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

ESTIMACIÓN DE COSTOS PARA LA FABRICACIÓN DE TUBERÍAS, UTILIZANDO REDES NEURONALES POLINÓMIALES CON ALGORITMOS GENÉTICOS

RAFAEL ANTONIO PAVEZ FERNÁNDEZ

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

Septiembre, 2014

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

ESTIMACIÓN DE COSTOS PARA LA FABRICACIÓN DE TUBERÍAS, UTILIZANDO REDES NEURONALES POLINOMIALES CON ALGORITMOS GENÉTICOS

RAFAEL ANTONIO PAVEZ FERNÁNDEZ

Profesor Guía**: Nibaldo Rodríguez Agurto**

Profesor Co-referente**: Wenceslao Palma Muñoz**

Carrera**: Ingeniería Civil Informática**

Septiembre, 2014

Dedicatoria.

Dedicado a Nieves, mi madre, por su constante cariño, sacrificio y apoyo durante esta gran estapa de mi vida.

Agradecimientos.

Agradezco a todas las personas que siempre confiaron y creyeron en mí. En especial a mi compañera de vida, Evelyn. A mis amigos y a mi profesor guía por su apoyo.

Resumen

La necesidad de conocer con anticipación los costos de producción, obliga a las organizaciones a contar con métodos y/o técnicas que permitan estimar dichos costos. Ésta investigación está enfocada en pronosticar el costo de elaborar tuberías de acero, para ello se presentan tres modelos de redes neuronales multiplicativas. La topología de la red es determinada a mediante la utilización de variantes del Algoritmo Genético. El mejor resultado se obtuvo con la variante denominada "Genérico", con una población de 35 individuos, el cual arrojó un Error Porcentual Absoluto Medio del 8,81 % y un 98,93 % de la varianza explicada.

Palabras Clave: *Redes Neuronales Polinomiales, Algoritmos Genéticos, Estimación de Costos, Fabricación de Tuberías.*

Abstract

The need to know in advance the cost of production, demands on organizations having methods and/or techniques to estimate it. This research is focused on forecasting the cost of producing pipes, in order to do that, three models of multiplicative neural networks are exposed. The network topology is determined using differents genetics algorithms. The best result was obtained with the variant called "Generic", with a population of 35 individuals, which produced a Mean Absolute Percentage Error 8.81 % and 98.93 % of the explained variance.

Keywords: *Polinomial Neural Networks, Genetic Algorithms, Cost Estimation, Pipe Manufacture.*

Índice General

Índice de Figuras

Índice de Tablas

Abreviaciones y Nomenclatura

GA	Algoritmo Genético – del inglés Genetic Algorithms –
PD	Definición Parcial – del inglés Partial Definition –
CER	Relaciones de Estimación de Costos – del inglés Cost Estimation Relationship –
ECM	Error Cuadrático Medio – del inglés Mean Squaredc Error –
GMDH	Método de Grupo para Manipulación de Datos – del inglés Group Method of Data Handling
MAPE	Porcentaje de Error Medio Absoluto – del inglés Mean Absolute Percentage Error –
MLP	Perceptrón Multi Capa – del inglés Multi Layer Perceptron–
PNN	Red Neuronal Polinomial – del inglés Polynomial Neural Network –
RMSE	Raiz del Error Cuadrático Medio – del inglés Root Mean Square Error–
RNA	Red Neuronal Artificial – del inglés Artificial Neural Network –

Capítulo 1

Introducción

En este capítulo se presenta la problemática que motivo la realización de este estudio. Además, como referencia, se establece los objetivos que se buscan alcanzar con ésta investigación.

1.1 Planteamiento del Problema

El costo es uno de los principales criterios para la toma de decisiones en las primeras etapas de un proceso de diseño de cualquier producto. En el mundo actual, el control de costos juega un papel importante para ser competitivo y poder mantener altos niveles de calidad.

El costo de un producto se puede definir como los gastos necesarios para mantener un proyecto, línea de procesamiento o un equipo en funcionamiento [1]. En una compañía estándar, la diferencia entre el ingreso (por ventas y otras entradas) y el costo de producción indica el beneficio bruto. El costo de producción tiene dos características opuestas, que algunas veces no están bien comprendidas. La primera es que para producir bienes se debe gastar; esto significa generar un costo. La segunda característica es que los costos deberían ser mantenidos tan bajos como sea posible y eliminar los innecesarios.

Por lo anterior, es que todo tipo de empresas manufactureras necesitan conocer cuales son sus costos de producción con anterioridad a la elaboración de sus productos. Debido a ésto es necesario pronosticar los costos de producción. La estimación de costos a menudo se utiliza para describir el proceso mediante el cual se pronostican las consecuencias presentes y futuras de los diseños de ingeniería [2].

La estimación de costos es un factor clave durante las fases de desarrollo de productos manufacturados, sobre todo en la fase de diseño, que es la etapa donde se deciden y seleccionan las características morfológicas principales de un producto, los tipos de materiales a utilizar y el proceso de producción. Estudios previos han demostrado que el mayor

potencial para reducir costos se presenta en las primeras fases de diseño, donde se decide alrededor de un 80 % del costo de un producto [3].

Dado que la fase de diseño, generalmente, representa un porcentaje relativamente pequeño del costo total de un desarrollo, es necesario dedicar un esfuerzo mayor para optimizar los costos del producto. A medida que el proceso de desarrollo avanza, el costo de modificar las características de un producto es más costoso que hacerlo en etapas tempranas del ciclo de desarrollo. Por ésto, es que la estimación de costos se hace vital para las empresas manufactureras, porque pueden tener una aproximación del costo final de un producto. Como se indica en [4], el conocimiento del costo de un producto permite, a las empresas, tomar medidas para diferenciarse de la competencia, elegir el mercado objetivo, ofrecer productos customizados y de alta calidad.

Según [5], existen tres enfoques principales para la estimación de costos, los cuales se resumen a continuación. Un primer enfoque es utilizar técnicas basadas en analogía, dichas técnicas se basan en la definición y análisis del grado de similitud entre un producto nuevo y otro que ha sido producido anteriormente. El segundo enfoque es el método paramétrico, donde el costo se expresa como una función analítica de un conjunto de variables que representan algunas de las características del producto y se supone que influyen con mayor proporción en el costo final de éste último. Estas funciones se denominan relaciones de estimación de costos (CER) y se construyen utilizando los métodos estadísticos. El último enfoque es la utilización de modelos analíticos, donde la estimación se basa en el análisis detallado de la proceso de fabricación y de las características del producto. El costo estimado del producto se calcula de una manera muy analítica, como la suma de sus componentes elementales, constituida por el valor de los recursos utilizados en cada etapa del proceso productivo (materias primas, componentes, mano de obra, equipo, etc). Debido a esto, el último enfoque puede ser utilizado solamente cuando todas las características del proceso de producción y del producto están bien definidas. Sin embargo, todos los enfoques mencionados son formas válidas para estimar costos de producción [6].

Una de las dificultades iniciales de la estimación de costos radica en que no existen procesos anteriores iguales que se puedan utilizar como base para una estimación de costos en forma directa, puesto que los proyectos son relativamente únicos. Sin embargo, esta problemática dio un puntapie inicial para la búsqueda, desarrollo y/o adaptación de técnicas o métodos que permitan lograr una estimación de costos precisa. De todas formas, dichas técnicas se basan en diferentes experiencias de procesos productivos previos, es decir, encuentran similitudes y relaciones (directas, indirectas o implícitas) existentes en la información previa, para encontrar nuevas soluciones.

El presente trabajo está centrado en la estimación de costos de producción de elementos de tuberías. Una tubería es una sección cilíndrica hueca, que es usada principalmente para el flujo de sustancias como líquidos, fluidos, masas y polvos químicos. Cada pieza, por separado, es llamada “unión” y los materiales más usados para su elaboración son el acero de carbón, aleaciones de acero y acero inoxidable [3].

En la presente investigación se desarrolla y evalúa un modelo de predicción utilizando

Redes Neuronales Polinomiales con Algoritmos Genéticos (GA), lo que permitirá estimar los costos de producir tuberías. Con ésto se pretende dar inicio a la búsqueda de alternativas para reducir estos costos, lo cual puede ser, por ejemplo, cambiando las condiciones estructurales y físicas de los productos.

Las Redes Neuronales Artificiales (RNA) se pueden definir como "un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso: La Neurona" [7]. Tanto su estructura como su estado interno proporcionan características fundamentales para su uso en el ámbito de la predicción: capacidad de aprender, adaptarse y generalizar. Además, se pueden diferenciar distintos tipos de redes en base a su topología y estado interno.

1.2 Objetivos

A continuación se dan a conocer los objetivos planteados para este proyecto. Por un lado se indica el objetivo general del proyecto, seguido de los objetivos específicos que servirán de apoyo a la obtención del objetivo general.

1.2.1 Objetivo General

Desarrollar un modelo de predicción utilizando redes neuronales polinomiales con algoritmos genéticos para estimar los costos de producción de elementos de tuberías.

1.2.2 Objetivos Específicos

- Explicar la estructura de las Redes Neuronales Polinomiales y Algoritmos Genéticos.
- Diseñar y estimar los parámetros del modelo de predicción de costos usando algoritmos genéticos.
- Evaluar el rendimiento del modelo neuroevolutivo utilizando métricas residuales.

1.3 Estructura del Documento

El Capítulo 2 hace referencia a los fundamentos de las *Redes Neuronales artificiales*, describiendo sus principales componentes y derivaciones, así como también se presentará su analogía de las neuronas biológicas. El Capítulo 3 aborda todos los conceptos de los *Algoritmos Genéticos*, definiendo los principales componentes, aplicaciones y su algoritmo. El Capítulo 4 describe y muestra el modelo genérico a desarrollar, variantes a este modelo, métricas a utilizar y se determinan los parámetros de los modelos propuestos. En el

Capítulo 5 se valida los modelos, presentados en el capítulo anterior, utilizando los datos destinados a este propósito y se analizan los resultados obtenidos. Finalmente, el Capítulo 6 consiste en las conclusiones del proyecto.

Capítulo 2

Redes Neuronales Artificiales

2.1 Introducción

En [8] se indica que el funcionamiento exacto del cerebro humano sigue siendo un misterio. Sin embargo, algunos aspectos de éste increíble procesador se conocen. En particular, el elemento básico del cerebro humano es un tipo específico de célula que, a diferencia del resto del cuerpo, no parecen regenerarse. Debido a que este tipo de células es la única parte del cuerpo que se sustituye lentamente o, simplemente, no lo hace. Se supone que estas células son las que nos proporcionan nuestra capacidad de recordar, pensar y aplicar experiencias anteriores para cada una de nuestras acciones.

Una Red Neuronal Artificial es una técnica o modelo computacional que trata de emular ciertas características propias de los humanos, como la capacidad de memorizar y de asociar hechos. Para lograr ésta analogía con el cerebro, es que se establecen propiedades particulares como la capacidad de aprender y adaptarse, generalizar, agrupar u organizar datos y cuya operación se basa en procesamiento paralelo. Las Redes Neuronales Artificiales poseen distintas características, tanto de su estructura como de la forma en la que aprenden a realizar la función por la cual fueron creadas.

En el presente capítulo, la sección 2 presenta una reseña histórica de las redes neuronales, desde sus comienzos, a través de los avances en esta materia y sus complicaciones. Luego en la sección 3 se presenta a la Neurona Biológica con todas sus características, para luego describir el concepto Red Neuronal Artificial, la cual es una analogía a nivel computacional de una red neuronal biológica. En la sección 5 se dan a conocer las distintas Topologías de Redes Neuronales, para luego, en la sección 6, continuar con la explicación acerca de la forma de aprendizaje de las redes neuronales artificiales. En la sección 7 se describen los tipos de redes utilizados en este caso de estudio: Redes Polinomiales. Finalmente en la sección 11 de este capítulo se dan a conocer distintas aplicaciones de las redes neuronales.

2.2 Reseña Histórica

La historia de las redes neuronales se remonta hacia el año 1936 en donde Alan Turing fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, en el año 1943 el neurofisiólogo Warren McCulloch y el matemático Walter Pitts concibieron una teoría que describe la forma de trabajar de las neuronas, la cual intenta modelar el comportamiento de una neurona "natural", similares a las que constituyen el cerebro humano. Dicha neurona es la unidad esencial con la cual se construye una red neuronal artificial, además modelaron una red neuronal simple mediante circuitos eléctricos. Una de la consideraciones de ésta teoría es que la actividad neuronal es un proceso "todo o nada", es decir, la neurona McCulloch- Pitts es un dispositivo binario, por lo tanto, sólo puede estar en uno de dos posibles estados. Cada neurona puede recibir entradas de sinapsis excitadoras o inhibidoras y estas entradas determinan el estado en que se encuentra la neurona. En la figura 2.1 se muestra una representación de la neurona concebida.

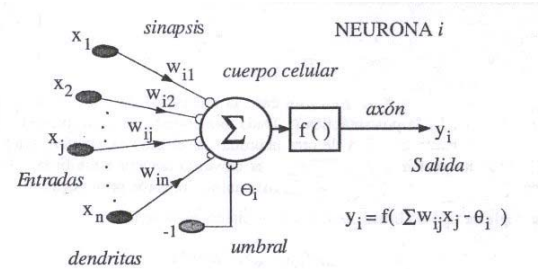


Figura 2.1: Neurona McCulloch-Pitts

En 1949 Donald Hebb, fue el primero en explicar los procesos del aprendizaje desde un punto de vista psicológico, desarrollando una regla conocida como *Regla de Hebb*, la cual sienta las bases de la teoría de las Redes Neuronales. Ésta regla explica el proceso de comunicación entre dos neuronas, que cuando están lo suficientemente cerca se provocaría la activación de ciertas partes de una neurona dando lugar al aprendizaje.

En 1950 Karl Lashley, en sus series de ensayos realizados, encontró que la información no era almacenada en forma centralizada en el cerebro, sino que era distribuida encima de él.

Haibt y Duda en 1956 realizan una de las primeras simulaciones computacionales para probar una bien formulada teoría neuronal basándose en el postulado de aprendizaje de Hebb. Además agregaron suposiciones que no existían en un principio, por ejemplo, se acotó el valor de la sinapsis.

En 1957 Frank Rosenblatt. Comenzó el desarrollo del Perceptrón, la cual es la red neuronal más antigua que aún es utilizada hoy en día para el reconocimiento de patrones. Éste modelo era capaz de generalizar, es decir, después de haber aprendido una serie de patrones podía reconocer otros similares, aunque no se le hubiesen presentado anterior-

mente.

Quince años luego de la publicación de McCulloch y Pitts un nuevo acercamiento al problema de reconocimiento de patrones fue introducido por Rosenblatt (1958) en su trabajo sobre el perceptrón. Éste constaba de 2 niveles y los pesos se ajustaban en proporción al error entre las salidas deseadas y obtenidas.

En 1960 Widrow y Hoff introdujeron el algoritmo *least mean-square* y lo usaron para formular el ADALINE (Adaptative Linear Element). Éste último, se diferencia del perceptrón en el proceso de entrenamiento y fue la primera red neuronal aplicada a un problema real.

En 1969 Minsky y Papera probaron matemáticamente que el Perceptrón no era capaz de resolver problemas tan fáciles como el aprendizaje de una función no lineal, lo cual produjo una abrupta caída en las investigaciones sobre redes neuronales. Esta prueba demostró una gran debilidad, dado que las funciones no-lineales son ampliamente utilizadas en computación y en problemas del mundo real.

En 1982 Hopfield publica un trabajo clave para el resurgimiento de las redes neuronales. En él se desarrolla la idea de usar una función de energía para comprender la dinámica de una red neuronal recurrente con uniones sinápticas simétricas. El principal uso de estas redes ha sido como memorias y como instrumento para resolver problemas de optimización como el problema del vendedor viajero.

En 1986 Rumelhart y Hinton redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation). A partir de 1986, el panorama fue alentador con respecto a las investigaciones y el desarrollo de las redes neuronales. En la actualidad, son numerosos los trabajos que se realizan y publican cada año, las aplicaciones nuevas que surgen (sobretodo en el área de control) y las empresas que lanzan al mercado productos nuevos, tanto hardware como software (sobre todo para simulación).

2.3 Red Neuronal Biológica

El cerebro consta de una gran red de elementos interconectados llamados neuronas. Éste es el elemento estructural y funcional más esencial en el sistema de comunicación neuronal. La mayoría de ellas secretran sustancias químicas (neurotransmisores) para la transmisión de la información, enviándola a través de distintas neuronas para la formación de redes, las cuales elaboran y permiten almacenar la información. Las neuronas tienen tres componentes principales, las dendritas, el cuerpo de la célula o soma, y el axón. Las dendritas, son el árbol receptor de la red, son como fibras nerviosas que cargan de señales eléctricas el cuerpo de la célula. El cuerpo de la célula, realiza la suma de esas señales de entrada. El axón es una fibra larga que lleva la señal desde el cuerpo de la célula hacia otras neuronas. El punto de contacto entre el axón de una célula y una dendrita de otra célula es llamado sinapsis, la longitud de la sinapsis es determinada por la complejidad del proceso químico que estabiliza la función de la red neuronal [9]. En la figura 2.2 se

muestra una representación de una neurona biológica.

