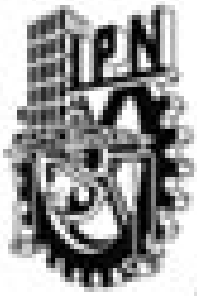


Instituto Politécnico Nacional

Centro de Investigación en Computación

Predicción de Series de Tiempo con Descomposición Empirica en Modos, Algoritmo Genético y Redes Neuronales Artificiales.

Tesis que para obtener el grado de:

Maestro en Ciencias de la Computación

Presenta el:

Ing. Victor Manuel Landassuri Moreno

Director de tesis

Dr. Jesús Guillermo Figueroa Nazuno

México D. F., Mayo 2006

Índice

Lista de Figuras

Capítulo III.

Capítulo IV.

Capítulo V.

Capítulo VI.

Capítulo VII.

Capítulo IX.

Lista de Tablas

Capítulo VII.

Capítulo IX.

Predicción de Series de Tiempo con Descomposición Empírica en Modos, Algoritmo Genético y Redes Neuronales Artificiales.

Resumen

En este trabajo se presenta la predicción de Series de Tiempo usando una técnica de análisis de señales llamada Descomposición Empírica en Modos (EMD), la cual proporciona más información del problema, para poder obtener una predicción más precisa. Se utiliza un algoritmo llamado GANN para diseñar arquitecturas de Redes Neuronales Artificiales (ANNs) utilizando Algoritmo Genético (GA). Las encargadas de realizar la predicción son las ANNs y su entrenamiento es realizado con el algoritmo de Levenberg-Marquardt, el cual está considerado como el más eficiente, cuando las Redes son de tamaño moderado (no sobrepasan unos cuantos cientos de pesos), además, está catalogado como un algoritmo de segundo orden, sin tener que calcular la segunda derivada. La evaluación de la predicción es llevada a cabo con: Raíz Cuadrada del Error Cuadrático Medio (RMSE) y Raíz Cuadrada del Error Cuadrático Normalizado (NRMS), usando predicción iterada y directa. Los resultados muestran que el usar Descomposición Empírica en Modos, puede ayudar a las Redes Neuronales a obtener predicciones más precisas, para la mayoría de las Series de Tiempo caóticas y complejas; también se puede determinar, que el algoritmo GANN es eficiente al automatizar la búsqueda de dichas Redes Neuronales, permitiendo predicciones muy precisas en diversas Series de Tiempo.

Time Series Forecasting with Empirical Mode Decomposition, Genetic Algorithm and Artificial Neural Networks

Abstract

In this work the Time Series forecasting is presented by using a signal analysis technique called Empirical Mode Decomposition (EMD), which provides more information about the problem, to be able to obtain more accurate forecasting. An algorithm called GANN is used to design Artificial Neural Networks (ANNs) architectures using Genetic Algorithm (GA). The forecasting is charged to ANNs and its training is made with the Levenberg-Marquardt algorithm, which is considered like the most efficient, when the Network's size is moderate (they do not exceed a little hundreds of weights), moreover, its cataloged like a second order algorithm, without having to calculate the second derived. The performance evaluation is made with: Root Mean Square Error (RMSE) and Normalized Root Mean Square Error (NRMS), using iterated and direct forecasting. The result shows that using Empirical Mode Decomposition can help to the Neural Networks to obtain a more accurate forecasting, for the most chaotic and complex Time Series; also it is possible to determine, that GANN algorithm is efficient to automate the search of such Artificial Networks, allowing an accurate forecasting of diverse Time Series.

Capítulo I. Introducción.

El poder conocer la más mínima información del futuro, para predecir un fenómeno atmosférico, o saber si las acciones de una empresa van a subir o bajar, es uno de los retos que el humano se ha puesto con el fin de poder tomar decisiones rápidas que lo beneficien; estas son algunas de la razones por la cual la Predicción de Series de Tiempo (ST) es tan importante; pero, ¿qué tan fácil es predecir una ST?

Sea f(x) la ST del Seno, suponiendo que no conocemos nada referente a su dinámica y pensando en querer conocer cuál será el siguiente valor de la Serie, es decir x_{n+1}, donde n es el número de datos; lo único que se tiene que hacer, es observar los últimos m puntos de la Serie, retroceder en ella un poco más de un periodo y encontrar el mismo conjunto de datos observados, y así poder predecir el siguiente valor, basándonos en dicha observación. Este es el caso más simple que se tiene de una serie periódica y estacionaria; ahora, para una ST más compleja, el procedimiento es casi el mismo, nada más que cuesta más trabajo encontrar los patrones o tendencias debido al comportamiento de la Serie, pero no sólo eso es suficiente, sino se debe tener alguna forma de poder hacer una generalización, para no tener que estudiar todos los casos en los que puede caer el fenómeno estudiado.

Las series empleadas en este trabajo son de muy diversos orígenes y tipos, se puede considerar que la mayoría de éstas son una muestra representativa de series difíciles de predecir, es decir, son series complejas y caóticas, en las que cuesta demasiado trabajo poder obtener sus patrones o tendencias, para poder hacer una predicción bastante precisa. También se tienen series con una dinámica no tan complicada, de esta forma, se trata de abarcar un mayor rango de ST (en cuanto a su dinámica). Este es el motivo por el cual se usaron dichas series (listadas en el capítulo VII. Resultados). En el capítulo de Resultados también podemos apreciar la predicción de la Serie Seno, la cual no es incluida en todo el conjunto final, pero se utilizó para probar el algoritmo con la serie más sencilla que se podía tener, es decir, una serie periódica, estacionaria y lineal. Así tenemos desde una Serie Seno, que no representa ningún problema para predecirla, hasta la Serie Ruido Blanco, la cual es una Serie Estocástica, conformada por ruido aleatorio uniforme, considerada entre las más difíciles de predecir por la información contenida en ella.

Como se puede apreciar del Resumen, aquí se ocupará la Descomposición Empírica en Modos para tratar de ayudar a una Red Neuronal a obtener una predicción más precisa, donde ayudar significa en este caso, reducir el error de la predicción.

Antes de continuar, es necesario mencionar algunos términos que se utilizarán a lo largo de trabajo, por ejemplo, cuando nos referimos al aprendizaje de una Red Neuronal, hacemos referencia a encontrar un conjunto de pesos apropiados. También ocupamos los términos como Red, Red Neuronal o ANN como equivalentes, así mismo, se emplea de igual manera el término Señal o Serie de Tiempo; a continuación mencionaremos una definición de cada uno.

Se entiende por Serie de Tiempo al conjunto de datos numéricos que se obtienen en períodos regulares a través del tiempo para un fenómeno. Por otro lado, una definición de señal sería: la energía transmitida detectable, que puede ser usada para transportar información. Pero cuando trabajamos con señales en las computadoras, es necesario hacer un muestreo y cuantificar con valores, ya que las computadoras no pueden trabajar en el dominio continuo. Es por esto que podemos verlos semejantes (señal y ST), hablando en términos computacionales, ya que ambos tienen datos discretos de algún fenómeno, cabe mencionar que las ST, también pueden obtenerse a partir de fenómenos artificiales, como podría ser el caso de un sistema de ecuaciones diferenciales.

1.1 Planteamiento del Problema.

Teniendo en cuenta que el poder predecir un fenómeno puede ayudarnos en diferentes aspectos, es de gran importancia poder hacerlo con una mayor exactitud de lo que actualmente se hace [1]-[11], por lo que nos vemos obligados a buscar diferentes métodos, algoritmos, técnicas e incluso la combinación de estos, para tratar de mejorar lo ya existente. El camino que se siguió en este trabajo y el cual está sujeto a comprobación,

es el de introducir más información a nuestro modelo para poder obtener una predicción más precisa, pero no se trata de información externa a la Serie de Tiempo, como podrían ser el caso de variables relacionadas con esta, sino información obtenida de la misma Serie, mediante la descomposición por componentes.

De acuerdo a esto, necesitamos dos metodologías como primer paso para realizar la predicción, la primera es tener una técnica de análisis de señales que permita la descomposición de una señal o serie; la segunda, es contar con un modelo que permita realizar predicción de Series de Tiempo.

Existen varios métodos de análisis de señales, como podría ser el análisis de Fourier, donde es necesario que los datos sean periódicos o estacionarios, esto limita al análisis de Fourier para ocupar las descomposiciones como entradas a las Redes para predecir, debido a que muy pocas ST podrían presentar dicho comportamiento; se tiene también el método de descomposición por ondeletas (wavelets), el cual ya ha sido aplicado a la predicción [4, 5, 10, 12] introduciendo las descomposiciones obtenidas, a las Redes Neuronales para realizar la predicción; otra técnica reciente de análisis de señales es la Descomposición Empírica en Modos (EMD) [13], la cual nunca se ha usado como ondeletas para la predicción, pero resulta ser un método adaptativo y altamente eficiente en la descomposición de una señal 1-dimensional, además de que no tiene la limitante que presenta el análisis de Fourier.

Ahora se tienen varias opciones en cuanto a la técnica o modelo de predicción, de entre las cuales podemos mencionar las Redes Neuronales Artificiales (ANNs) usadas en gran parte para realizar la predicción, las Máquinas de Soporte Vectorial (SVM) o Falsos Vecinos Cercanos (FVC) donde la predicción se realiza en el Espacio de Fase, entre otros métodos.

De acuerdo al título del trabajo, nos podemos dar cuenta que se utilizó EMD y Redes Neuronales, pero al ocupar ANNs como aproximadores de nuestra función de predicción, necesitamos otra herramienta que nos permita diseñarlas automáticamente, esto, debido a que trataremos de probar si el usar más información obtenida de EMD ayuda (reduce el error) en este caso a una ANN a realizar una predicción más precisa, lo que implica que se necesita probar la idea con más de una ST, por lo que es necesario construir una Red Neuronal para cada problema en particular, ya que una misma Red no puede ser ocupada para diversos problemas según el teorema del No Free Lunch (NFL) [14, 15], ya que ésta no tendría un desempeño óptimo para todos los casos, lo que nos da como resultado utilizar un método de optimización que nos ayude a encontrar las arquitecturas de dichas Redes. El método de optimización utilizado es el Algoritmo Genético (GA), el cual será el encargado de encontrar arquitecturas adecuadas de ANNs para realizar la predicción [1, 16].

Esto nos sirve como guía y nos da una idea de cómo vamos a resolver nuestro problema; falta mencionar que se utilizarán dos formas de realizar predicción Multi-paso mencionadas en [17], así como dos métodos de evaluación (RMSE y NRMS) observando el desempeño del algoritmo llamado GANN compuesto por GA y ANNs.

1.2 Justificación de los algoritmos y técnicas empleadas.

La razón por la que se ocupó a la Redes Neuronales y no a otro modelo de predicción (SVM, FVC, etc.), es porque las Redes Neuronales Artificiales están catalogadas como aproximadores universales [3], lo que significa en teoría, que una ANN es capaz de aproximar cualquier función deseada, siempre y cuando se pueda encontrar una arquitectura y un entrenamiento adecuado. Otro punto importante, es que son muy buenas generalizando, lo que implica que el conjunto de prueba no tiene que ser muy grande para poder obtener resultados adecuados, aunque dependiendo de la complejidad del problema esto puede variar drásticamente; también se conoce que el uso de Redes Neuronales multi-capa suele ser de gran utilidad en la predicción [3], lo más importante de ellas es su adaptabilidad, su no-linealidad y la habilidad de tener una función arbitraria de mapeo.

Aparte de todo esto, se cuenta con una gran cantidad y diversidad de trabajos de predicción con ANNs, esto permite su entendimiento e implementación de una forma más clara, también da la oportunidad de comparar los resultados. Para trabajos que realizan la predicción con otras técnicas, también es posible comparar los resultados, siempre y cuando la predicción sea exactamente con la misma serie, por ejemplo,

hay trabajos donde se hace la predicción de la Serie manchas solares “mensuales”, otros trabajos realizan la predicción de manchas solares “anuales”, aunque es la misma Serie de Tiempo, está muestreada de diferente forma, por lo que no es posible comparar los resultados de ambas predicciones. Así en la literatura se pueden encontrar varias ST clásicas empleadas para probar las técnicas de predicción, como son: Mackey-Glass, Sunspot y Lorenz entre otras.

Es muy difícil entender el mapeo de entradas-salidas realizado por las Redes Neuronales, lo que viene a significar una desventaja para éstas, por esa razón las Redes Neuronales se consideran una “caja negra” donde se conoce perfectamente los datos empleados, pero no se puede apreciar el mapeo no-lineal interno que estas realizan.

Se decidió utilizar el Algoritmo Genético (GA), debido a que es una heurística de búsqueda basada en la evolución natural que permite acercarnos a los mínimos globales de una forma confiable aunque no siempre segura, esto es que el GA no garantiza llegar a un resultado óptimo, pero nos ayuda en gran medida a obtener topologías o estructuras de Redes Neuronales automáticamente, que de otra forma, tendría que pasar por un procedimiento de prueba y error manual (por el especialista), el cual puede ser extremadamente intensivo en tiempo y costo para el ser humano, además de que se tiene que contar con un conocimiento previo del problema y la forma de empezar a resolverlo, lo que puede tomar de semanas a meses poder encontrar una arquitectura de Red lo suficientemente precisa; cabe mencionar, que aquí la Red Neuronal encontrada, únicamente nos serviría para un sólo fenómeno y se tendría que repetir lo mismo para los siguientes *n* casos. La ventaja que se tiene con el GA, es que el trabajo más pesado, que es el de prueba y error es automatizado por él.

Ahora hay que aplicar un método de análisis de señales, el cual pueda descomponer la ST y nos proporcione más información para mejorar la predicción. Pero no cualquier método nos es útil como se muestra en [13], para aplicar el análisis del Espectro de Fourier, el sistema debe ser lineal y los datos periódicos o estacionarios, lo cual no aplica a todas las ST del trabajo, ni tampoco a la mayoría de Series de Tiempo Naturales o Artificiales. La siguiente técnica es la descomposición con ondeletas, pero es necesario la selección de una ondeleta madre, con lo cual se está predisponiendo el filtrado, además de ser de naturaleza no-adaptativa, también se tiene que determinar el número de descomposiciones. Una desventaja más de las ondeletas, es que entre más bajemos en niveles (descomposiciones), más pequeñas en longitud van quedando, para ser más específicos, en cada nivel se reduce a la mitad de muestras la descomposición, lo que implica de cierto modo, redimensionar las descomposiciones al mismo tamaño (up-sampling).

Este es el motivo por el cual se usó este nuevo método llamado EMD, ya que es capaz de identificar oscilaciones intrínsecas dentro de la ST. La descomposición de una ST 1-dimensional con EMD nos da como resultado sus Funciones de Modo Intrínseco (IMF), las cuales son introducidas al GANN, para contribuir a encontrar una arquitectura adecuada de ANN y poder realizar una predicción más precisa. Se puede decir, que EMD nos proporciona diferentes perspectivas de la serie, con sus descomposiciones, dándonos tiempo y frecuencia al mismo tiempo.

1.3 Objetivos.

Objetivo general:

- **Mejorar la predicción de Series de Tiempo utilizando Descomposición Empírica en Modos.**
- **Automatizar la obtención de las arquitecturas de Redes Neuronales (Diseño automático de Redes Neuronales).**

Objetivos derivados del problema principal:

- **Implementar un algoritmo GANN, que permita predecir varias Series de Tiempo para probar si realmente ayuda introducir más información con EMD.**

- Utilizando el procedimiento de ventana corrediza, para estudiar toda la historia de la serie y así utilizar el GA, para optimizar la arquitectura de la Red Neuronal.

- **Estudiar que componentes (IMFs), pueden contribuir a obtener una predicción precisa.**
- **Realizar los experimentos con dos métodos diferentes de predicción y dos métricas de evaluación.**

1.4 Contribuciones.

Las contribuciones del presenta trabajo son la siguiente:

- **Se utilizó la Descomposición Empírica en Modos (EMD) para realizar la predicción de Series de Tiempo, lo cual nunca se había realizado como método experimental en la literatura.**

- **Se tiene un compendio de predicciones para 34 Series de Tiempo, con dos métodos diferentes de realizar la predicción y dos métricas distintas de evaluación.**

- **Implementación de un algoritmo compuesto por GA y ANNs (GANN) para el diseño de arquitecturas de estas últimas con EMD y/o los datos originales.**

- **Estudio experimental de las Funciones de Modo Intrínseco que son útiles para mejorar la predicción.**

1.5 Alcance de la tesis.

Para probar el algoritmo se emplearon 34 ST de diversa naturaleza, los experimentos se realizaron en tres etapas, dando un panorama más general al lector del problema de predicción, además de mostrar la forma en que se fueron resolviendo algunos inconvenientes presentados en el desarrollo del algoritmo; esto se puede ver como la evolución del algoritmo.

Límites de la tesis:

- Aplicar otras técnicas de análisis de señales para realizar predicción.
- Usar otros modelos diferentes de predicción.
- Predicción con ST multivariadas.
- Estudio y análisis de parámetros propios de las ST para realizar la predicción (entropía, correlación, etc.).
- Utilizar diversos algoritmos de entrenamiento.
- Probar y analizar diferentes algoritmos de optimización.
- Ocupar un método estadístico para determinar el porcentaje de exactitud en la predicción del algoritmo.
- Comparar los resultados obtenidos, entre los dos tipos de errores empleados, para determinar la mejor predicción.

1.6 Organización.

Este trabajo de tesis está organizado de la siguiente manera:

En el capítulo II, se comentan los antecedentes de la predicción de Series de Tiempo con Redes Neuronales y algunas otras técnicas con Algoritmo Genético para la evolución de las diferentes partes de una Red y también se tratan algunas técnicas de análisis de señales empleadas para predecir. Es de suma

importancia el teorema del No Free Lunch para conocer el alcance de los algoritmos, por lo que al final del capítulo se presenta una discusión del mismo.

El Capítulo III, habla sobre las Redes Neuronales, construcción, entrenamiento, etc. Aquí podemos encontrar información suficiente para comprenderlas y saber cómo utilizarlas para la predicción de Series de Tiempo, al final del capítulo podemos encontrar la forma de hacer predicción con el modelo de regresión, el cual es el usado para introducir información a las Redes Neuronales.

De igual forma el capítulo IV abordará los puntos más importante a considerar, en la construcción de un Algoritmo Genético.

La descripción de la Descomposición Empírica en Modos se presenta en el Capítulo V. Aquí se muestra el procedimiento para obtener las Funciones de Modo Intrínseco (IMF) de una Señal 1-dimensional, así como la expresión utilizada para determinar qué porcentaje de componentes son introducidas a las Redes Neuronales para realizar la predicción.

En el Capítulo VI abordaremos la implementación del algoritmo GANN entre Redes Neuronales Artificiales y Algoritmo Genético. Se describirá la forma en la que el GA se utiliza para hacer evolucionar las arquitecturas de las Redes, así como se mencionarán todas las variables implicadas en el procedimiento y el tamaño del espacio de búsqueda.

Posteriormente en el capítulo VII se presenta una descripción del método experimental, mostrando las diversas etapas del algoritmo. Esto con el propósito de mostrar la evolución del mismo y las configuraciones que no presentaron resultados adecuados o precisos, por lo que se presenta una discusión en cada etapa y se proponen modificaciones, con las cuales el algoritmo se vuelve un poco más preciso para las diferentes series utilizadas. Las predicciones presentadas en las diversas etapas van desde predecir un punto adelante, hasta los 240 puntos, es importante mencionar que la series en las que se pudo extender la predicción a más de 30 puntos, son series no muy complicadas de predecir, en donde los mejores resultados fueron obtenidos de los experimentos sin introducir IMF, es decir, sin introducir más información a la Red Neuronal con EMD.

En el Capítulo VIII presentamos las conclusiones obtenidas de acuerdo los resultados, así como las líneas de investigaciones futuras derivadas del trabajo, que sirvan para mejorar aún más el algoritmo.

En al capítulo IX Anexos, se presentan los trabajos derivados de la tesis, tablas en extenso del capítulo VII y algunas graficas que no fueron incluidas en dicho capítulo y también se presenta una descripción de las ST empleadas, donde se pueden observar las graficas como información concerniente a cada una de ellas. Contiene el glosario con algunos términos importantes al final del capítulo.

Capítulo II. Antecedentes.

2.1 Estado del Arte en Predicción de Series de Tiempo.

Este capítulo está dividido en dos partes, la primera está relacionada con el Estado del Arte en predicción de Series de Tiempo, la cual a su vez, está subdividido en predicción de ST, algoritmos que evolucionan Redes Neuronales y técnicas de análisis de señales empleadas para la predicción. La segunda parte está dedicada al teorema del No Free Lunch, donde se discutirá brevemente el mismo; este teorema es de suma importancia ya que nos da una idea de la capacidad que tiene un algoritmo de búsqueda u optimización para dar una solución óptima, por lo que lo comentaremos para tener conocimiento de ello.

2.1.1 Predicción de Series de Tiempo.

No es nueva la idea de realizar predicción de Series de Tiempo con Redes Neuronales Artificiales, esto se remonta alrededor de 1960 [3] donde investigadores y científicos empezaban a darse cuenta de todas las ventajas que podía otorgar una Red Neuronal para realizar la predicción, como es la capacidad de generalización y la de poder aprender mediante la experiencia.

Los primeros trabajos en Redes Neuronales fueron realizados por McCulloch y Pitts en 1943 [7, 18], donde buscan la representación computacional de una neurona biológica (se hace una analogía de las neuronas). Al paso del tiempo fue aumentando el conocimiento de las Redes y se fueron desarrollando más técnicas aplicadas a ellas, permitiendo una mejor predicción y superando incluso a los métodos estadísticos basados en la Regresión [3]; de esta forma surgió el algoritmo de Retro-propagación (Back-Propagation) [19] al principio de los 70s, desarrollado por diversos investigadores independientes como Werbos, Parker, Rumelhart, Hinton y Williams.

En la literatura existen muchos y muy diversos trabajos relacionados con la predicción de Series de Tiempo, algunos usan modelos matemáticos o estadísticos para predecir como en [2, 20], otros usan Redes Neuronales [21, 22, 23, 24]. También se pueden encontrar trabajos relacionados con la predicción de ST caóticas [2, 6], económicas / financieras [20, 25] o con diversas series de diferente naturaleza [3].

Se han desarrollado trabajos donde se realiza la predicción de Series de Tiempo a corto plazo con Redes Bayesianas [26], las cuales son extraídas a partir de secuencias discretas alineadas, utilizando algoritmos de aprendizaje, estas permiten detectar relaciones sutiles entre dos Series de Tiempo, en las que se puede realizar la predicción por causa-efecto entre ellas.

Otro ejemplo es en [27], donde se presenta un estudio experimental sobre la predictibilidad de Series de Tiempo, así mismo desarrollan una métrica que permite medir la misma; también investigan la relación de la predictibilidad con respecto al comportamiento dinámico de las ST, determinando una relación no lineal entre ambos; además se detecta que existe una relación entre la estructura de las ST y su predictibilidad, expresado por medio de patrones básicos. Se puede encontrar en dicho trabajo, dos técnicas relativamente nuevas de predicción de ST: GABoost y ComFEC.

En [28] se desarrolla un sistema para la extracción automática de Motifs (patrón temporal altamente representativo); también se obtienen relaciones temporales de ST multivariadas que describen el comportamiento de fenómenos complejos, así estas relaciones ayudan a la obtención de modelos descriptivos y predictivos. Otro trabajo donde se emplean Motifs es en [29], ahí se realiza un estudio de las características más importantes de los sistemas complejos y se desarrolla un entorno de simulación para la modelación de dichos sistemas. Como parte de las conclusiones, se destaca el hecho de que es muy difícil, tratar de determinar si existe un atajo para realizar la predicción al encontrar los Motifs del sistema.

Se pueden encontrar métodos en los que se hace un pre-procesamiento de la información o filtrado antes de realizar la predicción [30], también puede encontrar el trabajo realizado en [21], donde el modelo de predicción con una Red consta de tres fases:

1) Determinar los patrones de entrada con un análisis de autocorrelación. Para lograr esto, primero se calculan los coeficientes de autocorrelación de la ST, si se detecta una tendencia, ésta es removida con una diferenciación, esto se repite hasta un grado razonable y es necesario para determinar los retardos, posteriormente, se calculan coeficientes parciales de autocorrelación, para determinar las entradas de la Red Neuronal.
2) Determinar el número de neuronas ocultas con la regla *Baum-Haussler*, o mejor conocida como la regla de *Thumb* (regla del pulgar), donde se tiene la siguiente formula:

$$N_{hidden} \leq \frac{N_{train} E_{tolerance}}{N_{pts} + N_{output}} \qquad \textbf{(2.1)}$$

   donde N_{hidden} es el número de nodos en la capa oculta, N_{train} es el número de muestras de entrenamiento, $E_{\text{tolerance}}$ es el error de tolerancia, N_{pts} es el número de datos por muestra de entrenamiento y N_{output} es el número de neuronas de salida.
3) Construir la Red Neuronal, teniendo como base los dos puntos anteriores.

Después de tener la Red, ésta se puede entrenar con el algoritmo de Back-Propagation, el que nos sirve para encontrar una configuración adecuada de pesos; pero aún en la actualidad, no existe ningún método establecido que indique un procedimiento, para la construcción de Redes Neuronales que nos garantice un buen funcionamiento, por lo que su diseño puede llevar de días a meses, dependiendo del problema en cuestión. Por suerte existen métodos que nos permiten evitar el procedimiento manual de prueba y error, pero a cambio, se convierten en problemas de cómputo intensivo.

Debido a que es muy complicado encontrar arquitecturas adecuadas de Redes Neuronales [22], se le ha llamado a esta labor, como el “arte negro”. Cabe mencionar que no es imposible determinar, una buena o adecuada estructura para un problema determinado, lo que es imposible, es demostrar que esa arquitectura es óptima [22]. Esto también se cumple, para Redes encontradas con el algoritmo GANN.

Existen varias Series de Tiempo clásicas en la literatura, donde los investigadores ponen un gran esfuerzo e interés por predecirlas [1, 2, 3, 4, 5, 6, 7, 10, 11], algunas de estas series clásicas son: Mackey Glass, Sunspot, Lorenz y algunas Financieras entre otras; en este trabajo se decidió ocuparlas, para tener un punto de referencia en cuanto a predicción.

2.1.2 Algoritmos Evolutivos.

En el pasado, muchos investigadores han usado diversos métodos para poder evolucionar Redes Neuronales, algunos se basan en la idea de dejar fijas las entradas y buscar una arquitectura adecuada [31], otros dejan la arquitectura de Red fija y varían la cantidad de datos de entrada así como la longitud del conjunto de entrenamiento [8, 31]. Este trabajo de tesis, está basado en una combinación de ambas, ya que permite variar los datos de entrada y la arquitectura de la Red. Básicamente, la evolución de Redes Neuronales se puede dividir en tres partes: evolución de pesos, de arquitecturas y de técnicas de aprendizaje [32].

Algunos investigadores usan Algoritmo Genético para encontrar una combinación adecuada de pesos y *bias* (entrenamiento) [21, 33, 34, 35, 36], otros lo usan para diseñar arquitecturas de Redes [1, 22, 34, 35, 36]. En [16] utilizan un algoritmo, al que le llaman TWEANNs (Topology and Weight Evolving Artificial Neural Networks), este es una combinación de las dos formas mencionadas anteriormente, donde evolucionan tanto pesos como arquitecturas, usando un entrenamiento no supervisado y una codificación indirecta.

Al utilizar el algoritmo de Back-Propagation se tiene que especificar, a *priori*, el número y conexiones de las neuronas ocultas antes de iniciar el algoritmo, por lo que la simplificación de las Redes Neuronales se vuelve una tarea de prueba y error. Algunos autores atacan este problema, utilizando un enfoque constructivo en el que se aprende mientras se crece, donde normalmente se empieza con una neurona oculta [6, 16, 37].

Existen diversas formas de evaluar el rendimiento de las Redes, para procedimientos de búsqueda, se le suele llamar función de costo. Esta función es la que se tiene que minimizar [3, 21] y existen varias formas para hacerlo, por ejemplo: MAD (Mean Absolute Deviation), SSE (Sum of Squared Errors), MSE (Mean Squared Error), RMSE (Root Mean Squared Error), MAPE (Mean Absolute Percentage Error) y NRMS o NRMSE (Normalize Root Mean Squared Error). En nuestro caso, en las Redes Neuronales, se optimizarán los pesos con el algoritmo de entrenamiento Levenberg-Marquardt [38]; el GA será el encargado de optimizar la búsqueda de arquitecturas adecuadas de Redes Neuronales, donde la función de Adaptación, evaluará a cada Red generada, para obtener su valor de adaptabilidad.

En [31] se puede encontrar un trabajo extenso en evolución de ANNs; el algoritmo evolutivo empleado recibe el nombre de EANNs (Evolutionary Artificial Neural Networks), donde se utilizan diversos métodos de codificación (Directa: binaria o números reales e indirecta); se realiza la evolución de los pesos y las arquitecturas por separado o ambas al mismo tiempo; también se hace evolucionar a las entradas, determinando las mejores y quitando las que sean redundantes en información. Ahí mismo [31], se pueden encontrar 319 referencias relacionadas con el tema.

También se suelen utilizar los algoritmos evolutivos de Redes para problemas de control como en [39], donde utilizan un algoritmo al cual le llaman SANE (Symbolic Adaptative Neuroevolution). Ahí las neuronas tienen roles independiente o cruzados, volviéndolas cooperativas y robustas para los problemas de control.

Para terminar esta sección de optimización, es importante aclarar que no es lo mismo aprender que optimizar, como se menciona en [32], ya que en una optimización se busca el error más bajo y en un aprendizaje no forzosamente el error más bajo es el que nos da una buena generalización.

2.1.3 Técnicas de análisis de señales.

Casi no existen trabajos relacionados con la predicción que utilicen el análisis de Fourier, esto se puede deber, a que las restricciones que pide Fourier como estacionalidad o linealidad, no las cumplen la mayoría de series a predecir; pero para el caso de ondeletas es totalmente diferente, ya que existe más trabajos para este fin [4, 5, 10], sin embargo algunos autores no se aventuran a realizar predicciones a largo plazo con ondeletas, incluso, realizan predicciones a un sólo paso adelante [4, 5] y presentan gráficas donde dan la apariencia de predecir 100 o más puntos en la Serie de Tiempo, lo cual es falso y simplemente su algoritmo es capaz de predecir un paso adelante. Ellos mismos ponen, en el título de la figura, que se trata de predicciones a un paso, pero pareciera otra la situación de la forma en que lo presentan.

Así mismo se pueden encontrar algoritmos, en los que una Red feed-forward se encarga de predecir cada una de las descomposiciones realizada con ondeletas, para luego revertir el proceso de descomposición y obtener el valor real predicho [4], la desventaja, es que se usa una misma arquitectura de Red Neuronal para predecir las descomposiciones. Aquí se ahorra tiempo en la búsqueda de arquitecturas, pero nada nos asegura que una misma arquitectura nos pueda predecir *n* diferentes series (descomposiciones) de una forma adecuada, esto basado en el teorema del NFL. Al menos, debe de existir una arquitectura adecuada para cada descomposición.

Aunque la Descomposición Empírica en Modos no se ha ocupado para predicción, sí se ha ocupado para hacer grandes avances en cuanto al análisis de señales como se muestra en [40] donde por medio de las IMFs (descomposiciones con EMD), se detecta que existe una alta correlación entre los ciclos de manchas solares y el cuarto modo obtenido, siendo este un ciclo solar de 11 años. Así como la extracción de señales atmosféricas inter-anuales usando EMD [41] y el reconocimiento de la silueta o porte, por medio de las IMFs utilizando Redes Neuronales [42].

2.2 Teorema del No Free Lunch.

El teorema llamado No Free Lunch (NFL) para búsqueda, fue planteado por dos investigadores del Instituto de Santa Fe, Bill Macready y David Wolpert en 1995 [14] al formular la siguiente pregunta: ¿Existen algunos procedimientos de búsqueda que sean ***buenos*** procedimientos de búsqueda, sin importar el tipo de problema?

Para contestar está pregunta se plantea una solución matemática en donde se tiene una gran habitación en tres dimensiones y se divide ésta en pequeños cubos (de pequeño volumen), siendo el número obtenido de cubos bastante grande, por decir trillones. Posteriormente se considera la forma de asignarle un entero a cada cubo, así dicho valor se puede ver como la adaptabilidad de esa posición en el cuarto; después se formaliza un procedimiento para tomar *M* distintas muestras del trillón de cubos en la habitación; el procedimiento debe de especificar cómo tomar esas *M* muestras. Como se puede ver existen muchas y muy diversas formas de hacerlo, puede ser empezando en un cubo y tomar a sus vecinos, o bien de acuerdo a su adaptabilidad, etc.

Así el NFL menciona, que promediado sobre todas las posibles soluciones, no existe un procedimiento de búsqueda, que mejore a otro procedimiento.

Visto de otra forma, se puede decir que todos los algoritmos que buscan una extrema (un máximo o mínimo) de una función de costo, tienen el mismo rendimiento, cuando se promedian sobre todas las posibles funciones de costo. Por lo que se tiene que: si un algoritmo A mejora a un algoritmo B en algunas funciones de costo, entonces existirán otras funciones de costo donde B mejore a A.

En 1997 los mismos autores publican el teorema para problemas de optimización [15] que dice: si un algoritmo tiene un rendimiento elevado sobre alguna clase de problemas, entonces necesariamente pagará el precio con un rendimiento degradado sobre otra clase de problemas. ¿Qué significa esto?, que ningún algoritmo puede ser utilizado para resolver todos los problemas de forma óptima, por eso, existen en la literatura cientos de algoritmos que resuelven una tarea en específico.

Existen muchos y muy diversos trabajos que tratan el NFL, como por ejemplo en [43] donde hacen un estudio del NFL para funciones con distribución uniforme, también se menciona que la única forma de realizar un algoritmo rápido para una clase de funciones, es hacerlo lento en otras (otro punto de vista para el NFL en función del tiempo). Hay quienes abordan el NFL con GA y GP (programación Genética) [44], en donde mencionan que el NFL no es válido para representaciones usadas en GP, debido a la no uniformidad entre la descripción de un objeto y el objeto por si mismo. Otros tratan el NFL para la validación cruzada o Early Stopping [45], aquí se comenta que el error de generalización tiende a incrementarse, cuando se tiene un error de entrenamiento fijo sobre un error de entrenamiento mínimo, por lo que en [46] llegan a la conclusión de que los algoritmos de aprendizaje no son universalmente buenos; ahí también se presenta un estudio del NFL para predicciones con ruido.

Ya sabemos que no hay algoritmos universales, sin embargo, existen diversas herramientas que nos permiten realizar el procedimiento de prueba y error de una forma automática, así tenemos la capacidad de construir Redes Neuronales de acuerdo a cada problema con un Algoritmo Genético. Aunque sabemos que no vamos a poder resolver todos los problemas de una forma óptima (universalidad), sí nos va a permitir probar la idea de ayudar a una Red Neuronal a obtener una predicción más precisa (error más pequeño) si le introducimos más información con EMD. Así el NFL nos da una idea de la capacidad de un algoritmo para dar soluciones óptimas, ya que conocemos esto, el siguiente paso podría ser, el caracterizar qué tan efectiva es la optimización bajo ciertas restricciones razonables [47], lo cual esta fuera del alcance, de este trabajo de tesis.

En este capítulo se expuso la predicción de ST con Redes Neuronales y algoritmos evolutivos que son usados para evolucionar arquitecturas, pesos y técnicas de aprendizaje de dichas Redes, también se hizo referencia a las técnicas de análisis de señales utilizadas para descomponer el fenómeno, utilizando esa información obtenida para realizar la predicción; se comentó el teorema del NFL, el cual es de gran importancia para conocer el alcance de los algoritmos. En el siguiente capítulo se tratará a las Redes Neuronales (construcción, entrenamiento, etc.) junto con la predicción de ST.

Capítulo III. Redes Neuronales Artificiales.

En este capítulo veremos cómo esta conformada una Red Neuronal, cómo podemos construirla, las diferentes funciones de transferencia que existen y la forma de entrenarlas entre otros puntos; posteriormente se comentará la predicción de Series de Tempo así como dos formas diferentes de realizarla, mencionadas en la literatura.

Las Redes Neuronales tratan de hacer una analogía del cerebro humano, pero esto es algo realmente difícil de lograr debido a las grandes limitantes que existían y que existen, así como el poco entendimiento que tenemos del cerebro. Las ANNs se fueron desenvolviendo en diversos temas particulares, donde podían competir contra el ser humano procesando cantidades impresionantes de información, como es la clasificación de patrones, el control y por supuesto la predicción. La parte central de las Red Neuronal es la neurona, la cual se considera como un elemento de procesamiento simple, y al trabajar e interactuar junto con otras, permite aproximar funciones específicas. Las Redes Neuronales son una clase de arquitectura de procesamiento en paralelo.

Una Red Neuronal está catalogada como un aproximador universal [48], lo que significa que una Red Neuronal se puede comportar como cualquier función deseada en teoría, porque lo difícil es encontrar a una Red y un entrenamiento adecuado, además, no es posible determinar si dicha arquitectura es óptima para el problema en cuestión, porque debido a su estructura, puede darse el caso de que exista una mejor Red que nos reduzca dicho error. Así una Red Neuronal realiza un mapeo de un espacio de entrada a uno de salida.

Existen diversos teoremas que muestran que una Red Neuronal, con una sola capa oculta y un número suficiente de neuronas, es suficiente para aproximar cualquier función [3].

Pero la pregunta que se tiene que hacer a continuación es: ¿Cuántas neuronas son suficientes?, la respuesta es: dependiendo de los datos y del problema. Claro que tiene que variar el número de neuronas de acuerdo al problema a tratar, pero ¿hasta qué punto debemos de aumentar las neuronas?, en el peor de los casos es agregar cientos de ellas, lo malo de esto es que la Red puede empezar a decaer en el rendimiento, por lo que es más conveniente agregar más capas ocultas y disminuir drásticamente el número de nodos en la capa oculta, más específicamente se sabe que dos capas ocultas suelen proveer mayores beneficios que una sola [49]. También se tiene el conocimiento de que con dos capas ocultas se puede resolver la mayoría de los problemas, incluyendo la predicción [3, 48].

3.1 La Neurona.

Como ya se comentó anteriormente, la neurona es la parte central de las Redes Neuronales y ésta puede estar en dos tipos de estados, activo y pasivo, y puede interactuar con otras neuronas mediante las sinapsis (o pesos), normalmente denotados con la letra *w*; estos pesos pueden ser modificados de acuerdo a la experiencia de la Red (entrenamiento).

La neurona biológica del ser humano consta de cuatro partes:

1. Dendrita. Es la entrada de la neurona
2. Soma. Es la parte de procesamiento
3. Axon. Cambia de las entradas a las salidas
4. Synapses. La conexión con otras neuronas.

