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

PREDICCIÓN DE CAPTURA DE ANCHOVETA UTILIZANDO REDES WAVELET CON OPTIMIZACIÓN DE ENJAMBRE DE PARTÍCULAS

MANUEL ANDRÉS MENA PEÑA

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

DICIEMBRE 2008

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

PREDICCIÓN DE CAPTURA DE ANCHOVETA UTILIZANDO REDES WAVELET CON OPTIMIZACIÓN DE ENJAMBRE DE PARTÍCULAS

MANUEL ANDRÉS MENA PEÑA

Profesor Guía: **Nibaldo Rodríguez Agurto**

Profesor Co-referente: **Broderick Crawford Labrín**

Carrera: **Ingeniería Civil en Informática**

Diciembre 2008

A mis padres, que con su incansable esfuerzo del día a día me han permitido cumplir múltiples sueños y logros en la vida.

Índice

Glosario de Términos

Análisis Armónico:	Análisis basado en la descomposición de señales.
Estocástico:	Sistema que funciona, sobre todo, por el azar.
Fitness:	Calidad de la solución encontrada.
Nodos o neuronas:	Unidades de interconexión de una red neuronal.
Outlayer:	Parámetros fuera del rango esperado.
Perceptrón:	Modelo de red neuronal básico.
Suavizado:	Eliminación de ruido y de outlayers en los datos.
Redes Neuronales:	Sistemas computacionales que emulan el comportamiento colaborativos de los seres vivos o de las neuronas.
Wavelet:	Ondícula, ondeleta u ondita, representan una señal en términos de versiones trasladadas y dilatadas de una onda finita denominada Wavelet madre.

Lista de Abreviaturas o Siglas

ACO:	Optimización con colonia de hormigas - Ant Colony Optimization.
ADN:	Material genético biológico - Ácido desoxirribonucleico.
AG:	Algoritmo Genético.
AIC:	Criterio de Akaike – An Information Criterion.
ART:	Adaptative Resonance Theory, teoría de resonancia adaptativa.
CWT:	Transformada Continua de Wavelet - Continuous Wavelet Transform.
DPWT:	Parámetro Discreto de Transformada Wavelet - Discret Parameter Wavelet Transform.
DTF:	Transformada Discreta de Fourier – Discret Transform Fourier.
DTWT:	Transformada Discreta en el Tiempo de las Wavelets - Discret Time Wavelet Transform.
DWT:	Transformada Discreta de Wavelet - Discret Wavelet Transform.
FFT:	Transformada rápida de Fourier - Fast Fourier Transform.
MAE:	Error medio absoluto - Mean Absolute Error.
PSO:	Optimización de enjambre de partículas - Particle Swarm Optimization.
PSO Max-Min:	Optimización de enjambre de partículas acotando la velocidad a un máximo y a un mínimo.
R2 :	Coeficiente de eficiencia o determinación.
RMSE:	Raíz del error cuadrático medio - Square Root of the Mean Square Error.
RNA:	Redes Neuronales Artificiales o en ingles como ANN – Artificial Neural Networks.
STFT:	Transformada localizada de Fourier - Short Time Fourier Transform.
Vmax:	Velocidad máxima.

Simbología

$\int$:	Integral
$\sum$:	Sumatoria
ψ :	Wavelet madre
∞ :	Infinito
< :	Menor que
$\sqrt{}$:	Raíz cuadrada
a :	Dilatación
τ :	Traslación
Δt :	Variación de tiempo
$\lvert e \rvert$:	Valor absoluto del error encontrado
$\overline{R}$:	Valor medio
$log( )$:	Logaritmo
$\tilde{x}(t)$:	Promedio móvil

Índice de Ilustraciones

Lista de Tablas

Resumen

El desarrollo sustentable de la economía del país a través de la correcta explotación de los recursos naturales es un tema de gran importancia en la actualidad y más aún con la extensa costa que posee Chile con la cual se puede sacar un gran provecho de los recursos pesqueros. Ahora bien teniendo claro nuestro escenario, la optimización en la explotación de los recursos pesqueros y no desgastarlos antes de tiempo, en especial de la Anchoveta, es el foco al cual se debe apuntar hoy en día y para esto se utilizan distintos modelos predictivos que nos ayuden a tomar correctas decisiones al momento de distribuir nuestros recursos y esfuerzos.

Para obtener un modelo predictivo más preciso se utilizó en la investigación las redes neuronales wavelets en combinación con la optimización de enjambre de partículas, ambas técnicas muy utilizadas en la actualidad, que intentan simular y modelar una realidad en particular.

El modelo propuesto para el pronóstico de los datos no suavizados obtuvo un coeficiente de determinación de un 71% utilizando 5 neuronas de entrada y 10 neuronas en la capa oculta, los pesos sinápticos fueron estimados usando el algoritmo de optimización de partículas en su versión Max-Min. Posteriormente la data de captura de Anchovetas fue suavizada utilizando la técnica del promedio móvil y es así como el modelo de pronóstico logró explicar un 76% de la variabilidad total de las capturas de anchovetas mensuales del norte de Chile.

Palabras claves: Redes Neuronales Wavelet, Inteligencia de Enjambre, PSO.

Capítulo 1
Introducción

La utilización de redes neuronales como una herramienta de apoyo a la resolución de problemas cotidianos ha evolucionado en el transcurso de los últimos años, lo cual ha llevado a la utilización de estas en distintas áreas del conocimiento y del quehacer humano. El gran avance que han tenido las redes neuronales se debe, más que nada, al desarrollo y avance paralelo de distintas técnicas y disciplinas que se han ido complementando y potenciando unas con otras, obteniendo así mejores y mayores resultados de los encontrados anteriormente. Es así como existen [1] importantes avances en las redes neuronales, como lo son: el manejo de la data utilizada, la configuración optimizada de los parámetros de red, el descubrimiento y uso de nuevos algoritmos tanto genéticos como aquellos algoritmos evolutivos derivados de la inteligencia artificial que mejoran el entrenamiento y configuración de las redes neuronales.

Con la constante evolución de las distintas disciplinas, se han logrado realizar variadas investigaciones que combinan los actuales conocimientos, con lo cual ha sido posible contrastar las ventajas de los avances encontrados, ya que con la correcta combinación de estos se pretende obtener, en nuestro caso, modelos de pronósticos más precisos y potentes de los que actualmente se poseen, y así poder encontrar solución a problemas de predicción y pronóstico de un escenario o estado inmediato aplicando las mejoras que se encuentren.

La teoría aplicada en distintas disciplinas nos dice que conociendo los antecedentes históricos de un problema en cuestión, se podría determinar con cierta exactitud un estado futuro que nos ayude a tomar decisiones que favorezcan nuestros fines, ya que se espera que con este conocimiento anterior se puede esperar un comportamiento similar de los datos y poder obtener un pronóstico, ahora bien, el problema radica en encontrar el modelo interno que posee el problema en cuestión, el cual nos llevará a una solución según las entradas recibidas.

Una herramienta ampliamente utilizada en la actualidad y que ha servido de gran ayuda en materia de modelos de pronósticos son las Redes Neuronales Artificiales (RNA) que han incorporado en su implementación la inteligencia artificial para optimizar su configuración y entrenamiento con lo cual han logrado, según estudios recientes [1], una mayor utilidad, potencia y precisión en los modelos de predicción obtenidos. Es así como en el tema de investigación se incluye la utilización de las redes neuronales para obtener los modelos de pronósticos y además, por otra parte, se incorporan las propiedades de la función Wavelet para la implementación de la red con lo cual se pretende obtener un modelo de pronósticos más refinado en la manipulación de la data y en encontrar solución al problema de predicción.