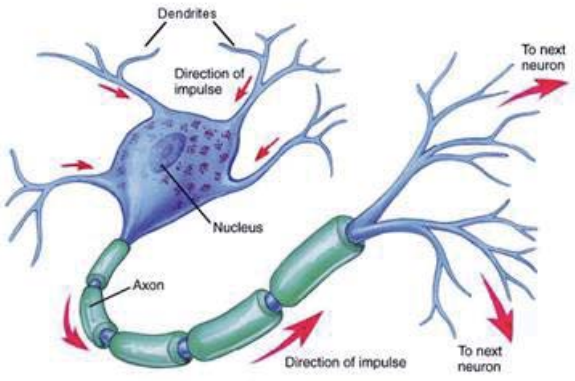


Figura 2.2: Neurona Biológica

Un neurotransmisor es una sustancia química que, cuando es liberada de las terminales de los axones, produce un cambio momentáneo del potencial eléctrico, provocando la estimulación de otra fibra nerviosa. Los neurotransmisores estimulan o inhiben neuronas adyacentes u órganos efectores, como músculos y glándulas.

El tipo más frecuente de sinapsis es el que se establece entre el axón de una neurona y la dendrita de otra (sinapsis axodendrítica). A medida que el axón se acerca, puede tener una expansión terminal (botón terminal) o puede presentar una serie de expansiones (botones de pasaje) cada uno de los cuales hace contacto sináptico. Otro tipo de sinapsis es el que se establece entre el axón de una neurona y el cuerpo celular de otra neurona (sinapsis axosomática).

En la figura 2.3 se muestra el proceso sináptico que ocurre entre dos neuronas. Simplificando el proceso, el mecanismo es el siguiente [10]:

- La sinapsis recoge información electro-química de las neuronas vecinas por medio de las dendritas.
- La información llega al núcleo, en donde es procesada para generar la respuesta que viajará por el Axón.
- Luego, la señal propagada por el axón se ramifica en los terminales axónicos y llega a las dendritas de otras neuronas en lo que se denomina sinapsis.

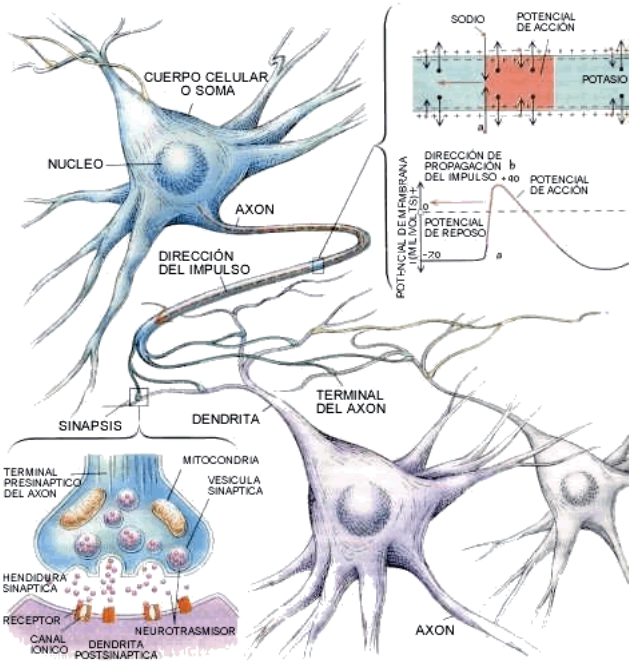


Figura 2.3: Comunicación entre neuronas

La capacidad de una red neuronal de aprender y de adaptarse a su entorno puede lograrse mediante dos mecanismos: la creación de nuevas conexiones sinápticas y la modificación de las conexiones sinápticas existentes [11]. Esto permite, a la red, modificar su estructura, además de otorgarle la cualidad de ser tolerante a fallos, dado que, se puede realizar más de un camino para que un impulso neuronal viaje a través de la red y provoque la respuesta adecuada. Además la comunicación entre neuronas se basa en el paso de mensajes pequeños, por lo que la información crítica no se transmite directamente, mas se captura y distribuye entre las interconexiones.

2.4 Red Neuronal Artificial (RNA).

Una red neuronal es un proceso distribuido, formado por unidades simples de procesamiento, que tiene una propensión natural para almacenar conocimiento experimental y hacerlo disponible para su uso [11]. Una RNA consiste en un sistema interconectado de neuronas artificiales, lo que forma una red colaborativa para producir un estímulo de salida.

Según [10] el modelo de una neurona presenta tres elementos básicos:

- Una serie de sinapsis, donde cada una está caracterizada por un peso el cual puede tomar valores negativos como positivos.

- Un sumador, el cual adiciona 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 [-1,1] según sea el caso. En la figura 2.4 se muestran las funciones de activación más utilizadas.

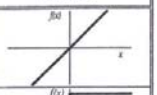
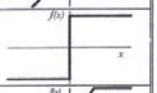
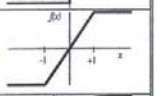
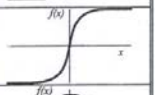
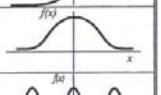
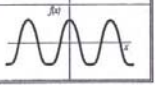
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Figura 2.4: Funciones de activacion usuales

Basado en la neurona McCulloch-Pitts y la regla de aprendizaje Hebbiano, [11] indica la estructura de una neurona básica, que incluye las señales de entradas, los pesos sinápticos, un valor numérico conocido como umbral, simbolizado por θ_k, una función de activación φ_k lo que lleva a la salida de la neurona. El umbral es un valor opcional que puede aumentar o disminuir el estado interno de la neurona, dependiendo si este es positivo o negativo respectivamente. En términos matemáticos, una neurona k puede ser descrita por medio de las ecuaciones 2.4.1 y 2.4.2.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.4.1}$$

Donde $x_1, x_2, ..., x_m$ corresponden a las señales de entrada a la neurona artificial, $w_{k1}, w_{k2}, ..., w_{km}$ a los pesos sinápticos de la neurona k, u_k es la salida obtenida de la combinación lineal, dada la señal de entrada.

$$y_k = \varphi(u_k + \theta_k) \tag{2.4.2}$$

Donde, θ_k corresponde al umbral, φ_k corresponde a la función de activación e y_k es la señal de salida de la neurona k.

Se puede ver una red neuronal, desde un punto de vista matemático, como un grafo dirigido y ponderado donde cada nodo representa una neurona y las conexiones sinápticas son los arcos que unen los nodos. Por ser un grafo dirigido, los arcos son unidireccionales. Considerando esta forma de ver la red neuronal, se asume que la información se popraga en un único sentido, es decir, desde una neurona origen a una neurona destino. Además al ser ponderado, cada conexión tiene asignado un valor o grado de importancia con respecto al resto de conexiones, si la ponderación es positiva la conexión es excitadora, mientras que si es negativa se dice que es inhibidora.

Generalmente las neuronas se agrupan en capas, por lo que una RNA está formada por varias capas de neuronas. Cada capa tiene un nombre específico, de acuerdo a la naturaleza de su comportamiento. Las capas que conforman una red neuronal, en general, son tres. La primera es la capa de entrada, donde se recepcionan los datos de entrada, comúnmente vectorizados. A continuación se encuentran la capa oculta, que pueden ser uno o varios niveles de neuronas agrupadas, y es donde se realiza gran parte del procesamiento de los datos. Finalmente se encuentra la capa de salida, que corresponde a la respuesta entregada por la red neuronal y, adicionalmente, puede que realice parte del procesamiento. En la figura 2.5 se muestra de forma gráfica dicha organización.

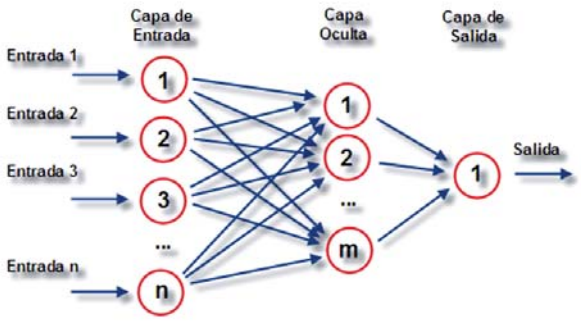


Figura 2.5: Arquitectura básica de una red neuronal

2.5 Topologías de Redes Neuronales.

La topología o arquitectura de una red neuronal consiste en la organización y disposición de las neuronas, formando capas o agrupaciones de neuronas. En este sentido, los parámetros fundamentales de la red son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas.

2.5.1 Según su número de capas

- **Red neuronal monocapa:** Son redes conformadas por una única capa, generalmente, la de salida. Las redes monocapas se utilizan generalmente en tareas relacionadas

con lo que se conoce como autoasociación. La autoasociación consiste en la regeneración de la información de entrada a la red, que se presenta de forma incompleta o distorsionada. Un ejemplo de este tipo de red se muestra en la figura 2.6

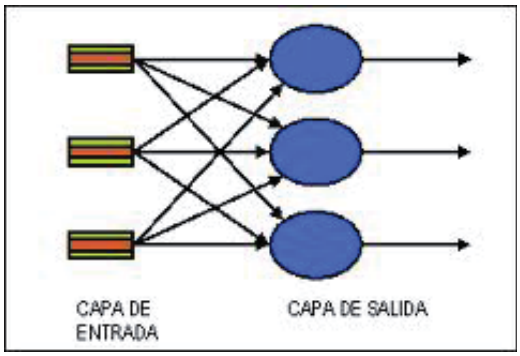


Figura 2.6: Red Neuronal Monocapa

- **Redes neuronales multicapa:** Las redes multicapa cuentan con al menos una capa de entrada y otra de salida, y, eventualmente una o varias capas intermedias. Dichas capas están jerarquizadas y normalmente todas las neuronas de una capa reciben señales de otra capa anterior y envían señales a la capa posterior. Por lo que, éste tipo de redes, es una generalización del modelo monocapa. Además dependiendo la forma en que se conectan sus capas, se pueden clasificar como parcial o totalmente conectadas, para mas información, véase 2.5.3.

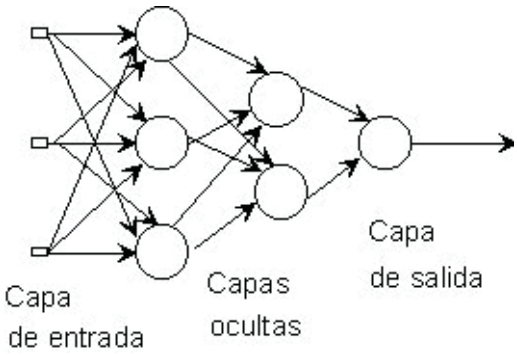


Figura 2.7: Red Neuronal Multicapa

2.5.2 Según el tipo de conexiones

- **Redes neuronales no recurrentes:** Este tipo de redes solo contienen conexiones hacia adelante y son conocidas como redes feedforward. El orden de las conexiones es de la capa de entrada, pasando por las capas ocultas hasta la capa de salida. Estas redes son las más sencillas de implementar y, además, son las más estudiadas, debido capacidad computacional de la época en que aparecieron.

- **Redes neuronales recurrentes:** Los tipos de conexiones de este tipo de redes pueden ser intercapas, es decir, se pueden conectar neuronas de una capa con las de otra capa (anterior y/o posterior), incluso con las del mismo nivel y al menos cuentan con un ciclo de retroalimentación. También son conocidas como redes feedback.

2.5.3 Según el grado de conexión

- **Redes neuronales totalmente conectadas**: En este caso todas las neuronas de una capa se encuentran conectadas con las de la capa siguiente (redes no recurrentes) o con las de la anterior (redes recurrentes).
- **Redes neuronales parcialmente conectadas**: En este caso no se da la conexión total entre neuronas de diferentes capas.

2.6 Aprendizaje de una RNA.

El aprendizaje de una red neuronal artificial se produce cuando sus parámetros libres, a través de un proceso de simulación, se adaptan al entorno en el cual está inserta, y, el aprendizaje, está determinado por la manera en que se produce el cambio en los parámetros [11]. El aprendizaje consiste en algoritmos basados en fórmulas matemáticas, que usando técnicas como minimización del error u optimización de alguna "función de energía", modifican el valor de los pesos sinápticos en función de las entradas disponibles y con ello, modifican la respuesta de la red para obtener las salidas deseadas.

El proceso usual de los algoritmos de aprendizaje es que la red ejecuta patrones iterativamente, cambiando los pesos de las sinapsis, hasta que convergen a un conjunto de pesos óptimos que representan a los patrones, entonces se mostrará una respuesta satisfactoria para esos patrones. Sus pesos sinápticos se ajustan para dar respuestas correctas al conjunto de patrones de entrenamiento. Dentro del aprendizaje se destacan tres tipos, aprendizaje supervisado, aprendizaje no supervisado y aprendizaje híbrido.

2.6.1 Aprendizaje Supervisado.

En el aprendizaje supervisado, los patrones de entrenamiento se dan en forma de pares de entrada y maestro (supervisor). El supervisor determina cuales debieran ser las respuestas entregadas por la red dada una entrada determinada. Este agente externo controla la salida y en caso de no corresponder a la deseada, se procede a modificar los pesos de las conexiones, con el fin de que la salida obtenida se aproxime lo más posible a la deseada, véase figura 2.8.

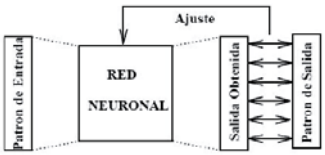


Figura 2.8: Aprendizaje Supervisado

Se consideran tres tipos de aprendizaje supervisado.

- **Aprendizaje por corrección de error**. Consiste en ajustar los pesos de la red en función de las diferencias, entre los valores deseados y los obtenidos.
- **Aprendizaje por refuerzo**. Se utiliza un mecanismo basado en probabilidades para el ajuste de los pesos, de acuerdo a si el resultado obtenido de la red se ajusta a la deseada o no. El supervisor toma un rol más crítico al indicar una señal de refuerzo si la salida es la deseada.
- **Aprendizaje estocástico**. Este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

2.6.2 Aprendizaje No Supervisado.

Las redes con aprendizaje no supervisado, conocido también como autosupervisado, no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información externa que indique si la salida generada en respuesta a una determinada entrada es o no correcta; por ello, suele decirse que estas redes son capaces de autoorganizarse. Estas redes deben encontrar características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presentan en su entrada. Existen varias posibilidades en cuanto a la interpretación de la salida de estas redes, que depende de su estructura y algoritmo de aprendizaje utilizado, véase figura 2.9.

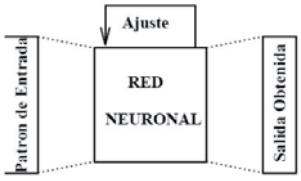


Figura 2.9: Aprendizaje No Supervisado

- **Aprendizaje Hebbiano.** Sugerida por Donald Hebb en 1949, a partir de estudios con neuronas reales. Este tipo de aprendizaje propone que los pesos sinápticos se incrementan, si las neurona de origen y destino están activadas, es decir, refuerza

los “caminos” usados frecuentemente en la red, lo que explicaría, por ejemplo, los hábitos y el aprendizaje por repetición.

- **Aprendizaje Competitivo y Comparativo**. Simplificando la definición, el aprendizaje competitivo usa inhibición lateral para activar una sola neurona, es decir, el ganador se lleva todo. La neurona seleccionada es aquella cuyos pesos se ajustan más al patrón de entrada. Algunas redes neuronales que emplean el aprendizaje competitivo son los mapas autoorganizativos.

2.6.3 Aprendizaje Híbrido

El aprendizaje híbrido corresponde a una mezcla de los tipos de aprendizaje: el aprendizaje supervisado y el aprendizaje no supervisado, descritos en las secciones 2.6.1 y 2.6.2

2.7 Red Neuronal Polinomial (PNN)

Las redes neuronales polinomiales (PNN) son del tipo feed-forward. La arquitectura de una red neuronal polinomial está basada en GMDH (Método de Grupo para Manipulación de Datos) [12].

La idea principal de GMDH es utilizar las redes feed-forward con funciones de transferencia polinómicas, cuyos coeficientes se obtienen utilizando una técnica de regresión. Ésta clasificación fue desarrollada por Ivakhneko a fines de los sesentas para identificar conexiones no lineales entre variables de entrada y salida, inspirada en la forma polinomial de Kolmogorov-Gabor [13]. Este tipo de redes presentan un alto nivel de flexibilidad, ya que cada nodo puede tener una cantidad diferente de entradas, por lo que se forman polinomios de diferente orden (lineales, cuadráticos, cúbicos, etc). La complejidad de este tipo de redes radica en el orden polinomial elegido para la solución.

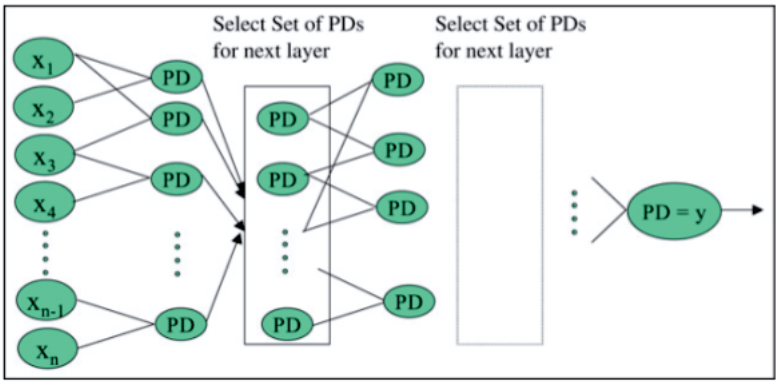


Figura 2.10: Red Neuronal Polinomial

En la figura 2.10 se puede apreciar la sigla PD (Definición Parcial), que representa el orden del polinomio creado para cada nodo perteneciente a las capas ocultas. Un ejemplo se puede apreciar en la siguiente figura:

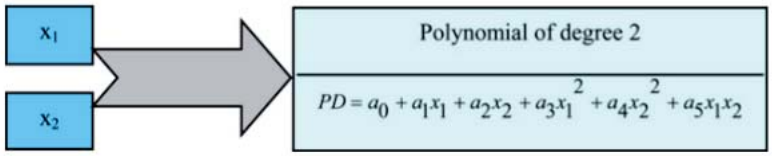


Figura 2.11: Definición Parcial

2.7.1 Red Neuronal Sigma-Pi

Son redes multicapa feed-forward formadas por una única capa oculta donde cada una de las neuronas presenta un carácter local, es decir, activa en una región diferente del espacio de patrones de entrada. Las neuronas de la capa de salida realizan una combinación lineal de las activaciones de las capas ocultas.