Algunos autores usan dichos términos al referirse a las partes de una neurona computacional [49, 50], en el caso de este trabajo de tesis, no se emplean dichos nombres, únicamente se utiliza: nodos o neuronas, conexión entre nodos, función de transferencia, entradas y salidas.

La neurona está constituida por *n* entradas, las cuales pueden provenir de datos de entrada o bien de otras neuronas, así como de *w* conexiones o pesos y el *bias*.

La función que calcula la neurona es la siguiente:

$$a = f(wp + b) \tag{3.1}$$

donde *a* es la salida de la neurona, *w* es el peso o valor de la conexión que conecta con la neurona, *p* es la entrada a la neurona, *b* es el *bias* y *f* es la función de transferencia. Si tenemos más de una entrada, entonces vamos a tener un peso para cada entrada, por lo que se realiza el producto punto entre el vector de entrada y los pesos de la neurona:

$$a = f(w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b) \tag{3.2}$$

Para abreviar se puede utilizar la siguiente notación:

$$a = f(Wp + b) \tag{3.3}$$

donde *W* es la matriz de pesos, y *p* es equivalentemente el vector de entradas.

El *bias* es establecido a uno y se puede ver como un valor de corrección agregado a la multiplicación realizada en la neurona, o bien, como un parámetro libre extra para poder aproximar mejor nuestra función. Cabe señalar que en los experimentos realizados en el trabajo, puede existir o no el *bias*, lo que nos permite buscar entre otro tipo de soluciones.

Una de las ventajas que nos proporcionan las Redes Neuronales, es que son capaces de darle más peso a las estradas que pueden aportar más información para resolver mejor nuestro problema, mientras que a otras las puede limitar en la medida que sea necesario, esto lo realiza mediante la actualización de pesos en el entrenamiento. Debido a que se realiza el producto punto entre entradas y pesos, estos pueden ser vistos como vectores, entonces si el vector de entradas apunta geométricamente hacia una dirección y el vector de pesos apunta en dirección contraria, el resultado será minimizado, en el caso contrario es maximizado el resultado, así se limita en la medida que sea necesario, la o las entradas correspondientes [51].

Ahora, si se tiene una capa con *R* entradas y *S* neuronas, se tiene la siguiente matriz de pesos:

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} \ldots w_{1,R} \\ w_{2,1} & w_{2,2} \ldots w_{2,R} \\ \vdots & \vdots \quad \vdots \\ w_{S,1} & w_{S,2} \ldots w_{S,R} \end{bmatrix}$$

donde los renglones indican el destino y las columnas indican la fuente.

Como podemos apreciar, las Redes Neuronales son un conjunto de multiplicaciones, sumas, y funciones de transferencias, que nos ayudan a aproximar nuestra función con ayuda del algoritmo de aprendizaje. Por ejemplo, si tenemos tres capas, nuestra función de salida será:

$$a^3 = f^3(W^{3,2} f^2(W^{2,1} f^1(W^{1,1}p + b^1) + b^2) + b^3) \tag{3.4}$$

donde a^3 es la salida de la capa 3 y los superíndices de *W* indican el destino y origen respectivamente de las matrices de pesos.

3.2 Tipos de Redes.

Según el tipo de problema, es el tipo de representación utilizada, para lo cual existen dos grandes grupos:

- Redes feed-forward (no tienen ciclos)
- Redes Recurrentes (tienen ciclos)

Estos dos tipos de topologías pueden ser clasificadas en: multi-capa, únicamente una capa, aleatoriamente conectados, localmente conectados, conexiones dispersas, completamente contados, etc. Para este trabajo se utilizaron Redes multi-capa, completamente conectadas, permitiendo hasta dos capas ocultas sin contar las capas de entradas y salidas, siendo estas feed-forward.

Las Redes multi-capa (Multi-Layer Perceptrons, MLP) son comúnmente usadas para realizar la predicción y debido a que no hay ningún teorema para su construcción, muchos autores usan de 1 capa hasta 3 o 4 para realizar la predicción [3].

3.3 Construcción de la Red.

Existen diversas formas en las que se puede construir una Red Neuronal, éstas dependen de nuestro problema, algunos autores tratan con arquitecturas grandes y van eliminando nodos según sea necesario, otros empiezan de arquitecturas pequeñas y van aumentando los nodos. Se puede emplear algunos teoremas para empezar con la construcción de éstas [21, 51], pero como ya se comentó en el Estado del Arte, no existe ninguna regla que nos permita diseñar adecuadamente una Red para una tarea en específico, así como tampoco nos garantiza que funcione adecuadamente. Lo que si es seguro, es que uno se puede tardar demasiado tiempo buscando una Red Neuronal adecuada que se ajuste a nuestras necesidades para un problema específico.

Aunque no existen reglas para diseñar a una Red Neuronal, se debe de tomar en cuenta el número de capas (capa de entradas, ocultas y de salida), el número de nodos por capa, la interconexión entre estos (feed-forward, feed-back, nodos completamente conectados u otros tipos de conexión), la presencia de *bias*, el tipo de función de transferencia en capas intermedias y de salida; si se normalizaran los datos de entrada, la cantidad de datos de entrada (corresponden directamente a los nodos de entrada), el número de nodos de salida, el tipo entrenamiento (Batch o secuencial), el algoritmo de Entrenamiento, el tamaño del conjunto de entrenamiento, los criterios de evaluación en el entrenamiento y en el conjunto de prueba (MSE, SSE, RMSE, etc.). Algunos de estos términos se describen a lo largo del capítulo y otros se pueden encontrar en el glosario, en la sección 9.4.

Por suerte algunos de estos parámetros ya están bien definidos para la predicción de ST. Por ejemplo, para una Red que se utilice para predecir, esta tiene que tener ser feed-forward, con una o más capas ocultas, la función de transferencia es tangente sigmoidal en las capas intermedias y lineal en la de salida, de preferencia es conveniente realizar una normalización de los datos, aunque la tarea más complicada es determinar el número de datos de entrada, el número de capas y los nodos por cada capa; todos estos parámetros dependen del problema, así como los nodos de salida de la Red Neuronal.

Se sobreentiende que se tienen datos suficientes para tener un conjunto de entrenamiento que nos represente la dinámica del sistema lo más acorde posible, es decir, que se obtendrán resultados muy pobres si queremos hacer predicción de ST con un conjunto de entrenamiento de 100 datos, en cambio, si tenemos alrededor de 1000 datos, de seguro obtendremos mejores resultados, ya que tenemos más información del sistema con lo cual podemos conocer mejor su dinámica.

El número de neuronas en la o las capas ocultas es determinante para resolver el problema, ya que aquí es donde se lleva a acabo el mapeo no-lineal de entrada-salida. El entrenamiento y los resultados dependen del tamaño de la Red Neuronal. Si tenemos una Red muy pequeña, es probable que no podamos aproximar del todo nuestra función deseada y consumiremos demasiadas *épocas* en el entrenamiento, en cambio si la arquitectura es muy grande, podemos tener un sobre entrenamiento de los datos en unas cuantas *épocas*.

El número de neuronas de entradas, depende directamente de las variables del sistema, en el caso de la predicción, el número de neuronas de entrada corresponde al número de datos necesarios para predecir el siguiente valor, y estos pueden variar de acuerdo al problema.

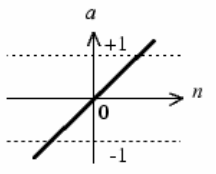
Como podemos apreciar es más difícil de lo que parece el diseño de las Redes Neuronales, pero los beneficios que nos pueden traer, si se diseñan y entrenan adecuadamente son bastantes; como se menciona en [3], el diseño de ANNs es más un arte que una ciencia.

3.4 Función de transferencia.

Las funciones de transferencia también son llamadas como funciones de Activación y estas determinan la relación entre los nodos de entrada y salida. Las funciones de activación pueden ser lineales, sigmoidales, tangente sigmoidal, etc.

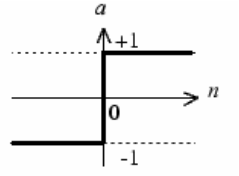
La ventaja de usarlas, es que meten a la Red Neuronal un factor de no-linealidad que resulta de utilidad para la relación entradas-salidas. La función escalón (Hard limiter), ya no se suele usar debido a que no mete dicho factor. Otra forma de verlas es como se menciona en [51], donde el objetivo de usarlas, es el de permitir a la salida de la neurona (sumatoria), variar con respecto al tiempo; en la figura 3-1 se muestran las funciones de transferencias.

Función lineal



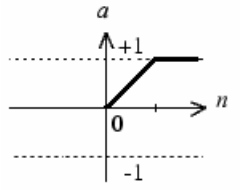
$$funlin(n) = n$$

Escalón



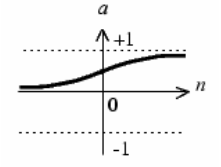
$$n < 0,\ a = -1$$
$$n \geq 0,\ a = 1$$

Rampa (ramping)



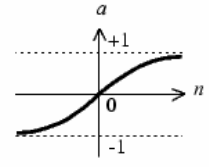
$$n < 0,\ a = 0$$
$$0 \leq n \leq 1,\ a = n$$
$$n > 1,\ a = 1$$

Sigmoide (logistica)



$$\log sig(n) = \frac{1}{(1+\exp(-n))}$$

Tangente Sigmoidal



$$\tan sig(n) = \frac{2}{(1+\exp(-2n))-1}$$

Figura 3-1. Funciones de Transferencia

3.5 Entrenamiento.

La Serie de Tiempo es dividida en dos conjuntos: el conjunto de entrenamiento y el conjunto de prueba. El conjunto de entrenamiento sirve para obtener los pesos de la Red, mientras que el conjunto de prueba sirve para validar la eficacia de la misma. Debido a que se quiere minimizar el error de nuestra función, el entrenamiento se puede llevar a acabo de diferentes formas, como por ejemplo, con el conocido algoritmo Back-Propagation e incluso con Algoritmo Genético [16, 22, 33, 34, 35, 36].

Antes de iniciar el entrenamiento es necesario inicializar todos los pesos de nuestra red, para eso es conveniente inicializarlos con un valor aleatorio pequeño. Existen dos forma de entrenamiento, el supervisado y el no supervisado o adaptativo.

Entrenamiento supervisado: Este tipo de entrenamiento comprende ir revisando la salida de la red, comparando esta con un valor objetivo (target), del cual se extrae un error para poder determinar el rendimiento de la Red Neuronal.

Entrenamiento no supervisado: Aquí se le deja todo el trabajo a la Red, ella tiene que ser capaz de sacar la información necesaria de los datos de entrada para poder realizar la tarea.

Para el entrenamiento supervisado se tiene dos formas de llevarlo acabo: con entrenamiento secuencial o batch.

Entrenamiento Secuencial: En este tipo de entrenamiento la actualización de los pesos se realiza en cada presentación del vector de entrada.

Entrenamiento Batch: La actualización de los pesos de la Red son actualizados después de que todos los datos del entrenamiento son presentados.

3.5.1 Conjuntos de Datos.

El conjunto de entrenamiento es el encargado de determinar los pesos de la Red Neuronal (aprendizaje), mediante un entrenamiento de la misma, el tamaño de este conjunto suele ser del 80% del conjunto de datos totales. No existe una regla determinada en cuanto al tamaño del conjunto de entrenamiento, pero entre mayor sea este, mayor información se tendrá del sistema; la desventaja es que entre más grande sea este conjunto, más tiempo le toma al algoritmo de entrenamiento ajustar los pesos de la Red, la ventaja es que ajusta mejor los pesos.

Conjunto de Prueba: Este conjunto es el que nos sirve para probar el rendimiento de la Red, verificando qué tan próxima está a la solución correcta, es decir, qué tan precisa es.

3.6 Generalización con Early Stopping.

En el entrenamiento de la Red Neuronal suele haber ciertas complicaciones, ya que debido a la cantidad de datos puede variar el número de *épocas* (iteraciones de las Redes para realizar el entrenamiento) usadas. Se puede dar el caso de tener un sobre entrenamiento, lo que significa, que la Red Neuronal se aprenda la dinámica del conjunto de entrenamiento, obteniendo un error de entrenamiento muy bajo, pero a la hora de presentarle los datos de prueba, el error se dispara considerablemente.

Una forma de resolver este problema es el utilizar un método llamado Early Stopping o validación cruzada, donde se divide el conjunto de entrenamiento en dos conjuntos, en uno se emplea normalmente un 80% de los datos usados para entrenar y el restante en un conjunto de validación. Así se entrena con este nuevo conjunto de entrenamiento y en cada *época* se le presenta a la Red el conjunto de validación, de aquí se extrae un error de validación. Es común ver que el error de validación suele disminuir mientras pasan las *épocas*; el entrenamiento es parado en el momento en el que el error de validación empiece a crecer, así podemos evitar un sobre entrenamiento y tener una mejor generalización.

3.7 Evaluación del Rendimiento.

Se pueden tener diversas formas de evaluar a nuestra Red y de hecho existen dos puntos, por así decirlo, donde se mide el rendimiento de esta, el primero es en el entrenamiento, donde se extrae un error que nos sirve para modificar los pesos de la Red, normalmente se usa el MSE de sus siglas en inglés o bien Error Cuadrático Medio (Mean Squared Error).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(X_i^p - X_i^o\right)^2 \qquad \textbf{(3.5)}$$

donde X_i^p es la predicción actual, X_i^o es el valor actual y *N* es el número de muestras presentadas a la Red Neuronal.

El segundo es extraído del conjunto de prueba y sirve para verificar qué tan bien se entrenó la Red, los errores utilizados en el trabajo se describen en la sección 5.4.

3.8 Reglas de aprendizaje.

Existen diferentes formas de llevar acabo el aprendizaje de una Red Neuronal:

Regla de Hebb: Plantea que si una neurona recibe una entrada de otra neurona y las dos son altamente activas, es decir, que ambas tienen el mismo signo, entonces el peso que las conecta es reforzado.

Regla de Hopfield: Esta es parecida a la anterior, con la diferencia de que aquí se especifica la magnitud del reforzamiento.

Regla Delta: Esta es una variación de la Regla de Hebb y está basada en la idea, de ir modificando continuamente los pesos que conectan las neuronas para reducir la diferencia (Delta) entre el valor deseado y el valor actual de la Red. Esta es la Regla más usada y modifica los pesos con el objetivo de reducir o minimizar el MSE, también es conocida como la Regla de Aprendizaje de Widrow-Hoff o la Regla Least Mean Square (LMS).

Regla de Gradiente Descendiente: Esta es similar a la de la Regla Delta, la cual usa la derivada de la función para modificar el error Delta (incremento del error) y aplicarlo a los pesos, con la diferencia de que aquí se usa una tasa de aprendizaje constante para modificar los pesos, esto sirve para darle estabilidad y una lenta convergencia a la Red.

Regla de Kohonen: Está basada en el aprendizaje de sistemas biológicos. Aquí las neuronas compiten por la oportunidad de aprender o que es lo mismo, actualizar sus pesos.

El Momento y la Tasa de Aprendizaje no son Reglas sino parámetros ajustables para controlar el aprendizaje.

Momento: Es una mejora realizada a la búsqueda de gradiente descendiente en el sentido de memoria, esto es, que mantiene el último incremento del peso para acelerar y estabilizar la convergencia. El valor de este término está entre cero y uno; cuando vale cero, el cambio en el peso está basado solamente en el gradiente; cuando vale uno, el nuevo cambio del peso es igual al último cambio y el gradiente simplemente es ignorado. El momento permite a la Red Neuronal, responder al gradiente local y a tendencias recientes en la superficie del error.

Tasa de Aprendizaje: Este parámetro es la velocidad con que la Red aprende, también es conocido como el tamaño de paso; a un valor muy grande, el algoritmo se vuelve inestable, si es muy pequeño, le toma mucho tiempo converger.

3.9 Algoritmo Back-Propagation.

La Regla Delta toma el error de la capa de salida y lo transforma en una derivada de la función de transferencia, así es usado en la capa anterior para ajustar los pesos, lo que significa que el error es propagado hacia las capas anteriores; esto es lo que se conoce como el Algoritmo de Back-Propagation [19], y su función principal es determinar el gradiente de la función y modificar los pesos de la Red de atrás hacia delante. Este algoritmo es utilizado también para entrenar Redes feed-forward, que busquen aproximar funciones como de Regresión no-lineal y clasificación de patrones entre otros.

Como se mencionó, este algoritmo funciona para cualquier arquitectura de Redes feed-forward, siendo aplicado localmente a cada neurona. Donde se tienen que estar calculando los errores locales de la salida, de acuerdo a las entradas. Primero se introduce una entrada y se manda a través de la Red para encontrar su salida, entonces se calcula el error correspondiente a la capa de salida y así sucesivamente hasta llegar a la capa de entrada; durante todo este proceso los pesos anteriores son usados para determinar el gradiente, una vez que se han encontrado todos los errores locales, se actualizan los pesos de la capa de salida y después los demás pesos de las capas ocultas.

El algoritmo más simple es el que contiene una capa de entrada, una de salida y a lo más una capa intermedia de neuronas, pero no forzosamente se tiene que construir así, podemos aumentar el número de capas intermedias según lo requiera nuestro problema. Existen muchas variantes o paradigmas del algoritmo de Back-Propagation para entrenar Redes feed-forward y una de las desventajas o ventajas, como se quiera ver, de esta configuración (feed-forward - Back-Propagation), es que se requiere de un entrenamiento supervisado con una gran cantidad de entradas-salidas (datos de entrenamiento), aparte de que no se puede entender bien el procedimiento de mapeo interno realizado por la Red (caja negra), lo que no garantiza de que el sistema converja a una solución aceptable [51].

3.10 Algoritmos de Entrenamiento.

Existen diversos algoritmos de entrenamiento de rápida convergencia, estos se dividen en dos categorías, los que usan heurísticas para actualizar los pesos (variable learning rate BP y resilient BP) y los que usan algoritmos de optimización (conjugate gradient, quasi-Newton y Levenberg-Marquardt). A continuación explicaremos el algoritmo que está relacionado con el trabajo de tesis, el cual es Levenberg-Marquardt. Antes de comentar este método, explicaremos un poco, qué es el gradiente.

3.10.1 Gradiente - búsqueda con pasos descendientes.

El gradiente representa los coeficientes de los pesos (w) en una superficie de rendimiento, que es donde se lleva acabo la búsqueda; presenta dos grandes ventajas por lo que: 1) el gradiente se puede calcular localmente; 2) el gradiente siempre apunta en la dirección del cambio más grande. La meta aquí es alcanzar el valor más pequeño, realizando la búsqueda en la dirección opuesta del gradiente.

El algoritmo inicia con una inicialización arbitraria de los pesos $w(0)$, después se calcula el gradiente de la superficie de rendimiento de $w(0)$ y se modifican los pesos iniciales proporcionalmente en la dirección negativa del gradiente. De esta forma se obtienen los pesos $w(1)$ y se repite el procedimiento. Esto es:

$$w(k+1) = w(k) - \eta \nabla J(k) \qquad \textbf{(3.6)}$$

donde η es el tamaño de paso, normalmente es un valor muy pequeño y $\nabla J(k)$ el gradiente de la superficie de rendimiento de la iteración k.

El gradiente no suele ser preciso, por lo que se tiene que estimar. Un estimado instantáneo del gradiente en la iteración *k* es simplemente el producto del la entrada actual con el peso del error actual, esta forma de realizarlo, es el algoritmo conocido como LMS (Least Mean Squared) o regla LMS. Tomando en cuenta esto, nos queda la ecuación:

$$w(k+1) = w(k) - \eta\varepsilon(k)x(k) \qquad \textbf{(3.7)}$$

Donde se tiene al final la multiplicación de la entrada actual con el peso del error. Para mas detalles del algoritmo se puede consultar la referencia [19, 38, 50] donde también se muestra que es equivalente la regla de la cadena (chain rule) con el algoritmo LMS. La figura 3-2 muestra un esquema de lo que sucede mientras se desciende en el gradiente.

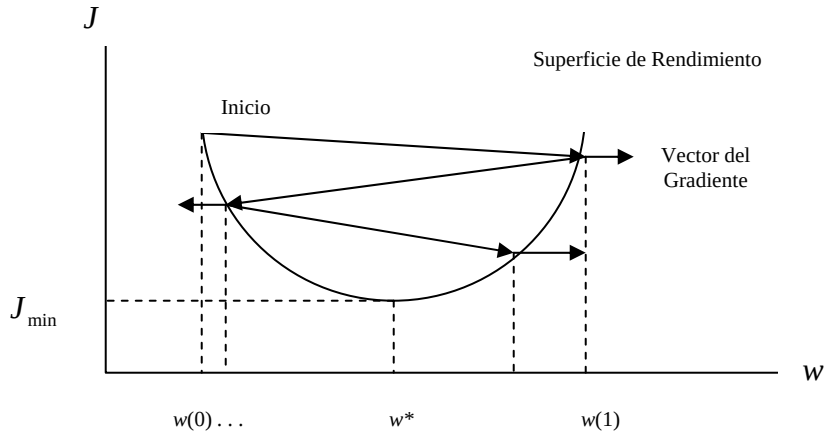


Figura 3-2. Búsqueda usando el gradiente de la información.

3.10.2 Levenberg-Marquardt.

El algoritmo de Levenberg-Marquardt es una técnica iterativa que localiza los mínimos de una función, la cual es expresada por una suma de cuadrados de una función no-lineal. Esta técnica fue diseñada alrededor de 1944 por K. Levenverg para tener un método que resolviera los problemas no-lineales con mínimos cuadrados. En 1963 D. W. Marquardt saca el artículo [52] titulado “An Algorithm for the Least-Squares Estimation of Nonlinear Parameters”.

La primera implementación de este algoritmo, para entrenar una Red Neuronal feed-forward en modo Batch fue realizado por M. T. Hagan y M. Menhaj [38] en 1994. Ahí el algoritmo es comparado con el algoritmo de gradiente conjugado y el de “variable learning rate”, de esta comparación se puede determinar que el algoritmo de Levenberg-Marquardt es más eficiente a los otros dos cuando las Redes no contienen más de unos cuantos cientos de pesos. También se menciona que muchas de las aplicaciones de mínimos cuadrados no-lineales a Redes Neuronales, se concentraban en implementaciones secuenciales, es decir, los pesos son cambiados en cada presentación del vector de entrada. Esta técnica es sólo útil cuando se necesita un entrenamiento en línea, así en [38] se realiza el algoritmo para el modo Batch, donde los pesos son actualizados después de que es presentado todo el conjunto de entrenamiento.

Este algoritmo está catalogado como de segundo orden sin tener que calcular la matriz *Hessiana* (segunda derivada), para ello ocupa el método de Gauss-Newton para aproximarla. En la aproximación de esta matriz, se utiliza la matriz *Jacobiana*, que contiene la primera derivada de los errores de la Red con respecto a los pesos y *bias*, así la actualización de los pesos, no requiere de las derivadas de segundo orden. A este algoritmo se le introduce un valor escalar, que sirve para determinar el comportamiento del mismo, este valor

es denominado con la letra lambda (λ), si es un valor muy grande, el algoritmo se vuelve de gradiente descendiente, con un tamaño de paso pequeño y por el contrario, si el valor es casi cero se utiliza el método de Newton usando la matriz *Hessiana* aproximada, con un tamaño de paso igual, al inverso del valor de λ.

El método de Newton es rápido y preciso cuando se está cerca del error mínimo, por lo que λ es ajustado durante el procedimiento, disminuyéndolo en cada paso mientras el error sea reducido, en el caso contrario se incrementa, así se empieza con grandes pasos (una rápida convergencia) mientras se va disminuyendo el error, para que cuando se esté próximo al mínimo se use el método de Newton teniendo pequeños pasos y una convergencia más precisa y rápida también.

Por lo mencionado anteriormente, este método es considerado entre los más eficientes, cuando se realiza una optimización no-lineal de parámetros [3, 53]. Levenberg-Marquardt es de los algoritmos más ampliamente usados para optimizar los pesos de las Redes Neuronales, encargadas de realizar la predicción, y está considerado como un método rápido para el entrenamiento de Redes feed-forward de tamaño moderado [54]. Debido a los motivos comentados anteriormente, se decidió utilizar dicho método para realizar el entrenamiento de la Redes Neuronales en este trabajo de tesis.

3.11 Normalización de los Datos.

La normalización de los datos es indispensable, debido a que las funciones de transferencia trabajan en el rango de 0 a 1 o de -1 a 1, por lo que meter datos en otros rangos, nos puede traer problemas a la hora de que la información pase por las funciones; lo ideal es normalizar los datos al rango determinado, para evitar que los datos se dispersen al pasar por dichas funciones, todo esto contribuye que el aprendizaje sea de una forma suavizada.

Existen diversas formas de realizar la normalización según el problema a tratar [3], aquí se utilizó una normalización completa de todos los datos, es decir, se utilizo todo el conjunto de entrenamiento junto con los targets (valor objetivo).

La expresión utilizada para la normalización de -1 a 1 en este trabajo es:

$$pn = \frac{2(p - \min p)}{(\max p - \min p) - 1} \qquad \textbf{(3.8)}$$

y para la desnormalización simplemente se despeja *p* de 3.8 quedando:

$$p = 0.5(pn + 1)(\max p - \min p) + \min p \qquad \textbf{(3.9)}$$

donde *p* es el dato actual de la serie, “min *p*” es el mínimo de la serie y “max *p*” el máximo.

3.12 Predicción.

Las Redes Neuronales multi-capa tienen la capacidad inherente de realizar un mapeo de entradas-salidas, lo que permite resolver una diversa cantidad de problemas en cuanto a predicción.

Todos los Modelos de Predicción asumen que existe una relación entre los datos de entrada, que son los valores pasados del fenómeno y los datos de salida que son los valores futuros, pero es muy difícil encontrar dicha función, sobretodo si no se tiene conocimiento previo del sistema, en está parte, las Redes Neuronales son de gran utilidad aproximando dicha función, y no sólo eso, sino que son capaces de aproximarse a los resultados correctos mediante una función no lineal.

Se tienen las siguientes características propias de la predicción:

1. Hay información del pasado.
2. Esa información puede ser cuantificada en forma de datos.
3. Se puede asumir, qué patrones del pasado continúan en el futuro.

Así podemos decir que esta predicción es cuantitativa, las personas que no están familiarizadas con la predicción cuantitativa, a veces suelen pensar que la información del pasado no puede describir el futuro de una forma precisa, porque se tiene la idea de que todo está en continuo cambio, que nada permanece igual, pero la historia se repite a sí misma en cierto sentido [55].

La predicción cuantitativa se puede dividir en dos grupos:

- Predicción de Series de Tiempo: se basa en los valores del pasado y su objetivo es describir patrones históricos y extrapolarlos a patrones futuros.
- Modelos causales: aquí se asume que el factor predicho exhibe una relación de causa-efecto con una o más variables independientes. Por ejemplo, para predecir la variable "ventas" de algún fenómeno, se tiene la función: ventas = F(entradas, precios, competencia, ...)

Existen diversas técnicas de predicción, que permiten realizarla de una forma rápida y sencilla, pero no confiable en todos los casos, ya que en algunos se debe de tener una tendencia bien definida. Como pueden ser:

- Single Moving Averages: Este método consiste en tomar n datos observados y obtener el promedio de estos, así el valor promedio, es tomado como la predicción del siguiente valor en la serie. Este método funciona relativamente bien cuando los datos son estacionarios.
- Single Exponential Smoothing: Aquí no se necesita tener un conjunto de datos, sino únicamente se necesitan dos valores de la serie que pueda determinar el siguiente. Este método, puede mejorar en parte al anterior pero no sirve para datos que no sean estacionarios.

Otros métodos como los presentados aquí se pueden encontrar en [51] algunos con un poco más de procesamiento para realizar la predicción. Estos métodos pueden verse como fórmulas generales, pero ninguno de estos métodos matemáticos, son capaces de predecir series que no sean estacionarias o que no presenten una tendencia bien definida. En esta parte, es de gran utilidad una Red Neuronal para predecir ST, que tienen una dinámica más compleja que las lineales y estacionarias, como pueden ser las cuasiperiodicas, no lineales, no estacionarias o caóticas entre otras.

3.12.1 Predicciones Multi-paso.

Para una serie de tiempo [x_1, x_2,..., x_t], muestreada de un sistema caótico, nosotros podemos predecir los siguientes n puntos en la ST, esto es [x_{t+1},..., x_{t+n}], lo que significa, que no necesitamos todos los puntos anteriores de la serie, para poder hacer una predicción precisa, únicamente con varios de estos puntos, podemos realizar una predicción adecuada.

Básicamente se tienen tres formas de realizar la predicción:

1. Predecir únicamente el siguiente punto en la serie, x_{t+1}
2. Predecir directamente x_{t+n} (predicción directa).
3. Predecir un punto adelante y usar la predicción como parte de la entrada para predecir el segundo y así sucesivamente hasta tener n predicciones (Predicción Iterada).

Aunque se mencionan tres formas de realizar la predicción, uno se puede dar cuenta de que el punto uno es un subconjunto del punto dos, al realizar $n = 1$ en el punto 1. En [17] se trata únicamente las predicciones mencionadas en los puntos 2 y 3, siendo estos, los métodos usados en el trabajo.

3.12.1.1 Predicción directa.

Supongamos que se tiene una ST, [x_1, x_2,…, x_t], y queremos predecir *n* pasos adelante. La predicción directa es un método en el cual el modelo es construido para predecir x_{t+n}

$$\begin{aligned} x_{t+n} &= g(x_t), \\ &= g(x_t, x_{t-1},...,x_{t-(n_d-1)}), \end{aligned}$$

donde $g(\cdot)$ es el modelo de salida.

3.12.1.2 Predicción iterada.

Es un método en el cual el modelo es construido para predecir un paso adelante y la predicción es usado como una parte del vector de entrada para predecir dos pasos adelante y así sucesivamente. En cada paso el modelo estima x_{t+i+1} . Por ejemplo,

$$\begin{aligned} x_t &= (x_t, x_{t-1}, x_{t-2},..., x_{t-(n_d-1)}), \\ x_{t+1} &= g(x_t), \\ x_{t+1} &= (x_{t+1}, x_t, x_{t-1},..., x_{t-(n_d-2)}), \\ x_{t+2} &= g(x_{t+1}), \\ x_{t+2} &= (x_{t+2}, x_{t+1}, x_t,..., x_{t-(n_d-2)}), \\ &\vdots \end{aligned}$$

Este proceso es iterado *n* pasos, alcanzando al final la predicción x_{t+n}. En los experimentos se emplearon ambos métodos antes descritos para predecir 30 puntos adelante en la ST. Algunas ST son más fáciles de predecir que otras debido a la dinámica del sistema, así en dichas series, nos podemos extender aún más en la predicción, por lo que en la sección de Resultados, se podrán apreciar predicciones de hasta 240 puntos adelante. En el trabajo [56] se realiza la predicción a un paso adelante, éste sirve como base para la tesis junto con [57, 58].

Para ambas formas de predicción, la Red obtenida, siempre tendrá un nodo de salida, así podemos apreciar gráficamente en la figura 3-3 para la predicción directa y 3.4 para la Iterada. Para la predicción directa se predice el primer punto, luego se reacomodan los datos para predecir 2 puntos adelante, se vuelve a entrenar con la nueva dinámica y se predice el segundo punto, y así sucesivamente. A diferencia de como se menciona en [3], donde la Red tiene *k* salidas, prediciendo en un sólo paso los *k* siguientes puntos.

Para la predicción iterada sin IMF, se aplica tal cual como dice el método en 3.12.1.2, pero cuando usamos IMF no podemos aplicarlo directamente y esto es debido a que la Red se entrena no sólo con los datos originales, sino que también se entrena con *n* descomposiciones (IMFs), entonces al predecir nos va a devolver el dato predicho, que es una entrada para predecir el siguiere número, pero nos faltan otras *n* entradas que corresponde a las descomposiciones para ese punto. Este problema se solucionó volviendo a calcular las IMF después de cada predicción, así se obtendrían las componentes tomando en cuenta ese nuevo valor predicho, y poder predecir el siguiente valor, esto hace que la predicción Iterada con IMF sea más pesada computacionalmente como se muestra en la sección 7.3.3.

Ahora bien, así es la forma convencional de realizar la predicción, pero nada nos dice que forzosamente podamos utilizar los t-n_d puntos anteriores para predecir el punto $t+1$, lo que significa, que podemos utilizar los t-n_d puntos anteriores pero no consecutivos.

Ahora se tienen dos problemas para los datos de entrada, el primero es poder determinar cuántos datos anteriores son necesarios para predecir el siguiente punto y el segundo es el calcular un tamaño de *retardo* entre dichos puntos. Esto se puede ver también como una ventana de tamaño fijo que se va desplazando por los datos de entrenamiento, estas variables son optimizadas por el GA, y se describen en el capítulo 6.

En este capítulo se comentó todo lo necesario para entender qué es una Red Neuronal y poder construirla, se observó cómo una neurona procesa la información y cómo es la función de salida de la Red Neuronal, lo que implica en cierto sentido, conocer el flujo de información dentro de ella. También pudimos ver cómo se realiza el entrenamiento con validación cruzada, evitando un sobre entrenamiento de la Red. Así mismo se mencionó las formas de realizar predicción.

El siguiente capítulo estará destinado a conocer el Algoritmo Genético tal cual, es decir el algoritmo GANN será mencionado en el capítulo VI.

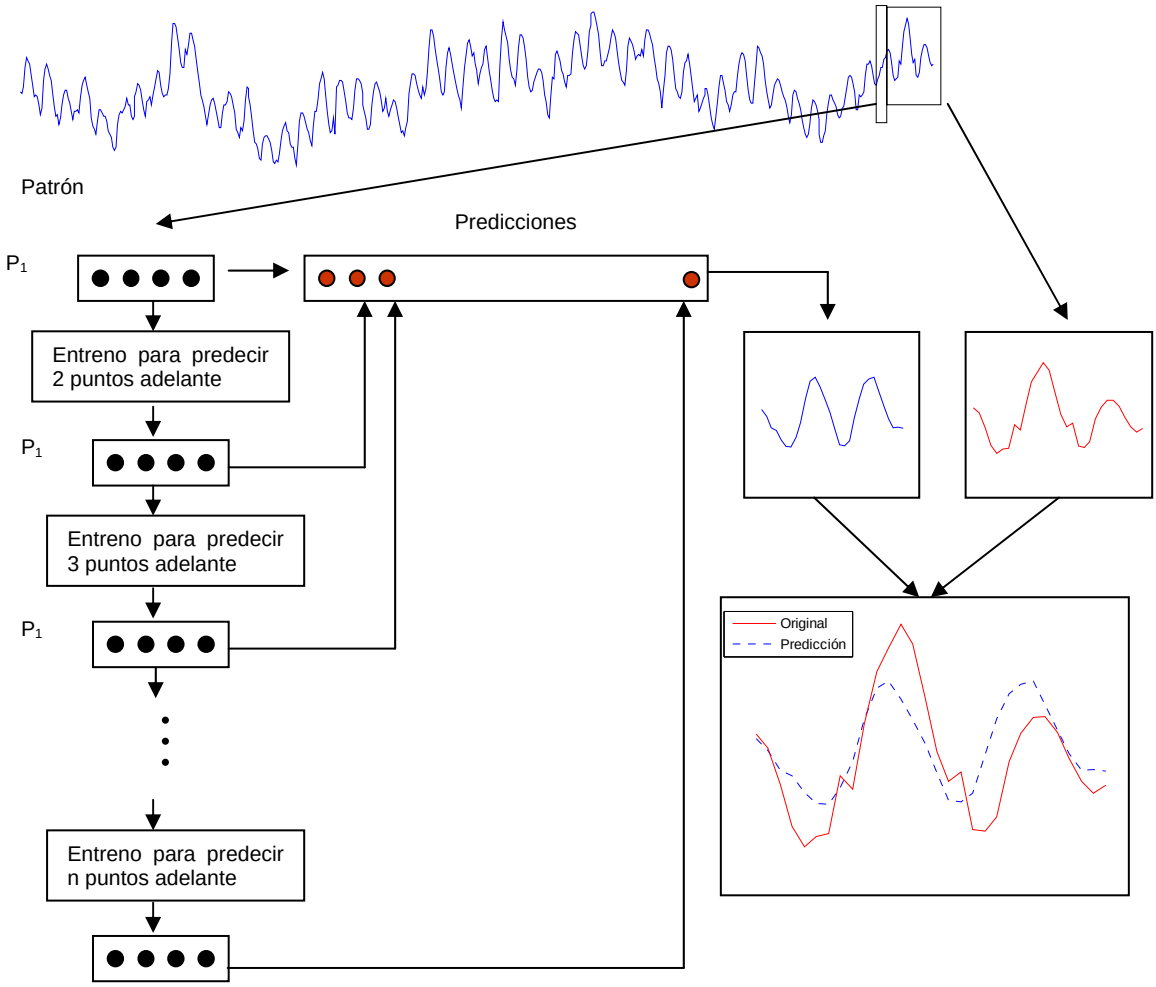


Figura 3-3. Predicción directa.

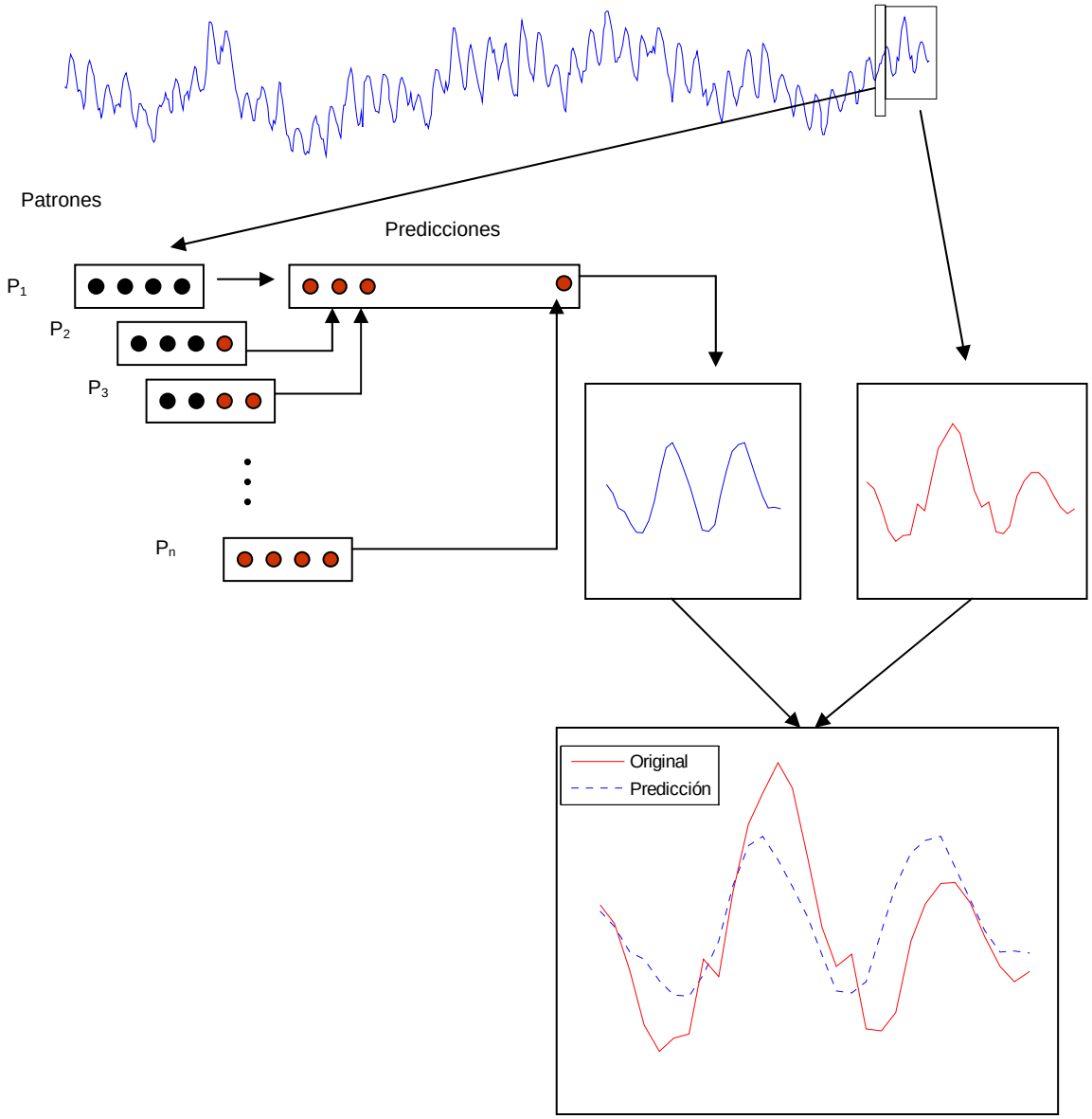


Figura 3-4. Predicción iterada.

