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

RED NEURONAL WAVELET RADIAL CON ALGORITMO PSO PARA LA ESTIMACIÓN DE COSTOS

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

Marzo 2013

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

RED NEURONAL WAVELET RADIAL CON ALGORITMO PSO PARA LA ESTIMACIÓN DE COSTOS

Por
Oscar Javier Mena Muñoz
Carrera
Ingeniería Civil Informática

Profesor Guía
Nibaldo Rodriguez Agurto
Profesor Correferente
Wenceslao Palma Muñoz
Marzo 2013

Dedicatoria

Dedicado a Oscar Mena y Adriana Muñoz, mis padres, y Ninoska Mena, mi hermana, por todo su apoyo, cariño y esfuerzo a lo largo de estos años, porque siempre han estado conmigo y a toda la familia, por su constante apoyo.

Agradecimientos

A mi familia por su apoyo incondicional y a mi profesor guía, Nibaldo Rodriguez, por el apoyo no solo en el proyecto, sino a lo largo de toda la carrera.

RESUMEN

La estimación de costos es un factor clave durante las primeras etapas de desarrollo de productos, incluidas las tuberías. En las primeras fases de desarrollo, se decide casi el 80% del costo total del producto, por lo que resulta primordial tener un buen estimador de estos costos.

Para realizar dicha estimación, se utilizó un modelo de predicción basados en redes neuronales wavelet radial con distintas variaciones del algoritmo de optimización por enjambre de partículas que permitieran estimar lo más exactamente posible los costos de elaboración de las tuberías.

El modelo IW-PSO fue el que obtuvo los mejores resultados, obteniendo un MAPE de 12.9274 y un valor de coeficiente de correlación (R^2) de 0.9611 en comparación con el modelo QPSO y UPSO.

Palabras claves: *Redes Neuronales Artificiales, Algoritmo por Optimización de Enjambre de Partículas (PSO), Estimación, Costo, Tuberías, Aprendizaje, Optimización, Wavelet.*

ABSTRACT

Cost estimation is a key factor in the early stages of product development, including pipelines. In the early stages of development, it was decided almost 80% of total product cost, so it is essential to have a good estimate of these costs.

To make this estimate, we used a prediction model based on wavelet radial neural networks with different variations of the optimization algorithm particle swarm that allowed most accurately estimate development costs of the pipes.

The IW-PSO model was the one that obtained the best results, obtaining a MAPE of 12.9274 and a value of the correlation coefficient (R^2) of 0.9611 compared with QPSO and UPSO model.

Keywords*: Artificial Neural Networks, Algorithm for Particle Swarm Optimization (PSO), Estimating, Cost, Pipes, Learning, Optimization, Wavelet.*

ÍNDICE DE CONTENIDOS

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

ABREVIACIONES Y NOMENCLATURA

PSO	Particle Swarm Optimization (optimización por enjambre de partículas)
RNA	Artificial Neuronal Network (red neuronal artificial)
MAPE	Mean Absolute Percentage Error (error porcentual absoluto medio)
IA	Inteligencia Artificial
RNWR	Red Neuronal Wavelet Radial
SI	Swarm Intelligence (inteligencia colectiva)
UPSO	Unified Particle Swarm Optimization (optimización por enjambre de partículas unificada)
QPSO	Quantum Particle Swarm Optimzation (optimización por enjambre de partículas quántica)
PSO-IW	Particle Swarm Optimization Inertia Weight (optimización por enjambre de partículas- peso de inercia)

CAPÍTULO 1

INTRODUCCIÓN

El problema de estimación de costos es un factor clave durante las fases de desarrollo de productos, incluidos los productos manufacturados. Las aproximaciones iniciales de costo como funciones de un conjunto de características generales ayudan a los diseñadores en decisiones como la selección de material, selección de procesos de producción y también las características morfológicas del producto.

Algunos estudios han demostrado que el gran potencial de reducción de costos se da en las etapas más tempranas, donde casi el 80% del costo del producto es decidido. El tomar decisiones erróneas en esta etapa es extremadamente costoso, en particular en los procesos de desarrollo debido a posibles modificaciones o alteraciones de proceso, ya que son más caras mientras más tarde ocurran en el ciclo de desarrollo, y suelen ocurrir, si la estimación esta mala.

Los métodos utilizados para realizar las estimaciones se clasifican en técnicas cualitativas y cuantitativas. Las técnicas cualitativas se basan en la experiencia que tienen los que realizan la estimación de costos, y se dividen en técnicas analógicas e intuitivas. Las técnicas cuantitativas se subdividen en técnicas paramétricas y analíticas.

Actualmente existen diversos estudios sobre la estimación de costos que utilizan distintas técnicas como regresión lineal [Ru, 2009], media móvil [Broersen, 2002], inteligencia artificial (IA) [Finnie, 1996], entre otras, y son estas últimas, las de inteligencia artificial, las que intentando imitar el comportamiento de los expertos, han tomado mayor importancia. La principal técnica que utiliza la IA en la estimación de costos es el razonamiento basado en casos, similar a la técnica cualitativa analógica. Las redes neuronales artificiales (RNA), una rama de la IA, son una técnica utilizada en la investigación de la estimación de costos.

Este proyecto desarrolla y evalúa un modelo de estimación para los costos de fabricación de intercambiadores de calor de carcasas y tubos utilizando RNA con algoritmos de optimización por enjambre de partículas (PSO), lo que permitirá predecir el costo de manufactura de manera bastante exacta. Además, se pretende esclarecer las características más significativas e influyentes en los costos de los productos.

1.1 OBJETIVOS GENERALES

Desarrollar y evaluar un modelo de estimación de costos mediante el uso de RNA con algoritmo PSO para lograr minimizar los costos de producción de manufactura**.**

1.2 OBJETIVOS ESPECÍFICOS

- Comprender y dar a conocer el marco teórico de las Redes Neuronales Artificiales y Optimización por Enjambre de Partículas.
- Estimar los parámetros del modelo basado en una Red Neuronal Artificial, utilizando como función de transferencia la función Wavelet Radial.
- Implementar y evaluar el modelo propuesto.

1.3 Organización del Texto

A continuación se detallaran los temas a tratar en este proyecto. Como complemento de la introducción, se detallaran los objetivos -generales y específicos- del trabajo y el marco de trabajo. En el capítulo 2 se hablará de los Costos de producción, algunas de sus técnicas antiguas y como se plantea enfrentar el problema a resolver. En el capítulo 3 se explicarán las RNA, de donde se basan para su existencia, un poco de historia, los distintos modelos que fueron apareciendo y además hablaremos de la función de transferencia que ocupara la red neuronal y finalmente el modelo completo de la red más la función, conocida como red neuronal wavelet radial. En el capítulo 4 se profundizara en el algoritmo PSO, de donde proviene esta idea, en que consiste y algunos modelos de estos. En el capítulo 5 veremos las distintas pruebas de training y testing para cada uno de los modelos, además de las distintas métricas que nos permitirán ir determinando la mejor configuración para cada modelo. Finalmente se dará una conclusión del trabajo realizado y lo que se obtuvo en el proyecto.

CAPÍTULO 2

COSTOS DE PRODUCCIÓN

En el presente capítulo se hará una revisión de la literatura sobre la estimación de costos en la producción, comenzando desde la definición hasta las técnicas que actualmente se utilizan.

2.1 ESTIMACIÓN DE COSTOS DE PRODUCCIÓN

La estimación del costo que tendrá un producto manufacturado puede ser definida como la determinación de costos de industria por un componente, sub-ensamble, o ensamble final, basado en un plan de proceso definido de manufactura. La función de planificación de manufactura es la determinación de un desarrollo de operaciones utilizado en el proceso, equipos y herramientas para producir cualquier producto manufacturado. Para llegar al precio de venta del producto y estimar el costo de fabricación, los costos industriales y márgenes de utilidad también deben ser considerados.

El costo directo es la suma del trabajo directo con el material directo. El costo directo es sumado con los costos indirectos de fabricación. El costo total del producto es obtenido al sumarle el costo de venta, los gastos generales y administrativos, costos de desarrollo, y contingencias del costo de fabricación.

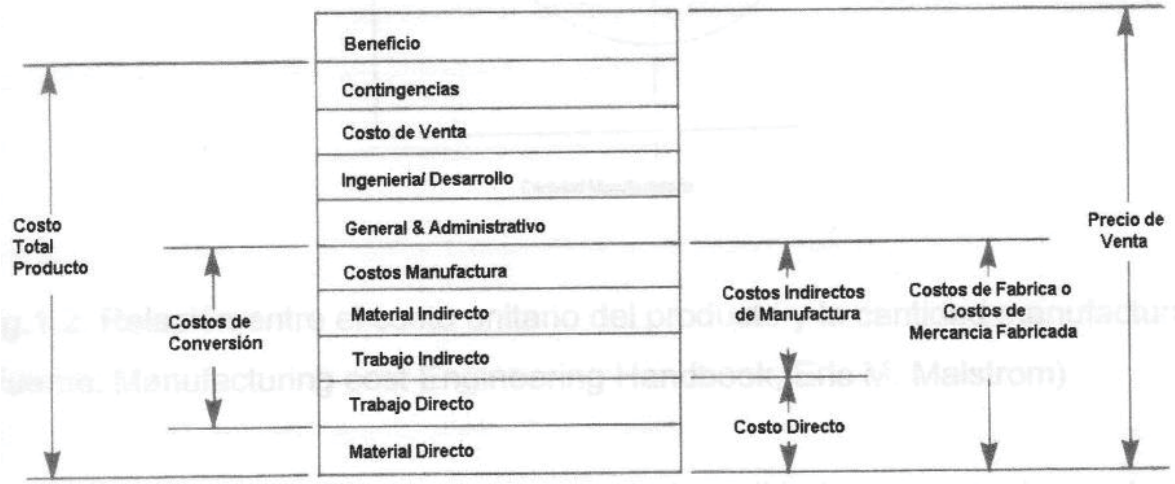

ILUSTRACIÓN 1: ESTRUCTURA DEL PRECIO DE VENTA

2.1.1 RELACIÓN COSTO-PRECIO

El costo unitario del producto es influenciado por la cantidad producida y por la calidad que tiene el producto terminado. El costo unitario del producto disminuye mientras la cantidad de producción se eleva hasta que la capacidad de manufactura llegue al máximo. Si la producción excede a la capacidad, el costo unitario empieza a subir, pero esta situación sucede muy pocas veces. Si la demanda del producto crece, el equipamiento y los procesos están usualmente modificados para incrementar la capacidad.

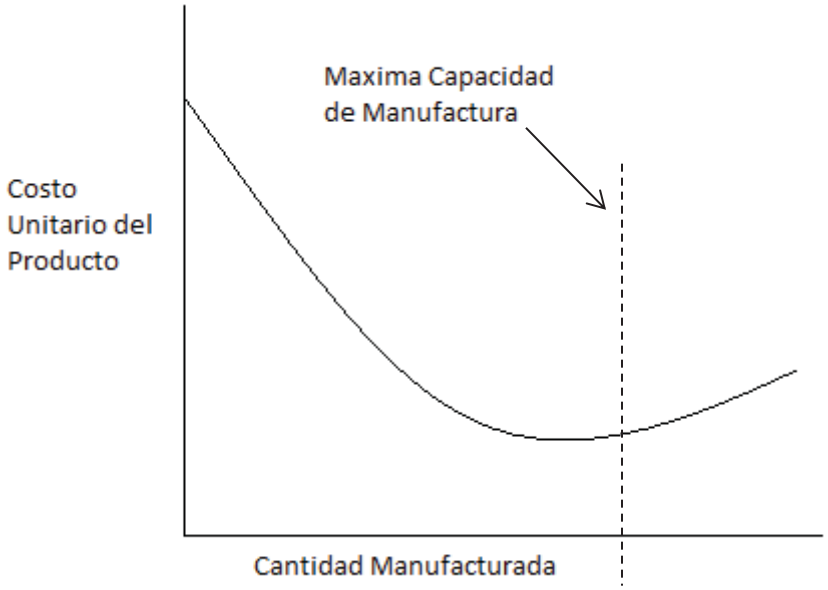

ILUSTRACIÓN 2: RELACIÓN COSTO CANTIDAD

Si el nivel de calidad se ve modificado el costo unitario crece, ya que la garantía del producto está incluida en el total del costo por unidad, en cambio niveles de calidad bajos tendrán como resultado un decremento del costo por unidad del producto.

ILUSTRACIÓN 3: RELACIÓN COSTO CALIDAD

2.1.2 EFECTIVIDAD PARA LA ESTIMACIÓN DE COSTOS

La efectividad, en la estimación de costos, puede ser definida como el acercamiento al valor real. El propósito de la estimación de costos no es dar la información del costo exacto, sino que dar figuras del valor teniendo una alta probabilidad de estar en un rango

aceptable. Si una estimación tiene un 10% de efectividad, se espera que el costo actual del producto este dentro del 10% de su valor estimado.

Existen dos tipos de errores graves que influyen en la efectividad para la estimación de costos: están los **errores no intencionales**, que son un mal juicio o confusión del estimador y los **errores sistemáticos**, que asumen que dadas las mismas condiciones, los costos tendrán la misma magnitud y dirección.

2.2 MODELOS DE ESTIMACIÓN TRADICIONALES

Tradicionalmente, la estimación de costos ha sido la evaluación de todos los costos directos e indirectos distribuidos en las actividades que componen el alcance del proyecto. Los objetivos de la estimación de costos son definir la magnitud económica del proyecto, confirmar el monto cotizado por terceros y además sirve de base para la planificación del proyecto y su flujo de caja.

Dentro de los modelos tradicionales se distinguen tres tipos de estimaciones [Araya, 2007]: la **estimación rápida**, que se basa en un solo parámetro de costos. Estos se establecen a partir de los datos de costos históricos en procesos anteriormente realizados, que tienen semejante fabricación. La estimación es netamente basada en la experiencia que el estimador posee. La efectividad que un estimador provee no logra superar el 30%. También está la estimación **preliminar**, que se basa en especificaciones tempranas de diseño del producto. Esta técnica de estimación puede usar un costo real pasado de un producto similar o con piezas comparables de estimaciones previas. Si la información de un producto no se encuentra, el juicio debe ser apoyado por un estimador con experiencia. Ambas estimaciones, rápida y preliminar, serán buenos métodos dependiendo de la experiencia que posea el estimador, aunque pueden ser mejoradas eficientemente a un costo mínimo. Por último está la estimación **detallada**, que se basa en un diseño completo de especificaciones. Estas son las más efectivas y usan estándares que han sido comprobados sobre un gran periodo de tiempo, tomando en cuenta todos los costos asociados en la elaboración del producto. La estimación detallada como única desventaja tiene el tiempo que se demora en realizar la estimación, que en comparación a las otras dos, es mucho mayor.

En la actualidad existen varias técnicas que buscan estimar los costos de producción, entre ellas se encuentran la lógica difusa y las redes neuronales artificiales. Estas últimas serán el método a utilizar para el desarrollo de este proyecto, y durante el capítulo siguiente se hará el estudio del estado del arte.

CAPÍTULO 3

REDES NEURONALES ARTIFICIALES

Uno de los grandes enigmas que han preocupado al hombre desde tiempos ancestrales es el de su propia naturaleza, como por ejemplo: ¿cuáles son las características que nos hacen humanos?, ¿qué tiene el hombre que no tienen los demás animales?, ¿que nos hace únicos entre todos los seres vivos? Éstas incógnitas han venido asociadas con la inteligencia, pues dentro de la naturaleza humana está el ser inteligente, la cual es una característica que discrimina absolutamente a nuestra especie. A medida que la ciencia y la tecnología han ido avanzando, el objetivo se ha ido perfilando: uno de los retos más importantes a los que se enfrenta el ser humano de nuestra generación es el de la construcción de sistemas inteligentes, o conocido también como Inteligencia Artificial (IA de ahora en adelante).

En la IA se distinguen 2 grandes áreas. IA Simbólica, que se ocupa de la construcción de sistemas con características que se puedan definir como inteligentes. En este caso, se define el problema a resolver y se diseña el sistema capaz de resolverlo siguiendo esquemas prefijados por la disciplina. La otra gran área, la IA Subsimbólica, no realiza diseños a alto nivel capaces de resolver problemas utilizando las técnicas de la disciplina, sino que se parte de sistemas genéricos que van adaptándose y construyéndose hasta formar por si mismos un sistema capaz de resolver el problema. La perspectiva subsimbólica trata de estudiar los mecanismos físicos que nos capacitan como seres inteligentes, frente a los programas de computador clásicos que son simples autómatas que obedecen órdenes muy concretas. El mecanismo fundamental que capacita a los seres vivos para la realización de tareas sofisticadas no preprogramadas directamente es el sistema nervioso. Desde este punto de vista la perspectiva subsimbólica trata de estudiar los mecanismos de los sistemas nerviosos, del cerebro, así como su estructura, funcionamiento y características lógicas, con la intención de diseñar programas basados en dichas características que se adapten y generen sistemas capaces de resolver problemas. Es en este campo donde se encuadran las Redes Neuronales Artificiales (RNA desde ahora)

Idealmente, el objetivo de las RNA es llegar a diseñar maquinas con elementos neuronales de procesamiento paralelo, de modo que el comportamiento global de esa red "emule", de la forma más fiel posible, los sistemas neuronales de los animales.

3.1 FUNCIONAMIENTO DE UNA RED NEURONAL

El aparato de comunicación neuronal de los animales y del hombre, formado por el sistema nervioso y hormonal, en conexión con los órganos de los sentidos y órganos efectores (músculos, glándulas), tiene la misión de recoger informaciones, transmitirlas y elaborarlas, en parte también almacenarlas y enviarlas de nuevo en forma elaborada. Este sistema se compone de 3 partes:

- Los receptores, que están en las células sensoriales, recogen las informaciones en forma de estímulos tanto del ambiente como del interior del organismo.

- El sistema nervioso, que recibe las informaciones, las procesa, en parte las almacena y las envía en forma elaborada a los órganos efectores y a otras zonas del sistema nervioso.
- Órganos efectores, que reciben la información y la interpretan en forma de acciones motoras, hormonales, etc.

El elemento básico en términos de estructura y funcionamiento del sistema de comunicación neuronal, es la neurona. Éstas pueden definirse como una unidad de procesamiento que recibe un estímulo eléctrico de otras neuronas principalmente a través de su árbol dendrítico. El estímulo eléctrico recibido al pasar de un cierto umbral causa que la neurona a su vez envié una señal eléctrica a través de su axón a otras sucesivas neuronas.

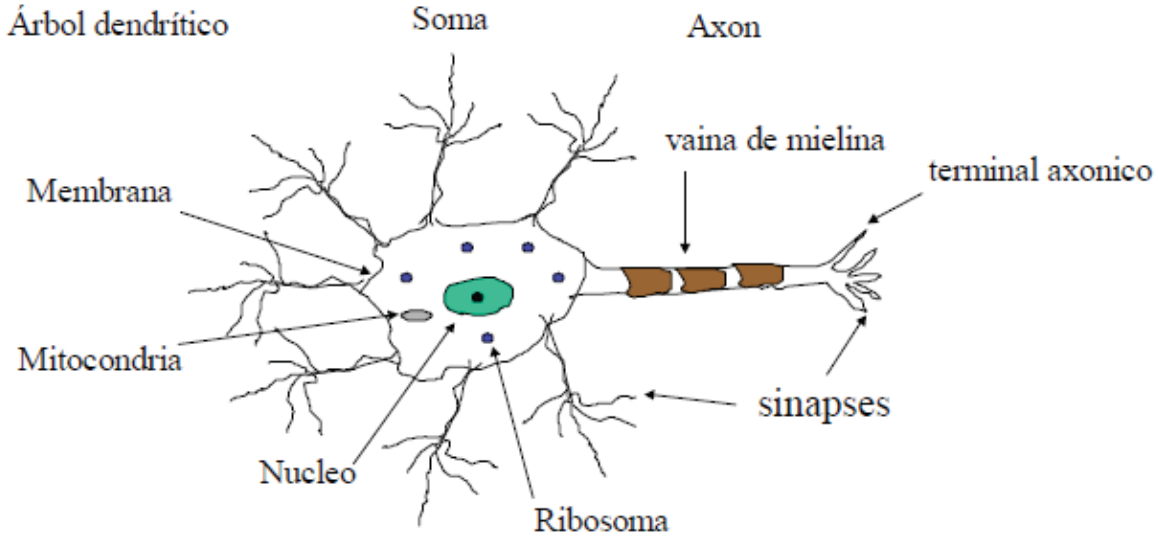

ILUSTRACIÓN 4: ÁRBOL DENDRÍTICO

Las neuronas comprenden generalmente cinco funciones parciales:

- Las neuronas recogen la información que llega a ellas en forma de impulsos procedentes de otras neuronas o receptores.
- Integran la información en un código de activación propio de la célula.
- Transmiten la información codificada en forma de frecuencia de impulsos a través de su axón.
- A través de sus ramificaciones el axón efectúa la distribución espacial de los mensajes.
- En sus terminales transmite los impulsos a las neuronas subsiguientes o a las células efectoras.