Estas redes nacen tras la necesidad de construir una red de neuronas que requiera un menor tiempo de aprendizaje que el Perceptrón Multicapa, para ser utilizadas por las aplicaciones en tiempo real. Esto se consiguió mediante las activaciones locales, donde pocas neuronas ocultas tendrían que ser procesadas para nuevos patrones de entrada.

Una Red Neuronal Sigma-Pi se encuentra formada por 3 capas [14]:

- Capa de Entrada: reciben las señales desde el exterior y las transmiten a la siguiente capa sin realizar ningún procesamiento sobre estas señales, además no presentan ningún peso asociado.
- Capa Oculta: en esta capa se realiza una productoria, para cada nodo, donde las entradas están elevadas a los pesos.
- Capa de Salida: actúa como salida de la red realizando, una sumatoria de las salidas de las capa oculta.

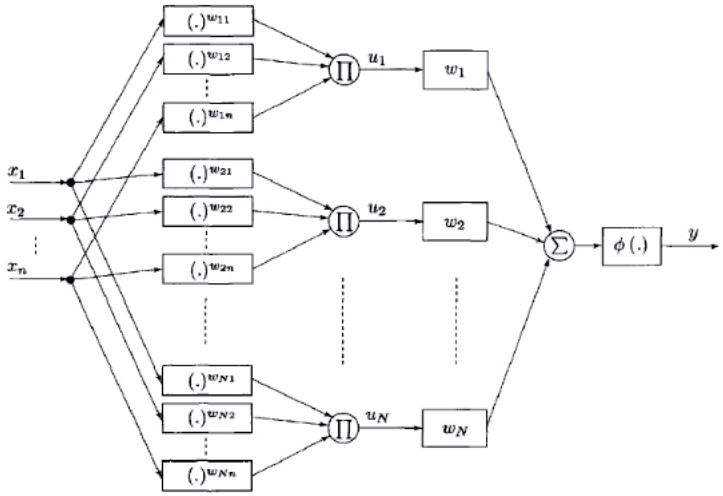


Figura 2.12: Red Sigma-Pi

En una red polinomial como la de la figura 2.12, con n neuronas en la capa de entrada, N en la capa oculta y una neurona de salida, se tiene que para la $j-ésima$ salida u_j, perteneciente a la capa oculta, que [15]:

$$u_j = \prod_{i=1}^{d} x_i^{w_{ji}} \tag{2.7.1}$$

donde $w_{ji}\,\epsilon\,\Re$ es la matriz de pesos no lineales, d es la dimensión de la entrada y x es el vector de entradas para la $j-ésima$ neurona.

La salida de la red neuronal se expresa como:

$$y = \sum_{j=1}^{N} w_j \times u_j \tag{2.7.2}$$

donde $w_j\,\epsilon\,\Re$ representa los pesos de la capa oculta y N es la cantidad de nodos ocultos.

2.8 Perceptrón

El perceptrón es una red de alimentación directa, es decir, la información fluye desde la capa de entrada hacia la capa de salida. Fue desarrollado por F. Rosenblatt hacia finales de la década de los cincuenta, basándose en la regla de aprendizaje de hebb y el modelo de la neurona McCulloch-Pitts. El Perceptrón es un clasificador, asigna a un vector de N valores un valor binario, usando una transformación no lineal. Así cada vector pertenece a una

de las particiones que crea el perceptrón. El perceptrón es una máquina de computación universal y tiene la expresividad equivalente a la lógica binaria ya que podemos crear un perceptrón que tenga el mismo comportamiento que una función booleana NAND y a partir de esta función se puede crear cualquier otra función booleana.

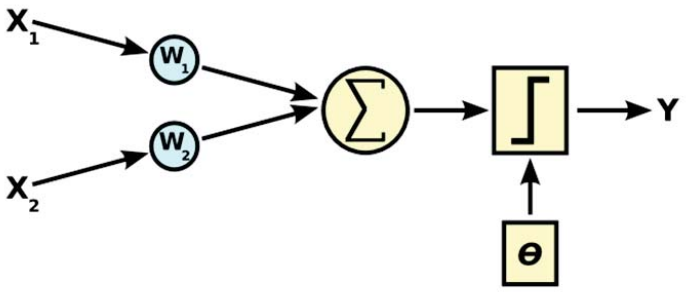


Figura 2.13: Perceptrón Simple

Como se aprecia en figura 2.13, la red consta de entradas x_1 y x_2 , una salida y y un umbral θ, el cual es utilizado como un factor de comparación para producir la salida.

Para obtener la salida de la red, primero se calcula la activación de la célula de salida, mediante la suma ponderada por los pesos de todas las entradas, matemáticamente se representa con la ecuación 2.8.1.

$$y' = \sum_{i=1}^{n} w_i \cdot x_i \tag{2.8.1}$$

En el Perceptrón la función de salida es una función escalón que dependerá del umbral, se detalla en las ecuaciones 2.8.2 y 2.8.3.

$$y = F(y',\theta) \tag{2.8.2}$$

$$F(s,\theta) = \begin{cases} 1 & Si \quad s > 0 \\ -1 & en\,otro\,caso \end{cases} \tag{2.8.3}$$

Según pase θ al lado contrario de la ecuación, es posible obtener la ecuación 2.8.4.

$$y' = \sum_{i=1}^{n} w_i \cdot x_i + \theta \tag{2.8.4}$$

$$F(s) = \begin{cases} 1 & Si \quad s > 0 \\ -1 & en\, otro\, caso \end{cases} \tag{2.8.5}$$

Con las ecuaciones 2.8.4 y 2.8.5 ya es posible determinar fácilmente la forma en que se puede clasificar mediante el modelo del Perceptrón.

Como se puede apreciar en la figura 2.14, si se produce una salida -1, la entrada pertenece a la clase B. Si se produce una salida 1, la entrada pertenece a la clase A.

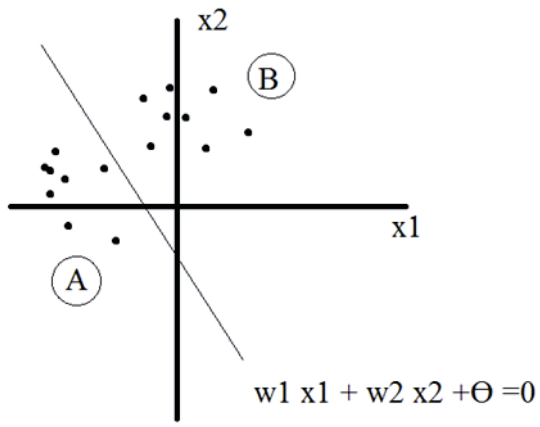


Figura 2.14: Separación de dos clases mediante un Perceptrón

Este ejemplo sólo es para dos dimensiones, viéndolo desde n dimensiones se podría expresar de la siguiente forma [10]:

"Dados conjuntos de puntos en $\Re^n$: $A = (a_1, ..., a_{na})$ y $B = (b_1, ..., b_{nb})$, obtener el hiperplano: $w_1a_1 + ... + w_na_n + \theta = 0$, de tal forma que":

$$\forall \overrightarrow{a} \in A : w_1 \times a_1 + w_2 \times a_2 + ... + w_n \times a_n \tag{2.8.6}$$

$$\forall \overrightarrow{b} \in B : w_1 \times b_1 + w_2 \times b_2 + ... + w_n \times b_n \tag{2.8.7}$$

Entonces para la red se requerirían encontrar los valores que hagan:

$$\forall \overrightarrow{a} \in A : y(\overrightarrow{a}) = 1 \tag{2.8.8}$$

$$\forall \overrightarrow{b} \in B : y(\overrightarrow{b}) = -1 \tag{2.8.9}$$

Con lo cual ya no es una recta la que separa los puntos, sino un hiperplano.

Para el modelo del Perceptrón se utilizan iteraciones de modo que los pesos de las conexiones se modifiquen paulatinamente.

2.9 Perceptrón Multicapa (MLP)

El Perceptrón multicapa (MLP) o red multicapa con conexiones hacia adelante, el cual se muestra en la figura 2.15. Surge como consecuencia de las limitantes presentadas por el Perceptrón simple que sólo podían resolver problemas que sean linealmente separables, esto es, problemas cuyas salidas estén clasificadas en dos categorías diferentes y que permitan que su espacio de entrada sea divido en estas dos regiones por medio de un hiperplano.

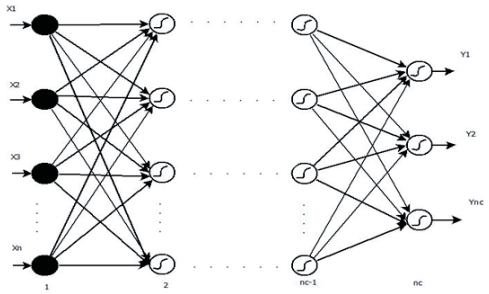


Figura 2.15: Arquitectura del Perceptrón multicapa

El funcionamiento del MLP, es básicamente, aplicar una entrada cuya salida se conoce, luego se calcula la salida de las neuronas de entrada, estas salidas son las entradas de las neuronas de la capa oculta, con estas entradas se calcula la salida de las neuronas ocultas, y con éstas como entrada para las neuronas de salida, se calculan las salidas finales.

En la capa de entrada encuentran las neuronas encargadas de recibir las señales o patrones del exterior para luego propagarlos a la siguiente capa que es la primera capa oculta. Las capas ocultas se encuentran entre la capa de entrada y la capa de salida, pudiendo haber una o más capas ocultas. Las neuronas de las capas ocultas realizan un procesamiento no lineal de los patrones recibidos. La capa se salida es la encargada de proporcionar al exterior la respuesta de la red para cada uno de los patrones de entrada.

Por lo general cada neurona de la capa t está conectada con todas las neuronas de la capa $t + 1$, estas conexiones están dirigidas hacia adelante, razón por la cual se les conoce como redes feedforward, además estas conexiones tienen un peso determinado, que se ajusta en la etapa de entrenamiento. Todas las neuronas de la red tienen un umbral que se trata como una conexión más a la neurona, su valor es constante e igual a uno.

Ahora para entender más en profundidad el Perceptrón multicapa se debe considerar:

• C capas, con $C-2$ capas ocultas y n_c neuronas en la capa c, para $c = 1, 2, \ldots, C$.

• $W^c = (w_{ij}^c)$ la matriz de pesos asociada a las conexiones de la capa c a las $c+1$ para $c = 1, 2, \ldots, C+1$, w_{ij}^c es el peso de la conexión de la neurona i de la capa c a la neurona j de la capa $c+1$.

• $U^c = (u_i^c)$ es el vector de umbrales de las neuronas de la capa c para $c = 2, \ldots, C$.

• a_i^c es la activación de la neurona i de la capa c. Esta se calcula de tres formas según sea el caso.

La activación de la capa de entrada para $i = 1, 2, 3, \ldots, n_1$ se obtiene según la ecuación 2.9.1.

$$a_i^1 = x_i \tag{2.9.1}$$

La activación de las neuronas de la capa oculta para $i = 1, 2, 3, \ldots, n_c$ y $c = 2, 3, \ldots, C-1$ se obtiene mediante la ecuación 2.9.2.

$$a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} \times a_j^{c-1} + u_i^c\right) \tag{2.9.2}$$

La activación de las neuronas de la capa de salida para $i = 1, 2, 3, \ldots, n_c$ se obtiene por medio de la ecuación 2.9.3.

$$y_i = a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} \times a_j^{c-1} + u_i^c\right) \tag{2.9.3}$$

Las funciones de activación más utilizadas son:

- La función sigmoidal (ecuación 2.9.4).

$$f_1(x) = \frac{1}{1+e^{-x}} \tag{2.9.4}$$

- La función tangente hiperbólica (ecuación 2.9.5).

$$f_2(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{2.9.5}$$

- Ambas funciones se relacionan mediante la ecuación 2.9.6

$$f_2(x) = 2 \times f_1(x) - 1 \tag{2.9.6}$$

La función de activación elegida dependerá de tipo de problema o de cómo se quieran expresar las salidas. Por lo general se suele usar una misma función para todas las neuronas de la red, pero esto puede variar según lo requiera el creador de la red.

2.10 Redes Neuronales Recurrentes (RNN)

Los tipos de conexiones de este tipo de redes pueden ser intercapas, es decir, se pueden conectar neuronas de una capa con las de otra capa (anterior y/o posterior), incluso con las del mismo nivel, por lo que existe retroalimentación. También son conocidas como redes feedback. En la figura 2.16 se muestran los tipos de conexiones de las redes recurrentes. a) Conexión con la misma neurona, b) Conexión con neuronas de la misma capa y c) Conexión con neuronas posteriores y anteriores.

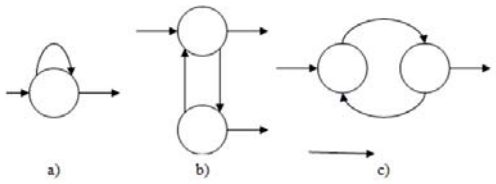


Figura 2.16: Tipos de Recurrencia

Debido a que una neurona, en este tipo de redes, puede estar conectada con cualquier neurona de la red, es necesario incluir la variable tiempo en la activación de la neurona. La ecuación 2.10.1 muestra la inclusión de la variable.

$$a_k(t+1) = f_k(\sum_j w_{kj} \times a_j(t)) \tag{2.10.1}$$

Donde el índice j varía en el conjunto de todas las neuronas que se encuentran conectadas a la neurona k. La presencia de $a_k(t+1)$ en las activaciones de las neuronas recurrentes hace que éstas posean un comportamiento dinámico o temporal.

Dentro de las redes recurrentes más reconocidas se encuentran las redes de Elman y Jordan. La diferencia fundamental que existe entre las redes de Jordan y Elman se basa en que las redes de Jordan reciben copias de las neuronas de la capa de salida y de sí mismas, y en las redes de Elman las neuronas de contexto reciben copias de las neuronas ocultas. Las neuronas de contexto actúan como neuronas receptoras de las conexiones recurrentes y funcionan como una memoria de la red en la que se almacenan las activaciones de las neuronas de una cierta capa de la red en el instante o iteración anterior.

2.11 Aplicaciones de las RNA

Las redes neuronales artificiales son una tecnología computacional que puede utilizarse en un gran número y variedad de aplicaciones. A continuación se presentan algunas áreas en las cuales se aplican redes neuronales artificiales:

- Asociación y Clasificación. Donde los patrones de entrada estáticos o señales temporales deben ser clasificados o reconocidos. Idealmente, un clasificador debería ser entrenado, para que, cuando se le presente una versión ligeramente distorsionada del patrón, pueda ser reconocida correctamente sin problemas.
- Agrupamiento y Categorización de datos que siguen un patrón común indeterminado, como es el caso de aplicaciones de compresión y minería de datos.
- Optimización de soluciones que satisfacen ciertas restricciones, con el fin de maximizar o minimizar la función objetivo. En la gestión empresarial, son decisiones de optimización encontrar los niveles de tesorería, de existencias, de producción, construcción de carteras óptimas, etc. ·
- Pronóstico y Predicción de series de tiempo y que tienen aplicaciones en una gran variedad de áreas.

Capítulo 3

Algoritmos Genéticos

3.1 Introducción

El desarrollo de los Algoritmos Genéticos se debe en gran medida a John Holland, investigador de la Universidad de Michigan. A finales de la década de los 60 desarrolló una técnica, que imitaba, en su funcionamiento, a la selección natural propuesta por Charles Darwin. Aunque originalmente esta técnica recibió el nombre de planes reproductivos, a raíz de la publicación en 1975 de su libro "Adaptation in Natural and Artificial Systems" [16], se conoce principalmente con el nombre de Algoritmos Genéticos. Los GA son, simplificando, algoritmos de optimización, es decir, tratan de encontrar la mejor solución a un problema dado entre un conjunto de soluciones posibles.

A grandes rasgos un GA consiste en una población de soluciones codificadas de forma similar a los cromosomas. Cada uno de estos cromosomas tendrá asociado un valor de bondad o fitness, que cuantifica su validez como solución al problema. En función de este valor se le darán más o menos oportunidades de reproducción. Además, con una cierta probabilidad se realizarán mutaciones de estos cromosomas.

Los GA son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso biológico de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulado por Darwin en 1859 [17]. Imitando el proceso descrito, el algoritmo genético, es capaz de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

La selección natural es el mecanismo que otorga a los individuos más adaptados al medio un mayor número de oportunidades de reproducirse y generar descendientes. Por el contrario individuos poco adaptados producirán un menor número de descendientes. Esto significa, que los genes de los individuos mejor adaptados, se propagarán en sucesivas

generaciones hacia un número creciente de individuos. De esta manera, las especies evolucionan obteniendo características cada vez mejor adaptadas al entorno en el que viven [18].

El poder de los GA proviene del hecho de tratarse de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en que otros métodos encuentran dificultades. Si bien no se garantiza que el GA encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria. El gran campo de aplicación de los GA se relaciona con aquellos problemas para los cuales no existen técnicas especializadas.

