrama de clases

3.4.1. DIAGRAMA DE CASOS DE USO

Los diagramas de Casos de Uso, son una representación gráfica de los actores[29] y su interacción con el sistema. Un diagrama de casos de uso muestra, los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relacionan con su entorno. A continuación, en la figura 18[30] se muestra el diagrama de casos de uso de NEURALMOTOR 1.0.

Figura 18. Diagrama de casos de uso de NEURALMOTOR 1.0

[29] Un actor es una entidad que utiliza alguno de los casos de uso del sistema
[30] Figura 18. Realizada por los autores

Para una descripción más detallada de la estructura de NEURALMOTOR 1.0, se ha dividido el esquema mostrado en la figura 18 en tres etapas, administración, entrenamiento y simulación, esto con se hace con el propósito de presentar claramente cada una de las funcionalidades de la herramienta, socializar el diseño de la interfaz y orientar la compresión del lector.

En la figura 19[31] se muestra la primera división y corresponde al diagrama de casos de usos de la administración de la RNA y comprende las operaciones habilitadas en el menú de la interfaz de usuario, donde se puede realizar diferentes procedimiento de manejo d las RNA, como crear, guardar cargar etc.

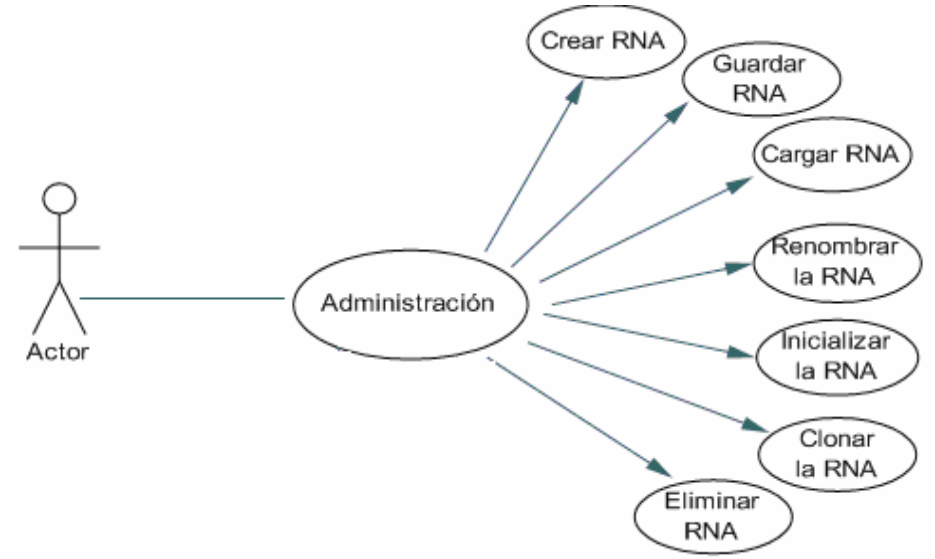


Figura 19. Diagrama de casos de usos de la administración de la RNA.

A continuación se muestra en detalle cada uno de los casos de uso para el componente Administración.

Crear la RNA	
Actor	Usuario
Propósitos	Crear una RNA
Resumen	Por medio de este caso de uso el actor crea una RNA
Flujo principal	1. El actor especifica los parámetros de creación de la RNA. 2. El sistema crea la red neuronal de acuerdo a estos parámetros. 3. El sistema pone a disposición del usuario la RNA creada. 1a: El actor cancela el proceso El sistema regresa al módulo principal.
Precondición	Ninguna

[31] Figura 19. Realizada por los autores

Guardar la RNA	
Actor	Usuario
Propósitos	Almacenar una RNA en un archivo válido
Resumen	Por medio de este caso de uso el actor puede guardar en un archivo la red seleccionada en cualquiera de los formatos disponibles.
Flujo principal	1. El actor selecciona la RNA a guardar. 2. El actor suministra el nombre y el tipo del archivo. 3. El sistema guarda la RNA en un archivo válido 2a: El actor cancela la operación. El sistema retorna al módulo principal
Precondición	Debe haber por lo menos una RNA creada

Cargar RNA	
Actor	Usuario
Propósitos	Cargar una RNA.
Resumen	Por medio de este caso de uso el actor puede cargar una RNA desde un archivo válido
Flujo principal	1. El actor selecciona la ubicación y el nombre del archivo. 2. El sistema crea una RNA de acuerdo a los datos almacenados en el archivo. 3. El sistema pone a disposición del usuario la RNA cargada. 1a: El actor selecciona un archivo no válido. El sistema cancela la operación y retorna al módulo principal. 1b: El actor cancela la operación. El sistema retorna al módulo principal.
Precondición	Ninguna

Renombrar la RNA	
Actor	Usuario
Propósitos	Cambiar el nombre a una RNA.
Resumen	Por medio de este caso de uso el actor puede cambiar el nombre de la RNA seleccionada.
Flujo principal	1. El actor selecciona la RNA. 2. El actor suministra el nuevo nombre de la RNA. 3. El sistema cambia el nombre de la RNA seleccionada. 2a: El actor suministra el nombre de una RNA existente. El sistema cancela la operación y regresa al módulo principal. 2b: El actor suministra un nombre no válido. El sistema cancela la operación y regresa al módulo principal.
Precondición	Debe existir por lo menos una RNA creada

.

Inicializar RNA	
Actor	Usuario
Propósitos	Inicializar una RNA.
Resumen	Por medio de este caso de uso el actor puede inicializar los pesos y/o centros de la RNA seleccionada
Flujo principal	1. El actor selecciona la RNA. 2. El sistema espera la confirmación por parte del actor. 3. El sistema inicializa la RNA seleccionada. 2a: El actor cancela la operación. El sistema regresa al módulo principal.
Precondición	Debe existir por lo menos una RNA creada

Clonar la RNA	
Actor	Usuario
Propósitos	Clonar una RNA.
Resumen	Por medio de este caso de uso el actor puede crear una RNA exactamente igual a la seleccionada.
Flujo principal	1. El actor selecciona la RNA. 2. El actor suministra el nombre de la nueva RNA. 3. El sistema crea una RNA exactamente igual a la seleccionada. 4. El sistema pone a disposición del actor la RNA clonada. 2a: El actor suministra el nombre de una RNA existente. El sistema cancela la operación y regresa al módulo principal. 2b: El actor suministra un nombre no válido. El sistema cancela la operación y regresa al módulo principal.
Precondición	Debe existir por lo menos una RNA creada

Eliminar RNA	
Actor	Usuario
Propósitos	Eliminar una RNA.
Resumen	Por medio de este caso de uso el actor puede eliminar la RNA seleccionada.
Flujo principal	1. El actor selecciona la red. 2. El sistema espera la confirmación por parte del actor. 3. El sistema elimina la red de la lista. 2a: El actor cancela la operación. El sistema regresa al módulo principal.
Precondición	Debe existir por lo menos una RNA creada

.

Ahora continuamos con la segunda división, el correspondiente al componente de entrenamiento, con se aprecia en la figura 20[32], esta es la división mas compleja, por que abarca la parte robusta de la herramienta software que es el tratamiento de datos y el entrenamiento.

Figura 20. Casos de uso del componente entrenamiento.

Datos	
Actor	Usuario
Propósitos	Ingresar a la interfaz de datos utilizados en entrenamiento
Resumen	Por medio de este caso de uso el actor acceder a la ventana de manejo de los datos de entrenamiento
Flujo principal	1. El actor ingresa al componente entrenamiento. 2. El sistema despliega la ventana de entrenamiento 3. EL actor ingresa a la ventana datos 4. El sistema despliega la ventana datos. 1a: El actor cancela el proceso El sistema regresa al módulo principal
Precondición	Se requiere que exista una red

[32] Figura 20. Realizada por los autores

Preprocesamiento de los Datos	
Actor	Usuario
Propósitos	Tomar los datos de entrada y aplicarle procesamiento de datos para eliminar ruido y organizarlos para las operaciones posteriores
Resumen	Por medio de este caso de uso el actor puede generar eliminar errores en los datos de entrada y eliminar el ruido.
Flujo principal	1. El actor suministra los datos de entrada. 2. El sistema detecta los errores en los datos. 3. El sistema filtra los datos y elimina el ruido. 1a: El actor cancela el proceso El sistema regresa al módulo principal
Precondición	Se requiere de los datos de entrada de la señal

Entrenar la RNA	
Actor	Usuario
Propósitos	Entrenar una RNA.
Resumen	Entrenar una RNA de acuerdo a los datos de entrenamiento suministrados.
Flujo principal	1. El actor selecciona la RNA. 2. El actor carga o introduce los datos de entrenamiento. 3. El actor configura los parámetros de entrenamiento. 4. El sistema entrena la RNA de acuerdo a los datos y parámetros suministrados.
Precondición	Debe haber por lo menos una RNA creada, unos datos de entrenamiento y unos parámetros válidos.

Gráfica de los Datos	
Actor	Usuario
Propósitos	Ingresar a interfaz de gráficas
Resumen	Por medio de este caso de uso el actor acceder a la ventana de gráficos.
Flujo principal	1. El actor ingresa al componente entrenamiento. 2. El sistema despliega la ventana de entrenamiento 3. EL actor ingresa a la ventana gráficas 4. El sistema despliega la ventana graficas. 1a: El actor cancela el proceso El sistema regresa al módulo principal
Precondición	Se requiere haber preprocesados los datos de la señal de entrada del motor

La tercera división corresponde a la componente simulación, en la figura 21[33] se muestra los casos de uso de este componente.

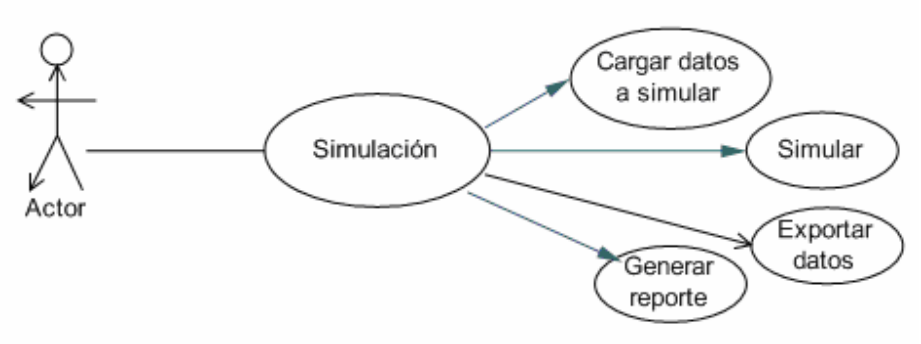


Figura 21. Casos de usos de la componente Simulación

Cargar Datos de Simulación	
Actor	Usuario
Propósitos	Cargar una datos de simulación de la red
Resumen	Por medio de este caso de uso el actor puede cargar los datos de simulación de la RNA desde un archivo válido
Flujo principal	1. El actor selecciona la ventana de simulación. 2. El actor selecciona la ubicación y el nombre del archivo. 3. El sistema carga los datos desde el archivo. 4. El sistema muestra los datos cargados El sistema cancela la operación y retorna al módulo principal. 1b: El actor cancela la operación. El sistema retorna al módulo principal.
Precondición	Ninguna

Simular la RNA	
Actor	Usuario
Propósitos	Simular una RNA de acuerdo a los datos de simulación suministrados.
Resumen	Por medio de este caso de uso el actor puede simular la RNA seleccionada con los datos especificados
Flujo principal	1. El actor selecciona la RNA. 2. El actor carga los datos de simulación. 3. El actor configura los parámetros de simulación. 4. El sistema simula la RNA de acuerdo a los datos y parámetros suministrados.
Precondición	Debe haber por lo menos una RNA creada, unos datos de simulación, y unos parámetros válidos

[33] Figura 21. Realizada por los autores

Exportar Datos	
Actor	Usuario
Propósitos	Exportar datos a excel
Resumen	Por medio de este caso de uso el actor puede generar un archivo de datos .xls
Flujo principal	1. El actor selecciona la ventana de simulación. 2. El sistema despliega la ventana de simulación 3. El Selecciona exportar datos 4. El sistema genera el archivo .xls 1a: El actor cancela la operación. El sistema regresa al módulo principal.
Precondición	Debe existir unos datos simulados

Generar Reporte	
Actor	Usuario
Propósitos	Producir un reporte, en base a opciones seleccionadas por el actor.
Resumen	Por medio de este caso de uso el actor puede generar un reporte de acuerdo a las opciones seleccionadas por él.
Flujo principal	1. El actor selecciona la RNA. 2. Entre las opciones disponibles, el actor escoge el contenido del reporte. 3. El actor suministra el nombre del reporte. 4. El sistema genera el reporte. 1a: El actor cancela la operación. El sistema regresa al módulo principal.
Precondición	Debe existir por lo menos una RNA creada.

3.4.2 DIAGRAMA DE CLASES

En UML, un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados para crear el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro. En la figura 22[34] se muestra el diagrama general de clases de NNEURALMOTOR 1.0

[34] Figura 22. Realizada por los autores

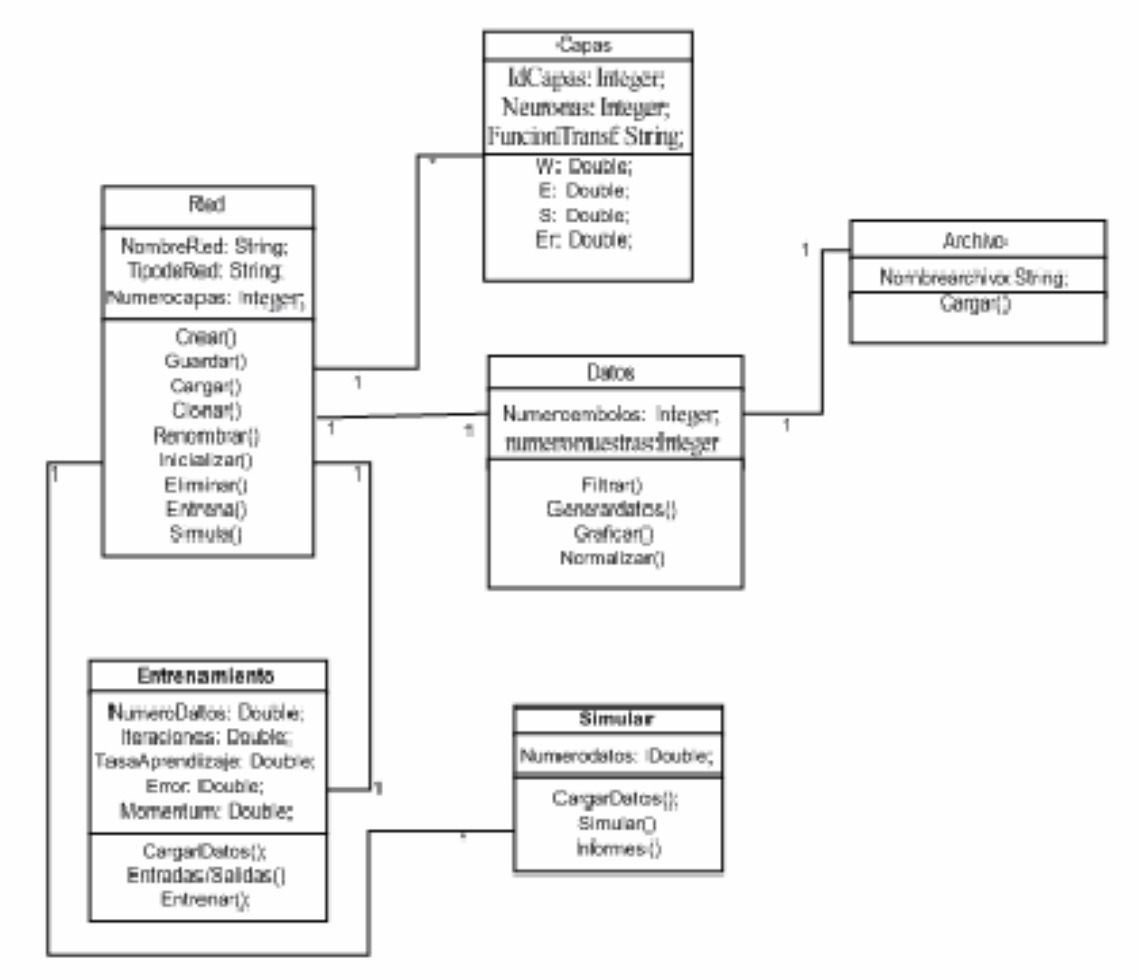


Figura 22. Diagrama de clases de NEURALMOTOR 1.0

En la figura 22, se aprecia el nombres, los atributos y las operaciones de las clase para cada una de las clases, según este esquema es fácil definir que la clase mas importante y mas grande es la llamada red, esta clase tiene relación directa con las demás clases excepto la clase archivos, la clase red tiene una relación 1 a muchos con la clase capas y simular, en cambio tiene relación 1 a 1 con la clase entrenamiento y datos. Por su parte la clase datos tiene relación 1 a 1 con la clase archivo y red. Las demás clases solo interactúan únicamente con la clase red.

3.4.3 DIAGRAMA DE SECUENCIAS

Los diagramas de secuencias son los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela en base a os casos de uso

En la figura 23[35] se muestra el diagrama de secuencia para los casos de uso crear red y entrenar red.

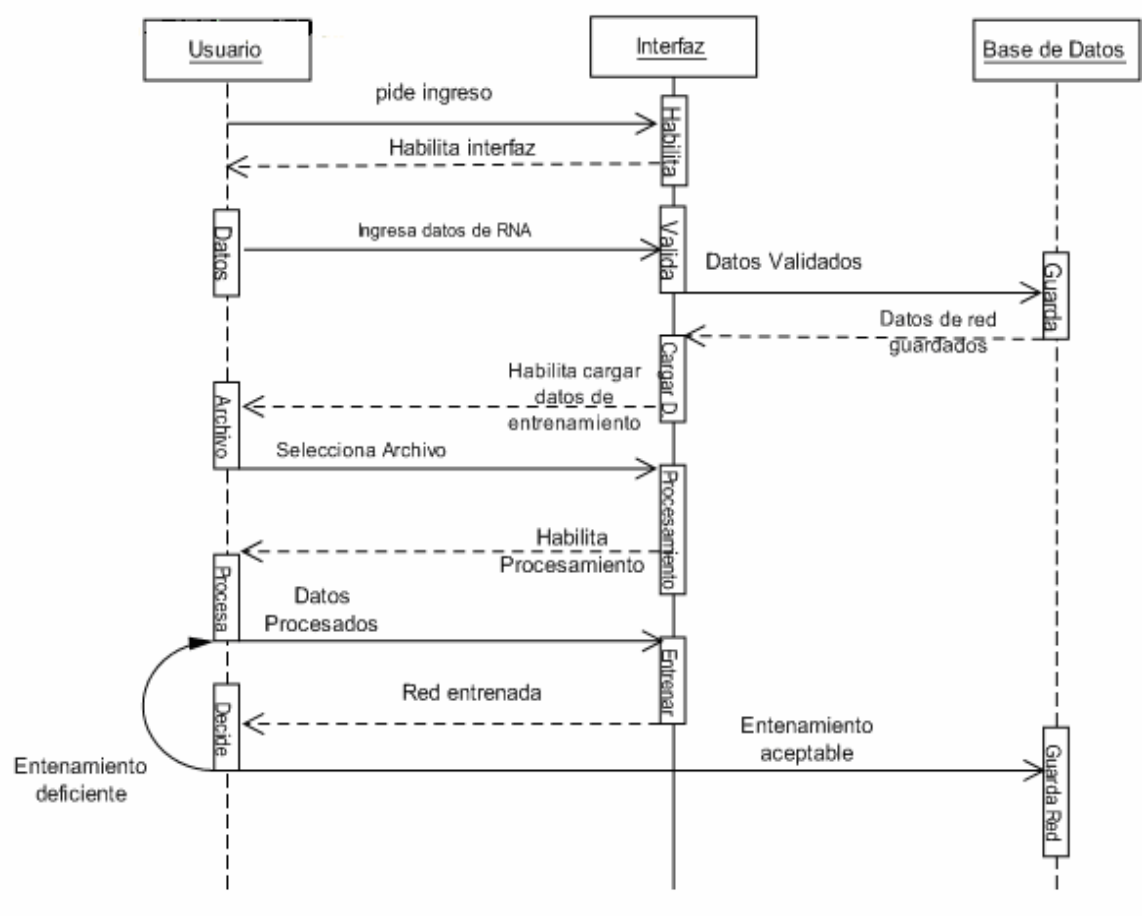


Figura 23. Diagrama de secuencia para crear y entrenar la RNA en NEURALMOTOR 1.0

✍ **Descripción del Diagrama.**

En la tabla 3[36] se muestra y enumera cada evento del diagrama de secuencia de creación y entrenamiento de una RNA, utilizando la herramienta software NEURALMOTOR 1.0, como se aprecia en este diagrama interviene tres actores, usuario, interfaz y base de datos. En su orden, se detallan los eventos que los actores generan y la respuesta del sistema. Tan solo se representan los eventos que *entran* al sistema, no los que *salen* del mismo. Tales eventos entrantes constituyen las operaciones del sistema, las operaciones que son activadas por alguna acción de un actor.