Se dice [1] que la clave del éxito en la obtención de modelos de pronósticos en redes neuronales recae en la precisión que se tenga en la estimación de los pesos en las relaciones de los distintos componentes de la red neuronal, los cuales deben representar el modelo interno que posee el problema y hacer posible así la obtención de la solución deseada. Para lograr este fin se pretende estudiar el éxito que tendrá implementar un algoritmo evolutivo muy utilizado en la actualidad como lo es la Optimización de Enjambre de Partículas y si es posible, con esta metodología propuesta, obtener una correcta y mayor aproximación estimada de los pesos de la red.

Cabe destacar y resaltar que los modelos de pronósticos pueden ser utilizados en diferentes ámbitos y áreas de estudio en los cuales se desea conocer y anticipar un escenario o estado para el cual se quiere estar preparado y poder así tomar decisiones anticipadas o preventivas que mejoran, de cierta manera, nuestra condición al momento de afrontar la situación cuando realmente ocurre. Para efectos de esta investigación se plantea la problemática de probar la eficiencia de una red neuronal en la predicción de recursos pesqueros pelágicos, en especial de los volúmenes de producción con que se puede contar mensualmente de Anchovetas en la zona norte de Chile. La dificultad de este tipo de problema radica en llevar a cabo de forma precisa y efectiva la predicción de los volúmenes de producción que se dispondrá en el siguiente período, para que así las distintas pesqueras y el Gobierno, puedan planificar y tomar decisiones sobre el recurso pesquero que es crítico al momento de proyectarse en el futuro y obtener un bienestar económico. Es así que para lograr este tipo de predicción de los volúmenes de captura, se debe contar con información histórica del comportamiento del recurso que se ha de estudiar, ya que se supone que ciertos patrones de la información disponible se repetirán en el futuro cercano respondiendo a un modelo interno del recurso.

La ventaja fundamental de este tipo de modelos de pronósticos frente a técnicas de estimación clásica radica en que estos modelos permiten la modelación de sistemas no lineales y no equilibrados, mediante la transformación no lineal de los datos de entrada y salida, lo que supone importantes ventajas frente a metodologías estadísticas convencionales, tales como las regresiones lineales [2], en las que la relación de los datos ha de ser lineal para ser estimados y responden a datos más bien homogéneos para obtener con éxito la solución.

La problemática estudiada en la cual se utilizarán estas técnicas y metodologías es un problema de predicción de abundancia de un determinado recurso natural, el cual es un tema muy en boga por la importancia de resguardar los recursos y llevar a cabo un desarrollo sustentable de un país. Chile por su larga costa posee innumerables recursos pesqueros y marítimos, los cuales muchas veces son mal utilizados y explotados de forma indiscriminada, por lo cual la importancia de obtener un modelo de pronósticos e implementarlo para predecir los volúmenes de recolección pesquera, en especial de Anchovetas, recae en el impacto económico que tiene la mala administración de estos recursos sobre la economía nacional, ya que el abuso en la explotación sin previo conocimiento del escenario con que se ve enfrentado podría agotar fácilmente los recursos pesqueros, poniéndolos en grave riesgo de extinción o veda, disminuyendo drásticamente las posibilidades de ser explotados nuevamente. Por otra parte las pesqueras involucradas al poder contar con un correcto modelo de pronóstico podrían planificar de mejor manera sus

recursos y así concentrar sus esfuerzos adecuadamente para obtener mejores ganancias y no obtener pérdidas de importancia al momento de decidir si fortalecer otras recolecciones o adoptar planes a futuro que mejoren las ganancias frente a un escenario pronosticado favorable o desfavorablemente para sus intereses.

1.1 Definición de Objetivos

A continuación se detallan el objetivo general del tema de investigación y de los objetivos específicos que se deben realizar para conseguir los resultados esperados con el estudio del problema en cuestión.

1.1.1 Objetivo General

Desarrollar un modelo de pronóstico de captura de volúmenes mensuales de Anchovetas de la zona norte chilena utilizando Redes Neuronales Wavelet con el algoritmo de entrenamiento de Optimización de Enjambre de Partículas.

1.1.2 Objetivos Específicos

- Comprender el funcionamiento de las arquitecturas de las Redes Neuronales Wavelet.
- Diseñar e implementar modelos de pronósticos utilizando los algoritmos de Optimización de Enjambre de Partículas: Clásico y Max-Min.
- Evaluar y contrastar el rendimiento de los predictores propuestos.

1.2 Metodología de Trabajo

Para llevar a cabo de forma correcta este tema de investigación es necesario, en primera instancia, dar una revisión a la información actual que se tiene de las distintas materias involucradas en el tema, como los son: las redes neuronales artificiales (RNA), el algoritmo de entrenamiento a utilizar y las evoluciones que han surgido en este sentido. Además será necesario investigar las ventajas que presenta implementar una red Wavelet en comparación a otros tipos de red y el funcionamiento que esta tiene, para poder elegir los parámetros adecuados en su implementación.

Teniendo claros los objetivos del proyecto y luego de haber reunido la información, se llevarán a cabo las tareas necesarias para obtener el resultado esperado, comenzando con el estudio del algoritmo de enjambre de partículas, luego se creará la arquitectura de la red neuronal Wavelet, posteriormente con la red obtenida se pasará a una etapa de entrenamiento con la data de prueba, para así finalmente obtener modelos de pronósticos, los cuales serán contrastados y validados al finalizar el proyecto según las variables consideradas y utilizadas en cada modelo, todo esto según los parámetros elegidos para su validación.

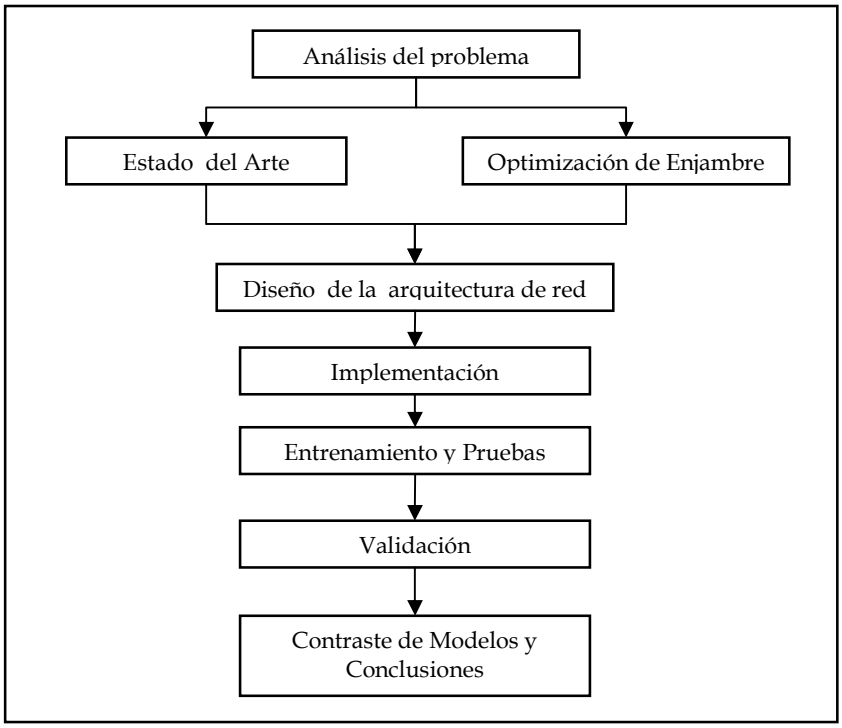

Ilustración 1.1: Esquema de trabajo.