3.2 Elementos Comunes de los GA

Al no existir una rigurosa definición de Algoritmo Genético generalmente se encasilla dentro del concepto a aquellos algoritmos que, al menos, comparten ciertas características comunes [19], a continuación se indican cuales son dichas características:

- Función objetivo: Conocida también como función fitness, esta función determina cuales son los individuos, una vez evaluados, que mejor se adaptan a su entorno. La regla general para construir una buena función objetivo es que ésta debe reflejar el valor del individuo de una manera válida. Cuando los individuos obtenidos no son válidos, se deben tener método o técnicas adicionales para solucionar esta problemática.

- Población de Cromosomas: La existencia de cierta población de individuos donde todos sus integrantes son posibles soluciones al problema es de vital importancia, ya que a través del proceso evolutivo por el cual pasarán, lograrán llegar a soluciones cercanas a la óptima, y en el mejor de los casos a la solución óptima. Pueden representarse como un conjunto de parámetros llamados genes (los genes pueden tomar una forma determinada llamada alelo) y el conjunto de genes forma los cromosomas.

- Proceso de selección: En este proceso se utiliza la función fitness para evaluar a cada individuo de la población existente, se asigna un valor a cada uno y este valor indica la capacidad del individuo para resolver el problema. Una vez evaluados los individuos, se selecciona el que tenga un mejor desempeño para crear descendencia.

- Cruzamiento: Consiste en la mezcla de cromosomas de 2 padres a partir de una posición seleccionada de manera aleatoria o en base a un patrón establecido, para la generación de nuevos descendientes. Los nuevos cromosomas presentan soluciones potencialmente mejores que las provistas por sus padres.

- Mutación: Consiste en el intercambio al azar de algún bit del cromosoma, con esto se entrega un pequeño grado de aleatoriedad a los individuos de la población, lo

cual contribuye en disminuir la probabilidad de que queden espacios de búsqueda sin examinar.

3.3 Codificación del Cromosoma

En los diseños iniciales de GA, propuestos por Holland y desarrollados posteriormente por su discípulo Goldberg en [20] se utiliza codificación binaria para representar a los genes que integran cada cromosoma de la población. Sin embargo, puede utilizarse cualquier alfabeto, siempre y cuando éste permita establecer un mapeo entre la representación de los genes en términos del propio alfabeto y el significado físico de los parámetros a los que éstos representan. La selección del esquema de codificación está condicionada generalmente por la naturaleza del problema a optimizar.

3.3.1 Codificación Binaria

Es el tipo de codificación más utilizada, a cada gen se le asigna o el valor 0 o el valor 1. De esta forma cada gen, en cada posición dentro del cromosoma, representa una característica del problema. En la figura 3.1 se muestra un ejemplo de codificación binaria.

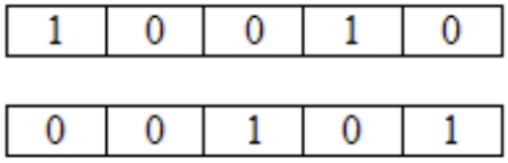


Figura 3.1: Codificación binaria

3.3.2 Codificación Real y Alfabética

En algunas aplicaciones, es más natural utilizar caracteres del alfabeto, números reales y enteros para codificar el cromosoma. Si bien, Holland, argumento que éste tipo de codificación presenta peores resultados que la binaria, algunos estudios comparativos han demostrado que la codificación por valor ha presentado mejor rendimiento [21].

3.4 Algoritmo Genético Básico

Un GA Simple funciona de la siguiente manera:

1. Inicializar aleatoriamente población inicial

2. Evaluar el fitness de todos los individuos de la población inicial
3. Mientras no se cumpla el criterio de parada, hacer
    a) Seleccionar 2 individuos para generar descendientes, en base a algún método se selección.
    b) Cruzar los individuos seleccionados, con una cierta probabilidad, en un punto seleccionado.
    c) Mutar, con cierta probabilidad, los descendientes creados.
    d) Evaluar fitness de individuos recién creados y agregarlos a la población.
    e) Eliminar de la población 2 individuos que tengan peor fitness.
4. Fin Mientras
5. Fin

El criterio de término es dependiente del problema. Generalmente los criterios de término son: número de iteraciones dado o algún criterio de convergencia.

3.5 Métodos de Selección

Los métodos que se detallarán en las secciones 3.5.1, 3.5.2, 3.5.3 y 3.5.4 consisten en la elección de los individuos de la población, de los cuales se crearán descendientes para la nueva generación, con la esperanza que tengan un mayor desempeño respecto de sus ancestros. Seleccionar sólo a los individuos mejor adaptados [22], puede llevar al algoritmo a converger rápidamente y solo a óptimos locales; a su vez, elegir sólo a individuos de baja adaptación puede provocar que la evolución de la población sea más lenta. Debido a lo anterior, es que el método de selección debe mantener un *tradeoff* entre la exploración y explotación sobre el espacio de búsqueda.

3.5.1 Selección por Ruleta

Con este método la probabilidad que tiene un individuo de reproducirse es proporcional a su adaptación. Una vez calculadas estas probabilidades, la selección de los individuos para reproducirse es aleatoria según estos valores. Se puede encontrar una descripción detallada de esta estrategia en [20]. De esta forma se persigue asignarle un mayor probabilidad a los individuos con mejor desempeño.

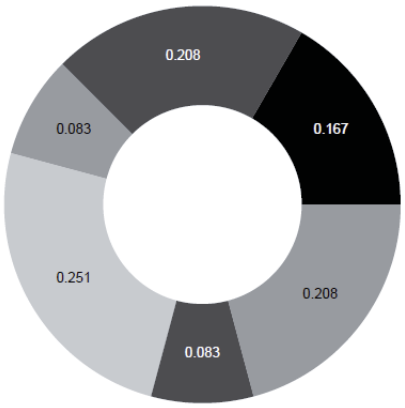


Figura 3.2: Operador de Selección de la Ruleta

3.5.2 Selección de Boltzman

Como se comentó anteriormente, durante la selección no sólo se debe seleccionar al individuo con mejor desempeño. El método de selección de Boltzman utiliza distintos grados de exigencia del fitness, el que aumenta a medida que se incrementa el número de iteraciones, es decir, en las primeras iteraciones, no se es tan riguroso con el desempeño de los individuos de la población que se seleccionan, con lo que se gana una mayor variabilidad y maximizando la exploración, pero a medida que aumentan las iteraciones, se puede incrementar la exigencia del fitness, con lo que se maximiza la capacidad de explotación del algoritmo.

3.5.3 Selección por Torneo

Es un método donde se eligen individuos al azar y en base a comparaciones directas entre los desempeños obtenidos de cado uno de los seleccionados se selecciona el mejor cromosoma para una nueva reproducción. Existen dos versiones de este tipo de selección:

- Determinística: Se escoge al azar un número x de individuos (generalmente 2), de los escogidos se selecciona al que presente un mejor desempeño o fitness.
- Probabilística: En esta versión no se elige el individuo con mejor desempeño, sino que se genera un número aleatorio dentro del intervalo [0,1] y se fija un parámetro k, generalmente 0,5. Si el número aleatorio es mayor que k se escoge el individuo que presente mejor *fitness*, de lo contrario al que presente un menor desempeño [20].

Si el grupo seleccionado es pequeño la probabilidad de seleccionar un valor bajo respecto al total de la población aumenta, lo que permite una mayor variabilidad de la población y una mayor capacidad de exploración del algoritmo. Por otro lado, si la selección es amplia (cercana al número total de individuos de la población) aumenta la probabilidad de

seleccionar al mejor individuo de la población. Mejorando así, la capacidad de explotación del algoritmo.

3.5.4 Selección por Elitismo

Este método fuerza a los algoritmos genéticos a mantener algunos de los individuos de cada generación que presenten un mejor desempeño, esto se hace para no perder los lugares ya explorados del espacio de búsqueda que son buenos candidatos, pues estos individuos pueden ser eliminados sino se reproducen o si son destruidos durante el cruce o mutación.

3.6 Operadores de Cruce

Una vez seleccionados los individuos, éstos son recombinados para producir la descendencia que se insertará en la siguiente generación [23]. A continuación se describen algunos tipos de cruce existentes en la literatura.

3.6.1 Cruce en un Punto

Este operador consiste en seleccionar aleatoriamente, para ambos padres, una posición en el cromosoma e intercambiar el final del cromosoma a partir de dicho punto. De esta manera los individuos descendientes heredan información genética de ambos padres. Para una mejor visualización de este operador es que se adjunta la siguiente imagen.

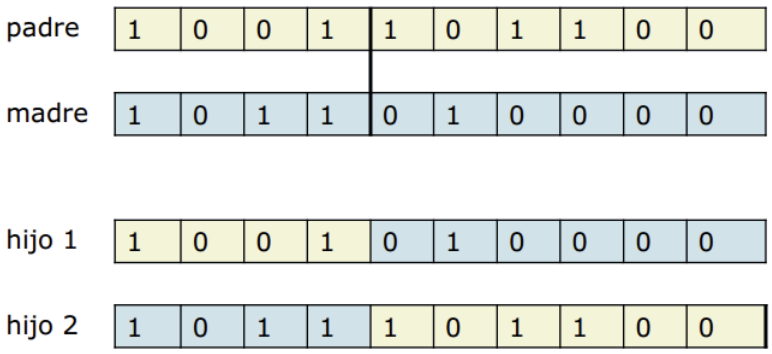


Figura 3.3: Cruce en un Punto

3.6.2 Cruce en Dos Puntos

Es una generalización del cruce en un punto. En vez de cortar por un único punto los cromosomas de los padres, como en el caso anterior, se realizan dos cortes. Se debe garantizar que se generen 3 segmentos de material genético. Luego, se intercambian los

segmentos laterales de uno de los padres con los del otro. Generalizando se pueden añadir más puntos de cruce dando lugar a algoritmos de cruce multipunto, sin embargo existen estudios que desaprueban esta técnica [24].

Figura 3.4: Cruce Dos Puntos

3.6.3 Cruce Aritmético

El cruce aritmético consiste en obtener los valores de los genes del hijo por medio de operaciones aritméticas con los valores de los genes de los padres creando una combinación lineal de ambos cromosomas [25]. Posibles tipos de operaciones pueden ser sumar, restar y promediar los genes de ambos padres. Para estos operadores también se pueden diseñar formas de cruce en las cuales se combinen los métodos anteriores, como por ejemplo, copiar una parte del genotipo del primer padre y obtener la otra parte promediando los valores de los genes de ambos padres.

3.6.4 BLX-α

Este operador genera un hijo, $H = (h_1, \ldots, h_i, \ldots h_n)$, donde h_i es obtenido aleatoriamente desde el intervalo $[\, c_i^1 - I \cdot \alpha\,,\, c_i^2 + I \cdot \alpha\,]$, donde c_i^1 y c_i^2 representan el valor mínimo y máximo, obtenidos de la posición $i - ésima$ de ambos padres. El valor de I se obtiene de la diferencia entre c_i^1 y c_i^2 [26].

El valor de α representa la distancia más allá de los límites establecidos por los genes de los padres que los hijos pueden tomar. Este método permite por lo tanto que los genes de los hijos tomen valores más allá del rango de sus padres, pero que tampoco se encuentren excesivamente alejados. El valor es el más utilizado normalmente para α es *0.5,* pero siempre es dependiente del problema a resolver por lo que debe ajustarse, de algún modo, para cada problema en particular.

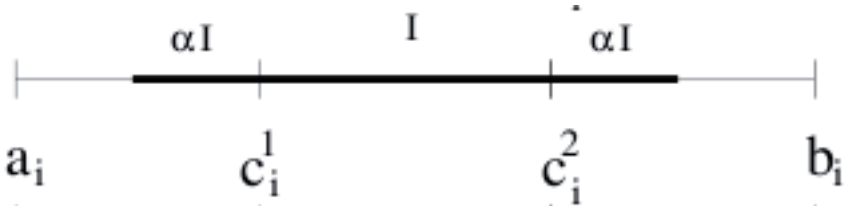


Figura 3.5: Operador de Cruce BLX-α

La figura 3.5 muestra un cruce por mezcla del gen i. El valor $I \cdot \alpha$ se corresponde con la exploración que puede realizar el operador, mientras que el valor I corresponde a la explotación. Para un valor $\alpha = 0$ no hay exploración, solo hay explotación. Por lo tanto el valor de α es el responsable de la exploración de este operador de cruce [26].

3.6.5 SBX-β

Fue propuesto inicialmente por Deb [27] con el objetivo de simular el comportamiento del operador de cruce por un punto en codificaciones binarias. En su trabajo define un factor β denominado factor de propagación que mide la propagación de los genes de los hijos respecto a los de sus padres y que se calcula, para un gen i, como el cociente entre la distancia entre los genes de los padres generados dividido por la distancia entre los genes de los hijos.

Este factor clasifica los operadores de cruce en tres tipos:

- Si $\beta < 1$ el operador de cruce es contractivo, es decir los genes de los hijos se encuentran localizados entre los padres, por lo que el algoritmo se centrará en a la explotación.
- Si $\beta > 1$ el operador de cruce es expansivo, es decir los genes de los padres se encuentran localizados entre los genes de los hijos, lo cual se traduce en que estos explorarán zonas donde no alcanzaban sus padres por lo que el algoritmo se centrará en la exploración.
- Si $\beta = 1$ el operador de cruce es estacionario, es decir los genes de los hijos son iguales que los sus padres.

Analizando la probabilidad de obtener un comportamiento contractivo, estacionario o expansivo se obtuvieron unas ecuaciones de ajustes polinómicas, para el cruce por un punto, que proporcionan la distribución de probabilidad de obtener o un cruce estacionario o expansivo. De este modo la distribución de probabilidad propuesta por Deb viene dada por la ecuación 3.6.1.

$$P(\beta) = \begin{cases} 0,5\,(\eta+1)\,\beta^{\eta} & si\ \beta \leq 1 \\ 0,5\,(\eta+1)\,\frac{1}{\beta^{\eta+2}} & si\ \beta > 1 \end{cases} \tag{3.6.1}$$

donde $\eta\ \epsilon\ \Re^{+}$ determina la probabilidad de generar nuevos individuos próximos o alejados de sus padres. Valores de η pequeños generan hijos muy alejados de sus padres mientras que valores de η grandes los generarán muy próximos a ellos. Para valores $2 \leq \eta \leq 5$ el comportamiento del operador será similar al de un operador de cruce simple. $\eta = 1$ genera cruces tanto expansivos como contractivos, siendo por lo tanto el valor usualmente empleado.

Seguidamente, la implementación se realiza del siguiente modo: primero se genera un número aleatorio $r\ \epsilon\ [0,1]$. A continuación se calcula el valor de β que tiene la misma probabilidad acumulada que r mediante la ecuación 3.6.2.

$$\beta = \begin{cases} 2r^{\frac{1}{\eta+1}} & si\ r \leq 0,5 \\ \left[\frac{1}{2(1-r)}\right]^{\frac{1}{\eta+1}} & si\ r > 0,5 \end{cases} \tag{3.6.2}$$

Una vez obtenido el valor de β, los hijos son obtenidos a partir de la ecuación 3.6.3.

$$\begin{aligned} h_1 &= 0,5\,[(1+\beta)P_1 + (1-\beta)P_2] \\ h_2 &= 0,5\,[(1-\beta)P_1 + (1+\beta)P_2] \end{aligned} \tag{3.6.3}$$

donde h_i serán los hijos a crear y P_i son los padres seleccionados para el cruce.

3.7 Operadores de Mutación

3.7.1 Mutación Uniforme

Un operador de mutación que sustituye el valor del gen seleccionado con un valor uniforme aleatorio seleccionado entre los límites (superior e inferior) especificados por el usuario para ese gen. Este operador mutación sólo se puede utilizar para cromosomas con representación real.

3.7.2 Mutación Gaussiana

Un operador de mutación que añade un valor aleatorio de la distribución de Gauss al gen seleccionado. El valor nuevo del gen se resta si está fuera de los límites (inferior

o superior) especificados por el usuario para dicho gen. Este operador mutación sólo se puede utilizar para cromosomas con representación real.

3.7.3 Mutación No Uniforme

Este operador fue creado por [25]. Donde t es la generación actual y g_{max} es el número máximo de generaciones, entonces el valor del gen mutado se obtiene de la ecuación 3.7.1

$$g_i^{'} = \begin{cases} g_i + \triangle(t, b_i - g_i) & si\ \tau = 0 \\ g_i - \triangle(t, g_i - a_i) & si\ \tau = 1 \end{cases} \tag{3.7.1}$$

donde τ es un número aleatorio que puede tomar el valor 0 o 1, y

$$\triangle(t, y) = y \cdot \left(1 - r^{\left(1 - \frac{t}{g_{max}}\right)^b}\right),$$

donde r es un número aleatorio dentro del intervalo $[0, 1]$ y b es un parámetro elegido por el usuario, que determina el grado de no uniformidad de la mutación, [25] sugiere utilizar $b = 5$. Ésta función obtiene un valor un valor dentro del rango $[0, y]$ y la probabilidad de retornar un número cercano a 0 aumenta a medida que avanza el algoritmo. Esto hace que este operador explore de manera global el espacio de búsqueda al inicio (cuando el valor de t es pequeño) y de manera más local en etapas posteriores.