Capítulo IV. Algoritmo Genético.

El Algoritmo Genético (GA) fue introducido por John Holland en 1970 [59]. El GA es un método para resolver problemas de optimización, imitando de forma metafórica a la selección natural y simulando la evolución biológica, está catalogado como un algoritmo de búsqueda de propósito general, para resolver problemas de naturaleza combinatoria. El GA en cada paso modifica a la población, selecciona individuos aleatoriamente de la población actual para ser padres y los usa para producir a los hijos que serán los miembros de la siguiente generación; después de generaciones sucesivas, la población evoluciona hacia una solución óptima.

Los Algoritmos Genéticos se han utilizado en la resolución de problemas difíciles de resolver, como los problemas NP-Completos, para el aprendizaje automático y para el de diseño o entrenamiento de Redes Neuronales [33] entre otros. La ventaja del GA es que se puede adaptar a una amplia gama de problemas donde se busque optimizar una función, además de tener una ventaja sobre otros métodos de optimización, realizando selecciones y cambios aleatorios en vez de ser determinista, lo que permite que los individuos evolucionen de una forma un poco más parecida a lo que sucede en la naturaleza. El GA usa tres reglas principales en cada paso para crear a la siguiente generación de la población actual: Selección, Cruzamiento y Mutación. A continuación se muestra un pseudo-código básico del Algoritmo Genético.

1. Se crea aleatoriamente la población inicial y se establecen el contador de generaciones a uno.
2. Se crea un ciclo en el cual el GA irá creando nuevas poblaciones o generaciones realizando los siguientes pasos:
    a. Aplicar la función de Adaptabilidad a cada miembro, obteniendo su adaptación.
    b. Escalar el valor crudo de adaptabilidad en un mejor rango para su selección.
    c. Seleccionar a los padres de acuerdo a su adaptabilidad y seleccionar los miembros de la *Elite*.
    d. Crear a los hijos o individuos de la siguiente generación a partir de los padres, usando cruzamiento y mutación.
    e. Incrementar el contador de generaciones.
3. El algoritmo termina cuando se tenga algún criterio de paro.

4.1 Codificación del problema.

Debido a que el GA se puede aplicar a la búsqueda de soluciones óptimas de diversos problemas, es necesario que se tenga en cuenta el tipo de codificación a usar (genotipo, definición en sección 9.4), ya que si no es una representación apropiada, al hacer evolucionar las variables, se podrán tener puras soluciones inválidas y en vez de avanzar, retroceder en la solución.

Este es un problema que se tiene desde los principios de la computación y más acentuado al aparecer la Inteligencia Artificial (IA), ya que se busca hacer una representación adecuada del problema en cuestión, que pueda manipular la computadora, para poder obtener una solución adecuada. Por desgracia en la mayoría de los casos no se puede tener una codificación perfecta, por lo que se tiene que buscar una aproximación adecuada, para que funcione lo más parecido a la realidad. Los tipos de codificaciones usadas suelen variar, pero las más comunes son: representación directa (por ejemplo números reales o binarios) y codificaciones más pesadas (indirecta), en los que se pone un poco más de esfuerzo computacional para representar el fenómeno.

4.2 Pasos realizados por el GA en cada iteración.

Todas las opciones listadas abajo (Selección, Cruzamiento, Mutación, etc.), tienen por lo regular diversas formas de llevarlas a cabo, por lo que se dará una breve descripción de ellas a continuación.

4.2.1 Selección.

Es la forma de escoger a individuos de la población actual (padres), para generar a los hijos (miembros de la siguiente población). La Selección es la base del Algoritmo Genético, porque permite el paso de la información contenida en los genes a través de las generaciones. Existen varias formas de poder realizar la Selección: de forma determinista o probabilística. Las diferentes formas de aplicar la Selección son:

- **Estocástica uniforme o Proporcional:** Aquí cada individuo tiene la probabilidad de ser escogido, de forma proporcional a su adaptabilidad escalada, algunos suelen describir diferente este método y realizan una represtación gráfica con algunas variaciones donde se dibuja una línea en la cual cada individuo corresponde a una sección de línea proporcional a su adaptabilidad escalada, entonces con un paso de tamaño fijo, se va recorriendo la línea escogiendo a los individuos. El primer paso suele ser un número aleatorio menor que el tamaño de paso original.

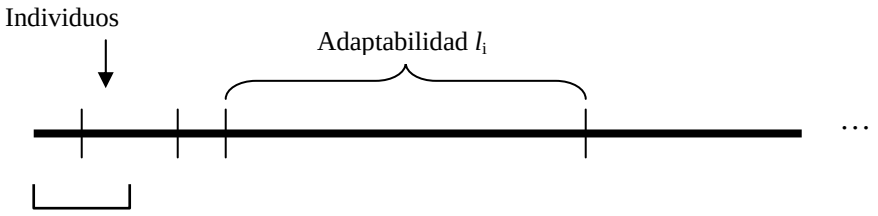


Figura 4-1 Selección Estocástica uniforme.

- **Residuo:** La selección por Residuo, asigna padres de forma determinista de la parte entera del valor de adaptabilidad escalado de cada individuo, usando la selección de Ruleta en la parte fraccional. Por ejemplo, si tiene un valor de adaptabilidad ya escalado de 2.3, tiene una probabilidad proporcional a 0.3 de ser seleccionado en este paso, si esto sucede, entonces se listará como padre dos veces.
- **Ruleta:** Este método escoge a los padres simulando una rueda de ruleta, donde cada individuo tiene una sección de la rueda proporcional a la expectación del individuo. Por lo que cada individuo tiene una probabilidad proporcional de ser seleccionado igual al de su área en la rueda, par lo que se usa un número aleatorio.
- **Torneo *k*/*l*.** La selección por Torneo permite escoger a un conjunto de individuos ***k***, los cuales competirán entre si, los ***l*** padres se escogen de los mejores individuos adaptados.
- Otros métodos, como el de Suma y Coma (Plus and Comma) son mencionados en [60]. La ventaja en el GA, es que cada quien puede implementar la función que se adapte mejor a sus necesidades y esto no sólo aplica a la Selección, sino también para el Cruzamiento y Mutación.

4.2.2 Cruzamiento.

El Cruzamiento combina a dos padres para formar a un hijo de la siguiente generación, la ventaja que tiene, es que puede combinar los genes de los individuos para tratar de encontrar mejores soluciones. Desde el punto de vista biológico, el combinar a dos individuos mezclando sus genes puede dar a un individuo mucho mejor adaptado que los padres; aquí también existen muchas y muy diversas formas de combinar a dos individuos, las más conocidas son:

- **Cruzamiento en un punto:** Teniendo el caso de usar un genoma compuesto de bits, se toma un valor aleatorio para dividir a dos padres, el hijo es creado de la primera parte del padre uno y la segunda parte del hijo es tomada de la segunda parte del padre dos.
- **Cruzamiento en dos puntos:** Es igual que el anterior con la diferencia que se hace en dos puntos distintos, no forzosamente consecutivos, incluso pueden ser aleatorios los puntos de cruce. Aquí no es forzoso que el tamaño de cruce de ambos padres sea igual.

- **Uniforme:** Los genes al ser intercambiados son escogidos aleatoriamente, de tal manera que se intercambie el promedio de la longitud de los genes.
- **Aritmético:** Se realizan operaciones aritméticas con los genes de los padres, para dar como resultado la codificación genética del hijo.
- **Disperso:** El Cruzamiento Disperso consiste en generar un vector aleatorio binario el cual nos indicará las posiciones a cruzar, si el vector creado tiene un "1" se toma al elemento del padre uno y si tiene un "0" se toma el elemento del padre dos para crear al hijo, como se muestra en la figura 4-2.

Padre1 = [1 1 0 1 0 1 0 1]

Padre2 = [1 0 0 1 1 0 0 1]

Vector = [0 1 1 1 0 1 0 1]

Hijo = [1 1 0 1 1 1 0 1]

Figura 4-2. Cruzamiento disperso.

- Existen otros métodos que se pueden encontrar en la literatura como el método de Cruzamiento Heurístico e Intermedio.

4.2.3 Mutación.

La Mutación aplica cambios aleatorios a un padre para formar un hijo permitiendo la diversidad genética, esta es de suma importancia para el GA, ya que permite explorar otras zonas del espacio de soluciones que normalmente no se cubrirían si se usara únicamente el Cruzamiento, lo que permite, no caer en máximos o mínimos locales. Algunas formas de realizar la mutación son:

- **Gausiana**: Agrega un número aleatorio tomado de una distribución gausiana con media cero al padre seleccionado.
- **Uniforme**: Aquí el algoritmo selecciona una parte del vector de entrada para ser modificado, donde cada entrada tiene una probabilidad *k* de ser modificado, luego el algoritmo remplaza dicha selección por un número aleatorio uniforme de acuerdo al rango de la entrada.
- **Inversión de Genes**: Se seleccionan genes aleatoriamente y se invierte su valor. Se suele utilizar en representaciones de cadenas de bits, cambiando ceros por unos o viceversa.
- **Cambio de Orden:** Se seleccionan dos genes aleatoriamente y se intercambian sus posiciones. Se utiliza en representaciones basadas en permutaciones.
- **Modificación de genes:** Se realizan pequeñas modificaciones en los genes, los cuales son escogidos aleatoriamente. Por ejemplo, en una codificación basada en números reales se realizan sumas de números muy pequeños positivos o negativos.

4.2.4 Escalamiento de Adaptabilidad.

Al evaluar con la función de Adaptación a los individuos, esta nos regresa un valor de adaptabilidad crudo, por así decirlo; estos valores no se encuentran en una distribución normal, por lo que, si se usara dicho valor crudo para la Selección, podría darse el caso de seleccionar a puros individuos no tan adaptados como otros, cuando en realidad debería de tener preferencia con los mayor adaptados, por lo que se opta por escalar dichos datos a valores más convenientes, esto se llama Escalamiento de Adaptabilidad y existen diversas formas de hacerlo:

- **Rank:** Este escalamiento cambia la adaptabilidad obtenida, por un valor que depende de su adaptabilidad, así el individuo mejor adaptado tendrá el valor de 1 como adaptación, el siguiente más adaptado 2 y así sucesivamente, este valor escalado es proporcional a $1/\sqrt{n}$ y la suma de todos los

valores escalados de la población es igual al número de padres necesitados para formar la siguiente generación.

- **Proporcional:** Este método cambia la adaptabilidad por un valor proporcional conforme a la adaptabilidad obtenida por la función objetivo.
- Existen otros métodos como el Cambio lineal estático, Cambio lineal dinámico y Top entre otros, que también son utilices para dicha labor.

4.3 Opciones de Reproducción.

El algoritmo genético crea tres tipos de hijos para la siguiente generación basados en:

La Fracción de Elite. Es el número de individuos mejor adaptados que se seleccionan de la población actual, si se tiene una Elite de dos, entonces se toman a los dos mejores individuos pasando a la siguiente generación sin sufrir ningún cambio.

La Fracción de Cruzamiento. Es la cantidad de individuos creados con cruzamiento, si el porcentaje es cero, entonces lo hijos serán iguales a los padres y sólo serán alterados con la mutación.

Fracción de Mutación: Son el número de individuos creados con mutación; esto previene que el GA caiga en mínimos o máximos locales. No es bueno tener un alto índice de Mutación, ya que en vez de hacer una búsqueda un poco más inteligente sería una búsqueda aleatoria, en el sentido de que se realizan ciertos pasos lógicos, para generar nuevos individuos, y dichos pasos están basados en la evolución natural y no emplea el término inteligente en el sentido que tenga inteligencia el algoritmo.

Ejemplo:

- Para una población de 20 individuos
- Con una fracción de Elite de 2
- Una fracción de Cruzamiento del 80%
- Y una fracción de Mutación del 20%.

Entonces al crear a la siguiente generación, se van a tomar a 2 individuos, los cuales son los mejor adaptados en la generación actual, 14 serán creados con cruzamiento y 4 con mutación.

4.4 Opciones de Población.

Tipo de Población: Se puede tener diferentes tipos de datos que conforma nuestro genotipo como: vectores, cadena de bits, etc. Sobre estos datos se realizan las operaciones básicas del GA.

Tamaño de la Población: Establece el número de individuos por población. Si el tamaño es muy grande se puede dar el caso, en el que los resultados no mejoren y sea muy tardado el algoritmo, si es muy chico, tiene muy pocas posibilidades de evolucionar por el cruzamiento.

Población Inicial: Podemos inicializar nuestra población de una manera aleatoria si no tenemos información que nos permita atacar el problema, esta es la forma recomendada, si se quiere iniciar con una población preestablecida, tenemos que estar seguros de que los individuos tengan una dispersión lo suficientemente grande para que el algoritmo no avance muy lentamente.

Diversidad de Población: La diversidad es un factor muy importante en el desempeño del algoritmo, esta se puede medir con la distancia promedio entre los individuos, si este valor es muy grande, la diversidad es alta, en el caso contrario es pequeña. En las etapas iniciales de la búsqueda, se debe mostrar una gran diversidad, mientras que al final debe disminuir para conseguir una solución adecuada.

4.5 Migración.

La migración especifica cómo se mueven los individuos entre las sub-poblaciones, el mejor individuo de una sub-población reemplaza al peor de otra, al moverse el individuo este se copia, es decir se queda el original en su lugar.

La migración ocurre de forma controlada, es decir, podemos especificar cuántas generaciones han de pasar entre las migraciones (intervalo). Por ejemplo, para un valor de 10, las migraciones ocurrirán cada 10 generaciones. También está la fracción de migración donde se especificará cuántos individuos se moverán entre las sub-poblaciones.

4.6 Criterios de Paro.

Existen diversos criterios con los que podemos detener el algoritmo sin que se cumpla el total de generaciones preestablecidas, como son:

Generaciones: Son las iteraciones que realiza el GA, a lo largo de ellas se da la evolución, con el objetivo de obtener mejores resultados.

Límite de Tiempo: Especifica el tiempo límite en segundos que puede estarse ejecutando el algoritmo.

Límite de Adaptabilidad: Es el valor preestablecido de adaptabilidad, cuando un individuo iguale o mejore dicha adaptabilidad el algoritmo se detendrá.

Paro por generaciones: El algoritmo se para si no hay una mejora de la adaptabilidad en el número de generaciones establecidas aquí.

Paro por tiempo: El algoritmo se detiene, si después de un tiempo determinado en segundos, no se mejora la adaptabilidad.

4.7 Espacio de Búsqueda.

Al conjunto de todas las posibles soluciones a un problema en particular, se le llama espacio de búsqueda, donde cada punto representa una posible solución. A cada posible solución se le asigna un valor de adaptación (fitness) que nos indicará qué tan apto es el candidato para la resolución del problema.

Entonces, de lo que se trata, es de encontrar un máximo o mínimo en el espacio de búsqueda. No es indispensable o necesario conocer algunos puntos en dicho espacio, ya que el GA se encarga de generar otras soluciones a medida que éste realiza la evolución, pero sí ayuda en gran medida si se conoce algún indicio para facilitar la búsqueda, ya que esta puede ser muy compleja debido al problema a tratar.

Se puede calcular el tamaño del espacio de soluciones, al obtener la combinación de las variables a optimizar, de esta forma, podemos conocer el porcentaje de individuos muestreados en cuanto al tamaño de nuestro espacio.

4.8 Complejidad Computacional.

Cuando se quiere resolver un problema, puede existir una gran cantidad de algoritmos aplicables. Se puede decir que el orden de complejidad de un problema, es la complejidad del mejor algoritmo que se conozca para resolverlo; así se clasifican los problemas teniendo las siguientes categorías: clase P, NP y NP Completos. A continuación se da una breve descripción de estos.

Clase P: Los problemas para los que se conocen algoritmos con complejidad polinomial, se dice que son de la clase P y estos algoritmos son tratables en el sentido de poder ser abordados en la práctica. Para aquellos problemas en el que la mejor solución conocida es mayor a la polinomial, se les suele llamar intratables (pero no por eso, no se tratan de resolver).

Clase NP: Algunos de estos problemas se pueden comprobar con algoritmos polinomiales para ver si la solución es válida o no. Esto lleva a soluciones no deterministas, aplicando heurísticas y comprobando a ritmo polinomial si dichas soluciones son válidas.

Clase NP Completos: Los problemas NP Completos son los problemas más complicados dentro de los NP, lo que los caracteriza, es que son todos iguales y si se descubriera una solución para los problemas NP-completos, esta sería aplicable a todos los problemas NP.

La mayoría de los problemas interesantes y de la vida real suelen ser NP o NP Completos. Una de las ventajas del GA, es que puede tratar estos problemas y puede aproximarse a una solución en tiempo polinomial; la desventaja es que el GA no garantiza llegar a un máximo ni un mínimo global.

4.9 Elementos del GA.

Por lo discutido anteriormente, podemos listar los puntos esenciales que debe tener un Algoritmo Genético:

1. Se debe de tener una estrategia de codificación.
2. Generar la población inicial (aleatoria o iniciarla con una solución aproximada).
3. Poder seleccionar a un subconjunto de la población actual.
4. Evaluación de los individuos (función de Adaptabilidad).
5. Escalamiento de la Adaptabilidad.
6. Tener una estrategia para que sobrevivan los individuos.
7. Cruzamiento entre los individuos.
8. Mutación.
9. Elitismo.
10. Migración de sub-poblaciones (aislamiento de sub-poblaciones por un límite de tiempo).
11. Auto-Adaptación.
12. Criterios de paro.

Algunos autores suelen incluir algunos puntos más como en [60] donde se incluye la co-evolución y el fuzzines.

4.10 Auto-Adaptación.

La auto-adaptación nace de la necesidad de poder ajustar los parámetros del GA de manera automática, para poder utilizar el algoritmo en diversas tareas, sin la necesidad de tener que buscar un balance adecuado entre sus parámetros; la forma de hacer esto, es metiendo en el genoma dichos parámetros y dejar que evolucionen, a esto se le llama auto-adaptación [60]. Los parámetros que se escogieron para realizar la búsqueda de ANNs se pueden encontrar en la sección de Resultados, dichos parámetros no se introducen al genoma para una auto-adaptación, ya que eso se encuentra fuera del alcance de este trabajo de tesis.

En este capítulo observamos un pseudo-código del GA y las operaciones que realiza en cada iteración, como Selección, Cruzamiento y Mutación entre otras, permitiendo una evolución de nuestro genotipo, lo que significa, optimizar nuestra función objetivo. También se vieron los criterios de paro y el espacio de búsqueda el cual es sumamente importante conocer, ya que es el espacio en el que se mueve el GA. En el siguiente capítulo veremos una técnica de análisis de señales, llamada Descomposición Empírica en Modos, la cual nos permitirá obtener mayor información de nuestro problema para introducirlo a nuestra Red Neuronal.

Capítulo V. Descomposición Empírica en Modos.

La Descomposición Empírica en Modos (EMD) fue introducida por Huang et al en 1998 [13]. El método puede descomponer cualquier señal complicada (no lineales y/o no estacionarias) en un número finito y a veces pequeño de "Funciones de Modo Intrínseco" (IMF de sus siglas en inglés). El método es adaptativo y altamente eficiente identificando estructuras embebidas en la señal. La descomposición está basada en las características locales de tiempo-escala de los datos.

La EMD es una técnica reciente de análisis de señales, las Funciones de Modo Intrínseco obtenidas de aplicar EMD, presentan tiempo y frecuencia al mismo tiempo, algo que el análisis de Fourier no es capaz de hacer, y sólo puede presentar uno a la vez, lo que significa una limitante para el método.

Aparte de lo mencionado, la transformada de Fourier es válida bajo ciertas condiciones:

- El sistema debe de ser lineal
- Los datos deben de ser estrictamente periódicos o estacionarios

De otra manera, el espectro de Fourier tendría poco sentido físico. Como se mencionó anteriormente, EMD no tiene dichas restricciones, por lo que este método, viene a resolver una mayor cantidad de problemas, que el análisis de Fourier.

5.1 Procedimiento de Descomposición.

Antes de continuar vamos a definir qué es una extrema: una extrema esta definida como un punto de inflexión, donde la derivada de la señal cambia de signo, para extraer las IMFs éstas tienen que satisfacer dos condiciones:

1. El número de extremas y el número de cruces con cero debe de ser igual o diferente a lo más por uno.
2. En cualquier punto, la media de la envolvente definida por el máximo local y mínimo local es cero.

Esto permite que cada IMF abarque un sólo modo oscilatorio, lo que significa, que no se permita que ondas complejas estén montadas en la señal.

El principio de esta técnica es el de dividir una señal X_t en una suma de funciones; donde se tienen que satisfacer las condiciones de IMF anteriormente mencionadas. El método esencialmente comprende dos pasos:

1. Dos curvas suavizadas son construidas conectando todos los máximos y mínimos de X_t respectivamente para obtener las envolventes $X_{max}(t)$ y $X_{min}(t)$. Las extremas son encontradas por los cambios de signo de la derivada de la señal. Todos los puntos deben de ser cubiertos por las envolventes superior e inferior.
2. La media de las dos envolventes m_1 es restada de la señal original para obtener la primera componente h_1

$$h_1 = X(t) - \frac{X_{\max}(t) + X_{\min}(t)}{2} \qquad \textbf{(5.1)}$$

Idealmente h_1 debería de ser una IMF, pero si no cumple con los requerimientos de ésta, el proceso es repetido. A cada iteración se le llama proceso de cambio (shifting process). Ahora h_1 es tomado como los datos y se tiene

$$h_{11} = h_1 - m_{11} \quad \textbf{(5.2)}$$

Este procedimiento de cambio se puede repetir *k* veces, hasta que h_{1k} sea una IMF, esto es

$$h_{1k} = h_{1(k-1)} - m_{1k} \quad \textbf{(5.3)}$$

La primera IMF se obtiene al final de los cambios

$$c_1 = h_{1k} \quad \textbf{(5.4)}$$

El proceso de cambio nos sirve para:

a) Eliminar las ondas montadas en la señal.
b) Suavizar las amplitudes desiguales.

Para garantizar que las componentes de IMF tengan suficiente significado físico de modulaciones en amplitud y frecuencia, se determina un criterio de paro, limitando el tamaño de la Desviación Estándar (SD) computada para dos cambios consecutivos.

$$SD = \sum_{t=0}^{T} \left[\frac{\left(h_{1(k-i)}(t) - h_{1k}(t)\right)^2}{h_{1(k-1)}^2(t)} \right] \quad \textbf{(5.5)}$$

En nuestros experimentos SD fue establecida a 0.3. Después de obtener c_1, podemos separar la información obtenida del resto de los datos con:

$$r_1 = X(t) - c_1 \quad \textbf{(5.6)}$$

Pero debido a que r_1 todavía contiene información útil, podemos repetir el mismo procedimiento mencionado arriba, ahora con r_1 como los datos de entrada y r_n como el residuo final.

$$r_n = r_1 - c_2 = r_2, \ldots, r_{n-1} - c_n \quad \textbf{(5.7)}$$

El proceso de cambio puede ser parado por algunos de los dos siguientes criterios a continuación mencionados: cuando c_n o el residuo r_n se vuelvan lo suficientemente pequeños; cuando el residuo r_n se convierta en una función monotónica, para la cual no se puedan extraer más IMFs. Cuando los datos llegan a tener media cero, el residuo final puede ser diferente de cero; para los datos con una tendencia, el residuo final, tendrá dicha tendencia. La señal original puede ser reconstruida usando la siguiente ecuación.

$$X(t) = \sum_{i=1}^{n} c_i + r_n \quad \textbf{(5.8)}$$

Una forma de ver este procedimiento de descomposición, es que las IMFs nos pueden proporcionar diferentes perspectivas del fenómeno y al introducir esa información a las Redes podemos obtener una predicción más precisa; que es la idea del trabajo a comprobar. Otra forma de explicar como trabaja EMD es

la siguiente: se saca la más alta frecuencia oscilatoria que se encuentra en la señal, así localmente, cada IMF contiene frecuencias oscilatorias bajas; esta propiedad puede ser de gran importancia para detectar cambios en la frecuencia, este cambio puede ser detectado mejor a nivel de IMF [61]. Al aplicar EMD nos da diversas IMFs según el tipo de señal y lo larga que ésta pueda ser, además se considera que no todas las IMFs pueden ser de utilidad, por lo que en el siguiente capítulo se realizarán experimentos introduciendo diversas IMFs para poder determinar si hay algún criterio en el que puedan ayudar más algunas IMFs que otras a la predicción. En la figura 5-1 se muestra la Serie Temperatura Mínima de Melbourne, con su correspondiente descomposición usando EMD, donde se puede observar que los modos van de altas a bajas frecuencias, ahí mismo se puede observar más claramente en las ultimas IMF, como ya no existen ondas montadas en la señal.

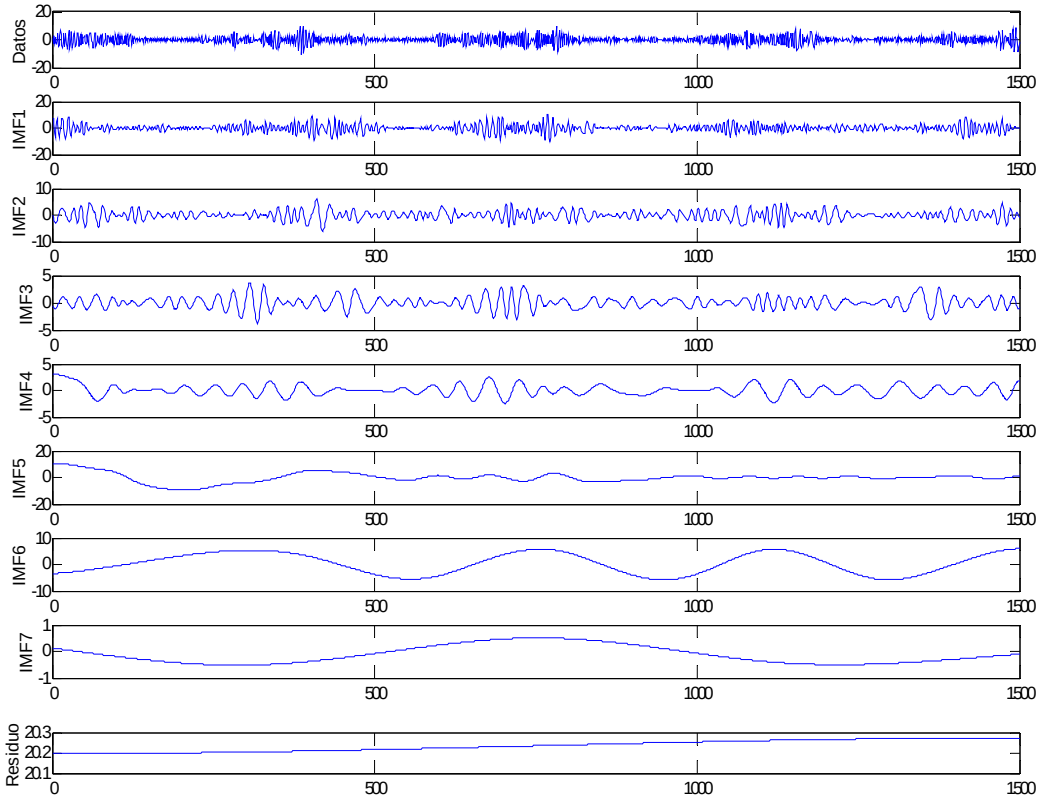


Figura 5-1. Temperatura Mínima de Melbourne con sus 7 IMFs y el residuo.

5.2 Discusión.

El número de componentes extraídas, depende de la dinámica del sistema y de la longitud del mismo. En la tabla 7.6 se muestran todas las ST empleadas en el trabajo, junto con el número de componentes obtenidas al aplicar EMD, ahí se puede notar que el menor número de IMFs obtenidas es 5, el máximo es de 10 IMFs, teniendo en cuenta la restricción de 1500 datos como máximo (restricción mencionada en el capítulo VII).

Cabe señalar que el residuo no es considerado como una IMF, sino como la tendencia final del sistema como se mencionó anteriormente. Aunque EMD no ha sido utilizado para la predicción de Series de Tiempo, hay una amplia gama de trabajos realizados con esta técnica dando resultados favorables [40, 41, 42, 61, 62, 63] en el análisis de señales. Las primeras tres referencias se comentaron brevemente en el Estado del Arte, donde [40] se ocupa para obtener los ciclos de manchas solares, [41] utiliza EMD para la extracción de

señales atmosféricas inter-anuales y [42] donde se realiza el reconocimiento de la silueta humana con Redes Neuronales utilizando EMD.

EMD también se ha ocupado para analizar señales sísmicas [61], donde se obtienen otras perspectivas del fenómeno que antes no se tenían. También se cuenta con trabajos donde se ve el comportamiento de EMD en situaciones estocásticas, cuando se le añade ruido a la señal [62] dando buenos resultados.

Por ejemplo, en [63] se presenta un análisis cuantitativo y cualitativo entre ondeletas y EMD. Se muestra como se dispersa mayor energía al obtener el espectro de Hilbert con las componentes obtenidas de ondeletas, en comparación con el mismo espectro, utilizando las IMFs; también se muestra que IMF es más resistente al ruido, cuando se trata de reconstruir una señal, su contraparte obtiene una mayor distorsión en la reconstrucción, eso puede ser debido al down-sampling y up-sampling que sufre la señal en la descomposición y reconstrucción con ondeletas, es decir, cada descomposición queda a la mitad de tamaño que la original (down-sampling) y cuando se tiene que reconstruir, se agregan ceros entre los valores para aumentar al doble de tamaño la descomposición (up-sampling).

El término de energía en una señal o Serie de Tiempo, se ha vuelto un término común empleado en las técnicas de análisis de señales. Así este término está relacionado con la amplitud de la serie (el valor máximo de la señal con respecto del origen), más específicamente, la amplitud al cuadrado nos da la densidad de energía contenida en la señal [13]. Se tiene la expresión 5.9, con la cual determinamos la energía en la señal o ST tomada de [64], así la energía se puede apreciar en los espectros de Fourier o Hilbert, según sea el caso. Pero en este trabajo de tesis, no es indispensable trabajar con el espectro de Hilbert, sólo es necesario, la obtención de las componentes (IMFs), por este motivo, no se puede utilizar el término de energía tal cual, por lo que se empleará dicha expresión para determinar a las componentes que tienen mayor o menor amplitud total, así se obtendrá un valor escalar de la expresión 5.9, que en principio es la suma de todas las amplitudes al cuadrado, y nos servirá para determinar qué componentes vamos a introducir a nuestra Red.

$$AT\left(f\left(t\right)\right)=\sum_{t=1}^{n} f\left(t\right)^2 \qquad \textbf{(5.9)}$$

Ya teniendo esta cuantificación, podemos realizar los experimentos mencionados en la sección 7.4, donde introducimos al modelo el 70% de las componentes de menor amplitud y el 70% de mayor amplitud, entre otras. La Tabla 5.1, nos muestra las amplitudes de la figura 5-1:

Tabla 5.1 Amplitudes totales para Temperatura Mínima de Melbourne con sus 7 IMFs y el residuo.

Serie/IMF	Amplitud total al cuadrado
Serie Original	2.1261e+005
IMF1	4562.7
IMF2	3049.6
IMF3	1530.7
IMF4	1689.2
IMF5	563.78
IMF6	15564
IMF7	529.2
Residuo	1.7974e+005

De esta forma podemos tener un porcentaje que nos indique qué componente o IMFs tienen mayores o menores amplitudes totales, y esto nos va a servir para realizar los experimentos en el capítulo VII, tratando de determinar, si es que existe un porcentaje de éstas que ayuden más a la predicción que otras.

En [13] se reporta un serio problema de ajuste de curvas al final de la señal, lo cual puede propagar distorsiones en ella, corrompiendo todo el conjunto de datos, en especial se ven afectadas las componentes de baja frecuencia, por este motivo se utilizó el algoritmo de EMD reportado en [13], pero con las modificaciones realizadas por G. Rilling, reportadas en [65].

En este capítulo nos dimos cuenta de cómo vamos a obtener más información de nuestro fenómeno (ST) para poder introducirlo a una Red Neuronal y poder así ayudar a obtener una predicción más precisa, la figura 5-1 la podemos tomar como ejemplo, si fuéramos a utilizar todas la IMFs para realizar la predicción, se tendrían 8 entradas a la Red (datos originales y las 7 IMFs) sin contar con el Residuo.

En el capítulo III ya vimos qué es una Red Neuronal, cómo se construye y entrena; en el capítulo IV vimos el GA para optimizar una función y en este capítulo observamos la técnica de análisis de señales empleada, ya sólo nos falta comentar cómo van a estar integrados y cómo van a interactuar éstos tres métodos, así en el próximo capítulo trataremos el algoritmo GANN el cual nos va a permitir evolucionar arquitecturas de Redes Neuronales de acuerdo a las entradas disponibles.

Capítulo VI. Algoritmo GANN

En el capítulo 3 y 4, nos dimos una buena idea de la forma en que trabaja el Algoritmo Genético y las Redes Neuronales, por lo que en este capítulo explicaremos la unión de ambos, dando como resultado el algoritmo GANN, donde principalmente se optimizan las arquitecturas de Redes. De una forma representativa, podemos observar el diagrama que se encarga de evolucionar a las Redes [66].

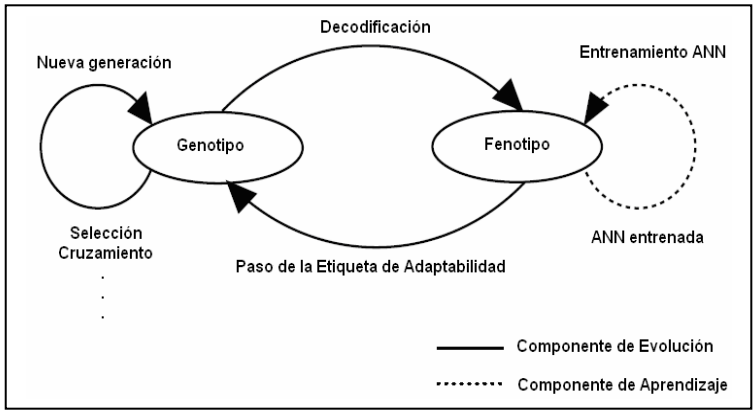


Figura 6-1. Diseño evolutivo de una Red Neuronal Artificial

Este sistema evolutivo trabaja sobre una población de genotipos, que son elementos de un espacio de búsqueda de mayor dimensionalidad. En este caso particular, el genotipo del GA es una cadena de bits, siendo esta la representación de la arquitectura de una Red Neuronal utilizando una codificación directa (más adelante se verá como se realiza esto). Se puede ver de una forma más general al genotipo, siendo este un arreglo de genes, donde cada gen toma un valor de acuerdo a un dominio, así cada genotipo está constituido por varios genes. La manipulación de dichos genes, nos permitirá realizar combinaciones para encontrar nuevas arquitecturas, siendo posible encontrar una combinación adecuada que nos minimice el error; esto ayudado también por la mutación, como ya se mencionó, para no caer en mínimos locales.

El procedimiento de evolución para la figura 6-1 es la siguiente: partimos de una población inicial (óvalo que dice genotipo), de ahí se decodifica este, en al fenotipo (Red Neuronal) para cada uno de los individuos, se entrenan y se evalúan con una función de Adaptabilidad regresando al óvalo que dice genotipo, en este punto ya tenemos un valor escalar, el cual nos permite saber qué tan adaptado está cada individuo, después aplicamos las funciones básicas del GA y obtenemos la nueva generación. Cabe señalar que se tiene preferencia por los mejor adaptados, asegurando el paso de información útil, este proceso de evolución es repetido hasta que se cumpla con alguna condición de paro, así al final se tiene al mejor individuo encontrado por el GA, el cual tiene la adaptabilidad más alta, lo que implica el error más pequeño encontrado.

6.1 Representación del Genotipo.

Existen principalmente dos formas de poder hacer la representación del genotipo: Codificación directa e indirecta. La forma de escoger la representación es crítica para los resultados, con estas codificaciones se pueden incluso buscar arquitecturas de Redes Neuronales Recurrentes como se realizó en [56] usando la Codificación directa.

6.1.1 Codificación Directa.

Este método no requiere de mucho trabajo o esfuerzo para realizar la codificación, por ejemplo si se tiene una matriz de conectividad (matriz binaria donde el elemento C_{ij} indica si hay conexión de i a j), lo único que hay que hacer es concatenar los renglones y obtener el genotipo, si contamos con más matrices, lo cual es nuestro caso, se concatenan todas las cadenas obtenidas de cada matriz. En este trabajo no se utiliza una codificación

indirecta, debido a que primero es necesario corroborar el rendimiento del algoritmo con una codificación básica.

6.2 Variables que evolucionan.

Nuestro fenotipo consiste de 5 matrices: Entradas, Retardos, *Bias*, Capas y Nodos/Capa, con las cuales podemos representar completamente a una Red Neuronal Artificial con sus entradas. Se usó la codificación directa para pasar del fenotipo al genotipo y viceversa, así el genotipo, específica en el genoma cada conexión y cada nodo que aparece en el fenotipo. La figura 6-2 nos muestra un ejemplo representativo de la codificación directa, donde por ejemplo, las entradas pueden ser los renglones y las capas de la Red pueden ser las columnas, así en este caso la entrada 1 no esta conectada a ninguna capa y la entrada 5 va a la capa 3, 4 y 5.