En la Ilustración 1.1 queda esquematizado el alcance de la investigación, a rasgos generales, los pasos seguidos en la investigación para obtener los resultados esperados para ser evaluados y así obtener las mejores conclusiones al finalizar la investigación.

1.3 Organización del Texto

El documento se encuentra dividido en capítulos que muestran las distintas etapas de la investigación y a su vez estos capítulos se dividen en subsecciones que representan los distintos alcances en la etapa de investigación.

En el presente capítulo (Capítulo 1) se presentan de forma general los alcances de la investigación, sus objetivos y la metodología a emplear. En el Capítulo 2 se dará una revisión del estado del arte de las redes neuronales, su definición, clasificación y aplicación, para luego presentar las propiedades de la función Wavelet, rescatando las ventajas que esta presenta frente a otras técnicas clásicas.

Por otra parte, en el Capítulo 3 del documento se explicará completamente y teóricamente el algoritmo elegido de Optimización de Enjambre de Partículas - PSO y de las distintas variantes que se utilizarán.

Posteriormente, en el Capítulo 4, se presentará en detalle la propuesta a evaluar con la investigación y los recursos que se utilizarán, en donde se propone dejar explicada la real aplicación del algoritmo de Enjambre de Partículas en las redes neuronales Wavelet, tanto en su implementación y de cómo será usado por la red en la arquitectura de red propuesta.

Al inicio del Capítulo 5 se explicarán las métricas de validación de la investigación, definiendo los estimadores necesarios en el análisis final de la simulación y discusión de los resultados obtenidos. Es en este mismo capítulo en donde se presentan los modelos obtenidos y se elige, con los parámetros antes descritos, el mejor modelo.

Finalmente en el Capítulo 6 se entregan las conclusiones obtenidas con la investigación, la solución propuesta y futuras mejoras al modelo obtenido.

Capítulo 2
Estado del Arte

2.1 Antecedentes del Problema

Chile es considerado uno de los principales países pesqueros a nivel mundial, con aproximadamente 5 millones de toneladas de capturas anuales [3]. Las especies pelágicas constituyen cerca de un 90% de las capturas anuales. Este tipo de pesca se realiza desde la década del 50, primero sobre la Anchoveta, luego esta pesca colapsa a mediados de los 70's, siendo reemplazada por la Sardina y que después de 1975 también es desplazada de cierta manera por las capturas de Jurel y Caballa [4].

La información disponible sobre variables y parámetros que describen los procesos ambientales, dan cuenta de importantes fluctuaciones en los volúmenes de las especies pelágicas. Es así como se han detectado ciertas variables que influyen directamente en la distribución de los recursos en relación al ambiente, determinándose algunas relaciones funcionales para la Anchoveta como: la alternación de los vientos cálidos y fríos que hacen posible la retención de esta especie en la zona norte del país [5], así como también se ha probado que la concentración en esta zona del país se ve beneficiada por la presencia de remolinos anticiclónicos en la zona norte y por la presencia del río Loa, sin verse perjudicada esta concentración de mayor manera por el Niño y las corrientes ascendentes de la zona [6]. Se ha demostrado, también, que estos volúmenes de Anchoveta tienen estrecha relación con la temperatura de las aguas y las fluctuaciones cálidas y frías durante 30 años [7] y que las concentraciones de esta especie, en particular, que se ve más frecuentemente sobre aguas frías influenciadas fuertemente por la corriente de Humboldt proveniente desde el Perú [8]. Sin embargo, la creación de un modelo completo que abarque la mayoría de estas variables ambientales, involucra tiempo y estudio de un modelo complejo que deba probar la influencia real de cada una de estas variables, asignándole un peso o ponderación real de influencia sobre el valor estudiado. Es por esto, que para razones de esta investigación se creará un modelo univariado centrado en los volúmenes de captura de Anchoveta de los meses históricamente registrados, lo cual se basa en el supuesto de que estos volúmenes obtenidos ya se encuentran influenciados por las distintas variables oceanográficas y ambientales que en el modelo se verán reflejadas solo como el resultado en toneladas de captura de Anchoveta y no como una suma de variables ponderadas.

Como data de prueba se contará con una base de datos histórica de los volúmenes de Anchoveta recogidos mensualmente por distintas pesqueras de la zona norte de Chile desde aproximadamente el año 1963 al año 2002, que han sido recopilados por distintos organismos ambientales y gubernamentales para su estudio, y que nos dan cuenta de las fluctuaciones de los volúmenes de Anchoveta a través de los meses (Ilustración 2.1).

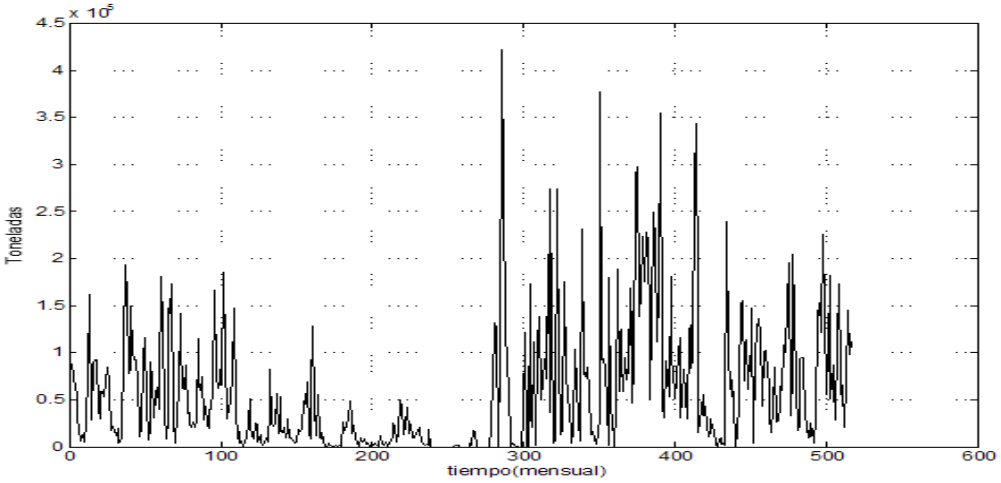

Ilustración 2.1: Gráfico de toneladas de Anchovetas recogidas por mes.

El estudio de estas fluctuaciones y la posible predicción de los volúmenes de Anchoveta para próximos meses se ha realizado utilizando distintas técnicas en los últimos años, incorporando en cada estudio las últimas técnicas y tendencias en algoritmos que puedan arrojar mejores resultados predictivos, intentando captar el modelo interno del problema en cuestión. A continuación se dará una revisión del estado actual de las redes neuronales, que en combinación con otras técnicas, como Wavelets, se intentará obtener un modelo calibrado que arroje una solución predictiva de los volúmenes de Anchoveta.