El mecanismo se puede describir a través de la siguiente dinámica: Las sinapsis recogen información electro-química procedente de las células vecinas a las que la célula en cuestión está conectada; esta información llega al núcleo donde es procesada hasta generar una respuesta que es propagada por el axón. Más tarde, la señal se ramifica y llega a dendritas de otras células a través de lo que se denomina sinapsis. La sinapsis son los elementos de unión entre axón y dendritas, y ocurren en el espacio sináptico. El espacio sináptico es un espacio líquido donde existen determinadas concentraciones de elementos ionizados, normalmente iones de sodio y potasio. Estos iones hacen que el espacio intersináptico posea ciertas propiedades de conductividad que activen o impidan, en cierto

grado, el paso del impulso eléctrico. De esta forma las sinapsis se convierten en potenciadores o inhibidores de la señal procedente de los axones, actuando como aislantes o amplificadores a conveniencia.

Algunas de las células nerviosas, los receptores, reciben información directamente del exterior -a esta información se le llama estimulo-. Los estímulos son el mecanismo de contacto de un organismo con el mundo exterior. Existen terminaciones nerviosas en casi todas las partes del organismo que se encargan de recibir la información visual, táctil, auditiva, etcétera. Esta información, una vez elaborada, pasa a ser tratada como el resto de la información del sistema nervioso y es convertida en impulsos electro-químicos. Son estos impulsos los que, básicamente, ponen en funcionamiento la red neuronal del sistema nervioso, la cual propaga todas las señales asincrónicamente hasta que llega a los efectores, los órganos, glándulas, músculos; que son capaces de transformar la información recibida en acciones motoras, hormonales, etcétera. Así es como un organismo recibe, procesa y reacciona ante la información recibida del exterior.

3.2 Red Neuronal Artificial

El objetivo de la RNA es modelar el funcionamiento del sistema nervioso de los animales. Para ellos se han ido desarrollando distintos modelos, con distintos niveles de complejidad. Para lograr esto se explicaran las partes que componen a una RNA y luego un análisis de las RNA más significativas en la historia, hasta llegar a la que será ocupada para el desarrollo del proyecto y una pequeña revisión de las redes recurrentes.

Según [Haykin, 1999], el modelo de una neurona presenta 3 elementos básicos:

- Una serie de **sinapsis**, donde cada una está caracterizada por un **peso** el cual puede tomar valores negativos como positivos.
- Un **sumador** que suma las señales de entrada pesadas por la respectiva sinapsis de la neurona.
- Una **función de activación** para limitar la amplitud de la salida de la neurona. Cabe destacar que ésta corresponde a una función no lineal que normaliza el estado interno de la neurona a un intervalo cerrado [0,1] o alternativamente [-1,1]. Ejemplos de función de activación serían: la función escalón, sigmoidea, gaussiana, polinomial, entre otras.

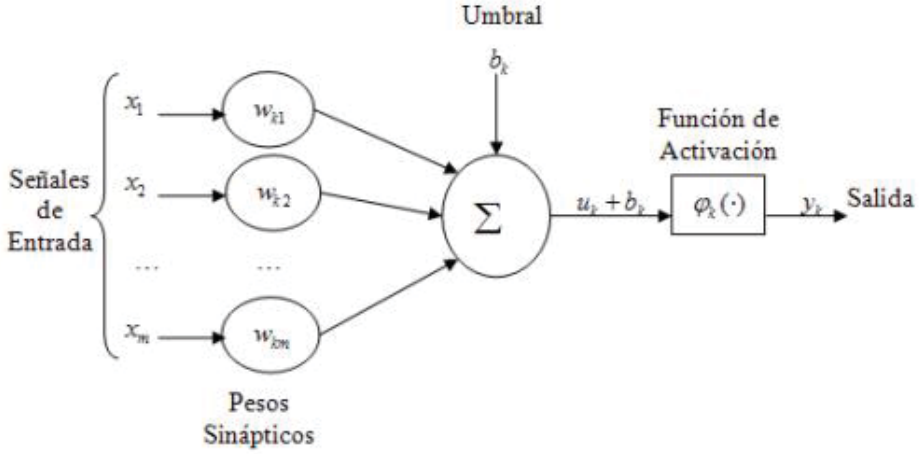

Ilustración 5: Representación de una Neurona Básica

El modelo neuronal presentado en la ilustración 5 presenta además un valor numérico conocido como **umbral** el cual se denomina por b_k y corresponde a un valor opcional que puede aumentar o disminuir el estado interno de la neurona dependiendo si es positivo o negativo, respectivamente.

En términos matemáticos, una neurona k puede ser descrita de la siguiente manera:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \quad e \quad y_k = \varphi(u_k + b_k) \tag{3.1}$$

La arquitectura de la red es la manera en que las células se conectan entre sí. La estructura básica de interconexión entre células es la de la red multicapa, ya que se trata de una estructura típica de implementación del paradigma conocido como Retropropagación. El primer nivel lo constituyen las células de entrada, a continuación hay una serie de capas intermedias, llamadas capas ocultas, cuyas unidades responden a rasgos particulares que pueden aparecer en los patrones de entrada. La salida de estas unidades sirve como salida de toda la red.

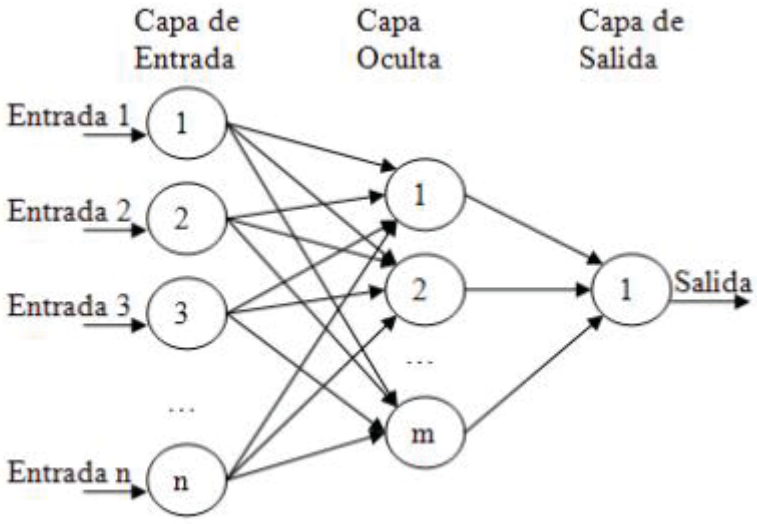

ILUSTRACIÓN 6: ARQUITECTURA BÁSICA DE UNA RED NEURONAL

La parte más importante de una RNA es el aprendizaje. El esquema de aprendizaje de una red es lo que determina el tipo de problemas que será capaz de resolver. Las RNA son sistemas de aprendizaje basados en ejemplos. La capacidad de una red para resolver un problema estará ligada de forma fundamental al tipo de ejemplos de que dispone del proceso de aprendizaje. El aprendizaje de la RNA consiste en la determinación de los valores precisos de los pesos para todas sus conexiones, que la capacite para la resolución eficiente de un problema. Los criterios de convergencia, para la finalización del proceso de aprendizaje son los siguientes: mediante un número fijo de ciclos, cuando el error descienda por debajo de una cantidad prestablecida y cuando la modificación de pesos sea irrelevante.

Dependiendo del esquema de aprendizaje y del problema a resolver, se pueden distinguir tres tipos de esquemas de aprendizaje:

- Aprendizaje supervisado: en este tipo de esquemas, los datos del conjunto de aprendizaje tiene dos tipos de atributos: los datos propiamente dichos y cierta información

relativa a la solución del problema. El esquema de aprendizaje supervisado utilizara esta información para modificar las conexiones. La manera más habitual de modificar los valores de los pesos es la representada en la ilustración 7. Cada vez que un ejemplo es introducido y se procesa para obtener una salida, dicha salida se comparar con la salida que debería haber producido. La diferencia entre ambas influirá en cómo se modificaran los pesos. Para este tipo de aprendizaje, "*se dice que hay un profesor externo encargado de determinar si la red se está comportando de forma adecuada, mediante la comparación entre la salida producida y la esperada, y de actuar en consecuencia modificando apropiadamente los valores de los pesos"* [Isasi y Galván, 2004].

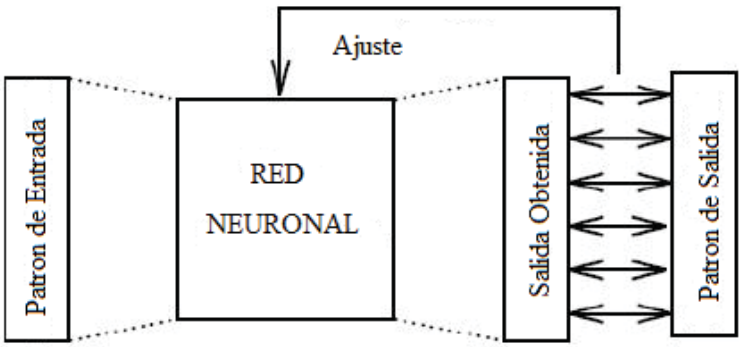

ILUSTRACIÓN 7: APRENDIZAJE SUPERVISADO

- Aprendizaje no supervisado: en este aprendizaje los datos del conjunto de aprendizaje sólo tienen información de los ejemplos, y no hay nada que permita guiar el proceso de aprendizaje. En este caso no existe profesor externo que determine el aprendizaje. La red modificara los valores de los pesos a partir de información interna. Cuando se utiliza aprendizaje no supervisado, la red trata de determinar características de los datos del conjunto de entrenamiento: rasgos significativos, regularidades o redundancias.

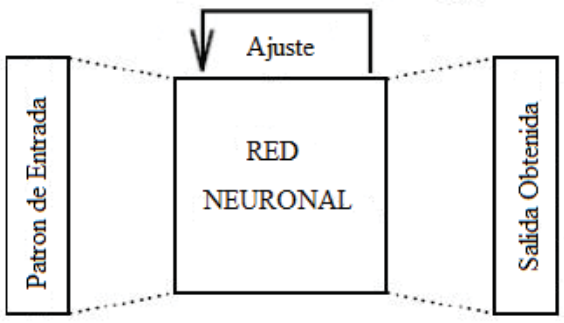

ILUSTRACIÓN 8: APRENDIZAJE NO SUPERVISADO

- Aprendizaje por refuerzo: variante del aprendizaje supervisado en el que no se dispone de información concreta del error cometido por la red para cada ejemplo de aprendizaje, sino que simplemente se determina si la salida producida para dicho patrón es o no adecuada.

3.3 TOPOLOGÍAS DE REDES NEURONALES

La topología o arquitectura de una red neuronal consiste en la organización y disposición de las neuronas que la conforman. Estas estructuras o modelos conexionistas se encuentran fuertemente ligados al algoritmo de aprendizaje utilizado en su elaboración y se pueden clasificar de diferentes formas dependiendo del criterio de uso [Soria y Blanco, 2001]:

3.3.1 SEGÚN SU NÚMERO DE CAPAS

- **Redes Neuronales Monocapa:** Corresponden a las redes neuronales más sencillas dado que solo se tiene una capa de entrada que proyecta la señal a una capa de salida en la cual se realizan diferentes cálculos. Dado que la capa de entrada no realiza ningún cálculo, de las dos capas nombradas previamente, solo se cuenta una capa (la de salida). De ahí el nombre monocapa. Estas redes se utilizan generalmente en relacionadas con lo que se conoce como auto-asociación.

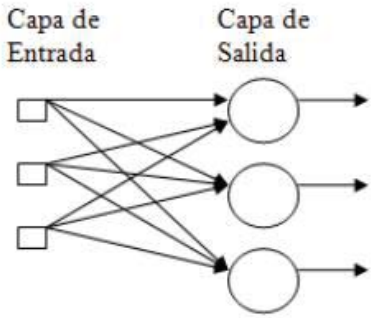

ILUSTRACIÓN 9: RED NEURONAL MULTICAPA

- **Redes Neuronales Multicapa**: Son aquellas que disponen de un conjunto de neuronas agrupadas en varios niveles o capas. Es una generalización de la anterior existiendo un conjunto de capas intermedias, entre la capa de entrada y salida, conocidas como capa oculta.

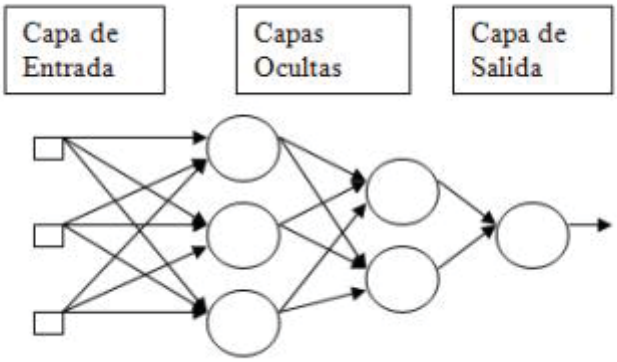

ILUSTRACIÓN 10: RED NEURONAL MULTICAPA

3.3.2 SEGÚN EL TIPO DE CONEXIONES

- **Redes Neuronales no recurrentes**: También conocidas como redes feed-forward, son aquellas donde los nodos se conectan solo a capas posteriores, sin generar ciclos de información.
- **Redes Neuronales Recurrentes**: También conocida como redes de retroalimentación (feedback), son aquellas donde las neuronas de una capa se conectan con neuronas de distintas capas (anteriores y posteriores), neuronas de la misma capa, o incluso con la misma neurona.

3.3.3 SEGÚN EL GRADO DE CONEXIÓN

- **Redes Neuronales totalmente conectadas**: En estas redes todas las neuronas de una capa se encuentran conectadas con las neuronas de la capa siguiente y con las de la capa anterior, en caso de ser una red neurona recurrente.
- **Redes Neuronales parcialmente conectadas**: En este caso no se da la conexión total entre neuronas de diferentes capas.

3.4 HISTORIA DE LAS REDES NEURONALES ARTIFICIALES

McCulloch y Pitts [McCulloch and Pitts, 1943] realizaron el primer modelo matemático de una RNA. McCulloch fue un siquiatra y neuroanatomista de formación, paso alrededor de 20 años pensando en la representación de un evento en el sistema nervioso. Pitts fue un prodigio de la matemática, que se unió a McCulloch en 1942. Ellos describieron un cálculo lógico de las redes neuronales que unió los estudios de la neurofisiología y la lógica matemática, concibiendo un modelo abstracto y simple de una neurona artificial, el elemento básico de procesamiento en una RNA.

Donald Hebb desarrollo posteriormente un procedimiento matemático de aprendizaje. Los estudios de Hebb sobre las neuronas y las condiciones clásicas de aprendizaje se describen en su libro *Organization of Behavior* [Hebb, 1949], donde el concepto de *modificación sináptica* fue introducido por primera vez. Hebb propuso que la conectividad del cerebro esta siempre cambiando como un organismo que aprende diferentes tareas funcionales, y los *conjuntos neuronales* son creados por estos cambios. Luego presento su ya famoso *postulado de aprendizaje*, que establece que la eficacia de una sinapsis variable entre 2 neuronas es incrementada por la activación repetitiva de una neurona por la otra a través de esa sinapsis.

Frank Rosenblatt generalizo el modelo de células de McCulloch-Pitts añadiéndole aprendizaje [Rosenblatt, 1957], [Rosenblatt, 1958]; llamo a este modelo el Perceptron (que se explicara más adelante). Primero desarrollo un modelo de dos niveles, que ajustaba los pesos de las conexiones entre los niveles de entrada y salida, en proporción al error entre la salida deseada y la salida obtenida. Rosenblatt intento extender su procedimiento de aprendizaje a un Perceptron de tres niveles, pero no encontró ningún método matemático sólido para entrenar la capa de conexiones ocultas.

Dos años después de la aparición del Perceptron, Bernard Widrow [Widrow, 1959], [Widrow, 1960] diseño una RNA muy similar al Perceptron, llamada *Adaptative Linear Elemento* o Adaline. El Adaline de dos niveles, muy parecido al Perceptron, ajusta los pesos entre los niveles de entrada y salida en función del error entre el valor esperado de salida y el obtenido. La diferencia entre los modelos es muy pequeña, pero las aplicaciones a las que van dirigidas son muy distintas (Perceptron con salidas de 1 y -1, Adaline con salidas reales).

Ya en la década del 60 encontramos a Stephen Grossberg como uno de los más influyentes y formal de todos los investigadores de RNA. Grossberg realizo importantes estudios sobre procesos y fenómenos psicológicos (mente) y biológicos (cerebro) de procesamiento humano de la información e intento juntar los dos (mente y cerebro) en una teoría unificada [Grossberg, 1964]. Los trabajos de Grossberg incluyen estrictos análisis matemáticos, que permitieron la realización de nuevos paradigmas de Redes de Neuronas. Estos permitían tener un acceso directo a la información mientras se operaba en tiempo real.

A fines de los sesenta, exactamente en 1969, se produjo la "muerte abrupta" de las RNA cuando Minsky y Papera probaron matemáticamente que el Perceptron no era capaz de resolver problemas tan fáciles como el aprendizaje de una función no lineal. Esto demuestra una gran debilidad, dado que las funciones no-lineales son ampliamente utilizadas en computación y en problemas del mundo real.

Terence Sejnowski es uno de los pocos investigadores que trabajaron con modelos matemáticos y biológicos. Una de sus más importantes contribuciones en este campo es el descubrimiento, junto con Geoff Hinton, del algoritmo de la máquina de Boltzmann [Hinton et al., 1984], y su extensión de mayor orden, la primera RNA que reconocía un algoritmo de aprendizaje para una red de tres niveles [Sejnowski and Hinton, 1986]. Aplicaron la máquina de Boltzmann a distintas áreas de visión [Kienker et al., 1986], [Sejnowski, 1986]. Más recientemente son conocidas sus contribuciones a la aplicación del algoritmo de Retropopagación, sobre todo para el reconocimiento de voz.

3.5 Modelos de Redes Neuronales Artificiales

Durante la historia de las RNA, los modelos han ido evolucionando tratando de imitar de mejor forma el funcionamiento de las redes neuronales de los animales. Ahora se explicaran los modelos y sus evoluciones a lo largo del tiempo, llegando al que se utilizara para el desarrollo del problema planteado en el inicio.

3.5.1 Modelo McCulloch-Pitts

El primer ejemplo que puede considerarse como una RNA, al menos estructuralmente, son las células de McCulloch-Pitts. Este modelo fue propuesto en 1943 en el artículo "*A Logical Calculus of the Ideas Immanent in Nervous Activity"* [McCulloch and Pitts, 1943]. En él modelizaban una estructura y un funcionamiento simplificado de las neuronas del cerebro, considerándolas como dispositivos con solo 2 estados posibles: apagado (0) y encendido (1).

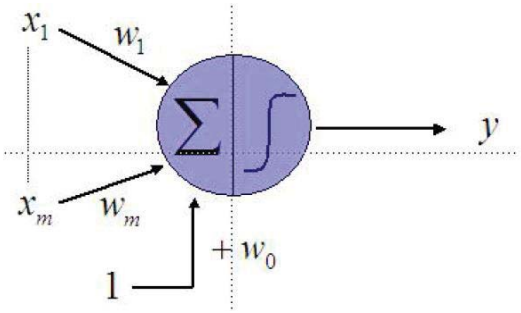

ILUSTRACIÓN 11: CÉLULA MCCULLOCH-PITTS

Este modelo está compuesto por el vector X, que es la entrada procedente de otras células o del entorno; el vector W^T, que es el vector de pesos, equivalente a las conexiones sinápticas en una red neuronal; W_0 es el umbral de acción (activación o inactivación de la neurona) e $\boldsymbol{y}$ es la salida de la unidad. La célula opera en tiempos discretos. La forma de procesar la entrada es la siguiente: la célula se activara y, por lo tanto, producirá un valor 1 si la suma de las entradas multiplicadas pos los pesos supera al umbral W_0.

$$y(t+1) = \begin{cases} 1 & si\ \sum_i W_i X_i(t) > W_0 \\ 0 & en\ caso\ contrario \end{cases} \qquad (3.2)$$

A partir del modelo de neurona de McCulloch-Pitts se define el primer modelo de red neuronal:

"Una red neuronal es una colección de neuronas de McCulloch y Pitts, todas con las mismas escalas de tiempos, donde sus salidas están conectadas a las entradas de otras neuronas" [Isasi y Galvan, 2004].