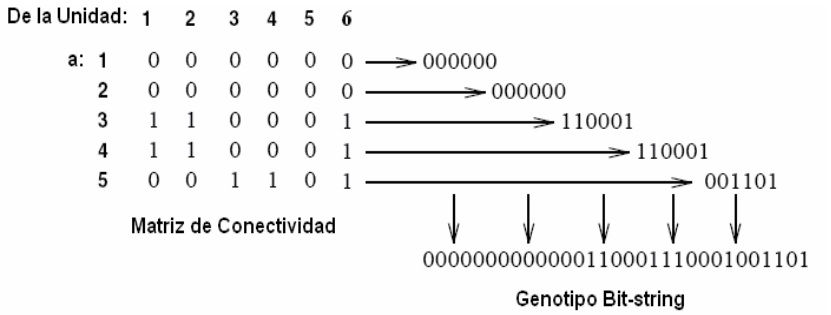


Figura 6-2. Codificación Directa

Con las últimas 3 matrices comentadas, podemos representar a una Red. Como se verá en el capítulo de Resultados, las dos primeras matrices son de gran importancia para mejorar el desempeño del algoritmo. La matriz de *bias* y capas son matrices de conectividad, donde algún elemento C_{ij} de una matriz indica la conexión de la unidad *j* con la unidad *i*, en esta implementación tomamos 1 como presencia de conexión y 0 como ausencia de la misma. Las otras matrices representadas por bits, equivalen a números naturales.

Nuestro genotipo es de longitud fija, lo que significa que tenemos matrices de tamaño fijo para representar a nuestra Redes Neuronales. Las matrices de Entradas, Retardos y Nodos/Capa están compuestas por la representación binaria de un escalar, indicando en el fenotipo cuantas entradas hay a la red, cuántos retardos de dichas entradas y cuántos nodos (neuronas) existirán en cada capa, las matriz de *Bias* indicará la presencia de *bias* o no en dicha capa y la matriz de Capas indicará la conexión entre dichas capas. A continuación se explicaran cada una de las matrices.

Entradas: Aquí se permite desde una entrada hasta siete, siendo los puntos anteriores en la ST, que determinarán al siguiente punto; el tamaño de la matriz es de 3x1. Ejemplo:

$$1\begin{bmatrix}1\\0\\0\end{bmatrix}; \qquad 5\begin{bmatrix}1\\0\\1\end{bmatrix}; \qquad 7\begin{bmatrix}1\\1\\1\end{bmatrix}$$

Al lado de cada matriz está la conversión de binario a decimal, este escalar representa las entradas de la Red.

Retardos: Esta matriz al igual que la anterior representa un escalar, la diferencia es que aquí el valor permitido va de cero a tres, esto es, que entre los valores de entrada podemos tener cero retardos, lo que significa tener los *n* valores de entrada anteriores para determinar el siguiente. Si el retardo es de uno vamos a

tomar un dato si y un dato no hasta completar los *n* valores de entrada, y así sucesivamente para los restantes. El tamaño de la matriz es de 2x1. Ejemplo:

$$0\begin{bmatrix}0\\0\end{bmatrix}; \qquad 1\begin{bmatrix}1\\0\end{bmatrix}; \qquad 2\begin{bmatrix}0\\1\end{bmatrix}$$

Bias: La matriz de *Bias* nos indica la presencia o ausencia de *bias* en una capa determinada, siendo su tamaño de 3x1, por lo que si una capa presenta conexión, se le sumará el *bias*, el cual tiene el valor de "1", esta es una matriz de conectividad. Ejemplo:

$\begin{bmatrix}0\\1\\1\end{bmatrix}$, la capa 1 no tiene *bias*, las capas 2 y 3 si cuentan con *bias*.

Capas: Esta también es una matriz de conectividad y nos indica la conexión que existe entre las capas siendo su tamaño de 3x3, originalmente diseñada de este tamaño, se permiten ciclos (Redes feed-back) [56], aunque en este trabajo no se permiten, por lo que únicamente son funcionales los elementos (2,1), (3,1), (3,2). Cabe señalar que en esta matriz no se permite que esté en ceros, esto es que únicamente exista la capa de entradas y ella misma sea la capa de salida, lo mínimo permitido es que exista una capa intermedia sin contar la de salida. Ejemplo:

$\begin{bmatrix}0\,0\,0\\1\,0\,0\\1\,1\,0\end{bmatrix}$, la capa 1 está conectada con la capa 2 y 3, la capa 2 está conectada con la capa 3.

Aquí la capa 3 es la capa de salida. Si se quisieran Redes feed-back, entonces se tomarían en cuenta también, todas las demás posiciones en donde aparecen ceros, por lo que si en alguna de estas posiciones aparece un uno, este se tomaría como un bucle de la unidad *j* a la *i*.

Nodos/Capa: Aquí se permiten hasta 15 nodos por cada capa, la última capa siempre va a tener un nodo de salida, para predecir un punto a la vez, así se empleara la predicción directa o iterada para predecir más puntos. El tamaño de la matriz es de 2x4. Ejemplo:

$\begin{bmatrix}0\,1\,1\,0\\1\,0\,1\,0\end{bmatrix}$, la capa 1 tiene 6 nodos y la capa 2 tiene 10 nodos.

Para este caso si la matriz de Capas nos indica que existen tres capas, entonces la capa 1 tiene 6 nodos, la 2 tiene 10 nodos y la última capa, que es la de salida tiene 1 nodo; si en la matriz de Capas únicamente hay dos capas, es decir que la línea 3 este en ceros, entonces la capa 1 tendrá 6 nodos y la capa 2, que ahora es la de salida, tendrá 1 nodo y no se toma en cuenta la segunda línea de la matriz de nodos/capa.

6.3 Espacio de Búsqueda.

Entre las 5 matrices que se tienen para nuestra Red Neuronal, podemos sacar el combinatorial de estas y ver en el tamaño del espacio de búsqueda en el que nos estamos moviendo. Claro está que el tamaño del espacio de soluciones es igual para cada problema en este caso, lo que significa una posible limitante del algoritmo, pero como se verá a continuación, el tamaño parece ser lo suficientemente grande, para poder adaptarse a diversos problemas y tal vez no presentar una limitante.

- Para la matriz de Entradas se tienen 3 bits lo que nos da $2^3 = 8$ pero no se permite que hayan 0 entradas, por lo que nos quedan 7 opciones.
- Retardos tiene 2 bits, $2^2 = 4$ opciones (de 0 a 3).
- bias tiene 3 bits, $2^3 = 8$ opciones.
- La matriz de Capas tiene otros 3 bits, para nuestro caso nos da 7 opciones si tomamos en cuenta que al menos debe de existir una capa oculta (restricción antes mencionada).
- Nodos/Capa tiene 4 bits para cada capa, por lo que tenemos $4^2 = 16$ opciones para una capa, pero como se permiten hasta 2 capas podemos tener la combinación entre ambas, es decir 16 x 16 opciones posibles.

Debido a esto, la combinación total es la multiplicación de los valores obtenidos: 7 x 4 x 8 x 7 x 16 x 16 = 401,408 candidatos posibles a la solución de acuerdo a lo establecido. En los experimentos, el GA se estableció a 300 generaciones (iteraciones) con 20 individuos por población, por lo que el número total de individuos muestreados es de 300 x 20 = 6000.

Conociendo esto, podemos calcular el porcentaje de individuos muestreados, siendo éste de 1.494 %; este número es muy bajo, pero debido a que la búsqueda de arquitecturas de Redes Neuronales es una tarea complicada, y no podemos aumentar demasiado dichos valores. Además, si nosotros obtenemos resultados adecuados con esta configuración en el GA, se estaría comprobando la eficacia que tiene para la búsqueda, al obtener resultados razonables con un porcentaje de búsqueda tan bajo.

6.4 Métodos de Evaluación.

Las funciones de costo escogidas para evaluar las Redes Neuronales fueron los siguientes:

Raíz Cuadrada del Error Cuadrático Medio (RMSE)

$$RMSE = \left[\frac{1}{N} \sum_{i=1}^{N} \left(X_i^p - X_i^o \right)^2 \right]^{\frac{1}{2}} \qquad \textbf{(6.1)}$$

donde N es el número de datos predichos, X_i^p corresponde al i-ésimo dato predicho y X_i^o corresponde al i-ésimo dato original.

Raíz Cuadrada del Error Cuadrático Normalizado (NRMS).

$$NRMS = \left[\frac{\sum_{i=1}^{N} \left(X_i^p - X_i^o \right)^2}{\sum_{i=1}^{N} \left(X_i^o - \overline{X^o} \right)^2} \right]^{\frac{1}{2}} \qquad \textbf{(6.2)}$$

donde N es el número de datos predichos X_i^p corresponde al i-ésimo dato predicho, X_i^o corresponde al i-ésimo dato original y $\overline{X^o}$ es la media del conjunto de datos originales.

También se ocuparon otras dos funciones para evaluar la predicción, Error Absoluto Promedio y Error Absoluto Máximo. Estas funciones fueron ocupadas en las dos primeras aproximaciones del algoritmo mencionado en la sección 7.1 y 7.2.

Error Absoluto Promedio

$$EAP = \frac{1}{N}\sum_{i=1}^{N} ABS\left(X_i^p - X_i^o\right) \quad \textbf{(6.3)}$$

Error Absoluto Máximo

$$EAM = \max\left(ABS\left(X_i^p - X_i^o\right)\right) \quad \textbf{(6.4)}$$

Al iniciar el algoritmo, los datos se dividen primero en dos conjuntos: el de entrenamiento y el de prueba; posteriormente, dentro del GANN, el conjunto de entrenamiento antes mencionado se vuelve a dividir en otros tres conjuntos, los cuales son:

- El primero es usado para el entrenamiento donde se computa el gradiente y actualizan los pesos de la Red, siendo su tamaño de un 80% del tamaño total del conjunto del que proviene.
- El segundo es un conjunto de validación donde su error es monitoreado durante el entrenamiento, este error de validación tiende a decrecer durante el entrenamiento, pero cuando la Red empieza a sobre entrenarse, este error empieza a subir, cuando el error de validación sigue creciendo en un determinado número de iteraciones, el entrenamiento es parado y se ocupan los pesos y *bias* anteriores a cuando empezó a crecer el error de validación. Esto permite obtener una buena generalización de la Red y evita un sobre entrenamiento de los datos, su tamaño es del 20%.
- Acabado el entrenamiento, se introduce el tercer conjunto que es el de prueba, de ahí se obtiene un error que se convertirá en la adaptabilidad de dicho individuo, este tamaño se estableció a 30 datos o puntos, ya que son los puntos base a predecir en el trabajo; así primero se separa este conjunto de prueba y después se obtiene el de entrenamiento y validación

La adaptabilidad obtenida es regresada al GA permitiendo que siga la evolución, esto es repetido para cada individuo en la población; cuando todos los individuos han sido evaluados, se aplican las reglas del GA para generar a la siguiente población. La Figura 6-1 muestra el ciclo del GAAN, mas detalladamente se puede apreciar el algoritmo completo en 6-3. En la figura 6-4 podemos apreciar cómo se obtienen los diferentes conjuntos de datos y se realiza la predicción en el GANN.

Una vez que ha terminado el GA, se introduce el primer conjunto de prueba, que se separó al inicio del algoritmo, de aquí se obtiene la predicción final presentada en los resultados.

Cabe señalar que la mayoría de series utilizadas en el trabajo, son bastante complicadas o difíciles de predecir, esto debido a que es muy difícil obtener patrones, relaciones o tendencias que puedan ayudar a obtener predicciones muy precisas en comparación con las series fáciles de predecir, que tienen una dinámica no tan compleja, y este sería el caso de la Serie Seno, que es periódica y estacionaria. En la sección de anexos se muestran todas las series utilizadas en el trabajo.

La implementación y programación del algoritmo presentado en este trabajo de tesis, fue realizado en Matlab, versión 7.0.0.19920 (R-14), apoyándonos en las cajas de herramientas de Redes Neuronales Artificiales y Algoritmo Genético. A continuación se presenta un diagrama, que muestra el comportamiento del algoritmo.

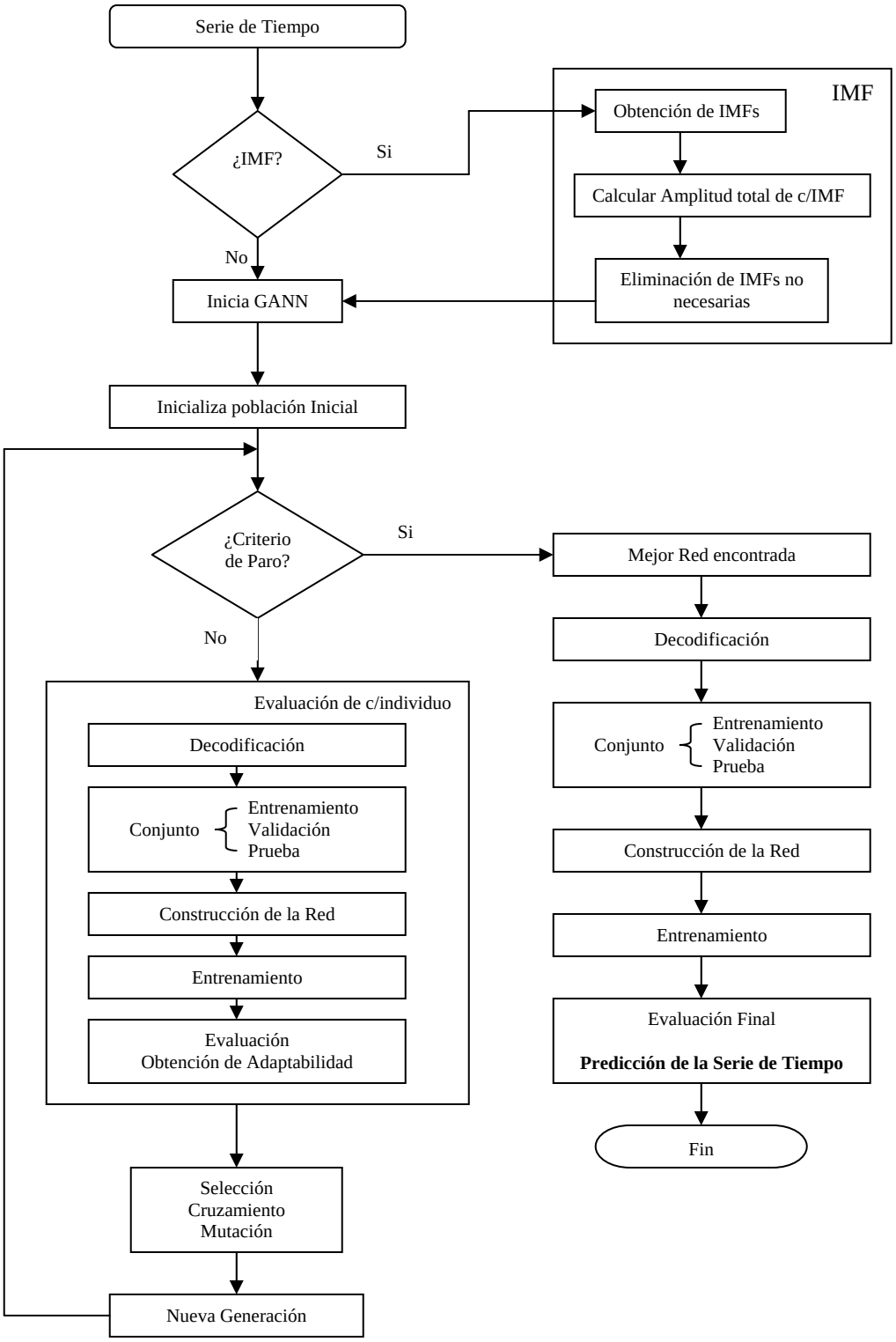


Figura 6-3. Flujo del algoritmo GANN

Los cuadros de proceso que tienen el texto “Decodificación”, se refiere a obtener del genotipo, las matrices y valores escalares que nos permiten construir nuestra Red Neuronal.

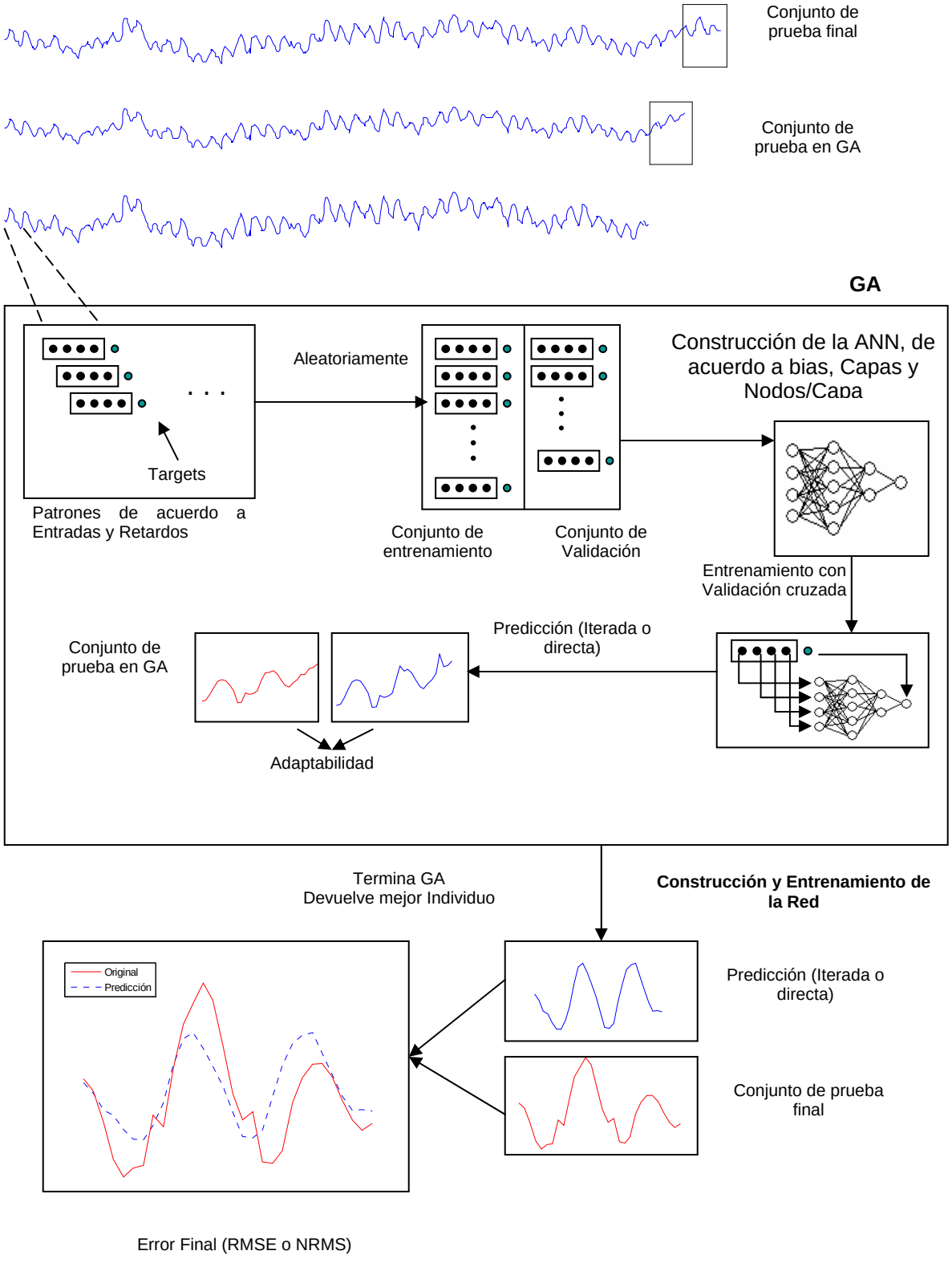


Figura 6-4. Conjuntos de datos y predicción en el GANN

No se decidió hacer una cuantificación de los resultados, en el aspecto de poder determinar, en que porcentaje el algoritmo GANN puede predecir correctamente el primer punto, el segundo, etc. Un análisis matemático clásico de dicha clase, seria valido cuando conocemos la función matemática en cuestión, de esa forma podemos asegurar que siempre se cumplirá la cuantificación realizada para esa función, pero para nuestro caso, cada problema representa una función diferente, más aun, dicha función es desconocida (mapeo no-lineal de entradas-salidas realizado por la Red), de esa forma, al llegar una Serie diferente, tendrá una función diferente y desconocida, con la cual puede darse el caso, de no concordar con dicha cuantificación.

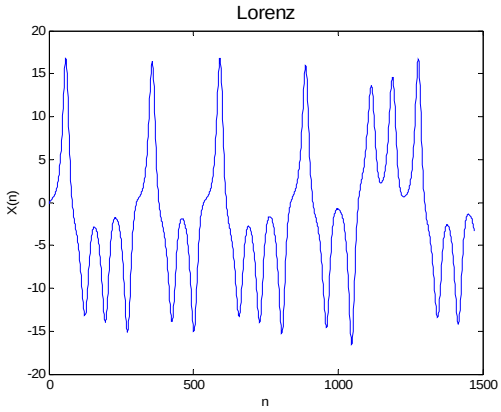


Figura 6-5. Serie Lorenz

Ejemplo: Supongamos que tenemos la Serie Lorenz, y realizamos una cuantificación como se comento anteriormente, teniendo en cuenta, que conocemos tanto su función generadora, como el mapeo no-lineal realizado por la Red Neuronal, que la predice de una forma lo suficientemente adecuada; con esta información podríamos asegurar que siempre se cumplirá la cuantificación realizada para dicha función, éste valor lo podríamos asociar también al algoritmo GANN, ya que sería el quien abría encontrado la ANN.

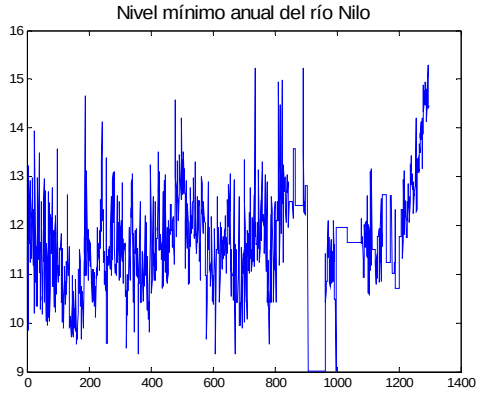


Figura 6-6. Serie Nivel mínimo anual del río Nilo

Ahora, supongamos que queremos predecir la Serie Nivel mínimo anual del río Nilo, de dicha serie no conocemos su función generadora, y al obtener la mejor Red encontrada por el GANN, tendremos una función que realiza un mapeo no-lineal de entradas-salidas desconocido; si realizamos la misma cuantificación que para la Serie Lorenz, seguramente no obtendremos los mismos resultados que para la serie de la figura 6-5, entonces la cuantificación únicamente nos sirve para cada serie particular, y no es posible utilizar dicho valor de forma global (para el algoritmo GANN); más aun, el valor que cuantifica a la Serie de la figura 6-6, estaría asociado con una función completamente desconocida, dado el hecho, de que una Red Neuronal es una caja negra. Esto que se comenta para una ST, se aplica directamente para un conjunto de ST predichas, y este es el caso de dicho trabajo de tesis, por lo que no es posible asociar un factor de precisión a nuestro algoritmo GANN, ésta afirmación se encuentra sustentada también por el NFL.

En este capítulo observamos como interactúan las Redes Neuronales con el Algoritmo Genético y con EMD o los datos originales, para poder obtener una arquitectura de Red que permita realizar una predicción lo suficientemente adecuada como para tomarla como tal, es decir, tener una predicción que sea válida y en la que se pueda confiar en la vida real. Se observó también la forma de realizar la codificación de las Redes Neuronales y se mencionó el tamaño del espacio de soluciones, junto con el total de individuos muestreados por el GA, el cual es muy bajo, lo que implica el poder que tiene el GA para realizar una búsqueda adecuada cuando se proporciona la información correcta a evolucionar (variables adecuadas). Asimismo se observó como obtener los diversos conjuntos de prueba y las diferentes expresiones utilizadas para evaluar el rendimiento de las Redes. El próximo capítulo estará destinado a los resultados experimentales, donde se podrán apreciar las diferentes etapas del algoritmo.

Capítulo VII. Resultados Experimentales.

En este capítulo mostraremos y comentaremos los resultados obtenidos de la versión final del algoritmo, así como algunos experimentos realizados para llegar a ésta. Esto es para darle al lector, el panorama de la evolución del algoritmo, lo que permite dar a conocer las pruebas que no funcionaron o que tuvieron un desempeño promedio. Aparte, de que esto sirve como guía para describir el camino que se siguió en la resolución de nuestro objetivo principal.

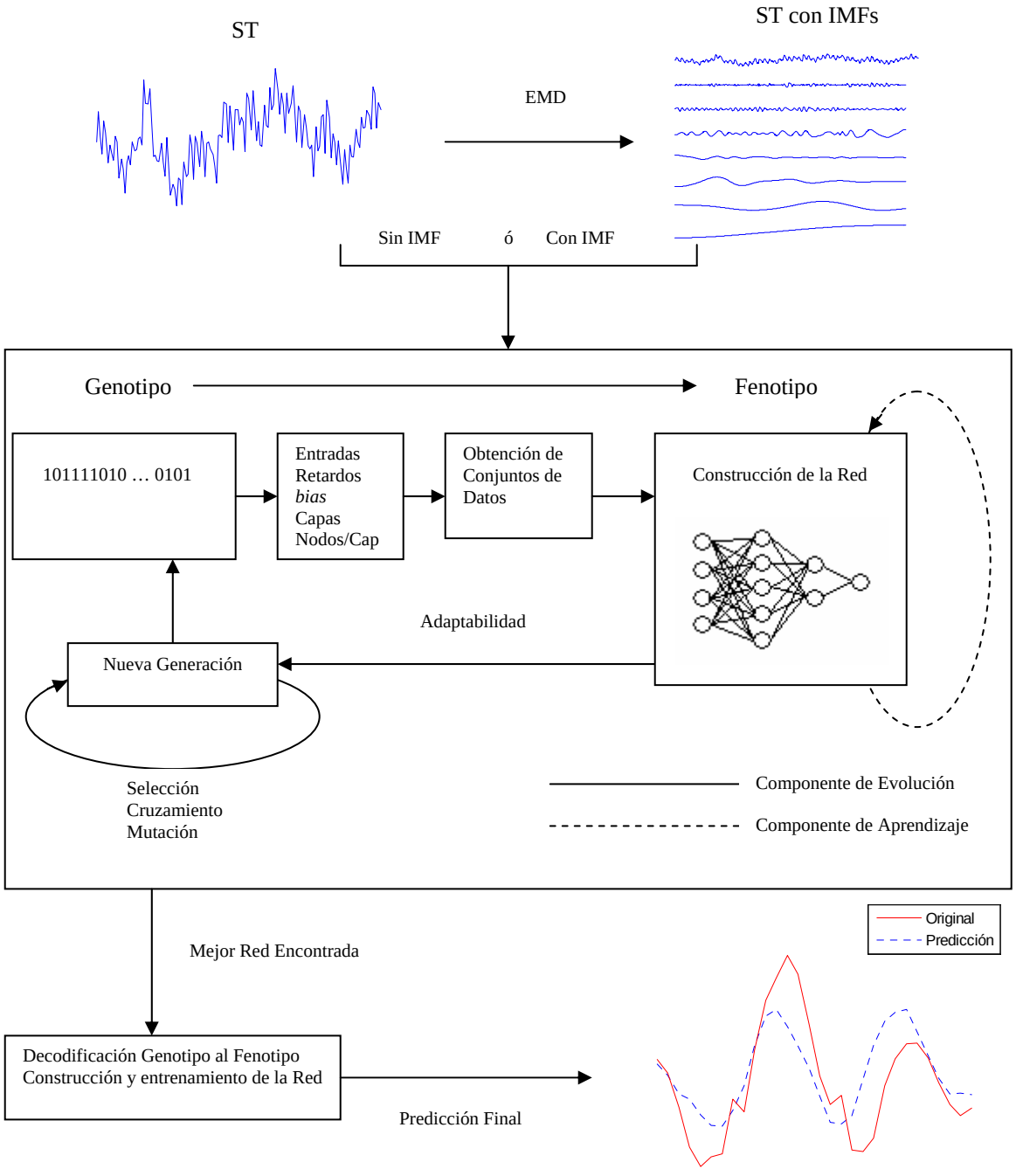


Figura 7-1. Algoritmo GANN para la predicción de ST.

En la figura 7-1 se presenta un diagrama explicativo del algoritmo en general; éste es similar al presentado en la figura 6-3 y 6-4, la única diferencia es la forma en que se presenta, esto con el objetivo de tener más figuras diferentes que sirvan para tener un mejor entendimiento de lo que está sucediendo en el algoritmo, y más precisamente observar como están evolucionando las arquitecturas de las Redes. Así pueden entrar los datos originales o bien los datos originales con las IMFs al algoritmo, posteriormente, en la primera generación del GA se crea aleatoriamente la primera población (genotipo), de ahí se decodifica al fenotipo (Red Neuronal) para cada individuo, de acuerdo a las matrices y valores escalares obtenidos del genoma, necesarios para construir la Red; posteriormente se entrena cada una de ellas y se introduce un conjunto de prueba, de donde saldrá su adaptabilidad, con esta información se pueden aplicar los operadores del GA (Selección Cruzamiento, etc.) para crear a la siguiente generación, y así el procedimiento es repetido hasta que se cumpla con alguna condición de paro. En ese momento el algoritmo nos regresa el mejor individuo encontrado, pero debido a que nos regresa el genoma, es necesario decodificarlo, entrenarlo y probarlo con el conjunto de prueba final.

En los primeros experimentos, no se introducen todas las series disponibles para predecirlas; esto debido a que primero es necesario realizar modificaciones al algoritmo, para ir mejorando las predicciones; por lo que resulta innecesario predecir todas las ST desde un principio, ya que con unas cuantas, se pueden ir identificando algunos problemas; por lo que en cada experimento se van aumentando el número de series empleadas, hasta llegar a la versión final. Las series empleadas son de muy diversos orígenes y tipos, por lo que se puede considerar que la mayoría de estas, son una muestra representativa de series difíciles de predecir, es decir, son series en las que cuesta demasiado trabajo poder obtener sus patrones o tendencias para poder hacer una predicción bastante precisa.

Si la serie de tiempo predicha sigue la tendencia de la original o existe una relación a simple vista en cuanto al comportamiento de ambas, aunque no exista una mejor predicción (otra que dé el error más bajo), ésta se tomará como una predicción buena o aceptable, lo que implica que la predicción es confiable para usarla como tal. En caso contrario, si la predicción no tiene relación alguna con la original, se tomará como una predicción incorrecta.

Cabe recordar, que la mayoría de las series empleadas, son complicadas de predecir, ya que es muy difícil extraer información útil que nos pueda ayudar a predecir de una forma muy precisa, lo que significa que la mayoría son series complejas.

7.1 Descripción del Método Experimental.

Una de las primeras aproximaciones y de las más simples, fue el de entrenar las Redes Neuronales con un número fijo de épocas, esto debido a la idea, de que se podía controlar el aprendizaje (en cuanto al número de épocas) de ellas de forma manual, pero como es de esperar, no en todos los casos podíamos ocupar las mismas épocas. Otro de los problemas que se presentan al dejarlas fijas, es el sobre entrenamiento, como ya se trató en la sección 3.6; para resolver esto se ocupó una validación cruzada para parar automáticamente el entrenamiento de la Red.

La descripción del método está dividido en tres partes, donde se presentará la evolución del algoritmo, siendo la sección 7.2 la primera etapa; así la segunda y tercera etapas mejoran a sus antecesores, obteniendo mejores resultados cada vez. Se realizó de esta forma, para dar un panorama más claro de lo que ha funcionado y de lo que no ha funcionado, ya que en la literatura casi no se explican las configuraciones que no dan buenos resultados. La figura 7-2 muestra la evolución del algoritmo GANN, donde se presentan las diferentes etapas por las que pasó, FF significa feed-forward y FB significa feed-back, los demás términos ya han sido mencionados anteriormente en el trabajo.

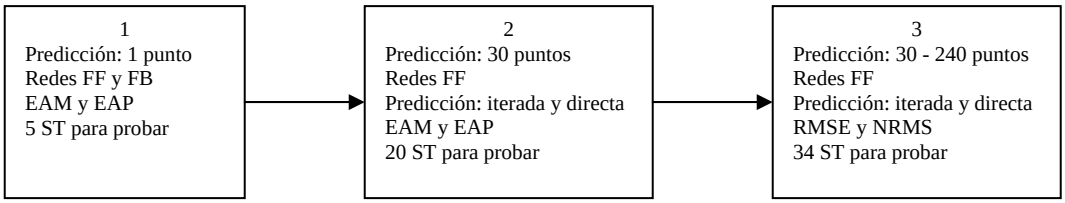


Figura 7-2. Evolución del Algoritmo GANN.

7.2 Primera Etapa.

Una vez resuelto el problema de las épocas, se empezó con el diseño del Algoritmo Genético para evolucionar topologías de Redes, que también se puede ver como el diseño de las mismas. El primer objetivo planteado, fue el de poder predecir a un paso adelante en la serie, usando la siguiente configuración:

ANNs
- Se permiten Redes feed-forward y feed-back
- Las entradas no forzosamente tienen que ir a la primera capa, puede ir a más de una.
- Número de entradas variables, hasta 13 entradas se permiten.
- Se permite hasta 2 capas ocultas con un máximo de 7 nodos por capa
- Se utiliza predicción directa a un paso adelante
- No hay restricción en la cantidad de datos para entrenar

GA
- 100 generaciones
- Paro por tiempo = inf

EMD
- Se utilizan las primeras 5 IMFs

Evaluación
- Error Absoluto Promedio
- Error Absoluto Máximo

A diferencia de lo mencionado en el capítulo VI, donde se comentó que había 7 entradas y hasta 16 nodos por capa, aquí se decidió utilizar los valores comentados anteriormente (hasta 13 entradas y 7 nodos por capa) porque se desconocía cómo iniciar el algoritmo, así lo mencionado en el capítulo VI corresponde a la versión final del algoritmo GANN.

Se decidió usar un máximo de 5 IMFs, porque es el menor número de componentes encontradas, al aplicar EMD a las series. Como se desconoce qué porcentaje de éstas es funcional para la predicción, se decidió usar el mínimo, para ir realizando las modificaciones al código; así en la primea y segunda etapa se ocupan únicamente 5 IMFs para las predicciones.

Se emplearon 5 Series de Tiempo de diferente origen para probar esta primera etapa, éstas son: 1) Temperatura máxima en Melbourne, 2) Temperatura mínima en Melbourne, 3) Precipitación diaria en Hveravellir, 4) Precio diario del oro, 5) Nivel mínimo anual del río Nilo.

La comparación, es llevada acabo con las series predichas con IMF y sin IMF (únicamente datos originales). Los resultados obtenidos se pueden apreciar en la tabla 7.1 y 7.2. El error más bajo por renglón es remarcado, para una mejor apreciación.

Tabla 7.1. Resultados con Error Absoluto Máximo (EAM)

Serie de Tiempo	Sin IMF	Con IMF
Temperatura max. Melbourne	15.804	**8.3562**
Temperatura min. Melbourne	5.9309	**3.5664**
Precipitación diaria Hveravellir	21.6	**17.851**
Precio diario del oro	**8.9345**	11.783
Nivel mínimo anual del río Nilo	2.3641	**1.3782**

Tabla 7.2. Resultados con Error Absoluto Promedio (EAP)

Serie de Tiempo	Sin IMF	Con IMF
Temperatura max. Melbourne	4.3116	**2.6022**
Temperatura min. Melbourne	1.8775	**1.5024**
Precipitación diaria Hveravellir	2.7118	**1.9534**
Precio diario del oro	**2.5932**	3.3267
Nivel mínimo anual del río Nilo	0.6929	**0.2998**

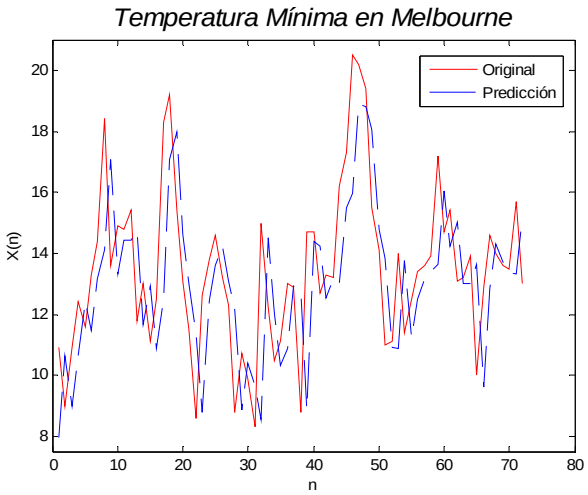


Figura 7-3. Temperatura mínima en Melbourne con IMF y EAP.

Todas la gráficas presentadas en el trabajo tienen el eje *x* como el eje del tiempo, es decir que *n* corresponde a un tiempo igual a t+1, t+2, … como predicciones. En el eje *y* no se introducen las unidades ya que no en todos los casos se disponía en ellos, sin embargo, podemos ver cómo en la figura 7-3 que el eje *y* corresponde a grados centígrados, porque se trata de temperatura, y no corresponde a grados farenheit debido a la escala; las series que cuenta dicha información, pueden ser revisadas en los anexos.

El error mostrado en la tabla 7.1 y 7.2 corresponde a un conjunto de prueba de 72 muestras; de esta parte claramente se puede apreciar que es mejor utilizar IMF para realizar la predicción a un paso adelante. Algunas de las predicciones aquí mostradas tienen una Red feed-back, se decidió probar con dichas Redes, para ver el desempeño que estas tenían, la desventaja es que no se puede ocupar ésta arquitectura para predecir más de un paso adelante, debido al retardo o retroalimentación que pueden presentar en capas posteriores, lo cual no ayuda a la hora de iterar o repetir el procedimiento para realizar predicciones a mayor plazo, de la única forma en que son útiles, es cuando las entradas son continuas y tienen una secuencia.

De la figura 7-3 se pensaría que es una predicción a 72 pasos adelante, más sin embargo, siempre se realiza una predicción a un paso adelante, esto es, que predecimos con el último dato el siguiente, ya teniendo la predicción, introducimos el siguiente dato original (no el predicho, pero corresponde a la posición del dato que se acaba de predecir) luego predecimos el siguiente y así sucesivamente; esto es algo que algunos autores realizan como lo mencionado en el capítulo 2, en los artículos [4, 5]; ya que mostrando gráficas como estas, le da la impresión al lector de que el algoritmo es muy preciso prediciendo a tan largo plazo. Los siguientes parámetros se mantienen iguales a lo largo de los diferentes experimentos por lo que únicamente se mencionarán aquí. Los parámetros de las Redes son escogidos de esa forma, porque se conoce que funcionan para la preedición de acuerdo a la literatura; los parámetros del GA simplemente se dejan fijos, de otro modo, tendríamos que optimizar también dichos valores y esto es algo que se encuentra fuera del alcance, de este trabajo de tesis.