2.2 Redes Neuronales

La creación de las redes neuronales como sistemas colaborativos ha presentado múltiples soluciones a problemas antes poco abordables, presentando mayores ventajas y capacidades que un sistema lineal, el cual se ve limitado por su capacidad más rápidamente. Es así que aplicando el concepto de red o malla se crearon los sistemas no lineales llamados redes neuronales o redes de neuronas o para ser más específicos en la aplicación que se les dará, serán denominadas redes neuronales artificiales, denotadas como RNA (o en ingles como ANN – Artificial Neural Networks).

En los campos de la inteligencia artificial las RNA se presentan como un paradigma de aprendizaje y procesamiento de información autónomo que intenta simular el comportamiento del sistema nervioso de los seres vivos. En otras palabras, se intenta crear un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida, frente a uno o varios estímulos de entrada definidos y entrenados para responder frente a patrones determinados.

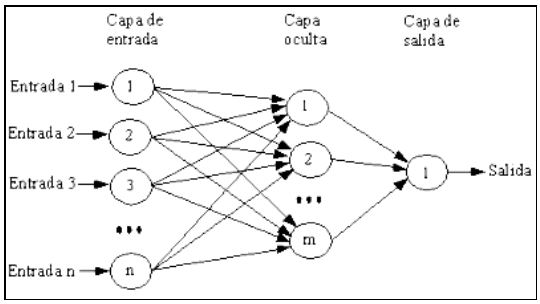

Ilustración 2.2: Red neuronal con *n* neuronas de entrada, *m* neuronas en su capa oculta y una neurona de salida.

Las Redes Neuronales fueron diseñadas originalmente como simulación abstracta de los sistemas nerviosos biológicos, formados por un conjunto de unidades llamadas "neuronas" o "nodos" conectadas unas con otras. Estas conexiones tienen una gran semejanza con las dendritas y los axones en los sistemas nerviosos biológicos.

El primer modelo de red neuronal fue propuesto en 1943 por McCulloch y Pitts [9] que seguía los términos de la actividad nerviosa para crear un modelo computacional. El modelo de McCulloch-Pitts es un modelo binario, y cada neurona tiene un escalón o umbral prefijado. Este primer modelo sirvió de ejemplo para los modelos posteriores de John Von Neumann [10], Marvin Minsky [11], Frank Rosenblatt [12], y muchos otros [13].

El funcionamiento de una red neuronal se explica a través de modelos matemáticos que intentan simular el comportamiento de las neuronas creando unidades llamadas nodos o neuronas propiamente tal, las cuales están interrelacionadas mediante mecanismos artificiales como redes de computadores, circuitos o simplemente lógica interna de programación, las cuales tienes cierto peso o grado de influencia en el enlace, con lo cual se intenta conseguir una respuesta frente a los estímulos que se ingresan, pasando a través de las distintas neuronas o capas de neuronas (Ilustración 2.2) hasta entregar una repuesta que representa la sumatoria de la colaboración de inteligencia de todas las neuronas.

Por lo general la salida que entrega una neurona viene dada fundamentalmente por tres funciones [14]:

1. **Función de Propagación** (también conocida como función de excitación), que por lo general consiste en la sumatoria de cada entrada multiplicada por el peso de su interconexión. Si el peso es positivo, la conexión se denomina excitatoria y si es negativo, se denomina inhibitoria.

2. **Función de Activación**, que modifica el resultado de la anterior. Puede no existir, siendo en este caso la misma salida de la función de propagación.

3. **Función de Transferencia**, que se aplica al valor devuelto por la función de activación. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que se quiera dar a dichas salidas. Algunas de las más

utilizadas son la función sigmoide (para obtener valores en el intervalo [0,1]) y la tangente hiperbólica para obtener valores en el intervalo [-1,1]).

Por lo general, existen dos fases en la creación y calibración de toda red neuronal o en su defecto, simplemente una etapa que mezcla ambas fases al mismo tiempo y que son: la fase de aprendizaje o entrenamiento y la fase de prueba.

1. **Fase de Aprendizaje**: La principal característica de las redes neuronales es la de aprender. Aprenden por la actualización o cambio de los pesos sinápticos que caracterizan a las conexiones. Se utiliza un conjunto de datos o patrones de entrenamiento para determinar los pesos de las conexiones del modelo neuronal, los pesos son adaptados de acuerdo a la información extraída de los patrones de entrenamiento nuevos que se van presentando. Normalmente, los pesos óptimos se obtienen optimizando (minimizando o maximizando) alguna función.

2. **Fase de Prueba**: Una vez entrenado el modelo, se usa la llamada fase de prueba, en la que se procesan los patrones de prueba que constituyen la entrada habitual de la red, analizando de esta manera la correcta funcionalidad de la red. Los resultados pueden ser tanto calculados de una vez como adaptados iterativamente, según el tipo de red neuronal, y en función de las pruebas.

2.2.1 Ventajas de las Redes Neuronales

Las redes neuronales presentan muchas ventajas debido a que están basadas en la estructura colaborativa e interconectada del sistema nervioso, con lo cual frente a un sistema común y corriente presentan grandes diferencias y propiedades que las distinguen:

- **Aprendizaje**: tienen la capacidad de aprender mediante la etapa aprendizaje, en donde se le entregan determinados datos de entrada y se le indica los datos esperados de salida. Luego con este aprendizaje se espera que la red se comporte de forma similar frente a un problema similar.

- **Auto organización y adaptividad**: crea su propia representación de la información, obteniendo las relaciones e interconexiones que convengan, mediante algoritmos de aprendizaje que adaptan y auto organizan las relaciones y pesos en la red.

- **Tolerancia a fallos**: se almacena la información de forma redundante, en el sentido de que esta red se encuentra completamente interconectada y puede seguir respondiendo aceptablemente aún si se daña parcialmente alguna neurona.

- **Flexibilidad**: puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada y a su vez puede llevar a cabo procesamiento de problemas no lineales con altas capacidades de aproximación, clasificación y solución.

- **Tiempo real**: debido a que puede implementarse paralelamente en un gran número de neuronas con un alto grado de interconectividad, se pueden obtener respuestas en tiempo real.

Estas y otras características dan a las redes neuronales un sin fin de aplicaciones cotidianas y especificas, ya que al poseer, por ejemplo, la capacidad de aprendizaje presentan mayor facilidad de resolver problemas que necesitan un conocimiento previo de la situación problema a resolver.

2.2.2 Familias de Redes Neuronales

Existen muchas formas de clasificar las redes neuronales, en donde la literatura y los distintos estudiosos del tema las distinguen según sus distintas propiedades y en la forma de implementarlas, entre las cuales se distinguen los siguientes tipos de clasificación:

- **Según el modelo que siguen:** existe una serie de modelos de redes neuronales que aparecen en la mayoría de estudios académicos y la bibliografía especializada (tabla 2.1), que van aumentando la complejidad y proponen mejoras a los modelos que los preceden. Estos modelos por lo general fueros propuestos inicialmente por un determinado autor para resolver problemas específicos, pero se han generalizado al ser capaces de resolver problemas de distinta naturaleza.