Una red neuronal de células de McCulloch-Pitts tiene la capacidad de computación universal, es decir, cualquier estructura que pueda ser programada en un computador, puede ser modelada mediante una red de células de McCulloch-Pitts. Esto se puede intuir mediante la modelización de funciones lógicas. Los computadores están constituidos de elementos de cálculo simples. Si cada uno de estos elementos puede ser modelado mediante una estructura de células de McCulloch-Pitts, los cómputos realizados por éste podrían ser sustituidos por su correspondiente red de células.

3.5.2 PERCEPTRON

Este modelo se concibió como un sistema capaz de realizar tareas de clasificación de forma automática. La idea era disponer de un sistema que, a partir de un conjunto de ejemplos de clases diferentes, fuera capaz de determinar las ecuaciones de las superficies que hacían de fronteras de dichas clases. La información sobre la que se basaba el sistema estaba compuesta por los ejemplos existentes de las diferentes clases. A esto se le conoce como patrones o ejemplos de entrenamiento. Son dichos patrones de entrenamiento los que aportaban la información necesaria para que el sistema construyera las superficies discriminantes, y además actuara como un discriminador para ejemplos nuevos desconocidos. El sistema al final del proceso, era capaz de determinar, para cualquier ejemplo nuevo, a que clase pertenecía.

La arquitectura del Perceptron es muy simple. Se trata de una estructura monocapa, en la que hay un conjunto de células de entrada y una o varias células de salida. Cada una de las células de entrada está conectada a todas las células de salida, y son estas conexiones las que determinan las superficies de discriminación del sistema.

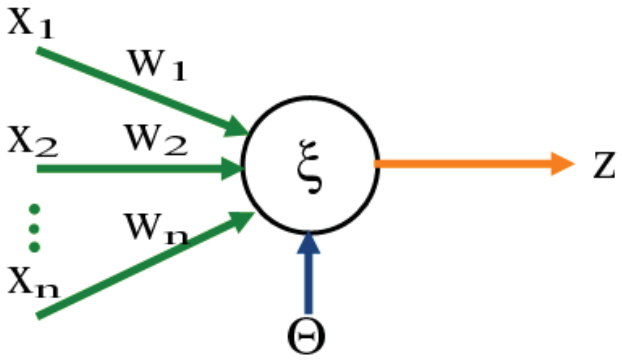

ILUSTRACIÓN 12: PERCEPTRON CON N ENTRADAS Y UNA SALIDA

Aquí tenemos las entradas que van desde X_1 hasta X_n, los pesos que van desde W_1 hasta W_n y la salida Z. Además existe un umbral denotado por θ. El umbral se utiliza como factor de comparación para producir la salida y habrá tantos como células de salida existan en la red, uno por cada una. En este esquema la salida se obtiene de la siguiente forma: primero se calcula la activación de la célula de salida mediante la suma ponderada de los pesos de todas las entradas.

$$y' = \sum_{i=1}^{n} W_i X_i \qquad (3.3)$$

La salida definitiva se produce al aplicarle una función de salida al nivel de activación de la célula. En un Perceptron la función de salida es una función escalón que depende del umbral.

$$y = F(y', \theta)$$

$$F(s, \theta) = \begin{cases} 1, & si\ s > \theta \\ -1, & en\ caso\ contrario \end{cases}$$

La salida de ***F*** es binaria y de gran utilidad en este modelo ya que al tratarse de un discriminante de clases, una salida binaria puede ser fácilmente traducible a una clasificación en 2 categorías de la siguiente forma:

- Si la red produce salida 1, la entrada pertenece a la categoría A
- Si la red produce salida -1, la entrada pertenece a la categoría B

En el caso de 2 dimensiones, tendríamos lo siguiente:

$$W_1 X_1 + W_2 X_2 + \theta = y \qquad (3.4)$$

Igualando la función a cero

$$W_1 X_1 + W_2 X_2 + \theta = 0 \qquad (3.5)$$

Que es la ecuación de una recta de pendiente (-w1/w2) y que en el origen de sus coordenadas pasa por (-o/w1). En este caso, con dos células en la entrada, los patrones de entrenamiento pueden representarse como puntos en un espacio bidimensional. Si además dichos puntos pertenecen a una de dos categorías, A o B, la separación de dichas categorías podrá hacerse mediante la recta anterior, es decir, la red define una recta, que en el caso de ser la solución al problema discriminará entre las clases existentes en los datos de entrenamiento. Gráficamente se podría representar de la siguiente manera

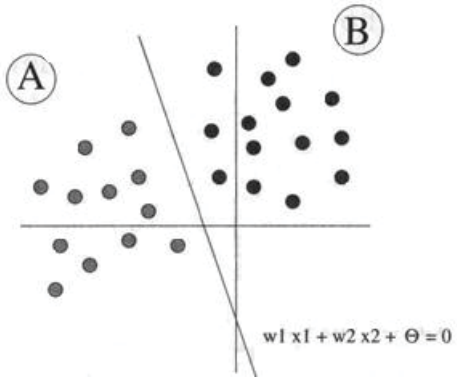

ILUSTRACIÓN 13: CLASIFICACIÓN VÍA PERCEPTRON

Si las clases están separadas puede demostrarse que dicha recta discriminante existe; el problema es como determinar la ecuación de tal recta, a partir de los datos de entrenamiento. Esto es lo que hace el Perceptron en su proceso de aprendizaje.

Formalmente seria de la siguiente manera: dados conjuntos de puntos en R^n: $A = (\overrightarrow{a_1}, \dots, \overrightarrow{a_{na}})$ y $B = (\overrightarrow{b_1}, \dots, \overrightarrow{b_{nb}})$, obtener el hiperplano: $w_1 a_1 + \cdots + w_n a_n + \theta > 0$ y $w_1 b_1 + \cdots + w_n b_n + \theta < 0$. Este proceso constituye el aprendizaje del Perceptron y se realiza mediante un proceso iterativo en el que paulatinamente se van modificando los valores de los pesos de las conexiones, hasta encontrar los valores que determinan las ecuaciones discriminantes. En función de la red neuronal esto sería equivalente a encontrar los valores de las conexiones que hagan

$$\forall \vec{a} \in A: y(\vec{a}) = 1 \text{ Y } \forall \vec{b} \in B: y(\vec{b}) = -1 \qquad (3.6)$$

El proceso de aprendizaje consiste en lo siguiente: se introduce un patrón de los del conjunto de aprendizaje, perteneciente por ejemplo, a la clase A. se obtiene la salida que genera la red para dicho patrón. Si la salida producida es 1, la respuesta de la red para dicho patrón es correcta, y no se realizara ninguna acción; se procederá simplemente a introducir un nuevo patrón. Si por el contrario la salida producida es -1, entonces la respuesta de la red es incorrecta y es en este caso cuando se produce el aprendizaje. El aprendizaje consiste en modificar los valores de las conexiones. En este caso, dado que la salida producida es inferior a la que debería haber obtenido, los pesos son incrementados para que en la próxima presentación del mismo patrón pueda superar el umbral y producir la salida deseada, 1. Si el patrón que se introduce es el de la clase B, y también se produce un error de clasificación, el proceso se invierte; los pesos se decrementan por la misma razón.

Si se llama X a cada uno de los patrones de entrenamiento y $d(X)$ a su clase asociada, tomando valores en (1,-1), el proceso de aprendizaje puede describirse de la siguiente manera:

1. Empezar con valores aleatorios para los pesos y el umbral
2. Seleccionar un vector de entrada X del conjunto de ejemplos de entrenamiento
3. Si $y \neq d(X)$, la red da una respuesta incorrecta. Modificar w_i de acuerdo con:

$$\Delta w_i = d(X)x_i$$

4. Si no se ha cumplido el criterio de finalización, volver a 2.

3.5.3 ADALINE

El Perceptron es un sistema de aprendizaje basado en ejemplos capaz de realizar tareas de clasificación. Sin embargo, existen un gran número de problemas abordables desde la perspectiva del aprendizaje basado en ejemplos que no se reducen a tareas de clasificación. La característica de clasificador del Perceptron viene dada por la naturaleza de sus salidas. En la capa de salida las células codifican una función de salida en escalón (1 o -1), que transforma la suma ponderada de las entradas que es un valor real, en una salida binaria. Al ser binarias, solo pueden codificar un numero de discreto de estados. Si las salidas fueran números reales, podrían codificar cualquier tipo de salida y se convertirían en sistemas de resolución de problemas generales. En este caso, podrían resolver problemas a partir de ejemplos en los que es necesario aproximar una función cualquiera $F(x)$, definida por un conjunto de datos de entrada.

Los datos de entrenamiento en este caso son un conjunto de valores enteros, compuestos por un vector de entrada y su salida asociada:

$$P = \{(\vec{x}_1, y_1), \dots, (\vec{x}_m, y_m)\}$$

En este caso la función a aproximar seria:

$$F(\overrightarrow{x_i}) = y_i, \forall p \in P \qquad (3.7)$$

En otros términos, habrá que buscar una función $F(x)$tal que aplicada a cada una de las entradas x_i del conjunto de aprendizaje P produzca una salida y_i correspondiente en dicho conjunto de aprendizaje.

Sin embargo, la naturaleza de la regla de aprendizaje de Perceptron no permite producir salidas reales. Esta regla es fundamentalmente una regla de aprendizaje por refuerzo en la que las salidas correctas se potencian y las salidas incorrectas no se tienen en cuenta. No existe ninguna graduación en la regla que indique en que medida resulta errónea la salida producida, y, refuerce proporcionalmente a dicha medida de error.

En el año 1960, Widrow y Hoff [Widrow, 1960] propusieron un sistema de aprendizaje que sí tuviera en cuenta el error producido, y diseñaron lo que denominaron ADAptive Linear NEuron, o en su acrónimo, Adaline. Ésta es una estructura prácticamente idéntica al Perceptron, pero es un mecanismo físico, capaz de realizar aprendizaje. Es un elemento combinador adaptativo, que recibe un conjunto de entradas y las combina para

producir una salida. Esta salida puede transformarse en binaria mediante un conmutador bipolar que produce un 1 si la salida es positiva y un -1 si es negativa.

$$\tilde{y} = \sum_{i=1}^{n} W_i X_i + \theta \tag{3.8}$$

El aprendizaje en este caso incluye la diferencia entre el valor real producido en la capada de salida para un patrón de entrada x^p y el que debería haber producido dicho patrón, es decir, su salida esperada d^p, que está en el en conjunto de aprendizaje ($|d^p - y^p|$). A esta regla de aprendizaje se le conoce con el nombre de *regla Delta*.

La diferencia con respecto a la regla de aprendizaje del Perceptron es la manera de utilizar la salida, una diferencia fundamental entre ambos sistemas. El Perceptron utiliza la salida de la función umbral para el aprendizaje; sin embargo, la regla Delta utiliza directamente la salida de la red, sin pasarla por ninguna función umbral. El objetivo es obtener un valor determinado $y = d^p$ cuando el conjunto de valores $X_i^p, i = 1, \dots, n$ se introduce la entrada. Por lo tanto, el problema será obtener los valores de $w_i, i = 1, \dots, n$ que permitan realizar lo anterior para un número arbitrario de patrones de entrada.

No es posible obtener una salida exacta, pero si minimizar la desviación cometida por la red, esto es, minimizar el error cometido por la red para la totalidad del conjunto de patrones de ejemplo. En otros términos, hay que evaluar globalmente el error cometido por la red para todos los patrones, o sea, elegir una medida del error global. Habitualmente, la medida de error global utilizada es el error cuadrático medio, pero pueden ser utilizadas otras medidas de error.

$$E = \sum_{p=1}^{m} E^p = \frac{1}{2}\sum_{p=1}^{m}(d^p - y^p)^2 \tag{3.9}$$

Al ser ésta una medida de error global, la regla intentará minimizar este valor para todos los elementos del conjunto de patrones de aprendizaje. La manera de minimizar este error es recurrir a un proceso iterativo en que se van presentando los patrones uno a uno, y modificando los pesos de las conexiones.

$$\Delta_p w_j = \gamma(d^p - y^p)x_j \tag{3.10}$$

Si se compara con la expresión de aprendizaje del Perceptron, se aprecia que la diferencia es precisamente la introducción de la diferencia entre la salida deseada y la obtenida en la regla de aprendizaje. Si se incluye a la salida del Adaline el acoplador bipolar comentado con anterioridad para "binarizar" la salida, la regla Delta quedaría así:

$$\Delta_p w_j = \begin{cases} \gamma x_j & si\ d^p > y^p \\ -\gamma x_j & si\ d^p < y^p \\ 0 & en\ caso\ contrario \end{cases} \tag{3.11}$$

Que para el valor de $\gamma = 1$se convierte en la regla del Perceptron. El procedimiento de aprendizaje por la regla Delta será:

1. Inicializar los pesos de forma aleatoria.

2. Introducir un patrón de entrada.
3. Calcular la salida de la red, compararla con la deseada y obtener la diferencia: $(d^p - y^p)$.
4. Para todos los pesos, multiplicar dicha diferencia por la entrada correspondiente, y ponderarla por una tasa de aprendizaje γ.
5. Modificar el peso restando del valor antiguo la cantidad obtenida en 4.
6. Si no se ha cumplido el criterio de convergencia, regresar a 2; si se han acabado todos los patrones, empezar de nuevo a introducir patrones.

En resumen, se puede apreciar que las diferencias con su antecesor, el Perceptron son las siguientes:

- En el Perceptron la salida es binaria, en Adaline es real
- En el Perceptron la diferencia entre entrada y salida es 0 si ambas pertenecen a la misma categoría y ± 1 si pertenece a categorías diferentes. En el Adaline se calcula la diferencia real entre entrada y salida.
- En el Adaline existe una medida de cuánto se ha equivocado la red; en el Perceptron sólo se determina si se ha equivocado o no.
- En el Adaline hay una razón de aprendizaje (γ) para regular cuánto va a afectar cada equivocación a la modificación de los pesos. Es siempre un valor entre o y 1 para ponderar el aprendizaje.

3.5.4 PERCEPTRON MULTICAPA

El Perceptron multicapa o red multicapa con conexiones hacia adelante es una generalización del Perceptron presentado con anterioridad y surgió como consecuencia de las limitaciones de dicha arquitectura en lo referente al problema de la separabilidad no lineal. Minsky y Papert [Minsky and Papert, 1969] mostraron que la combinación de varios Perceptrones simples -inclusión de neuronas ocultas- podía resultar una solución adecuada para tratar ciertos problemas no lineales, sin embargo, los autores no presentaron una solución al problema de cómo adaptar los pesos de la capa de entrada a la capa oculta, pues la regla de aprendizaje del Perceptron simple no puede aplicarse en este escenario. No obstante, estos estudios sirvieron para trabajos posteriores realizados por Rumelhart, Hinton y Willians en 1986 [Rumelhart et al., 1986b]. Estos autores presentaron una manera de retropropagar los errores medidos en la salida de la red hacia las neuronas ocultas, dando lugar a la llamada regla delta generalizada, que no es más que una generalización de la regla delta.

Diferentes autores ([Cybenko, 1989], [Hornik et al., 1989]) han demostrado de manera independiente que el Perceptron multicapa es un aproximador universal, en el sentido de que cualquier función continua sobre un compacto de R^npuede aproximarse con un Perceptron multicapa, con al menos una capa oculta, y es gracias a esto, además de su fácil uso y aplicabilidad, es actualmente una de las arquitecturas más utilizadas en la resolución de problemas. Han sido utilizadas en problemas: de reconocimiento de habla [Cohen et al., 1993], reconocimiento de caracteres ópticos [Sackinger et al., 1992], reconocimiento de caracteres escritos [Guyon, 1991], control de procesos [Werbos, 1989], modelización de sistemas dinámicos [Narendra and Parthasarathy, 1990], conducción de vehículos [Pomerleau, 1992], etcétera. Cabe señalar que aunque sea una de las más

utilizadas no quiere decir que sea una de las más potentes y con mejores resultados en las diferentes áreas de la aplicación.

La arquitectura del Perceptron multicapa se caracteriza porque tiene sus neuronas agrupadas en capas de diferentes niveles. Cada una de estas capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas: la capa de entrada, las capas ocultas y la capa de salida.

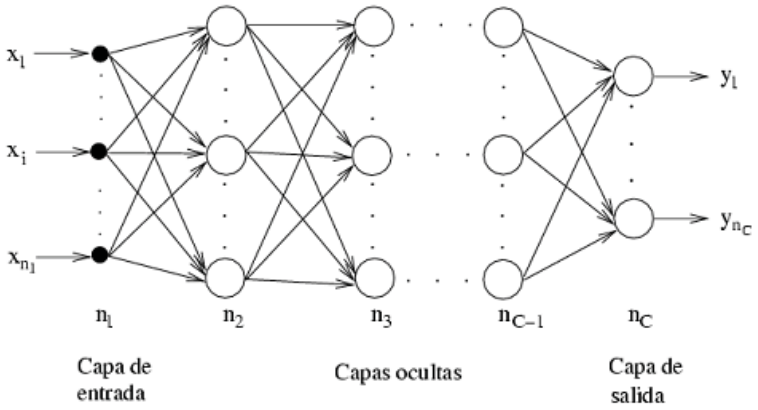

ILUSTRACIÓN 14: ARQUITECTURA DEL PERCEPTRON MULTICAPA

Las neuronas de la capa de entrada no actúan como neuronas propiamente tal, sino que se encargan únicamente 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. Se observa en la figura que todas las conexiones están dirigidas hacia adelante, es decir, las neuronas de una capa se conectan con la de la capa siguiente, de ahí que reciba el nombre de redes alimentadas hacia adelante o "feedforward". Las conexiones entre las neuronas tienen 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 Perceptron 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; si este es el caso, se dice entonces que hay una conectividad total o que la red está totalmente conectada.

El Perceptron multicapa define una relación entre variables de entrada y las variables de salida de la red. Esta relación se obtiene propagando hacia adelante los valores de las variables de entrada. Para ello, cada neurona de la red procesa la información recibida por sus entradas y produce una respuesta o activación que se propagan, a través de las conexiones hacia las neuronas de la siguiente capa.

Sea un Perceptron multicapa con C capas, C-2 capas ocultas yn_c neuronas en la capa c, para $c = 1,2, \dots, C$. Sea $W^c = (w_{i,j}^c)$ la matriz de pesos asociada a las conexiones de la capa c a la capa $c+1$ para $c = 1,2, \dots, C-1$, donde $w_{i,j}^c$ representa el peso de la conexión de la neurona i de la capa c a la neurona j de la capa $c+1$; y sea $U^c = (u_i^c)$ el vector de umbrales de las neuronas de la capa c para $c = 2, \dots, C$. Se denota a_i^c a la activación de la neurona i de la capa c; estas activaciones de calculan de la siguiente manera:

• Activación de las neuronas de la capa de entrada (a_i^1). Las neuronas de la capa de entrada se encargan de transmitir hacia la red las señales recibidas del exterior. Por tanto:

$$a_i^1 = x_i \ para \ i = 1,2,\dots,n_i$$

Donde $X = (x_1, x_2, \dots, x_{n1})$ representa el vector o patrón de entrada a la red

• Activación de las neuronas de la capa oculta c (a_i^c). Las neuronas ocultas de la red procesan la información recibida aplicando al función de activación f a la suma de los productos de las activaciones que recibe por sus correspondientes pesos, es decir:

$$a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{j\ i}^{c-1} a_j^{c-1} + u_i^c\right) \qquad (3.12)$$
$$para \ i = 1,2,\dots n_c \ y \ c = 2,3,\dots,C-1$$

• Activación de las neuronas de la capa de salida (a_i^C). Al igual que en el caso anterior, la activación de estas neuronas viene dada por la función de activación f aplicada a la suma de los productos de las entradas que recibe por sus correspondientes pesos.

$$y_i = a_i^C = f\left(\sum_{j=1}^{n_{C-1}} w_{j\ i}^{C-1} a_j^{C-1} + u_i^C\right) para \ i = 1,2,\dots n_C \qquad (3.13)$$

Donde $Y = (y_1, y_2, \dots, y_{nC})$ es el vector de salida de la red.