ANNs

- Función de transferencia tangente sigmoidal en capas intermedias y lineales en la capa de salida.
- El *bias* es opcional en las capas.
- Algoritmo de entrenamiento Levenberg-Marquardt.
- Un nodo en al capa de salida.
- Evaluación del rendimiento en el entrenamiento: MSE.
- Validación cruzada para evitar sobre entrenamiento y tener generalización.

GA

- Población de cadena de bits.
- 20 individuos por población.
- 2 individuos de Elite.
- Fracción de cruzamiento = 0.8.
- Intervalo de Migración = 20.
- Fracción de Migración = 0.2.
- Función de escalamiento tipo Rank.
- Función de selección tipo Estocástica uniforme.
- Función de Cruzamiento tipo Disperso.
- Función de Mutación tipo Gausiana.
- Codificación directa como se menciono en el capítulo VI.

7.2.1 Análisis y discusión de la Primera Etapa.

Debido a la forma en que se realiza la predicción, no tiene mucho sentido poner las gráficas de todas las ST, algunas de las gráficas son muy parecidas a la figura 7-1 en cuanto a su comportamiento; otras sin embargo no pudieron converger a una solución adecuada como es el caso de la Serie Temperatura máxima en Melbourne con Error Absoluto Máximo y los datos originales. A continuación se presentan algunas desventajas encontradas en esta primera aproximación.

Primera: No se tiene claro el número de entradas necesarias para predecir el siguiente punto adelante. Los experimentos de esta parte, están conformados por la repetición del mismo experimento con 1 hasta 13 entradas, las tablas y la gráfica mostradas, son de los resultados con 9 entradas a la Red; esto es debido, a que fueron los datos más precisos, con el error más bajo obtenido, para la mayoría de los casos, pero el problema sigue presente, se puede repetir para los siguientes experimentos con una entrada, hasta *n* o bien buscar otra forma de poder resolver dicho problema.

Segunda: Solamente se realiza la predicción a paso adelante. En algunos casos puede ser de gran utilidad conocer únicamente el siguiente valor en la serie, pero en otros es necesario conocer un poco más, por lo que es necesario hacer modificaciones para lograr predecir más puntos en la ST.

Tercera: No se puede seguir utilizando Redes feed-back, debido a que es necesario una introducción secuencial de los vectores de entradas, para que los retardos en la Red tengan efecto.

Cuarta: Debido a que se tiene también la predicción iterada, como se mencionó anteriormente, surge la duda de saber si alguna (predicción iterada y directa) nos puede dar un mejor resultado en las predicciones.

Esta etapa sirvió para poder mostrar lo que algunos autores ponen en sus trabajos y para hacer predicción a un paso adelante. Hasta este punto, estas fueron algunas de las limitantes encontradas, por lo que se busca una forma de resolverlas en la Segunda Etapa.

7.3 Segunda Etapa.

El número de entradas de la Red se resolvió con el GA, dejando que este sea el encargado de determinar dicho número. El número máximo de entradas se estableció a 7 por dos cuestiones: la primera, es porque los resultados del experimento de la sección 7.2 con 7 y 9 entradas eran muy similares y segundo, porque al aumentar el número de variables a optimizar, se incrementa el espacio de soluciones, por eso se decide ocupar un máximo de 7 entradas.

También se introdujo un retardo en los datos de entrada, comentado al final del Capítulo 3, los cuales pueden de ser de hasta 3 y se explica más detalladamente en el Capítulo VI.

Aquí sólo se permite que los datos de entrada únicamente vayan a la primera capa oculta, a diferencia de la primera etapa, donde se permite que vayan a cualquier capa; de no hacerlo así, el número de multiplicaciones en las capas intermedias se eleva demasiado, trayendo como consecuencia un mayor tiempo de ejecución, por lo que no resulta práctico el utilizarlo. Se aclara, que no se ha demostrado experimentalmente, que sea mejor o peor dejar que las entradas puedan conectarse con cualquier capa, aparte de la de entrada.

El número de nodos en las capas ocultas se aumentó a un máximo de 15, tratando de observar si existe una mejora en los resultados, dando más parámetros libres en las Redes.

La segunda desventaja de la prueba anterior se resolvió modificando el algoritmo para que realizara la predicción de 30 puntos adelante en la ST, utilizando la predicción iterada y directa con el Error Absoluto Promedio y el Error Absoluto Máximo; así introducimos predicciones a mediano plazo con dos técnicas diferentes de predecir (cuarta desventaja).

Por último sólo se ocuparon Redes feed-forward para probar los algoritmos, debido a que los datos de entrada son introducidos a la Red de una forma aleatoria para el entrenamiento y para utilizar el otro tipo de

Red (feed-back), se necesita que la introducción de los datos sea secuencial, esto significa que las Redes feed-back pueden resolver problemas de dependencia temporal [22].

Se limitó el tamaño de las ST para realizar la predicción, esto es para probar si es posible obtener predicciones precisas con un conjunto de entrenamiento más pequeño. El inconveniente que se puede presentar, es el perder información que sea útil para construir la Red y para predecir, se decidió no utilizar más de 1500 datos por serie.

El número de generaciones en el GA se aumentó a 300 y se mantuvo a los 20 individuos por población; se utilizaron otras ST y se probó el algoritmo con la Serie Seno, la cual por su simplicidad tenía que ser fácilmente predicha por nuestro algoritmo GANN, por lo que resumiendo se tiene la siguiente configuración de parámetros en el algoritmo:

ANNs
- Solo Redes feed-forward
- Las entradas únicamente van conectadas a la primera capa
- Número de entradas variables, hasta 7 entradas se permiten.
- Se permiten hasta 3 retardos en las entradas
- Se permite hasta 2 capas ocultas con un máximo de 15 nodos por capa
- Se utiliza predicción iterada y directa
- Se utiliza un máximo de 1500 datos de la ST, 80% para entrenar y 20% para probar

GA
- 300 Generaciones
- Paro por tiempo = 600

EMD
- Se utilizan las primeras 5 IMFs

Evaluación
- Error Absoluto Promedio
- Error Absoluto Máximo

Se introdujo el Paro por tiempo = 600, ya que de no hacerlo, el algoritmo se puede dilatar hasta semanas para encontrar a una solución para una serie, lo cual no resulta práctico para probar con varias de estas. A continuación se muestran las tablas 7.3 y 7.4 de los experimentos obtenidos en esta etapa.

Tabla 7.3 Predicción con EAM y Rangos de las ST

#	Series de Tiempo	Predicción Directa		Predicción Iterada		Rangos	
		Sin IMF	Con IMF	Sin IMF	Con IMF	Min	Max
1	Agregado semanal: Índice en horas	2.8202	18.952	3.0299	**2.0833**	50.9	103.9
2	Brownian Motion	**1.7332**	6.7069	3.1659	6.3877	-3.923	6.216
3	Flujo diario del río Jokulsa	9.2983	6.2324	16.377	**3.3092**	22	143
4	Down Jones	154.29	264.8	**132.24**	185.12	31.8	59923
5	Precio diario del oro	27.501	13.341	30.243	**12.23**	285	593.7
6	HDNA	1.6215	1.7942	**1.5**	1.8984	1	4
7	Precio de las acciones de IBM	4.5984	3.4981	5.9715	**3.1074**	49	175
8	Nivel de agua mensual del lago Eriel	2.9449	4.0695	2.9719	**2.6646**	10	20
9	Lorenz	0.1342	12.224	**0.0609**	9.618	-18.79	18
10	Mackey-Glass	0.1303	0.8951	**0.0011**	0.0103	0.177	1.396
11	Nivel mínimo anual del río Nilo	9.552	2.6645	3.6744	**1.5412**	9.01	15.3
12	Precipitación diaria Hveravellir	5.176	5.7894	**4.727**	5.8023	0	79.3
13	QP2	0.1130	0.2584	**0.0952**	0.3421	2.605	4.391
14	Seno	0.0004	---	0.0229	---	-1	1
15	Southern Oscillation	26.644	39.16	26.359	**25.122**	-38.8	33.1
16	SP500	4.6199	17.178	**3.8727**	6.7105	98.22	425.2
17	No. Manchas solares mensuales	**39.7**	73.012	43.601	51.316	0	253.8
18	Temperatura max. Melbourne	18.471	**17.048**	19.365	21.4547	7	43
19	Temperatura min. Melbourne	7.1061	6.7429	6.425	**5.8792**	-0.8	26.3
20	Salarios diarios de Inglaterra	20.831	25.808	**19.854**	23.9	2.15	49.99

Tabla 7.4 Predicción con EAP

ST	Predicción Directa		Predicción Iterada	
	Sin IMF	Con IMF	Sin IMF	Con IMF
1	2.0371	2.5053	**0.9257**	3.1375
2	0.6843	3.0247	0.8618	**0.3389**
3	4.9004	2.2541	**1.3561**	4.2622
4	66.353	71.145	**55.547**	71.337
5	8.7774	**4.5649**	33.168	11.228
6	0.9435	**0.9239**	1.0033	1.2332
7	2.5184	1.9933	2.5416	**1.6168**
8	1.4378	1.7967	**0.6317**	2.7923
9	2.5975	3.4424	**0.4410**	0.6992
10	0.0326	0.1639	**0.0003**	0.0119
11	2.1325	1.4056	**0.4861**	0.8327
12	2.182	1.8904	1.6097	**1.0485**
13	0.0207	0.1145	**0.0141**	0.0671
14	0.0001	0	0.0002	0
15	10.322	15.293	9.4884	**8.8564**
16	2.3934	5.5888	3.2955	**2.1649**
17	34.659	49.508	**16.233**	31.788
18	5.0597	5.4047	**5.0447**	5.1209
19	2.2243	2.8439	**1.7493**	4.399
20	8.7132	11.311	8.1119	**7.8521**

Ahora veamos cómo es que vamos a realizar la comparación: primero, nosotros no podemos comparar peras con manzanas (cosas distintas), así que no nos enfocaremos a comparar los resultados de las dos tablas (con EAM y EAP), habiendo aclarado esto podemos continuar. Lo que sí podemos comparar son los renglones de una tabla, ya que todos corresponden a un mismo error; ahora bien, con la configuración de experimentos que tenemos, podemos determinar el que haya realizado la predicción más precisa, es decir, la predicción que nos proporcione el error más pequeño, así por ejemplo:

> Si una serie se puede predecir muy bien (error más pequeño) con predicción iterada con IMF, entonces todos los demás casos los podremos desechar, es decir, no nos sirven los experimentos con predicción directa con y sin IMF ni tampoco la predicción iterada sin IMF, ya que estos fueron los experimentos en los que el GANN, no pudo encontrar una Red lo suficientemente adecuada, para poder obtener un error de predicción más pequeño en comparación con la predicción Iterada con IMF.

De la tabla 7.3 se muestran las predicciones con Error Absoluto Máximo, las ST empleadas, ambos tipos de predicción con y sin IMF y los rangos en los que la ST está. Los rangos se incluyeron para dar una mejor idea de cómo se está realizando la predicción, así para el Error Absoluto Máximo de la ST 1 el error más pequeño fue de 2.0833 (predicción más precisa), los mejores errores por renglón serán remarcados, para una mejor apreciación.

Resumiendo los resultados de la tabla 7.3 tenemos 9 ST donde la mejor predicción se hace con IMF y 10 ST sin IMF, tomando en cuenta la predicción directa e iterada en ambos casos. Para la tabla 7.4 tenemos 8 ST que tienen una predicción más precisa con IMF y 11 series que fueron más precisas sin IMF. Estos resultados los podemos ver desde el siguiente punto de vista: si no hubiéramos aplicado IMF a esos experimentos (los mejores obtenidos con IMF de las tablas 7.3 y 7.4) no hubiéramos obtenido una predicción tan precisa como la conseguida, más aún, IMF nos da predicciones más precisas en los primeros puntos, para varias Series de Tiempo. Si sólo evaluamos los primeros puntos, tendríamos más series que presentarían un error más pequeño con IMF, como se mostrará en las siguientes figuras.

El lector se preguntará, por qué no comentar los resultados por columnas, es decir, para la tabla 7.3 con predicción directa sin IMF se obtuvieron 2 ST mejor predicas, con IMF 1 ST; para predicción iterada sin IMF 8 ST mejor y con IMF 8 ST. Si lo hiciéramos de esta forma, nos serviría para comparar ambas formas de predicción, es decir, si una es mejor que otra, y hasta cierto punto ese no es el objetivo principal del trabajo, el cual es poder determinar si es que IMF puede ayudar a una Red Neuronal a realizar una predicción mas

precisa. Alternamente se puede apreciar que para la tabla 7.3 y 7.4 la predicción iterada dio mejores resultados, ya que la mayoría de las mejores predicciones es con predicción iterada, por dicho motivo, se realizó la comparación de esta forma, es decir, globalmente se obtienen los mejores resultados con y sin IMF, sin fijarnos en los métodos de predicción. Esos métodos de predicción (iterada y directa) nos sirven para poder encontrar en algunos casos, mejores predicciones, que normalmente no se obtendría si únicamente ocupáramos un método.

La ST Seno se introdujo para probar si el algoritmo, era capaz de predecir una serie con una dinámica tan sencilla y por lo mismo no se aplicó IMF a dicha serie, ya que una IMF, es un modo oscilatorio que no tiene más ondas montadas y la Serie Seno es exactamente eso, aparte de que el Seno cumple con los dos requerimientos de una IMF, así el Seno, no está compuesto por más componentes, por esta razón no es lógico aplicar EMD al Seno.

También destaca el hecho de que el error obtenido para el Seno es mucho muy pequeño, aunque hay que tomar en cuenta que su rango original está de -1 a 1; la figura 7-4 nos muestra la gráfica del Seno con Error Absoluto Máximo y predicción iterada, graficando 70 puntos anteriores a la predicción, los siguientes 30 son las predicciones obtenidas por la Red en cuestión.

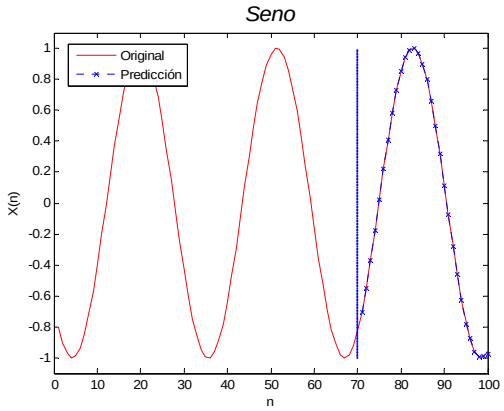


Figura 7-4 Serie Seno con EAM y predicción iterada.

En este trabajo se utilizará por convención tres tipos de líneas para mostrar las gráficas:

- Líneas punteadas para mostrar la predicción donde se distinga claramente la serie original y la predicción.
- Líneas punteadas con cruces, para mostrar predicciones más precisas, donde no se alcancen a distinguir entre la serie original y la predicción.
- Líneas continúas para los datos originales.

De la figura 7-4 podemos observar que el algoritmo encontró una Red capaz de predecir perfectamente a simple vista la Serie Seno, esto era algo que se esperaba, debido a la simplicidad de la misma. A continuación se mostrarán algunas gráficas representativas de los resultados, la primera parte será destinada para las ST que fueron mejores con IMF de las tablas 7.3 y 7.4, mientras que en la segunda parte se mostraran los casos contrarios.

7.3.1 Predicciones más precisas con IMF.

Podemos observar que la dinámica de las siguientes series mostradas, así como de la mayoría es no-lineal y no-estacionaria, lo que vuelve más complicado el poder encontrar una arquitectura de Red Neuronal adecuada, en comparación con la Serie Seno. Aquí se puede observar que si fue mejor utilizar las IMFs de la señal para ayudar a las Redes Neuronales tanto para encontrar la arquitectura como para predecir.

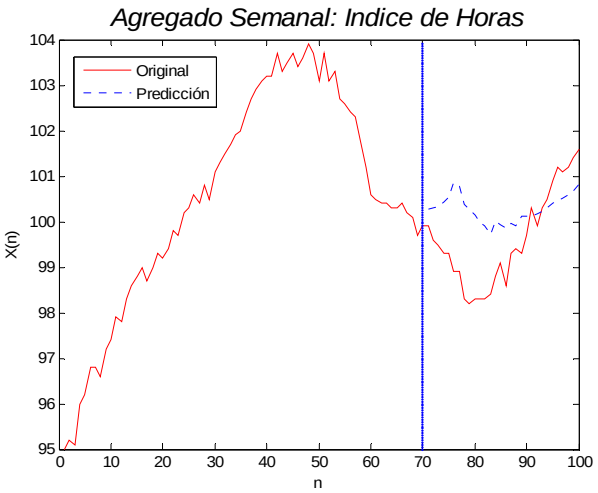


Figura 7-5. ST 1. Agregado semanal: Índice en horas con EAM = 2.0833, con IMF y predicción iterada.

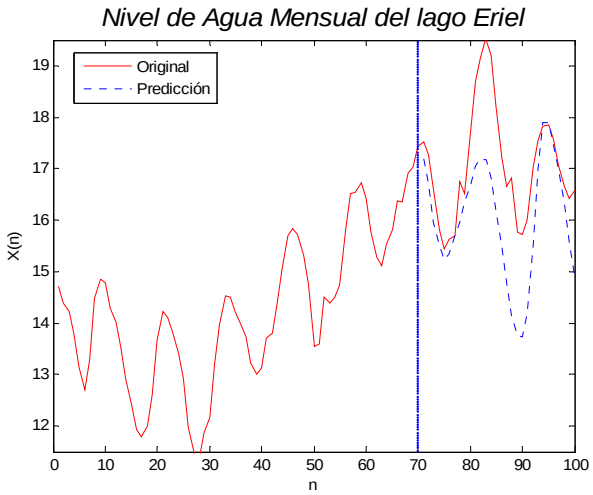


Figura 7-6. ST 8. Nivel de agua mensual del lago Eriel con EAM = 2.6646, con IMF y predicción iterada

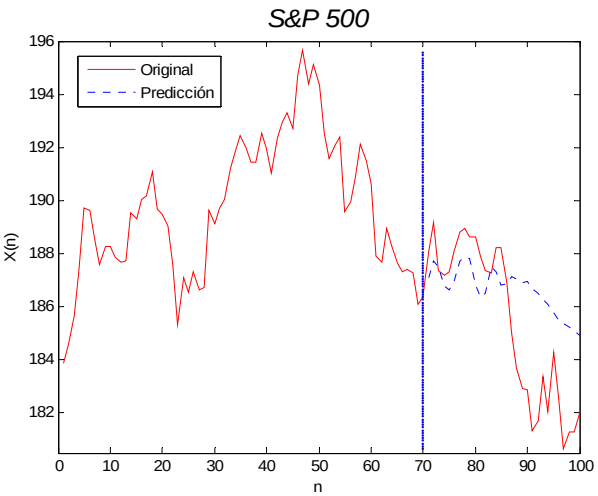


Figura 7-7. ST 16. SP500 con EAP = 2.1649, con IMF y predicción iterada

7.3.2 Predicciones más precisas sin IMF.

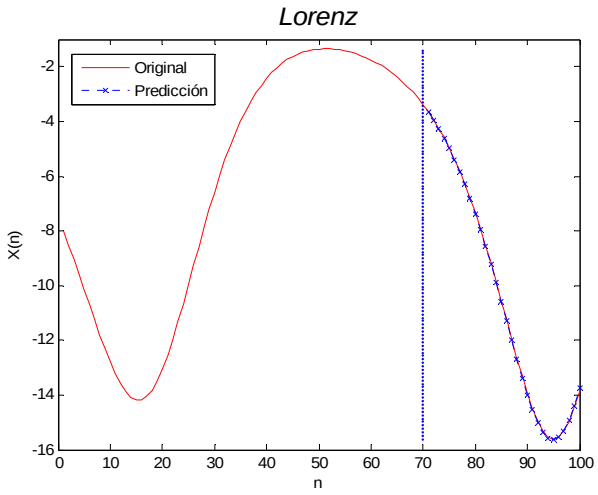


Figura 7-8. ST 9 con EAM = 0.06090 sin IMF y predicción iterada

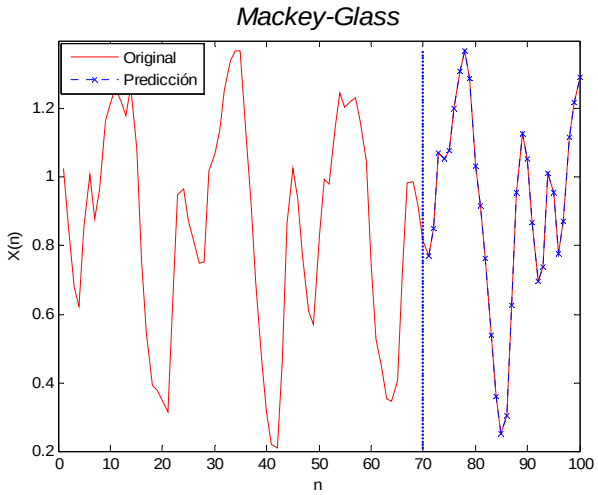
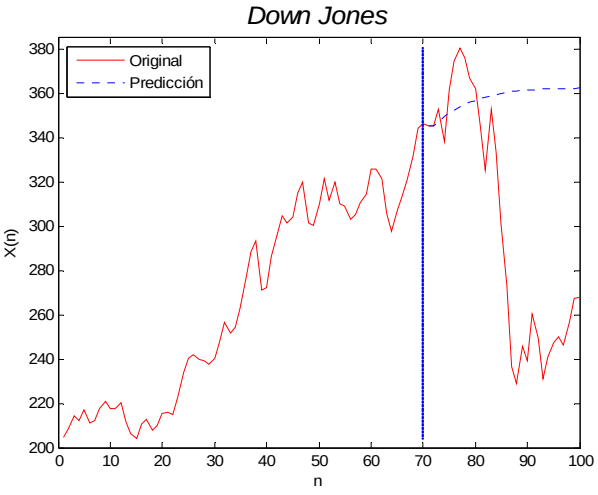
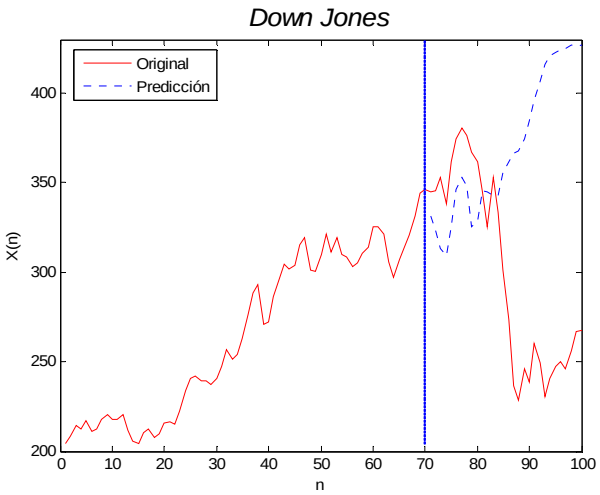


Figura 7-9. ST 10 con EAP = 0.00034 sin IMF y predicción iterada.



a) Sin IMF, EAM = 132.24



b) Con IMF, EAM = 185.12

Figura 7-10. ST 4 con EAM y predicción iterada.

Para la Serie Mackey (fig. 7-9), la predicción con IMF es muy parecida a la figura 7-9 (predicción sin IMF), por esta razón no es incluida en esta sección, y puede ser revisada en los anexos. A partir de la figura 7-9 empieza a suceder algo, un tanto diferente que la anterior (fig. 7-8), aquí (fig. 7-9), ambas predicciones con y sin IMF obtienen resultados muy precisos, observando el error se puede apreciar que no son iguales. Lo que está sucediendo es que con IMF también se puede obtener una predicción bastante aceptable en comparación con su contraparte.

En la figura 7-10 sucede algo totalmente diferente a las dos últimas figuras mostradas (figura 7-8 y 7-9), aquí nos dice que la predicción sin IMF tiene un error por debajo que con IMF, lo que implica que es mejor la predicción sin IMF, figura 7-10 inciso a); pero se puede apreciar claramente que es más preciso con IMF en los primeros 10 puntos aproximadamente, figura 7-10 inciso b). Incluso la predicción sin IMF no dio una solución adecuada como se puede ver, mientras que con IMF si pudo predecir mejor los primeros 10 puntos; esto sucede porque se está ocupando un EAM y al final de la figura 7-10 inciso b) el error es mas grande que en el inciso a) para la misma figura.

7.3.3 Análisis y discusión de la Segunda Etapa.

Pareciera que la mayoría de las predicciones sin IMF fueron mejores que con IMF, pero algunas predicciones sin IMF no pudieron encontrar una solución adecuada y sucede lo mismo que en la figura 7-10 inciso a), para lo cual hacemos el siguiente análisis:

Las soluciones que están catalogadas como mejores (más precisas, tomando en cuenta la comparación con el error obtenido) y que no dieron una predicción adecuada, es decir, el algortito no encontró una solución que nos dé una predicción confiable en términos prácticos para la Tabla 7.3 es:

- ST 4 y 12, es más precisa con predicción iterada e IMF en los primeros puntos.
- ST 6, 16 y 20, ninguna arrojo una solución aceptable.
- Las ST más precisas sin IMF son: ST 2, 9, 10, 13 y 17.
- Con IMF son: 1, 3, 5, 7, 8, 11, 15, 18 y 19.
- Por lo tanto, 5 ST fueron más precisas sin IMF, 9 con IMF, 2 más precisas con IMF en los primeros puntos y 3 ST que no se pudieron predecir.

Las soluciones que están catalogadas como mejores y que no dieron una predicción adecuada para la Tabla 7.4 son:

- ST 4, 18, 19 no dieron una solución aceptable.
- Las ST más precisas sin IMF son: 1, 3, 8, 9, 10, 11, 13 y 17.
- Con IMF son: 2, 5, 6, 7, 12, 15, 16 y 20.
- Esto nos da como resultado 3 ST que no se pudieron predecir, 8 con predicciones más precisas sin IMF y 8 con IMF.

Aunque la Red no de una solución aceptable, en algunos casos se aproxima mejor a los primeros puntos con IMF, lo que nos da una idea, de que sí funciona introducir más información con EMD; también se puede observar que algunas ST dieron diferentes resultados en las tablas 7.3 y 7.4.

Por ejemplo, para la ST 1 en la tabla 7.3 fue más precisa con IMF y en la tabla 7.4 fue más precisa sin IMF; esto se puede deber principalmente a dos factores: el tipo de error, que como se verá más adelante, si no se escoge el adecuado, se puede estar evaluando mal a las Redes, y posiblemente estar eliminando a candidatos potenciales de una solución adecuada; el segundo es el tamaño del espacio de soluciones, el cual es demasiado grande (menciono en la sección 6.3), por este motivo y por lo difícil que es realizar la predicción de Series de Tiempo, se puede dar el caso de estarnos moviendo (con el GANN) muy lejos del óptimo global, dando como resultado una predicción no tan precisa y una variación en los resultados, qué significa esto, que en algunos casos la predicción más precisa es con IMF y en su contraparte (la otra tabla con otro error) es más precisa sin IMF, como se mencionó anteriormente; esto también se puede repetir para los dos métodos de predicción. Aunque es un inconveniente del algoritmo, éste nos permite buscar automáticamente una arquitectura de Red Neuronal, y como se verá en la tercera etapa, se pueden encontrar arquitecturas que permiten realizar predicciones mucho más precisas, sin tomar en cuenta si es con o sin IMF, para este caso IMF viene a significar una ayuda extra en el diseño de ANNs.

Existe una desventaja al usar estos dos tipos de errores, para el Error Absoluto Promedio se puede sobreestimar o subestimar los datos predichos. Con el Error Absoluto Máximo puede suceder, que una gran parte de las predicciones tengan un error muy pequeño, pero con un error que sea demasiado grande es suficiente para catalogar a la Red como mala, aun cuando es muy precisa en la mayoría de los puntos; por lo que en la tercera aproximación del algoritmo usaremos otras medidas un poco más precisas tratando de evitar los problemas presentados.

En esta sección como en la anterior se usó a las primeras 5 IMFs para obtener las Redes y predecir. Si le dejáramos al Algoritmo Genético que optimizara también este valor (introducir las IMFs que ayuden más a la predicción), el espacio de soluciones se multiplicaría aún más; por lo que en los siguientes experimentos se trata de verificar de manera manual qué porcentaje de IMFs pueden ser útiles para realizar la predicción.

Existe un problema con esta versión y está relacionado con la construcción del Algoritmo Genético. Cuando se programa un GA, los miembros de la Elite pasan automáticamente a ser miembros de la siguiente generación, pero no se suele pasar su adaptabilidad, por lo que son evaluados de nuevo en la siguiente generación, esto mismo sucede también en la implementación del GA de Matlab; esto ocasiona un problema, ya que al construir la Red esta se tiene que entrenar, lo que significa que cuando pase a la siguiente generación se volverá a entrenar y desafortunadamente el entrenamiento con el algoritmo de Levenberg-Marquardt, es un problema de optimización y no siempre da exactamente los mismos resultados; por esto, los pesos de las Redes suelen variar de un entrenamiento a otro, lo que ocasiona que no tengan el mismo desempeño a la hora de predecir y en ese momento puede llegar un mejor individuo que lo desplace, siendo esta Red desplazada, un posible candidato a la solución.

A continuación se muestra en la Figura 7-11 un ejemplo de lo sucedido; siendo los puntos superiores, la media del error de la población y los puntos inferiores el mejor individuo por población.

En la figura 7-12 se muestra el comportamiento normal del GA para la optimización de una función matemática.

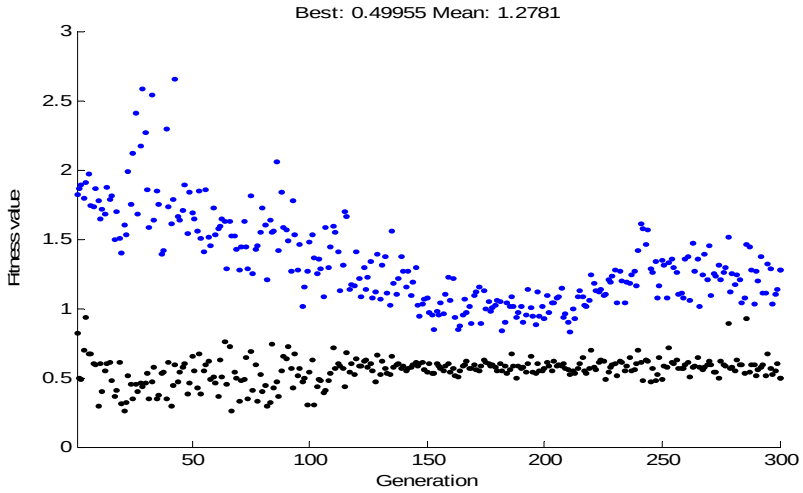


Figura 7-11. GANN mejor individuo por generación y la media del conjunto

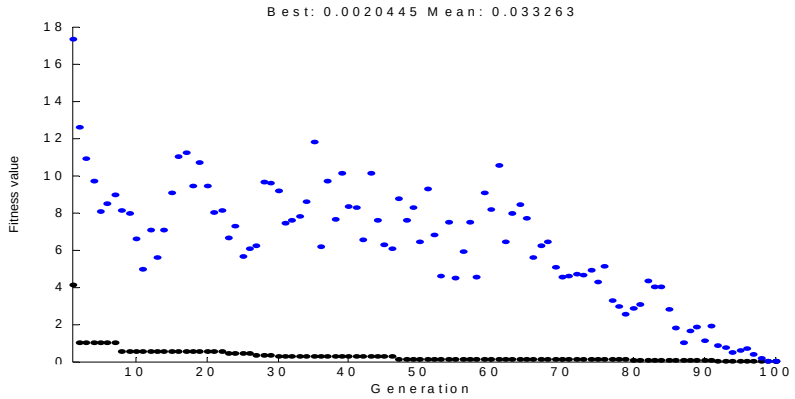


Figura 7-12 Función Rastrigin, mejor individuo por generación y la media del conjunto

La ecuación de la función Rastrigin es la siguiente:

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10\left(\cos 2\pi x_1 + \cos 2\pi x_2\right) \qquad \textbf{(7.1)}$$

Para la ecuación 7.1, no importa cuantas veces la evaluemos con un determinado valor de x_1 y x_2, siempre obtendremos el mismo valor, pero no sucede lo mismo para las Redes. En la figura 7-12 se observa como el mejor individuo pasa a la siguiente generación y así se mantiene hasta que llega otro mejor que lo desplaza (no se pierde al posible candidato); en cambio en la Figura 7-11 ya no se mantiene el mismo error para la siguiente generación y a veces baja o sube el error, esto debido a lo mencionado anteriormente, entonces perdemos a la Red que nos dio el error más bajo y nos quedamos simplemente con la que nos regresa al final el GA, por lo que en la siguiente sección se propone una forma para evitar dicho problema.

Otro problema encontrado, es con el parámetro "Paro por tiempo" del GA, en la mayoría de las predicciones sin IMF con predicción iterada y predicción directa (con y sin IMF) se alcanzó el total de generaciones del algoritmo, pero para las predicciones con IMF y predicción iterada no sucede lo mismo, esto por lo explicado al final del capítulo III, donde se menciona que es necesario volver a calcular las IMFs después de cada predicción; esto ocasiona, que no de tiempo al algoritmo de encontrar un mejor individuo en el tiempo estipulado y por consiguiente se para el GA en ese momento, quedándose con el mejor individuo que pudo encontrar.

Aunque esto se puede ver como un problema, dado el hecho de que no se cumplió con una amplia búsqueda (no se terminaron las generaciones), lo podemos ver a favor las IMFs; lo que significa que en muy pocas generaciones, las Redes encontradas y entrenadas con IMF, son capaces de superar a su contraparte (la mayoría con 300 generaciones). En la tabla 7.5 se muestran las generaciones alcanzadas para cada ST con EAP.

Cabe la posibilidad, de que el algoritmo no haya encontrado o no haya llegado un nuevo individuo que mejorará al anterior encontrado y por consiguiente, se ocasionaría la terminación del algoritmo en el tiempo estipulado, como es el caso para la Serie Mackey sin IMF la cual dio como resultado, predicciones con errores muy pequeños, que difícilmente podrían ser mejorados por otra Red. En el caso de la Serie Mackey con IMF y predicción iterada de la tabla 7.5 no sucede lo mismo, ya que únicamente se iteró una vez (1 generación), por lo que no pudo haberse dado el caso anterior.

Tabla 7.5 Épocas alcanzadas en el GA para EAP

#	Series de Tiempo	Predicción Directa		Predicción Iterada	
		Sin IMF	Con IMF	Sin IMF	Con IMF
1	Agregado semanal: Índice en Horas	300	300	300	6
2	Brownian Motion	300	300	300	1
3	Flujo diario del río Jokulsa	300	300	300	1
4	Down Jones	300	17	300	1
5	Precio diario del oro	300	300	300	1
6	HDNA	300	7	300	8
7	Precio de las acciones de IBM	129	3	105	13
8	Nivel de agua mensual del lago Eriel	300	19	300	14
9	Lorenz	300	300	206	1
10	Mackey-Glass	11	2	11	1
11	Nivel mínimo anual del río Nilo	300	6	300	1
12	Precipitación diaria Hveravellir	300	300	300	1
13	QP2	20	6	13	1
14	Seno	300	---	300	---
15	Southern Oscillation	300	300	300	1
16	SP500	300	85	300	1
17	No. Manchas Solares Mensuales	300	300	300	1
18	Temperatura max. Melbourne	300	300	300	1
19	Temperatura min. Melbourne	300	108	300	1
20	Salarios diarios de Inglaterra	300	300	300	11

Por desgracia no es posible poner muy grande el valor de Paro por tiempo o bien ponerlo a infinito como en la primera aproximación. Se probó el GA con las 300 generaciones y un Paro por tiempo = inf, dando como resultado, semanas para encontrar una solución para un sólo experimento con IMF y predicción iterada.

7.4 Tercera Aproximación.

Aquí vamos a emplear más ST y se incrementará el parámetro Paro por tiempo, para permitir que el algoritmo alcance más generaciones, para los experimentos con IMF y predicción iterada; también se utilizarán diferentes IMFs, tratando de identificar si es que existe un porcentaje de componentes (IMFs) que sea más útil para realizar la predicción.

Debido al problema que se puede presentar con los errores EAM y EAP, se emplean dos nuevos estimadores del error, tratando de evitar la sobreestimación y subestimación de los resultados; estos errores son: Raíz Cuadrada del Error Cuadrático Medio (RMSE) y Raíz Cuadrada del Error Cuadrático Normalizado (NRMS).

La Serie Seno ya no fue considerada en los experimentos debido a que se demostró que no había ninguna dificultad en realizar su predicción. Todo lo anterior nos da la siguiente configuración de parámetros para esta última etapa:

ANNs
- Sólo Redes feed-forward
- Las entradas únicamente van conectadas a la primera capa
- Número de entradas variables, hasta 7 entradas se permiten.
- Se permiten hasta 3 retardos en las entradas
- Se permite hasta 2 capas ocultas con un máximo de 15 nodos por capa
- Se utiliza predicción iterada y directa
- Se utiliza un máximo de 1500 datos de la ST, 80% para entrenar y 20% para probar

GA
- 300 Generaciones
- Paro por tiempo = 1000

EMD
- Se utiliza las primeras 5 IMFs
- Se usa el 70% de IMFs de menor amplitud total
- Se usa el 70% de IMFs de mayor amplitud total
- Se emplean todas las IMFs de la Serie

Evaluación
- RMSE
- NRMS

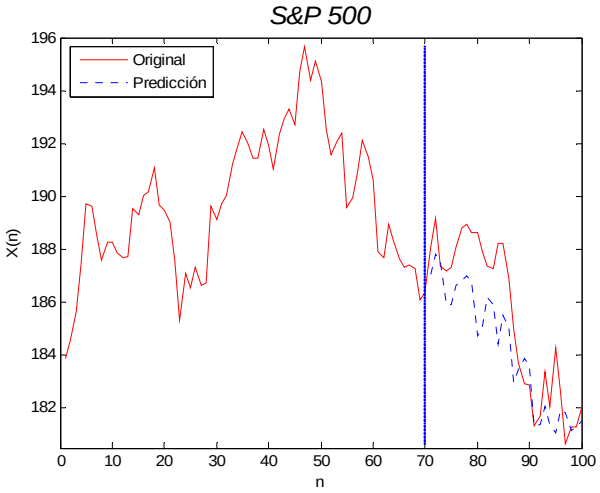
Se decidió utilizar de nuevo los experimentos con 5 IMFs, para tener un punto de apreciación de cuánto cambia la predicción con los nuevos errores.