[35] Figura 23. Realizada por los autores
[36] Tabla 3. Realizada por los autores

Usuario	Interfaz	Base de datos
1. Pide ingreso		
	2. La interfaz habilita la el menú de administración	
3. ingresa los datos de los parámetros de la estructura de red		
	4. Valida los datos de red	
		5. Recibe y guarda los datos de los parámetros de red
	6. recibe y muestra los datos de los parámetros de red	
	7. Habilita Cargar datos de entrenamiento	
8. Selecciona el archivo de los datos de entrenamiento		
	9. Habilita el formulario de preprocesamiento del datos	
10. Realiza el preprocesamiento de los datos de entrenamiento		
	11. Recibe los datos de entrenamiento y entrena la red	
12. Recibe los resultados del proceso de entrenamiento de la red		
13. Decide si la red aceptada, sino es aceptada regresa 10.		
		14. Si el entrenamiento es aceptable guarda la red.

Tabla 3. Descripción del diagrama de secuencia de crear y entrenar la RNA

En la figura 24[37], se muestra el diagrama de secuencia del proceso de simulación de la RNA utilizando la herramienta software NEURALMOTOR 1.0, como se aprecia en este diagrama interviene tres actores, usuario, interfaz y base de datos. En su orden, se detallan los eventos que los actores generan y la respuesta del sistema. A diferencia del anterior diagrama de secuencia, en este se representan los eventos que *entran* al sistema y salen del sistema. Estos eventos constituyen las operaciones del sistema que son activadas por acción de un actor.

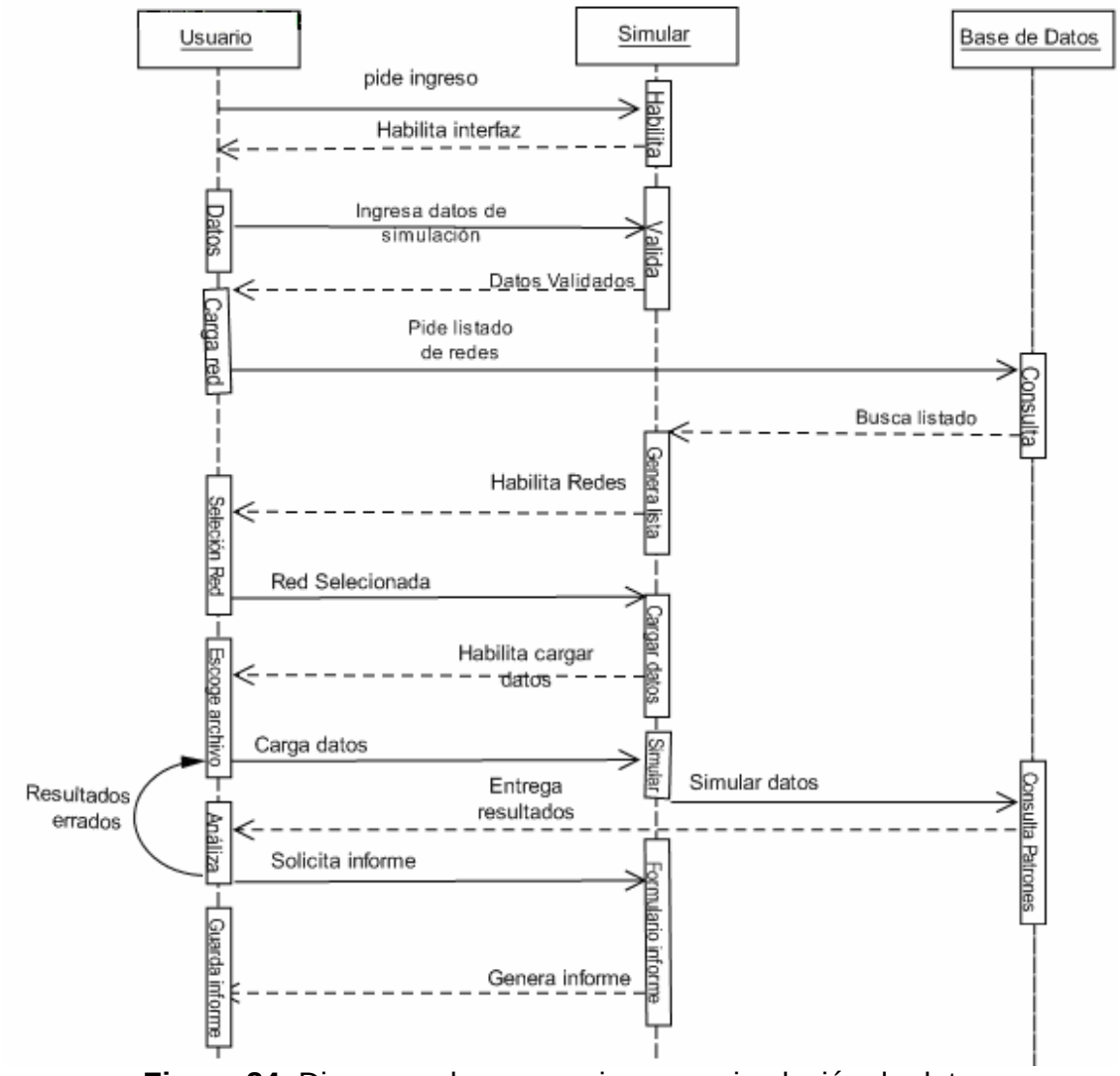


Figura 24. Diagrama de secuencias para simulación de datos.

A continuación en la tabla 4[38], se referencia los eventos que se presenta en el diagrama de secuencia para proceso de simulación de la RNA

[37] Figura 24. Realizada por los autores
[38] Tabla 4. Realizada por los autores

Usuario	Interfaz	Base de datos
1. Pide ingreso		
	2. La interfaz habilita la el menú de simulación	
3. ingresa los datos de simulación		
	4. Valida y muestra los datos de simulación.	
5. Pide listado de redes creadas o cargadas		
		6. Busca la lista de redes
	7. Recibe y muestra el listado de redes	
	8. Habilita el listado de redes	
9. Selecciona red de la lista		
	10. Habilita carga datos de simulación	
11. Selecciona el archivo de los datos		
	12. Valida y carga los datos de simulación	
	13. Habilita el formulario de simulación	
		14. consulta patones de la red seleccionada
15. Recibe los datos de la simulación		
16. Decide si genera informe, si no desea puede crear un informe termina o puede regresar a 11		
	17. Si decide generar informe, se genera el informe.	
18. guarda el informe generado		

Tabla 4. Descripción del diagrama de secuencia de simulación de la RNA

CAPITULO 4

METODOLOGÍA

4.1 PLANTEAMIENTO DE LA METODOLOGÍA

Se debe plantear una metodología consecuente, para evitar que durante el trabajo investigativo y de construcción de la herramienta software, se caiga en errores que no facilitan el normal desarrollo del proceso de construcción, aunque en la obtención de las estructuras de red se utiliza el procedimiento de ensayos y error, se requiere diseñar una metodología de prueba, que permita avanzar en el propósito de encontrar las variables que afectan el proceso de aprendizaje de la RNA.
Debido a la complejidad en el tratamiento de los datos en éste caso de identificación de sistemas dinámicos no lineal mediante la utilización de la RNA, se debe fijar un método que divida todo el proceso en etapas consecutivas que abarque la totalidad del mismo. En cada una de estas etapas se debe determinar claramente los pasos a seguir y los indicadores que señalen cuando se debe proseguir a la siguiente etapa y cuando se debe retornar a una anterior
Ahora, debemos definir en cuantas etapas se debe dividir el proceso, sin incurrir en errores, pues si se divide en etapas muy pequeñas se corre el riesgo de perder de vista el objetivo del proceso global, y, si se divide, en etapas demasiado grandes se pueden obtener resultados poco representativos. Después de analizar, detenidamente, el caso de estudio se diseñó una metodología que contiene las siguientes etapas.

- Adquisición y análisis de los datos
- Preprocesamiento de los datos
- Definir el número y el rango de entradas
- Definir el número y rango de las salidas
- Realizar normalización de las entradas/salidas de ser necesario
- Elección del tipo de redes a implementar
- Búsqueda de la arquitectura de red
- Entrenamiento de las redes halladas
- Prueba y elección de las redes
- Realizar una afinación de redes en base a los resultados.
- Reporte de resultados

Esta metodología es intuitiva y permite un hallazgo en varias etapas, cubriendo las fases más importantes de nuestro proceso de investigación y abarca desde la entrada de los datos hasta el respectivo reporte de resultados. En la figura 25[39], se muestra las etapas que componen de la metodología implementada en el desarrollo de NEURALMOTOR 1.0

[39] Figura 25, realizada por los autores

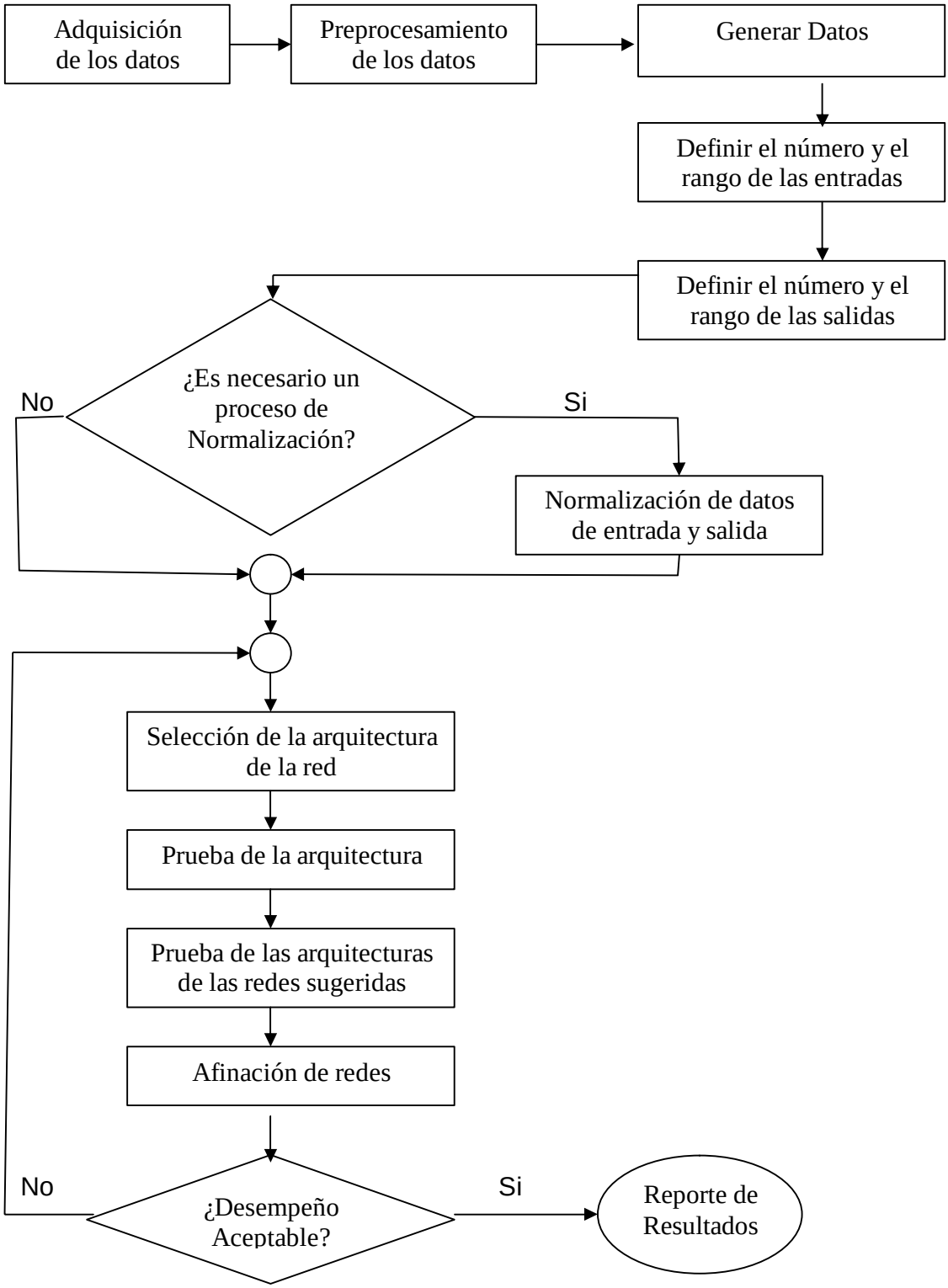


Figura 25. Diagrama de la Metodología

Con la construcción de la herramienta software **NEURALMOTOR 1.0** se obtiene un aplicativo flexible que nos permite generar distintas arquitecturas dentro de las redes de tipo backpropagation y de función de base radial, y se puede detallar el proceso para la consecución de una red adecuada, sino que también es útil para todas aquellas aplicaciones de clasificación y predicción en general.

4.1.1 ADQUISICIÓN Y ANÁLISIS DE LOS DATOS DE ENTRADA

La descripción del ejercicio de identificación de sistemas dinámicos No lineales o caso de investigación, inicia cuando se eligen los datos de un determinado motor. Para esta investigación inicialmente hemos tomado los datos de un motor de cuatro émbolos en buen estado, se recibió seis archivo de texto, cada uno de los cuales describen el comportamiento de este motor en escenarios diferentes, en cada escenario se varia la velocidad y las presiones en los cilindros, así se obtiene seis casos específicos con los cuales s puede realizar las aplicaciones de RNA.

Estos datos provienen de un laboratorio especializado, donde se cuenta con un sensor capaz de capturar los datos de la presión generada en los émbolos del motor de combustión interna, como se muestra en la figura 26[40], las mediciones hechas son consignadas en un archivo de texto que esta conformado por dos columnas, la primera representa el ángulo de giro del cigüeñal y la segunda representa la presión generada por el cilindro precisamente en el instante en que la biela describe el respectivo ángulo.

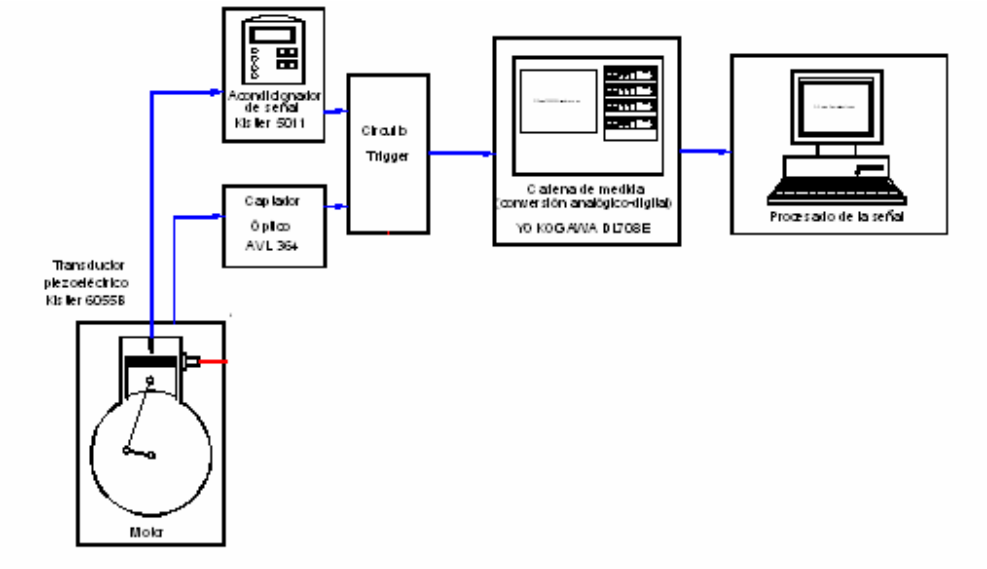


Figura 26. *Cadena de medida utilizada en el estudio experimental*

[40] Figura 26. Tomada de la tesis Diagnósticos de fallos mediante la utilización de información incompleta e incierta. Aplicación a motores diesel, Chacón Velasco, Jorge Luís, UPV – DMMT, Valencia ,2001

La Figura 26 muestra la instalación experimental y la cadena de medida utilizada para el registro y procesado de las señales del estudio experimental[41] en una de las salas de ensayos del Departamento de Motores y Máquinas Térmicas de la Universidad Politécnica de Valencia.

El dinamómetro eléctrico instalado en la sala de ensayo es un dinamómetro asíncrono de la marca *AVL - ELIN EBG Elektronik,* modelo *APA 202/E.*

El control del freno se puede realizar desde el ordenador de control y de trabajo de la sala o bien desde el panel de *electrónica de control y monitorizado* conocida con el nombre de *EMCON*. Este elemento de control electrónico esta situado en la mesa de control del operario encargado de la sala, con este dispositivo el operario controla de forma libre mediante los puntos de funcionamiento del motor y freno requeridos a través de su panel de control.

La *EMCON 300* es una unidad digital de control electrónico para el control manual y automático del dinamómetro en los ensayos, está comunicado al sistema *PUMA* mediante el *FEM-D* y al ordenador mediante el puerto *PC LINK*. Sus funciones principales son las siguientes:

- Selección del modo de control.
- Control de los valores demandados
- Control del sistema.
- Control y regulación del motor y dinamómetro*.*

Para realizar y obtener el control y adquisición de datos general de toda la instalación, se ha usado un sistema muy avanzado, este sistema se denomina *PUMA 5 COMPACT* de la firma austríaca *AVL*, y que permite realizar un control y la adquisición de datos de una forma totalmente automática.

Una particularidad general del sistema es la comunicación entre el usuario y la instalación, llevada a cabo mediante un PC de control general que incorpora un software programado por *AVL*, este software permite visualizar por pantalla el estado general de la instalación y modificar los parámetros regulados por los sistemas auxiliares que se comunican con el *PUMA*.

Además de controlar todos y cada unos de los sistemas de control y regulación del dinamómetro, balanza, throttle, etc., permite además, el control del encendido y de la electrónica propia del motor a la hora de arrancar, parar y modificar en *cierta medida* la cartografía del motor.

[41] Descripción del estudio experimental, tomado de la tesis Diagnósticos de fallos mediante la utilización de información incompleta e incierta. Aplicación a motores diesel, Chacón Velasco, Jorge Luís, UPV – DMMT, Valencia ,2001

4.1.1.1 CARACTERÍSTICAS DEL MOTOR EMPLEADO EN LOS ENSAYOS

Para el estudio experimental del modelo expuesto anteriormente se utilizo un motor Diesel moderno PSA(Peugeot–Citroen), modelo denominado por su fabricante *PSA DW12 TED4*, motor turboalimentado con turbina de geometría variable controlada por un sistema de vacío, con sistema de inyección tipo Common – Rail, gestionado por un moderno sistema de control electrónico BOSCH el cual controla el funcionamiento general del motor. Las principales características se describen en la Tabla 5[42].

Descripción	Característica
Marca	PSA
Modelo	DW12 TED4
N º de cilindros	4
Cilindrada	2200 cm^3
Distribución	Doble árbol de levas en cabeza
Válvulas	16
Sistema sobrealimentación	Turbocompresor Garret - TGV
Diámetro (mm)	85
Carrera (mm)	96
Sistema de inyección	Bosch common rail
Control de inyección	Electrónica Bosch
Secuencia de encendido	1-3-4-2
Par máximo / rpm	316 N- m / 2250
Potencia nominal máxima / rpm	136 CV / 4000
Sistemas adicionales	EGR- Swirl variable- Intercooler

Tabla 5. Características del motor Diesel PSA DW12 TED4

[42] Tabla 5, tomado de la tesis Diagnósticos de fallos mediante la utilización de información incompleta e incierta. Aplicación a motores diesel, Chacón Velasco, Jorge Luís, UPV – DMMT, Valencia ,2001

Se conoce que el funcionamiento de los motores se ajustan al comportamiento de una función periódica, entonces cuando se realice el gráfico de cada escenario se generan varios picos, cada uno de los estos picos representa la presión máxima generada por un émbolo, de antemano sabemos que este motor es de cuatro cilindros, por consiguiente el primer, el quinto, el noveno (etc.) pico del gráfico corresponde al mismo cilindro, la misma correspondencia se tiene con el segundo, sexto, décimo (etc.), así sucesivamente para el tercero y cuarto émbolo, en resumen esta gran cantidad de datos es una representación continua de un motor de cuatro cilindros.