3.7.4 Mutación Parameter-Based

Este operador fue propuesto por [27] y se utiliza conjunto con el operador de cruce SBX-β. Su objetivo es crear una solución c en la vecindad de una solución padre y. Se presupone que el padre y está acotado a ($y\ \epsilon\ [y_l, y_u]$). El procedimiento es el siguiente:

1. Se genera un número aleatorio u entre 0 y 1
2. Calcular:
$$\delta_q = \begin{cases} [2u]^{\frac{1}{\eta+1}} - 1 & si\ u \leq 0,5 \\ 1 - [2(1-u)]^{\frac{1}{\eta+1}} & si\ u > 0,5 \end{cases}$$
El parámetro η es el índice de distribución para la mutación y toma cualquier valor no negativo. [27] sugiere usar: $\eta = 100 + t$, donde t es la generación actual.
3. El valor de la posición mutada se determina usando: $y' = y + \delta_q \triangle max$, donde $\triangle max$ es la máxima perturbación permitida: $\triangle max = y_u - y_l$.

3.8 Ventajas de los GA

Una de las ventajas más sobresalientes de los GA corresponde a su característica de ser intrínsecamente paralelos, es decir, que a diferencia de los algoritmos en series, los que exploran el espacio de búsqueda de una única dirección, los Algoritmos Genéticos, al tener descendencia múltiple, exploran el espacio de búsqueda en varias direcciones, con esto, no debe comenzar de nuevo en caso de encontrarse con una solución sub-óptima.

Son muy efectivos eludiendo los óptimos locales y descubriendo el óptimo global, esto se debe a que durante el cruzamiento ocurre una transferencia de información entre los candidatos con mejor desempeño, beneficiándose unos de otros, mezclándose y potenciándose, pues eventualmente, la descendencia puede tener todas las virtudes de los padres y ninguna de sus debilidades.

Los GA no dependen de un conocimiento previo, parten con una población aleatoria, realizando cambios de la misma forma en los individuos, y usando el *fitness* para determinar si los cambios producen mejoras. En contraste, las técnicas que dependen de conocimientos previos fracasan cuando éste se encuentra ausente, además, parten descartando camino y perdiendo así cualquier solución novedosa que exista.

3.9 Desventajas de los GA

Una de las principales desventajas radica en la dificultad para definir la representación del problema. La forma para especificar las soluciones debe ser tolerante a cambios aleatorios sin producir errores fatales o resultados sin sentidos. La manera generalmente utilizada es definir a los individuos como listas de números, ya sean binarios, reales o enteros.

La dificultad para definir una función *fitness*, pues el proceso no es trivial, ya que debe permitir encontrar el mejor desempeño y que éste signifique una mejor solución para un problema determinado, una definición deficiente o inadecuada puede ser incapaz de resolver el problema o bien dar solución a un problema distinto.

El cuidado que requiere en la selección de elementos claves como; tamaño de la población, ritmo de mutación, cruzamiento y el tipo de selección, pues si el tamaño de la población es muy pequeño, es posible que el GA no sea capaz de abarcar todo el espacio de posibles soluciones o si el ritmo de mutación, o el método de selección no es el adecuado, la población puede entrar en una catástrofe de errores.

3.10 Aplicaciones de los GA

Según la literatura cuando un problema posee un espacio de búsqueda amplio, y no se tiene mucha información de éste, probablemente los Algoritmos Genéticos tengan un buen desempeño.

- Problemas de Predicción: Se utilizaron Algoritmos Genéticos para predecir el futuro rendimiento de un total de 1600 acciones ofertadas públicamente [28].
- Problemas de Optimización: Se utilizaron GA para definir la ubicación de las órbitas de los satélites de manera de minimizar los apagones a causa de la cobertura [29].
- Evolución de redes neuronales: Los GA fueron utilizados para evolucionar Redes Neuronales Artificiales que jugaran partidas de damas, los resultados fueron exitosos, llegando a jugar con el mejor jugador de damas del mundo, y aunque perdió 2-4, la causa fue simplemente la carencia de una base de datos de finales de partida.

Capítulo 4

Redes Neuroevolutivas

4.1 Introducción

Una Red Neuroevolutiva se puede definir como toda aquella Red Neuronal Artificial cuya estimación de parámetros internos se realiza mediante la utilización de Algoritmos Evolutivos. La neuroevolución consiste en utilizar algoritmos genéticos para hacer evolucionar una población de individuos utilizando redes neuronales [30]. El algoritmo genético tomará una población de posibles soluciones al problema y generará una nueva, mejor o igual que la anterior, a partir de la recombinación y el uso de operadores genéticos de cruce y mutación.

En el proceso de evolución son esenciales los operadores de selección (cruce y mutación) para introducir, por una parte, presión selectiva, , por otra, diversidad, de forma tal que el algoritmo de búsqueda de la mejor red presente un compromiso entre su capacidad de explotar la localización de las mejores soluciones y explorar otras localizaciones del espacio, con el fin de obviar el problema de la obtención de óptimos locales.

En este capítulo se presenta el modelo de Red Neuroevolutiva ultilizando para dicha estimación un Algoritmo Genético. En una primera sección se presentan las principales métricas utilizadas para medir la calidad del modelo y en las siguientes secciones se presentan los modelos desarrollados.

Hay que señalar que el modelo genérico es en una RNP utilizando AG, además se presentarán variantes de AG con la finalidad de originar nuevos modelos para abordar y dar solución a la problemática. Finalmente se presentan los resultados del estudio comparando los modelos presentados de manera que se determine cuál de ellos presenta una mejor capacidad predictiva frente a la estimación del costo de producir elementos de tuberías.

4.2 Descripción y Tratamiento de los Datos

Los datos utilizados corresponden a los valores asociados a los costos de producir tuberías de acero. Estos datos se encuentran ordenados según el peso, el tipo de soldadura, el diámetro y la dificultad, y en total se tienen 2253 registros por cada clasificación. Se desecharon otros tipos de inductores de costos, ya que en [3] se indica que con los datos mencionados anteriormente se predice de mejor manera el costo total de fabricar tuberías.

Para el modelo propuesto, los datos asociados fueron normalizados con la ecuación (4.2.1). A continuación, los datos fueron divididos en dos grupos: 80 % de los datos se utilizaron durante la etapa de entrenamiento, mientras que un 20 % se emplearon en a fase de validación. Ambos grupos fueron separados en entradas y salidas.

$$y(t) = \frac{y(t)}{max(t)} \tag{4.2.1}$$

4.3 Función Fitness

4.3.1 Porcentaje de Error Medio Absoluto

Es una de las métricas más recomendadas en la estimación de costos para medir la precisión de los modelos creados y es recomendada por varios libros, como por ejemplo en [31]. El Error Porcentual Medio Absoluto (MAPE) proporciona una indicación de que tan grandes son los errores de pronóstico comparados con los valores reales de la serie. También, correspondiente a la operación interna de la sumatoria se encuentra el Porcentaje de Error Absoluto (APE).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(y_i - \hat{y}_i)}{y_i}\right| \ , y_i \neq 0. \tag{4.3.1}$$

Donde y representa los valores observados, $\hat{y}$ los valores pronosticados y n es el tamaño de la muestra.

4.4 Métricas de Evaluación

Existe una gran cantidad de índices que buscan medir la bondad de un modelo, la elección se hará en base a los objetivos que persigue el investigador. Por lo general, para medir la eficacia de los modelos se realiza una división de la muestra en dos grupos, la primera parte de la muestra es con la cual construye el modelo y con la segunda se evalúa la eficacia de este mismo.

A continuación se indica la nomenclatura a utilizar y, luego, una breve descripción de las métricas a utilizar con su correspondiente ecuación matemática.

- $\hat{y}$ denota el valores pronosticados por el modelo correspondiente.
- $\bar{y}$ es la media de los datos observados.
- y son los datos observados.
- $e = y - \hat{y}$ corresponde al error residual del pronóstico.
- n corresponde al tamaño de la muestra.

4.4.1 Error Cuadrático Medio

El Error Cuadrático Medio (ECM), corresponde a una de las técnicas más utilizadas para medir la calidad y posterior selección de un modelo que busque explicar una serie de datos en particular. Es por esto, que el ECM será utilizado para determinar la mejor configuración de los parámetros internos de la Red Neuronal Polinomial. Para lograr encontrar los valores adecuados de dichos parámetros es que se hará uso de la ecuación (4.4.1), la cual consiste en la suma de las diferencias al cuadrado entre lo real y lo proyectado por el modelo. Su ecuación es la siguiente:

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.4.1}$$

4.4.2 Coeficiente de Determinación

El Coeficiente de Determinación (R^2) mide la dependencia que existe entre los datos reales y los pronosticados. Un valor muy cercano a 0 muestra independencia y dependencia en el caso que en que el valor se acerque a 1. Se calcula de la siguiente manera:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})}, \tag{4.4.2}$$

donde $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ corresponde a la varianza de los errores y $\sum_{i=1}^{n}(y_i - \overline{y})$ a la varianza de la variable real.

4.4.3 Raíz del Error Cuadrático Medio

El Error Cuadrático Medio (RMSE) corresponde a la raíz cuadrada del MSE y es una estimación de la desviación estándar del error de predicción. Se calcula a través de la siguiente ecuación:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \tag{4.4.3}$$

4.5 Software y Hardware utilizado

La implementación se ha realizado a través del software matemático *Matlab* en su versión R2011a, el cual ofrece un entorno de desarrollo integrado y su propio lenguaje de programación denominado M. Es a través de este entorno de desarrollo que se han implementado tanto el algoritmo genético en conjunto con sus variantes, así como también el diseño de las interfaces y el análisis de los resultados.

Las características del sistema que se dispuso para la realización de las pruebas fueron las siguientes:

- Procesador Intel(R) CORE(TM) i3 M330 2.13 GHz.
- Memoria RAM 3.0 Gb SoDimm DDR3 800 MHz.
- Sistema Operativo Windows 7 32 bits.

4.6 Descripción del Modelo Genérico

El modelo general, consiste en una Red Neuronal Polinomial, cuyos parámetros internos serán obtenidos a través de la utilización de Algoritmos Genéticos. La idea central es que cada individuo perteneciente a la población simbolice una posible solución perteneciente al espacio de búsqueda, de esta manera se logrará a través de un proceso iterativo obtener aquella solución que, bajo los criterios de evaluación, represente de mejor manera el comportamiento de los datos.

A continuación se presentará el conjunto de pasos que permitirán llevar a cabo el procedimiento anteriormente descrito:

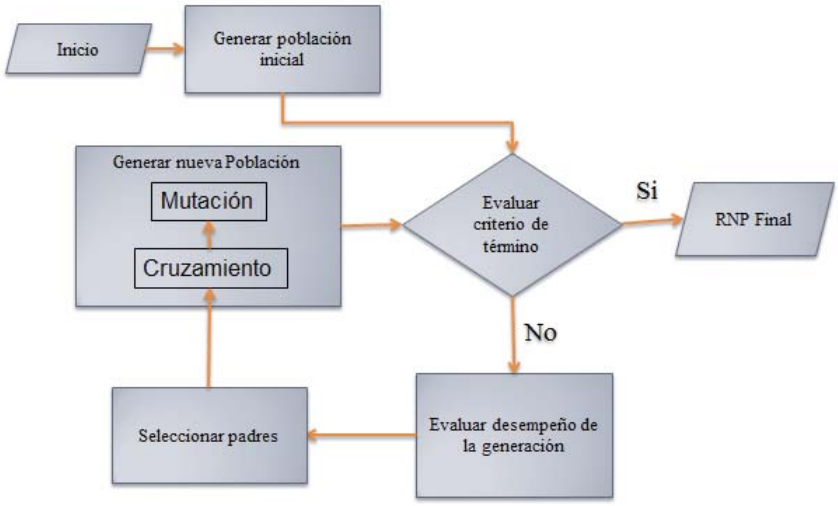


Figura 4.1: Modelo General PNN-GA.

4.6.1 Secuencia de Pasos

- **Paso 1: Crear Población Inicial.** consiste en un total de N individuos, cada uno de estos representados como un cromosoma, su codificación será real y se inicializará con valores aleatorios.
  Cada cromosoma es una posible solución al problema y cuentan con la siguiente estructura:

W_{11}	...	W_{1n}	...	W_{m1}	...	W_{mn}	...	H_1	...	H_m

Tabla 4.1: Codificación Cromosoma

  Siendo m el número de neuronas ocultas y n el número de neuronas de entrada, $W_1, \ldots, W_m$ el conjunto de pesos lineales asociados a la neurona oculta i $(1 < i < n)$ y H_j son los pesos lineales ubicados entre la capa oculta y la capa de salida.

- **Paso 2: Evaluación de la Población Inicial.** La población inicial será evaluada utilizando una Red Neuronal Polinomial, para obtener el valor fitness de cada individuo o cromosoma, para el cálculo de éste valor se utiliza el Error Cuadrático Medio, el cual es considerado como función fitness.

- **Paso 3: Operador de Selección.** Este operador consiste en seleccionar los individuos que tendrán descendencia. En este caso en particular, se seleccionan 2 individuos aleatoriamente desde la población, los cuales serán los padres de la nueva generación.

- **Paso 4: Operador de Cruce.** Consiste cruzar el material genético de los padres seleccionados y obtener una descendencia. En lugar de cruzar genes individuales se cruzarán neuronas completas. El objetivo es mantener estructuras complejas, evitando que sean destruidas por el operador de cruza [32]. Todos los pesos que parten

de una misma neurona serán tratados como una unidad indivisible. La figura 4.2 muestra gráficamente los puntos de cruza para una red neuronal con dos neuronas de entrada y 4 neuronas de salida.

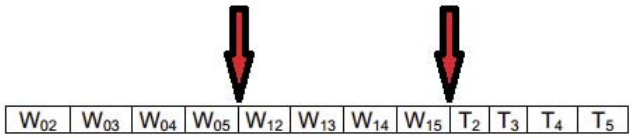


Figura 4.2: Puntos de Cruce

Para este caso en particular se utiliza el cruce uniforme, logrando que los hijos creados contengan material genético de ambos padres. En la figura 4.3 se muestra un ejemplo del tipo de cruce utilizado.

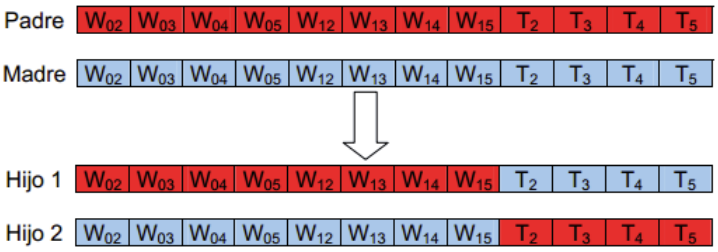


Figura 4.3: Cruce en un punto

- **Paso 5: Operador de Mutación.** Se utiliza mutación gaussiana, debido a que la codificación de los cromosomas es de tipo real y se aplica a un gen, elegido aleatoriamente, de los hijos obtenidos en los pasos anteriores. Este tipo de mutación consiste en sumar un valor aleatorio, al gen seleccionado, tomado de una distribución gaussiana de media 0 y varianza 0.5.
- **Paso 6: Obtención del Valor Fitness**. Este valor es calculado a través de la evaluación de la función fitness. Esta función, recibe como entrada el resultado de los valores pronosticados por la Red Neuronal Polinomial, la funcion fitness considerada corresponden a el Error Cuadrático Medio.
- **Paso 7: Actualización.** Los 2 individuos nuevos son añadidos a la población existente, luego se eliminan 2 individuos, los cuales representan las peores soluciones al problema.
- **Paso 8: Finalización**. Se verifica si se cumple con el criterio de término definido, de no ocurrir lo anterior, se vuelve al paso 3, en caso contrario continuar.

- **Paso 9: Obtención de los Parámetros Óptimos.** Una vez que se ha terminado con la ejecución del Algoritmo Genético, se obtiene el individuo que mejor representa la solución del problema, es decir, cuyo valor fitness presentó el menor valor de la población. En términos de representación, las topologías se denotarán de la forma (E, O, S), siendo E el número de neuronas de entrada, O el número de neuronas ocultas y S el número de nodos de salida de la red.

- **Paso 10: Evaluación del Modelo.** Luego de obtener los parámetros del modelo, éste debe someterse al proceso de evaluación con los datos pertenecientes a la etapa de validación, a través del uso de métricas definidas en 4.4.

4.6.2 Selección de topología y parámetros del Modelo

El primer modelo a evaluar es el que utiliza el Algoritmo Genético más simple o básico. La configuración topológica de la red se establecerá mediante prueba y error. En un inicio se comenzará el proceso de ajuste con 100 iteraciones, una población de 10 individuos y con la totalidad de los datos de entrada, es decir, 4 neuronas de entrada. Además, cabe señalar que se utilizarán los datos de entrenamiento para determinar los parámetros. El método de selección de las configuraciones de prueba fueron realizadas empleando la siguiente modalidad y orden:

- **Neuronas Ocultas:** Para obtener el número óptimo de nodos ocultos, basado en los valores indicados en la sección 4.6.2, se probará variando el número de nodos ocultos entre 1 y 8, asignando a este parámetro el valor que presente el promedio más bajo del MAPE. Para determinar éste parámetro se utilizarán como parámetros iniciales 4 neuronas de entradas, 100 iteraciones y 10 individuos conformarán la población. Se realizarán 20 iteraciones por cada configuración.