Tabla 2.1: Modelos teóricos de redes neuronales

Tipo	Descripción
Perceptrón	Uno de los primeros modelos utilizados en los que la entrada y salida pueden tomar solo valores binarios o bipolares.
Adaline	A diferencia del anterior modelo, esta trabaja con patrones de entrada y salida reales.
Perceptrón Multicapa	Posee una serie de capas de perceptrones en su implementación, entre las que se encuentran la capa de entrada, oculta y de salida.
Memorias Asociativas	Permiten recuperar información a partir de conocimiento parcial de su contenido, sin saber su localización de almacenamiento en la memoria.
Máquina de Boltzmann	Usa estados binarios, conexiones bidireccionales, transiciones probabilísticas y puede tener unidades ocultas.
Máquina de Cauchy	Similar a la anterior excepto en lo que concierne a la función probabilidad y a una función de temperatura que establece el plan de templado o enfriamiento de la red.
Redes de Elman	Poseen dos capas, cada una compuesta de una red tipo Backpropagation (propagación hacia atrás), con la adición de una conexión de realimentación desde la salida de la capa oculta hacia la entrada de la misma capa oculta.
Redes de Hopfield	Es una red monocapa autoasociativa no lineal.
Red de Contrapropagación	Crea una serie de subredes mediante memorias hetereoasociativas.
Redes de Neuronas de Base Radial	Ocupan funciones cuya salida depende de la distancia a un punto denominado centro.

Tipo	Descripción
Redes de Neuronas de Aprendizaje Competitivo	Está formada por una red excitadora hacia adelante y una red inhibidora lateral. Esta red selecciona un ganador, normalmente por medio de un método de aprendizaje competitivo.
Mapas Autoorganizados	También llamados redes de Coñeen**,** son un tipo de red neuronal no supervisada, competitiva, distribuida de forma regular en una rejilla de dos dimensiones, cuyo fin es descubrir la estructura subyacente de los datos introducidos en ella.
Crecimiento Dinámico de Células	Es una red que va aumentando su tamaño dependiendo de las necesidades.
Gas Neuronal Creciente	Es una estructura auto organizativa, con aprendizaje no supervisado, en donde los fundamentos para preservar la topología se basan en la obtención de la triangulación inducida de Delaunay
Redes ART (Adaptative Resonance Theory)	Solucionan el dilema de la estabilidad y plasticidad del aprendizaje mediante un mecanismo de retroalimentación entre las neuronas competitivas de la capa de salida.

- **Según su topología:** en función del patrón de interconexiones que presentan sus nodos internos y el flujo en que los datos o estimulaciones nerviosas fluyen a través de la red se han definido tres tipos básicos de redes (tabla 2.2), y que por lo general mantienen esta estructura solo variando el número de nodos o de capas.

Tabla 2.2: Topologías de redes neuronales

Tipo	Descripción
Redes de Propagación Hacia Delante (Feedfordward) o Acíclicas	Todas las señales van desde la capa de entrada hacia la salida sin existir ciclos, ni conexiones entre neuronas de la misma capa. Las cuales pueden ser Monocapa (por ej.: perceptrón y Adaline) o Multicapa (por ej.: perceptrón multicapa).
Redes Recurrentes	Presentan al menos un ciclo cerrado de activación neuronal internamente (por ej.: Elman, Hopfield y máquina de Bolzman).

- **Según el tipo de Aprendizaje:** se pueden clasificar en función del tipo de aprendizaje que se puede llevar a cabo según la definición de operación interna de la red. Para cada tipo de aprendizaje se encuentran varios modelos propuestos (tablas 2.3) y que son dependientes de la data de entrenamiento y de cómo esta es utilizada en el entrenamiento.

Tabla 2.3: Tipos de aprendizaje neuronal

Tipo	Descripción
Aprendizaje Supervisado	Necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta se conoce.
Aprendizaje no Supervisado o Autoorganizado	No necesitan de un conjunto previo de datos.
Redes Híbridas	Es un enfoque mixto en el que se utiliza una función de mejora para facilitar la convergencia.
Aprendizaje Reforzado	Se sitúa a medio camino entre el supervisado y el autoorganizado.

- **Según el tipo de entrada:** esta clasificación se describe según el tipo de entrada que sea capaz de procesar la red y de la información que estas entregan (tabla 2.4). El tipo de datos define las distintas funciones y transformaciones que sufrirán estos datos para entregar un estímulo de salida.

Tabla 2.4: Tipos de redes según la entrada.

Tipo	Descripción
Redes Analógicas	Procesan datos de entrada con valores continuos y, habitualmente, acotados.
Redes Discretas	Procesan datos de entrada de naturaleza discreta; habitualmente valores lógicos y booleanos.

2.2.3 Aplicaciones de las Redes Neuronales

Las redes neuronales poseen propiedades que las hacen apropiadas para resolver problemas que presentan un modelo representativo programable para encontrar su solución, ya que se cuentan con la posibilidad de contar con un conjunto de ejemplos básicos de entrada para su entrenamiento y ajuste, para así luego resolver problemas tanto o más complejos que los representados. Son altamente robustas tanto al ruido como a la distorsión de los elementos analizados y son fácilmente paralelizables.

Entre las aplicaciones más comunes destacan los problemas de clasificación y reconocimiento de patrones de voz, imágenes y señales [15]. También se han utilizado para hacer predicciones de todo tipo como estimaciones financieras y de tiempo atmosférico. Se han utilizado también con éxito en problemas en los que no existen modelos matemáticos precisos o algoritmos con complejidad razonable, como por ejemplo en el problema del vendedor viajero, que no tiene solución exacta pero que con las redes neuronales entrega buenas aproximaciones.

En el último tiempo las redes neuronales se han aplicado en conjunto con los algoritmos genéticos (AG) y evolutivos para crear mejores modelos. En este tipo de aplicaciones el algoritmo ayuda a generar y elegir los parámetros de la red (topología, aprendizaje, función de activación, entre otros parámetros) y el mejoramiento de la red viene dado por el resultado del comportamiento exhibido de la red, la cual se va calibrando según los resultados esperados.

La red neuronal utilizada en el presente trabajo de predicción de captura de Anchoveta puede clasificarse como una red neuronal multicapa acíclica, que trabaja con valores analógicos y es entrenada con un algoritmo de aprendizaje híbrido, que será detallado en un capítulo aparte ya que pertenece a una nueva tendencia y área de investigación en desarrollo como lo son los algoritmos genéticos y evolutivos.

Como se ha dicho en el actual capítulo, las redes neuronales están compuestas internamente de funciones matemáticas que las caracterizan unas de otras, es así como se han aprovechado las propiedades de distintas funciones matemáticas para lograr un modelado interno más exacto al diseñar las redes neuronales, y es por esto que se nos presenta la posibilidad de utilizar las propiedades de la función Wavelet para crear nuestra red.

2.3 Redes Wavelet

Las Wavelets (ondículas, ondeletas u onditas) representan una señal en términos de versiones trasladadas y dilatadas de una onda finita denominada Wavelet madre.

En términos históricos, el desarrollo de las Wavelets reúne varias líneas de pensamiento, todo comienza a partir del trabajo de Alfred Haar a principios del siglo XX [16] y contribuyeron de modo notable al avance de la teoría Wavelet autores como Goupillaud, Grosman y Morlet con su formulación de lo que hoy conocemos como transformada Wavelet continua, Jan Olov-Strömberg con su temprano trabajo sobre Wavelets discretas (1983), Ingrid Daubechies, con su propuesta de Wavelets ortogonales con soporte compacto (1988), Stephane Mallat y Yves Meyer, con su marco multiresolución (1989), Delrat con su interpretación de la transformada Wavelet en tiempo-frecuencia (1991), Newland, con su transformada Wavelet armónica, y muchos otros desde entonces que han complementado el estudio de la teoría Wavelet.