El personal experto en el análisis de motores de combustión interna, conocen que cada motor tienen un orden de encendido, que dicho de otra manera es el orden en que se produce la combustión en cada cilindro del motor, para este motor especifico se conoce que el orden de encendido es; el primer pico corresponde al émbolo 1, el segundo pico corresponde al émbolo 3, el tercer pico corresponde al émbolo 4 y el cuarto pico corresponde al émbolo 2, el quinto pico del grafico correspondería al émbolo 1, y se repetiría sucesivamente, para el quinto, sexto, séptimo (etc.). En la siguiente sección profundizaremos en la explicación de la correspondencia de estos picos basados en las gráficas.

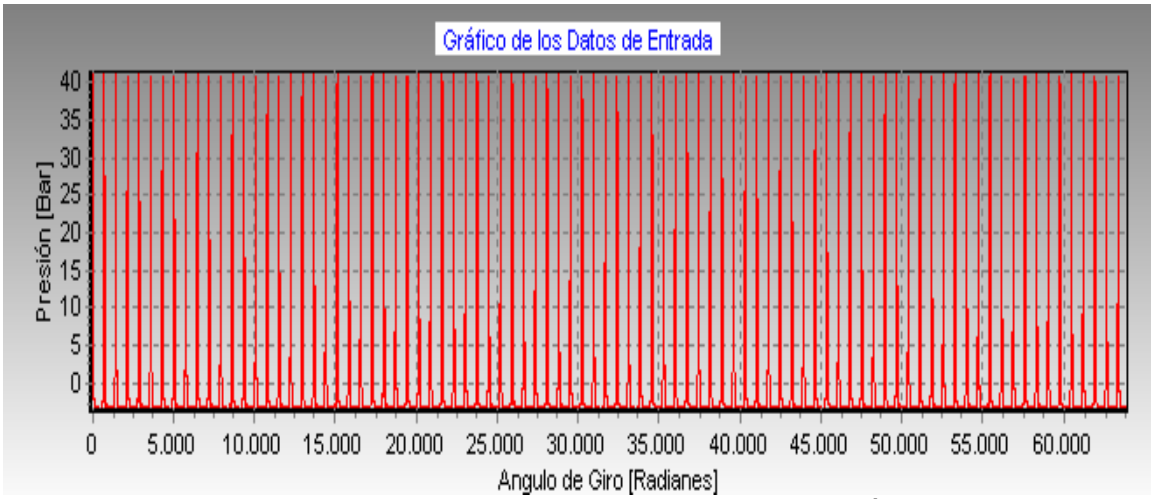
4.1.2 PREPROCESAMIENTO DE LOS DATOS

Se inicia el preprocesamiento de los datos tomando un archivo tipo texto, es indispensable aclarar que el preprocesamiento se debe hacer por separado para cada archivo, puesto que cada uno de estos guarda los datos de un escenario diferente.

El proceso continua, se asigna al vector X los datos de la primera columna, que contiene los datos del ángulo de giro del cigüeñal, y la segunda columna se asigna al vector Y, en donde se guardan los datos de la presión del émbolo para sus respectivo ángulo de giro. Se sabe que no todos los datos guardados en los vectores X y Y, son representativos para nuestro proceso de preprocesamiento, puesto que muchos de estos datos corresponden al ruido captado en la medición, en la figura 27[43] se muestra la gráfica de los datos de la señal de entrada del motor, donde se grafican todos puntos cargados desde el archivo de datos.

El principal problema para el tratamiento de los vectores X y Y, es su tamaño, pues son muchos datos para el entrenamiento de la red y lógicamente esto haría que nuestra herramienta software se sature y no se obtenga respuesta.

[43] Figura 27, realizada por los autores utilizando la herramienta software NEURALMOTOR 1.0



Grafica 27. Señal de entrada del motor de combustión interna

Basándose en la información técnica del motor en estudio, es posible saber que un motor de cuatro (4) cilindros y su orden de encendido es 1- 3 -4 -2. En la tabla 6[44] se ilustra la correspondencia de los picos según el orden de encendido del motor

Pico	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
Pico	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
Pico	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
Pico	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2

Tabla 6. Correspondencia de picos según el o*rden de encendido*

Basándose en la tabla 6, se aprecia que la correspondencia para cada valor se repite consecutivamente cada cuatro picos, entonces se diseño una estrategia para reducir el tamaño del vector X y Y, el método que se propuso para reducir esta cantidad de datos del vector de la señal de entrada, es promediar los picos de en factor del numero de cilindros del motor, en este caso seria de cuatro (4), precisamente por este el numero de cilindros del motor, se debe tener presente que si estuviésemos trabajando con un motor de seis (6) cilindros, 6 seria el factor con el cual dividiríamos los datos del vector de la entradas Y.

De la tabla 6, se identifico que el valor del pico 1 tiene correspondencia con el valor del pico 5, 9, 13 etc. así sucesivamente hasta el pico 61, pues todos estos son datos representativo del émbolo 1, el mismo procedimiento hacemos para el

[44] Tabla 6, realizada por los autores

émbolo 3, 4 y 2. Ahora el siguiente paso a seguir es ir promediando los datos correspondiente a cada uno de los cilindros, así se saca la media para cada émbolo. Este proceso termina hasta cuando se consiga promediar los datos que cubran el 70% de la totalidad de los datos del vector de entrada Y, en este caso se cumple cuando se promedian los datos de los picos correspondientes el émbolo 2, que es el último émbolo en el orden de encendido, y se llegue a promediar el pico 44.

Después de realizar el promedio del 70% del vector Y, se consigue N muestras promediadas del vector Y, que se asignan al vector B. Seguidamente tomamos los N datos del vector X y se asignan al vector A, esto se puede hacer, ya que el movimiento que describen los émbolos corresponden a una función periódica, de igual formas esta cantidad de datos no es aceptable para el entrenamiento de la red, se necesita reducir los datos de los nuevos vectores A y B.

Hasta el momento hemos reducido sustancialmente la cantidad de valores del vector de la señal de entrada, pero aun sigue siendo sumamente grande para poder intentar entrenar la red, debemos seguir procesando este vector hasta reducirlo a una cantidad razonable de datos de entrenamiento, pero que sigan representando a la 70% de la señal de entrada.

Se continua el proceso de preprocesamiento de los datos de entrada, hábilmente se decidió aplicar la transformada de Fourier a esta este nuevo vector de la señal B, pasando del espectro del tiempo al espectro de la frecuencia, con este procedimiento, se puede atenuar el ruido de la señal de entrada, hallado los armónicos principales del vector B. Ahora debemos detectar el rango de armónicos representativos de los datos del vector B, ósea, que tomamos el rango de las frecuencias bajas, y obtenemos un nuevo vector C, entonces C se guardan los datos representativo del 70% de los datos de entrada.

Pero C es un vector que contiene números complejos, que indiscutidamente nos complica el entrenamiento de la red, el siguiente paso es aplicar la transformada inversa de Fourier para regresar al espectro del tiempo y estos datos se asignan al vector D. ahora en D si tenemos un numero razonable de datos que puede ser utilizado en el entrenamiento de la red.

Pero realizando un minucioso análisis del vector D, lo que se observa, es que no seria practico introducir estos datos de D, pues algunos de éstos datos corresponden al ruido de la señal de entrada, entonces decidimos utilizar para el entrenamiento de la red la presión máxima generada en cada uno de los cilindros.

En cada escenario se puede obtener dos tipos de respuesta en la salida para cada émbolo, una será que este funcionando correctamente y otra que este experimentando alguna falla, por consiguiente, la RNA estará en condiciones de identificar el estado del motor basando en las salidas generadas a partir del

análisis del valor máximo de presión en cada émbolo, es suficiente que alguno de los cuatro cilindros este fallando para dictaminar que el motor no esta funcionando correctamente. Por recomendación de los técnicos, se conoce que el valor de la presión máxima en el émbolo no puede ser menor que el 97% de la presión máxima del émbolo en estado normal, entonces se maneja un rango del 3% dentro del cual se admite que el émbolo esta funcionando correctamente, si baja del 97% se dictamina que el émbolo esta fallando.

Con todo este proceso reducimos el 70% de los datos de la señal de entrada del motor, a solo cuatro datos, el numero de cilindros del motor, entonces se tiene 2 a la N combinaciones posibles en los datos de entrada para el entrenamiento de la red, ósea, 16 combinaciones se pueden presentar en los datos de entrenamiento.

Al restante 30% de los datos de señal de entrada, se les realiza el mismo procedimiento de preprocesamiento que se le realizo al 70%, e igualmente, se toman los valores máximos de la presión en el émbolo para probar la eficacia de la RNA.

4.1.3 DEFINICIÓN DEL NÚMERO Y EL RANGO DE LAS ENTRADAS

Como este motor es de cuatro cilindros, el número de entradas para red será de cuatro, el rango de las entradas se determina para cada caso específico, pues los valores varían de un escenario a otro. A continuación se exponen las principales recomendaciones a tener en cuenta para determinar las variables de entrada y salida:

- Realizar una definición previa de las variables, (continuas y discretas).
- Hay que evitar el uso de valores continuos para representar conceptos simbólicos, por ejemplo la representación de los animales.
- No representar los meses del año del 1 al 12, categorizar de otra manera.
- No confundir las entradas continuas con las entradas discretas, estado civil es discreto (1 o 0), mientras que la temperatura es continua (0, 10, 15.5, 30,40.2…).
- No mezclar escalas (kilogramos con toneladas, metros con kilómetros, años con meses…).
- Evitar las variables con altas variaciones (máximos y mínimos), si es necesario normalizarlas.
- Usar como patrones de entrada los diferentes períodos en que puedan afectar la salida (si es por meses, el promedio de cada mes y no los datos de cada mes).
- Entre mayor sea el número de entradas, más casos de entrenamiento requiere, lo que puede conducir a arquitecturas complejas, altos costos computacionales y difícil interpretación de resultados.

4.1.4 DEFINIR EL NÚMERO Y RANGO DE LAS SALIDAS

El número de las salidas también corresponden al numero de cilindros del motor, puesto que la red se debe pronunciar puntualmente sobre cada émbolo, basándose en los patrones de entrenamiento va examinado los datos de cada émbolo, por consiguiente tendremos cuatro salidas en la red, una para cada émbolo, no obstante, el correspondiente rango de la salida de la red será el valor de uno (1) cuando el émbolo este trabajando correctamente y de cero (0) si esta presentado alguna falla. En la figura 28[45] se muestra el procedimiento de entrada y salidas de los datos de la RNA.

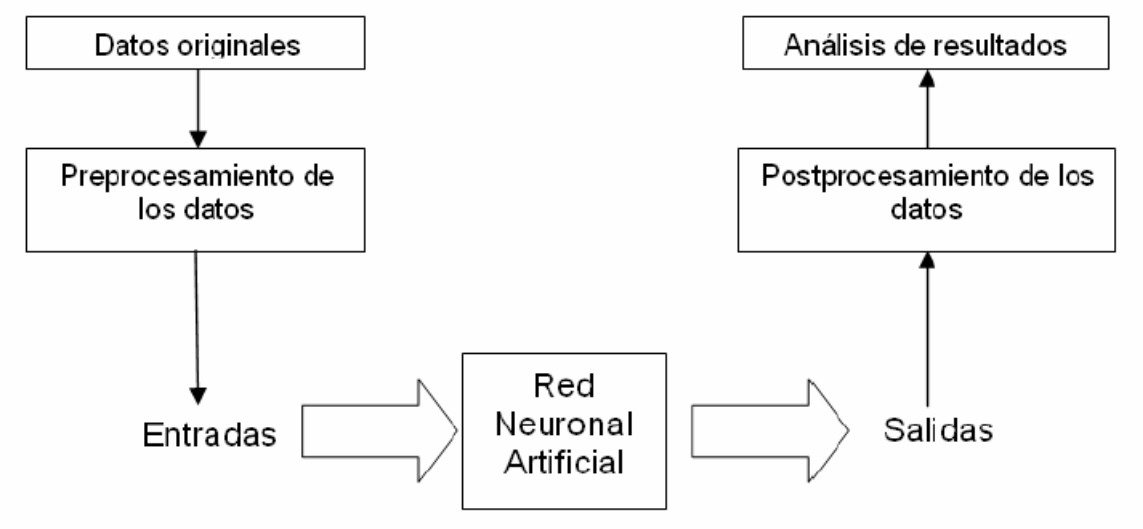


Figura 28. *Proceso de los datos para las Entradas/Salidas de una RNA*

4.1.5 REALIZAR LA NORMALIZACIÓN DE LAS ENTRADAS Y SALIDAS

Es recomendable realizar la normalización de los datos de entrada y salida de red puesto que con esta opción que ofrece **NEURALMOTOR 1.0** se pueden manipular los datos para que sean mas manejables durante el proceso de aprendizaje de la red, debido a que los valores de los datos de entrada son superiores a 10, la redes no realizan un buen aprendizaje, en este caso de estudio es recomendable realizar la normalización de los datos de entrada y salida para mejorar el proceso de aprendizaje de la RNA.

[45] Figura 28, realizada por los autores

4.1.6 ELECCIÓN DEL TIPO DE RNA A UTILIZAR

Después de analizar los datos y estudiar el estado del arte, llegamos a la conclusión que la metodología recomendada para abordar y solucionar estos problemas complejos de sistemas dinámicos no lineales a partir de sus datos reales, sería la utilización de las redes neuronales artificiales.

Las (RNA), ofrecen un enfoque recursivo, que hace que la complejidad de un sistema dinámico no lineal se reduzca debido al hábil tratamiento de los datos de entrada imaginado por sus creadores. Esto acrecentó nuestro interés en utilizar este tipo de modelos abstractos en el desarrollo de la investigación, pues estos modelos, permiten un avance significativo en el propósito de encontrar respuestas satisfactorias a nuestro caso en estudio y ofrecen ventajas en el manejo de los recursos.

Profundizando en el mundo de las RNA, es posible asumir que los primeros algoritmos de entrenamiento de RNA fueron diseñados para redes de una sola capa, con la desventaja que estas sólo resolvían problemas linealmente separables, motivo por el cual surgieron las RNA multicapa para resolver esta dificultad.

Entre las diferentes opciones de RNAs que se pueden utilizar en nuestro ejercicio de investigación, descartamos algunas por inadecuadas y hemos decidido aplicar sólo dos de ellas, pues se asume que las RNA perceptrón multicapa con algoritmo de retropropagación (Backpropagation) y las Funciones de Base Radial (RBF) son ideales para conseguir los objetivos propuestos en la investigación.

4.1.6.1 FUNDAMENTOS PARA EMPLEAR LA RED DE PERCEPTRON CON ALGORITMO BACKPROPAGATION

A continuación se exponen algunas razones que fundamentaron nuestra decisión de emplear la red de perceptron multicapa con algoritmo Backpropagation.

- Es una Red de entrenamiento no supervisado, ósea, solo necesita datos de entrada y salidas de un proceso de aprendizaje
- Utiliza el aprendizaje supervisado, en donde los pesos se adaptan de acuerdo con reglas de aprendizaje no supervisadas
- Puede aprender en ausencia de muestreo
- Al recibir un estimulo, este se propaga a través de la red hasta hallar una señal de salida, la cual se compara con la salida deseada y se calcula una señal de error que se propaga hacia atrás, otorgándole a cada neurona una fracción de la señal de error, igual al aporte entregado a la señal de salida. Este proceso se repite hasta conseguir que cada neurona reciba una señal de salida equivalente a su contribución relativa al error total. Así se actualizan los pesos de conexión de cada neurona hasta que la red

converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento.

- Aprovecha su naturaleza paralela, reduce los tiempos requeridos por un procesador para determinar la correspondencia de patrones.
- Su destreza para solucionar problemas de reconocimiento de patrones en entornos ruidosos en corto tiempo.
- Su Auto-adaptación para aprender de la relación entre un conjunto de patrones adquiridos y aplicarlos a nuevos patrones de entrada.
- La capacidad para identificar características de una entrada arbitraria que se asemeje a patrones conocidos, excluyendo la señal de ruido.
- A medida que se entrena la red, las neuronas de capas intermedias se auto-organizan de tal modo que aprenden a reconocer características del espacio total de entrada. Cuando se ingresa un patrón arbitrario de entrada que contenga ruido o este incompleto, las neuronas de las capas ocultas de la red responderán activamente si la nueva entrada contiene un patrón que se asemeje a aquellas características que las neuronas individuales hayan aprendido a reconocer en su entrenamiento. Y en el caso contrario, cuando no haya reconocimiento de características, las unidades de las capas ocultas se abstendrán de activarse, lo que indica que estas entradas son desconocidas para la red, y por lo tanto la red debe clasificarlas de acuerdo a las características que compartan con los ejemplos de entrenamiento.

4.1.6.2 FUNDAMENTOS PARA EMPLEAR LA RED DE LA FUNCIÓN DE BASE RADIAL (RBFN)

A continuación se exponen algunas razones que fundamentaron nuestra decisión de utilizar las RBF:

- Utiliza el método de aprendizaje hibrido, fase no supervisada en la que determina los centros y amplitudes de las neuronas de la capa oculta y fase supervisada, donde se determina los pesos y umbrales de la capa de salida.
- Aprendizaje rápido
- Aplicaciones en tiempo real
- Ideal para el tratamiento no lineal, debido a que cada neurona de la capa oculta de la red RBF construye una aproximación local y no lineal en una determinada región de dicho espacio.
- aproximan relaciones complejas mediante una colección de aproximaciones locales menos complejas, dividiendo el problema en varios subproblemas menos complejos.
- Clasifica el espacio de entrada en diferentes clases. El representante de cada clase será el centro de la función de base radial y la desviación vendrá dada por la amplitud de cada clase. El número de clases es el número de neuronas ocultas en la red de base radial.

- ✍ Los patrones de entrada se asocian a los centro de las clases más cercanas, se promedian los patrones de cada clase y hallan la posición de los nuevos centros, hasta conseguir que la nueva posición de centro sea igual a la anterior. Por lo tanto, el proceso de aprendizaje está guiado por la minimización de una función error computada en la salida de la red, utilizando mínimos cuadrados se logra que la convergencia sea bastante rápida, consiguiendo una solución en un número pequeño de iteraciones o ciclos de aprendizaje.

4.1.7 BÚSQUEDA DE LA ARQUITECTURA DE LA RNA

Se tienen dos opciones de RNA, PMC y RBF, ambas son paralelas, es decir, lo que se implementa en backpropagation, se implementa en base radial, y viceversa. Aprovechando esta situación, es posible construir y probar modelos simultáneamente en ambos tipos de redes. Sometiéndolas a los posteriores análisis para observar cuál se comporta mejor.

Tanto el PMC como la de función de base radial presentan sus ventajas y desventajas, las principales son ilustradas en la Tabla 7[46]. Por esta razón depende de la experiencia del usuario, así como, de la cantidad de datos disponibles y la calidad de los mismos.

CRITERIO	PMC CON BACKPROPAGATION	BASE RADIAL
Tamaño	Menor o igual que la de función de base radial	Igual o mayor a la backpropagation
Costo computacional	Tiene menor costo computacional que la de base radial	Requiere una mayor cantidad de cálculos, al ser de mayor tamaño
Capas	Desde dos capas en adelante(aunque se aconseja que no sea mayor a 4)	Siempre 3, una capa de entrada, una oculta y una de salida
Aprendizaje	Supervisado	Hibrido (no supervisado en la capa oculta y supervisado en la capa de salida)
Entrenamiento	Hay que cuidar que no se sobre-ajuste la red	No hay problemas de sobre-ajuste

Tabla 7. Comparación de PMC con RBF

[46] Tabla 7, realizada por los autores

Para hallar las arquitecturas de red, que pronostiquen resultados acertados para nuestra investigación, utilizamos la función de *generar red*, la herramienta software se encarga de generar automáticamente red, este componente facilita la labor, puesto que procedemos a entrenar la red generada, así se va identificando cuales estructuras de red son mas efectivas para el tratamiento de los datos de entrada, este proceso se puede aplicar para cada topología, sea PMC o RBF.

4.1.7.1 RED PMC CON ALGORITMO BACKPROPAGATION

Se dijo anteriormente la red con el error más bajo durante el entrenamiento no siempre es la que mejor se desempeña, así mismo, se puede decir de las redes de topología Backpropagation, que no siempre la red más grande es la que mejor aprende. La determinación de las neuronas adecuadas y de las capas adecuadas debe estar de acuerdo a estudios previos del desempeño que han tenido cada una de las arquitecturas, por esta razón, es recomendable, en el caso de las redes Backpropagation, se construyan diversos prototipos de redes con algunos criterios de guía en su construcción:

- Asumir que el número de neuronas en la capa oculta sean menos de la mitad de la suma del número de entradas más el de las salidas.
- Con tres capas las redes backpropagation aproximan la mayoría de las funciones, con cuatro se estima que las aproxima todas, por lo tanto no es aconsejable que utilice más de cuatro capas.
- Ya se sabe que el número de neuronas en la capa de entrada es igual a las entradas, y que el número de neuronas en la capa de salida es igual a las salidas, por lo tanto puede tener en cuenta la regla $N > W/E$, donde N es la cantidad de ejemplos de entrenamiento, W la cantidad de nodos ocultos y E el error deseado.
- La tasa de aprendizaje varia comúnmente entre 0.01 y 10, pero la más usada es 0.1

4.1.7.2 RED DE BASE RADIAL (RBF)

Para la RNA de función de base radial (RBF) los criterios varían. Este tipo de red no sufre de sobreentrenamiento, pero tiene mayor costo computacional, requiere mayor número de neuronas que las Backpropagation para realizar la misma labor, sin embargo, su aprendizaje es más rápido.