Tabla 7.6 Series de Tiempo usadas con sus Rangos e IMFs

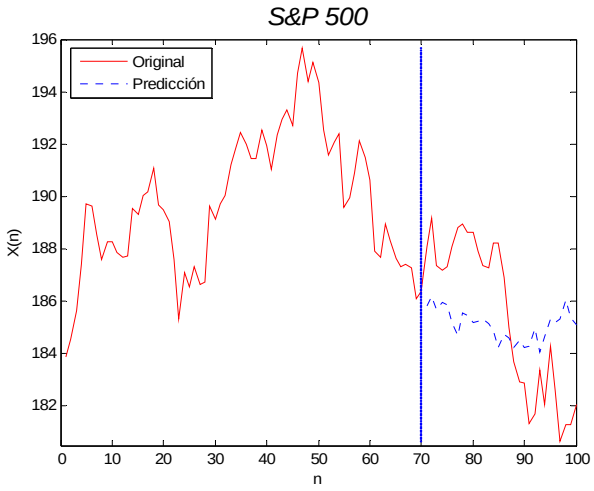
Grupos	#	Series de Tiempo	Rangos		No. IMF
			Min	Max	
1	1	Agregado semanal: Índice en horas	50.9	103.9	5
	2	Inversión total de todos los Bancos Comerciales	70.3	1980.1	5
	3	Precio diario del oro	285	593.7	7
	4	Precio de las acciones de IBM	49	175	8
	5	Salarios diarios de Inglaterra	2.15	49.99	7
	6	SP500	98.22	425.2	7
2	7	Flujo diario del río Jokulsa	22	143	9
	8	Flujo mensual del río Colorado	0.070	9.810	7
	9	Nivel de agua mensual del lago Eriel	10	20	6
	10	Nivel mínimo anual del río Nilo	9.01	15.3	8
	11	Precipitación diaria Hveravellir	0	79.3	9
3	12	Número de manchas solares mensuales	0	253.8	8
	13	Temperatura max. Melbourne	7	43	8
	14	Temperatura min. Melbourne	-0.8	26.3	8
	15	Southern Oscillation	-38.8	33.1	8
4	16	QP2	2.605	4.391	7
	17	QP3	2.8359	4.3592	7
5	18	Logistic	0.0951	0.9750	8
	19	Lorenz	-18.79	18	5
	20	Mackey-Glass	0.177	1.396	7
	21	Rossler	-9.1042	11.4287	5
	22	Ikeda	-0.6751	1.7135	9
	23	Henon	-1.2828	1.2730	10
6	24	D1	0.0420	1.171	8
	25	Laser	2	255	7
	26	Down Jones	31.8	59923	8
	27	Kobe	-34522	42428	9
	28	EEG	-0.7080	0.9082	8
	29	HDNA	1	4	8
	30	Lovaina	-0.6393	0.6066	6
	31	Primos	1	72	10
	32	Star	0	34	6
7	33	Brownian Motion	-3.923	6.216	7
	34	Ruido Blanco	0.001	0.999	9

Las Series de Tiempo usadas para este experimento son organizadas en grupos para enlistarlas. Para el primer grupo se toman en cuentas las series del tipo social/económicas (de Agregado semanal a SP500); el segundo está formado por series hidrológicas (de Flujo diario del río Jokulsa a Precipitación diaria en Hveravellir); el tercero consta de series físicas (de No. de manchas solares a Southern oscillation); el cuarto de series cuasiperiódicas (QP2 y QP3); el quinto con series catalogadas con una dinámica caótica (de Logistic a Henon); Sexto con series que tienen su dinámica compleja (de D1 a Star); el séptimo y ultimo está conformado por series que tienen en su dinámica un comportamiento estocástico (Brownian Motion y Ruido blanco).

En la tabla 7.6 se pueden apreciar las series utilizadas, del lado izquierdo se presenta el grupo al que pertenecen.



a) Predicción iterada con todas las IMFs, RMSE = 1.7547



b) Predicción directa sin IMFs, NRMS = 0.9645

Figura 7-13. ST 6, SP 500, las mejores predicciones con ambos tipos de error.

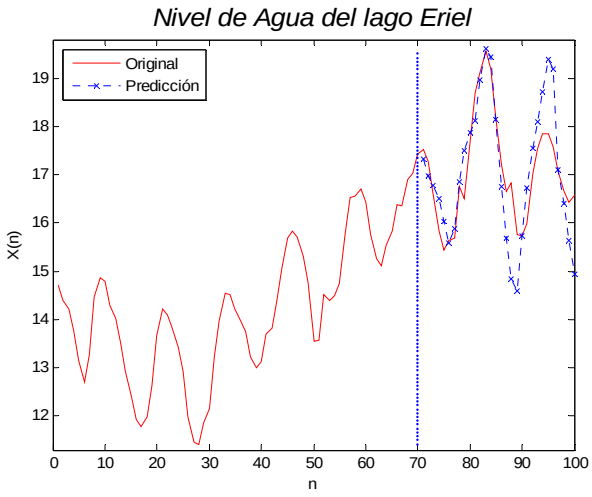
Debido al tamaño de las tablas de resultados para esta etapa (9.1 y 9.2), estas fueron introducidas en la sección de anexos. En las tablas 9.1 y 9.2 se presentan las predicciones con RMSE y NRMS respectivamente.

Resumiendo, de la tabla 9.1 tenemos 17 predicciones más precisas sin usar IMF entre ambas formas de predecir y 17 predicciones con IMF. Para la tabla 9.2 se tienen 12 ST con predicción más precisa sin IMF y 22 predicciones con IMF.

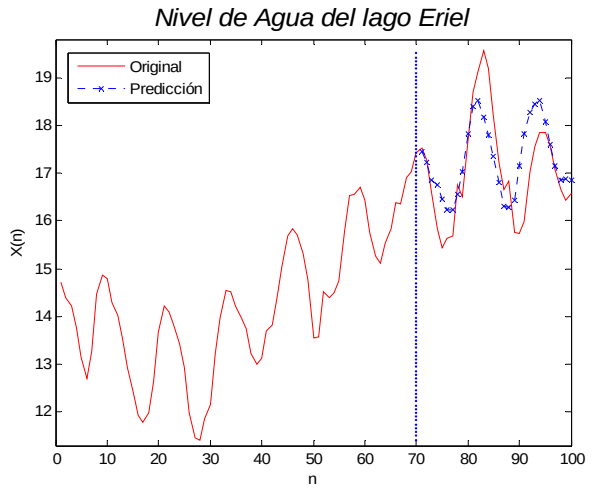
Estos resultados no ayudan a poder determinar, si es que hay algún porcentaje de IMFs que pueda servir más que otro para realizar la predicción. La ventaja obtenida aquí, es que se pudieron predecir mejor dichas series al probar con diferentes componentes (IMFs), lo que significa que no para todas las series les es funcional el mismo porcentaje de IMFs.

Cabe señalar que aunque el Paro por tiempo se estableció a 1000, 400 más que en la segunda aproximación, tampoco dio tiempo a que todas las generaciones transcurrieran para las predicciones con IMF y predicción iterada; al utilizar estos nuevos errores RMSE y NRMS, se pudo evitar las sobreestimaciones y subestimaciones que se tenían anteriormente. En las gráficas mostradas a continuación, se observa las mejores predicciones de algunas series.

Para la Series SP 500 figura 7-13, se puede apreciar una cierta mejora en la tendencia general con la figura 7-13 a), pero la figura 7-7 sigue siendo más precisa en los primeros puntos. También es de observar que aunque el error más pequeño para el NRMS fue con una predicción sin IMF en la figura 7-13 b), esta no puede mejorar a las dos mencionadas anteriormente. Cuando nos referimos a una predicción sin IMF, nos estamos refiriendo a que es una predicción en la que se ocuparon los datos originales únicamente, para encontrar una Red con el GANN y para predecir.



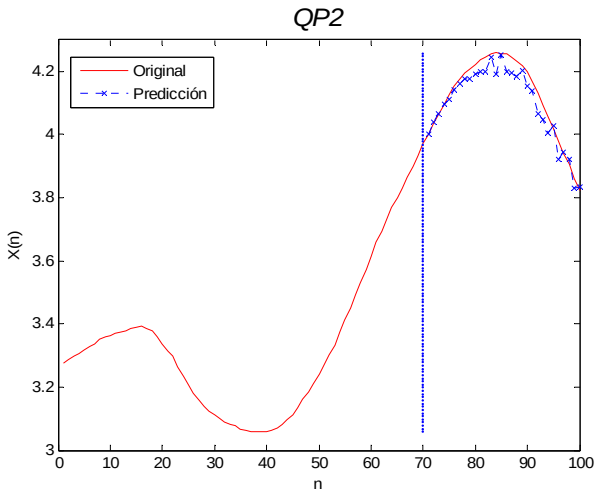
a) Predicción iterada con todas las IMFs, RMSE = 0.7995



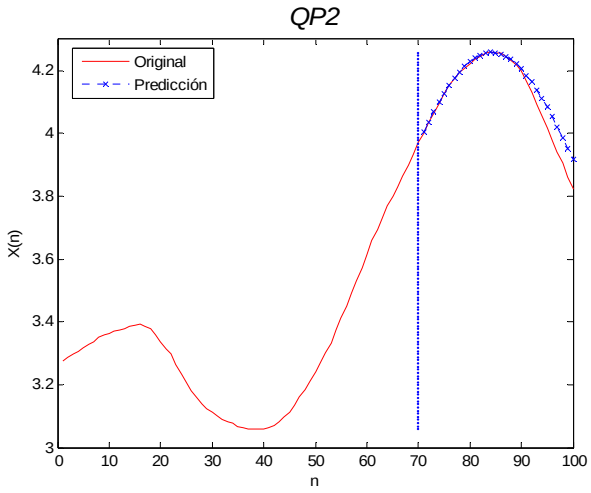
b) Predicción iterada con el 70 % IMFs de menor amplitud total, NRMS = 0.7015

Figura 7-14. ST 9, Nivel de agua mensual del lago Eriel, las mejores predicciones con ambos tipos de error.

Para la Serie 9 figura 7-14, sí se puede mejorar la predicción, de los resultados mostrados en la segunda aproximación, aunque la afirmación realizada aquí depende únicamente de la apreciación a simple vista de las gráficas, que si se observan, sí se nota una mejoría para estos experimentos.



a) Predicción directa sin IMF, RMSE = 0.0340



b) Predicción iterada sin IMF, NRMS = 0.3182

Figura 7-15. ST 16 QP2, las mejores predicciones con ambos tipos de error.

La Serie QP2 mantuvo una consistencia en las predicciones, éstas fueron sin IMF en la segunda y tercera etapa, lo que puede implicar que no es necesario utilizar IMF para esta señal, utilizando el algoritmo GANN. En la Serie QP3 sucede algo similar como en la anterior, la gráfica de la misma se puede revisar en los anexos.

La Serie Logistic (figura 7-16) presenta predicciones muy precisas en los primeros puntos, aunque pareciera que la dinámica es muy compleja, el algoritmo es capaz de encontrar Redes que permiten realizar dicha predicción, aunque no fueron con el mismo número de IMFs, para ambos tipos de error, la gráfica para la mejor predicción con NRMS es muy similar a la figura 7-15, prediciendo muy bien los primeros 10 puntos, por lo que su figura no se agrega en esta sección. La gráfica para la Serie Lorenz con ambos tipos de error puede ser observada en los anexos.

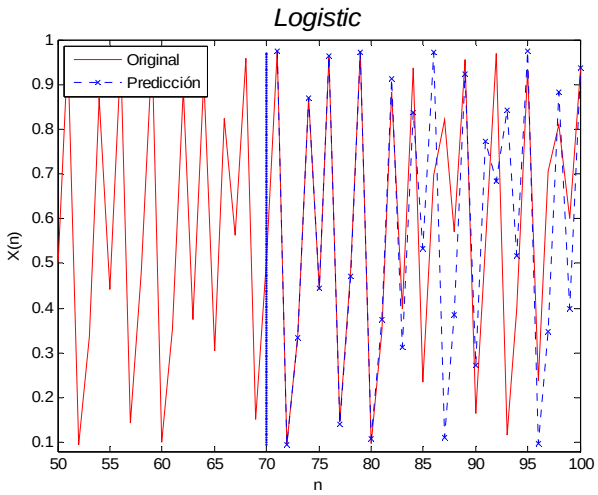


Figura 7-16 ST 18, Logistic, Predicción iterada con el 70% de IMFs de mayor amplitud total, RMSE = 0.2316.

La ST Mackey, tuvo una predicción muy similar a la presentada en la segunda aproximación por lo que no se expone dentro de estas gráficas. En las Series Logistic y Henon presentadas aquí, únicamente se graficaron 20 puntos anteriores a la predicción, debido a los grandes cambios que presenta la series en el transcurso del tiempo; esto es para ampliar más la figura y tener una mejor apreciación.

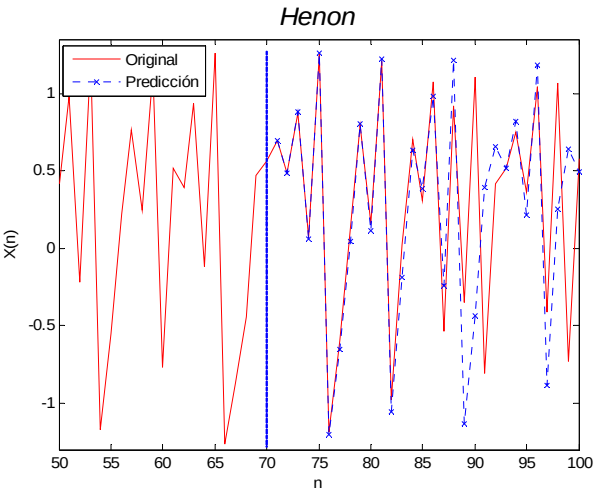


Figura 7-17. ST 23, Henon, predicción iterada con el 70% de IMFs de menor amplitud total y NRMS = 0.7205.

Para la Serie Henon, la predicción con RMSE es muy similar a la de NRMS, prediciendo muy bien los primeros puntos, por ese motivo no se presenta la gráfica para dicho error. Las gráficas de las Series Rossler, Laser, Lovaina y Star se pueden observar en los anexos; en las mismas se aprecia que si la serie presenta grandes amplitudes, y bajas frecuencias, en algunos casos el introducir IMF no ayuda, pero en otros el introducirlas casi iguala a los resultados sin IMF como es el caso de la Serie Rossler, Mackey y Lorenz, donde si se logran predicciones muy precisas con y sin IMF, pero no da una ventaja sobresaliente el usar las IMFs.

Como ya tenemos las mejores predicciones que podemos obtener, vamos a utilizar la Redes encontradas para predecir a más de 30 puntos, para tratar de determinar el alcance del algoritmo. Sólo podemos realizar esto para las series que son hasta cierto punto sencillas de predecir, porque como se pudo observar

anteriormente, para las que son más complicadas, apenas se pueden predecir los primeros puntos, como podría ser el caso de la figura 7-16 Logistic o 7-17 Henon.

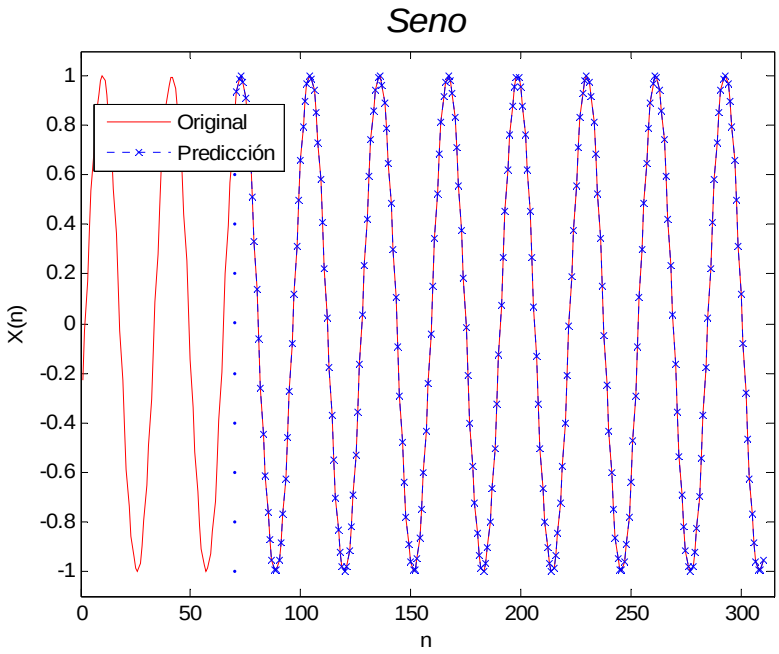


Figura 7-18. Serie Seno con predicción iterada, EAM = 0.0024 y sin IMF. Predicción a 240 puntos adelante.

Vamos a empezar con la serie más sencilla que se tenía y para eso ocuparemos a la Red obtenida de la segunda aproximación de la Serie Seno. Se realizaron pruebas para predecir cada vez el doble de puntos, esto es 60, 120, 240 puntos y así sucesivamente, hasta que los datos disponibles nos lo permitieran o bien, hasta que la predicción ya no fuera precisa. La figura 7-18, muestra la predicción para la Serie Seno.

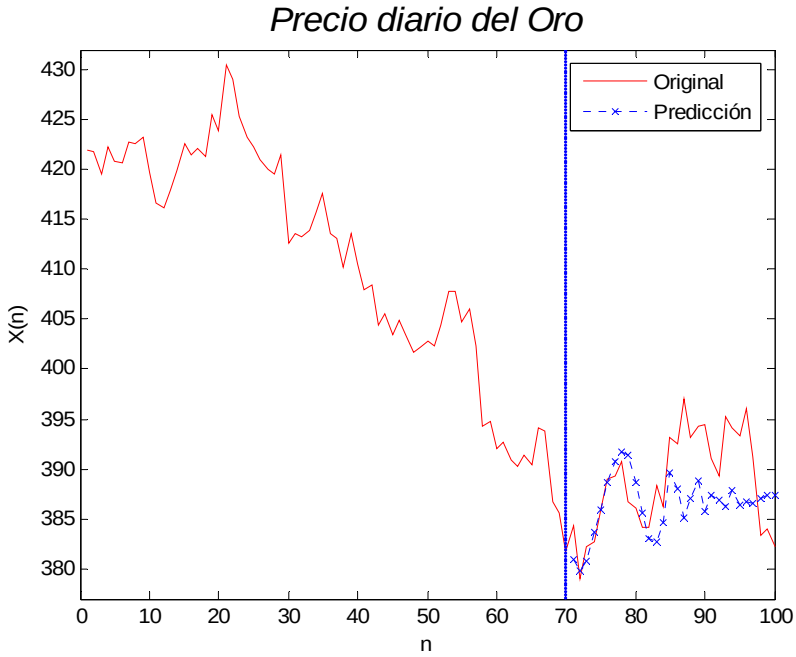


Figura 7-19. Serie Precio diario del oro con NRMS = 1.0299, predicción iterada y 5 IMFs.
Predicción a 30 puntos adelante

Desafortunadamente no se tienen datos suficientes para varias Series de Tiempo, lo que ocasiona que no podamos extender la predicción, como es el caso de la Series: Agregado semanal, Inversión total, Salarios diarios, Flujo del río Colorado, Nivel de agua mensual del lago Erie y Star.

Para la ST 4, 6, 7, 11, 13, 14, 15, 16, 18, 19, 22, 23, 24, 25, 26, 27, 28, 29, 31, 33, 34 no se puede extender el periodo de predicción a más de 30 puntos adelante, debido a la complejidad que presentan las series.

Para la figura 7-19, no se puede extender la predicción a 60 puntos, porque esta es una de las series más difíciles de predecir.

La ST 10 Nivel de Agua del río Nilo, no tuvo una predicción tan precisa, para predicciones mayores a 30 puntos, pero sí da bastante bien la tendencia de la serie. Para la Serie 12, Número de manchas solares con NRMS también se obtuvo una Red que no es tan precisa en los detalles, pero sí es capaz de seguir la tendencia de la serie. Para la Serie 17 QP3 con NRMS, si obtuvo una predicción bastante precisa a 240 puntos adelante como se muestra en la figura 7-20, esta figura se presenta un poco más grande que las demás, para alcanzar a ver los detalles, que de otra forma se perderían. La mejor Red encontrada con RMSE, no pudo obtener una predicción igual de precisa que con NRMS, a la misma cantidad de puntos.

La serie 19 Lorenz, ya no tiene una predicción tan precisa a 60 puntos para el NRMS, sin embargo, si se obtiene una predicción precisa para RMSE, como se muestra en la figura 7-21.

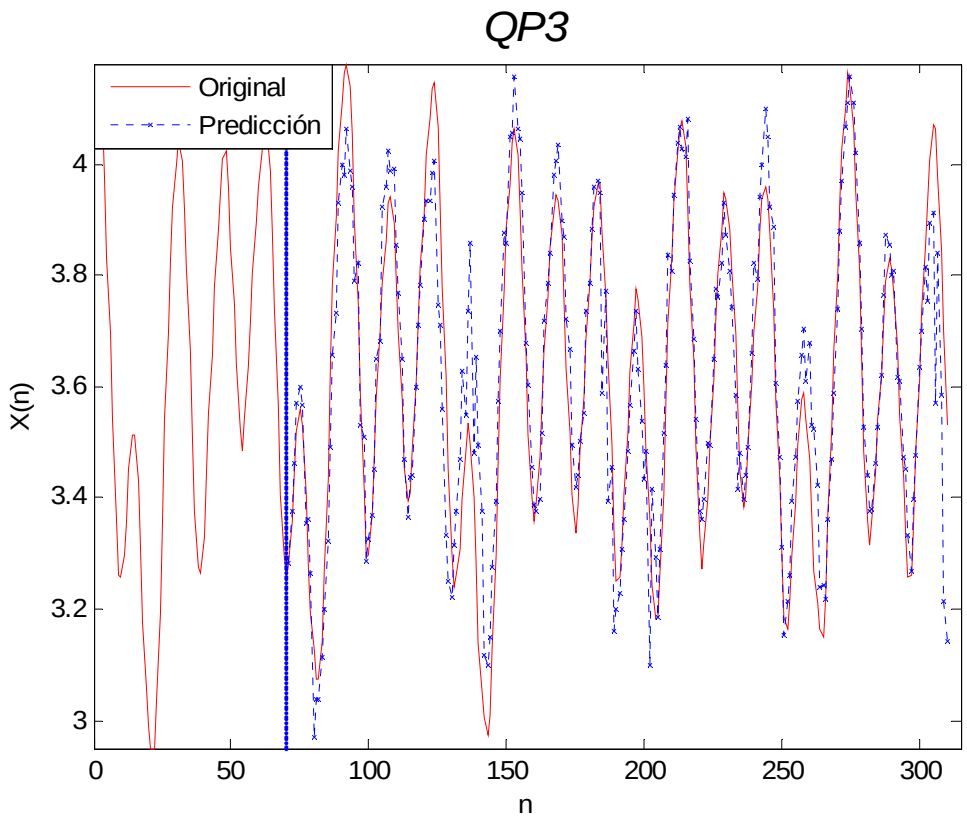


Figura 7-20. Serie QP3 con NRMS = 0.4142, predicción directa sin IMF a 240 puntos adelante

La Serie 20 Mackey, figura 7-22, sí es bastante precisa a 240 puntos adelante con ambas Redes encontradas para los diferentes errores, en la figura 7-22 se muestra la predicción con NRMS.

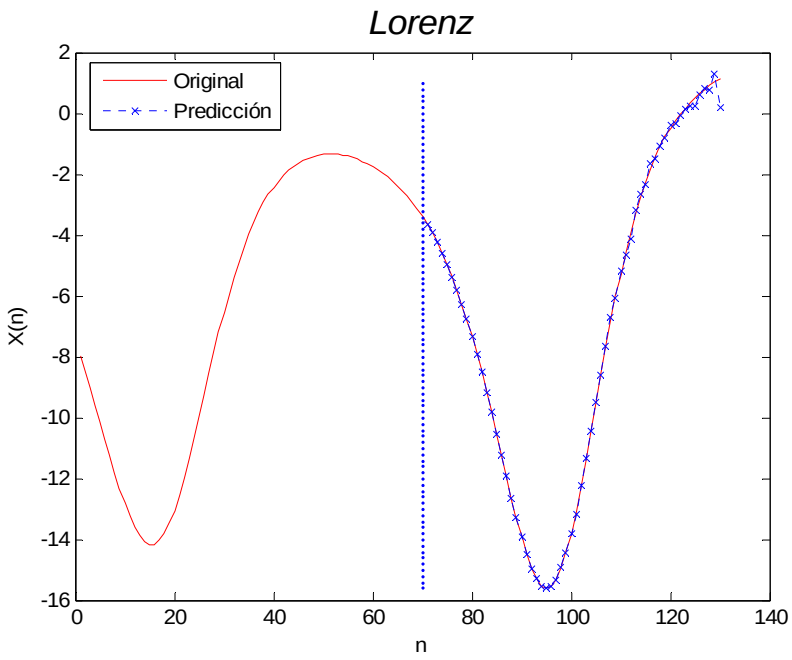


Figura 7-21. Serie Lorenz con RMSE = 0.1421, predicción directa sin IMF a 60 puntos adelante

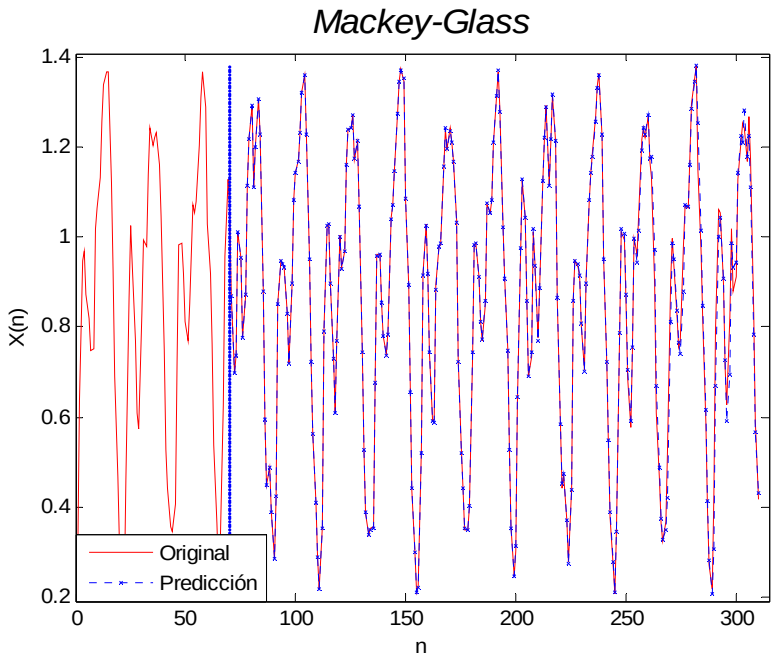


Figura 7-22. Serie Mackey-Glass con NRMS = 0.0760, predicción iterada sin IMF a 240 puntos adelante.

Para la Serie 21 Rossler, figura 7-23, sucede algo similar a la Serie 19, con RMSE se puede predecir bastante bien a 240, pero la red encontrada con NRMS no puede ni predecir 60 puntos adelante.

La Serie 30 Lovaina, figura 7-24, no se puede obtener una predicción aceptable a 60 puntos con RMSE, pero con NRMS si es un poco más precisa, como se muestra en la figura 7-24.

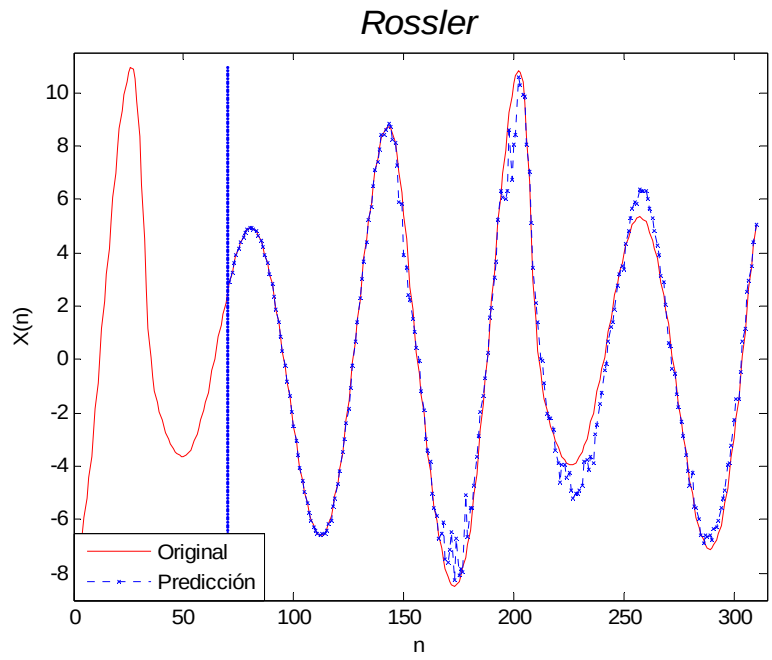


Figura 7-23. Serie Rossler con RMSE = 0.6580, predicción directa sin IMF a 240 puntos adelante.

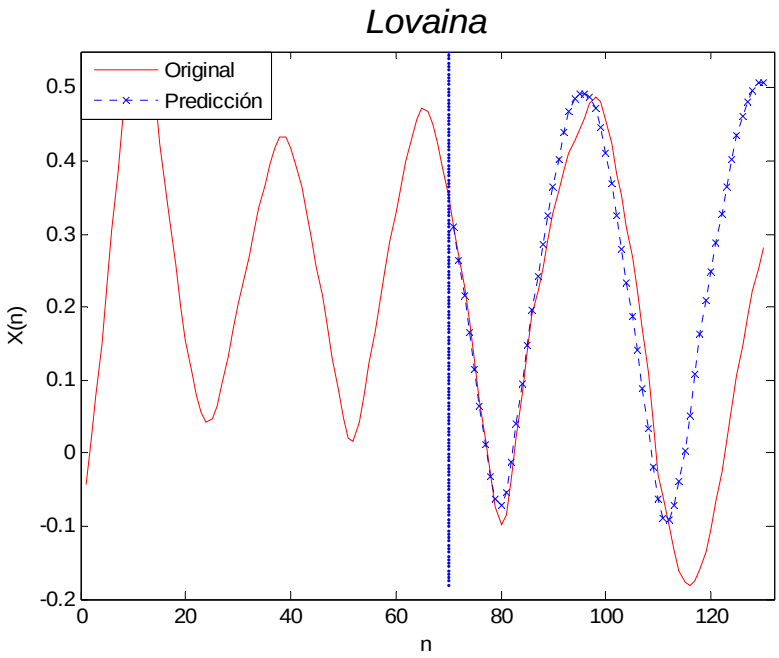


Figura 7-24. Serie Lovaina con NRMS = 0.7952, predicción iterada sin IMF a 60 puntos adelante.

En la sección de anexos, se pueden encontrar todas las variables que se utilizaron en la evolución del GANN, ahí se presentan los datos para las mejores predicciones únicamente, así con dichos datos se podrán reproducir los resultados entes mencionados. Estos parámetros corresponden a los resultados de la última etapa.

7.5 Discusión de resultados.

En este capítulo observamos la evolución del algoritmo desde su primera etapa, hasta la tercera, donde se le realizan cambios y mejoras, que como se vio, sí obtuvo al final predicciones muy precisas en diferentes series de Tiempo. Para Series Caóticas o Complejas se pueden predecir muy bien los primeros puntos, para otras fue mejor no utilizar IMF y nos pudimos alargar en la predicción hasta 240 puntos adelante, como se acaba de ver arriba, esto implica principalmente que sí funciona el introducir las IMFs de la serie para determinar una arquitectura de ANN y para predecir, ya que de no haber sido así, todas las predicciones hubieran sido más precisas sin IMF. Estos resultados también nos dan una indicación, de que el Algoritmo Genético es capaz de encontrar Redes Neuronales adecuadas para realizar la predicción con o sin IMF; hay que recordar que las arquitecturas se determinan por los datos de entrada y por la evolución del genoma en el GA. Si no funcionara el GANN como se muestra en los resultados, no tendríamos predicciones útiles o válidas para utilizarlas como tal, sin embargo, al ver los resultados de esta última etapa, podemos darnos cuenta de que a pesar, de que se determina automáticamente las estructuras de las Redes sin la intervención de un diseñador especializado, éstas pueden proporcionarnos una predicción muy precisa.

Por desgracia no podemos establecer alguna regla de los últimos experimentos, en cuanto al número de IMFs requeridas para predecir, ya que los resultados son muy dispersos, pero como se puede observar, el ocupar diferentes IMFs ayuda diversas ST a obtener una predicción más precisa, que de otro modo no se hubiera logrado, por lo que se puede decir que no a todas las series, les funciona el mismo porcentaje de IMFs para predecir, pero sí resulta útil el variarlas, para tratar de obtener mejores resultados.

Otro punto a favor del algoritmo en su totalidad, es que aunque el espacio de soluciones es extremadamente grande, se pueden encontrar arquitecturas adecuadas para realizar predicciones lo suficientemente aceptables; otras presentan predicciones tan precisas, que es difícil determinar la original de la predicción, si no se tuvieran marcas en los puntos predichos, como es el caso de las Series Mackey-Glass, Lorenz, Star, Laser, Rossler y QP2, para predecir 30 puntos adelante sin IMF; Henon y Logistic en los primeros puntos con IMF.

Pero como es de esperarse no para todas las series se pudo realizar una predicción tan precisa, esto debido a su comportamiento, como en el caso de las Series 13 y 14 con ambos tipos de error, los cuales no tuvieron una predicción útil, hablando es sentidos prácticos. Lo mismo sucede para la Serie de Ruido blanco, ya que no convergió a una solución aceptable, pero también hay que aclarar que son series complicadas de predecir, incluso no existe un predictor para el Ruido Blanco, debido a su naturaleza.

Por desgracia, muchos de los artículos que tratan la predicción de Series de Tiempo y que usan alguna serie de las usadas aquí, no tienen los mismos parámetros de evaluación para hacer una comparación [2, 3, 4, 5, 6, 9, 11]. Por ejemplo en [10] realizan la predicción, aplicando ondeletas y Redes Neuronales para la Serie de manchas solares, pero la diferencia, es que hacen el promedio de manchas solares anuales, y las predicciones mostradas en este trabajo de tesis, es para manchas solares promediadas por mes, lo cual hace que sean diferentes las series y por lo tanto no comparables los resultados. Lo mismo sucede en [7], donde se realiza la predicción de dicha serie a 15 puntos adelante, sin embargo, cada punto corresponde a un año, así realizan la predicción para los siguientes 15 años, en nuestro caso, realizamos la predicción a 30 puntos adelante, lo que corresponden a 30 meses, por lo tanto no es posible hacer comparaciones.

Pero sí hay algunos con los que se puede comparar los resultados de la tesis como con [1], donde realizan la predicción de la Serie Mackey-Glass utilizando un algoritmo hibrido entre GA y ANNs, evolucionando la arquitectura de estas últimas. Utilizan una codificación indirecta en el GA y utilizan el NRMS, lo cual nos permite hasta cierto punto comparar directamente nuestros resultados; esto es debido, a que en 100 puntos de su serie, ésta presenta 2 periodos; en 100 puntos de nuestra serie se pueden apreciar entre 4 y 5 periodos (más del doble), lo que significa que las series están muestreadas de diferente forma. En sus experimentos el error más pequeño reportado con NRMS es de 0.1868 para predecir 500 puntos adelante, de entre 5 experimentos diferentes para la misma serie, mientras que en nuestros resultados, para la Figura 7-22, para predecir 240 puntos adelante, tiene el valor de NRMS = 0.0760, menos de la mitad de lo que ellos obtienen y nuestros 240 puntos, equivalen a más de sus 500 puntos por el tipo de muestreo. Este error es significativamente más pequeño, por lo que nuestro algoritmo resulta ser más preciso y competitivo, al menos para la Serie Mackey,

porque en [1] todo el esfuerzo se pone para predecir una ST y este trabajo está dedicado 34 ST, por lo que es una ventaja obtener una predicción bastante precisa, tomando en cuenta que no nos enfocamos en una ST únicamente.

En [8] sucede algo similar a lo anteriormente mostrado, nada más que ahí, en 100 puntos de su serie se pueden apreciar entre 9 y 10 periodos, prácticamente el doble que nuestra serie. Ellos realizan la predicción a 100 puntos adelante, que sería lo equivalente para nosotros, a 200 o 240 puntos que predecimos en la figura 7-22 la predicción es para la misma serie con el mismo error, lo único que cambia es que aquí se realiza la optimización de los pesos con el GA, su error más pequeño reportado es NRMS = 0.1049, el cual no supera tampoco a nuestro error de NRMS = 0.0760.

Como podemos observar, no sólo se lograron predecir muy bien diferentes fenómenos, sino también se pudieron mejorar algunos resultados reportados en la literatura, como es el caso de la Serie Mackey.

También se observó la evolución del algoritmo, con lo que se espera, que el lector tenga un mejor criterio de lo que funciona o no funciona en una configuración de éste tipo, además de que se realiza un análisis, de lo que no funciona, comentando el por qué no funciona o dando las posibles causas que lo ocasionan, además de que se sugiere una solución en cada etapa. En el próximo capítulo se verán las conclusiones a las que se llegaron, junto con las posibles mejoras que podría tener el algoritmo, esto es, trabajos futuros.

Posteriormente en el capítulo IX, se presentan los anexos, donde se pueden consultar algunas figuras y tablas del capítulo VII, también se pueden apreciar todas las series empleadas, con parámetros propios de ellas, junto con un glosario de términos y trabajos derivados de la tesis.

Capítulo VIII Conclusiones.

8.1 Conclusiones.

Después de los resultados obtenidos podemos elaborar las siguientes conclusiones:

- **Bajo condiciones controladas, los resultados muestran la contribución que proporcionan las IMFs en la Predicción de ST con Redes Neuronales y Algoritmo Genético.**

- **La evolución de arquitecturas de Redes Neuronales con Algoritmo Genético, permite encontrar una Red adecuada para cada problema, obteniendo predicciones muy precisas para diversas Series de Tiempo.**

- **Es posible obtener predicciones más precisas, al ajustar el número de IMFs introducidas a la Red Neuronal, en cada caso particular.**

- **Los resultados obtenidos, nos indican que al realizar la predicción de una Serie de Tiempo, es conveniente probar con predicción Iterada y Directa, escogiendo el método que nos proporcione mejores resultados.**

- **En el presente trabajo se muestran resultados, en donde se lograron predicciones muy precisas a corto plazo, usando EMD en Series de Tiempo Caóticas o Complejas. Así mismo, se obtuvieron predicciones muy precisas a largo plazo, en donde no fue necesaria la introducción de las IMFs.**

- **Un análisis de la predicción usando técnicas clásicas de la matemática no seria útil ni posible, debido a que no conocemos la expresión que determina su comportamiento. De esta forma, no podemos asegurar que se siga cumpliendo dicho análisis, cuando se presente un nuevo fenómeno, con otra dinámica diferente, es decir, con una función nueva desconocida.**

8.2 Trabajos futuros.

A partir del presente trabajo, se pueden seguir varias líneas de investigación, como son:

- Continuar y ampliar el estudio de las Funciones de Modo Intrínseco en la predicción de Series de Tiempo, para la obtención de predicciones más precisas.