La función f es la llamada *función de activación.* Para el Perceptron multicapa, las funciones de activación más utilizadas son la función sigmoidal y la función tangente hiperbólica. Dichas funciones poseen como imagen un rango continuo de valores dentro de los intervalos [0,1] y [-1,1] respectivamente y vienen dadas por las siguientes expresiones.

$$f = \frac{1}{1+e^{-x}} \quad y \quad f = \frac{1-e^{-x}}{1+e^{-x}} \qquad (3.14)$$

3.5.4.1 ALGORITMO DE RETROPOPAGACIÓN

La regla o algoritmo de aprendizaje es el mecanismo mediante el cual se van adaptando y modificando todos los parámetros de la red. En el caso del Perceptron multicapa se trata de un algoritmo de aprendizaje supervisado, es decir, la modificación de los parámetros se realiza para que la salida de la red sea lo más próxima posible a la salida deseada. Puesto que el objetivo es que la salida de la red sea lo más próxima posible a la salida deseada, el aprendizaje de la red se formula como un problema de minimización del siguiente modo:

$$Min_w E$$

Siendo W el conjunto de parámetros de la red -pesos y umbrales- y E una función de error que evalúa la diferencia entre las salidas de la red y las salidas deseadas. En la mayor parte de los casos, la función error se define como:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n) \qquad (3.15)$$

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

$$e(n) = \frac{1}{2}\sum_{i=1}^{n_C}(s_i(n) - y_i(n))^2 \quad (3.16)$$

Siendo $Y(n) = (y_1(n), \dots, y_{nc}(n))$ y $S(n) = (s_1(n), \dots, s_{nc}(n))$ los vectores de salida de la red y salidas deseadas para el patrón n respectivamente.

De este modo, si W es un minimo de la funcion error E, en dicho punto el error es próximo a cero, lo cual implica que la salida de la red es próxima a la salida deseada, alcanzando así la meta de la regla de aprendizaje, por tanto, el aprendizaje del Perceptron multicapa es equivalente a encontrar un mínimo de la función error.

Debido a que las neuronas de la red están agrupadas en capas de distintos niveles, es posible aplicar el método del gradiente de forma eficiente, resultando el algoritmo de Retropopagación [Rumelhart et al., 1986b] o *regla delta generalizada*. El termino Retropopagación se utiliza debido a la forma de implementar el método del gradiente en el Perceptron multicapa, pues el error cometido en la salida de la red es propagado hacia atrás, transformándolo en un error para cada una de las neuronas ocultas de la red.

3.5.5 Redes Neuronales Recurrentes

Las redes neuronales recurrentes son aquellas que se caracterizan por la generación de ciclos en red mediante el uso de conexiones recurrentes. Pueden existir conexiones de una neurona a otra, de una neurona a ella misma o conexiones entre neuronas de la misma capa o conexiones de una neurona a una capa anterior.

La activación ahora depende del tiempo debido a que no solo interesan las dependencias de activación con la capa anterior, sino que también depende del estado de cualquier otra neurona que se encuentre conectada a ella.

$$a_i(t+1) = f_i\left(\sum_j w_{ji} a_j(t)\right) \quad (3.17)$$

El índice *j* varía en el conjunto de todas las neuronas que se encuentran conectadas con la neurona *i*. Para que las activaciones de las neuronas recurrentes tengan un comportamiento temporal, se agrega $a_i(t+1)$.

Las redes recurrentes más conocidas y utilizadas son las de Jordan, y las de Elman. En los capítulos siguientes se mostrará la arquitectura que tiene cada una de ellas.

3.5.5.1 Red Neuronal de Jordan

La red de Jordan [Jordan, 1986] se caracteriza por que las neuronas de contexto reciben una copia de las neuronas de salida de la red y de ellas mismas. Además las conexiones recurrentes de la capa de salida a la neurona de contexto llevan un parámetro μ asociado en el dominio de [0,1].

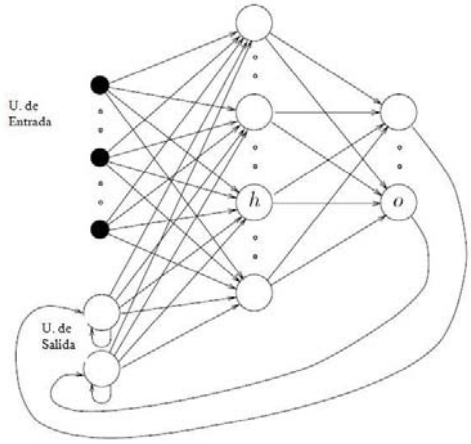

ILUSTRACIÓN 15: RED NEURONAL DE JORDAN

3.5.5.2 RED NEURONAL DE ELMAN

En la red de Elman [Elman, 1990] las neuronas de contexto reciben copias desde neuronas ubicadas en la capa oculta de la red, como aparece en la Figura 10 y no traen asociado ningún tipo de parámetro. Por lo que habrá tantas neuronas de contexto como neuronas ocultas tenga la red.

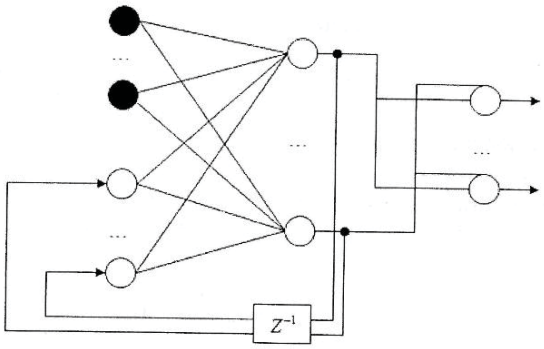

ILUSTRACIÓN 16: RED NEURONAL DE ELMAN

La red neuronal que se utilizó en este proyecto es la Wavelet Radial, para lo cual se explicaran algunas funciones wavelet además de la utilizada.

3.6 FUNCIÓN WAVELET

El desarrollo de las Wavelets tiene sus inicios a partir del trabajo de Alfred Haar a principio del siglo XX y también a Goupillaud, Grosman y Morlet, que contribuyeron de gran modo al avance de la teoría Wavelet con su formulación de lo que hoy conocemos como transformada de Wavelet continua. Entre los autores más influyentes tenemos a Jan Olov-Strömberg con su trabajo sobre las Wavelets discretas, Ingrid Daubechies con su propuesta de Wavelets Ortogonales de soporte compacto, Stephane Mallat e Yves Meyer con su marco multiresolución, Delrat con el trabajo sobre la interpretación de la transformada Wavelet en tiempo y frecuencia y Newland, con su transformada Wavelet armónica.

3.6.1 TRANSFORMADA WAVELET

Las wavelets son familia de funciones que se encuentran en el espacio y se emplean como funciones de análisis, examinan a la señal de interés para obtener sus características de espacio, tamaño y dirección; la familia está definida por:

$$h_{a,b} = \frac{h\left(\frac{x-b}{a}\right)}{\sqrt{|a|}}; a, b \in \mathbb{R}, a \neq 0 \tag{3.18}$$

Y son generadas a partir de funciones madre $h(x)$. A esa función madre se le agregan un par de variables que son la escala (a)que permite hacer dilataciones y contracciones de la señal y la variable de transacción (b), que permite mover a la señal en el tiempo. Estas variables son números reales y obviamente para $a = 0$ la función se indefine.

3.6.2 WAVELET HAAR

Es la función wavelet más simple y las más antigua. Es una función sencilla que se utiliza para el análisis de señales usando transformadas discretas y continúas. La función se describe de la siguiente manera:

$$h(x) = \begin{cases} 1; & 0 \leq x < \frac{1}{2,} \\ -1; & \frac{1}{2} \leq x < 1, \\ 0; & \text{otro valor.} \end{cases} \tag{3.19}$$

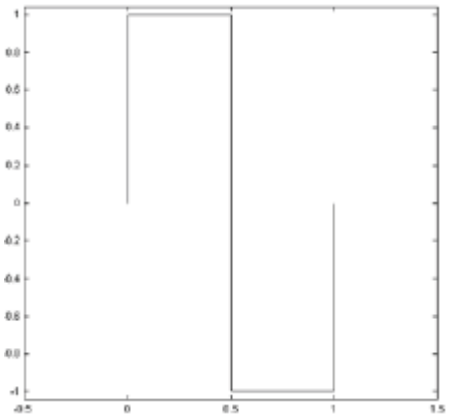

ILUSTRACIÓN 17: FUNCIÓN WAVELET HAAR

3.6.3 FUNCIÓN WAVELET MEXICAN HAT

El nombre se da por la forma que genera la gráfica, que está definida por la siguiente función:

$$mexh(x) = \frac{2(1-x^2)e^{-\frac{x^2}{2}}}{\pi^{\frac{1}{4}}\sqrt{3}} \tag{3.20}$$

La mexican hat es la segunda derivada de la función de densidad de probabilidad gaussiana. Como es una función simétrica, permite examinar las señales de un modo simétrico y lineal en la fase.

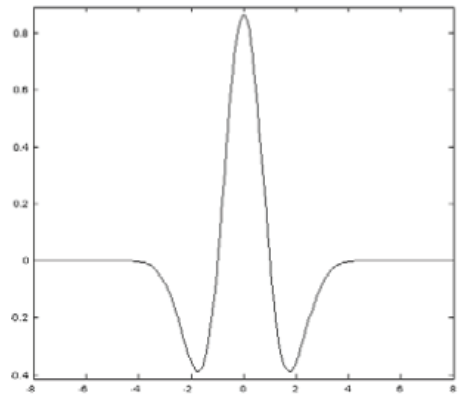

ILUSTRACIÓN 18: FUNCIÓN WAVELET MEXICAN HAT

3.6.4 FUNCIÓN WAVELET MORLET

Es una función simétrica, que sirve solo para realizar la transformada continua de wavelets. La fórmula que la define es la siguiente:

$$morl(x) = e^{-\frac{x^2}{2}} \cdot \cos(5x) \tag{3.21}$$

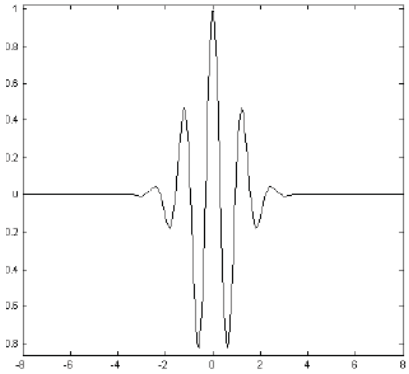

ILUSTRACIÓN 19: FUNCIÓN WAVELET MORLET

3.6.5 FUNCIÓN DERIVADA DE LA GAUSSIANA

Finalmente se presenta la función a utilizar en nuestro modelo, la cual es la derivada de la función Gaussiana. La fórmula es la siguiente:

$$f(x) = -x * e^{(-\frac{1}{2} * x^2)} \tag{3.22}$$

Y la gráfica es la siguiente:

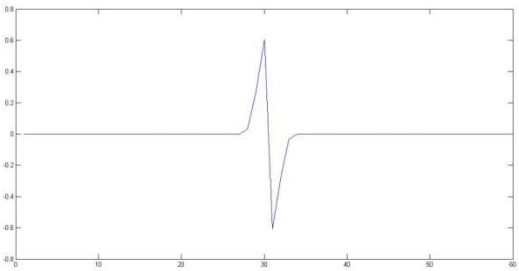

ILUSTRACIÓN 20: FUNCIÓN GAUSSIANA

3.7 RED NEURONAL WAVELET RADIAL

El modelo que se utilizo fue una Red Neuronal Wavelet Radial. Las RNWR combinan la teoría de procesamiento wavelet con las redes neuronales. En esta arquitectura, la capa oculta está formada por *wavelons* cuyos parámetros de entrada están definidos por peso, parámetro de traslación y dilatación. La salida de la red es una combinación lineal de las funciones wavelet de cada *wavelon.* La red en cuestión es la siguiente:

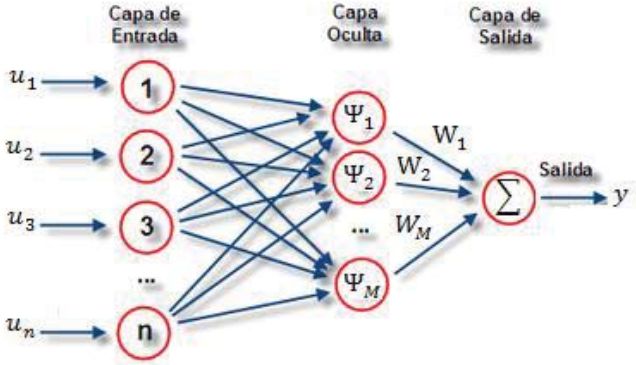

ILUSTRACIÓN 21: RED NEURONAL WAVELET RADIAL

Donde u representa a cada una de las entradas que tiene la red. Ψ representa a cada uno de los *wavelones* que posee la red, uno por cada nodo oculto. E y es la salida de la red.

En esta red aparecen los parámetros de traslación y dilatación. El **factor de dilatación** denotado por la letra λ es una cantidad entera positiva que tiene un efecto compresor mientras más pequeño sea o un efecto de estiramiento al adoptar un valor grande. Los valores pequeños en el factor de escalamiento sirven para detectar cambios rápidos en la función que se está analizando mientras que los valores grandes provocan una baja frecuencia que detecta cambios suaves en la señal. El **factor de traslación** denotado por la letra t tiene un efecto de retraso o avance en la señal. Cuando se efectúa el cálculo de los coeficientes, el factor de traslación se modifica para que cubra toda la señal con el valor de escalamiento específico.

La salida de la red, está dada por la siguiente función:

$$y = \sum_{i=1}^{M} w_i \Psi_i(u_1, \dots, u_n) \quad (3.24)$$

$$\Psi(u_1, \dots, u_N) = \prod_{i=1}^{N} \psi_{\lambda i,ti}(u_i) \quad (3.25)$$

$$\psi_{\lambda,t}(u) = \psi\left(\frac{u-t}{\lambda}\right) \quad (3.26)$$

Donde Ψ es el *wavelon*, ψ es la función *wavelet madre.* La función madre a utilizar será la función derivada de la Gaussiana

CAPÍTULO 4

OPTIMIZACIÓN POR ENJAMBRE DE PARTÍCULAS

El algoritmo de Optimización por Enjambre de Partículas - PSO, del inglés Particle Swarm Optimization -[Kennedy, 1995] es un algoritmo bioinspirado, que se basa en una analogía con el comportamiento social de ciertas especies. Se reconocen dos influencias básicas de inspiración en PSO:

1. El movimiento de bandadas de aves, en las que cada individuo se desplaza mediante reglas simples de ajuste de su velocidad en función de las observaciones que realiza sobre los individuos próximos en la bandada.

2. Un modelo social, en el que cada partícula representa una creencia, y las influencias entre individuos, la aproximación de unos individuos a otros por difusión de dichas creencias.

En líneas generales, el algoritmo busca encontrar el óptimo global de una función mediante el movimiento de un conjunto de partículas – enjambre - en un espacio definido por el número de parámetros de la función. Cada partícula es una posible solución, es decir, un conjunto de parámetros, que se evalúa calculando el valor de fitness que les correspondería. En dicho movimiento, las partículas utilizan información histórica - el resultado de su exploración pasada -, así como información sobre sus vecinos - otras partículas del enjambre -.

Por otro lado, para completar la especificación del algoritmo, hay que definir ciertos parámetros del algoritmo - criterio de parada, tipo de vecindario, valores de algunas constantes, forma de inicialización -. El enjambre comienza disperso por el espacio de búsqueda, pero con el tiempo las partículas convergen hacia una zona reducida del espacio en la que intensifican la búsqueda de la solución.

4.1 EL PARADIGMA DE LA INTELIGENCIA COLECTIVA

La Inteligencia Colectiva (SI, del inglés Swarm Intelligence) es un paradigma que agrupa técnicas de inteligencia artificial basadas en el estudio del comportamiento colectivo observado en la naturaleza. Este comportamiento es intrínseco a los orígenes de los individuos, los cuales tienen una fuerte tendencia a asociarse a otros y socializar [Kennedy, 2001]. De esta forma, reglas, conocimiento, experiencias y creencias pueden intercambiarse. La SI consta de una población de agentes simples interactuando localmente con los demás y con el ambiente que los rodea. En general no existe una estructura de control centralizada previendo cómo deberían actuar los agentes individuales. No obstante, las interacciones locales entre tales agentes a menudo convergen en un comportamiento global del cual todos se benefician. Ejemplos de esto son las colonias de hormigas, los cardúmenes, las bandadas y las manadas, en donde todos los agentes se mueven en conjunto. Se han efectuado diversos estudios para entender el comportamiento coordinado de los grupos, y éstos han sido utilizados en diferentes heurísticas de inteligencia colectiva. Una de ellas es la que se describe a continuación.

4.2 TERMINOLOGÍA DEL PSO

Esta heurística se basa en el modelo psico-social de entidades colectivas. PSO posee influencia y aprendizaje social [Kennedy, 2003], que se refleja en cada agente de la entidad. Los individuos - partículas - en la heurística emulan una característica simple: adaptan su comportamiento al de los individuos dentro de su propio vecindario o de la población completa - en inglés, swarm -.

Una partícula se mueve dentro del espacio de búsqueda siguiendo una trayectoria definida por su velocidad, y la memoria de dos buenos valores: el mejor encontrado por ella misma en su pasado y el mejor valor encontrado por alguna partícula de la población. Estos dos valores generan una "atracción" hacia los lugares más prometedores, y a esta atracción o fuerza se la denomina presión social.

4.3 ESTRUCTURA DE UNA PARTÍCULA

La estructura básica de una partícula consta de cinco componentes:

• Valor *objetivo* o aptitud *fitness*, representa la calidad de la solución representada por el vector $\vec{x}$, obtenido a través del cálculo de la función de evaluación correspondiente al problema específico.

• Un vector v que representa la velocidad de la partícula. Este vector, al igual que $\vec{x}$, se modifica en cada iteración del algoritmo reflejando así el cambio de dirección que sufre la partícula dentro del espacio de búsqueda.

• p*best*, mejor valor de fitness encontrado por la partícula hasta el momento.

• *gbest*, mejor valor de fitness encontrado por alguna partícula del enjambre - swarm - .Este valor está presente si se utiliza un modelo global, es decir, un modelo en el que los individuos son influenciados por el mejor de toda la población.

• *lbest*, mejor valor de fitness encontrado en el vecindario de una partícula. Este valor está presente si se utiliza un modelo local, es decir, un modelo en el que los individuos son influenciados por el mejor de un grupo pequeño de individuos - vecindario -. La determinación del vecindario depende de la forma en la que se efectúa el agrupamiento de individuos de la población.

4.4 TRAYECTORIA DE UNA PARTÍCULA

La trayectoria de la partícula dentro del espacio de búsqueda está definida con base en el vector de ubicación $\vec{x}$ y el de velocidades $\vec{v}$, los cuales son sumados para obtener la nueva dirección. Justamente estos cambios de trayectoria son los que determinan la característica principal del algoritmo PSO, ya que a través de los mismos las partículas son forzadas a buscar soluciones en las áreas más promisorias del espacio de búsqueda.

Cabe destacar que si los valores de $\vec{v}$ no se modificaran, la partícula sólo se movería con pasos uniformes en una única dirección. Más específicamente, en cada iteración del algoritmo, la dirección que tomaría la partícula es modificada considerando el valor de pbest - la atracción hacia la mejor posición alcanzada por la partícula - y el vector gbest - la atracción hacia la mejor posición alcanzada por alguna partícula del cúmulo o lbest si se consideran vecindarios -.

Además PSO introduce valores aleatorios de ajuste para las dos últimas atracciones. Estos valores aseguran que el tamaño del paso que realizan las partículas dentro del espacio sea variable, asegurando que las mismas no "caigan en una rutina", es decir, que no se muevan siempre con el mismo paso. La ilustración 12 ilustra la obtención de la nueva posición x' de la partícula x como consecuencia de la suma de la velocidad y las memorias de los mejores.

La influencia relativa de la memoria que lleva asociada cada partícula - *pbest* - es denominada influencia cognitiva, mientras que la memoria del mejor valor encontrado por alguna partícula de la población - *gbest* o *lbest* - es denominada influencia social. Ambas influencias determinan la velocidad de aprendizaje del cúmulo y esto permite determinar con qué rapidez el cúmulo convergerá a la solución.