Se presenta la “maldición de la *dimensionalidad*” que indica que el tamaño de la red aumenta exponencialmente con respecto a los casos que debe resolver. Cuando sea posible se recomienda colocar tantas neuronas en la capa oculta como ejemplos de entrenamiento hayan, sin embargo, debido a que la cantidad de ejemplos puede llegar a ser considerable se debe tener cuidado al construir una red de tamaño excesivo, por que su entrenamiento puede llegar a ser inviable.

4.1.8 ENTRENAMIENTO DE LAS REDES SELECCIONADAS

Para cada una de las redes generadas, procedemos a entrenar las redes con los datos de la señal de entrada, que son los datos de entrenamiento para todas las RNA generadas, se recomienda normalizar los datos antes del entrenamiento.

4.1.9 PRUEBA Y ELECCIÓN DE LAS RNA

De acuerdo a lo sugerido por la literatura actual, se conoce que aquellas redes que su error de convergencia es mayor del 5%, es poco probable que ofrezcan resultado eficientes, pero no se debe pensar que aquella red que consiguen el valor de error mas bajos es la que mejor se desempeña, para despejar la duda de cual de las redes seleccionadas tienen mas eficacia se debe simular cada una con los datos de prueba. Después de haber entrenado todas las redes generadas bajo las mismas condiciones, proseguimos con la prueba de cada una de ellas, para esto se toma el 30% de los datos de entrada de señal dispuesto para la prueba y se le hace el preprocesamiento de los datos, se cargan a **NEURALMOTOR 1.0** y posteriormente se normalizan, esto datos se simulan y de acuerdo a los resultado que entregue cada red, elegimos cual de ellas entrega resultados mas ajustado a al realidad.

4.1.10 REALIZAR UNA AFINACIÓN DE LAS REDES EN BASE A LOS RESULTADOS

Teniendo preseleccionadas un número razonable de redes, usualmente 10 redes entre uno u otro tipo y, teniendo claro, cuáles requisitos mínimos debe cumplir la red o redes ganadoras, se comienza el proceso de afinación, que consiste en realizar variaciones relativamente pequeñas alrededor de los parámetros de cada red, número de neuronas, taza de aprendizaje, número de iteraciones, con el fin de encontrar una que se ajuste aún mejor que las que ya existían. Este proceso es iterativo, y termina cuando se cumplan los requisitos propuestos para la red.

4.1.11 REPORTE DE RESULTADOS

Se toman los resultados de la simulación de los datos de prueba y se genera los reportes de resultados de acuerdo a los indicadores requeridos por el usuario, **NEURALMOTOR 1.0** identifica claramente cual émbolo del motor esta fallando y con esta etapa se termina la metodología diseñada para nuestra investigación.

4.2 APLICACIÓN DE LA METODOLOGÍA

4.2.1 ESCENARIO PC1_90 (PRESION EN EL CILINDRO A 900 RPM)

4.2.1.1 ADQUISICIÓN Y ANÁLISIS DE LOS DATOS DE ENTRADA

Para este caso de estudio se reciben un archivo de texto con dos columnas, en cada columnas se guarda la información de aproximadamente sesenta y cuatro mil (64.000) muestras, entonces tenemos un archivo de dos columnas por 32.000 filas. Estos datos serán similares cuando el cilindro este funcionando correctamente y se observa una diferencia sustancial cuando el cilindro este experimentando fallas.

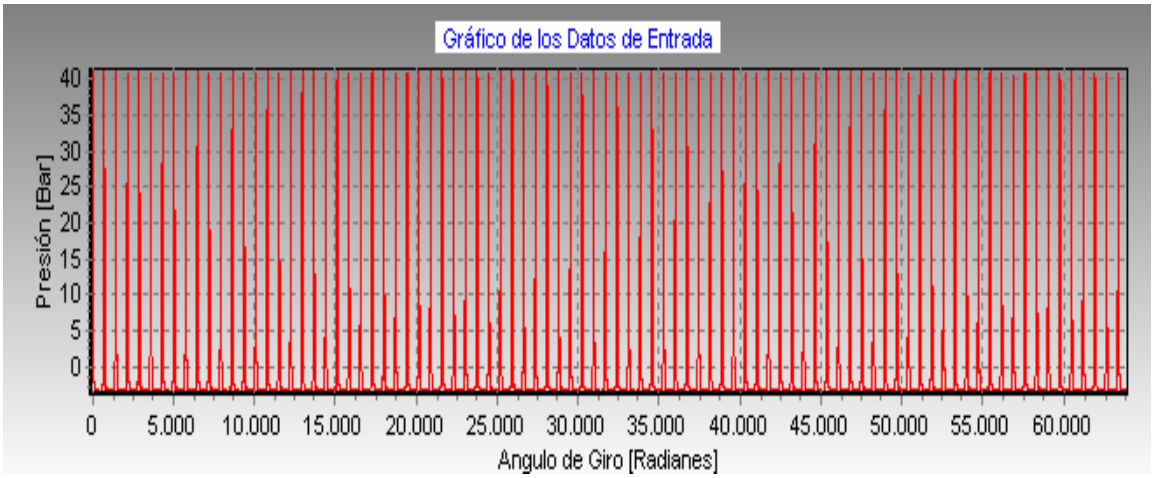


Figura 29. Grafico de la señal de Pc1_90

Para hacer un análisis de los datos basado en la gráfica 29[47] es bastante difícil obtener información relevante de este caso; por lo tanto recurrimos a software matlab, y creamos una aplicación que nos permita disminuir la complejidad de este caso graficando solo las 10.000 primeras muestras de esta señal como se puede apreciar en la figura 30.

En la figura 30[48], se aprecia que este caso corresponde a una grafico de una función periódica, en el que se aprecia varios picos, cada uno de los estos picos representa las presión máxima generada por un émbolo, de antemano sabemos que este motor es de cuatro cilindros, por lo consiguiente el primer, el quinto, el noveno (etc.) pico del grafico corresponde al mismo cilindro, la misma correspondencia se tiene con el segundo, sexto, décimo (etc.), así sucesivamente para el tercero y cuarto émbolo, en resumen esta gran cantidad de datos es una representación continua de un motor de cuatro cilindros.

[47] Figura 29, realizada por los autores con NEURALMOTOR 1.0
[48] Figura 30, realizada por los autores usando Matllab 6.5

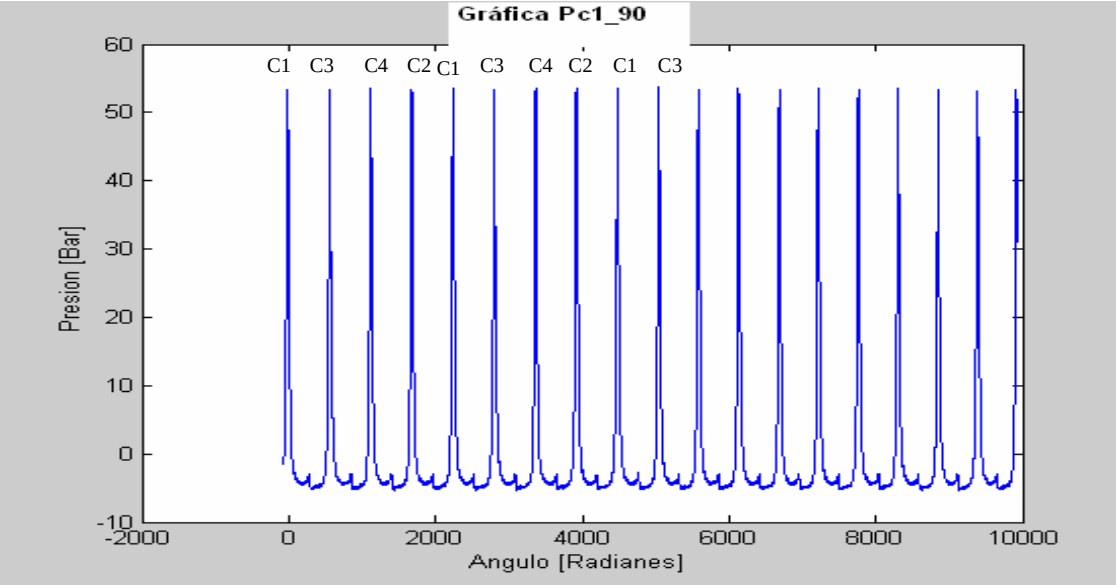


Figura 30. Grafico de PC1-90 tomando 10.000 muestras

En la figura 30 se muestra en los valores de los picos y los nombres[49] correspondientes a los cilindros de acuerdo al orden de encendido del motor.

.

4.2.1.2 PREPROCESAMIENTO DE LOS DATOS

Iniciamos el preprocesamiento de los datos de este archivos, ahora lo que se hace es asignar al vector X los datos de la primera columna, que contiene los datos del ángulo de giro del cigüeñal, y la segunda columna se asigna al vector Y, en donde se guardar los datos de la presión del émbolo para sus respectivo ángulo de giro del cigüeñal, entonces el vector X y Y, tendrán un tamaño de treinta y dos mil datos.

Analizando las gráfica del vector Y Vs vector X, Figura 30, donde se tomaron únicamente 10.000 datos de cada vector, notamos que no todos los datos son representativos para nuestro proceso de preprocesamiento, puesto que podemos identificar que hay rangos de la gráfica que corresponde al ruido captado al momento de la medición.

El principal problema para el tratamiento de estos datos, es el tamaño de los mismos, para el entrenamiento de la red no podemos utilizar la totalidad de los datos de estos vectores, por que lógicamente esto haría que nuestra herramienta software se sature y no obtenga una solución.

[49] C1: Cilindro 1, C2: Cilindro 2, C3: Cilindro 3, C4: Cilindro 4

Basándonos en la información técnica del motor, sabemos que este es un motor de cuatro (4) cilindros y su orden de encendido es 1- 3 -4 -2. En un análisis detallado de la gráfica 29, se hallaron 63 picos, ósea, que estos seria los picos según el orden de encendido para este motor en este escenario específico,

Pico	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
Pico	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
Pico	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
Pico	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
Pico	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
Émbolo	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
Pico	81	82	83	84	85	86	87	88	89							
Émbolo	1	3	4	2	1	3	4	2	1							

Tabla 8. Nombre de los picos de la gráfica 29 de acuerdo al o*rden de encendido para PC1-90*

En base en la tabla 8[50], se aprecia que la correspondencia para cada pico se repite consecutivamente cada cuatro picos, entonces diseñamos una estrategia para reducir el Tamaño del vector X y Y, la maniobra que ideamos para reducir esta cantidad de datos seria promediar los picos de la gráfica 29, en factores de cuatro (4), precisamente por este motor tiene cuatro cilindros, se debe tener presente que el numero de cilindros del motor siempre será el factor por el cual se divida los datos del vector Y, así si estuviésemos trabajando con un motor de seis (6) cilindros, 6 seria el factor con el cual dividiríamos los datos del vector de la entradas Y.

Ahora el siguiente paso a seguir es ir promediando los datos correspondientes a cada uno de los cilindros, así se saca la media para cada émbolo, así conseguimos reducir el tamaño de la muestra a 2854 datos, que son datos representativos del vector de entrada y ahora se le asigna al vector B.

Seguidamente tomamos los 2854 datos del vector X y se asignan al vector A, esto se puede hacer, ya que el movimiento que describen los pitones corresponden a una función periódica. En la figura 31[51] se muestra el grafico de A Vs B.
De todas formas esta cantidad de datos no es aceptable para el entrenamiento de la red, se necesita reducir los datos de los vectores A y B.

[50] Tabla 8, realizada por los autores
[51] Figura 31, Realizada por los autores en NEURALMOTOR 1.0

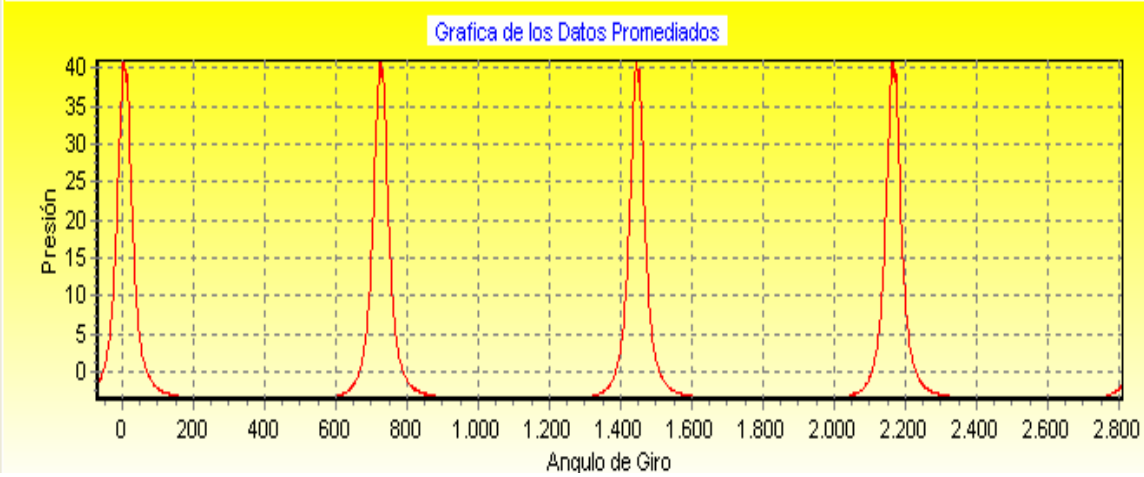


Figura 31. *Grafico del promedio de datos de entrada de PC1-90*

Hasta el momento se redujo, sustancialmente, la cantidad de valores del vector de la señal de entrada, pero aun sigue siendo sumamente grande para poder intentar entrenar la red, debemos seguir procesando este vector hasta reducirlo a una cantidad razonable de datos de entrenamiento, pero que sigan representando la de señal de entrada.

Continuamos en nuestro proceso de preprocesamiento de los datos de entrada, hábilmente decidimos aplicar la transformada de Fourier a esta este nuevo vector de la señal B, pasando del espectro del tiempo al espectro de las frecuencia, con este procedimiento podemos atenuar el ruido de la señal de entrada, hallado los armónico principales del vector B. detectamos el rango de armónicos representativos de los datos del vector B, ósea que tomamos el rango de las frecuencias bajas, y tomamos 256 muestras y se lo asignamos al vector C, entonces C es un vector de 256 datos, que son datos representativo de los datos de entrada.

Pero C es un vector que contiene valores complejos, que indiscutidamente nos complica el entrenamiento de la red, el siguiente paso es aplicar la transformada inversa de Fourier para regresar al espectro del tiempo y estos datos se asignan al vector D. En la figura 32[52] se gráfica el vector D.
Este si es un numero razonable de datos que puede ser utilizado en el entrenamiento de la red.

Pero realizando un minucioso análisis observamos que seria inoficioso introducir estos Doscientos (256) datos para entrenar la red, y que con solo con la presión máxima generada en cada uno de los cilindros seria suficiente para el entrenamiento de la red.

[52] Figura 32, Realizada por los autores en NEURALMOTOR 1.0

Figura 32. *Gráfica de los datos filtrados para Pc1_90*

En cada escenario podemos obtener dos tipos de repuesta en la salida para cada émbolo, una seria que este funcionando correctamente y otra que este experimentando alguna falla, por consiguiente la RNA estará en condiciones de identificar el estado del motor basando en las salidas generadas a partir del análisis del valor máximo de presión en cada émbolo, es suficiente que alguno de los cuatro cilindros este fallando para dictaminar que el motor no esta funcionando correctamente.

4.2.1.3 DEFINICIÓN DEL NÚMERO Y EL RANGO DE ENTRADAS

Como este motor es de cuatro cilindros, el número de entradas para red será de cuatro, el rango de la entradas provienen de los valores máximos de presión en cada émbolo, después del Preprocesamiento de los datos de entrada. Ver tabla 9[53].

Émbolo 1	Émbolo 3	Émbolo 4	Émbolo 2
41.39 [bar]	41.43 [bar]	41.42 [bar]	41.47 [bar]

Tabla 9. *Presión máxima en cada émbolo después del Preprocesamiento de Pc1_90*

4.2.1.4 DEFINIR EL NÚMERO Y RANGO DE LAS SALIDAS.

El número de las salidas, también, corresponde al número de cilindros de motor, puesto que la red se debe pronunciar puntualmente sobre cada émbolo, entonces se fijan cuatro salidas para este escenario, y el rango de la salida 40.22= x < 41.47 y x > 41.47 se le asigna un valor de uno (1), y estos valores son aceptados como salidas correctas, cuando x < 40.22 se le asigna un valor de cero (0) en este caso se pronostica una falla.

[53] Tabla 9, realizada por los autores

4.2.1.5 NORMALIZACIÓN DE LOS DATOS DE ENTRADAS Y SALIDAS.

Es recomendable realizar la normalización de los datos de entrada y salida de la red para que los datos sean más manejables en el proceso de aprendizaje de la red. Aplicando la normalización todo valor de x < 40.22 estará por debajo de cero (0), y si 40.22< x > 41.47, su equivalente entre cero y uno x > 41.47 su valor estará por encima de uno (1).

4.2.1.6 ELECCIÓN DE TIPO DE REDES A UTILIZAR.

Para todos los escenarios de este motor aplicaremos las redes de topología PMC con algoritmo Backpropagation y RBF, pues uno de las finalidades de la investigación es identificar cual de estas dos metodologías es más eficaz para el tratamiento de este ejercicio de identificación de sistemas dinámicos no lineales.

4.2.1.7 BÚSQUEDA DE LA ARQUITECTURA DE LA RED.

Para hallar las arquitecturas de red, que pronostiquen resultados acertados para nuestra investigación, se generan los datos de entrenamiento, se emplea la funcionalidad para generar los datos de entrenamiento, la herramienta software se encarga de generar automáticamente los datos de entrenamiento, este componente nos facilita este labor, puesto que procedemos a entrenar las redes generadas con estos datos, para a nuestro ejemplo del escenario Pc1_90[54], los resultados obtenidos después de aplicar el proceso de preprocesamiento de los datos de entrada de señal, se presenta en la tabla 10[55].

De acuerdo a estos rangos se diseña la tabla de datos de entrenamiento de la red, entre mas datos de entrenamiento se le den a conocer a la red mayor será la posibilidad de acierto de la misma, pero no es bueno entrenar la red con mas de 100 casos, puesto que redes de RBF requiere de un costo computacional bastante grande. Para mejor entendimiento de la tabla 9, la entrada y la salida 1 corresponden al émbolo 1, la entrada y salida 2, corresponde al émbolo 3, así sucesivamente.

Se aplica la normalización de máximos y mínimos a estos datos de entrenamiento y se toma como el rango de entrada máximo 41.47 y mínimo 40.22, y el rango de las salidas máximo 1 y mínimo 0, ahora los datos de entrenamiento que eran menores de 40.22 se da un valor menor que cero y los datos 40.22< x < 41.47 valores entre cero y uno y aquellos que eventualmente sea superiores a 41.47 se les da un valor superior de 1.

[54] Véase tabla de símbolos
[55] Tabla 10, Realizada por los autores

	Émbolo 1	Émbolo 3	Émbolo 4	Émbolo 2	Émbolo 1	Émbolo 3	Émbolo 4	Émbolo 2
No	Entrada 1	Entrada 2	Entrada 3	Entrada 4	Salida 1	Salida 2	Salida 3	Salida 4
1	41.35	40.94	40.83	40.81	1	1	1	1
2	40.53	40.78	41.02	39.74	1	1	1	0
3	41.25	40.87	39.21	40.45	1	1	0	1
4	41.05	40.65	39.68	39.48	1	1	0	0
5	40.85	39.01	40.65	41.25	1	0	1	1
6	40.95	39.95	41.12	39.65	1	0	1	0
7	40.35	39.87	39.28	41.12	1	0	0	1
8	41.09	39.24	39.32	39.45	1	0	0	0
9	40.02	39.25	39.25	39.61	0	0	0	0
10	39.75	39.02	39.65	41.25	0	0	0	1
11	39.85	39.45	40.24	39.24	0	0	1	0
12	39.68	39.36	41.36	40.61	0	0	1	1
13	39.02	41.25	39.32	39.24	0	1	0	0
14	39.65	41.15	39.69	41.24	0	1	0	1
15	39.44	41.26	40.46	39.52	0	1	1	0
16	39.65	40.28	41.28	40.68	0	1	1	1

Tabla 10. *Datos de entrenamiento de la red para el escenario PC1_90*

Ahora se generan las estructuras de red para que sean de topología backpropagation, solo se fija el número de entradas y salida de la red y se sabe que son cuatro. **NEURALMOTOR 1.0** genera automáticamente las siguientes redes que se resumen en la tabla 11[56].