La teoría Wavelets está relacionada con muy variados campos y áreas de estudio en donde la señal puede ser representada de manera trasladada y dilatada. Todas las transformaciones Wavelet pueden ser consideradas formas de representación en tiempo-frecuencia y, por tanto, están relacionadas con el análisis armónico de la señal. Las transformadas Wavelets son un caso particular de filtro de respuesta finita al impulso. Las Wavelets, continuas o discretas, como cualquier función L^2, responden al principio de incertidumbre de Hilbert (conocido por los físicos como principio de incertidumbre de Heisenberg), el cual establece que el producto de las dispersiones obtenidas en el espacio directo y en el de las frecuencias no puede ser más pequeña que una cierta constante geométrica.

2.3.1 Transformada de Wavelet

En los años 80’s se introduce el estudio de señales mediante transformada Wavelet. Esta transformada permite realizar un análisis de la señal tanto en el dominio frecuencial como en el temporal, generalizando el concepto de transformada de Fourier. Desde sus comienzos hasta el momento actual esta herramienta se ha estudiado y desarrollado en profundidad, y está resultando de sumo interés en diversos problemas [17].

Las Wavelets son funciones matemáticas que se pueden utilizar para filtrar series temporales de datos y analizar la variación de su contenido espectral, ofreciendo una

representación tiempo-frecuencia más precisa para las señales no estacionarias que el análisis tradicional de Fourier. Representando la evolución temporal del espectro de la señal se puede localizar en el tiempo la ocurrencia de discontinuidades, impulsos y variaciones que escapan a los métodos habituales de análisis.

La teoría de Wavelets se explica análogamente al análisis tradicional de señales: una función periódica puede representarse como una suma infinita de otras funciones, por ejemplo una base de exponenciales complejas es en el caso de las series de Fourier. La transformada de Fourier permite analizar señales periódicas y es reversible, es decir, permite pasar de un dominio a otro (tiempo y frecuencia en este caso), pero no conserva la información relativa al tiempo estando en el dominio de la frecuencia, y viceversa. La transformada de Wavelet también es reversible, con la diferencia que cualquier base de funciones Wavelets es de soporte compacto, lo que permite localizarla simultáneamente en tiempo y frecuencia, obteniendo la señal original al invertirla.

Los algoritmos de transformada rápida de Fourier (Fast Fourier Transform - FFT) y de transformada localizada de Fourier (Short Time Fourier Transform - STFT) son la alternativa clásica cuando se busca la evolución temporal de un espectro de frecuencias. En estos métodos se aplican y solapan ventanas temporales con el objeto de localizar las frecuencias en el tiempo. El inconveniente es que la resolución está dada por el tamaño de la ventana, que es fijo. La STFT utiliza el algoritmo FFT en ventanas de longitud fija, pero pierde resolución temporal a frecuencias mayores de la longitud de la ventana. Con el solapamiento de ventanas se reduce este problema aunque no se resuelve por completo. Una posible solución sería utilizar ventanas de longitud variable en función de la frecuencia buscada, descartando la información que dan para frecuencias mayores. Esta operación es mucho más simple con un algoritmo basado en la transformada de Wavelet; no se calcula ningún promedio sino que se identifica la frecuencia de un determinado ciclo de la serie temporal y se da su posición.

Muchas señales son no estacionarias. La energía y/o el espectro de una señal pueden variar con el tiempo. Una caracterización completa de una señal no estacionaria en el dominio de la frecuencia debe incluir el aspecto del tiempo, resultando un análisis de la señal en el tiempo – frecuencia.

En el análisis de tiempo-frecuencia de una señal no estacionaria, existen dos requisitos contradictorios. El ancho de la ventana T debe ser bastante grande para dar resolución a la frecuencia deseada pero debe ser también suficientemente corta tal que no interfiera los eventos dependientes del tiempo. Como se describió anteriormente, una buena resolución en tiempo o en frecuencia implica una buena localización en tiempo o en frecuencia. Ventanas muy estrechas idealmente un impulso, da una resolución perfecta en el tiempo, pero una resolución pobre en frecuencia, porque tiene un ancho de banda infinito. Por otro lado, un mismo filtro ancho de banda angosta da una buena localización en frecuencia, pero pobre localización en el tiempo porque la respuesta del impulso no se deteriora rápidamente con el tiempo.

2.3.2 Fundamentos Teóricos de la Transformada Wavelet

La transformada de Wavelet, se comporta similar a la STFT, en función del tiempo dentro de funciones de dos dimensiones *a* y τ. El parámetro *a* es denominado peso o escala, este peso usa una función para compresión o estrechamiento, y τ es la traslación de la función Wavelet a lo largo del eje de tiempo. La señal *s(t)* asume el cuadrado de la integral, denotada de la siguiente manera:

$$\int s^2(t)dt < \infty \qquad \textbf{(2.1)}$$

A partir de una Wavelet base o madre $\psi(t)$ se van obteniendo según *a* y τ las distintas Wavelet hijas. Teóricamente se distinguen en la literatura cuatro tipos básicos de transformadas Wavelet:

1. La Transformada Continua de Wavelet (**C**ontinuous **W**avelet **T**ransform)

$$CWT(a,\tau) = \frac{1}{\sqrt{a}}\int s(t)\psi\left(\frac{t-\tau}{a}\right)dt \qquad \textbf{(2.2)}$$

La CWT es considerada un paralelo a la Transformada de Fourier, en cuanto al análisis armónico de la señal.

2. El Parámetro Discreto de Transformada de Wavelet (**D**iscret **P**arameter **W**avelet **T**ransform)

$$DPWT(m,n) = a_o^{\frac{-m}{2}}\int s(t)\psi(a_o^{-m}t - n\tau_o)dt \qquad \textbf{(2.3)}$$

En donde los parámetros *a, τ* son discretos para $a = a_0^m$ y $\tau = n\tau_0 a_0^m$, con a_o, τ_o intervalos de muestreo y *m, n* enteros.

3. La Transformada Discreta en el Tiempo de las Wavelets (**D**iscret **T**ime **W**avelet **T**ransform)

$$DTWT(m,n) = a_o^{\frac{-m}{2}}\sum_k s(t)\psi(a_o^{-m}k - n\tau_o) \qquad \textbf{(2.4)}$$

En donde el tiempo es discreto, con *t=kT* y el intervalo de muestreo *T=1*. Similar a la Transformada Discreta de Fourier *DTF*.

4. La Transformada Discreta de Wavelet (**D**iscret **W**avelet **T**ransform)

$$DWT(m,n) = 2^{-\frac{m}{2}} \sum_{k} s(t)\psi(2^{-m}k - n) \qquad \textbf{(2.5)}$$

En donde *m, n* son enteros y *k* es discreto.

En general, dos de los cuatro tipos de transformadas Wavelets (ecuación 2.2, ecuación 2.3, ecuación 2.4 y ecuación 2.5) son las más comúnmente usadas. Por una parte la transformada discreta (ecuación 2.5) que actúa sobre la señal de forma semejante a un banco de filtros pasa-altos y pasa-bajos, separando las contribuciones de alta y baja frecuencia de la señal. Y por otro lado tenemos la transformada continua (ecuación 2.2) que actúa como un filtro pasa-banda, dejando sólo los componentes de frecuencias deseadas.