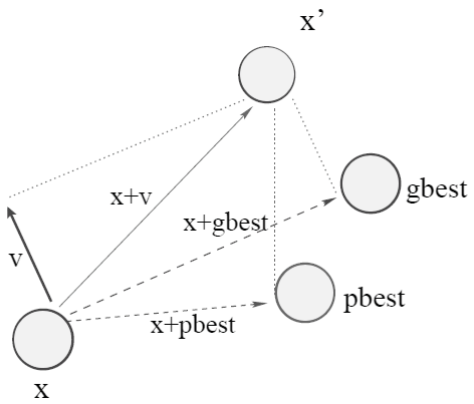

ILUSTRACIÓN 22: TRAYECTORIA PARTÍCULA X

Matemáticamente, el proceso de aprendizaje puede describirse utilizando una fórmula para la actualización de la velocidad de cada partícula y otra para la actualización de su posición dentro del espacio de búsqueda.

4.5 ALGORITMO

Considerando un problema de D-dimensiones, para cada partícula i de d dimensiones con $d \in [1, D]$, el aprendizaje está determinado por las siguientes ecuaciones:

$$v_{id} = v_{id} + \rho_1 r_1 (pbest_{id} - x_{id}) + \rho_2 r_2 (pbest_d - x_{id}) \quad (4.1)$$

$$x_{id} = x_{id} + v_{id} \quad (4.2)$$

donde v_{id} es el valor de la velocidad de la partícula i en la dimensiónd , ρ_1 es el factor de aprendizaje cognitivo, ρ_2 es el factor de aprendizaje social, r_1 y r_2 son valores aleatorios uniformemente distribuidos en el rango [0,1] , x_{id} es la posición corriente de la partícula i en la dimensión d, $pbest_{id}$ es el valor en la dimensión d de la partícula con el mejor valor objetivo encontrado por la partícula i , y $gbest_d$es el valor en la dimensión d del individuo del enjambre que encontró el mejor valor objetivo. Como puede observarse en la fórmula, un incremento de ρ_2 sobre ρ_1 aumenta la influencia del valor *gbest*, lo cual resulta en una mayor exploración del espacio de soluciones; decrementando el valor de ρ_2 sobre ρ_1 causa que la partícula se mueva en dirección más cercana a *pbest*, lo cual resulta

en una mayor **explotación** del espacio. Cuando se habla de exploración se considera la habilidad del algoritmo para explorar las diferentes regiones del espacio de búsqueda a fin de encontrar buenas soluciones. Mientras que la explotación es la habilidad que el algoritmo posee de concentrarse en una porción del espacio que sea promisoria con el fin de refinar soluciones buenas, y encontrar posiblemente el óptimo.

Además de estas fórmulas, un algoritmo PSO cuenta con los siguientes parámetros:

- *N*: número de partículas involucradas en la resolución del problema.
- *D*: número de variables del problema, equivalente al número de dimensiones de una partícula.
- $Xmax$ y $Xmin$: parámetros que limitan el área de búsqueda.
- $Vmax$ y $Vmin$: parámetros opcionales que limitan la velocidad de las partículas. Son opcionales porque si bien es necesaria una restricción de los valores que la velocidad puede alcanzar, existen otros métodos como el factor de constricción que cumplen con la finalidad de limitar dichos valores.
- *Condición de Parada* del algoritmo: puede usarse un criterio que establezca un nivel de error aceptable entre el ideal y el óptimo obtenido, una cantidad máxima de iteraciones - ciclos de vuelo - o alguna otra que se prefiera.

Observando el Algoritmo, la fase de Inicialización del enjambre asigna a cada dimensión de cada partícula de la población un valor aleatorio en el rango establecido por $[Xmin, Xmax]$. Este rango depende exclusivamente del problema que se pretende resolver con la heurística. Lo mismo sucede con los vectores de velocidad - uno por cada partícula -, los que son inicializados en un rango $[Vmin, Vmax]$. Los valores objetivo son calculados y almacenados. A cada partícula *pbest* se le copia el valor de cada dimensión de su individuo correspondiente, al igual que la partícula *gbest* a la que se le asigna el individuo con mejor valor objetivo. La segunda y más importante es la fase de búsqueda, en la cual el enjambre recorre el espacio para hallar la mejor solución posible. En un proceso repetitivo - hasta que se cumpla una condición de parada -, se actualizan las fórmulas de aprendizaje - velocidad y posición - con base en los valores actuales de *pbest* y *gbest*. Los valores de fitness son recalculados y utilizados para remplazar las partículas *pbest* y *gbest*, si es que en la presente iteración se obtuvieron mejores valores objetivo.

(* Fase de Inicialización del enjambre *)

PARA i HASTA N HACER
    PARA d HASTA D HACER
        $x_{id} = Random(Xmin, Xmax)$
        $pbest_{id} = x_{id}$
        $v_{id} = Random(Xmin, Xmax)$
    FIN
    $fitness_{x_i} = Fitness(x_i)$
    $fitness_{pbest_i} = fitness_{x_i}$
    SI $fitness_{x_i}$ mejor $fitness_{gbest}$ ENTONCES
        $gbest = x_i$
    FIN
FIN

ILUSTRACIÓN 23: FASE DE INICIALIZACIÓN DEL ENJAMBRE

(* Fase de Búsqueda *)

MIENTRAS no se cumpla condición de parada HACER
PARA i HASTA N HACER
PARA d HASTA D HACER
$x_{id} = Posicion()$
$v_{id} = Velocidad()$
FIN
$fitness_{x_i} = Fitness(x_i)$
SI $fitness_{x_i}$ mejor $fitness_{gbest}$
$gbest = x_i$
FIN
SI $fitness_{x_i}$ mejor $pbest_i$ ENTONCES
$pbest_{x_i} = x_i$
$fitness_{pbest_i} = Fitness(pbest_i)$
FIN
FIN
FIN

ILUSTRACIÓN 24: FASE DE BÚSQUEDA DEL ENJAMBRE

4.6 VECINDARIOS EN PSO

Una de las principales características que posee PSO es la interacción social que se observa entre los individuos. Por eso es importante el estudio de las posibles estructuras sociales que pueden incluirse en PSO. Todos los individuos pertenecientes a la misma estructura social comparten información, aprenden y siguen a los mejores dentro de su propia estructura.

La comunicación entre los individuos de un grupo o población así como su desempeño, están íntimamente ligados a la estructura social que posee el grupo. Así también, en PSO, dichas interacciones se observan particularmente entre los vecinos inmediatos adyacentes, y son estas interacciones las que provocan que el algoritmo funcione, ya que cada partícula por sí misma no podría evolucionar demasiado.

La estructura más simple que se puede plantear es aquella donde todos los individuos son influenciados exactamente de la misma forma, por la mejor solución encontrada por algún miembro de la población. Este tipo de modelo recibe el nombre de *gbest* - mejor global -, y es equivalente a una estructura donde todos los individuos están conectados entre sí. Otra estructura que surge naturalmente recibe el nombre de *lbest* - mejor local -.

La elección de la estructura social es uno de los principales inconvenientes que presenta la implementación de PSO, debido a la importancia y fuerte influencia que ésta ejerce en el desempeño del algoritmo. El modelo *gbest* tiende a converger prematuramente, debido a la pérdida de la diversidad. El modelo *lbest* es más lento de converger, pero conserva la diversidad [Kennedy, 2001]. Los resultados obtenidos con algoritmos que incluyen vecindarios suelen ser, generalmente, mejores cuando son comparados con los obtenidos con modelos *gbest*.

Varias estructuras *lbest* pueden ser utilizadas, algunas de estas son [Kennedy, 1999]:

• Circular o Anular: cada individuo está conectado con sus k vecinos inmediatos, intentando imitar al mejor de ellos. En la *Figura a* puede observarse un ejemplo con $k = 2$. En este tipo de estructura, cada vecindario se encuentra solapado para facilitar el intercambio de información y, de esta manera, converger a una única solución al final del

proceso de búsqueda. La información entre vecindarios fluye lentamente, lo cual provoca que la convergencia sea más lenta pero al mismo tiempo más segura. En funciones multimodales - aquéllas que poseen varios óptimos - este tipo de estructura suele obtener un buen desempeño con respecto a otras estructuras. Muchas variantes de este tipo de estructura pueden ser creadas, en donde la conexión de las partículas puede ser elegida aleatoriamente o siguiendo algún criterio, dependiendo del problema. La *Figura b* muestra un ejemplo.

- Rueda: un único individuo - el central - está conectado a todos los demás, y todos éstos están conectados solamente al individuo central. La *Figura c* ilustra el concepto. En esta estructura una única partícula sirve de punto focal, y toda la información entre los individuos es canalizada por ese punto. La partícula focal compara el desempeño de todas las partículas en el vecindario para ajustar las posiciones a la mejor. Tan pronto como la mejor posición sea detectada, la información es enviada a todas las partículas para que efectúen el cambio. La estructura de rueda propaga las buenas soluciones en forma demasiado lenta lo cual puede no ser del todo bueno, porque el proceso de búsqueda se tornaría demasiado lento. Algunas variantes pueden ser introducidas, como desconectar algunas partículas de la partícula focal y conectarlas a otras partículas en el vecindario - *Figura d-*.
- Arcos Aleatorios: para N partículas, N conexiones simétricas son asignadas entre pares de individuos. Como su nombre lo indica, las conexiones son asignadas de forma aleatoria entre las N partículas. La *Figura e* muestra esta estructura para un tamaño de población de 12 individuos.
- Pirámide: la idea de esta topología es la creación de una estructura tridimensional que comunique a las partículas. Se asemeja a un modelo de alambre - wire-frame - tridimensional, como se observa en la *Figura f*.
- Toroidal: Esta topología puede utilizarse como estructura de vecindario [Gardner, 1975]. Es una superficie cerrada con base en la topología de malla - *Figura g-* pero a diferencia de esta última donde cada partícula posee dos vecinos, en la toroidal posee 4 vecinos directos - *Figura h-*.

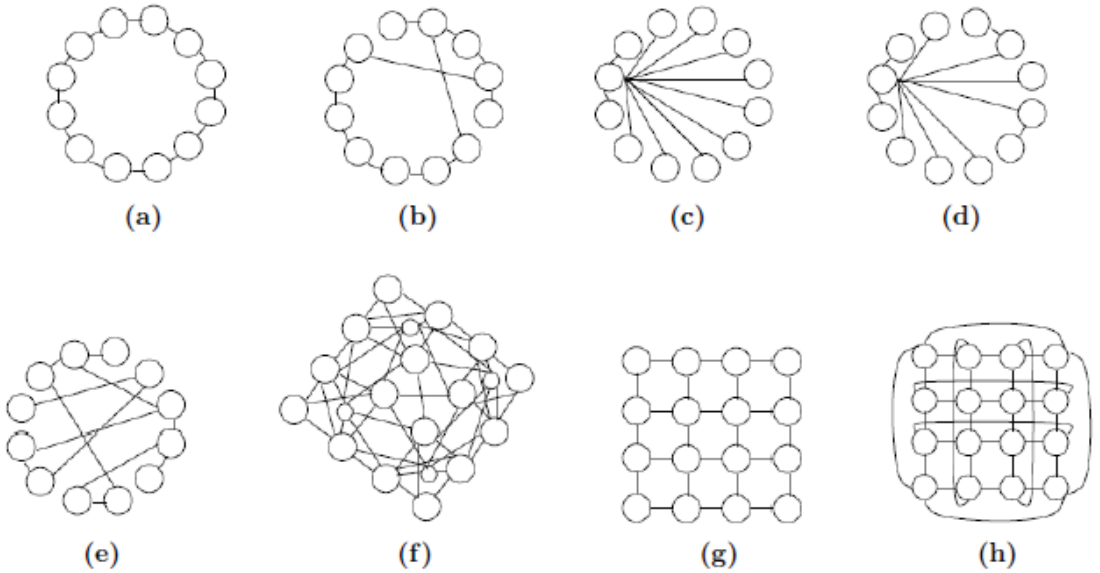

ILUSTRACIÓN 25: TOPOLOGÍAS DE VECINDARIOS DE PARTÍCULAS

Cada una de estas topologías, a su vez, puede variar algunas de sus conexiones, para constituir subvecindarios. Lo importante es lograr una topología que mejore el desempeño del algoritmo. Actualmente, no existe una topología que sea buena para todo tipo de problema. Generalmente, las estructuras completamente conectadas son mejores para problemas unimodales - un único óptimo -, mientras que las menos conectadas funcionan mejor para problemas multimodales, dependiendo del grado de interconexión entre las partículas [Kennedy, 1999] y [Peer, 2003].

4.7 MODIFICACIONES DEL ALGORITMO PSO

Durante la primera década de la utilización del algoritmo PSO, aparecieron numerosos trabajos que, partiendo del enjambre original, proponían modificaciones con el fin de mejorar sus características, normalmente de forma empírica. Asimismo aparecieron algunos trabajos que realizaron análisis teóricos de las ecuaciones del enjambre, con el fin de comprender mejor el mecanismo de su funcionamiento: la estabilidad, convergencia, estudio de trayectorias e influencia de los parámetros.

Algunos de los problemas de las primeras versiones de PSO se debían a su sensibilidad a la elección de los parámetros. Los valores escogidos no sólo condicionaban la posibilidad de encontrar el óptimo global de la función de fitness; se descubrió además el fenómeno de explosión del enjambre. Éste consiste en que, con ciertos valores de los parámetros, las velocidades de las partículas - v_i - tienden a crecer hasta valores elevados, dispersando el enjambre de forma que la distancia entre partículas era cada vez mayor.

Con el fin de estudiar la convergencia del algoritmo, y limitar la explosión del enjambre, se estudiaron varios mecanismos:

4.7.1 CONTROL DE VELOCIDAD

Consiste simplemente en fijar un límite superior al valor absoluto de las componentes de velocidad. Es decir, se fijaba un parámetro $Vmax$ y se imponía la condición de limitación siguiente:

$$v_i^{t+1} = min(v_i^{t+1}, Vmax) \tag{4.3}$$

Este mecanismo tiene el problema de que el parámetro $Vmax$ depende del tamaño del espacio de búsqueda $Xmax$, ya que en espacios de búsqueda grandes, velocidades pequeñas supondrían que el enjambre no recorrería todo el espacio o necesitaría un número de iteraciones elevado para hacerlo. En los trabajos iniciales se solía proponer el valor $Vmax = Xmax$.

4.7.2 FACTOR DE INERCIA (PSO-IW)

Se introdujo un parámetro w llamado factor de inercia [Shi, 1998]. Se argumenta que el término de inercia de la ecuación es importante para que el enjambre pueda realizar búsquedas globales, más allá de las fronteras del espacio definido por las posiciones de las partículas iniciales. La ecuación de velocidad se remplaza por:

$$v_{id} = wv_{id} + \rho_1 r_1(pbest_{id} - x_{id}) + \rho_2 r_2(pbest_d - x_{id}) \qquad (4.4)$$

Los autores proponen incluir un parámetro explícito con el objetivo de permitir calibrar el equilibrio entre exploración y explotación, al añadir la posibilidad de variar el valor de dicho parámetro en función de la iteración o estado del algoritmo. En su análisis concluyen que, cuando el valor de w es pequeño, el enjambre tiende a comportarse como un algoritmo que realiza búsquedas locales, y depende con más fuerza de la inicialización. Cuando *w* crece, pasa a realizar una búsqueda más global, pero encuentra el óptimo con más dificultad y mayor número de iteraciones. Mediante el estudio experimental realizado, se propone un valor de compromiso 0.9< *w* <1.2.

En el trabajo, se propone también el uso de un valor de *w* decreciente con el tiempo. Con posterioridad, esta propuesta se analizó de forma detallada en [Shi, 1999]. Aunque en este estudio el factor de inercia decreciente parece mejorar el comportamiento del enjambre, en [Zheng, 2003] se muestra un estudio en el que PSO tiene mejores resultados cuando se utiliza un valor de *w* que crece con el tiempo en lugar de decrecer. Por lo tanto, el comportamiento resulta ser dependiente de los problemas considerados.

4.7.3 Parámetros de Constricción

Se propone una alternativa a la inclusión del factor de inercia [Clerc, 1999], que se estudia con más detalle en [Clerc, 2002]. En este trabajo se realizó un estudio teórico de la dinámica del enjambre y se observó que se podía garantizar la convergencia del algoritmo en ciertas condiciones mediante la inclusión de parámetros adicionales llamados parámetros de constricción - *c* -. Estos resultados se obtienen mediante el análisis de versiones simplificadas del enjambre, y específicamente versiones en las que se eliminan los factores aleatorios. Con posterioridad se confirman estas propiedades empíricamente usando la versión completa del enjambre. Los parámetros de constricción son una alternativa al mecanismo de inercia y de limitación de velocidad. La ecuación de velocidad se remplaza por la función de constricción.

$$\begin{aligned} v_{id} &= v_{id} + \rho_1 r_1(pbest_{id} - x_{id}) + \rho_2 r_2(pbest_d - x_{id}) \\ v_{id} &= \chi(v_{id} + \rho_1 r_1(pbest_{id} - x_{id}) + \rho_2 r_2(pbest_d - x_{id})) \end{aligned} \qquad (4.5)$$

Donde el factor χ se define como:

$$\chi = \frac{2k}{\sqrt{|2 - \emptyset - \sqrt{\emptyset(\emptyset - 4)}|}} \qquad (4.6)$$

Siendo:

$$\emptyset = \emptyset_1 + \emptyset_2$$
$$\emptyset > 4$$
$$\emptyset_1 = \rho_1$$
$$\emptyset_2 = \rho_2$$
$$k \in [0,1]$$

El parámetro k en la ecuación controla la exploración y explotación del cúmulo. Para valores de k cercanos a 0, se obtiene una rápida convergencia con explotación local. Para valores de k cercanos a 1, la convergencia se produce más lentamente y se experimenta un grado mayor de exploración. Aunque el valor de k es constante, algunas variaciones comienzan con valores altos y, a medida que transcurren los ciclos, se va decrementando a fin de aumentar la explotación.

4.7.4 UNIFIED PARTICLE SWARM OPTIMIZATION (UPSO)

El desarrollo del UPSO fue motivado por la intención de combinar las variantes de exploración y explotación que se ejecutan en PSO normal de una manera generalizada, con el objetivo de generar nuevos esquemas que combinan sus propiedades.

UPSO fue desarrollado por Parsopoulos y Vrahatis (2004) como un algoritmo que aprovecha la variante de PSO global y local en un esquema unificado, sin imponer una carga computacional adicional en términos de evaluaciones de la función. El coeficiente de constricción de actualización de velocidad de PSO fue utilizado, aunque el régimen unificado se puede definir para diferentes esquemas también. Sea N el tamaño del enjambre, y n denota la dimensión del problema en cuestión. Además, $G_i(t+1)$ denota la actualización de velocidad de la partícula i-ésima, x_i , para la variante global de PSO con coeficiente de constricción, que se define como:

$$G_{ij}(t+1) = \chi[v_{ij}(t) + c_1R_1\left(p_{ij}(t) - x_{ij}(t)\right) + c_2R_2\left(p_{gj}(t) - x_{ij}(t)\right)] \quad (4.7)$$

$$\text{y } L_i(t+1) = \chi[v_{ij}(t) + c_1R_1\left(p_{ij}(t) - x_{ij}(t)\right) + c_2R_2\left(p_{lj}(t) - x_{ij}(t)\right)] \quad (4.8)$$

$$\chi = \frac{2}{2-\varphi-\sqrt{\varphi^2-4\varphi}} \quad (4.9)$$

$$\varphi = c_1 + c_2, y\ \varphi > 4$$

Donde g denota el índice de la mejor posición general, mientras que l denota la mejor posición en el barrio de x_i. Entonces, el principal sistema de UPSO se define como [Parsopoulos y Vrahatis, 2004]:

$$v_{ij}(t+1) = uG_{ij}(t+1) + (1-u)L_{ij}(t+1) \quad (4.10)$$
$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1)$$

Donde $u \epsilon$ [0,1] es un nuevo parámetro, denominado factor de unificación, que controla la influencia de la velocidad de actualización global y lo local. El resto de los parámetros son los mismos que para el PSO estándar.

De acuerdo con las ecuaciones anteriores, el cambio de la nueva posición de una partícula en UPSO consta de una combinación ponderada de los cambios de posición gbest y lbest. El factor de unificación equilibra la influencia de estas dos direcciones de búsqueda. Todos los valores intermedios, $u \epsilon$ (0,1), define variantes UPSO que combinan las propiedades de exploración / explotación de gbest y lbest.