Red 1	Numero de capas	Capa	neurona	Función de activación
	3	Oculta1	4	Sigmoidal
		salida	4	Sigmoidal
Red 2	Numero de capas	Capa	neurona	Función de activación
	3	Oculta1	4	Sigmoidal
		salida	4	Tangente-hiperbólica
Red 3	Numero de capas	Capa	neurona	Función de activación
	4	Oculta1	4	Sigmoidal
		Oculta2	6	Sigmoidal
		salida	4	Sigmoidal
Red 4	Numero de	Capa	neurona	Función de

[56] Tabla 15,realizada por los autores

	capas			activación
	4	Oculta1	4	Sigmoidal
		Oculta2	6	Sigmoidal
		salida	4	Identidad
Red 5	Numero de capas	Capa	neurona	Función de activación
	3	Oculta1	4	Sigmoidal
		salida	4	Identidad

Tabla 11. *Resumen de las estructuras de red Backpropagation generadas para PC1_90*

Este mismo proceso se utiliza para generar las RNA de topología de Función Base Radial (RBF), en la tabla 12[57] se muestran la redes generadas.

Red 6	Numero de capas	Capa	neurona	Función de activación
	3	Oculta1	20	Gaussiana
		salida	4	Sigmoidal
Red 7	Numero de capas	Capa	neurona	Función de activación
	3	Oculta1	40	Gaussiana
		salida	4	Sigmoidal
Red 8	Numero de capas	Capa	neurona	Función de activación
	3	Oculta1	160	Gaussiana
		salida	4	Sigmoidal
Red 9	Numero de capas	Capa	neurona	Función de activación
	3	Oculta1	40	Gaussiana
		salida	4	Identidad
Red 10	Numero de capas	Capa	neurona	Función de activación
	3	Oculta1	100	Gaussiana
		salida	4	Identidad

Tabla 12. *Resumen de las estructuras de red RBF generadas para PC1_90*

[57] Tabla 12, realizada por los autores

4.2.1.8 ENTRENAMIENTO DE LAS REDES SELECCIONADAS

Para cada una de las redes generadas anteriormente, procedemos a entrenar cada una de ellas con los datos de entrenamiento normalizados, en la tabla 13[58], se muestra los resultados obtenidos para cada una de estas redes Backpropagation.

Red 1	Numero de iteraciones	Tasa de aprendizaje	Momentum	Error
	1000	0.1	0.8	0.079%
Red 2	Numero de iteraciones	Tasa de aprendizaje	Momentum	Error
	3	0.1	0.8	3.47%
Red 3	Numero de capas	Tasa de aprendizaje	Momentum	Error
	3	0.1	0.8	0.029%
Red 4	Numero de capas	Tasa de aprendizaje	Momentum	Error
	3	0.1	0.8	0.388%
Red 5	Numero de capas	Tasa de aprendizaje	Momentum	Error
	3	0.1	0.8	2.25%

Tabla13. *Entrenamiento de las redes Backpropagation generadas para PC1_90*

En la tabla 14[59] se muestra los resultados obtenidos en el entrenamiento de las redes generadas anteriormente de topología (RBF)

Red 6	Numero de iteraciones	Tasa de aprendizaje	Momentum	Error
	1000	0.1	0.8	16.07%
Red 7	Numero de iteraciones	Tasa de aprendizaje	Momentum	Error
	3	0.1	0.8	0.025%
Red 8	Numero de capas	Tasa de aprendizaje	Momentum	Error
	3	0.1	0.8	4.12%

[58] Tabla 13, realizada por los autores
[59] Tabla 14, realizada por los autores

Red 9	Numero de capas	Tasa de aprendizaje	Momentum	Error
	3	0.1	0.8	5.29%
Red 10	Numero de capas	Tasa de aprendizaje	Momentum	Error
	3	0.1	0.8	12.25%

Tabla 14. E*ntrenamiento de las redes RBF generadas para PC1_90*

4.2.1.9 PRUEBA Y ELECCIÓN DE LAS REDES

De acuerdo a lo sugerido por la literatura actual, se conoce que aquellas redes que su error de convergencia es mayor del 5%, es poco probable que ofrezcan resultado óptimos, pero no se debe pensar que aquella red que consiguen el valor de error mas bajos es la que mejor se desempeña, para despejar la duda de cual de las redes halladas tienen mas eficacia debemos simularlas con los datos de prueba. Entonces después de haber entrenado todas las redes generadas bajo las mismas condiciones, proseguimos con la prueba de cada una de ellas, para esto se toma el 30% de los datos de entrada de señal dispuesto para la prueba y se le hace el pre-procesamiento de los datos, se cargan a **NEURALMOTOR 1.0**.

Posteriormente, se normalizan, esto datos se simulan y de acuerdo a los resultado que entregue cada red, elegimos cual de ellas entrega resultados mas ajustados a al realidad, en la tabla 15[60] se muestra los resultados obtenidos para cada red y se determina de acuerdo a nuestra intuición cual es la red que mejores resultados proporciona.

Tipo	Nombre	Error	concepto	Calificación 1 a10
Backpropagation	Red 1	0.11%	Tiene pequeños problemas de resultado, que eventualmente pueden ser resueltos, cambiando las condiciones de entrenamiento	9.0
Backpropagation	Red 2	0.13%	Tiene pequeños problemas de resultado, que eventualmente pueden ser resueltos, cambiando las condiciones de entrenamiento	9.5
Backpropagation	Red 3	4.43%	Tiene graves problemas de resultado que las dos redes	8.0

[60] Tabla 15,realizada por los autores

			anteriores, aunque pueden se puede corregir es mas difícil su afinación.	
Backpropagation	Red 4	0.39%	Tiene problemas de resultados que difícilmente pueden ser corregidos	6.0
Backpropagation	Red 5	5.62%	No es acta	5.0
RBF	Red 6	6.87%	No es esta	4.0
RBF	Red 7	1.84%	Presenta problemas para entregar resultados, pero estos se pueden corregir manipulando los parámetros de entrenamiento	8.5
RBF	Red 8	0.96%	Presenta problemas para entregar resultados, pero es la red mas efectiva de las este tipo.	9.0
RBF	Red 9	6.92%	No es acta	5.0
RBF	Red 10	2.08%	No es Acta	4.0

Tabla 15. Prueba de las *redes generadas para PC1_90*

La red backpropagation elegida es la red 2 y la red RBF elegida es la red 8, estas son las redes elegidas para ser afinadas.

4.2.1.10 AFINACIÓN DE LAS REDES

Para obtener resultados mas acordes a la realidad de nuestro caso de identificación de sistemas dinámicos no lineales, procedemos a afinar las redes 2 y 8, en este proceso se determino que la red 2 redujo su error de acierto cuando se le modifico la tasa de aprendizaje de 0.1 a 0.2 y en el caso de la red 8 se redujo el error de acierto cuando se modifico la tasa de aprendizaje de 0.1 a 0.4, como se muestra en la tabla 16[61].

Topología	Nombre	Error	Concepto
Backpropagation	Red 2	2.47%	Mejoro la predicción de resultados
RBF	Red 8	2.45%	Mejoro la predicción de resultados

Tabla 16. Resultados de afinación de las *redes elegidas para PC1_90*

[61] Tabla 16, realizada por los autores

4.2.1.11 REPORTE DE RESULTADOS

En la tabla 17[62] se muestra el reporte de resultados para red 2, antes del proceso de afinación y en la tabla 18[63] se muestran los resultados después de su afinación.

Dato	Entrada1	Entrada2	Entrada3	Entrada4	Salida1	Salida2	Salida3	Salida4
1	1	1	1	0.065	0.999408	0.999982	0.998843	0.503496
2	-0.106	1	-0.097	1	0.501667	0.999947	0.499088	0.999081
3	1	0.034	1	1	0.999662	0.506779	0.998597	0.999646
4	-0.055	1	1	0.031	0.501701	0.999972	0.99974	0.505467

Tabla 17. Resultados de la red Backpropagation (red2) *para PC1_90*

Dato	Entrada1	Entrada2	Entrada3	Entrada4	Salida1	Salida2	Salida3	Salida4
1	41.47	41.47	41.47	40.225	1.037531	1.032581	0.978276	0.503504
2	40.287929	41.47	40.248444	41.47	0.497624	0.975343	0.499507	1.012663
3	41.47	40.230182	41.47	41.47	1.028699	0.500694	1.010322	0.986082
4	40.290396	41.47	41.47	40.264485	0.504696	1.029264	1.012408	0.50065

Tabla 18. Resultados de la red Backpropagation (red2) afinada *para PC1_90*

Se debe aclarar que los valores en las salidas y > 0.5, nos pronostica que la entrada de ese émbolo es optima, pero si en la salida y < 0.5 nos indica que la entrada es incorrecta y, por lo tanto ese émbolo tiene alguna falla.

Pero para entender los resultados de la tabla 17, se tiene que los datos x < 40.225 después de la normalización toman un valor de menor que cero, entonces el valor mas alejado de este punto es 40.29, lo que nos indica que el rango que determina si el funcionamiento es correcto o no se desvía 0.065 lo que equivale a 0.1567%, ahora no se tiene un rango de 3%, sino de 2.8432%, un resultado bastante aceptable.

Dato	Entrada1	Entrada2	Entrada3	Entrada4	Salida1	Salida2	Salida3	Salida4
1	1	1	1	0.19	0.999999	0.99999	1	0.500847
2	0.111	1	-0.052	1	0.524578	1	0.503043	1
3	1	0.258	1	1	1	0.504782	0.999999	0.999978
4	-0.093	1	1	0.071	0.496256	1	1	0.50244

Tabla 19. Resultados de la red RBF (red8) *para PC1_90*

[62] Tabla 17, realizada por los autores utilizando NEURALMOTOR 1.0
[63] Tabla 18, realizada por los autores utilizando NEURALMOTOR 1.0

En la tabla 19[64] se muestra los datos de la red 8, antes de la afinación y en la tabla 20[65] los datos de la red 8 después de su afinación.

Dato	Entrada1	Entrada2	Entrada3	Entrada4	Salida1	Salida2	Salida3	Salida4
1	41.47	41.47	41.47	40.25239	0.971652	0.977303	0.955973	0.499428
2	40.22998	41.47	40.0507	41.47	0.502127	0.998819	0.502811	0.997701
3	41.47	40.29223	41.47	41.47	0.991557	0.49995	0.957784	0.973915
4	40.236205	41.47	41.47	40.298455	0.498858	0.999317	0.998907	0.50053

Tabla 20. Resultados de la red RBF (red8) afinada *para PC1_90*

En la tabla 20 se aprecia que hay valores en las entradas por debajo y por encima cercanos a 40.225 que es punto de división entre los valores correctos y los incorrectos para la presión máxima en los cilindros para este escenario. Se observo que para algunos casos particulares donde los valores 40.05 = X = 40.29, el dictamen no es exacto como quisiéramos, pero igualmente la red maneja un error de entrenamiento que permite que esto ocurra, seguramente el usuario en la aplicación real resolverá este dilema. Entonces, el resultado mas alejado del limite 40.225, donde se evidencio error en el dictamen es de 0.175, que equivale a 0.4219% que es un valor un poco grande pero bastante bueno para una red RBF.

4.2.1.12 ANÁLISIS DE RESULTADOS

Según los datos obtenidos para este caso de Pc1_90, las redes de topología backpropagation, entregaron resultados mas ajustados a la realidad comparados con los datos obtenidos con las redes de topología RBF, los resultados de simulación lo corroboran, pues la red backpropagation obtuvo un valor de error en el acierto de 0.1567%, en cambio la red RBF obtuvo un valor de 0.4219%.

4.2.2 ANALISIS DE OTROS ESCENARIOS

En adelante presentamos la información concerniente al tratamiento de los datos de otros escenarios, debido a que la aplicación de metodología es la misma, no es lógico volver a repetir paso a paso el proceso, como lo hicimos anteriormente, por este motivo se presenta solo los resultados obtenidos para cada uno de estos escenarios.

[64] Tabla 19, realizada por los autores utilizando NEURALMOTOR 1.0
[65] Tabla 20, realizada por los autores utilizando NEURALMOTOR 1.0

4.2.2.1 ESCENARIO Pc1_140

En la tabla 21[66] se resumen la información mas relevante de la aplicación de la metodología a este escenario, vale la pena recordar que para este escenario el numero de entradas y salidas de la red es igual a cuatro, debido a que este es el número de cilindros de este motor.

Tipo de red	Estructura de red		Error de la red	Análisis de resultados
backpropagation	Número de capas	3	0.798%	Esta red tiene un buen rendimiento y supero a la RBF
	Capa oculta	4 T[67]		
	Capa salida	4 S[68]		
	Número de iteraciones	1.000		
	Tasa de aprendizaje	0.1		
	Momentum	0.8		
RBF	Número de capas	3	4.77%	Aunque se esperaba encontrar una red de mas bajo error, no obstante esta red predice buenos resultados
	Capa oculta	160 G[69]		
	Capa salida	4 S[70]		
	Número de iteraciones	1.000		
	Tasa de aprendizaje	0.01		
	Momentum	0.8		

Tabla 21. Resumen de resultados *para el escenario PC1_140*

4.2.2.3 ESCENARIO Pc1_340

En la tabla 22[71] se resumen la información mas relevante de la aplicación de la metodología a este escenario, en este caso también el numero de entradas y salidas de la red es igual a cuatro.

[66] Tabla 21, realizada por los autores utilizando NEURALMOTOR 1.0
[67] función de activación Tangencial-hiperbólica
[68] función de activación Sigmoidal
[69] función de activación Gaussiana
[70] función de activación Sigmoidal
[71] Tabla 22, realizada por los autores utilizando NEURALMOTOR 1.0

Tipo de red	Estructura de red		Error de la red	Análisis de resultados
backpropagation	Número de capas	3	0.098%	Esta red tiene un buen rendimiento y supero a la RBF
	Capa oculta	4 S[72]		
	Capa salida	4 S[73]		
	Número de iteraciones	1.000		
	Tasa de aprendizaje	0.1		
	Momentum	0.8		
RBF	Número de capas	3	1.77%	Aunque se esperaba encontrar una red de mas bajo error, no obstante esta red predice buenos resultados
	Capa oculta	23 G[74]		
	Capa salida	4 S[75]		
	Número de iteraciones	1.000		
	Tasa de aprendizaje	0.01		
	Momentum	0.8		

Tabla 22. Resumen de resultados *para el escenario PC1_340*

[72] función de activación Sigmoidal
[73] función de activación Sigmoidal
[74] función de activación Gaussiana
[75] función de activación Sigmoidal

Capitulo 5

CONCLUSIONES

A continuación se describen las conclusiones más relevantes de este estudio.

- La identificación de sistemas es una metodología efectiva para resolver casos donde solo se tiene información referente a las entradas y salidas del sistema; la aplicación de las redes neuronales artificiales facilita la identificación de patrones, ofreciendo respuestas a sistemas de difícil resolución, como son los sistemas dinámicos no lineales.
- El software representa un adelanto en el estado del arte, debido a que se implementaron y documentaron nuevas tecnologías emergentes en nuestro ámbito local, lo que permite un avance científico en la solución de problemas evidentes en vida diaria, los cuales carecen de estudios claros por que tiene un comportamiento dinámico no lineal.
- La metodología desarrollada, facilitó la búsqueda de la arquitectura de RNA apropiada, evitando procesos tediosos que en ocasiones tiene poca probabilidad de acierto
- Las arquitecturas de red desarrolladas, cumplieron con los requisitos propuestos en la metodología, la red PMC con algoritmo backpropagation superó a la red de Función de Base Radial (RBF), en cuanto a resultados, rendimiento y capacidad de generalización.
- Los resultados obtenidos en esta investigación cumplieron con los objetivos propuestos inicialmente, pues utilizando la identificación de sistemas dinámicos no lineales, se pudo implementar y aplicar las redes Backpropagation y RBF al caso en estudio, análisis de presión en motores de combustión interna, encontrase, respuestas precisas con bajos márgenes de error y que tiene correspondencia con la realidad.
- **NEURALMOTOR 1.0**, se convierte en una herramienta novedosa, puesto que no se conocen antecedentes de la construcción de una herramienta de estas características, donde se halla implementado con éxito las dos topologías de red PMC y RBF en la identificación de sistemas dinámicos no lineales, enfocados al análisis de motores de combustión interna.
- La herramienta desarrollada queda con la facultad de migrar hacia otro tipo de problemas, en la que se requiera de la identificación de sistemas dinámicos no lineales, y otras aplicaciones de MCI.
- Se pudo establecer que la aplicación de las redes neuronales artificiales de topología Backpropagation son mas prácticas para el usuario final, ofrecen mas variables de entrenamiento y requiere menos recursos de computación en comparación con las redes de topología Función de Base Radial (RBF).

CAPITULO 6

RECOMENDACIONES

- En ocasiones se desechan algunas estructuras de red generada, por que se cree que aquellas redes que tienen menor error de entrenamiento son mas efectivas y no siempre ocurre así, por lo tanto se debe probar todas las redes generadas cuyo error de entrenamiento sea menor al 5%, puesto que son potencialmente acertadas.
- Implementar y experimentar más tipos de RNA.
- Cuando se requiera aumentar la preedición de una red, una buena idea es aumentar el número de datos de entrenamiento, así se consigue respuestas mas acertadas, pero al aumentar los datos de entrenamiento de las RNA se requiere de más recurso de máquina para el entrenamiento, esta es una situación especial que usuario debe manejar, puesto que a mayor cantidad de datos el error en las salidas se disminuye, pero se aumenta el tiempo de procesamiento.
- Se debe combinar la RNA con la lógica borrosa, para analizar la eficacia en la identificación de sistemas no lineales.
- Avanzar hacia la implementación de la herramienta en la Web

Capitulo 7

ANEXOS

7.1 MANUAL DE USUARIO

7.1.1 BIENVENIDO A LA AYUDA DE NEURALMOTOR 1.0

En esta ayuda se le orientará para que maneje el aplicativo de NEURALMOTOR 1.0 iniciando desde cero. Para esto es necesario que usted tenga fundamentos en redes neuronales artificiales, y conozca las ventajas y desventajas de trabajar con los dos tipos de red que maneja el software, Backpropagation y Redes de Función Base Radial (RBF)

Una de las utilidades de este software es permitirle manipular múltiples varios prototipos de arquitecturas antes de llegar a la que se acerque a la generalización que quiera conseguir, quizá le ayude el Conociendo a NEURALMOTOR 1.0

La finalidad de este producto, es ofrecerle una herramienta de cómodo manejo y que facilita la búsqueda de la arquitectura de red que más se ajuste a las necesidades de su modelo, para esto siga las Recomendaciones.

7.1.2 CONOCIENDO A NEURALMOTOR 1.0

NEURALMOTOR en su primera versión trata de ser lo más útil posible para el usuario, con varios extras en su ejecutable, el usuario puede permitirse un análisis documentado de sus datos y de las redes creadas para su estudio.

Con NEURALMOTOR 1.0 puede:

Crear una red neuronal de Perceptrón Multicapa con algoritmo Backpropagation (BPM) o de Función de Base Radial (RBF).
Cargar datos para el entrenamiento de la red desde un archivo de texto con dos opciones avanzadas, Normalización y escogencia de Entradas/Salidas, pudiendo Reserva de ese conjunto de datos un porcentaje para realizar pruebas durante la ejecución, así como la oportunidad de asignar un Nombre a las Variables.

Realizar el Entrenamiento de cualquier red neuronal ya creada, con parámetros cambiantes como los ciclos de entrenamiento, la taza de aprendizaje y el momentum, pudiendo detener este entrenamiento en cualquier momento.
Simular cualquier red neuronal ya creada, cargando los datos de prueba desde un archivo de texto, que le permiten mayor flexibilidad al usuario, de Normalizar si lo necesita, exportar estos datos a Excel para que pueda trabajarlos más cómodamente.

Acciones como guardar, Cargar, Clonar, Cambiar el Nombre, Inicializar, mostrar los Detalles y Eliminar están activas para cada red.
Generación de Reportes en los que se pueden incluir datos de simulación, Gráficas en 2D, generalidades y especificaciones de la red neuronal, están disponibles para los usuarios.

7.1.3 RECOMENDACIONES PARA EL USUARIO

Para un óptimo desarrollo de NEURALMOTOR 1.0, es necesario que tenga en cuenta las siguientes recomendaciones antes de empezar a trabajar:

Debido a la naturaleza de las redes neuronales artificiales, el sistema requiere de un buen desempeño del computador durante su ejecución, por lo tanto, trate en lo posible de no utilizar programas que puedan obstaculizarlo, especialmente en el uso de la unidad de procesamiento.
Para las redes perceptrón multicapa con algoritmo Backpropagation, el aplicativo trae habilitadas hasta seis capas, sin embargo por experiencia se sabe que para casi todos los modelamientos, una red con máximo cuatro capas es más que suficiente.