Algunos ejemplos de funciones Wavelets madres $\psi(t)$ comúnmente usadas son las siguientes:

- Modulación Gaussiana (Morlet)

$$\psi(t) = e^{jwt} e^{-\frac{t^2}{2}} \qquad \textbf{(2.6)}$$

- Segunda Derivada de una Gaussiana

$$\psi(t) = (1 - t^2) e^{-\frac{t^2}{2}} \qquad \textbf{(2.7)}$$

- Haar

$$\psi(t) = \begin{cases} 1 & 0 \le t \le 1/2 \\ -1 & 1/2 \le t \le 1 \\ 0 & otros \end{cases} \qquad \textbf{(2.8)}$$

- Shannon

$$\psi(t) = \begin{cases} 1 & \pi \le \omega \le 2\pi \\ 0 & otros \end{cases} \qquad \textbf{(2.9)}$$

2.3.3 Aplicaciones de las Wavelets

En cuanto a sus aplicaciones, la transformada Wavelet discreta se utiliza por lo general para la codificación de señales, mientras la transformada Wavelet continua se utiliza en el análisis de señales. Como consecuencia, la versión discreta de este tipo de transformada se utiliza fundamentalmente en ingeniería e informática, mientras que la continua se utiliza sobre todo en ciencias como la física.

Este tipo de transformadas están siendo cada vez más empleadas en un amplio campo de especialidades, a menudo sustituyendo a la transformada de Fourier por no manejar de buena forma la dupla tiempo-frecuencia, sin poder rescatar la señal original [18]. Se puede observar este desplazamiento en el paradigma en múltiples ramas de la física, como la dinámica molecular, la astrofísica, la geofísica de los sismos, la óptica, el estudio de las turbulencias y la mecánica cuántica, así como en otros campos muy variados como el procesamiento digital de imágenes, los análisis de sangre, el análisis de electrocardiogramas, el estudio del ADN, el análisis de proteínas, la meteorología, el procesamiento de señal en general, el reconocimiento de voz, los gráficos por computador, el análisis multifractal y en el campo de biometría.

Las propiedades de estas funciones Wavelets son las estudiadas para representar internamente el funcionamiento de la red neuronal, lo que se llamara más adelante red Wavelet, en donde la suma de las distintas Wavelet hijas dará como resultado la señal original y nos permitirá representar el modelo interno del problema de predicción. Ahora nuestra atención se centrará en como ajustar los valores de a de dilatación y τ de traslación al interior de la red, los cuales se verán ajustados con el algoritmo de entrenamiento elegido.

Capítulo 3
Optimización de Enjambre de Partículas

3.1 Descripción

La optimización de enjambre de partículas (Particle Swarm Optimization - PSO), es considerado dentro de un nuevo tipo de algoritmos que buscan resolver problemas emulando el movimiento de grandes comunidades de individuos, en donde cada individuo puede modificar su propia opinión basándose en tres factores: su conocimiento sobre el entorno (su valor de fitness), su conocimiento histórico o experiencias anteriores (su memoria) y el conocimiento histórico [19].

La optimización con enjambre de partículas, es una de las técnicas estocásticas de cálculo evolutivo. Tanto PSO como la optimización con colonia de hormigas (Ant Colony Optimization - ACO) son los dos métodos más populares y utilizados en el área de la inteligencia artificial. Ambos tratan de imitar los comportamientos sociales de un grupo colectivo a partir de la interacción de los individuos entre sí y con el entorno. Actualmente estos métodos de optimización heurísticos basados en poblaciones han despertado el interés de la comunidad científica debido a su capacidad para explorar eficientemente espacios de soluciones multimodales y multidimensionales.

Para entender el funcionamiento de PSO como algoritmo de optimización, se toma por lo general el siguiente ejemplo: el comportamiento que exhibe un enjambre de abejas en su movimiento sobre un campo cubierto con diferentes concentraciones de flores. Sin ningún conocimiento a priori del espacio de búsqueda, las abejas inician su movimiento desde posiciones aleatorias y con velocidades aleatorias. En su desplazamiento, el objetivo del enjambre se centra en encontrar el emplazamiento con la mayor densidad de flores. Cada abeja tiene memoria y puede recordar la posición visitada con mayor densidad de flores y también conoce, por mecanismo de comunicación con sus iguales, la localización donde otras abejas encontraron una densidad de flores significativa. Esta dupla de información es utilizada por la abeja para modificar continuamente su trayectoria, acelerando en ambas direcciones y volando hacia un punto espacial intermedio que dependerá de su posición actual y de cómo influyan sobre su decisión las así denominadas nostalgia o memoria y cooperación o conocimiento social. De esta forma, las abejas se encuentran permanentemente sobrevolando el campo en busca de posiciones con mayor densidad de flores, redirigiendo la trayectoria del enjambre cada vez que se encuentran configuraciones de mayor calidad. Con el transcurso del tiempo, una vez ha sido explorado

el espacio de soluciones en su totalidad, el conjunto del enjambre se encontrará volando alrededor de la zona la mayor concentración de flores de todo el campo. En esta situación, las abejas, incapaces de encontrar posiciones alternativas mejores, son permanentemente atraídas hacia dicha posición.

El comportamiento social que exhiben éste y otros organismo se puede entender como un método de optimización en el que el espacio de búsqueda se puede extender a las *N* dimensiones del problema a optimizar, y en donde cada partícula se identifica como una posible solución potencial al problema, caracterizada por un vector velocidad y un vector posición, ambas en *N* dimensiones. El problema, entonces, se reduce a establecer la ecuación que dicte como debe moverse cada partícula de la población en el espacio *N*-dimensional para mimetizar la inteligencia de estas comunidades y evitar a su vez caer en soluciones locales.

3.2 Reseña Histórica y Fundamentos

Los inicios del PSO como método de optimización se remontan a los estudios realizados por Kennedy y Eberhart [20] quienes se propusieron como objetivo: simular el movimiento sincronizado e impredecible de grupos tales como los bancos de peces o las bandadas de aves, intrigados por la capacidad de estos grupos para separarse, reagruparse y encontrar alimento. Paralelamente trabajos previos en el ámbito de la biología y de la sociología, concluyen que el comportamiento, inteligencia y movimiento de estas agrupaciones está relacionado directamente con la capacidad para compartir información y aprovecharse de la experiencia acumulada de sus iguales.

Kennedy y Eberhart introducen el término general de partícula o agente para representar a los peces, pájaros, abejas, hormigas o cualquier otro tipo de individuos que exhiban un comportamiento social como grupo, en forma de una colección o cúmulo de partículas que interactúan entre sí.

De acuerdo con los fundamentos teóricos de este método, el movimiento de cada una de las partículas hacia un objetivo común en dos dimensiones está condicionado por dos factores básicos: la memoria autobiográfica de la partícula o nostalgia y la influencia social de todo el enjambre. Como método de optimización, puede extenderse como un espacio *N*-dimensional de acuerdo con el problema bajo análisis. La posición instantánea de cada una de las partículas de la población en el espacio *N*-dimensional representa una solución potencial, siendo *N* el número de incógnitas del problema a resolver. Básicamente, el proceso evolutivo se reduce a mover cada partícula dentro del espacio de soluciones con una velocidad que variará de acuerdo a su velocidad actual, a la memoria de la partícula y a la información global que comparte el resto del enjambre, utilizando una función de "fitness" para cuantificar la calidad de la solución entregada por cada partícula en función de la posición que esta ocupe.