La actualización del valor u será un incremento linear que va desde 0 hasta 1, de acuerdo a la siguiente formula obtenida del estudio de Parsopoulos and Vrahatis (2007):

$$u(t) = \frac{t}{T_{max}} \tag{4.11}$$

Donde t es la iteración actual y T_{max} es el total de iteraciones.

4.7.5 Quantum Particle Swarm Optimization (QPSO)

Según el modelo clásico de PSO, el estado de las partículas está dado por el vector de posición x_i y el vector de velocidad v_i, determinando así la trayectoria de las partículas, las que se mueven siguiendo los principios básicos de la mecánica Newtoniana. Sin embargo, si consideramos la mecánica cuántica, el comportamiento dinámico de la partícula difiere ampliamente del comportamiento tradicional de PSO, donde los valores exactos de velocidades y posiciones no pueden ser determinados simultáneamente.

Por lo tanto, si las partículas pertenecientes al enjambre tienen un comportamiento cuántico, el rendimiento del algoritmo PSO estará muy lejos del PSO clásico.

En el modelo cuántico de PSO, el estado de cada partícula está representado por la función de onda $\psi(x,t)$, en vez de la posición y la velocidad. De acuerdo al significado estadístico de la función de onda, la probabilidad de que una partícula aparezca en una determinada posición se puede obtener de la función de probabilidad de la densidad $\psi(x,t)^2$.

En QPSO, las partículas se mueven de acuerdo a las siguientes funciones:

$$x(t+1) = p + \beta * |m_{best} - x(t)| * \ln\left(\frac{1}{n}\right); Si\ k \geq 0.5 \tag{4.12}$$

$$x(t+1) = p - \beta * |m_{best} - x(t)| * \ln\left(\frac{1}{n}\right); Si\ k < 0.5 \tag{4.13}$$

Donde

$$p = r * P_{ij} + (1-r) * P_{gj} \tag{4.14}$$

$$m_{best} = \frac{1}{M}\sum_{i=1}^{M} p_i \tag{4.15}$$

$$\beta = (1-0.6) * \frac{T_{max} - iter}{T_{max}} + 0.6 \tag{4.16}$$

$$n, r\ y\ k\ valores\ aleatorios$$

El valor de x representa al enjambre en un espacio de dimensión K, P_{ij} corresponde a la mejor partícula previa, P_{gj} corresponde a la mejor partícula global del enjambre y m_{best} corresponde a la media de la población.

En PSO tenemos que cada una de las partículas del enjambre es una posible solución al problema, por lo que es necesario saber el tamaño de dicha partícula. En base a lo explicado en el capítulo anterior sobre la red neuronal wavelet radial, es que el tamaño está determinado de la siguiente manera.

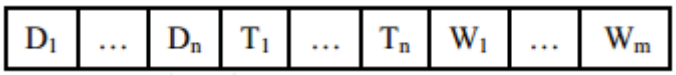

Ilustración 26: Dimensión de la Partícula

Donde el parámetro D será la dilatación, el parámetro T la traslación y el parámetro W serán los pesos desde los nodos ocultos hacia la salida. La dimensión será calculada de la siguiente manera:

$$Dimension = (Nodos\ Entrada * Nodos\ Ocultos * 2) + Nodos\ Ocultos$$

Y se explica de la siguiente manera: por cada nodo de entrada tenemos 2 valores que deben ir a cada nodo oculto, estos son los de dilatación (D) y traslación (T), y luego desde cada nodo oculto (W) hacia nuestra única salida.

CAPÍTULO 5

ESTIMADOR NEURONAL WAVELET RADIAL

Ya explicada la red neuronal, la función de transferencia y el algoritmo de aprendizaje, es hora de probar los distintos modelos y realizar las pruebas para la estimación de sus parámetros – nodos ocultos, iteraciones y partículas – para poder elegir con certeza el mejor modelo por cada configuración realizada.

Para poder realizar dicha elección, necesitamos de ciertas métricas que nos permitirán decidir el mejor modelo de cada configuración, y es lo que se explicara a continuación.

5.1 MÉTRICAS

En el presente estudio, para la selección de las topologías se utilizaron un conjunto de métricas de exactitud calculadas entre los datos observados (valores reales) y los datos pronosticados (entregados por la red). Estas métricas se nombran a continuación.

- Error Cuadrático Medio (Mean Squared Error, **MSE**): El error instantáneo (residuo) del modelo se define por la diferencia entre lo deseado d y lo pronosticado $\hat{y}$. Esto es: $e_i = d_i - \hat{y}_i$. Por lo tanto, el Error Cuadrático Medio (MSE) es:

$$MSE = \frac{1}{N}\sum_{i=1}^{N} e_i^2 \tag{5.1}$$

  Donde N representa el tamaño de la muestra.
- Raíz del Error Cuadrático Medio (Root Mean Squared Error, **RMSE**):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(d_i - \hat{y}_i)^2}{N}} \quad o \quad \sqrt[2]{MSE} \tag{5.2}$$

- Coeficiente de Correlación (R): El Coeficiente de correlación mide qué tanto se relacionan linealmente dos variables entre sí.

$$R = \frac{Cov(d,y)}{\sigma d * \sigma y} = \frac{<A,B>}{||A|| * ||B||} \tag{5.3}$$

- Coeficiente de Determinación (R^2): Mide la dependencia entre los datos reales y los pronosticados. El 0 muestra independencia y el 1 lo contrario.

$$R^2 = 1 - \frac{A}{B} \tag{5.4}$$

$$A = \sum_{i=1}^{N}(di - yi)^2 \quad B = \sum_{i=1}^{N}(di - \bar{d})^2 \tag{5.5}$$

- Error Porcentual Absoluto Medio (Mean Absolute Percentage Error, MAPE): Proporciona una indicación de que tan grandes son los errores de pronóstico comparados con los valores reales de la serie.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{di-yi}{di}\right| * 100\% \quad (5.6)$$

También dentro del análisis de los resultados obtenidos se utiliza otra métrica para hacer referencia al costo computacional que conlleva el entrenamiento del modelo. Corresponde al tiempo promedio de entrenamiento que presenta cuánto tarda en promedio un modelo en realizar el entrenamiento.

Finalmente, se debe destacar que el equipo en el cual se realizarán todos los entrenamientos y pruebas corresponde a un Intel Core i3 2.53 GHz y 3 GB de RAM

5.2 CONFIGURACIÓN DEL MODELO IW-PSO

El método de selección de las configuraciones de prueba fue ensayo y error, ingresando en principio las configuraciones que podrían presentar un buen desempeño e intentando mejorar las que ya lo hicieron.

Primero se estimó la cantidad de nodos ocultos, manteniendo las partículas y las iteraciones fijas, los nodos de entrada siempre serán cuatro. Luego se estimó la cantidad de iteraciones, fijando los demás parámetros. Y por último se seleccionó la cantidad de partículas que daban el mejor resultado de la red en la etapa de training. Una vez seleccionada la mejor configuración, se aplican las métricas anteriormente mencionadas.

Para la elección de todos los parámetros, se hizo lo siguiente: Se hicieron 20 repeticiones por configuración y se fueron variando una de las variables (los nodos ocultos, iteraciones y partículas), obteniendo un promedio entre el mayor y el menor error que registraron. Luego se seleccionó la que entregaba el promedio de error más bajo.

La función Fitness que se escogió para determinar el promedio de menor error es el error porcentual absoluto medio (MAPE). El primer modelo es el IW-PSO.

La elección de nodos ocultos se puede apreciar en la siguiente tabla:

Nodos Ocultos	MAPE mínimo	MAPE máximo	MAPE promedio
3	15,9368	36,6201	26,5403
4	**19,6097**	**35,5497**	**25,9736**
5	14,679	36,2813	26,3128
6	16,8323	40,6857	27,1701
7	16,4341	37,7363	27,94
8	18,1063	41,4566	27,7452
9	17,1206	37,0509	26,4234

TABLA 1: PRUEBA Y SELECCIÓN DE NODOS OCULTOS

Como se mencionó anteriormente, se hicieron 20 repeticiones por modelo para ver cuál entregaba el menor promedio, por lo que se eligió la configuración con 4 nodos ocultos y se dejaron fijas las partículas (15) y las iteraciones (100).

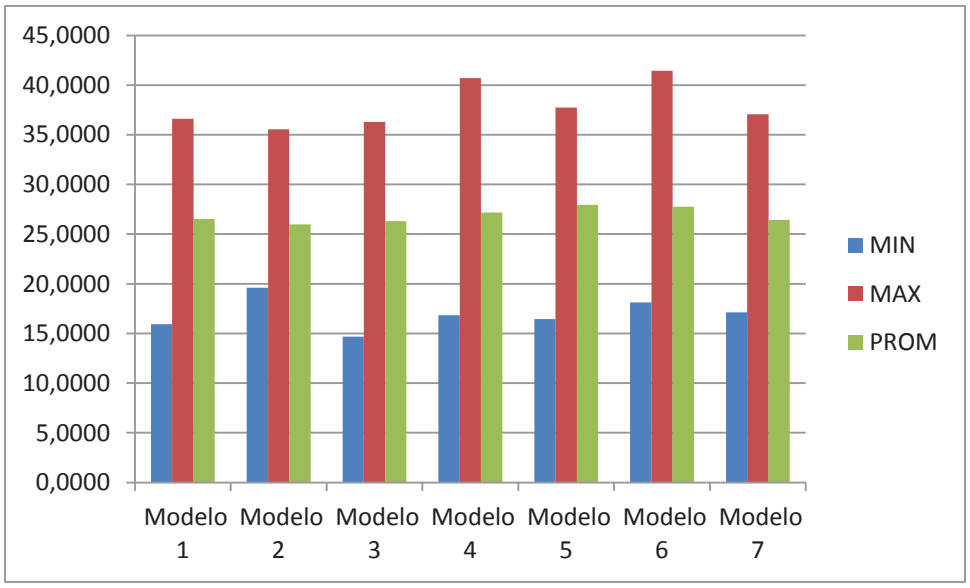

ILUSTRACIÓN 27: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE NODOS OCULTOS

Después de haber estimado el mejor número de nodos ocultos, se dio paso a la elección de iteraciones, dejando fijos los nodos ocultos en 4 y las partículas en 15.

Iteraciones	MAPE mínimo	MAPE máximo	MAPE promedio
100	14,0926	32,6801	21,6801
200	13,2448	32,8365	19,041
300	11,7553	29,5337	16,9435
400	**9,6388**	**27,0154**	**16,9184**
500	12,0908	27,2772	17,684

TABLA 2: PRUEBA Y SELECCIÓN DE ITERACIONES

Aquí se escogió la configuración con 400 iteraciones, ya que fue la que obtuvo el menor MAPE promedio, con un 16,9184.

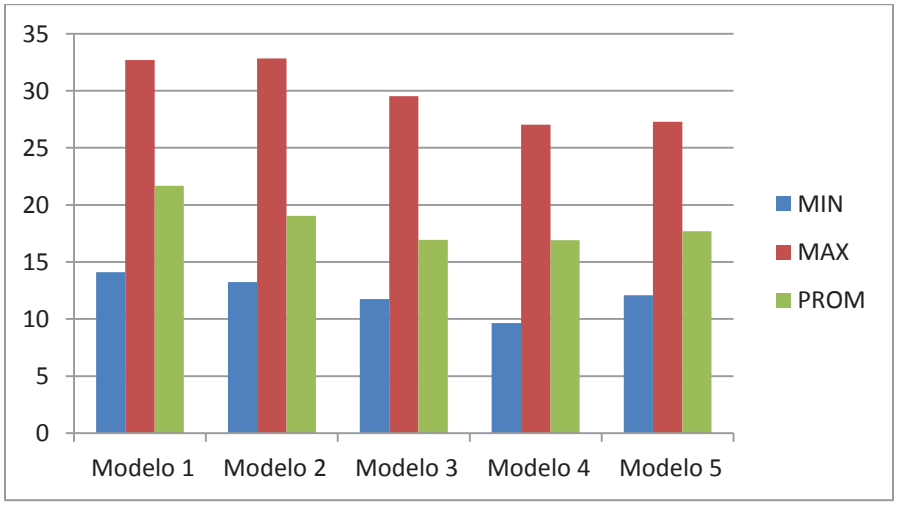

ILUSTRACIÓN 28: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE ITERACIONES

Finalmente, se estimaron las partículas dejando fijos los nodos ocultos en 4 y las iteraciones en 400.

Partículas	MAPE mínimo	MAPE máximo	MAPE promedio
15	19,6097	35,5497	25,9736
20	14,9557	35,9776	25,0629
25	13,663	33,3184	24,0269
30	15,4904	31,2521	24,1593
35	**14,0926**	**32,6801**	**21,6801**
40	13,9821	30,8818	25,0381
45	14,5544	31,8605	23,5788

TABLA 3: PRUEBA Y SELECCIÓN DE PARTÍCULAS

Se escogió la configuración con 35 partículas debido a que fue la que entrego el menor MAPE promedio.

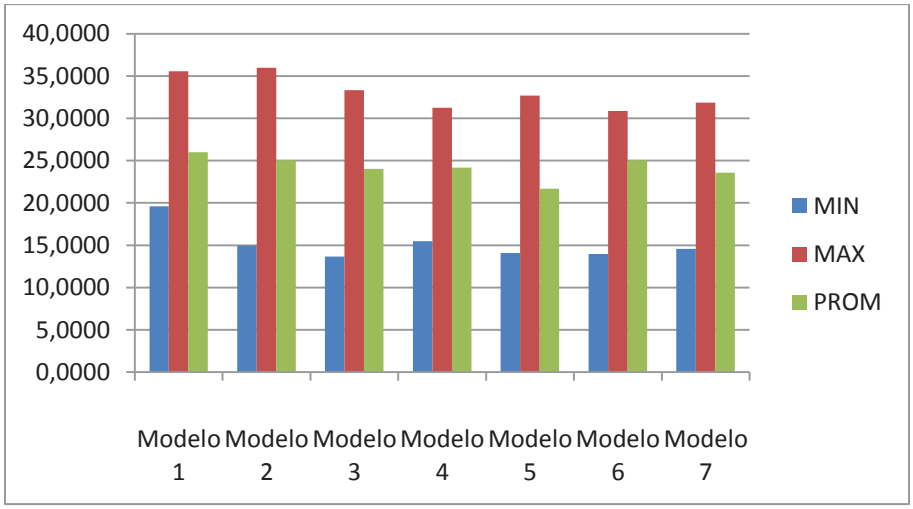

ILUSTRACIÓN 29: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE PARTÍCULAS

Finalmente la elección de los parámetros quedo establecida así:

Nodos Ocultos	4
Partículas	35
Iteraciones	400

5.2.1 TRAINING Y TESTING MODELO IW-PSO

Ya explicado el modelo y las métricas a utilizar, se dio paso a la realización de las pruebas de Training y Testing para ver el real nivel de estimación que posee la red, utilizando el algoritmo PSO.

En la etapa de Training la red nos entrega los siguientes resultados:

Error Cuadrático Medio (MSE)	0.0003
Raíz Error Cuadrático Medio (RMSE)	0.0169
Coeficiente de Correlación (R)	0.9844
Coeficiente de Determinación (R^2)	0.9690
Error Porcentual Absoluto Medio (MAPE)	9.6388

TABLA 4: DATOS TRAINING

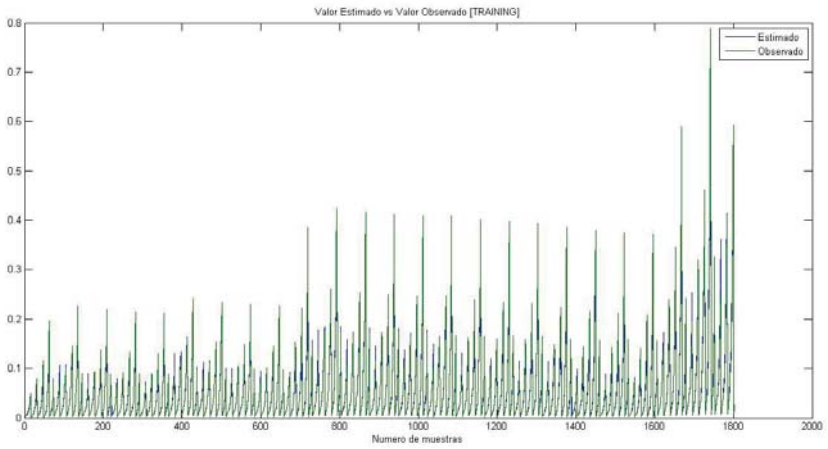

ILUSTRACIÓN 30: GRÁFICA DE TRAINING

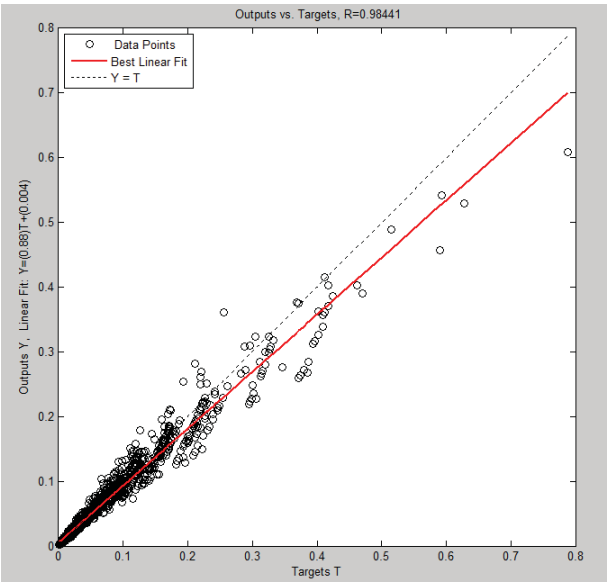

ILUSTRACIÓN 31: GRÁFICA COEFICIENTE DE CORRELACIÓN TRAINING

En la etapa de Testing se obtuvieron los siguientes valores:

Error Cuadrático Medio (MSE)	0.0012
Raíz Error Cuadrático Medio (RMSE)	0.0348
Coeficiente de Correlación (R)	0.9804
Coeficiente de Determinación (R^2)	0.9611
Error Porcentual Absoluto Medio (MAPE)	12.9274

TABLA 5: DATOS TESTING

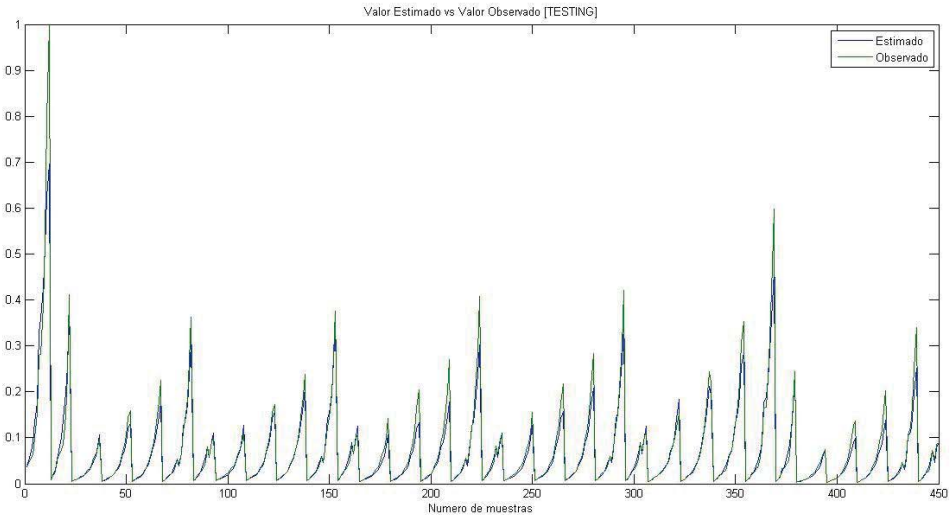

ILUSTRACIÓN 32: GRÁFICA DE TESTING

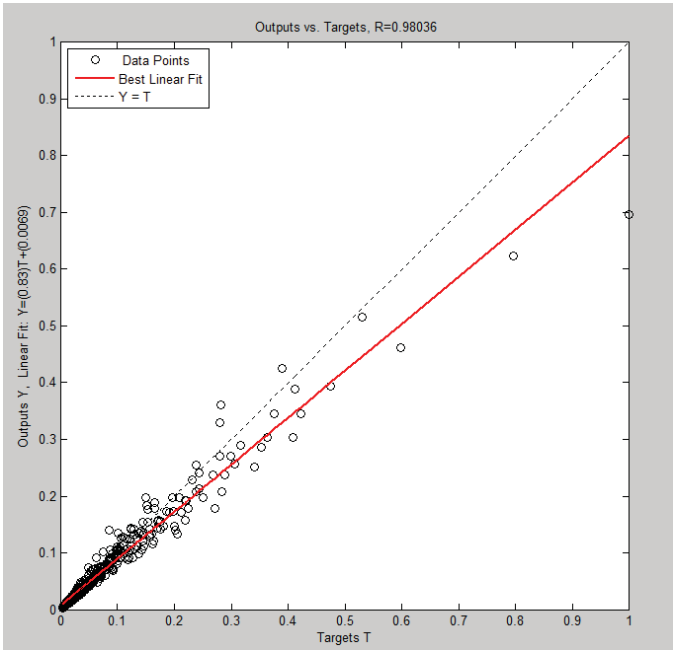

ILUSTRACIÓN 33: GRÁFICA COEFICIENTE DE CORRELACIÓN TESTING

5.3 CONFIGURACIÓN DEL MODELO QPSO

Ya teniendo los resultados del primer modelo, damos paso al segundo modelo para ver si tiene mejores resultados. Este modelo es el QPSO y se harán las mismas pruebas de ensayo y error para lograr estimar cada uno de los parámetros.