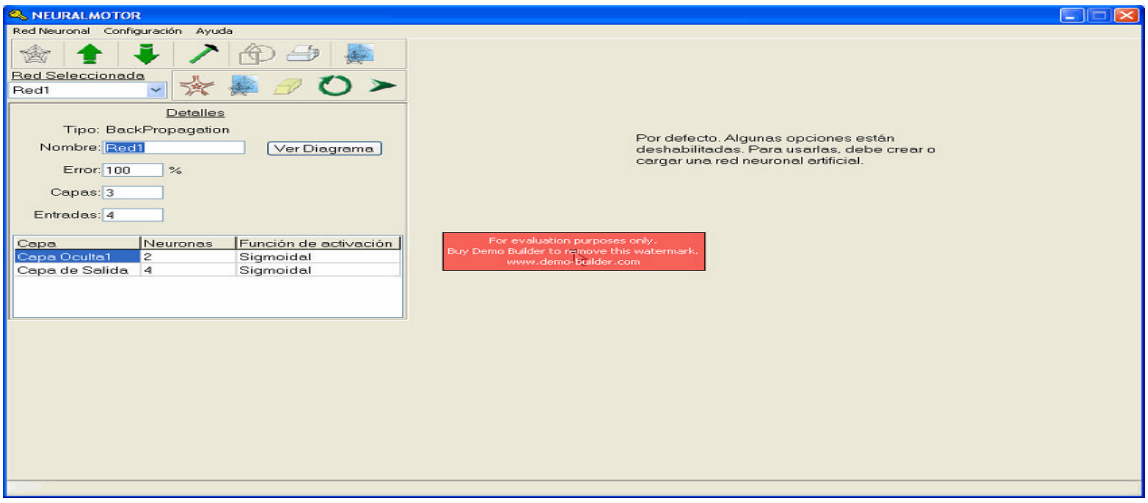
Entre más neuronas hayan en una arquitectura, más demorará el sistema en cada ciclo de entrenamiento, lo mismo pasa cuando los datos de entrenamiento son excesivos, por tanto se recomienda ser consecuentes con estas variables y estudiar bien los modelos a proponer para evitar que el entrenamiento, simulación y generación de reportes sean tediosos.
Cuando tenga arquitecturas que le han dado resultado, no olvide guardarlas, estas le servirán más adelante como guías para la afinación del modelo.

Tenga en cuenta que cada computador tiene definido su operador separador decimal, por lo tanto asegúrese de seleccionar el correcto para sus datos antes de empezar a trabajar.

7.1.4 PRESENTACIÓN DE NEURALMOTOR 1.0

NEURALMOTOR 1.0 fue creada con el propósito de resolver un sistema dinámico no lineal a partir de los datos obtenidos experimentalmente correspondientes a escenarios donde se pueden examinar el desempeño óptimo o defectuoso del sistema.

Concretamente NEURAMOTOR 1.0 es una herramienta software capaz de diagnosticar el estado real de motor de combustión a partir de los datos Angulo de giro del cigüeñal Vs presión generada en sus cilindros.



Para conocer mas a fondo las funcionalidades de NEURALMOTOR 1.0 diríjase ha empezando Desde Cero

7.1.5 DESDE CERO

En este tutorial se muestran todas las posibilidades que como principiante en el uso de NEURALMOTOR 1.0 debe conocer. En los siguientes pasos, aprenderá a manejar el software de una manera gráfica, sencilla y práctica.

A continuación se listan las opciones principales del sistema:

Crear

Guardar

Cargar

Entrenar

Simular

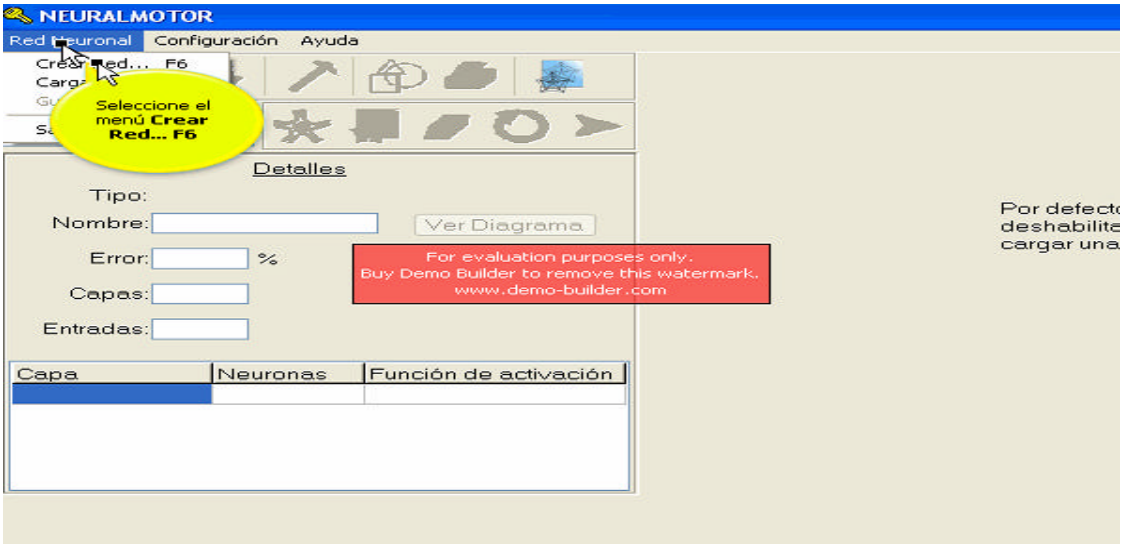
7.1.5.1 ¿CÓMO CREAR UNA RED?

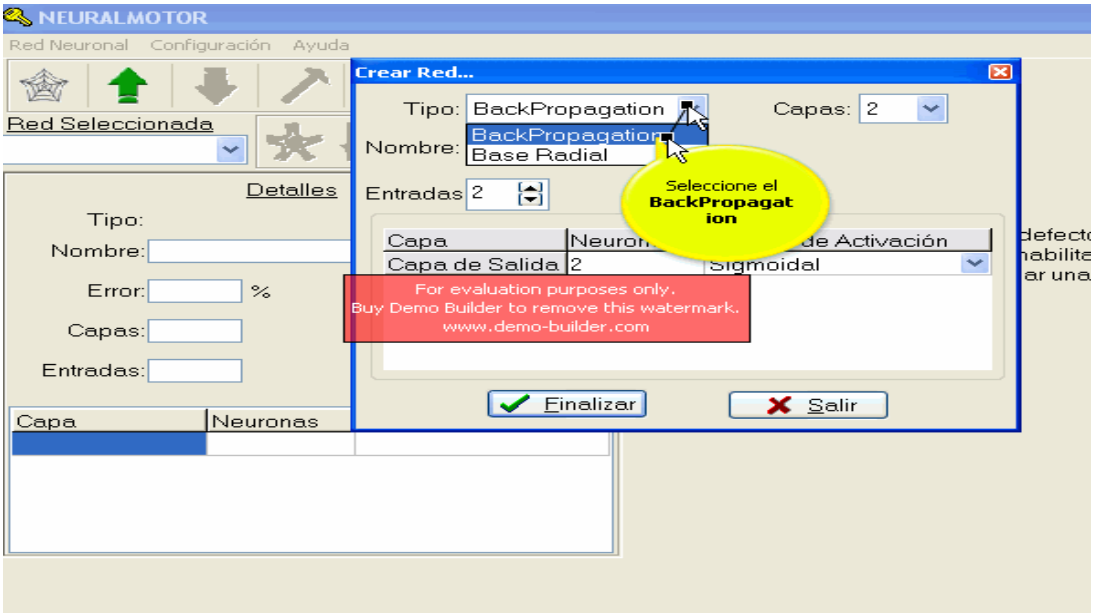
Para crear una red neuronal artificial (RNA), hay que tener primero en claro:

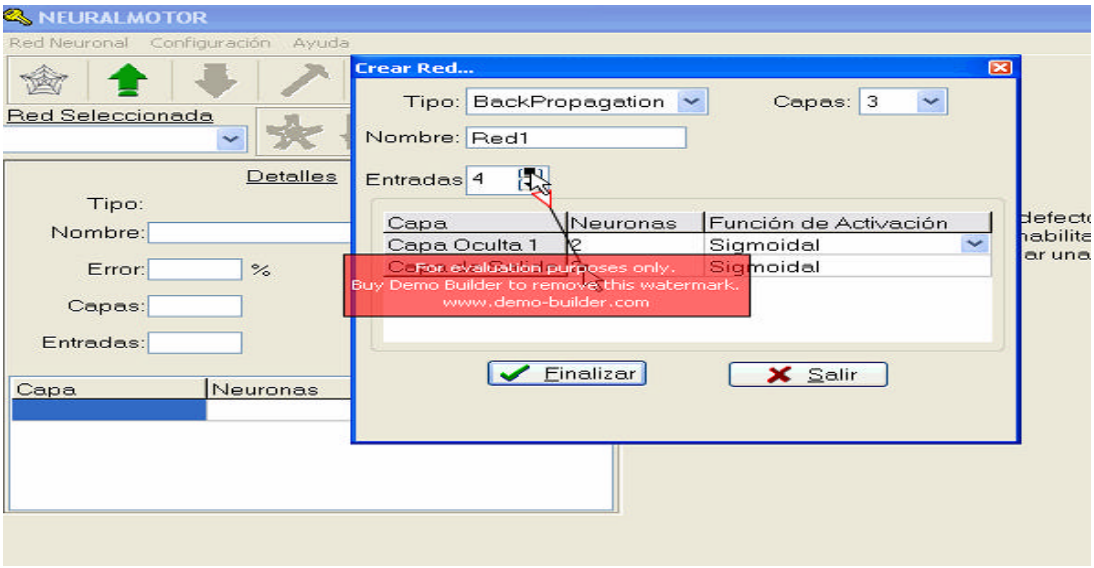
- ¿Cuántas entradas y cuantas salidas va tener la red?
- ¿Cuál va a ser el rango que van a manejar?
- ¿Hay que normalizar las entradas o no?
- ¿Qué tipo de modelo se quiere realizar?, ejemplo: predictivo, clasificador.

Después de contestadas estas dudas, hay dos opciones para la generación de la red, por el menú de principal de NEURALMOTOR 1.0 o por el botón de creación de red.

Abra NEURALMOTOR 1.0 y siga las instrucciones que se muestran en siguientes figuras para crear la red para un motor de cuatro cilindros, en donde la red tendrá cuatro entradas y cuatro salidas y elegimos dos capaz ocultas.





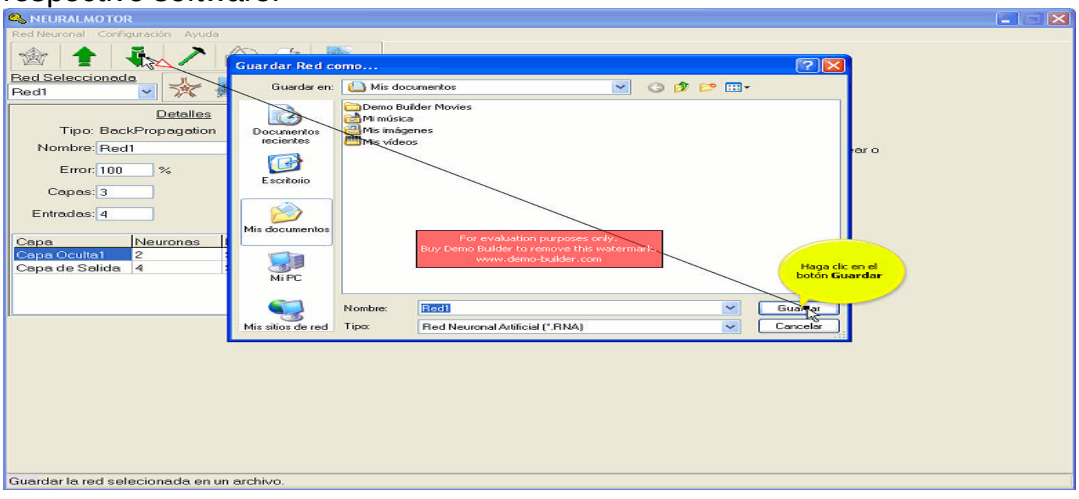


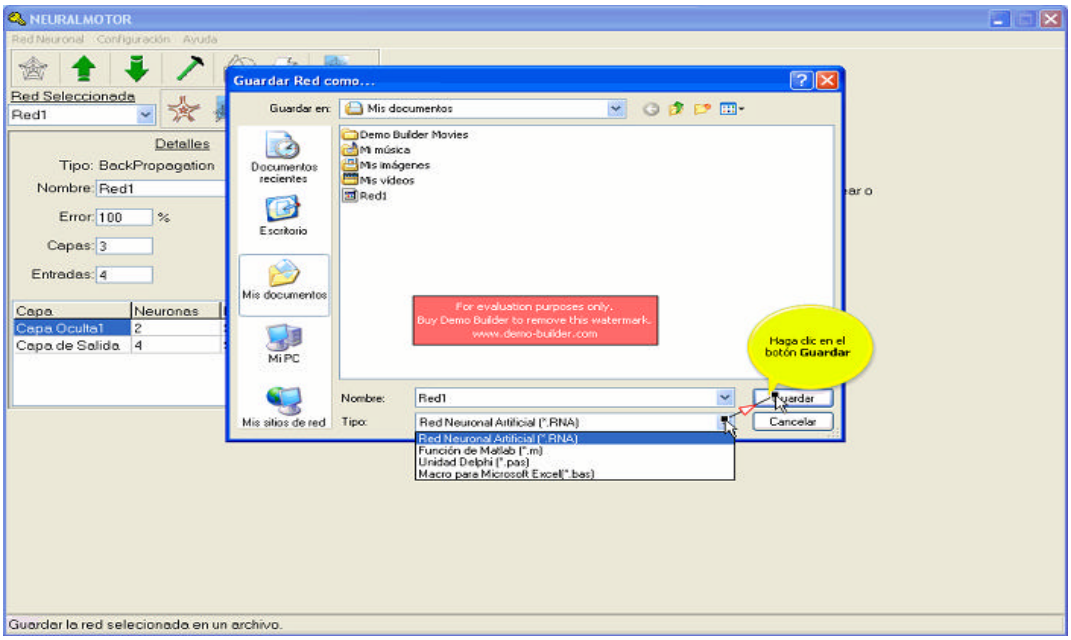
7.1.5.2 ¿CÓMO GUARDAR UNA RED?

Para Guardar las redes ya creadas esta la opción Guardar en el menú Red neuronal allí va tener varias opciones de salvado, entre las que cuentan:

Como archivo .m de Matlab®
Como código fuente para Delphi® y Kylix® dentro de un .pas
Como archivo .rna para ser utilizado por NEURALMOTOR.
Como Macro de Microsoft Excel®

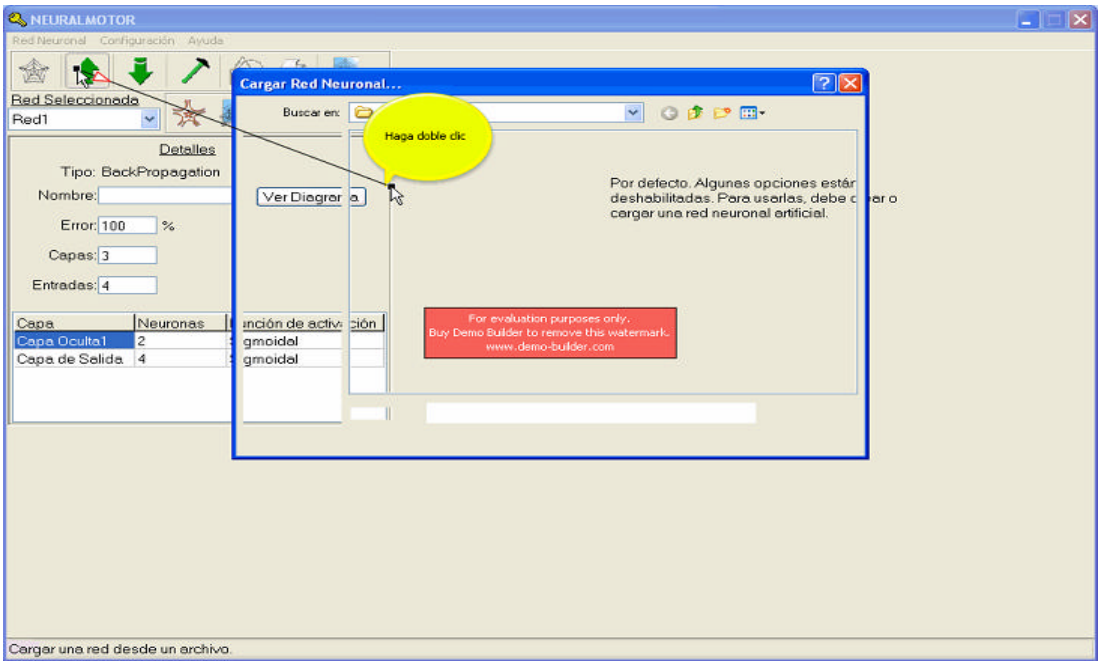
Seleccionando la opción Guardar red del menú red neuronal se llega a la conocida ventana de guardar archivo como, en la parte inferior se encuentran las opciones anteriormente nombradas. Solo se selecciona la red que se desee guardar y oprima guardar, el archivo así guardado queda listo para ser utilizado con su respectivo software.

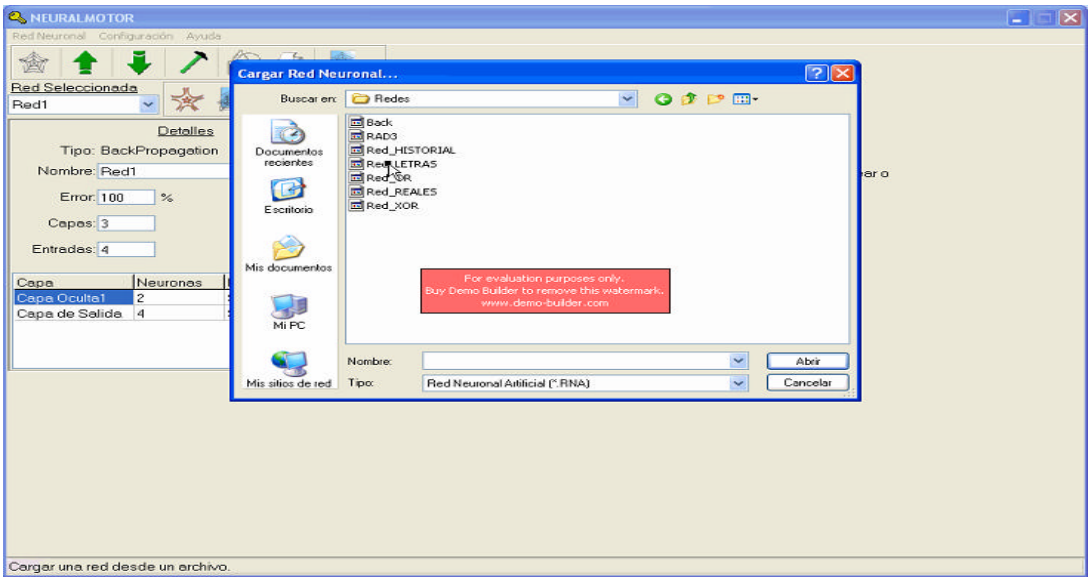




7.1.5.3 ¿CÓMO CARGAR LA RED?

En la opción cargar red del menú red neuronal, se encuentra en el cuadro de diálogo, por medio de esta funcionalidad se puede acceder a las redes guardadas en memoria con el formato .rna.