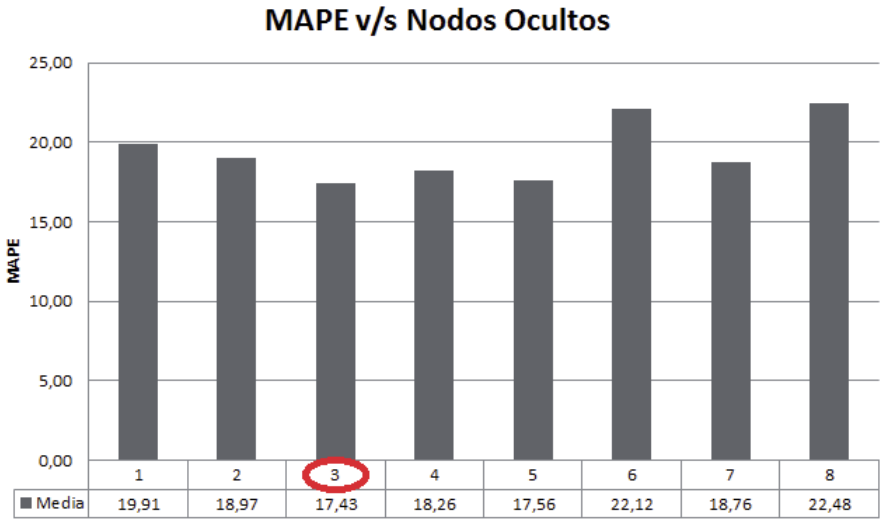


Figura 4.4: Modelo Genérico - Nodos Ocultos

Se puede observar, en la imagen 4.4, que el número de neuronas ocultas que tiene el MAPEmás bajo es 3, por lo tanto éste es el valor que se utilizará para las neurona ocultas de aquí en adelante en este modelo.

- **Número de Iteraciones:** Para obtener el número óptimo de iteraciones, basado en los valores indicados en la sección 4.6.2 y los ya obtenidos, se probará variando el número de iteraciones entre 100 y 1300, asignando a este parámetro el valor que presente el promedio más bajo de MAPE. Para determinar éste parámetro se utilizarán como parámetros 4 neuronas de entradas, 3 nodos ocultos y 10 individuos conformarán la población. Se realizarán 20 iteraciones por cada configuración.

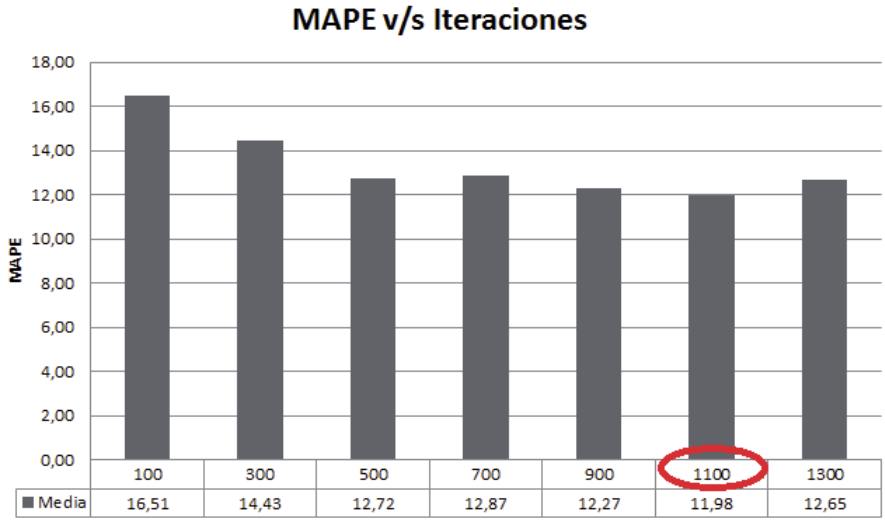


Figura 4.5: Modelo Genérico - Iteraciones

Se puede observar, en la imagen 4.5, que el número de iteraciones que tiene el MAPE más bajo, es 1100, por lo tanto éste es el valor que se utilizará para las iteraciones de aquí en adelante en este modelo.

- **Número de Individuos:** Para obtener el número óptimo de individuos, basado en los valores indicados en la sección 4.6.2 y los ya obtenidos, se probará variando el número de individuos entre 10 y 60, asignando a este parámetro el valor que presente el promedio más bajo de MAPE. Para determinar éste parámetro se utilizarán como parámetros 4 neuronas de entradas, 1100 iteraciones y 3 nodos ocultos. Se realizarán 20 iteraciones por cada configuración.

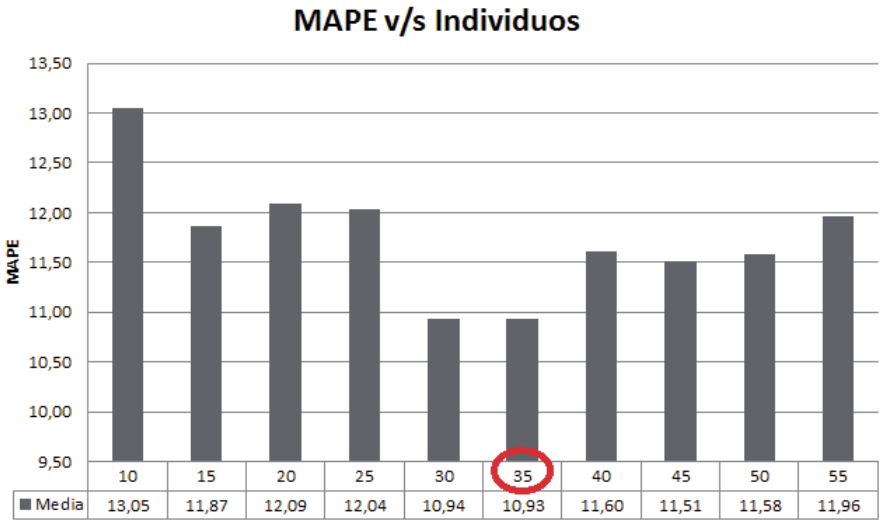


Figura 4.6: Modelo Genérico - Individuos

Se puede observar, en la imagen 4.6, que el número de individuos de la población que tiene el MAPE más bajo es 35, por lo tanto éste es el valor que se utilizará para el número de individuos de aquí en adelante en este modelo.

Finalmente se determinó que la topología (4,3,1) fue la que mejores resultados obtuvo, además de encontrar que el número de iteraciones e individuos para el modelo es de 1100 y 35 respectivamente.

4.7 Modelo Family Competition

El segundo modelo a evaluar es el que utiliza una variante del Algoritmo Genético en conjunto con una estratégia, la cual se denomina Family Competition. Esta estratégia consiste en que cada individuo de la población actual tendrá su propia "familia" o subpoblación de nuevos individuos, de estos individuos sólo formarán parte de la nueva generación sólo los mejores de cada "familia". El operador de cruce a utilizar es el BLX-α y para la mutación se utilizará el operador Mutación No Uniforme, ambos descritos en el Capítulo 3. En la imagen 4.7 se esboza un diagrama del funcionamiento de la estratégia Family competition.

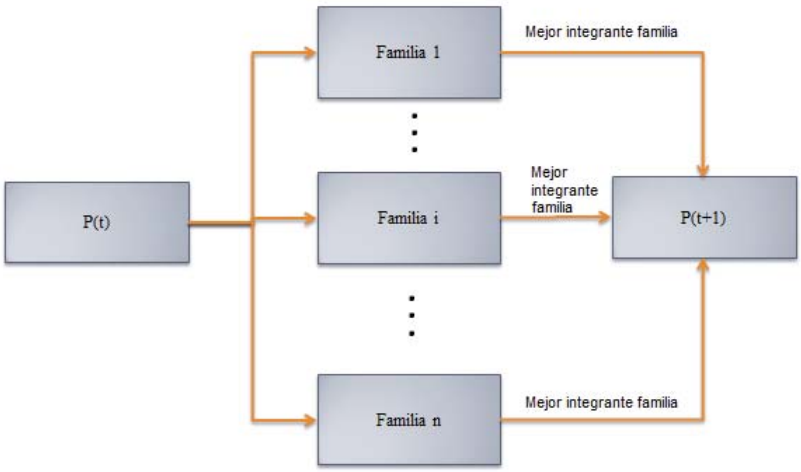


Figura 4.7: Modelo Family Competition

La configuración topológica de la red se establecerá mediante prueba y error. En un inicio se comenzará el proceso de ajuste con 100 iteraciones, una población de 10 individuos y con la totalidad de los datos de entrada, es decir, 4 neuronas de entrada. También se debe ajustar el tamaño de las subpoblaciones, el valor inicial será 10 individuos, luego este valor se modificará y será $\frac{1}{3}$ del tamaño de la población. Además, cabe señalar que se utilizarán los datos de entrenamiento para determinar los parámetros.

4.7.1 Selección de topología y parámetros del Modelo

El método de selección de las configuraciones de prueba fueron realizadas empleando la siguiente modalidad y orden:

- **Número de Iteraciones:** Para obtener el número óptimo de iteraciones, basado en los valores indicados en la sección 4.7 y los ya obtenidos, se probará variando el número de iteraciones entre 100 y 900, asignando a este parámetro el valor que presente el promedio más bajo de MAPE. Para determinar éste parámetro se utilizarán como parámetros 4 neuronas de entradas, 10 individuos conformarán la población, 10 individuos será el tamaño de las subpoblaciones y se utilizarán 5 nodos ocultos. Además, se realizarán 20 iteraciones por cada configuración.

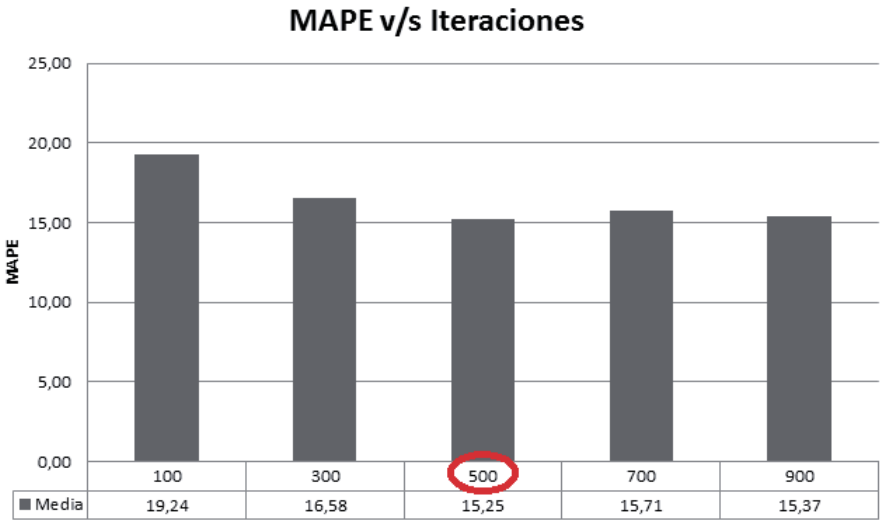


Figura 4.8: Modelo Family Competition - Iteraciones

Se puede observar, en la imagen 4.8, que el número de iteraciones que obtiene el mejor MAPE, es 500, por lo tanto éste es el valor que se utilizará para la cantidad de iteraciones de aquí en adelante en este modelo.

- **Neuronas Ocultas:** Para obtener el número óptimo de nodos ocultos, basado en los valores indicados en la sección 4.7, se probará variando el número de nodos ocultos entre 1 y 7, asignando a este parámetro el valor que presente el promedio más bajo de MAPE. Para determinar éste parámetro se utilizarán como parámetros 4 neuronas de entradas, 10 individuos conformarán la población, 10 individuos será el tamaño de las subpoblaciones y la cantidad de iteraciones es 500. Además, se realizarán 20 iteraciones por cada configuración.

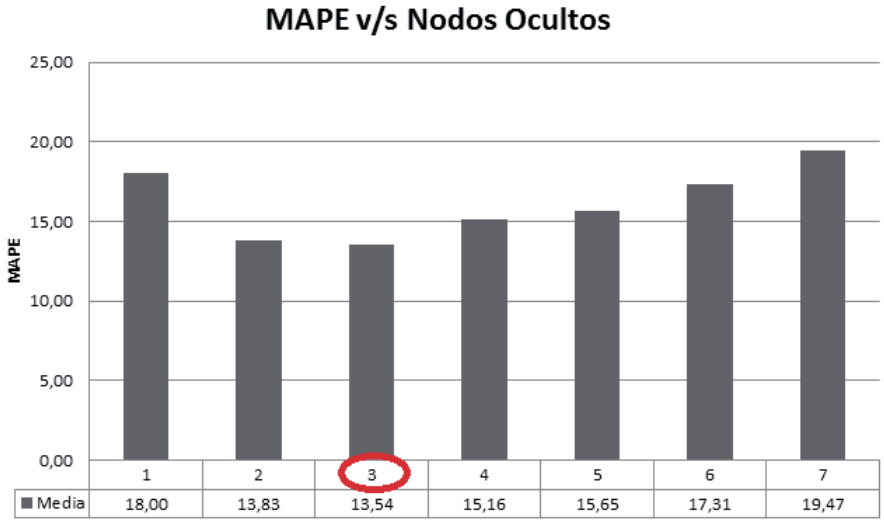


Figura 4.9: Modelo Family Competition - Nodos Ocultos

Se puede observar, en la imagen 4.9, que el número de nodos ocultos que obtiene el mejor MAPE, es 3, por lo tanto éste es el valor que se utilizará para la cantidad de iteraciones de aquí en adelante en este modelo.

- **Número de Individuos:** Para obtener el número óptimo de individuos, basado en los valores indicados en la sección 4.7 y los ya obtenidos, se probará variando el número de individuos entre 10 y 50, asignando a este parámetro el valor que presente el promedio más bajo de MAPE. Para determinar éste parámetro se utilizarán como parámetros 4 neuronas de entradas, 3 neuronas ocultas, 10 individuos será el tamaño de las subpoblaciones y la cantidad de iteraciones es 500. Además, se realizarán 20 iteraciones por cada configuración.

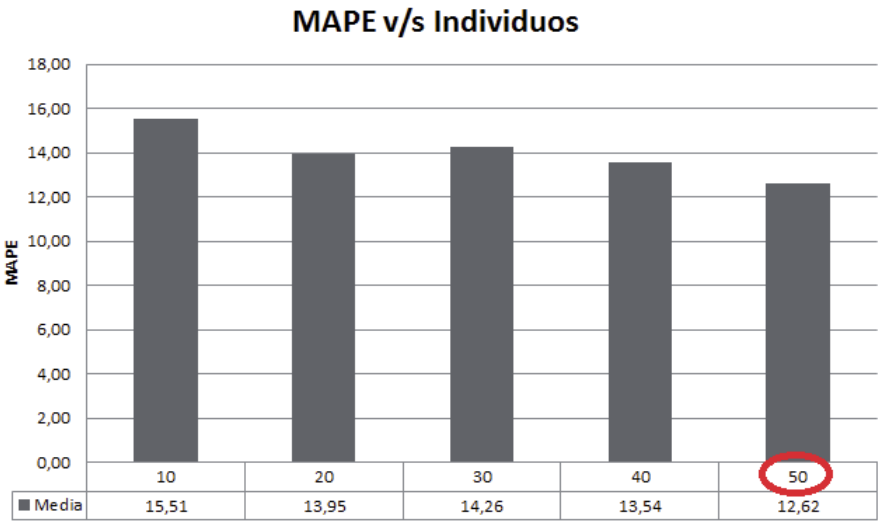


Figura 4.10: Modelo Family Competition - Individuos

Se puede observar, en la imagen 4.8, que la cantidad de individuos que obtiene el mejor MAPE, es 50, por lo tanto éste es el valor que se utilizará para la cantidad de iteraciones de aquí en adelante en este modelo.

- **Tamaño de las subpoblaciones:** Este parámetro tendrá un valor aproximado a $\frac{1}{3}$ del tamaño de la población, es decir, su valor es 16.

Finalmente se determinó que la topología (4,3,1) fue la que mejores resultados obtuvo, además de encontrar que el número de iteraciones e individuos para el modelo es de 500 y 50 respectivamente, además, el tamaño de las subpoblaciones a utilizar es 16.

4.8 Modelo SBX

El tercer modelo es similar al Modelo Genérico, puesto que mantiene su estructura y no utiliza una estratégia en particular como el modelo anterior. Sin embargo, los operadores

de cruce y de mutación se han modificado. El operador de cruce a utilizar es conocido como SBX-β y el operador de mutación es conocido como Mutación Parameter-Based, ambos descritos en el Capítulo 3.

La configuración topológica de la red se establecerá mediante prueba y error. En un inicio se comenzará el proceso de ajuste con 100 iteraciones, una población de 10 individuos y con la totalidad de los datos de entrada, es decir, 4 neuronas de entrada. Además, cabe señalar que se utilizarán los datos de entrenamiento para determinar los parámetros.

4.8.1 Selección de topología y parámetros del Modelo

El método de selección de las configuraciones de prueba fueron realizadas empleando la siguiente modalidad y orden:

- **Número de Iteraciones:** Para obtener el número óptimo de iteraciones, basado en los valores indicados en la sección 4.8 y los ya obtenidos, se probará variando el número de iteraciones entre 100 y 1700, asignando a este parámetro el valor que presente el promedio más bajo de MAPE. Para determinar éste parámetro se utilizarán como parámetros 4 neuronas de entradas, 10 individuos conformarán la población y se utilizarán 3 nodos ocultos. Además, se realizarán 20 iteraciones por cada configuración.