Para la elección de todos los parámetros, se realizó lo siguiente: Se hicieron 20 repeticiones por configuración y se fue modificando cada una de las variables (los nodos ocultos, iteraciones y partículas), obteniendo un promedio entre el mayor y el menor error que registraron. Luego se seleccionó la que entregaba el promedio de error más bajo.

La función Fitness que se escogió para determinar el promedio de menor error es el error porcentual absoluto medio (MAPE).

La elección de nodos fue la siguiente:

Nodos Ocultos	MAPE mínimo	MAPE máximo	MAPE promedio
3	**28,5636**	**73,9943**	**51,27895**
4	44,1576	74,8006	59,4791
5	40,4411	72,4068	56,42395
6	42,377	80,2567	61,31685
7	40,6882	82,6404	61,6643
8	45,4196	76,7375	61,07855
9	38,4396	81,7398	60,0897

TABLA 6: PRUEBA Y SELECCIÓN DE NODOS OCULTOS

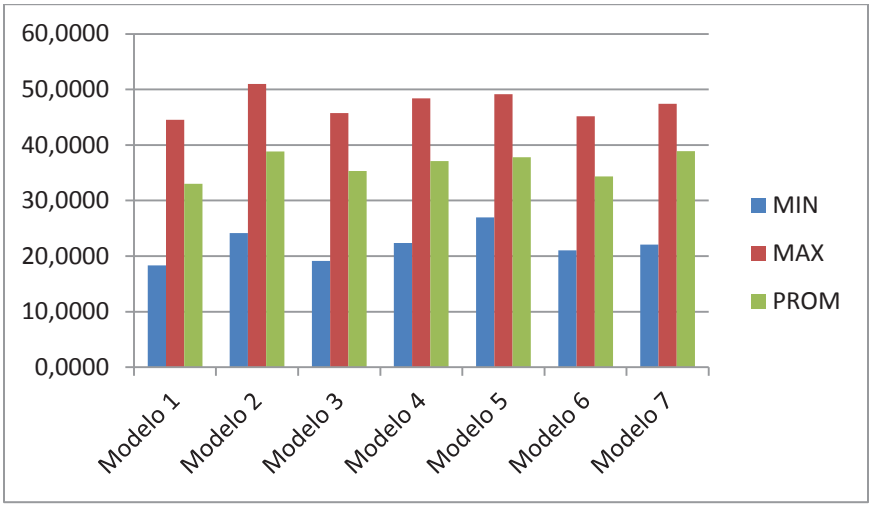

ILUSTRACIÓN 34: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE NODOS OCULTOS

Donde se obtuvo que el menor MAPE promedio fuera de 51, con 5 nodos ocultos dejando fijos los valores de las partículas (15) y las iteraciones (100).

Ya teniendo el número de nodos ocultos, y aún con las iteraciones en 100, damos paso a la elección de partículas, la cual se muestra en la siguiente tabla:

Partículas	MAPE mínimo	MAPE máximo	MAPE promedio
15	18,3185	44,5499	33,0346
20	21,9413	46,9336	31,2349
25	17,3536	39,575	30,0117
30	17,4929	45,798	32,0956
35	24,8234	38,2739	31,8829
40	**16,8921**	**36,8921**	**28,3784**

TABLA 7: PRUEBA Y SELECCIÓN DE PARTÍCULAS

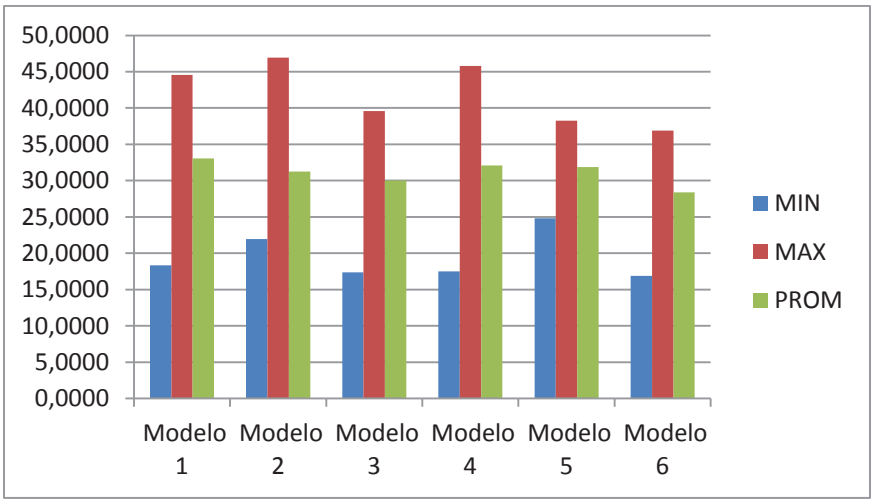

ILUSTRACIÓN 35: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE PARTÍCULAS

Donde se eligió el modelo con 40 partículas, dejando fijos los nodos ocultos (5) obtenidos en la estimación anterior y las iteraciones en 100.

Ya habiendo elegido el número de nodos ocultos y las iteraciones, se da paso a la elección del número de partículas que tendrá el modelo. El detalle se muestra a continuación:

Iteraciones	MAPE mínimo	MAPE máximo	MAPE promedio
100	16,8921	36,8921	28,3784
200	14,5866	34,973	26,1553
300	17,162	41,5835	35,9609
400	13,1406	38,6302	23,7551
500	**14,9247**	**33,0999**	**23,0088**

TABLA 8: PRUEBA Y SELECCIÓN DE ITERACIONES

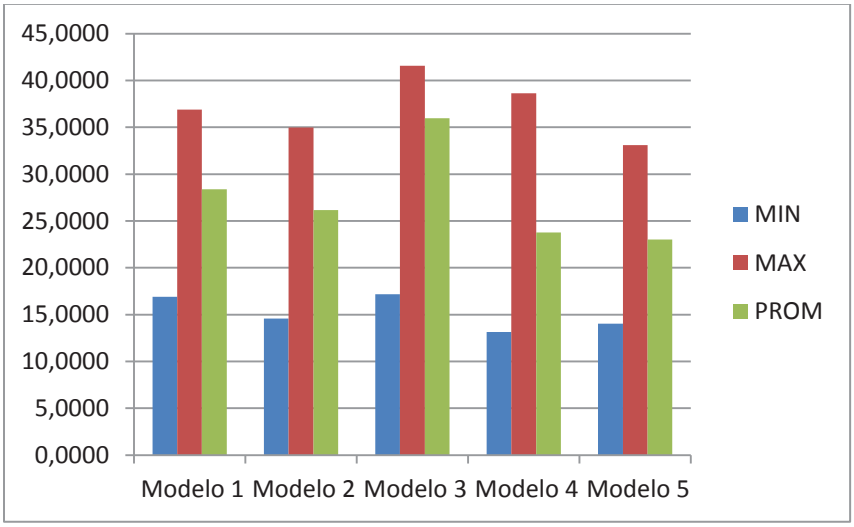

ILUSTRACIÓN 36: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE ITERACIONES

Con lo que el modelo final de QPSO queda con los siguientes parámetros

Nodos Ocultos	3
Partículas	40
Iteraciones	500

5.3.1 TRAINING Y TESTING MODELO QPSO

Ya explicado el modelo, las métricas a utilizar, se pasará a realizar la prueba de Training y Testing para ver el real nivel de estimación que posee la red, utilizando el algoritmo QPSO.

En la etapa de Training la red nos entrega los siguientes resultados:

Error Cuadrático Medio (MSE)	0.0002
Raíz Error Cuadrático Medio (RMSE)	0.0150
Coeficiente de Correlación (R)	0.9827
Coeficiente de Determinación (R^2)	0.9656
Error Porcentual Absoluto Medio (MAPE)	14.5569

TABLA 9: DATOS TRAINING

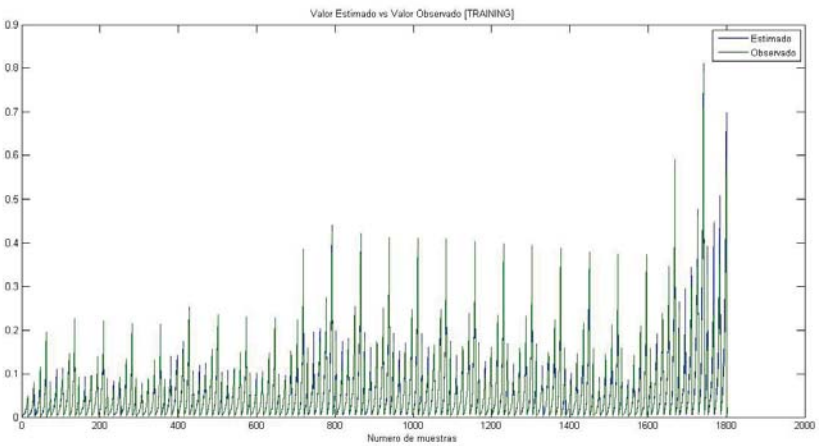

ILUSTRACIÓN 37: GRÁFICA DE TRAINING

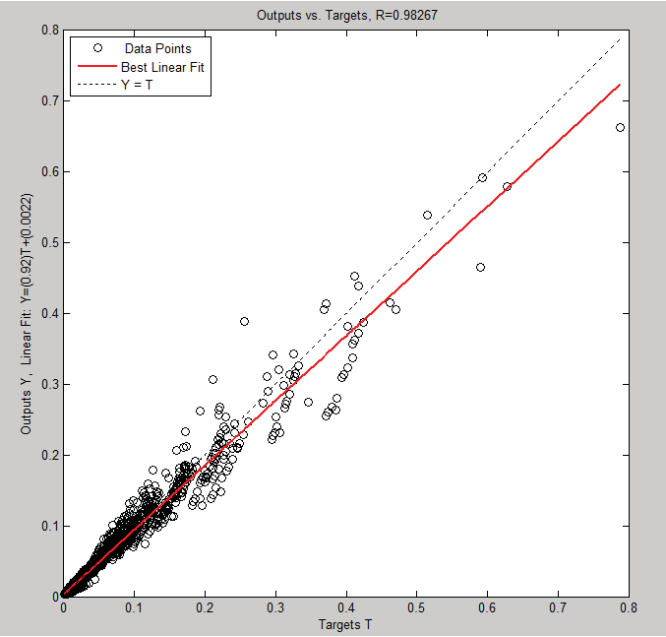

ILUSTRACIÓN 38: GRÁFICA COEFICIENTE DE CORRELACIÓN TRAINING

En la etapa de Testing se obtuvieron los siguientes valores:

Error Cuadrático Medio (MSE)	0.0004
Raíz Error Cuadrático Medio (RMSE)	0.0203
Coeficiente de Correlación (R)	0.9844
Coeficiente de Determinación (R^2)	0.9690
Error Porcentual Absoluto Medio (MAPE)	14.9247

TABLA 10: DATOS TESTING

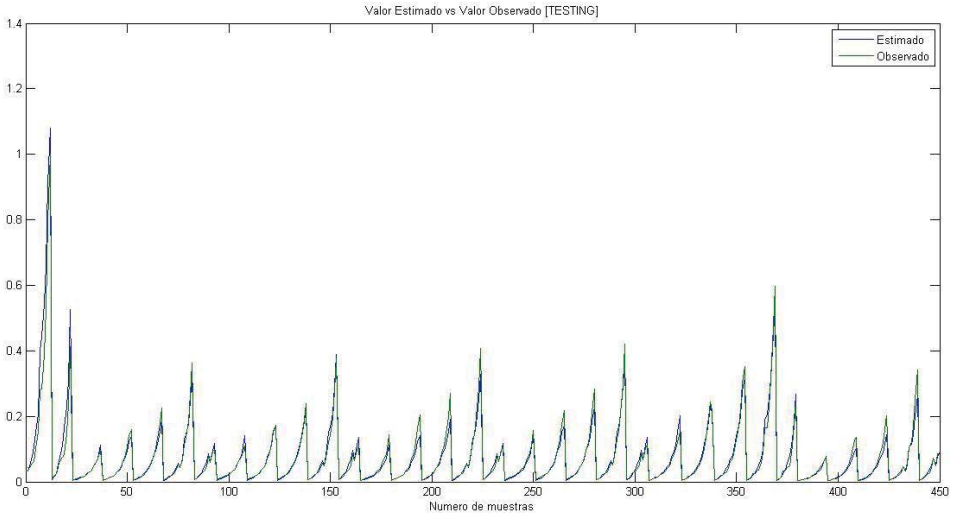

ILUSTRACIÓN 39: GRÁFICA DE TESTING

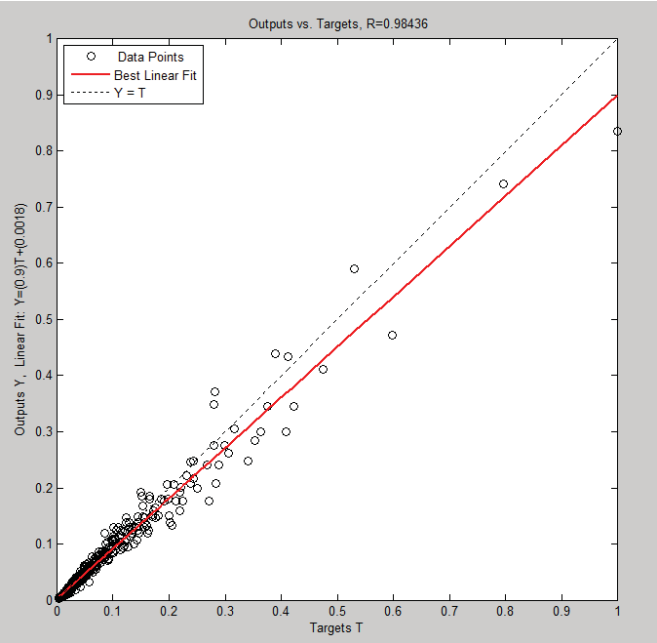

ILUSTRACIÓN 40: GRÁFICA COEFICIENTE DE CORRELACIÓN TESTING

5.4 CONFIGURACIÓN MODELO UPSO

El tercer modelo, utilizando UPSO, al igual que los modelos anteriores, se basara en un enfoque de prueba y error para cada uno de sus parámetros, donde la medida de error a utilizar será el error porcentual medio MAPE, donde cada parámetro con el MAPE más pequeño será el que se ha de utilizar.

La selección de nodos fue la siguiente:

Nodos Ocultos	MAPE mínimo	MAPE máximo	MAPE promedio
3	35,1496	73,8113	54,48045
4	**40,5256**	**62,8662**	**51,6959**
5	31,527	72,6924	52,1097
6	35,5947	72,5742	54,08445
7	30,4616	79,4616	54,9616
8	38,2171	75,1599	56,6885
9	34,2832	78,46	56,3716

TABLA 11: PRUEBA Y SELECCIÓN DE NODOS OCULTOS

Donde se escogió el modelo con 4 nodos ocultos, dejando las iteraciones fijas en 100 y las partículas quietas en 15.

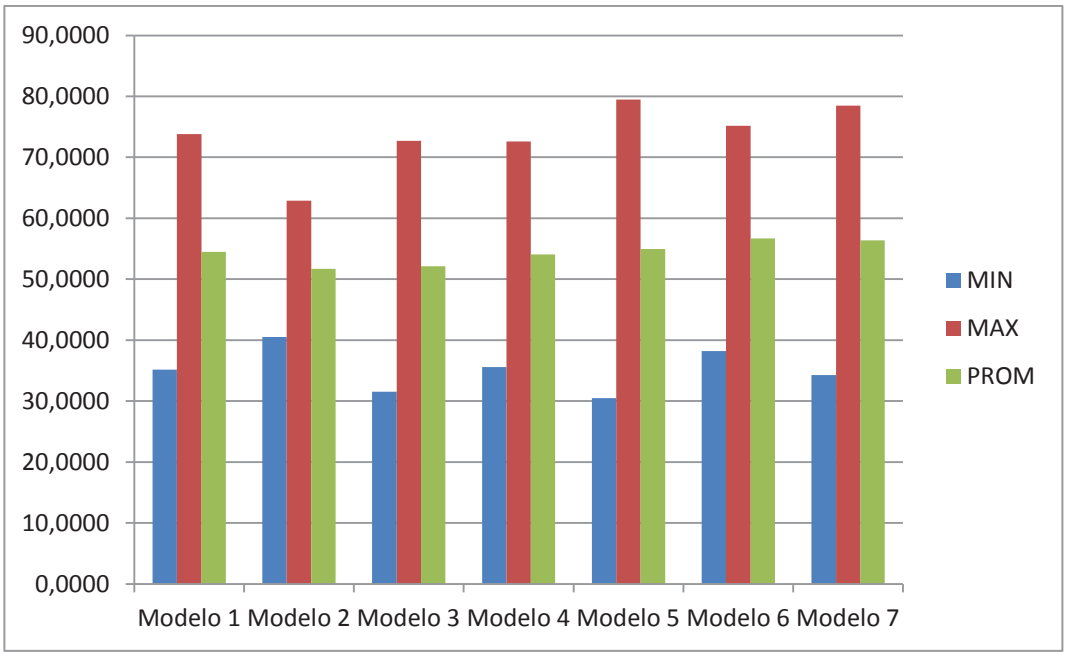

ILUSTRACIÓN 41: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE NODOS OCULTOS

Luego se dio paso a la elección de partículas, la cual fue la siguiente:

Partículas	MAPE mínimo	MAPE máximo	MAPE promedio
15	40,5256	62,8662	51,6959
20	34,6455	75,7112	55,17835
25	38,8731	63,838	51,35555
30	42,0143	68,7324	55,37335
35	38,1549	67,9406	53,04775
40	32,6516	64,8839	48,76775
45	**30,91**	**62,8016**	**46,8558**

TABLA 12: PRUEBA Y SELECCIÓN DE PARTÍCULAS

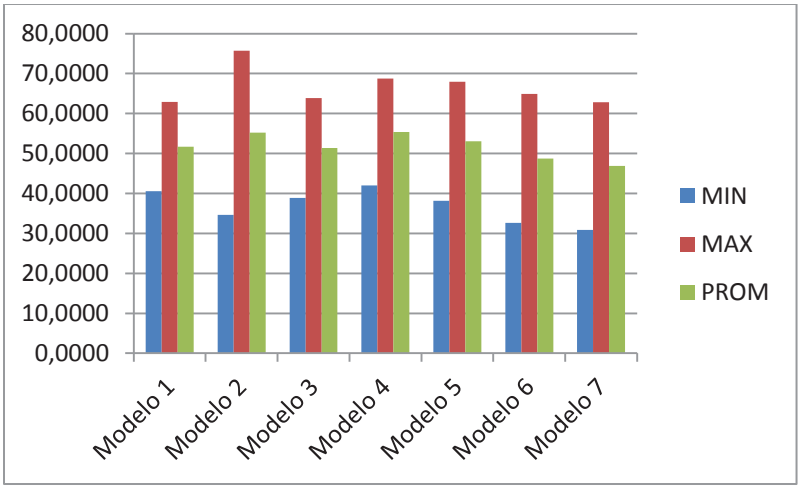

ILUSTRACIÓN 42: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE PARTÍCULAS

Estos resultados se obtuvieron dejando fijos los nodos ocultos en 4 (obtenidos en la estimación anterior) y las iteraciones en 100. En el último paso se elegirán el total de iteraciones, donde se obtuvieron los siguientes valores:

Iteraciones	MAPE mínimo	MAPE máximo	MAPE promedio
100	30,91	62,8016	46,8558
200	**31,172**	**57,4877**	**44,32985**
300	33,1214	57,5187	45,32005
400	36,1981	62,1017	49,1499
500	33,6754	57,5855	45,63045

TABLA 13: PRUEBA Y SELECCIÓN DE ITERACIONES

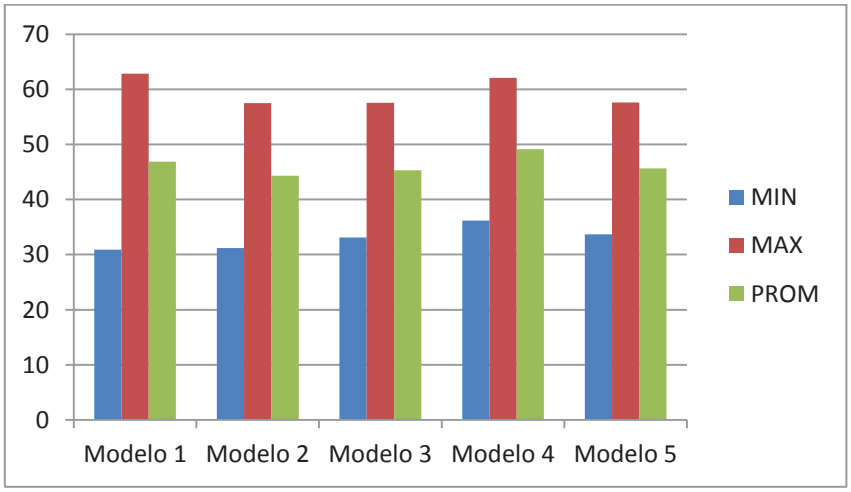

ILUSTRACIÓN 43: MÁXIMO, MÍNIMO Y PROMEDIO DEL MAPE EN LA ELECCIÓN DE ITERACIONES

Finalmente tenemos que los parámetros a utilizar en nuestro modelo final son los siguientes:

Nodos Ocultos	4
Partículas	45
Iteraciones	200

5.4.1 TRAINING Y TESTING MODELO UPSO

Ya explicado el modelo, las métricas a utilizar, se realizará la prueba de Training y Testing para ver el real nivel de estimación que posee la red, utilizando el algoritmo UPSO.