- Estudiar bajo que condiciones un método de predicción puede proporcionar mejores resultados que otro.

- Implementar algún otro tipo de codificación en el GANN que mejore a los resultados aquí presentados.

- Utilizar otros tipos de Redes, que utilicen métodos de aprendizaje diferentes.

IX Anexos

9.1 Trabajos derivados de la tesis.

Durante la elaboración de la tesis, se derivaron tres trabajos presentados en congreso [56, 57, 58]

1. El título del primer artículo es: *Genetic Algorithm to Design Artificial Neural Networks in Time Series Forecasting with Intrinsic Mode Functions*. Publicado en: Research on Computing Science. Advances in Computer Sciences in Mexico. Vol. 13, pp. 101-110. ISSN: 1665-9899. en este trabajo se presenta la predicción de Series de Tiempo a un paso adelante.

2. El segundo artículo esta titulado: *GAAN en la predicción de Series de Tiempo con funciones de Modo Intrínseco*. Publicado en las Memorias del XLVIII Congreso Nacional de Fisica, 3MG3, pág. 110. Para este trabajo se aumentan las ST empleadas y se introducen las variables de entradas y retardos, usando EAP y EAM como se muestra en la segunda aproximación.

3. Título: *Algoritmo Híbrido GANN en la Predicción de Series de Tiempo con Descomposición Empírica en Modos*. Publicado en el ROC&C'2005 -- CP-41. Aquí se introducen nuevamente más ST, utilizando las medidas de evaluación: RMSE y NRMS y utilizando al mejor individuo encontrado y no el último que nos regresaba el GA.

9.2 Tablas en Extenso.

A continuación se presentan las tablas de la tercera etapa del algoritmo, presentado en el capítulo VII Resultados, por renglón se puede encontrar a la predicción más precisa para cada Serie de Tiempo, el valor está remarcado para una mejor apreciación.

Tablas 9.1 Predicciones con RMSE

#	**Predicción Directa**					**Predicción Iterada**				
	Sin IMF	Con IMF 5	Con IMF 70 >	Con IMF 70<	Con MF Todas	Sin IMF	Con IMF 5	Con IMF 70>	Con IMF 70<	Con IMF Todas
1	2.7138	3.6077	3.9083	2.5597	3.5224	1.2015	1.8357	3.486	1.314	**1.1439**
2	234.82	249.38	256.34	191.32	231.83	**138.16**	582.76	650.08	188.04	905.79
3	17.606	**5.8616**	35.893	7.7092	12.185	16.06	9.216	24.5	6.9932	44.292
4	2.1513	1.8456	5.9264	2.6168	8.7671	1.9391	1.6366	2.9788	**1.5091**	1.7815
5	11.273	14.701	17.875	14.187	13.501	**11.235**	11.435	11.925	14.954	11.568
6	4.9592	5.2294	7.0937	5.2742	5.3492	2.7915	4.7428	2.5153	2.9354	**1.7547**
7	5.3404	**3.3844**	10.322	7.8617	9.146	5.7857	6.8153	497.97	7.9065	5.7353
8	0.3107	**0.2734**	0.3100	0.3501	0.3536	0.7103	0.3024	0.3403	0.6231	0.8382
9	1.3926	2.1209	1.6532	2.3793	1.9281	0.8043	1.784	1.8443	2.8142	**0.7995**
10	2.2649	1.1187	1.4818	1.537	**1.0188**	1.591	1.1589	1.4871	1.6586	1.348
11	2.613	2.3254	5.8423	4.4901	12.665	2.4386	1.9704	**1.8834**	2.0868	2.6873
12	**21.755**	35.971	77.946	43.521	39.955	33.151	28.287	63.237	36.018	25.982
13	7.2514	7.172	7.041	6.9457	6.8914	6.5856	8.9047	6.9451	7.5965	**6.3652**
14	2.5331	3.4525	2.9486	3.1773	3.1191	**2.3025**	2.3205	3.0842	3.1581	3.0969
15	11.957	17.58	26.79	14.692	13.546	12.316	**8.068**	11.184	13.109	20.848
16	**0.0340**	0.1208	0.1464	0.1502	0.1867	0.1048	0.3432	0.1106	0.5593	0.7382
17	**0.1251**	0.2196	0.1992	0.2146	0.2164	0.2849	0.2470	0.2546	1.3254	0.2643
18	0.2683	0.2812	0.2834	0.2797	0.2752	0.2445	0.3147	**0.2316**	0.3183	0.3543
19	**0.2075**	3.0856	1.0049	0.7325	1.9038	0.6229	0.4961	0.2571	1.4387	0.9578
20	0.0512	0.2615	0.1987	0.1717	0.2522	**0.0017**	0.0241	0.0133	0.0864	0.0038
21	**0.0043**	1.6854	0.3302	0.3635	0.8316	0.0338	2.7	0.9223	11.946	1.2227
22	0.4911	0.6702	0.6006	0.5990	0.5175	**0.4630**	0.6234	0.6497	0.6006	0.5448
23	0.6579	0.9871	0.6608	1.3068	1.1716	**0.6229**	0.8900	0.8011	1.0002	0.7904
24	**0.2370**	0.3126	0.2959	0.3155	0.2995	0.2793	0.2837	0.3224	0.2835	0.2813
25	**10.664**	43.376	38.489	48.397	39.091	25.319	22.407	47.532	45.434	40.824
26	122.15	75.304	162.63	116.94	225.36	**65.263**	76.921	72.219	89.762	87.498
27	5845.5	6300.5	5023	6801.8	4833.7	4666.4	**4482.6**	6648.1	8548.2	6084.5
28	0.2287	0.2758	0.2711	0.2999	0.2913	0.2186	0.2231	0.3299	**0.2149**	0.3182
29	**0.9966**	1.1826	1.1618	1.027	1.0722	1.1063	1.5584	1.3928	1.2461	1.19
30	**0.0973**	0.2841	0.2692	0.3112	0.1617	0.1186	0.1977	0.1180	0.5209	0.1493
31	5.8039	5.6017	5.6286	5.7801	6.3397	5.671	5.5513	6.3993	9.5625	**5.4359**
32	0.3916	7.1108	0.6198	10.651	1.2119	**0.3577**	1.1184	2.0528	0.8984	0.5443
33	0.9912	3.155	3.1498	3.5768	3.3571	1.1406	**0.8108**	1.0735	1.7762	0.9854
34	0.2986	0.3009	0.3493	0.3224	0.3191	0.2996	**0.2737**	0.3016	0.3024	0.3029

Tablas 9.2 Predicciones con NRMS

#	Predicción Directa					Predicción Iterada				
	Sin IMF	Con IMF 5	Con IMF 70 >	Con IMF 70<	Con IMF Todas	Sin IMF	Con IMF 5	Con IMF 70>	Con IMF 70<	Con IMF Todas
1	1.2433	5.9394	3.485	3.5673	4.1298	0.9292	1.1974	**0.7692**	1.2779	0.9831
2	2.0306	2.2752	2.8843	2.1545	2.2893	**0.8388**	6.0906	2.9299	10.089	3.2214
3	1.1425	2.0792	6.1494	2.0679	10.046	3.7359	**1.0299**	1.154	1.0488	12.238
4	1.6569	1.192	1.7767	1.1426	1.1712	1.3158	**1.0361**	3.994	1.2233	1.5469
5	1.511	2.1849	2.1932	2.3872	2.0363	1.4887	**1.1485**	1.8293	1.1922	1.5305
6	**0.9645**	1.7348	1.8004	1.8795	1.6633	1.3444	1.3081	1.6581	1.123	2.211
7	11.799	5.7457	20.388	19.501	11.46	15.244	**2.0949**	2.4033	4.8216	11.463
8	1.439	1.2197	1.3658	1.3856	1.2426	1.7095	1.4435	1.0773	1.4547	**1.045**
9	1.3585	2.1342	1.4462	2.1417	2.3491	2.9719	0.7846	1.0821	**0.7015**	6.3526
10	6.2773	3.2199	2.9384	2.9916	3.3276	3.0392	3.0254	4.7695	**2.5403**	3.4037
11	1.2868	1.3279	1.8196	2.3741	1.9461	1.7255	1.0344	**1.0139**	1.0906	3.2803
12	**0.9384**	1.4903	1.3137	1.4516	1.4536	1.6437	2.5235	3.8477	1.97	1.8714
13	1.0942	1.154	1.1025	1.0879	1.1375	**1.0147**	1.0655	1.0773	1.0446	1.1016
14	1.4391	1.6776	1.254	1.4741	1.3598	1.1522	1.1065	**1.0953**	1.1143	1.5149
15	**1.3168**	3.1836	3.6367	2.3003	2.1053	1.7214	1.9464	1.4936	3.3891	1.7713
16	0.3402	0.9844	1.6781	1.1638	1.1846	**0.3182**	0.8555	1.0823	0.6172	6.4981
17	**0.3461**	0.7359	0.6896	0.7945	0.6646	0.7547	0.8196	0.8668	1.632	0.7654
18	0.9288	1.0134	1.055	0.9135	1.0065	1.0629	0.7434	0.9799	0.9844	**0.6156**
19	0.0659	0.3953	0.9121	0.4446	0.4795	0.1346	1.7811	**0.0598**	0.2734	0.2277
20	0.1995	0.9081	0.6053	0.7070	0.5450	**0.0025**	0.0186	0.0160	0.0226	0.0413
21	0.0040	1.6392	0.5434	0.1193	0.5973	0.0199	0.8356	1.4181	**0.0025**	0.1409
22	1.0825	1.4502	1.2875	1.4062	1.3542	**0.9629**	1.1289	1.0443	1.4219	3.6961
23	0.8207	1.4926	2.1786	1.15	1.507	1.026	1.2517	1.6663	**0.7205**	1.4552
24	**0.8571**	1.1715	1.0506	1.752	0.9924	1.0003	3.1107	1.1547	1.0152	0.9571
25	0.2179	0.6097	0.6948	0.9298	0.6991	0.4503	**0.1890**	0.9849	1.4044	0.6151
26	1.794	1.8604	**0.6378**	2.0821	2.3462	1.4742	0.8905	0.9120	1.627	6.7114
27	1.5504	0.9546	**0.9368**	1.8752	1.1328	1.0795	1.1461	2.5478	1.2217	0.9746
28	0.8866	0.9254	1.0739	1.0991	1.131	0.8747	0.9930	1.2173	**0.7822**	1.8065
29	1.1846	1.0655	1.0919	1.1835	1.0909	1.0905	1.376	1.1568	1.3116	**1.0604**
30	0.6576	1.2352	2.3864	1.4429	1.3926	**0.3081**	0.8347	0.6362	0.8344	0.7599
31	1.027	1.1187	1.093	1.0945	1.1059	1.0601	1.042	1.0093	1.2317	**0.9238**
32	**0.0314**	0.6296	0.0333	0.3797	0.0989	0.0741	0.0743	0.0684	0.0794	0.0519
33	1.4663	5.9639	5.7881	5.2338	5.5179	2.686	**1.2656**	2.71	1.2668	3.354
34	1.1043	1.3741	1.1198	1.1016	1.0885	1.2138	1.1314	1.1288	1.1788	**1.0382**

A continuación se mostraran los parámetros con los que se pueden construir la Redes antes mencionadas, únicamente se comentan las mejores Redes por renglón, para cada tabla, debido a que fueron las más precisas en la predicción. Los “puntos y coma” (;) indican el salto de línea, así se pueden leer los parámetros como se explicó en el capítulo VI. Algoritmo GANN.

Tabla 9.3 Parámetros de las mejores Redes encontradas con RMSE

#	Error	Entradas	Retardos	*bias*	Layer connect	Nodos/Capa
1	1.1439	3	1	[0;0;0]	[0 0 0;1 0 0;1 1 0]	[11;11;1]
2	138.16	1	2	[0;1]	[0 0;1 0]	[8;1]
3	5.8616	2	0	[1;1;0]	[0 0 0;1 0 0;1 1 0]	[4;6;1]
4	1.5091	4	0	[1;1;1]	[0 0 0;1 0 0;1 1 0]	[5;9;1]
5	11.2350	3	1	[0;1]	[0 0;1 0]	[14;1]
6	1.7547	7	2	[0;1;1]	[0 0 0;1 0 0;1 1 0]	[11;13;1]
7	3.3844	4	0	[1;1]	[0 0;1 0]	[7;1]
8	0.2735	4	0	[1;1]	[0 0;1 0]	[5;1]
9	0.7995	6	3	[0;1;1]	[0 0 0;1 0 0;1 1 0]	[10;1;1]
10	1.0188	4	0	[1;0;1]	[0 0 0;1 0 0;1 1 0]	[9;14;1]
11	1.8834	1	2	[1;0;1]	[0 0 0;1 0 0;0 1 0]	[11;5;1]
12	21.7550	7	3	[1;0;0]	[0 0 0;1 0 0;1 1 0]	[4;15;1]
13	6.3652	1	2	[1;0]	[0 0;1 0]	[1;1]
14	2.3025	7	1	[1;1;1]	[0 0 0;1 0 0;0 1 0]	[9;12;1]
15	8.0680	1	1	[0;0;1]	[0 0 0;1 0 0;0 1 0]	[12;5;1]
16	0.0340	7	3	[1;0;1]	[0 0 0;1 0 0;0 1 0]	[11;9;1]
17	0.1251	7	1	[0;1;1]	[0 0 0;1 0 0;0 1 0]	[14;11;1]
18	0.2316	1	0	[1;1;1]	[0 0 0;1 0 0;1 1 0]	[1;1;1]
19	0.2075	5	0	[1;0;1]	[0 0 0;1 0 0;0 1 0]	[6;9;1]
20	0.0017	7	1	[1;0;1]	[0 0 0;1 0 0;1 1 0]	[14;11;1]
21	0.0043	5	1	[1;1;1]	[0 0 0;1 0 0;1 1 0]	[4;8;1]
22	0.4630	7	3	[0;1;1]	[0 0 0;1 0 0;0 1 0]	[5;1;1]
23	0.6229	6	0	[0;0;1]	[0 0 0;1 0 0;1 1 0]	[12;4;1]
24	0.2370	7	0	[0;1;0]	[0 0 0;1 0 0;0 1 0]	[12;5;1]
25	10.664	7	1	[1;0;0]	[0 0 0;1 0 0;1 1 0]	[14;9;1]
26	65.263	5	0	[0;0;1]	[0 0 0;1 0 0;0 1 0]	[13;3;1]
27	4482.60	1	3	[1;0;0]	[0 0 0;1 0 0;0 1 0]	[5;15;1]
28	0.2149	5	3	[0;1;0]	[0 0 0;1 0 0;1 1 0]	[11;13;1]
29	0.9966	5	3	[1;1;1]	[0 0 0;1 0 0;0 1 0]	[3;6;1]
30	0.0973	7	0	[0;0;0]	[0 0 0;1 0 0;1 1 0]	[11;15;1]
31	5.4359	6	0	[0;0;0]	[0 0 0;1 0 0;1 1 0]	[1;2;1]
32	0.3577	5	3	[0;1;1]	[0 0 0;1 0 0;1 1 0]	[5;9;1]
33	0.8108	7	1	[1;0;0]	[0 0 0;1 0 0;0 1 0]	[10;10;1]
34	0.2738	2	2	[0;1]	[0 0;1 0]	[7;1]

Tabla 9.4 Parámetros de las mejores Redes encontradas con NRMS

#	Error	Entradas	Retardos	*bias*	Layer connect	Nodos/Capa
1	0.7692	1	1	[0;0;1]	[0 0 0;1 0 0;0 1 0]	[3;1;1]
2	0.8388	1	3	[1;0;0]	[0 0 0;1 0 0;1 1 0]	[1;9;1]
3	1.0299	4	1	[1;1;0]	[0 0 0;1 0 0;1 1 0]	[7;8;1]
4	1.0361	5	1	[0;1;0]	[0 0 0;1 0 0;1 1 0]	[10;1;1]
5	1.1485	1	2	[1;1;1]	[0 0 0;1 0 0;1 1 0]	[1;1;1]
6	0.9645	1	3	[0;0]	[0 0;1 0]	[4;1]
7	2.0949	6	1	[0;0;1]	[0 0 0;1 0 0;1 1 0]	[14;11;1]
8	1.045	3	0	[1;0;1]	[0 0 0;1 0 0;0 1 0]	[1;8;1]
9	0.7015	5	1	[1;1;1]	[0 0 0;1 0 0;0 1 0]	[11;10;1]
10	2.5403	1	2	[0;0;0]	[0 0 0;1 0 0;0 1 0]	[3;10;1]
11	1.0139	5	0	[0;0]	[0 0;1 0]	[5;1]
12	0.9384	7	2	[0;0;0]	[0 0 0;1 0 0;1 1 0]	[4;11;1]
13	1.0147	3	2	[0;1;1]	[0 0 0;1 0 0;1 1 0]	[2;9;1]
14	1.0953	1	2	[0;1;1]	[0 0 0;1 0 0;1 1 0]	[4;1;1]
15	1.3168	6	0	[0;0;0]	[0 0 0;1 0 0;0 1 0]	[14;2;1]
16	0.3182	6	0	[0;1;1]	[0 0 0;1 0 0;0 1 0]	[13;15;1]
17	0.3461	7	1	[1;1;0]	[0 0 0;1 0 0;1 1 0]	[8;12;1]
18	0.6156	1	2	[1;0;0]	[0 0 0;1 0 0;0 1 0]	[14;8;1]
19	0.0598	2	2	[1;1;0]	[0 0 0;1 0 0;1 1 0]	[5;5;1]
20	0.0025	7	2	[1;1;1]	[0 0 0;1 0 0;1 1 0]	[15;7;1]
21	0.0025	7	3	[1;0;0]	[0 0 0;1 0 0;1 1 0]	[3;13;1]
22	0.9629	5	3	[1;1;0]	[0 0 0;1 0 0;0 1 0]	[13;1;1]
23	0.7206	4	0	[1;1;0]	[0 0 0;1 0 0;1 1 0]	[13;11;1]
24	0.8571	7	0	[1;1;0]	[0 0 0;1 0 0;0 1 0]	[14;7;1]
25	0.1890	5	1	[1;1;1]	[0 0 0;1 0 0;1 1 0]	[5;13;1]
26	0.6378	7	1	[0;0;1]	[0 0 0;1 0 0;0 1 0]	[1;2;1]
27	0.9368	7	0	[0;0;0]	[0 0 0;1 0 0;1 1 0]	[1;6;1]
28	0.7822	7	3	[0;0;1]	[0 0 0;1 0 0;1 1 0]	[13;1;1]
29	1.0604	3	1	[1;1;0]	[0 0 0;1 0 0;1 1 0]	[1;11;1]
30	0.3081	7	3	[1;0;1]	[0 0 0;1 0 0;1 1 0]	[13;15;1]
31	0.9238	4	0	[0;0]	[0 0;1 0]	[1;1]
32	0.0314	7	2	[0;1;1]	[0 0 0;1 0 0;0 1 0]	[8;15;1]
33	1.2656	6	3	[0;0;1]	[0 0 0;1 0 0;0 1 0]	[15;1;1]
34	1.0382	4	3	[1;0;0]	[0 0 0;1 0 0;1 1 0]	[1;2]

9.3 Gráficas.

9.3.1 Gráficas de la Segunda Etapa.

De la sección 7.3.2, se presenta la mejor predicción sin IMF para la Serie Mackey, a continuación se presenta la predicción con IMF, para el mismo error y la misma forma de predicción.

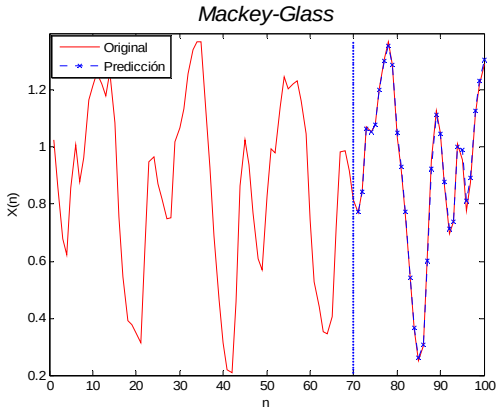
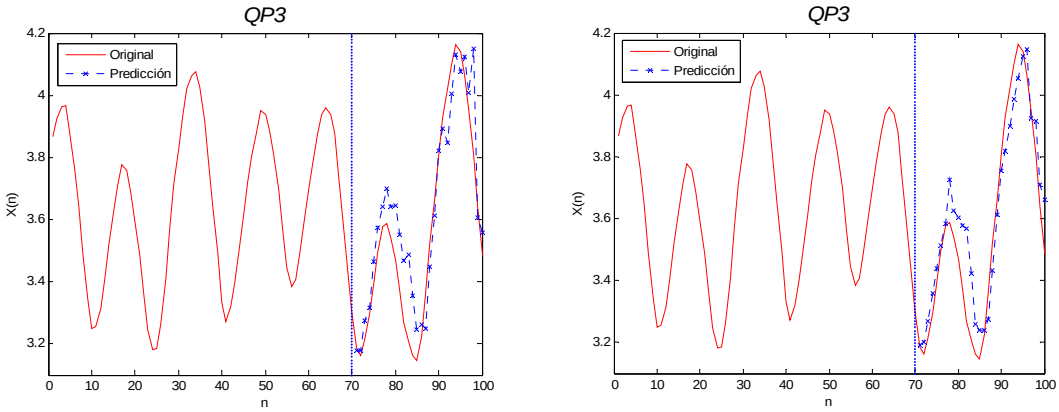


Figura 9-1. ST 10 Mackey Glass con EAP = 0.011986, con IMF y Predicción iterada

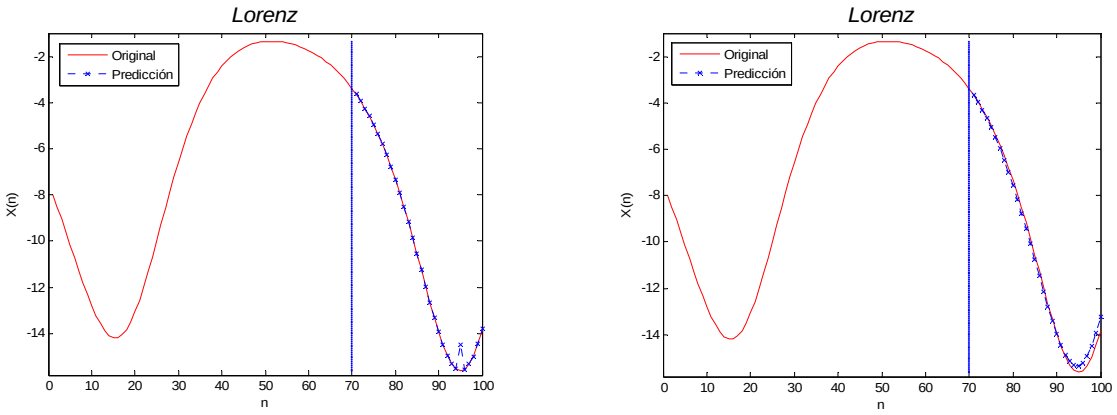
9.3.2 Gráficas de la Tercera Etapa.



a) Predicción directa sin IMF
RMSE = 0.1251

b) Predicción directa sin IMF
NRMS = 0.3461

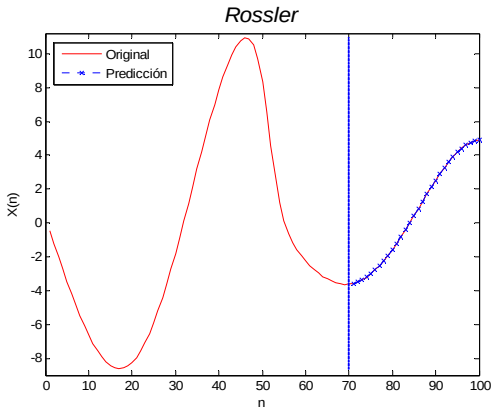
Figura 9-2. ST 17 QP3, las mejores predicciones con ambos tipos de error.



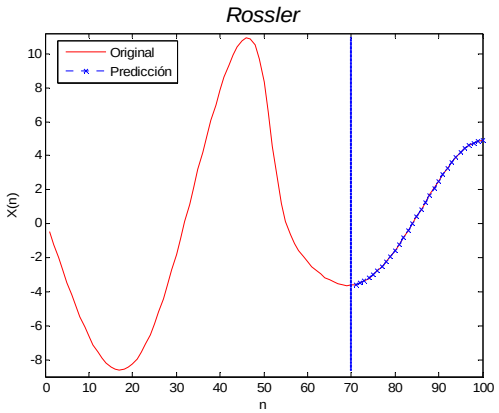
a) Predicción directa sin IMF
RMSE = 0.2075

b) Predicción iterada con 70% de IMFs de mayor amplitud total, NRMS = 0.0598

Figura 9-3. ST 19, Lorenz, las mejores predicciones con ambos tipos de error.

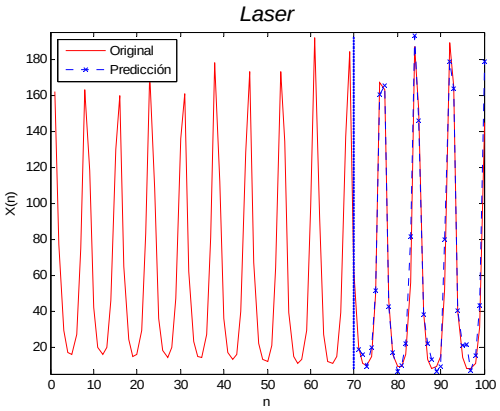


a) Predicción directa sin IMF
RMSE = 0.0043

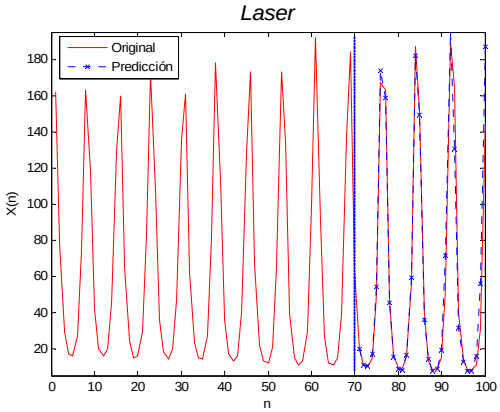


b) Predicción iterada con el 70% de IMFs de menor amplitud total, NRMS = 0.0025

Figura 9-4. ST 21, Rossler, las mejores predicciones con ambos tipos de error.

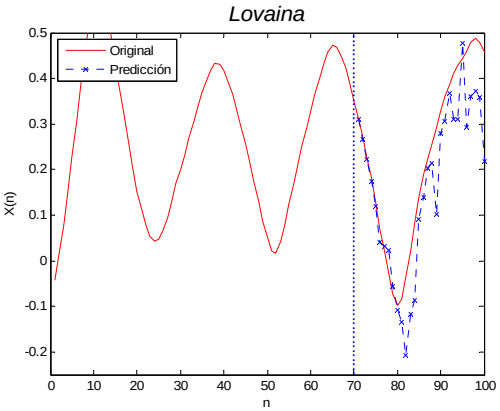


a) Predicción directa sin IMF
RMSE = 10.664

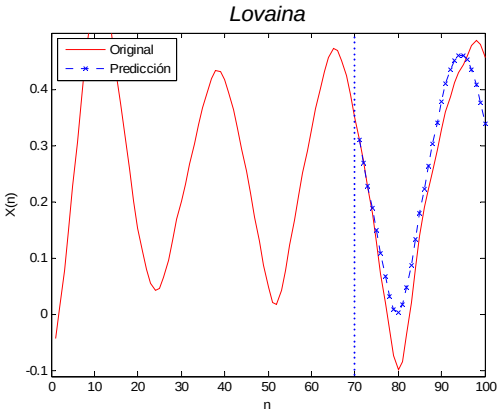


b) Predicción iterada con 5 IMFs
NRMS = 0.1890

Figura 9-5. ST 25, Laser, las mejores predicciones con ambos tipos de error.

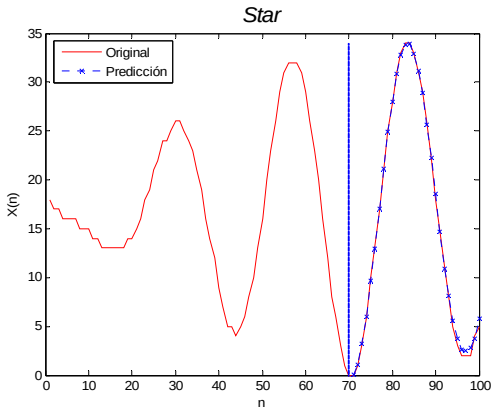


a) Predicción directa sin IMF
RMSE = 0.0973

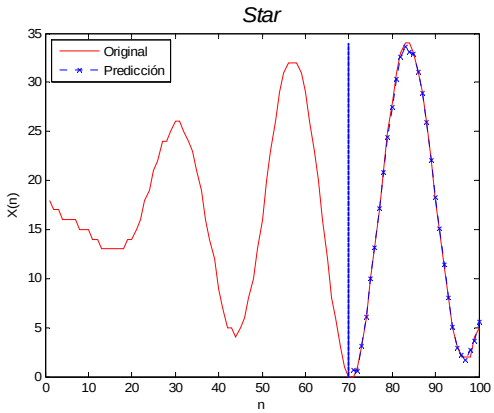


b) Predicción iterada sin IMF
NRMS = 0.3081

Figura 9-6. ST 30, Lovaina, las mejores predicciones con ambos tipos de error.



a) Predicción iterada sin IMF
RMSE = 0.3577



b) Predicción directa sin IMF
NRMS = 0.0314

Figura 9-7. ST 32, Star, las mejores predicciones con ambos tipos de error.

9.4 Series de Tiempo.

A continuación se presenta una descripción de las Series de Tiempo empleadas, acompañadas por su gráfica correspondiente, la longitud de los datos en algunos casos ha sido redimensionada para mostrar con mayor claridad el comportamiento de la serie.

La cantidad de datos disponibles representa el número de datos utilizados para realizar el entrenamiento y prueba de las Redes; en las que se tiene un valor de 1500, significa que se contaban con más datos, pero como se explicó anteriormente, sólo se ocupa dicha cantidad como máximo. No todas las series presentan los mismos campos, debido a que su origen es diferente y no todos proporcionan la misma información del fenómeno.

Las series que no cuentan con el campo “Descripción”, significa que no se tiene más información que el título para describirlas. El campo “Dinámica” de algunas series se obtuvo de [8, 27].

1 **Agregado semanal: Índice en Horas**

Descripción: Se refiere a la situación de los empleados en cuanto a horas trabajadas del total de Industrias privadas.
Categoría: Financiera.
Origen: Natural
Frecuencia: a pesar del nombre de la serie esta tiene una frecuencia mensual y no semanal como podría parecer.
Fuente: U.S. Department of Labor: Bureau of Labor Statistics.
Rango de fechas: 01/01/1964 al 01/02/2005.
Datos disponibles: 494.
IMFs: 5
Unidades: Horas.

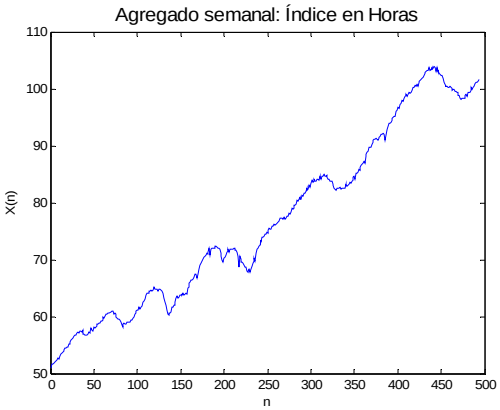


Figura 9-8

2 **Inversión total de todos los Bancos Comerciales.**

Descripción: para bancos de Estados Unidos
Categoría: Financiera.
Origen: Natural.
Frecuencia: Mensual.
Fuente: Board of Governors of the Federal Reserve System.
Rago de fechas: 01/01/1947 al 01/01/2005.
Datos disponibles: 697.
IMFs: 5
Unidades: Billones de Dólares

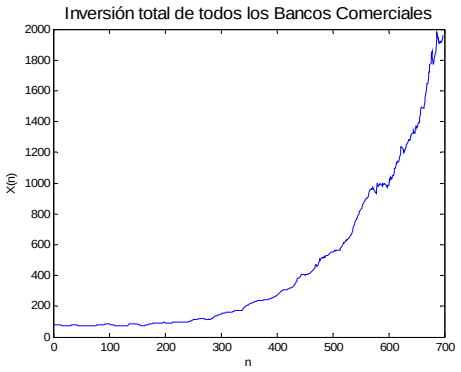


Figura 9-9

3 **Precio diario del oro.**

Categoría: Financiera.
Origen: Natural
Frecuencia: Diario.
Rango de Fechas: 01/01/1985 al 31/03/1989.
Datos disponibles: 1074.
IMFs: 7
Unidades: Dólares.

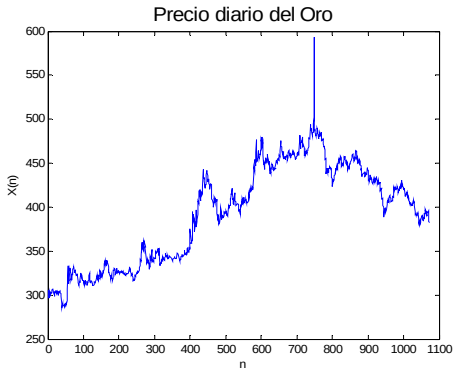


Figura 9-10

4 **Precio de las acciones de IBM.**

Categoría: Financiera
Origen: Natural
Frecuencia. Diario.
Fuente: Hipel and Mcleod (1994)
Rango de fechas: 01/01/1980 al 08/08/1992
Datos disponibles: 1500.
IMFs: 8

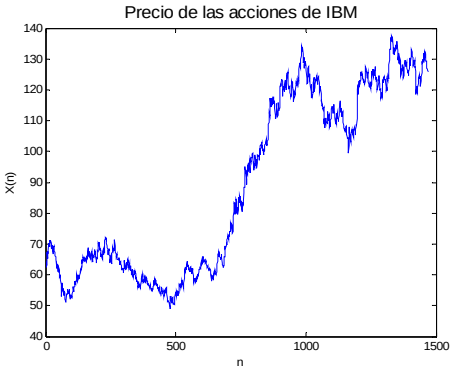


Figura 9-11

5 **Salarios diarios de Inglaterra.**

Descripción: Salarios diarios reales de Inglaterra.
Categoría: Fianciera.
Origen: Natural
Frecuencia: por día
Fuente: Makridakis, Wheelwright and Hyndman (1998)
Rango de fechas: 1260 a 1994
Datos disponibles: 735.
IMFs: 7
Unidades: Libras

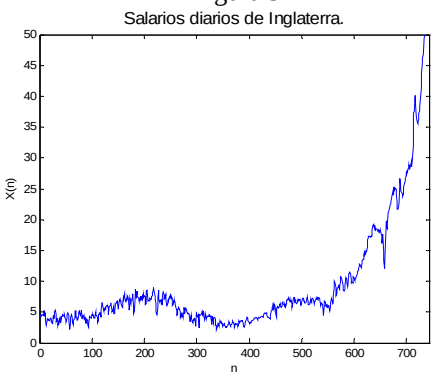


Figura 9-12

6 **SP500**

Descripción: índice financiero de Standard & Pool de las 500 empresas más importantes de NJ.
Categoría: Financiera.
Origen: Natural
Frecuencia: por día.
Fuente: Hipel and Mcleod (1994)
Rango de fechas: 06/06/1960 al 06/06/2000
Datos disponibles: 1500.
IMFs: 7
Dinámica: Compleja

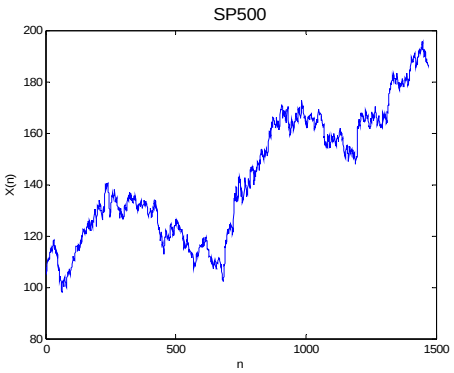


Figura 9-13

7 **Flujo diario del río Jokulsa**

Descripción: Flujo diario medio del rio Jokulsa Eystri
Categoría: Hidrológica.
Origen: Natural
Frecuencia: por día.
Fuente: Hipel and Mcleod (1994)
Rango de fechas: 01/01/1972 al 31/12/1974
Datos disponibles: 1096.
IMFs: 9
Dinámica: Compleja
Unidades: cm.