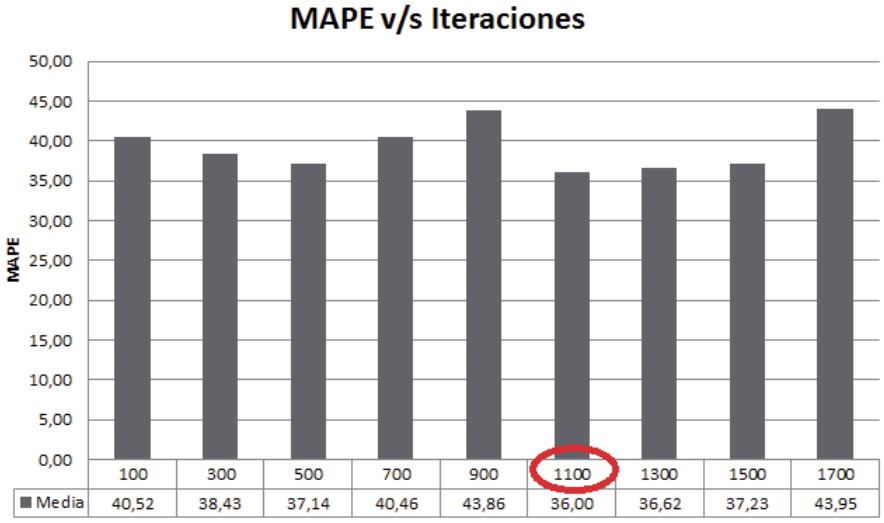


Figura 4.11: Modelo SBX - Iteraciones

Se puede observar, en la imagen 4.11, que el número de iteraciones que obtiene el mejor MAPE, es 1100, por lo tanto éste es el valor que se utilizará para la cantidad de iteraciones de aquí en adelante en este modelo.

- **Neuronas Ocultas:** Para obtener el número óptimo de nodos ocultos, basado en los valores indicados en la sección 4.8, se probará variando el número de nodos ocultos

entre 1 y 8, asignando a este parámetro el valor que presente el promedio más bajo de MAPE. Para determinar éste parámetro se utilizarán como parámetros 4 neuronas de entradas, 10 individuos conformarán la población y la cantidad de iteraciones es 1100. Además, se realizarán 20 iteraciones por cada configuración.

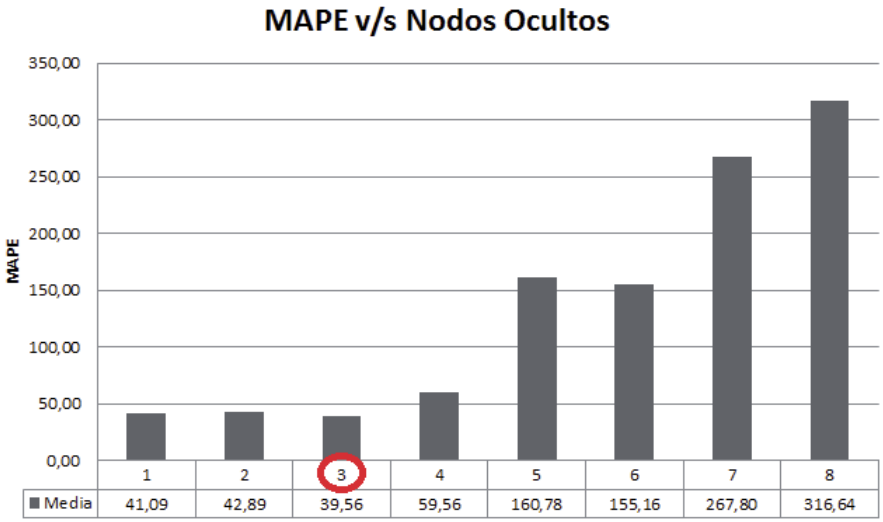


Figura 4.12: Modelo SBX - Nodos Ocultos

Se puede observar, en la imagen 4.12, que el número de nodos ocultos que obtiene el mejor MAPE, es 3, por lo tanto éste es el valor que se utilizará para la cantidad de iteraciones de aquí en adelante en este modelo.

- **Número de Individuos:** Para obtener el número óptimo de individuos, basado en los valores indicados en la sección 4.8 y los ya obtenidos, se probará variando el número de individuos entre 10 y 50, asignando a este parámetro el valor que presente el promedio más bajo de MAPE. Para determinar éste parámetro se utilizarán como parámetros 4 neuronas de entradas, 3 neuronas ocultas y la cantidad de iteraciones es 1100. Además, se realizarán 20 iteraciones por cada configuración.

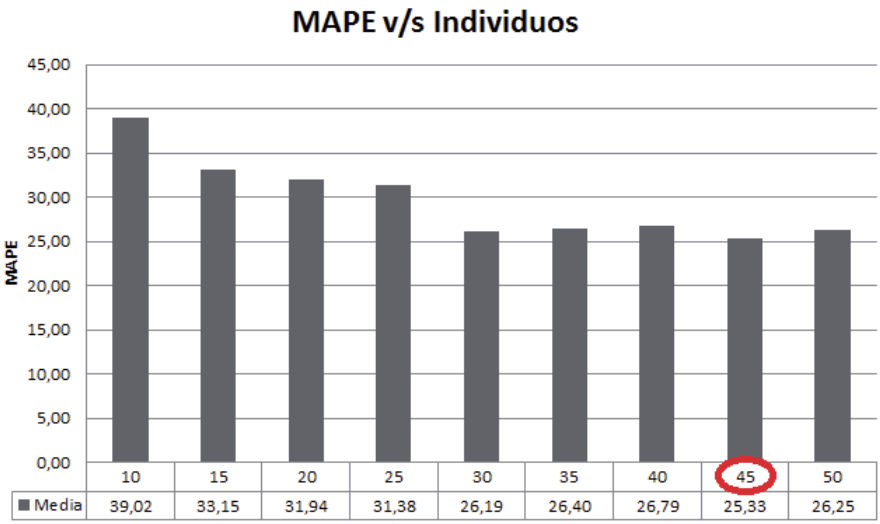


Figura 4.13: Modelo SBX - Individuos

Se puede observar, en la imagen 4.11, que la cantidad de individuos que obtiene el mejor MAPE, es 45, por lo tanto éste es el valor que se utilizará para la cantidad de iteraciones de aquí en adelante en este modelo.

Finalmente se determinó que la topología (4,3,1) fue la que mejores resultados obtuvo, además de encontrar que el número de iteraciones e individuos para el modelo es de 1100 y 45 respectivamente.

Capítulo 5

Análisis de Resultados

5.1 Introducción

En el presente capítulo se realizará el análisis de resultados de los modelos propuestos, dichos modelos se han denominado: Génerico, Family Competition y SBX, debido a los operados de Algoritmos Genéticos utilizados. En cuanto a los modelos desarrollados cabe mencionar que se conforman de una Red Neuronal Polinomial con optimización de parámetros a través de una variante de Algoritmo Genético. Por otro lado, el proceso de evaluación se desarrolló en dos etapas, la de entrenamiento y la de validación. La etapa de entrenamiento se hizo a partir del 80 % de los datos, mientras que la de validación se realizó con el 20 % restante. Los datos se normalizaron dentro del intervalo $[0, 1]$, para evitar que datos con valores altos puedan absorber los valores menores.

En el proceso de entrenamiento se desarrollaron diferentes configuraciones de los modelos, para Genérico se obtuvieron 25 modelos, para el Family Competition se probaron 17 modelos y para el SBX se lograron 26 modelos, obteniendo así un total de 68 modelos. Por otro lado, función *fitness* utilizada para el proceso de entrenamiento es el MAPE.

5.2 Análisis Resultados Obtenidos

A continuación se presentan las tablas correspondientes a los resultados finales además de los gráficos de cada una de las configuraciones elegidas para cada modelo. En todos los modelos se utilizó la configuración obtenida en el capítulo anterior, ya que presentaban los mejores resultados frente al problema. A continuación se presentan las tablas de resultados de los distintos modelos y posteriormente las gráficas del mejor pronóstico, MAPE y R^2.

Parámetros	Valor
Nodo Entradas	4
Nodos Ocultos	3
Salida	1
Tamaño Población	35
Iteraciones	1100

Tabla 5.1: Parámetros Modelo Genérico

Modelo Genérico (4,3,1)			
Ejecución	MAPE	R^2	RMSE
1	9,433	0,9772	0,01981
2	**8,817**	**0,9893**	**0,01060**
3	10,676	0,9673	0,02726
4	9,658	0,9710	0,01811
5	9,687	0,9878	0,01355
6	9,793	0,9736	0,02242
7	10,327	0,9489	0,03306
8	8,939	0,9867	0,01198
9	9,776	0,9881	0,01187
10	9,070	0,9731	0,01723
Media	9,618	0,9763	0,01859
Min	8,817	0,9489	0,01060
Max	10,676	0,9893	0,03306
Varianza	0,346643829	0,00015841	0,00005342

Tabla 5.2: Resultados Modelo Genérico

Parámetros	Valor
Nodo Entradas	4
Nodos Ocultos	3
Salida	1
Tamaño Población	50
Tamaño Subpoblaciones	16
Iteraciones	1100
α	0.5
b	5

Tabla 5.3: Parámetros Modelo Family Competition

Modelo Family Competition (4,3,1)			
Ejecución	MAPE	R^2	RMSE
1	**8,948**	**0,9680**	**0,02222**
2	10,124	0,9566	0,02536
3	9,445	0,9640	0,02089
4	10,376	0,9745	0,02367
5	10,960	0,9346	0,03006
6	10,209	0,9414	0,03509
7	11,378	0,9221	0,03954
8	9,552	0,9760	0,01584
9	10,276	0,9501	0,02735
10	11,669	0,9712	0,03702
Media	10,294	0,9559	0,02770
Min	8,948	0,9221	0,01584
Max	11,669	0,9760	0,03954
Varianza	0,737511	0,00033726	0,00005845

Tabla 5.4: Resultados Modelo Family Competition

Parámetros	Valor
Nodo Entradas	4
Nodos Ocultos	3
Salida	1
Tamaño Población	45
Iteraciones	1100
η	1

Tabla 5.5: Parámetros Modelo SBX

Modelo SBX (4,3,1)			
Ejecución	MAPE	R^2	RMSE
1	21,978	0,9364	0,02859
2	23,380	0,9688	0,02178
3	23,795	0,9333	0,03594
4	27,776	0,9507	0,02508
5	24,841	0,9516	0,02551
6	21,290	0,9335	0,02836
7	21,753	0,9390	0,02527
8	23,660	0,9390	0,03194
9	**18,600**	**0,9689**	**0,01969**
10	28,008	0,9536	0,02422
Media	23,508	0,9475	0,02664
Min	18,600	0,9333	0,01969
Max	28,008	0,9689	0,03594
Varianza	8,309649379	0,00018236	0,00002274

Tabla 5.6: Resultados Modelo SBX

Modelo	Tiempo Promedio
Genérico	0min 5seg
Family Competition	13min 46seg
SBX	0min 7seg

Tabla 5.7: Tiempo promedio ejecución

Como bien se observa en las tablas 5.2, 5.2 y 5.3, en un total de 10 iteraciones los modelos Genérico y Family Competition presentan una gran similitud en lo que respecta al MAPE y están por sobre el entregado por el SBX, cuyo valor es cercano a 20. Sin embargo, el modelo Genérico es el que obtiene un mayor valor en esta métrica.

Por otro lado, el tiempo promedio de una ejecución es considerablemente mayor para Family Competition, esto se puede apreciar en la tabla 5.7 y se explica por la estructura del modelo, la cual es descrita en la sección 4.7.

Si bien el modelo Family Competition demora un tiempo mayor por cada iteración, la velocidad de convergencia es alrededor de un 50 % mayor respecto de los modelos restantes. A continuación se presentan los gráficos de estimación de los modelos.

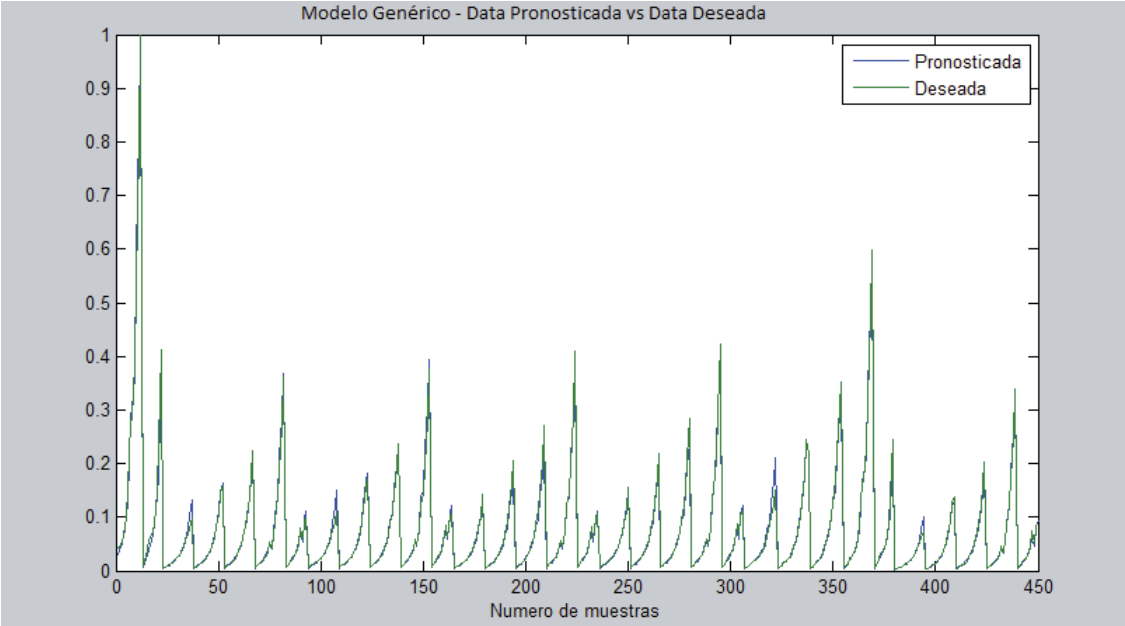


Figura 5.1: Mejor estimación Modelo Genérico

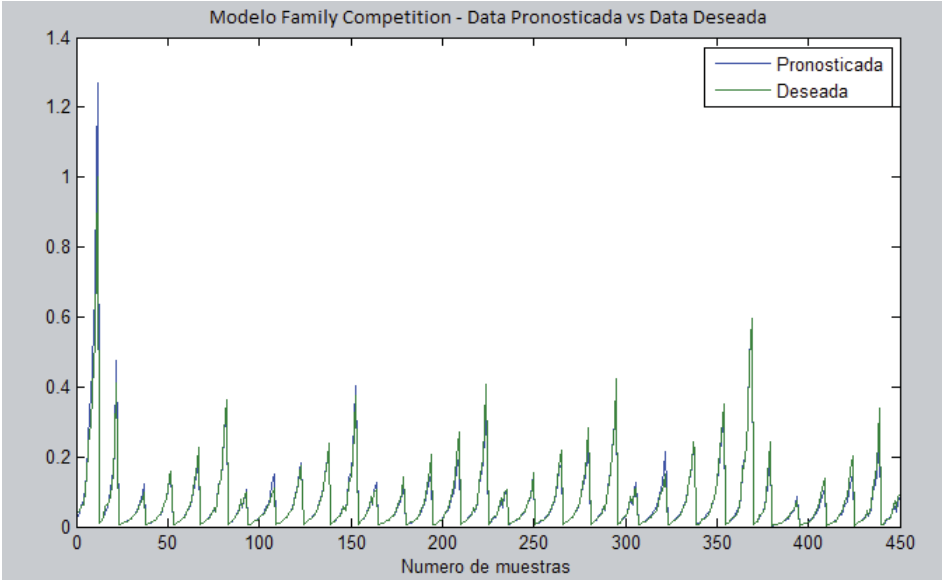


Figura 5.2: Mejor estimación Modelo Family Competition

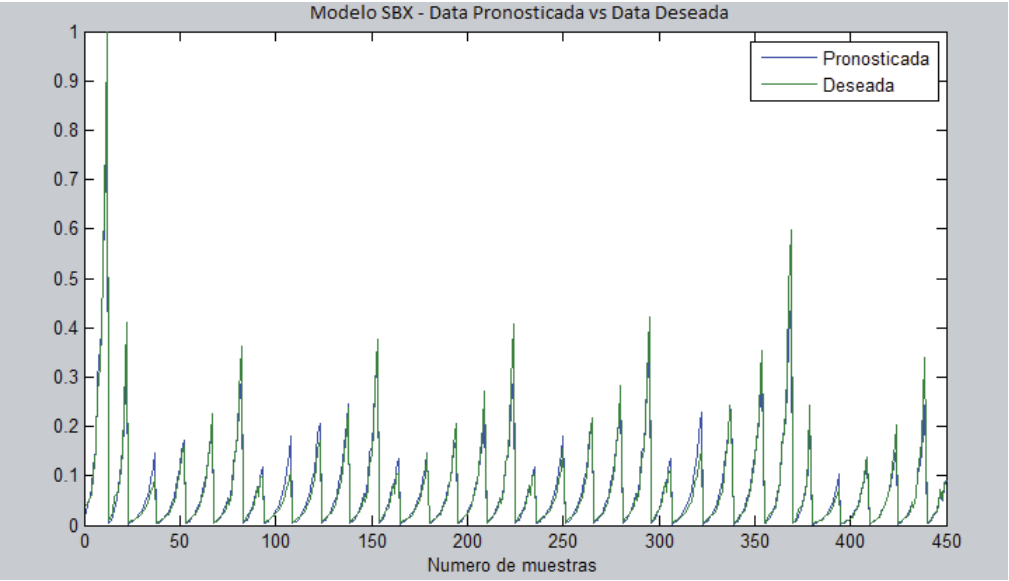


Figura 5.3: Mejor estimación Modelo SBX

Al comparar las tres figuras anteriores, se comprueba visualmente que el modelo Genérico, al presentar un menor error cuadrático medio y un mayor coeficiente de determinación, logrando que la data pronosticada se superponga con la data observada en muchos más puntos que los otros modelos, siendo el modelo Family Competition el segundo que entrega un pronóstico más cercano a los datos observados.