En la etapa de Training la red nos entrega los siguientes resultados:

Error Cuadrático Medio (MSE)	0.0027
Raíz Error Cuadrático Medio (RMSE)	0.0515
Coeficiente de Correlación (R)	0.9634
Coeficiente de Determinación (R^2)	0.9282
Error Porcentual Absoluto Medio (MAPE)	31,172

TABLA 14: DATOS TRAINING

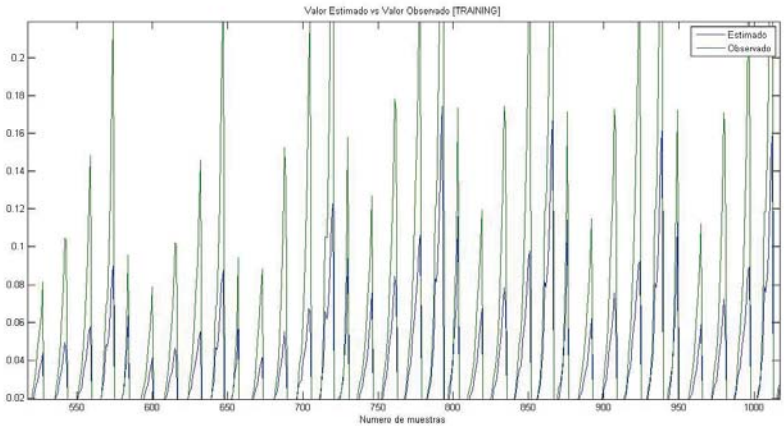

ILUSTRACIÓN 44: GRÁFICA TRAINING

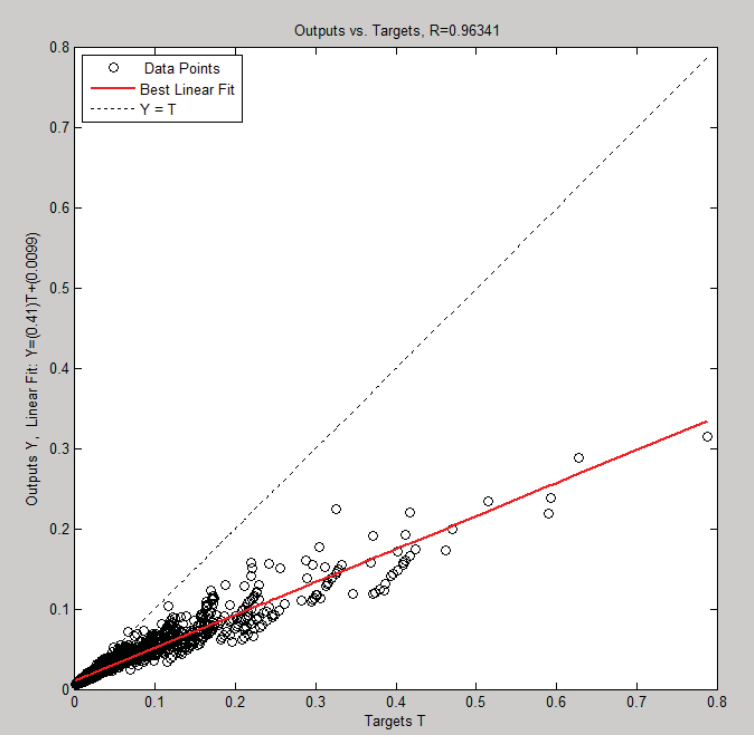

ILUSTRACIÓN 45: GRÁFICA COEFICIENTE DE CORRELACIÓN TRAINING

En la etapa de Testing los resultados fueron los siguientes:

Error Cuadrático Medio (MSE)	0.0046
Raíz Error Cuadrático Medio (RMSE)	0.0681
Coeficiente de Correlación (R)	0.9489
Coeficiente de Determinación (R^2)	0.9004
Error Porcentual Absoluto Medio (MAPE)	30,6169

TABLA 15: DATOS TESTING

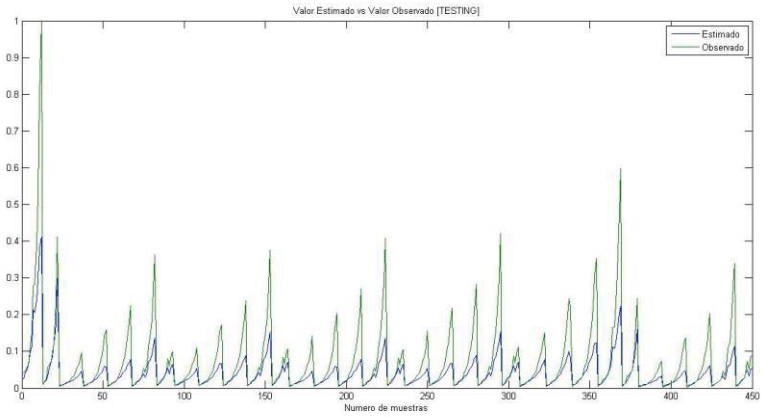

ILUSTRACIÓN 46: GRÁFICA TESTING

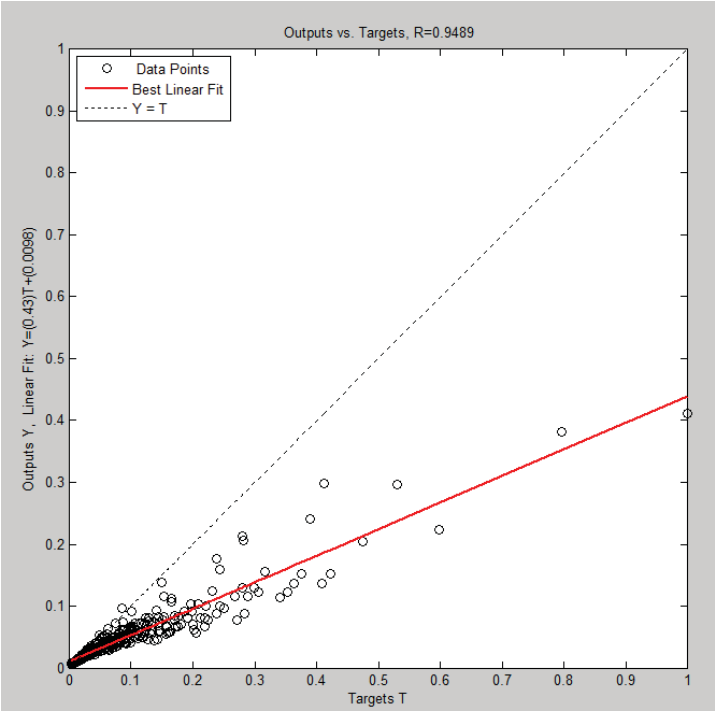

ILUSTRACIÓN 47: GRÁFICA COEFICIENTE DE CORRELACIÓN TESTING

CAPÍTULO 6

CONCLUSIONES

El presente trabajo tuvo por objetivo sentar las bases del conocimiento mínimo requerido para poder entender en que se enfocara el problema y para poder entender los resultados obtenidos.

Entender de donde provienen las Redes Neuronales, su real potencia en el desarrollo de optimización, control, clasificación y también en la predicción, que es el tema que nos convoca. Al utilizar las Redes Neuronales Artificiales con un algoritmo de aprendizaje supervisado, como lo es el Algoritmo de Optimización de Enjambre de Partículas, y una función de transferencia, con sus parámetros bien definidos y correctamente calibrados, se obtendrá una buena estimación de los costos de producción de intercambiadores de calor de carcasas y tubos. Pero es aquí donde también es visible el principal problema de las Redes Neuronales, su gran complejidad es no poseer un modelo definido que permita directamente la elección de los parámetros, y que estos deban ser elegidos mediante prueba y error.

Entender también que el problema de estimación de costos es una problemática que se presenta en todo tipo de empresa que se dedique a la producción de objetos tangibles (tubos, sillas, etcétera) como también intangibles (software).

El algoritmo de enjambre de partículas presenta un funcionamiento bastante definido y robusto, por lo que resulta ser una buena alternativa para incorporarse en la creación de un modelo en el cual los parámetros resultan ser complicados de hallar. Esta técnica busca dentro del espacio de la solución, la más óptima, es decir, la mejor solución. Obviamente como todo problema de optimización, la existencia de varias soluciones al problema hacen que la tarea de encontrar el óptimo global sea más difícil, porque existe la posibilidad de que el algoritmo converja hacia un óptimo local y no pueda salir de ese subespacio de soluciones. Es por esto que el enjambre de partículas presenta algunas variaciones en su funcionamiento con el afán de recorrer todo el espacio de solución en busca del óptimo global sin caer en los óptimos locales.

Basándose en los conceptos adquiridos en esta etapa del proyecto, se puede vislumbrar la viabilidad de construir un modelo para la estimación de costos de tuberías bajo estas herramientas.

Luego de haber obtenido los resultados por cada uno de los modelos propuestos, los cuales son PSO-IW, QSPO y UPSO se procedió a realizar la elección de cuál de estos es mejor. Dicha comparación se realizó con la medida del error porcentual medio (MAPE) obtenido en cada uno de los modelos en sus respectivas etapas de Testing, y se detallan a continuación: el modelo PSO-IW obtuvo de MAPE 12,9274; el modelo QPSO obtuvo de MAPE 14,9247 y el modelo UPSO obtuvo 30,6169.

Aquí se puede apreciar inmediatamente que el modelo inicial propuesto obtuvo los mejores resultados en base al MAPE en la estimación de costos de tuberías, aunque el segundo modelo no presenta un resultado muy alejado. Para poder apreciar de mejor manera dicha diferencia se realizó una estimación para ver cuánto gana porcentualmente los modelos 2 y 3 en relación al mejor modelo, el PSO-IW. El MAPE obtenido en el modelo PSO-IW tiene una ganancia de *13,40%* en comparación al QPSO y de un *57,7%* en

comparación del modelo UPSO, mostrando claramente la diferencia entre el mejor modelo y los otros 2.

A continuación se mostraran los resultados obtenidos para los coeficientes de determinación: el modelo PSO-IW obtuvo 0,9611 en su coeficiente de correlación; el modelo QPSO obtuvo 0,9690 en su coeficiente de correlación y el modelo UPSO obtuvo 0,9004 en su coeficiente de correlación.

En esta valores se refleja que el mejor resultado se obtiene en el segundo modelo, pero la diferencia es mínima en comparación con el modelo 1.Ahora para apreciar mejor dicha diferencia, aplicaremos la misma estimación realizada con el MAPE para comparar la ganancia del mejor modelo con los otros 2. Aquí el mejor modelo es el QPSO y obtiene una ganancia de *0,82%* en relación al PSO-IW y un *7,61%* en relación al UPSO. La diferencia con el modelo PSO-IW es mínima comparada con la diferencia que se obtiene con el modelo UPSO. Como se decidió que el modelo PSO-IW fue el mejor, veremos la ganancia que tiene frente a los otros 2 modelos. El modelo PSO-IW pierde un *0,82%* en comparación con el modelo QPSO y gana un *6,74%* en relación al modelo UPSO. Dada las ganancias del modelo PSO-IW se escogió como el mejor modelo de este proyecto.

Se puede apreciar que tanto el modelo de red neuronal wavelet radial + PSO-IW y el modelo con QPSO son buenos modelos de estimación, mostrando una característica de los datos, la fuerte relación que hay entre los datos de entrada, ya que se obtuvieron buenos resultados con pocas iteraciones, partículas y nodos ocultos.

Se pudo cumplir con los puntos del plan de trabajo establecidos para el proyecto, lo que permitió cumplir los objetivos específicos establecidos en un comienzo. Tomando en cuenta el trabajo hecho, se propusieron modelos que permitieron estimar los costos de producción de tuberías, pudiendo concluir con el trabajo propuesto.

REFERENCIAS

[Araya, 2007] Araya, A. (2007), Modelo De Estimación De Costos Para La Fabricación De Engranajes. PUCV, Chile.

[Broersen, 2002] Broersen, Piet M. T.(2002) Cost of order selection in time series analisys.

[Clerc 1999] Clerc, M. (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceeding of 1999 Congress on Evolutionary Computacion, 1999.

[Clerc, 2002] Clerc, M. and Kennedy, J. (2002). The particle swarm – explosion, stability and convergence in a multidimensional complex space. *IEEE Trans. Evolutionary Computation*.

[Cohen et al., 1993] Cohen, M., Franco, H., Morgan, N., Rumelhart, D., and Abrash, V. (1993). *Advances in Neural Information Processing Systems*. Morgan Kaufmann.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superposition of signmoidal function. *Mathematical of Control, Signals and Systems.*

[Elman, 1990] Elman, J. (1990). Finding structure in time. *Cognitive Science*.

[Finnie, 1996] Finnie, Gavin R. (1996). AI tools for software development effort estimation.

[Gardner, 1975] Gardner, M. (1975). Mathematical games: on the remarkable császár polyhedron and its applications in problem solving. En Sci. Amer.

[Grossberg, 1964] Grossberg, S. (1964). *The theory of embedding fields with applications to psychology and neuropsychology*. Rockefeller Institute of Medical Research, New York.

[Guyon, 1991] Guyon, I. (1991). Applications of neural networks to character recognition. *International Journal of Pattern Recognition and Artificial Intelligence*.

[Haykin, 1999] Haykin, S. (1999). Neural Networks: A Comprehensive Foundation.

[Hebb, 1949] Heeb, D. (1949). *Organitzation of Behavior*. John Wiley & Sons, New York.

[Hinton et al., 1984] Hinton, G., Ackley, D., and Sejnowski, T. (1984). Boltzmann machines: Contraint satisfaction networks that learn. Technical Report CMU-CS-84-119, Carnegie-Mellon University, Department of Computer Science Technical Report.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., adn White, H. (1962). Multilayer feedforward networks are universal approximators. *Neural Networks*.

[Isasi y Galvan, 2004] Isasi, V., Galvan, L., Redes de neuronas artificiales, un enfoque practico. Pearson Prentice Hall.

[Jordan, 1986] Jordan, M. (1986). Serial Order: a parallel distributed processing approach. Technical Report, Institute for Cognitive Science. University of California.

[Kennedy, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*.

[Kennedy, 1999] Kennedy, J. (1999). Small world and mega-minds: E_ects of neighborhood topology on particle swarm performance. Proceedings of the 1999 Congress on Evolutionary Computation.

[Kennedy, 2001] Kennedy, J. y Eberhart, R. (2001). *Swarm Intelligenge*. Morga Kaufmann Publisher.

[Kennedy, 2003] Kennedy, J. and Mendes, R. (2003). Neighborhood topologies In fully-informed and best-of-neighborhood particle swarm. *Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications.*

[Kienker et al., 1986] Kienker, P., Sejnowski, T., Hinton, G, and Schumacher, L. (1986). Separating figure from ground with a parallel network.

[McCulloch and Pitts, 1943] McCulloch. J. and Pitts, W. (1943). A logical calculus of the ideas immanen in nervous activity. *Bulletin of Mathematical Biophysics*

[Minsky y Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons: an introduccion to computational geometry.* The MIT Press.

[Narendra, 1990] Narendra, K. and Parthasarathy, K. (1990). Identification and control of dynamical systems. *IEEE Transactions on Neural Networks*.

[Parsopoulos y Vrahatis, 2004] Parsopoulos, K. E., & Vrahatis, M. N. (2004). UPSO: a unified particle swarm optimization scheme. In T. Simos & G. Maroulis (Eds.), Lecture Series on Computer and Computational Sciences. Zeist, The Netherlands: VSP International Science Publishers.

[Parsopoulos y Vrahatis, 2007] Parsopoulos, K. E., & Vrahatis, M. N. (2007). Parameter selection and adaptation in unified particle swarm optimization. Mathematical and Computer Modelling.

[Peer, 2003] Peer, E. S., v. d. B. F. and Engelbrecht, A. P. (2003). Using neighborhoods with the guaranteed convergence pso. *Proceedings of the IEEE Swarm Intelligence Symposium*

[Pomerleau, 1992] Pomerleau, D. (1992). *Neural networks perception for mobile robot guidance*. PhD thesis, Computer Science, Carnegie Mellon University.

[Rosenblatt, 1957] Rosenblatt, F. (1957). The perceptron: A perceiving and recognicing automation. Technical Report 85-460-1, Cornell Aeronautical Laboraory.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A theory of statistical separability in cognitive systems. Technical Report VG-1196-G-1, Cornell Aeronautical Laboratory.

[Ru, 2009] Ru Jiang (2009). Application of BP Neural Network Approach for Cost Estimation of Wastewater Treatment Plants: A Case Study of Taiwan Region.

[Rumelhart et al., 1986b] Rumelhart, D., Hinton, G., and Willians, R. (1986b*). Parallel Distributed Processing, chapter Learning repreentations by back-propagation errors*. MIT Press.

[Sackinger et al., 1992] Sackinger, E., Boser, B., Bromley, J., LeCun, L., and Jackel, L. (1992). Application of anna neural network chip to high-speed character recognition. *IEEE Transactions on Neural Networks*.

[Sejnowski, 1986] Sejnowski, T. (1986). Higher-order boltzmann machines. In Denker, J., editor, *AIP Conference Proceedings*, Neural Network for computing New York. American Institute of Physics.

[Sejnowski, 1986] Sejnowski, T. and Hinton, G. (1986). *Separating figure from ground with a Boltzmann machine*. Vision, Brain and Copperative Computation. MIT Press-Bradford Books, Cambridge

[Shi, 1998] Shi., Y. and Eberhart, R. (1998). Empirical study of particle swarm optimization. *Proceedings of 1999 Congress on Evolutionary Computation, 1999.*

[Soria y Blanco, 2001] SORIA, E. Y BLANCO, A. (2001): Redes neuronales artificiales. ACTA (Autores científico-técnicos y académicos).

[Werbos, 1989] Werbos, P. (1989). Backpropagation and neurocontrol: a review and prospectus. In *International Joint Conference on Neural Networks*.

[Widrow, 1959] Widrow, B. (1959). Adaptative samplet-data systems- a statistical theory of adaptation.

[Widrow, 1960] Widrow, B. (1959). An adaptative "adaline" neuron using chemical "memistors". Technical Report 1553-2, Stanford Electronics Laboratory.

[Zheng, 2003] Zheng, Y., M. L. Z. L. and Qian, J. (2003). Empirical study of particle swarm optimizer with an increasing inertia weight. *Proceedings of The 2003 Congress on Evolutionary Computation, 2003.*