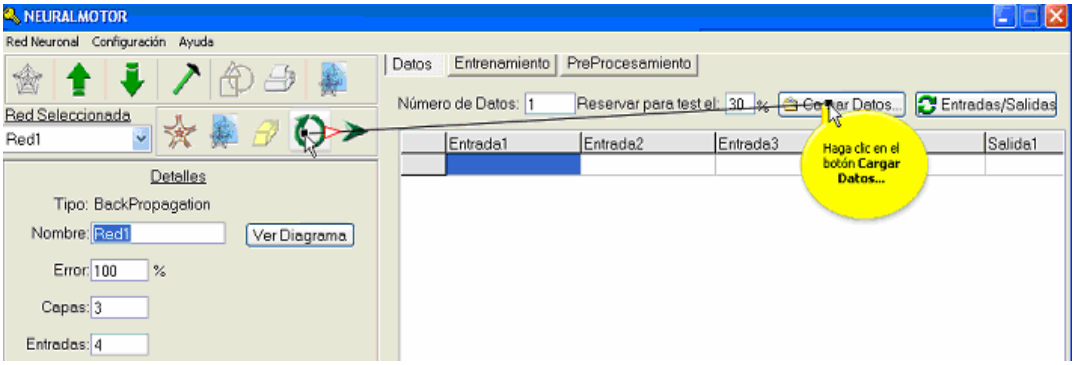


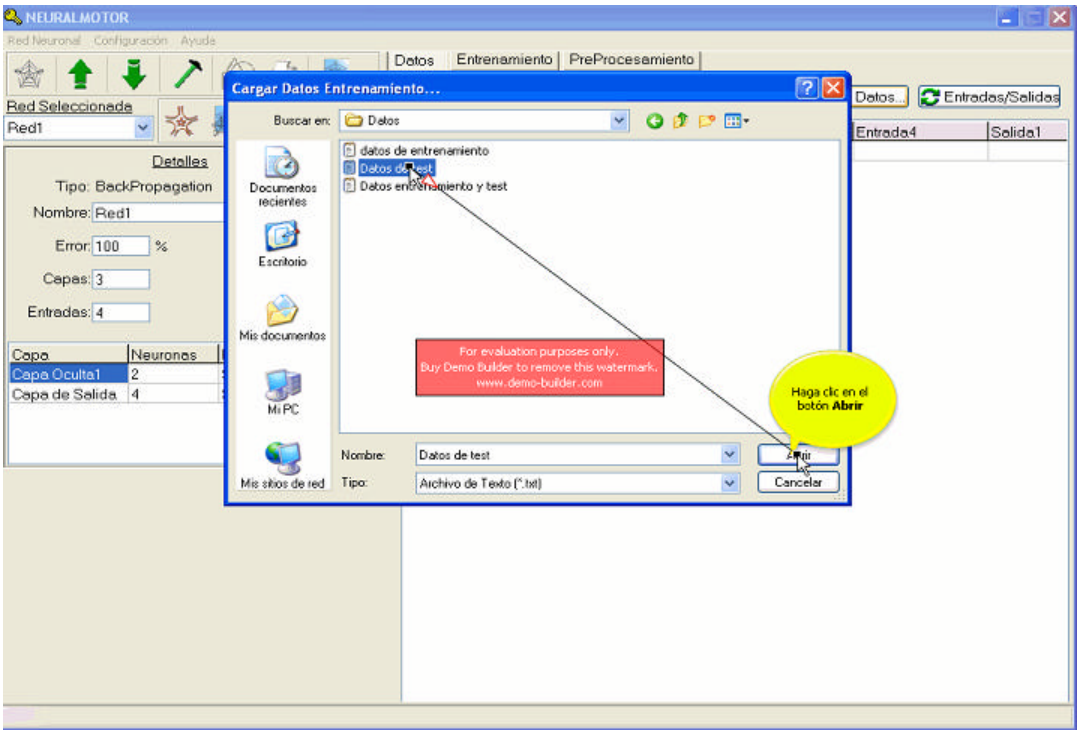


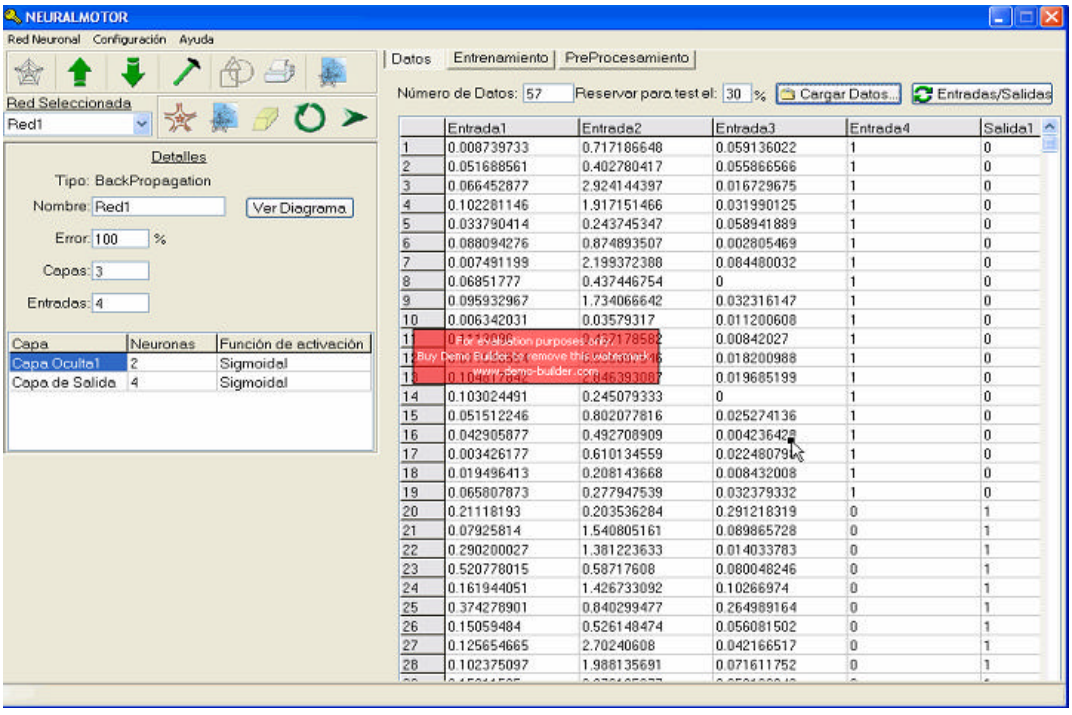
7.1.5.4 ENTRENAMIENTO DE LA RED

El entrenamiento es el proceso de más cuidado en la construcción de un modelo basado en redes neuronales, el rango de las entradas, las entradas irrelevantes, un conjunto de entrenamiento parcializado o mal escogido, en fin, múltiples variables que hay que tener en cuenta si no se quiere perder valioso tiempo en analizar datos fallidos y en construir, escoger y entrenar nuevos prototipos de red. Es por esto que antes de poner a entrenar cualquier red, es necesario analizar si el conjunto de entrenamiento realmente aporta a un generalización o si simplemente son datos sesgados, muy extremos, demasiado pocos o en determinados casos demasiados datos.

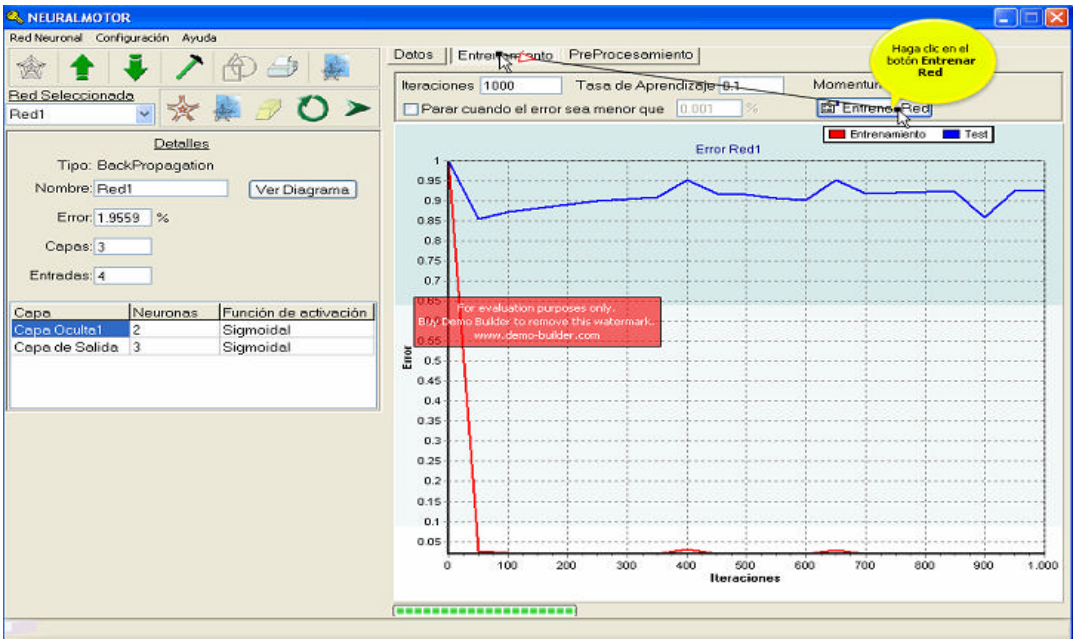
Una ayuda que quizá le pueda servir es el subconjunto de test, que no es sino un porcentaje que usted asigna al cargar los datos y que permite conocer el estado de la generalización por medio del error que se obtiene al evaluar el conjunto de test en cada iteración. Así mismo la normalización es un concepto que se puede aplicar con mucho éxito en los datos de entrada de la red neuronal.





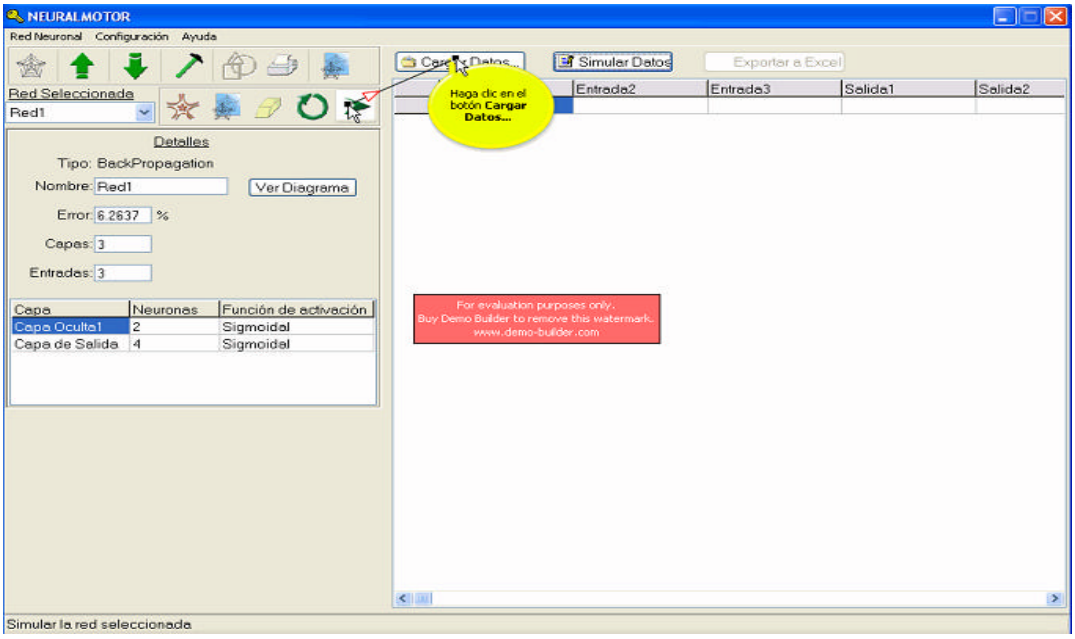


Advertencia: Para que NEURALMOTOR 1.0 no genere errores, los datos cargados deben cumplir con los requisitos. Ver Cargar Datos

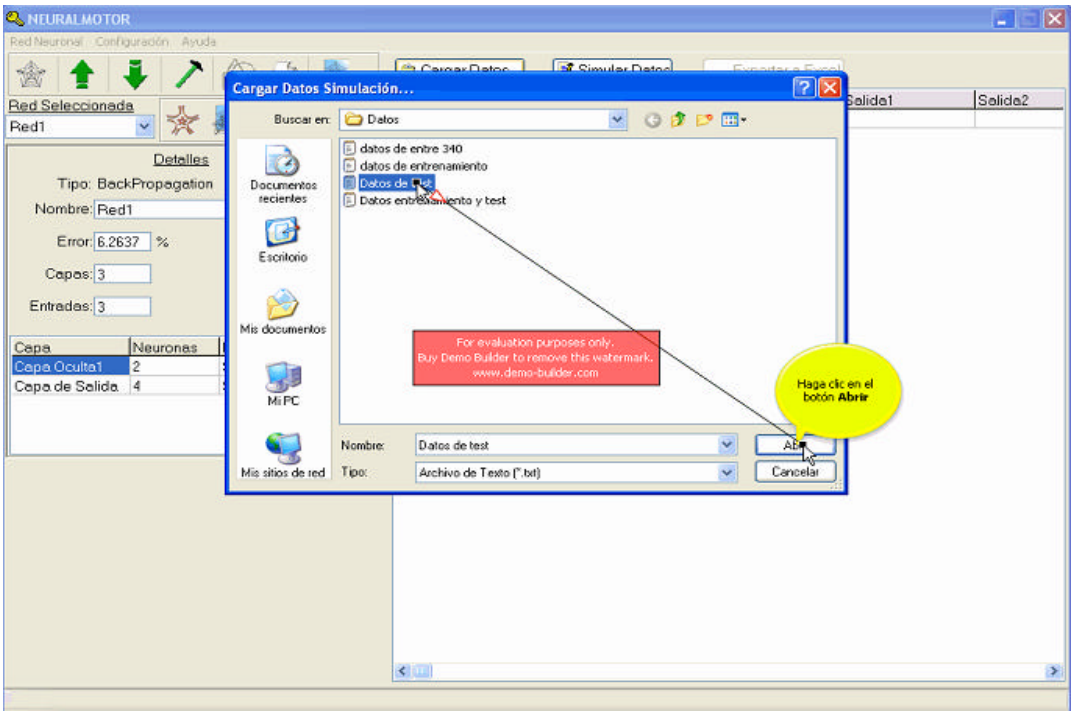


7.1.5.5 SIMULACIÓN DE LOS DATOS

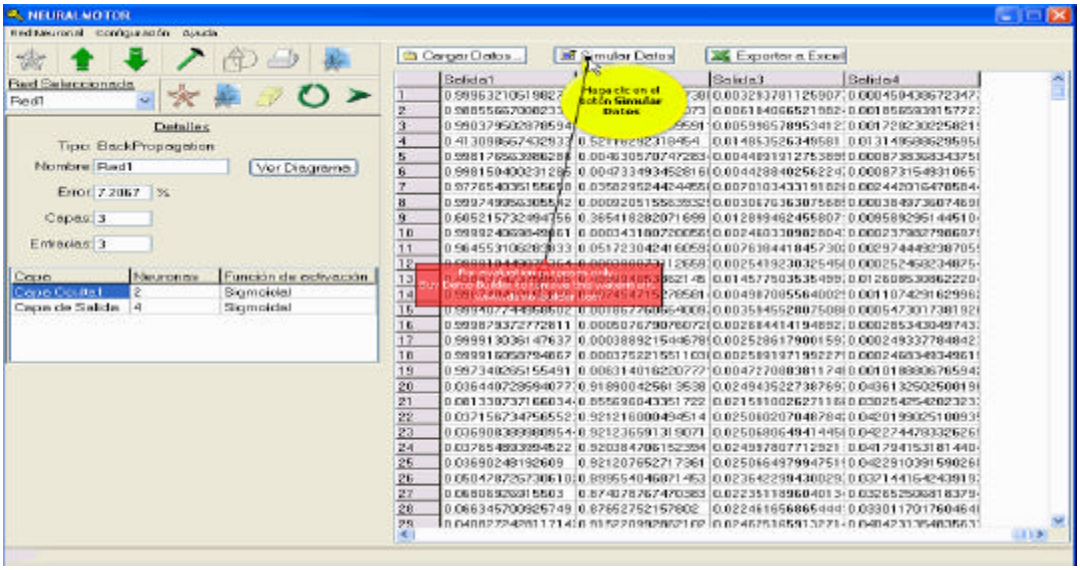
En simulación, se cargan los datos de test desde archivo .txt, y paso siguiente, se selecciona la red a simular, en este punto se activa el botón simulación, inmediatamente las casillas de salida se rellenan con los datos obtenidos por la simulación de la red seleccionada, no olvide que también puede normalizar las entradas antes de realizar la simulación.



Advertencia: Para que NEURALMOTOR 1.0 no genere errores, los datos cargados deben cumplir con los requisitos. Ver Cargar Datos



Después de haber cargado los datos se simulan para obtener los resultados de acierto de la red neuronal artificial, los valores obtenidos en las salidas corresponden a los patrones de respuesta establecidos en el entrenamiento



7.1.6 FUNCIONES AVANZADAS

Estas funciones no son vitales para el manejo, la generación y experimentación con redes en el NEURALMOTOR 1.0, sin embargo, facilitan en buena medida la experimentación con las redes. Estas funciones son puestas a su consideración en los siguientes apartes:

Edición

Normalización

Entradas/Salidas

Exportar a Excel

Reportes

Gráficas en 2D

Separador Decimal

7.1.6.1 EDICCIÓN DE LA RED

Al oprimir el botón derecho del ratón sobre alguna red neuronal, en la lista de redes en el lado izquierdo de la ventana principal, aparece un menú, accesible también por medio de la opción edición en el menú Red neuronal, desde este menú se pueden realizar acciones como mostrar los detalles de la red, Cambiar el nombre, Inicializar, clonar, y borrar la red seleccionada.

7.1.6.3 DETALLES DE LA RED

En detalles se le informa al usuario las características estructurales de la red, numero de capas, numero de neuronas en cada capa y la función de activación.

7.1.6.3 CAMBIAR NOMBRE A LA RED

Con esta opción se puede cambiar el nombre a la red.

7.1.6.4 INICIALIZAR LA RED

En ocasiones, el usuario puede cometer un error al momento de entrenar la red, por lo tanto se hace necesario utilizar la opción de inicializar para comenzar nuevamente el entrenamiento.

7.1.6.5 CLONAR LA RED

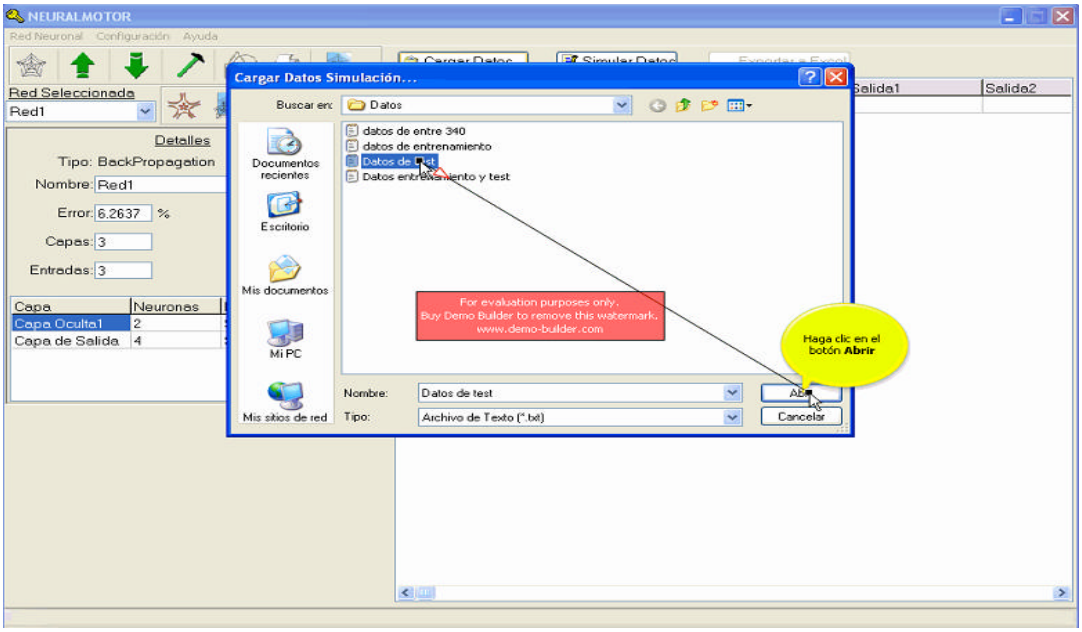
Con esta opción se accede a una red de iguales características

7.1.6.6 ELIMINAR LA RED

Con esta opción se elimina una red existente

7.1.6.7 CARGAR DATOS

NEURALMOTOR 1.0 lee datos de archivos tipo texto de extensión .txt, los cuales deben cumplir con algunos requisitos para que el software no genere errores en el transcurso de la aplicación.