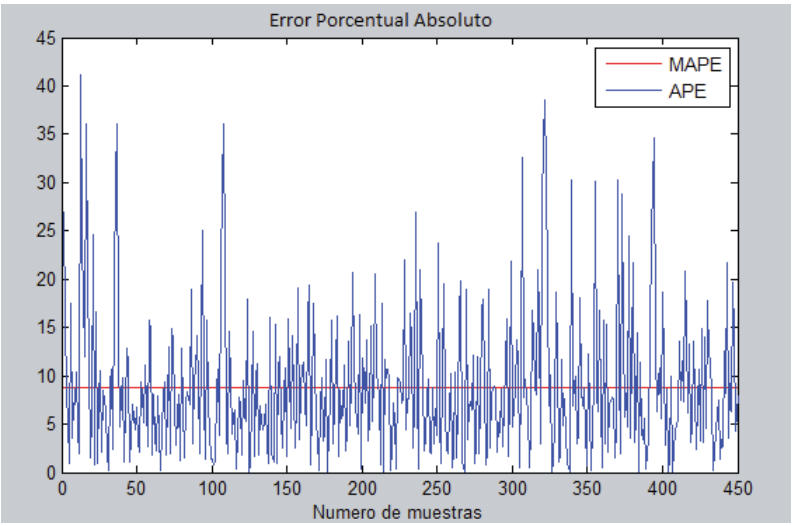


Figura 5.4: MAPE Modelo Genérico

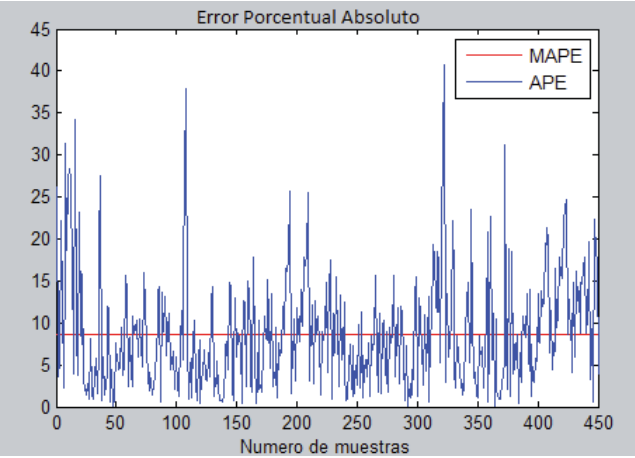


Figura 5.5: MAPE Modelo Family Competition

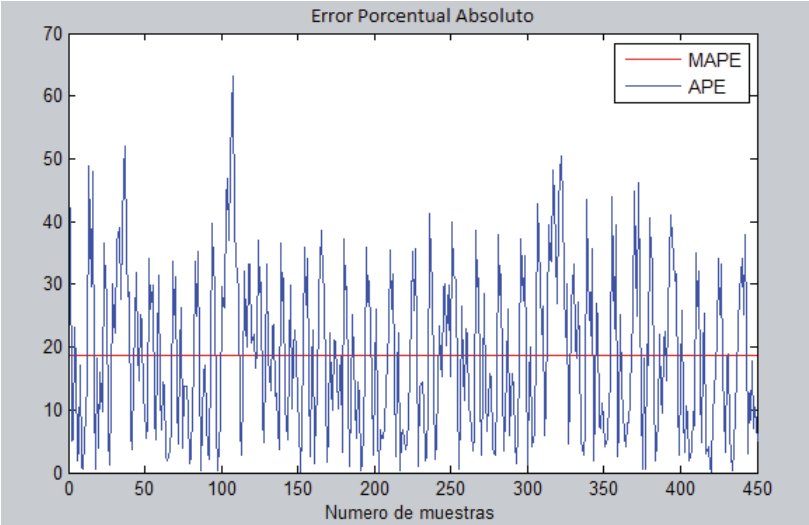


Figura 5.6: MAPE Modelo SBX

Los gráficas presentados en las figuras 5.4, 5.5 y 5.6 corresponden a los valores del MAPE y del APE. Un alto valor del MAPE implica un alto error de pronóstico comparado con los valores observados. Además, se puede apreciar que los picos que presentan los datos observados llevan a una distorsión en el pronóstico en los puntos inmediatamente siguientes, puesto que viene un valle en los datos observados causando que el APE se torne caótico en estos tramos.

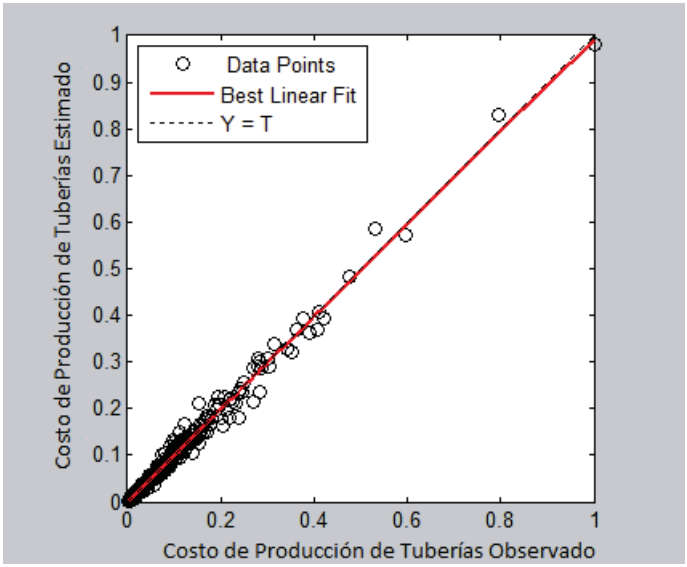


Figura 5.7: Correlación Modelo Genérico

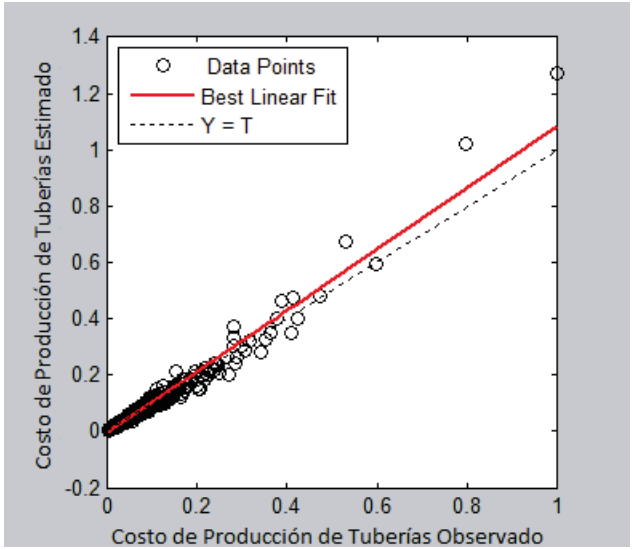


Figura 5.8: Correlación Modelo Family Competition

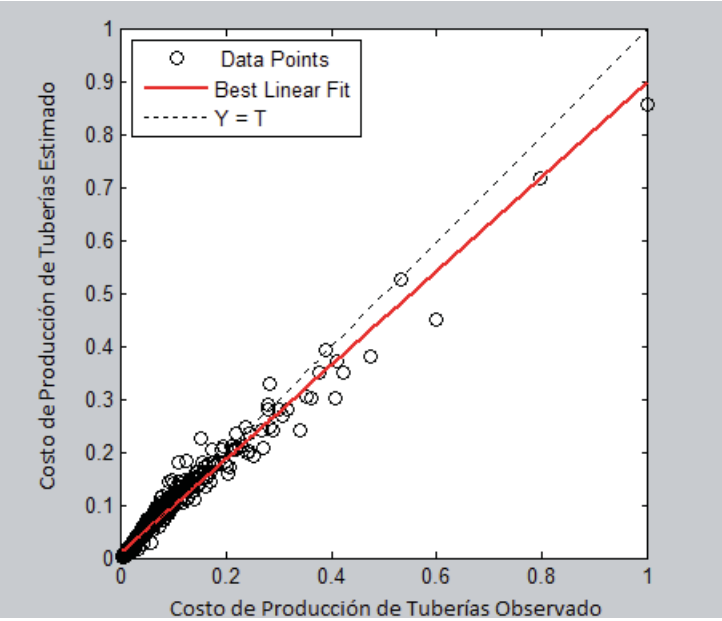


Figura 5.9: Correlación Modelo SBX

Los gráficos correspondientes a las figuras 5.7, 5.8 y 5.9 presentan la correlación existente entre la data pronosticada y la observada. Aquí, el modelo Genérico refleja un R^2 del 98,93 % de la varianza explicada presentando los datos pronosticados más apegados a la recta que los demás modelos, sin embargo los tres modelos presentan resultados similares en cuanto a esta métrica se refiere. A modo de explicación, los otros modelos, al entregar pronósticos con un grado de dispersión mayor que el modelo Genérico, da a entender que los datos que éstos pronostican no varían tan acorde con la data real observada como lo hace éste último.

Luego de contrastar los resultados entregados por cada uno de los modelos, según las topologías encontradas y descritas en el Capítulo 4, se determina que, si bien el modelo el modelo Family Competition y Genérico presentan valores de MAPE cercanos, la media obtenida para esta métrica es menor en el modelo Genérico. Además, la dispersión de los resultados obtenidos a través de las 10 iteraciones de validación muestra que el modelo Genérico presenta una varianza de 0,34 frenta al 0,73 que es entregado por el modelo Family Competition. Debido a lo anterior, se determina que el modelo que obtiene un mejor desempeño frente al pronóstico del costo de producir elementos de tuberías, en esta investigación, es el modelo Genérico.

Capítulo 6

Conclusiones

El mundo actual, que es globalmente competitivo, en donde el margen de beneficios es decreciente y las cuotas de mercado disminuyen rápidamente, es que se hace indispensable estimar los costos asociados a los productos elaborados para ser competitivos y mantener altos niveles de calidad. Es por ello que se utilizan e investigan diversas técnicas para estos fines y una de las técnicas más recientes, presentes en la literatura, son las redes neuroevolutivas. En la presente investigación se desarrolaron tres modelos neuroevolutivos, que se componen de una red neuronal polinomial en conjunto con variantes del algoritmo genético simple.

En el Capítulo 2 se describieron los fundamentos de las Redes Neuronales. El estudio de sus distintas topologías, y condiciones internas, junto con las aplicaciones encontradas en la literatura, han dejado claro que las RNAs corresponden a un modelo computacional capaz de resolver una infinidad de problemas, y que dependiendo de lo que se desee resolver se debe configurar una red que satisfaga dichos requerimientos. Luego, en el Capítulo 3, se describieron los fundamentos de los Algoritmos Genéticos, sus principales elementos y se presentó su algoritmo más simple. En el Capítulo 4 se describe las características de los modelos desarrollados, así como la forma de determinar su topología y sus parámetros internos. Finalmente, en el Capítulo 5 se muestran los resultados obtenidos con datos destinados para la validación de los modelos.

Como se puede notar en el Capítulo 5, el modelo Genérico obtiene los mejores resultados frente a sus rivales para la métrica MAPE, logrando una mejora del 0,36 % frente al modelo Family Competition y de un 26,90 % frente al modelo SBX. En cuanto al coeficiente de determinación el ganador continúa siendo el mismo modelo, logrando una mejora del 0,73 % y del 0,70 % versus el Family Competition y SBX respectivamente.

En base a lo mencionado en el párrafo anterior, se puede concluir que el modelo Genérico con una topología 4,3,1 presenta un mejor desempeño en la estimación del costo de producir elementos de tuberías. Esto, tanto para los resultados obtenidos como en el tiempo de entrenamiento. Por otra parte, el estudio de las métricas residuales dió un mayor entendimiento en cuanto a lo que el modelo arrojó, ya que permitió medir de forma

cuantitativa los resultados obtenidos.

Una vez analizados los resultados, se puede inferir que las variantes del modelo Genérico propuesto entregan peores resultados que éste, por lo tanto no representan una mejora para la estimación de costos en la fabricación de tuberías de acero y, en caso de utilizar una red neuroevolutiva para estos fines. se debe optar por el modelo Genérico.

Finalmente, cabe destacar que el modelo Genérico obtiene 98,93 % de la varianza explicada y un Error Porcentual Absoluto Medio del 8,81 %. Esto demuestra que el modelo es capaz de reducir la incertidumbre relacionada con la estimación del costo de los elementos de tuberías y sin dejar de lado los demás modelos, que también obtuvieron resultados positivos. Además, cabe señalar, que los modelos perdedores en esta investigación pueden mejorar con la elección de nuevos operadores, ya que pequeños cambios en éstos permite mejorar los resultados logrados utilizando las capacidades exploratorias de los Algoritmos Genéticos.

Referencias

[1] A. Zugarramurdi, M. Parin, and H. Lupin, *Economic engineering applied to the fishery industry*. FAO Fisheries Technical Paper, 1995.

[2] W. Sullivan, E. Wikcs, and J. Luxhoj, *Ingeniería económica de DeGarmo*. Engineering Economy, 2004.

[3] O. Durán, J. Maciel, and N. Rodriguez, "Comparisons between two types of neural networks for manufacturing cost estimation of piping elements," *Expert Systems with Applications*, vol. 39, pp. 7788 – 7795, 2012.

[4] B. Verlinden, J. Duflou, P. Collin, and D. Cattrysse, "Cost estimation for sheet metal part using multiple regression and artificial neural networks: A case study," *International journal of production economics*, vol. 111, pp. 484–492, 2007.

[5] S. Rajasekaran and T. Kannadasan, "Optimization of shell and tube heat exchangers using modified genetic algorithm," *International Journal of Control and Automation*, vol. 3, 2010.

[6] F. H'Mida, P. Martin, and F. Vernadat, "Cost estimation in mechanical produccion: The cost entity approach applied to integrated product ingineering," *International journal of production economics*, vol. 103, pp. 17–35, 2006.

[7] J. Matich, *Redes Neuronales: Conceptos Básicos y Aplicaciones*. Universidad Técnologica Nacional, 2001.

[8] D. Anderson and G. McNeill, *Artificial Neural Networks Technology*. Rome Laboratory, 1992.

[9] V. Isasi and L. Galván, *Redes Neuronales Artificiales: Un enfoque práctico*. Pearson Educación - Prentice Hall, Septiembre 2004.

[10] P. Viñuela and I. León, *Redes de neuronas artificiales: un enfoque práctico*. Pearson Educación, 2004.

[11] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 1994.

[12] Y.-C. Chang and M.-C. Chiu, "Numerical optimization of single-chamber mufflers using neural networks and genetic algorithm," *Turkish J. Eng. Env. Sci.*, vol. 32, pp. 313–322, 2008.

[13] S. Dehury and S.-B. Cho, "Multi-criterion pareto based particle swarm optimized polynomial neural network for classification: A review and state-of-the-art," *Computer Science Review*, vol. 3, pp. 19–40, 2009.

[14] A. Patrikar and J. Provence, "Nonlinear system identification and adaptive control using polynomial networks," *Math. Comput. Modeling*, vol. 123, pp. 159–173, 1996.

[15] L. Jin, N. Homma, and M. Gupta, *Static and Dynamic Neural Networks*. Jhon Wiley & Sons, 2003.

[16] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The University of Michigan Press, Ann Arbor, 1975.

[17] C. Darwin, *On the Origin of Species by Means of Natural Selection*. John Murray, 1859.

[18] J. J. Grefenstette, "Genetic algorithms for changing enviroments," *Paper presented at the Parallel Problem Solving from Nature, Bruselas, Bélgica.*, 1992.

[19] M. Mitchell, "An introduction to genetic algorithms," *MIT Press*, 1999.

[20] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning,*. Addison-Wesley, Reading, MA., 1989.

[21] A. Wright, *Genetic algorithms for real parameter optimization.* Foundations of Genetic Algorithms, 1991.

[22] E. Cantú-Paz, "Selection intensity in genetic algorithms with generation gaps," *Paper presented at the Genetic and Evolutionary Computation Conference, Las Vegas, Nevada, USA.*, 2000.

[23] T. Baeck, D. B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. 1997.

[24] K. A. De Jong, "An analysis of the behavior of a class of adaptative systems," Master's thesis, University of Michigan, 1975.

[25] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1992.

[26] L. Eshelman and J. Schaffer, "Real-coded genetic algorithms and interval-schemata," *In Foundation of Genetic Algorithms 2*, pp. 187–202, 1993.

[27] K. Deb and R. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, pp. 115–148, 1995.

[28] S. Mahfoud and G. Mani, "Financial forecasting using genetic algorithms," *Applied Artificial Intelligence*, 1996.

[29] E. Williams, W. Crossley, and T. Lang, "Average and maximum revisit time trade studies for satellite constellations using a multiobjective genetic algorithm," *Journal of the Astronautical Sciences*, 2001.

[30] X. YAO, "Evolving artificial neural networks," *IEEE*, vol. 87, pp. 1423 – 1447, Sep 1999.

[31] J. E. Hanke and A. G. Reitsch, *Business forecasting*. Englewood Cliffs, NJ : Prentice Hall, 5th ed ed., 1995.

[32] L. Bertona, "Entrenamiento de redes neuronales basado en algoritmos evolutivos." Facultad de Ingeniería, Universidad de Buenos Aires, Nov 2005.