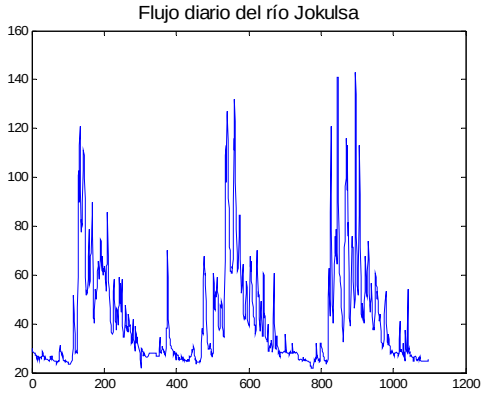


Figura 9-14

8 **Flujo mensual del río Colorado.**

Categoría: Hidrológica.
Origen: Natural
Frecuencia: Mensual
Fuente: McMahon and Mein, pp.351-352
Rango de fechas: 1911 - 1972
Datos disponibles: 744.
IMFs: 7

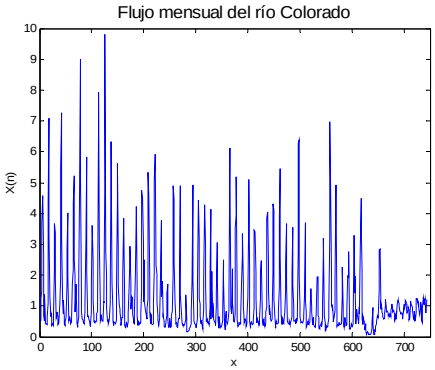


Figura 9-15

9 **Nivel de Agua Mensual del lago Erie**

Categoría: Hidrológica.
Origen: Natural
Frecuencia: Mensual
Rango de fechas: 1921 - 1970
Datos disponibles: 600.
IMFs: 6

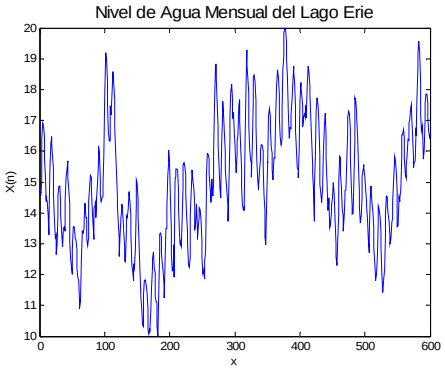


Figura 9-16

10 **Nivel Mínimo Anual del río Nilo.**

Categoría: Hidrológica.
Origen: Natural
Frecuencia: Anual
Fuente: Hipel and Mcleod (1994).
Rango de fechas: 622 - 1921
Datos disponibles: 1297.
IMFs: 8

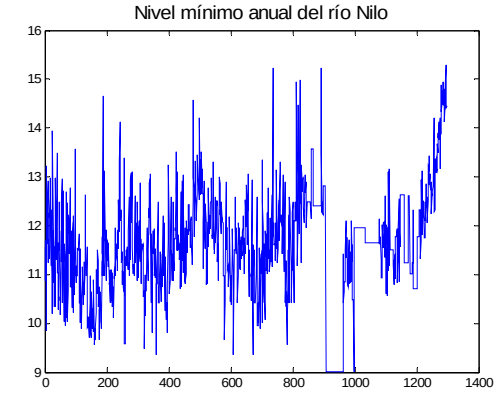


Figura 9-17

11 **Precipitación diaria Hveravellir.**

Categoría: Meteorológica.
Origen: Natural
Frecuencia: por día
Fuente: Hipel and Mcleod (1994).
Rango de fechas: 01/01/1972 al 31/12/1974
Datos disponibles: 1096.
IMFs: 9
Unidades: mm

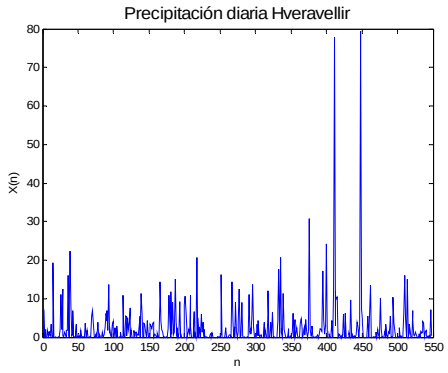


Figura 9-18

12 **Número de manchas solares mensuales.**

Descripción: Promedio Mensual de manchas solares por día
Categoría: Física.
Origen: Natural
Frecuencia: Mensual
Fuente: Andrews & Herzbe (1985).
Rango de fechas: Enero 1749 a Marzo 1977
Datos disponibles: 1500.
IMFs: 8

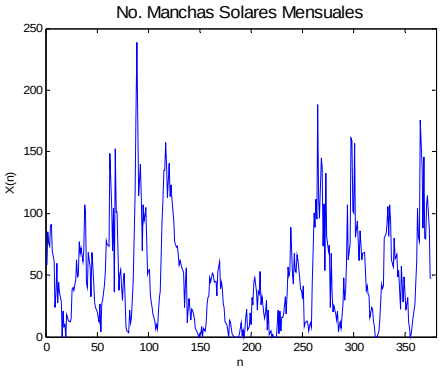


Figura 9-19

13 **Temperatura máxima en Melbourne**

Descripción: Temperatura máxima diaria en Melbourne - Australia
Categoría: Meteorológica.
Origen: Natural
Frecuencia: por día.
Fuente: Australian Bureau of Meteorology
Rango de fechas: 1981-1990
Datos disponibles: 1500.
IMFs: 8
Unidades: °C

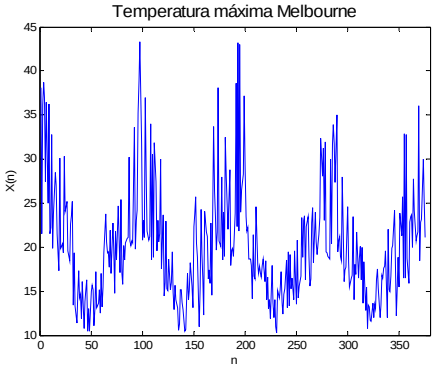


Figura 9-20

14 **Temperatura mínima en Melbourne.**

Descripción: Temperatura mínima diaria en Melbourne - Australia
Categoría: Meteorológica.
Origen: Natural
Frecuencia: por día.
Fuente: Australian Bureau of Meteorology
Rango de fechas: 1981-1990
Datos disponibles: 1500.
IMFs: 8
Unidades: °C

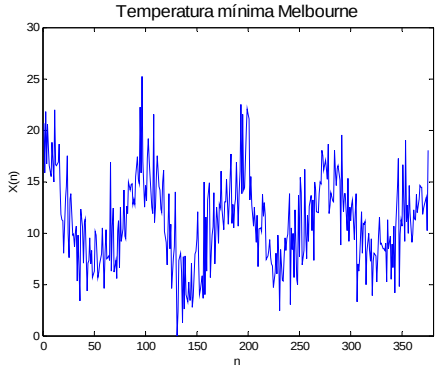


Figura 9-21

15 **Southern oscillation.**

Descripción: Oscilaciones mensuales en el meridiano, índice de diferencia de presión sobre la superficie del mar entre Darwin y Taití.
Categoría: Meteorológica.
Origen: Natural
Frecuencia: Mensual
Rango de fechas: Enero 1882 a Mayo 1993
Datos disponibles: 1325.
IMFs: 8

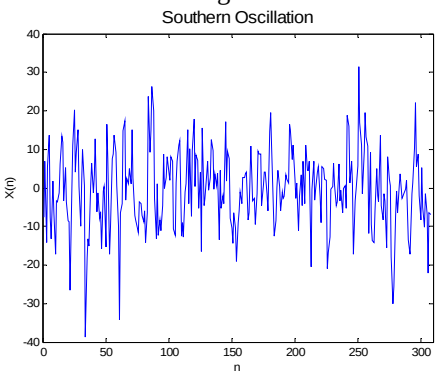


Figura 9-22

16 **QP2.**

Descripción: variable de velocidad en un experimento anular para reproducir el flujo de Couette.
Origen: Artificial
Frecuencia: 0.1 s
Datos disponibles: 1500.
IMFs: 7
Dinámica: Cuasi periódica
Unidades: cm/s

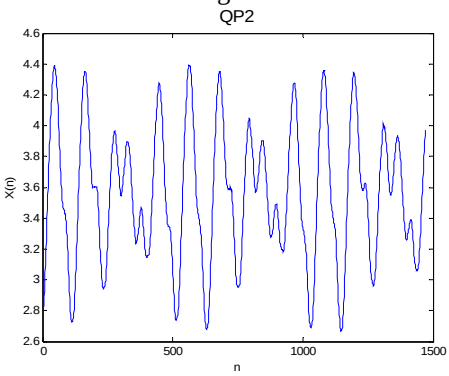


Figura 9-23

17 **QP3.**

Descripción: variable de velocidad en un experimento anular para reproducir el flujo de Couette.
Origen: Artificial
Frecuencia: 0.4 s
Datos disponibles: 1500.
IMFs: 7
Dinámica: Cuasi periódica
Unidades: cm/s

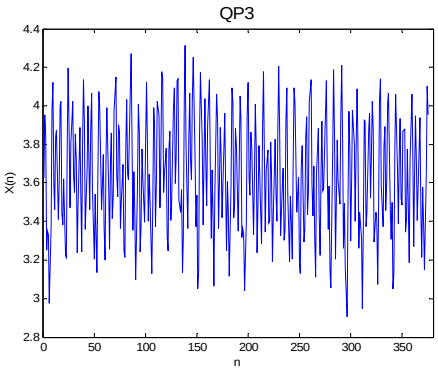


Figura 9-24

18 **Logistic.**

Descripción: Serie generada por un mapa.
Origen: Artificial
Datos disponibles: 1000.
IMFs: 8
Dinámica: Caótica

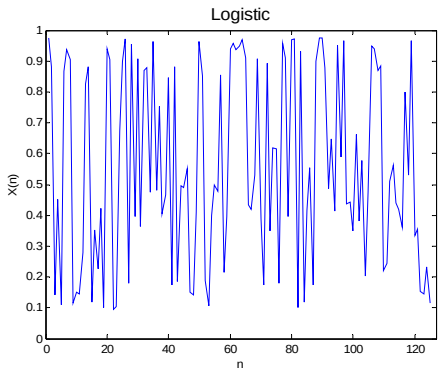


Figura 9-25

19 **Lorenz.**

Descripción: Serie generada por un sistema de ecuaciones diferenciales, modelo de convección de fluidos (Rayleigh-Benard)
Origen: Artificial
Datos disponibles: 1500.
IMFs: 5
Dinámica: Caótica

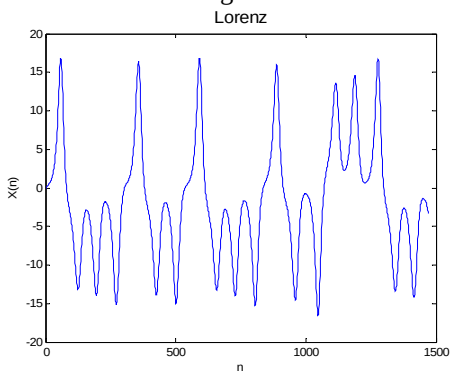


Figura 9-26

20 **Mackey-Glass.**

Descripción: Serie generada por una ecuación diferencial de retardo temporal, modelo de formación de Linfocitos.
Origen: Artificial
Datos disponibles: 1500.
IMFs: 7
Dinámica: Caótica.
Unidades:

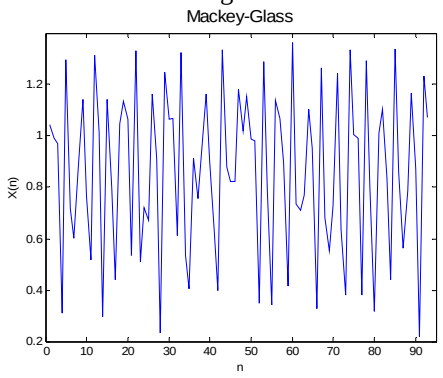


Figura 9-27

21 **Rossler.**

Descripción: Serie generada a partir de un sistema de ecuaciones diferenciales para el modelo simplificado de Lorenz
Origen: Artificial
Datos disponibles: 1500.
IMFs: 5
Dinámica: Caótica

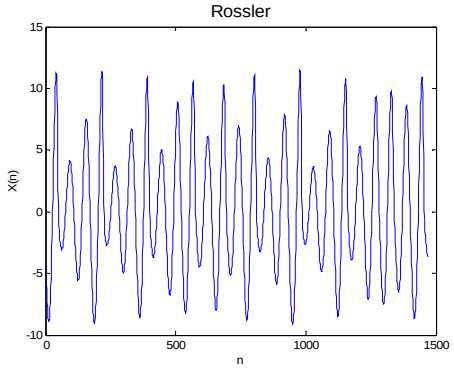


Figura 9-28

22 **Ikeda.**

Descripción: Serie generada a partir de un mapa construido en un plano complejo.
Origen: Artificial
Datos disponibles: 1500.
IMFs: 9
Dinámica: Caótica

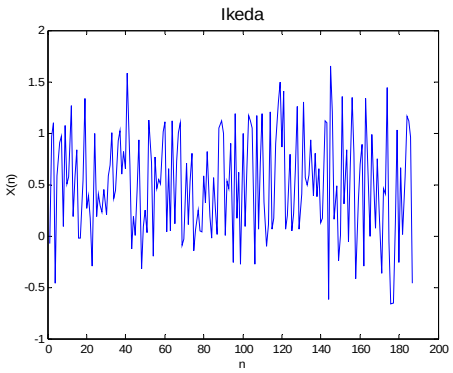


Figura 9-29

23 **Henon.**

Descripción: Serie construida a partir del modelo simplificado del mapa de Poncaré para el modelo de lorenz.
Origen: Artificial
Datos disponibles: 1500.
IMFs: 10
Dinámica: Caótica

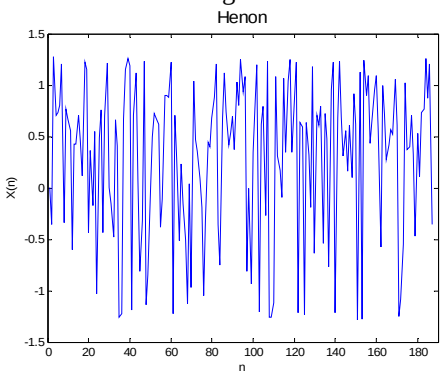


Figura 9-30

24 **D1.**

Descripción: Serie del concurso de Santa Fe
Origen: Artificial
Datos disponibles: 1500.
IMFs: 8
Dinámica: Compleja

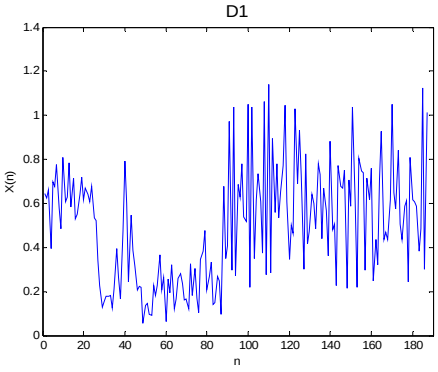


Figura 9-31

25 **Laser.**

Descripción: Serie del concurso Santa Fe
Origen: Artificial
Datos disponibles: 1500.
IMFs: 7
Dinámica: Compleja

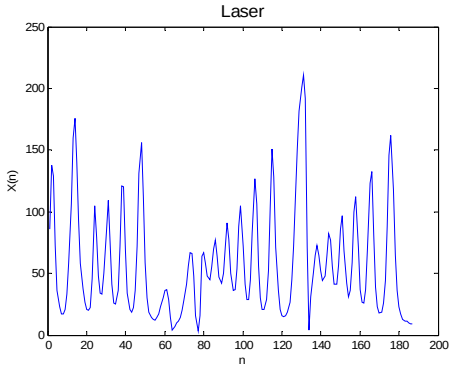


Figura 9-32

26 **Down Jones.**

Categoría: Financiera.
Origen: Natural
Frecuencia: por día
Fuente: Brockwell and Davis
Rango de fechas: Agosto 28 a Diciembre 18 de 1972.
Datos disponibles: 1500.
IMFs: 8
Dinámica: Compleja

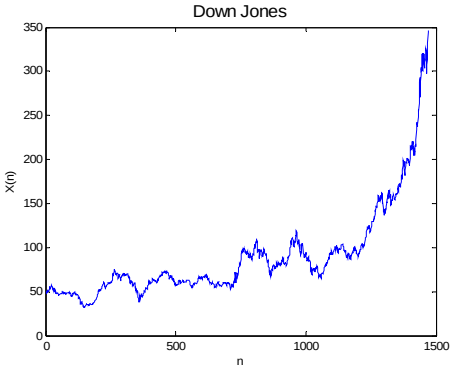


Figura 9-33

27 **Kobe.**

Descripción: Serie de un acelerograma del sismo de Kobe, el 16/01/1995.
Categoría: Sismos.
Origen: Natural
Frecuencia: segundos
Datos disponibles: 1500.
IMFs: 9
Dinámica: Compleja

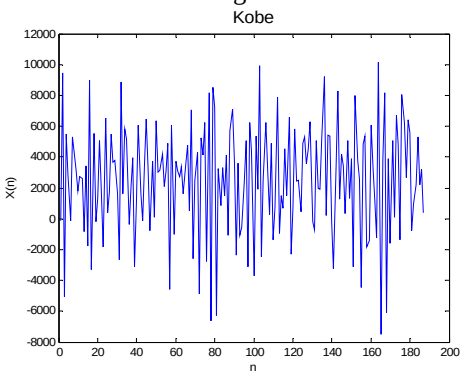


Figura 9-34

28 **EEG.**

Descripción: Serie de un encefalograma humano.
Origen: Natural
Datos disponibles: 1000.
IMFs: 8
Dinámica: Compleja

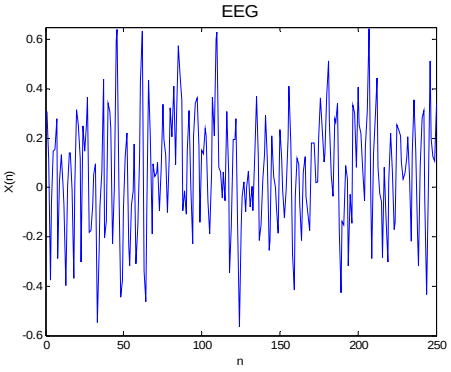


Figura 9-35

29 **HDNA.**

Descripción: DNA humano.
Origen: Natural
Datos disponibles: 1500.
IMFs: 8
Dinámica: Compleja

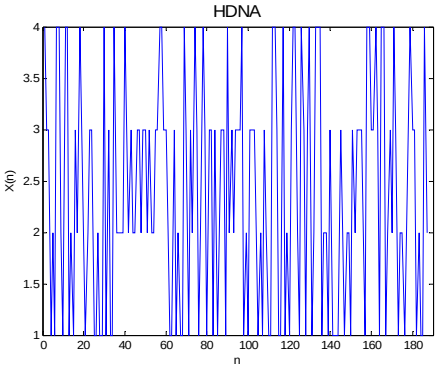


Figura 9-36

30 **Lovaina.**

Descripción: Concurso de la universidad de Lovaina.
Origen: Artificial
Datos disponibles: 1500.
IMFs: 6
Dinámica: Compleja

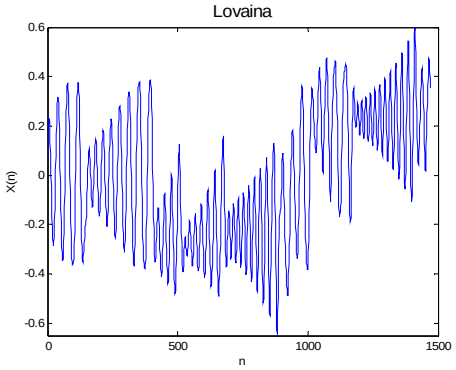


Figura 9-37

31 **Primos.**

Descripción: Serie creada a partir de los números primos.
Origen: Artificial
Datos disponibles: 1500.
IMFs: 10
Dinámica: Compleja

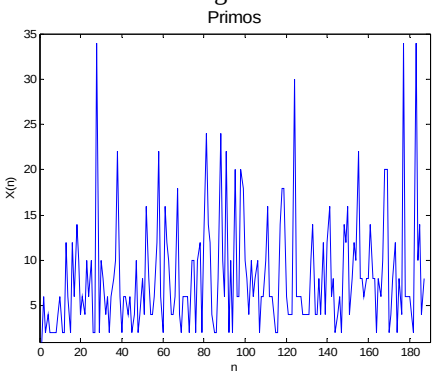


Figura 9-38

32 **Star.**

Descripción: Intensidad luminosa de una estrella.
Categoría: Física.
Origen: Natural
Datos disponibles: 600.
IMFs: 6
Dinámica: Compleja

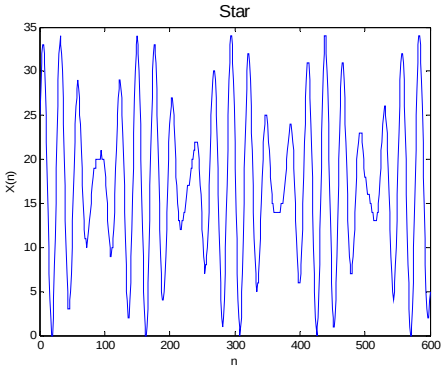


Figura 9-39

33 **Brownian Motion.**

Descripción: Movimieto Browniano (Einstein)
Origen: Artificial
Datos disponibles: 1000.
IMFs: 7
Dinámica: Estocástica.

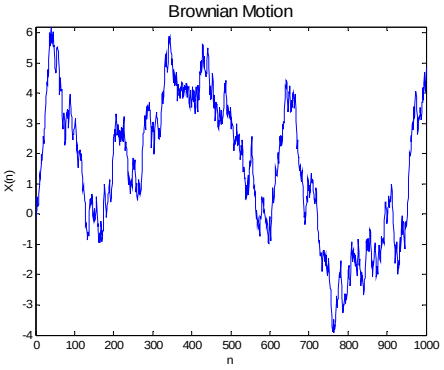


Figura 9-40

34 **Ruido Blanco.**

Descripción: ruido aleatorio uniforme
Origen: Artificial
Datos disponibles: 1000.
IMFs: 9
Dinámica: Estocástica

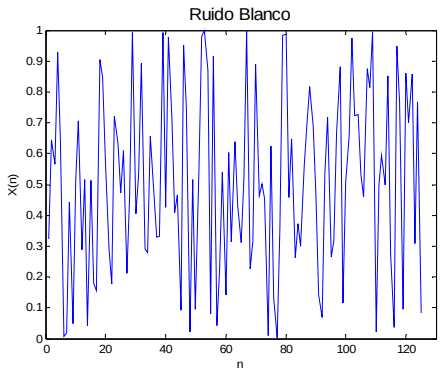


Figura 9-41

9.5 Glosario.

Términos de las ANNs

Back-Propagation (BP): Es un método de entrenamiento supervisado en el cual el error de salida, es retro-propagado (hacia atrás) en la red, alterando los pesos, hasta minimizar el error.

Capa de Entradas: Es un conjunto de nodos, que permiten introducir las entradas a la Red.

Capa(s) oculta(s): Es la capa que no está conectada directamente ni a las entradas ni a las salidas.

Capa de Salida: Es la capa de nodos que produce la salida de la Red. A la salida de esta capa, se conectan los datos deseados (targets) para realizar el entrenamiento supervisado.

Conexión: Es una liga que conecta a los nodos y sirve para pasar información de uno a otro, cada conexión tiene asociado un valor llamado peso.

Épocas: Son las iteraciones de las Redes Neuronales para realizar el entrenamiento, donde el conjunto de prueba es presentado a la Red Neuronal.

Entrenamiento: Es el proceso en el cual se le hace pasar repetidamente un conjunto de datos a la Red, cambiando sus pesos para mejorar el rendimiento.

Generalización: Es la habilidad de una Red Neuronal de responder correctamente a datos que no se han usado en el entrenamiento.

Gradiente Descendiente: Es un proceso de aprendizaje que cambia los pesos de una Red Neuronal, dando "pasos" a través de un camino, para llegar a un punto mínimo de error.

Nodo: es una simple neurona, o elemento de procesamiento, consta usualmente de varias entradas y varias salidas, las entradas son multiplicadas por el peso, posteriormente se puede o no sumar el *bias*, y al resultado pasa por una función de transferencia antes de salir.

Peso: Es un valor ajustable asociado a cada conexión entre los nodos en la Red.

Sobre entrenamiento: Es cuando la Red memoriza los datos de entrenamiento y no es capaz de generalizar.

Términos del GA

Adaptabilidad: Es el valor que nos regresa la función de adaptabilidad; nos dice qué tan apto es el individuo para ser candidata a la solución.

Diversidad: Se refiere a la distancia promedio entre los individuos de la población del GA.

Fenotipo: Es el problema es sí, en nuestro caso el Fenotipo es la Red Neuronal ya construida; así la ANN es evaluada con la función de Adaptación.

Función de Adaptabilidad (fitness function): Esta función permite evaluar a los individuos del GA para determinar qué tan aptos son para resolver nuestro problema (función a optimizar o función objetivo), en sí la función de Adaptabilidad es aquella que resuelve nuestro problema y el mejor individuo será aquel que nos de el valor más pequeño, si queremos minimizar nuestra función; en el caso contrario, maximizar.

Generaciones: Una generación es una iteración para el GA, donde realiza diversas operaciones con la población actual para generar la siguiente.

Genotipo: Es la representación interna del problema utilizada por el GA, para nuestro caso el Genotipo es una cadena de bits.

Hijo(s): Son los individuos de la siguiente población dentro del GA, creados a partir de los padres de la generación actual.

Individuo: Es aquel, al que se le aplica la función de Adaptabilidad, dándonos normalmente un valor escalar, que nos sirve para compararlo contra los demás.

Población: Es un conjunto de individuos.

Padre(s): Los padres son individuos seleccionados de la población actual de acuerdo a su adaptabilidad bajo un criterio y son usados para generar a los hijos dentro del GA.

GA y ANNs (términos compartidos por ambos, en cuanto a optimización)

Mínimo Global: Es el error más pequeño que se puede encontrar en nuestra superficie de error.

Mínimo Local: Es un punto de una región, que tiene un error pequeño, pero no el mínimo. Se puede ver como un hundimiento o abolladura en la superficie del error.

Términos de la Descomposición Empírica en Modos (EMD)

EMD: Descomposición Empírica en Modos (Empirical Mode Decomposition), técnica de análisis de señales reciente, introducida por Huang et al en 1998. Es capaz de descomponer una señal 1-dimensional en diferentes modos oscilatorio en tiempo-frecuencia, dando un número finito y a veces pequeño de IMFs.

I**MF:** (Función de Modo Intrínseco, Intrinsic Mode Function), una IMF es una descomposición de la serie, obtenida al aplicar EMD. Se pueden tener diversas IMFs del fenómeno junto con el residuo, el cual normalmente contiene la tendencia del sistema (si es que tiene tendencia el sistema).

Términos de la estimación del Error

EAP: Error Absoluto Promedio (Mean Absolute Error).

EAM: Error Absoluto Máximo (Max Absolute Error).

RMSE: Raíz Cuadrada del Error Cuadrático Medio (Root Mean Squared Error).

NRMS: Raíz Cuadrada del Error Cuadrático Normalizado (Normalize Root Mean Squared Error).

La abreviación de RMSE y NRMS, se dejó de las siglas de su significado en inglés, y no en español como EAP y EAM, debido a la convención utilizada en la literatura.

Bibliografía

[1] Helmut A. Mayer and Roland Schwaiger, “Evolutionary and coevolutionary approaches to time series prediction using generalized multi-layer perceptrons”. In Proceedings of the Congress on Evolutionary Computation (1999). volume 1, 6-9.

[2] Doyne Farmer and John J. Sidorowich, “Predicting Chaotic Time Series”. Physical Review Letters (August 1987). 59(8), 845.

[3] G. Zhang, B. Patuwo, M. Hu, “Forecasting with artificial neural networks: The state of the art”. Elsevier, International Journal of Forecasting (1998). 14 35-62, N. H.

[4] A. Aussem and F. Murtagh, “Combining neural network forecasts on wavelet-transformed time series”. Connection Science (1997). 9(1), 113-121.

[5] B. Zhang, R. Coggins, M. A. Jabri, D. Dersch and B. Flower, “Multiresolution Forecasting for Futures Trading Using Wavelet Decompositions”. IEEE Transactions on Neural Networks (July 2001). Vol. 12, no. 4, Pág. 765.

[6] Tomas J. Cholewo and Jacek M. Zurada, “Sequential network Construction for Time Series Prediction”. *In Proccedings of the IEEE International Join Conference on Neural Networks*, Houston, Texas, USA (June 1997). Pp 2034-2039.

[7] Bikas K. Chakrabarti and Prabir K. Dasgupta, “Modelling neural networks”. Physica A 186 (1992) 33-48.

[8] I. de Falco, A. Iazzetta, P. Natale, and E. Tarantino, “Evolutionary neural networks for nonlinear dynamics modeling, in Parallel Problem Solving from Nature” (1998). Lectures Notes in Computer Science, Springer. Vol. 1498. Pp. 593-602.

[9] Harri Jäske, “One-step-ahead prediction of sunspots with genetic programming”. In Jarmo T. Alander editor, Proceedings of the Second Nordic Workshop on Genetic Algorithms and their Applications (2NWGA), pages 79-88, Vaasa (Finland), 1996. University of Vaasa.

[10] S. Soltani, S. Canu and D. Boichu, "Time series prediction and the wavelet transform". International Workshop on Advanced Black Box modelling, Leuven, Belgium, Jul 8-10, 1998.

[11] K.R. Muller, A.J. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, “Using support vector machines for time series prediction” in: B. Scholkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, 1999, pp. 243–254.

[12] Bruce A. Whitehead, “Scalar Wavelet Networks For Time Series Prediction”. University of Tennessee Space Institute Tullahoma, TN 37388.

[13] N. Huang, Z. Shen, S. Long, M. Wu, H. Shih, Q. Zheng, N. Yen, C. Tung and H. Liu, “The empirical mode decomposition and Hilbert spectrum for nonlinear and non-stationary time series analysis”. Proc. of the Royal Society of London vol. 454 (1998), pp. 903–995.

[14] D. Wolpert and W. Macready, “No free lunch theorems for search”. Santa Fe Institute, SFI-TR-02-010. 1995.

[15] D. Wolpert and W. Macready, “No Free Lunch Theorems for Optimization”. IEEE Transactions on Evolutionary Computation, April 1997. Vol. 1, No. 1.

[16] Kenneth O. Stanley and Risto Miikkulainen, “Efficient reinforcement learning through evolving neural network topologies”. In Morgan Kaufmann, editor, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002), 2002.

[17] J. McNames, “Local modeling optimization for time series prediction”. Electrical & Computer Engineering, Physical review E, (2000).

[18] S. Haykin, “Neural Networks, A Comprehensive Foundation”. Prentice Hall, Inc. (1999). Second edition. ISBN: 0-13-273350-1

[19] D. E. Rumelhart, G. E. Hinton and R. J. Williams, “Learning Representations by Back-Propagating Errors”, Nature (1986), vol. 323, pp. 533-536.

[20] Yoshua Bengio, “Using a Financial Training Criterion Rather than a Prediction Criterion”. Int. J. Neural Syst (1997). 8(4): 433-443.

[21] Feng Lin, Xing Huo Yu, Shirley Gregor and Richard Irons, “Time Series Forecasting with Neural Networks”. Complexity International (1995). Vol 2. ISSN: 1320-0682

[22] J. Branke, “Evolutionary Algorithms for Neural Networks Design and Training”. Technical Report No. 322, University of Karlsruhe, Institute AIFB.

[23] Georg Dorffner, “Neural networks for time series processing”. Neural Network World (1996), 6(4):447-468.

[24] Emad W. Saad, Danil V. Prokhorov and Donald C. Wunsch II, “Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks”. IEEE Transactions on Neural Networks, Novemver 1998. Vol 9, No. 6: 1456—1470.

[25] Filippo Castiglione, “Forecasting Price Increments Using an Artificial Neural Network”. Advances in Complex Systems (2001), 4(1), 45-56.

[26] Medina Apodaca, J. M., 2005. “Herramienta para la Extracción de Redes Bayesianas Predictivas a partir de Bases de Datos Temporales”. tesis de Maestría. Centro de Investigación en Computación, Instituto Politécnico Nacional.

[27] Bautista Thompson, E. F., 2005. “Medición de la Predictibilidad de Series de Tiempo: un estudio experimental”. tesis Doctoral. Centro de Investigación en Computación, Instituto Politécnico Nacional.

[28] Avila Sánchez, C., 2005. “Extracción Automática de Motifs para el Estudio de Fenómenos Complejos”. tesis de Maestría. Centro de Investigación en Computación, Instituto Politécnico Nacional.

[29] Menchaca Méndez, R., 2005. “Análisis de Estructura y Comportamiento de Modelos de Redes para el Estudio de Sistemas Complejos”. tesis de Maestría. Centro de Investigación en Computación, Instituto Politécnico Nacional.

[30] Eric A. Wan, Alex T. Nelson, “Dual Kalman Filtering Methods for Nonlinear Prediction, Smoothing and Estimation”. NIPS 1996: 793-799

[31] Yao Xin, “Evolving Artificial Neural Networks”. School of Computer Science. The University of Birmingham, (1999). B15 2TT.

[32] Xin Yao, “Evolutionary Artificial Neural Networks”. Int. J. Neural Syst. (1993), 4(3): 203-222.

[33] R. Belew, J. Mclnerney, N. Shraudolph, “Evolving Networks: Using the Genetic Algorithm with Connectionist Learning”. CSE Technical Report #CS90-174. Cognitive Computer Science Research Group June (1990), Computer Science & Engr. Dept. (C-014)

[34] D. Dasgupta and D.R. McGregor, “Designing Application-specific Neural Networks Using the Structured Genetic Algorithms”. Proc. of the International Workshop on Combinations of Genetic Algorithms and Neural Networks. 1992. Pp. 87-96.

[35] H. Braun and J. Weisbrod, “Evolving neural feeforward networks”. In *Proceeding of the Conference on Artificial Neural Nets and Genetic Algorithms.* Springer Verlag. (1993), pages 25-32.

[36] K. J. Hintz and J. J. Spofford, "Evolving a Neural Network", Fifth IEEE Symposium on Intelligent Control 1990, A. Meystel, J. Herath and S. Gray, eds., vol. 1, pp. 479-484, 5-7 September, 1990, Phil., PA.

[37] Juan Manuel Torres Moreno, “Redes Neuronales Constructivas que aprenden mientras crecen”. Linear Logic. Theoretical Computer Science, 50:1-102, 1987. Tor98.

[38] Hagan, M. T., and M. Menhaj, "Training feedforward networks with the Marquardt algorithm", IEEE Transactions on Neural Networks, 1994. Vol. 5, no. 6, pp. 989-993.

[39] David E. Moriarty and Risto Miikkulainen, “Forming Neural Networks Through Efficient and Adaptive Coevolution”. Evolutionary Computation, 1997. 5(4):373–399.

[40] K. Coughlin, K. Tung, “11-year solar Cycle in the stratosphere extracted by the empirical mode decomposition method”. Adv. Space Res. (2004a.), 34, 323-329.

[41] K. Coughlin and K. Tung, “Empirical Mode Decomposition of Climate Variability in the Atmospheric”. Paper in Hilbert-Huang Transform: Introduction and Applications; edited by N. Huang and S. Shen; World Scientific Publishing (2005)

[42] P. Kuchi and S. Panchanathan, “Intrinsic Mode Functions for GAIT Recognition”. International Symposium on Circuits and Systems (ISCAS '04), Vancouver, Canada, 2004.

[43] Thomas M. English, “Evaluation of Evolutionary and Genetic Optimizers: No Free Lunch”. Evolutionary Programming V: Proc. of the Fifth Annual Conf. on Evolutionary Programming, Cambridge, MA, MIT Press, In Fogel, L.J. Angeline, P.J., Bäck, T., eds. 1996 163-169.

[44] John R. Woodward, “GA or GP? That is *not* the question”. Congress on Evolutionary Computation, Canberra, Australia, 8th - 12th December 2003.

[45] Zehra Cataltepe, Yaser S. Abu-Mostafa and Malik Magdon-Ismail, “No Free Lunch for Early Stopping”. *Neural Computation* 11, (1999). 995-1009.

[46] Malik Magdon-Ismail, “No Free Lunch For Noise Prediction”. *Neural Computation* (2000). Vol 12, No. 3.

[47] S.Christensen and F. Oppacher, “What can we learn from No Free Lunch? A First Attempt to Characterize the Concept of a Searchable Function”. In L. Spector et al (eds), Proc. of GECCO 2001, Morgan Kaufmann, pp. 1219-1226. 10(3): 263-282.

[48] G. Cybenko, “Approximation by superpositions of a sigmoidal function”. Mathematical Control Signals Systems 2 (1989), 303–314

[49] A. R. Barron: A comment on ‘‘Neural networks: A review from a statistical perspective’’. Statistical Science 9 (1) (1994), 33–35.

[50] Jose C. Principe, Neil R. Euliano, W. Curt Lefebvre. Neural and Adaptive Systems: Fundamentals through Simulations. John Wiley Sons, Inc. 2000 USA, ISBN: 0-471-35167-9 (paper)

[51] D. Anderson and G. McNeill, “Artificial Neural Networks Technology”. A DACS State-of-the-Art Report. Contract Number F30602-89-C-0082. ELIN: A011. 1992. Prepared for: Rome Laboratory RL/C3C. Griffiss AFB, NY 13441-5700.

[52] D. W. Marquardt. “An algorithm for least-squares estimation of nonlinear parameters”. Journal of the Society for Industrial and Applied Mathematics, 11(2):431--441, 1963.

[53] E. Alba, J. F. Chicano, “Training Neural Networks with GA Hybrid Algorithms”. K. Deb (ed.), Proceedings of GECCO’04, Seattle, Washington, 2004. LNCS 3102, pp. 852-863.

[54] G. N. Yannakakis, J. Levine, J. Hallam, and M. Papageorgiou, "Performance, robustness and effort cost comparison of machine learning mechanisms in FlatLand," in Proceedings of the 11th Mediterranean Conference on Control and Automation MED'03. IEEE, June 2003.

[55] Spyros Makridakis & Steven C. Wheelwright. “Forecasting Methods & Applications”. John Wiley & Son. 1978. ISBN: 0-471-93770-3

[56] V. Landassuri-Moreno, J. Figueroa-Nazuno, “Genetic Algorithm to Design Artificial Neural Networks in Time Series Forecasting with Intrinsic Mode Functions”. Research on Computing Science. Advances in Computer Sciences in Mexico. May (2005). Vol. 13, pp. 101-110. ISSN: 1665-9899.

[57] V. Landassuri-Moreno & J. Figueroa-Nazuno, “GAAN en la predicción de Series de Tiempo con funciones de Modo Intrínseco”. XLVIII Congreso Nacional de Física. Guadalajara, Jalisco, Octubre de 2005. 3MG3, pág. 110.

[58] V. Landassuri-Moreno & J. Figueroa-Nazuno, “Algoritmo Hibrido GANN en la Predicción de Series de Tiempo con Descomposición Empírica en Modos”. ROC&C’2005. Acapulco, Guerrero. Diciembre de 2005. CP-41.

[59] J. Holland, “Adaptation In Natural and Artificial System”. The University of Michigan Press, Ann Arbor.1975

[60] A. Kuri Morales, “A Comprehensive Approach to Genetic Algorithms in Optimization and Learning, Theory and Applications”. Colección de Ciencia de la Computación, Centro de Investigación en Computación, Instituto Politécnico Nacional. México (1999). Vol 1. Fundations.

[61] I. Magrin, R. Baraniuk, “Empirical mode decomposition based time–frequency attributes”. Rice University, Houston Texas (1999)

[62] P. Flandrin, Rilling and Goncalves, P., “Empirical Mode Decomposition as a filter bank”. IEEE Sig. Proc Lett., (2003)

[63] H. Solís-Estrella, A. Angeles-Yreta, V. Landassuri-Moreno and J. Figueroa-Nazuno, “A quantitative and qualitative comparison between the Empirical Mode Decomposition and the Wavelet Analysis”, Avances en Control, Instrumentación Virtual, Sistemas Digitales, Arquitecturas de Computadoras, Robótica y Procesamiento de Señales Vol. 9, pp 381-390, September (2004).

[64] Mörchen, F., “Time series feature extraction for data mining using DWT and DFT”. Technical Report No. 33, Departement of Mathematics and Computer Science Philipps-University Marburg, 2003

[65] G. Rilling, P. Flandrin and P Gonçalvès, “On Empirical Mode Decomposition and its algorithms”. IEEE-EURASIP, Workshop on Nonlinear Signal and Image Processing. NSIP-03, Grado (I), June 2003

[66] K. Blakrishnan, V. Honovar, “Evolutionary Design of Neural Architectures – A preliminary Taxonomy and Guide to Literature”. CS TR # 95-01. Department of Computer Science. Iowa State University (1995).