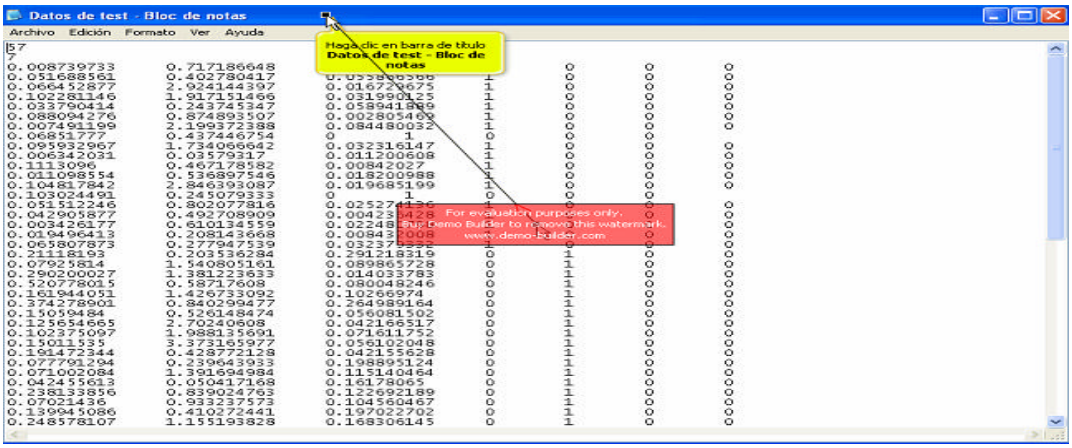
REQUISTOS:

El archivo de extensión .txt en su primera fila debe especificar la cantidad de filas de datos del archivo, Ej. Si se tiene un archivo corresponde a una matriz de 57 por 7, en la primera fila del archivo debe tener el numero cincuenta y sietes y en la segunda fila debe tener el siete que corresponde al numero de columnas del archivo.

Para la lectura correcta de los datos, estos deben venir organizados a una distancia de separación unificada como se muestra en siguiente figura

Antes de procesar los datos se debe verificar que estos tengan correspondencia a las entradas y salidas, pues si se crea una red de 3 entrada y 4 salidas, NEURALMOTOR 1.0 asigna las tres primeras columnas de datos a las entrada y las restantes cuatro a las salidas, pero se puede cometer el error de crear una red de 4 entradas y 3 salidas y software procesara los datos pero la cuarta columna

que corresponde en realidad a salida 1, NEURALMOTOR 1.0 la asumirá como la entrada 4, lo que genera un grueso error de procesamiento. Por lo tanto se recomienda seguir los pasos de Creación de Red y así se evitaran esta clase de confusiones.



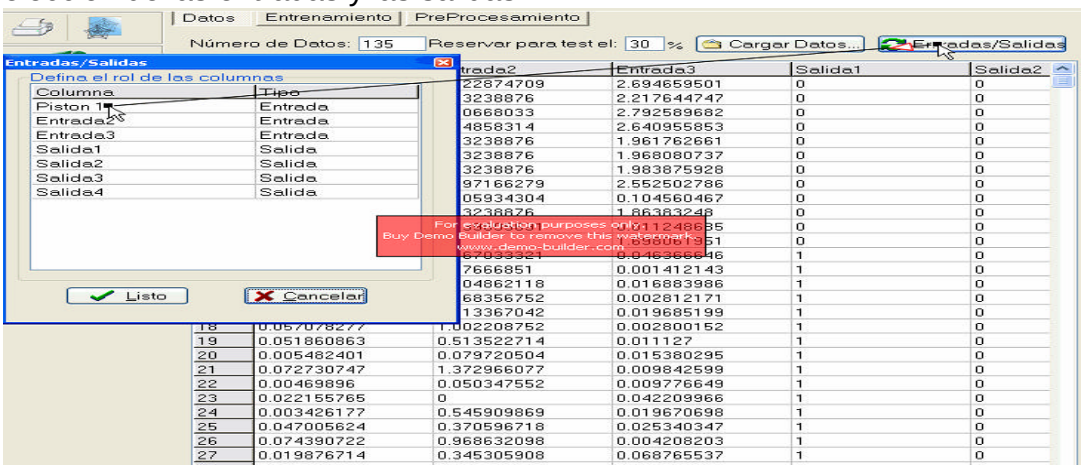
7.1.6.8 NORMALIZACIÓN DE LOS DATOS

Esta funcionalidad es apropiada para minimizar los datos de entrenamiento y simulación con el objetivo de reducir el error de entrenamiento.

7.1.6.9 ENTRADAS Y SALIDAS DE LA RED

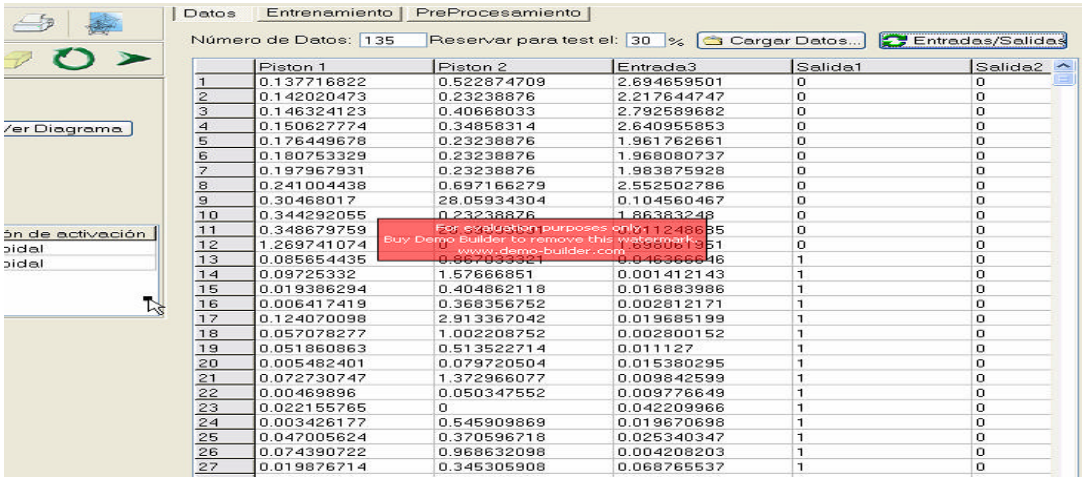
Un conjunto de entrenamiento no necesariamente está sujeto a que las salidas sean siempre las últimas de la fila, aquí es posible escoger que columnas son entradas y que columnas son salidas, para esto oprima el botón localizado en la pestaña de carga de datos, que se habilita cuando hay una red.

Después de seleccionar la opción, aparece una caja de diálogo que permite la elección de las entradas y las salidas.



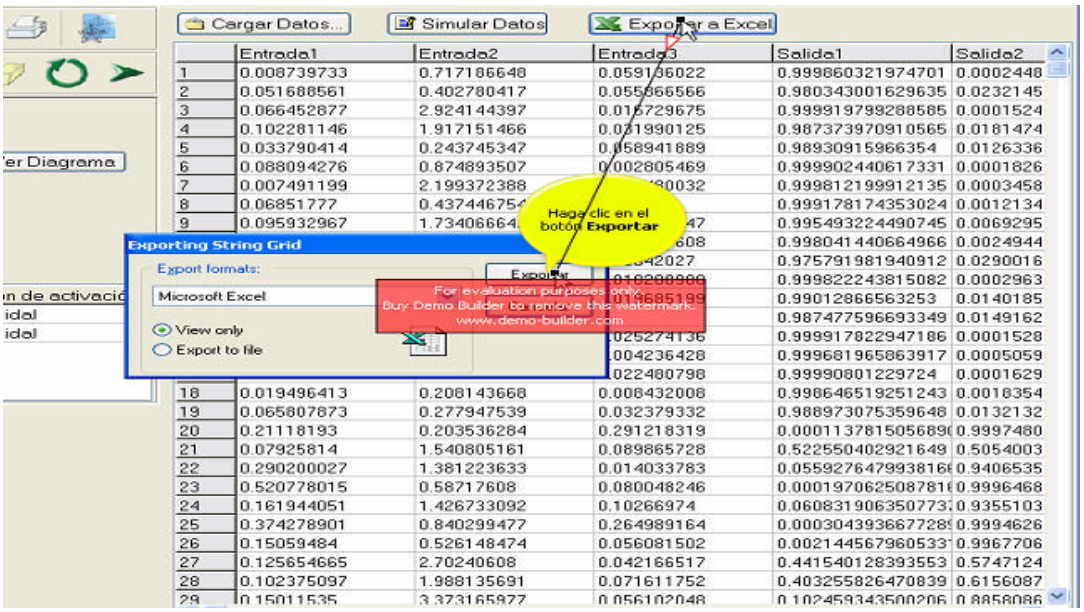
Así mismo es posible asignar nombres a las variables, para permitir una mayor comprensión de las entradas y las salidas de la red, esta opción esta disponible haciendo clic sobre la caja de texto donde se encuentre la variable escogida.

En este ejemplo se modifico los nombres de las dos primeras entradas y se le asigno el nombre de émbolo 1 y émbolo 2 respectivamente



7.1.6.10 EXPORTAR DATOS SIMULADOS A EXCEL

Con los datos de simulación es necesario muchas veces seguir un trabajo de interpretación, en el que se le realizan operaciones, es por esto que esta opción se activa cada vez que se realiza una simulación, para permitir un cómodo trabajo en Excel.



7.1.6.11 GRÁFICAS DEL PROCESAMIENTO DE LOS DATOS

Con esta opción se genera un reporte de gráficas de los datos de simulación.

7.1.6.12 SEPARADOR DECIMAL

Cada computador puede tener su separador decimal en coma (,) o en punto (.) lo que hace difícil la exportación/importación de los datos, afortunadamente en la opción Separador decimal del menú de configuración se puede seleccionar el operador de separación decimal.

7.2 MANUAL DE INSTALACIÓN

NEURALMOTOR 1.0

7.2.1 ACERCA DE LA INSTALACIÓN

Notas generales de instalación:

- Descomprima la carpeta NEURALMOTOR 1.0 y evite modificar los nombres de los archivos para que no se presenten errores de instalación, este software no necesita reiniciar el equipo para que sea reconocido, En Windows 2000 y superior (incluyendo Windows XP) Microsoft ofrece un servicio de instalador automático.
- NEURALMOTOR 1.0 es un software creado con fines educativos, y puede ser usado por la comunidad educativa de la Universidad Industrial de Santander sin previa autorización, si usted no hace parte de esta comunidad debe solicitar la debida autorización a la universidad.
- NEURALMOTOR 1.0 necesita de la instalación previa de Borland Delphi versión 7.0 o superior.

Precauciones

- Garantice que sus equipo tenga instalado borland Delphi 7.0 o superior y los requisitos del sistema

7.2.2 REQUISITOS MÍNIMOS DEL SISTEMA.

- Procesador Intel Pentium a 400 MHz o superior
- Microsoft Windows 2000, XP y vista
- Aproximadamente se requiere de 5 Mb de espacio en disco para una instalación completa.

7.2.3 DESINSTALACIÓN DE ESTE PRODUCTO.

- NEURALMOTOR 1.0 provee la opción de desinstalar automáticamente el software de su computadora.
- Puede realizar el siguiente paso, abra el panel de control y haga doble clic en el icono Agregar o Quitar programas. Seleccione NEURALMOTOR 1.0 De la lista y haga clic en el botón quitar.

==========

7.3 COMPARACION DE RENDIMIENTO DE NEURALMOTOR 1.0 CONTRA REDES IMPLEMENTADAS EN MATLAB 6.5

Con el objetivo de garantizar la predicción de **NEURALMOTOR 1.0**, se decidió comparar su rendimiento con las redes entrenadas y simuladas en Matlab versión 6.5, por lo tanto se ha procedido a diseñar redes de iguales características en su estructura y en iguales condiciones de numero iteraciones y datos de entrenamiento para luego analizar los resultados obtenidos con relación al tiempo de ejecución y error de test en cada uno de los caso.

Para realizar este comparativo hemos propuesto realizar el análisis con datos provenientes de la función AND, XOR y un caso de identificación de sistemas dinámicos no lineal Pc1_90, esta pruebas se va a utilizar un equipo Pentium 4 de 2.1GHz y 512MB de RAM.

- **Función AND**
  En la siguiente tabla se establecen los datos de entrenamiento para la red de una Función AND

Entrada 1	Entrada 2	Entrada 3	Salida
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

- **Función XOR**

  En la siguiente tabla se establecen los datos de entrenamiento para la red de una Función XOR

Entrada 1	Entrada 2	Entrada 3	Salida
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

- **PC1_90**

  En la siguiente tabla se presenta los datos de entrenamiento para la red del escenario PC1_90

	Émbolo 1	Émbolo 3	Émbolo 4	Émbolo 2	Émbolo 1	Émbolo 3	Émbolo 4	Émbolo 2
No	Entrada 1	Entrada 2	Entrada 3	Entrada 4	Salida 1	Salida 2	Salida 3	Salida 4
1	41.35	40.94	40.83	40.81	1	1	1	1
2	40.53	40.78	41.02	39.74	1	1	1	0
3	41.25	40.87	39.21	40.45	1	1	0	1
4	41.05	40.65	39.68	39.48	1	1	0	0
5	40.85	39.01	40.65	41.25	1	0	1	1
6	40.95	39.95	41.12	39.65	1	0	1	0
7	40.35	39.87	39.28	41.12	1	0	0	1
8	41.09	39.24	39.32	39.45	1	0	0	0
9	40.02	39.25	39.25	39.61	0	0	0	0
10	39.75	39.02	39.65	41.25	0	0	0	1
11	39.85	39.45	40.24	39.24	0	0	1	0
12	39.68	39.36	41.36	40.61	0	0	1	1
13	39.02	41.25	39.32	39.24	0	1	0	0
14	39.65	41.15	39.69	41.24	0	1	0	1
15	39.44	41.26	40.46	39.52	0	1	1	0
16	39.65	40.28	41.28	40.68	0	1	1	1

7.3.1 ANÁLISIS DE RESULTADOS

A continuación se presenta los resultados obtenidos en el entrenamientos y la simulación de datos de una señal de entrada, y que soportan nuestro proceso comparativo de NEURALMOR 1.0 contra Matlab 6.5.

- **Función AND**

Tipo Red / Parámetro	NEURALMOTOR 1.0		MATLAB 6.5	
	Backpropagation	RBF	Backpropagation	RBF
Iteraciones	1000	1000	1000	-
Numero de capas	3	3	3	3
Neuronas en la capa oculta	4	8	4	8
Error de entrenamiento	0.41%	0.025%	0%	-
Tiempo de ejecución de entrenamiento	0.34 Segundos	0.41 Segundos	0.45 Segundos	-

- **Función XOR**

Tipo Red / Parámetro	NEURALMOTOR 1.0		MATLAB 6.5	
	Backpropagation	RBF	Backpropagation	RBF
Iteraciones	1000	1000	1000	
Numero de capas	3	3	3	3
Neuronas en la capa oculta	4	8	4	8
Error de entrenamiento	0.032 %	0.015 Segundos	0	-
Tiempo de ejecución de entrenamiento	0.37 Segundos	0.41 Segundos	0.22 Segundos	-

- **Pc1_340**

Tipo / Red / Parámetro	NEURALMOTOR 1.0		MATLAB 6.5	
	Backpropagation	RBF	Backpropagation	RBF
Iteraciones	1000	1000	1000	
Numero de capas	3	3	3	3
Neuronas en la capa oculta	4	160	4	-
Error de entrenamiento	0.15%	0.42	13.24%	-
Tiempo de ejecución de entrenamiento	0.56 Segundos	0.78 Segundos	1.26 Segundos	-

La conclusión de este comparativo es que Matlab 6.5 le gana a NEURALMOTOR 1.0, en el error de entrenamiento, por alcanza valores mas pequeños, pero recuerde que el error no es directamente proporcional al rendimiento, por lo tanto la diferencia no es sustancial; en cambio, NEURALMOTOR 1.0 proporciona funcionalidades de cargar datos, Gráficas, reportar resultados, generación de datos de entrenamiento y simulación etc. Que hace que el análisis de un MCI se realice en muy poco tiempo comparado con el tiempo que tomaría hacer el mismo proceso utilizando matlab 6.5, lo que si es representativo. Para terminar se recomienda utilizar NEURALMOTOR 1.0 que le ayudara hacer su trabajo mas fácil y en menos tiempo.

BIBLIOGRAFÍA

1. GARRIDO, Santiago. Identificación, estimación y control de sistemas no lineales mediante RGO. Universidad Carlos III, Leganés, 1999.
2. HILERA, José y MARTINEZ, Víctor. Redes neuronales artificiales. Alfaomega. Madrid. España, 2000.
3. PRESSMAN, Roger. Ingeniería del Software. Un enfoque práctico. 6ª edición. En este libro se encuentra información acerca de Ingeniería del Software.
4. GALLARDO, José. Diseño e implementación de un sistema de control avanzado para procesos complejos. Universidad Católica del Norte. Antofagasta. Chile.
5. IZAURIETA, Fernando y SAAVEDRA, Carlos. Redes neuronales artificiales. Departamento de física, Universidad de Concepción, Chile.
6. ACOSTA, Jesús, FERNANDEZ, Joseph y BECERRRA, Lenin. Aplicación de las Redes Neuronales en la Identificación de un Sistema no lineal. Departament d'Enginyeria de Sistemes, Automàtica e Informàtica Industrial. Universitat Politècnica de Catalunya. Barcelona, España. Depto. de Ingeniería Electrónica Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO). Barquisimeto, Venezuela.
7. CRUZ, Alejandro. Tesis de maestría *Estabilidad de entrada-estado (ISS) para identificación con redes neuronales dinámicas.* Departamento de control automático Instituto Politécnico Nacional, México, 2003.
8. BLANCO, V. Ricardo. Extracción de Reglas de Redes Neuronales Artificiales. Departamento de sistemas informáticos y computación, Universidad Politécnica de valencia, España,
9. MELIN, O. Elba y CASTILLO, L. Oscar. Redes neuronales y sus aplicaciones. Instituto Tecnológico de Tijuana, mexico.
10. REYES FIGUEROA, Juan Carlos, Metodología para la Generación de Reglas Borrosas Y Ajuste Adaptativo de Funciones de Pertenencia por medio de una Red Neuronal, Universidad Industrial De Santander (2007)
11. CHACÓN VELASCO, Jorge Luis, Identificación de un Motor Diesel In: IX Congreso Latinoamericano de Control Automático, 2000, Cali, Colombia., Memorias IX Congreso Latinoamericano de Control Automático. Cali, Colombia: Asociación Colombiana de Automática-ACA, 2000. v.1. p.1 - 5
12. J. rumbaugh, I. Jacobson, G.Booch, El Lenguaje Unificado de Modelado Manual de Referencia, Pearson educación, Madrid, 2000
13. F. Aminzadeh, J. Barhen, and N.B. Toomarian, "Estimation of Reservoir Parameter using a Hybrid Neural Network", *Journal of Petroleum Science and Engineering*, vol. 24, no. 1, 1999, pp. 49-56.
14. H.H. Nguyen and C.W. Chan, "Petroleum Production Prediction: A Neural Network Approach", Proceedings of 5th International Joint Conference on Engineering Design and Automation 2001 (EDA 2001), 5-8 August 2001, Las Vegas, USA, pp.85-90, 2000

15. GRECH, Pablo, Introducción a la Ingeniería. Un enfoque a través del diseño.Editorial Prentice Hall, Bogotá, 2001.
16. Dawson, Martín G. El proyecto de fin de carrera en ingeniería informática: una
17. guía para el estudiante. Editorial Prentice Hall. Madrid. 2002
18. H.H. Nguyen and C.W. Chan, "A Comparison of Data Preprocessing Strategies for Neural Network Modeling of Oil Production Prediction", Proceedings of the Third IEEE International Conference on Cognitive Informatics, (ICCI 2004)
19. Bharath Rao. Multiphase flow models: Range of applicability. www.ctes.com.1998.
20. Miguel Garre et al. Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software. Universidad de Alcalá. Madrid.
21. Osman, E.A.: "Artificial Neural Networks Models for Identifying Flow Regimes and Predicting Liquid Holdup in Horizontal Multiphase Flow", paper SPE 68219, marzo 2001.
22. Mohaghegh, S.D.: "*Virtual Intelligence Applications in Petroleum Engineering: Part 1–Artificial Neural Networks,*" paper SPE 58046,
23. Dukler, Wicks & Cleveland, A.I.Ch.E. Journal, 10, 44. 1974
24. Beggs H.D. and Brill, J.P.: "A Study of Two Phase Pressure Drop In Inclined
25. Pipes" JPT, May 1973 (Cap. 3).
26. http://www.epsem.upc.edu/~esaii/assign/ident/Tema%201.pdf, Revisado 13 de marzo de 2007
27. http://www.epsem.upc.edu/~esaii/assign/ident/Tema%204.pdf, Revisado 13 de marzo de 2007
28. http://es.wikipedia.org/wiki/Diagrama_de_secuencia, revisado 15 de abril de 2007
29. http://www.gc.ssr.upm.es/inves/neural/ann2/anntutorial.html Revisado el 12 de mayo del 2007.
30. http://ingenieria.udea.edu.co/investigacion/mecatronica/mectronics/redes.htm Revisado el 12 de mayo del 2007.
31. http://www.tecnociencia.org/n/205/redes-neuronales-artificiales/ Revisado el 20 de mayo del 2007
32. http://www.gc.ssr.upm.es/inves/neural/ann2/concepts/taxonomy.htm Revisado el 20 de mayo del 2007