Más allá del alcance que se pueda dar al método, los esquemas existentes para su implementación son muy diversos. Dependiendo de cómo se actualicen las posiciones de las partículas surgen las versiones síncrona y asíncrona del algoritmo. Adicionalmente dependiendo de cómo se haga fluir la experiencia acumulada por el enjambre sobre el movimiento de cada una de las partículas que lo integran, se puede distinguir entre PSO local y global. La combinación de estas cuatro variantes resume los esquemas desarrollados e investigados comúnmente, y es así como en la literatura existen múltiples variantes a los esquemas convencionales, en la mayoría de los casos fruto de modificaciones introducidas por los propios autores para mejorar el rendimiento del algoritmo original en aplicaciones concretas.

3.3 Formulación del PSO

En la formulación de PSO se define la velocidad de partícula como el único operador disponible para controlar la evolución de la optimización. De forma genérica, supongamos una población de *i* partículas donde cada partícula del enjambre se identifica con dos variables de estado inicializadas de forma aleatoria dentro del espacio *N*-dimensional que establece el problema a optimizar: un vector de velocidad *Vi*=(*Vi1*,*Vi2*,...,*ViN*), y un vector de posición *Xi*=(*Xi1*,*Xi2*,...,*XiN*). Este último vector corresponde a una solución potencial al problema de optimización. Los límites de los parámetros a optimizar *Xn* perteneciente al intervalo *[Xn.min, Xn.max]* y que conforman en su conjunto el espacio de búsqueda al cual debe restringirse el movimiento del enjambre.

Adicionalmente, cada partícula *i* mantiene en memoria información de la posición espacial asociada con la mejor solución históricamente visitada por ésta *Pi*=(*Pi1*,*Pi2*,...,*PiN*), y también conoce la posición de la mejor partícula o solución encontrada por todos sus congéneres, *G=(g1*,*g2*,...,*gN)*. El movimiento del enjambre se realiza en pasos temporales, que se traducen a nivel de algoritmo en iteraciones contiguas del conocimiento acumulado.

En cada iteración *k*-ésima del método, cada una de las partículas de la población recorre el espacio de soluciones con una velocidad *Vi* hacia nuevas posiciones *Xi*, de acuerdo con su propia experiencia *Pi*, y con la experiencia aportada por el mejor de sus vecinos *G*. En las primeras versiones del algoritmo PSO, esta formulación se reduce a las ecuaciones:

$$Vin(k+1) = Vin(k) + c1 * r1(k) * \left[Pin(k) - Xin(k)\right] + c2 * r2(k) * \left[G(k) - Xin(k)\right] \qquad \textbf{(3.1)}$$

$$Xin(k+1) = Xin(k) + Vin(k+1) * \Delta t \qquad \textbf{(3.2)}$$

En la ecuación (3.1), *Vin*(*k*) y *Xin*(*k*) representan, respectivamente, la velocidad y posición en la iteración o instante de tiempo *k* de la partícula *i* en la dimensión *n*-ésima del espacio de búsqueda. Los factores c*1* y *c2* son las denominadas constantes de aceleración cognitiva y social, que determinan en qué medida influyen sobre el movimiento de la partícula su propia memoria y la cooperación entre los individuos, respectivamente. Los términos *r1*(*k*) y *r2*(*k*) son dos números aleatorios uniformemente distribuidos entre 0 y 1, U[0,1], cuyo objetivo es emular el comportamiento estocástico y un tanto impredecible que exhibe la población del enjambre. Después de calcular la nueva velocidad de la partícula *i* en la dimensión *n*, la nueva posición *Xin*(*k + 1*) se actualiza directamente de acuerdo con la ecuación (3.2), donde se asume que la velocidad se aplica durante un cierto período de tiempo Δt, típicamente de valor unitario. El proceso descrito se extiende al espacio *N*-dimensional, de forma que se van componiendo iterativamente nuevos vectores de posición *Xi*, utilizando, como en cualquier otro método de cómputo evolutivo, una función de fitness para ponderar la calidad de dicha solución parcial, actualizando los vectores *Pi* y *G* si se detectan resultados mejores.

El movimiento de los agentes sobre el espacio de soluciones y, en consecuencia, el rendimiento del algoritmo, está condicionado por el grado de contribución de las tres componentes de la velocidad, que son: la memoria, nostalgia o autoaprendizaje para incluir la experiencia de la propia partícula; la cooperación o conocimiento social de la información compartida; y por último el intercambio de información o comportamiento social como grupo.

A la formulación del PSO, antes detallada, se le llama PSO clásico o convencional. Existen muchas variantes que mejoran en algunos aspectos el PSO convencional, aportando mejores resultados de los ya encontrados. En nuestro caso estudiaremos una variante que acota la velocidad de la partícula, se especifica un valor máximo *vmax* que restringe la velocidad en cada dimensión al intervalo [-*vmax*, *vmax*]. Si el valor de *vmax* toma valores extremadamente pequeños las partículas explorarán el espacio de soluciones muy lentamente y podrán quedar atrapadas alrededor de soluciones locales, incapaces de librarse de la base de atracción. Sin embargo, si no se introdujese el parámetro de control *vmax*, el enjambre no convergería hacia un punto, sino que sufriría el fenómeno conocido como explosión del PSO, consistente en un comportamiento oscilatorio y creciente de la posición de las partículas, careciendo el método de potencial alguno como algoritmo de optimización.

El efecto de *vmax* fuertemente vinculado con la naturaleza del problema a optimizar, lo convierte en el parámetro principal a calibrar en PSO, sin ninguna regla aparente que dicte las pautas a seguir en su elección para asegurar la convergencia del enjambre.

3.4 Dominio de aplicación de PSO

Llevada a la práctica la filosofía de PSO al campo de las ciencias de la inteligencia artificial y del cómputo evolutivo, este algoritmo presenta una extremada simplicidad de programación y un número reducido de parámetros de los que depende el funcionamiento interno del algoritmo, lo que le convierten en una alternativa ventajosa al momento de ser seleccionado.

Ha sido aplicado con éxito al diseño de antenas y un sin número de componentes electromagnéticos [18], en la optimización de funciones y resolución de problemas matemáticos complejos (por ejemplo: el problema del vendedor viajero). Ha sido probado en el entrenamiento de redes neuronales, optimización de sistemas dinámicos, procesado de señal, gestión, planificación y optimización de recursos en redes de distribución de energía eléctrica, gestión de redes de sensores, planificación de red en servicios de telecomunicaciones, gestión empresarial y teoría de juegos, entre otros [21].

Capítulo 4
Implementación y Simulación

4.1 Descripción del Modelo Propuesto

Con la propuesta llevada a cabo se buscó obtener la implementación de una arquitectura de red neuronal que pudiese utilizar las propiedades de la función Wavelet en la configuración interna de las distintas capas de la red. Cabe destacar que una parte importante en la creación de la red neuronal es la elección de los parámetros que esta posee, para lo cual se utilizó la optimización de enjambre de partículas (PSO) para entrenar la red y obtener así estos valores, siendo implementadas dos variantes de PSO que posteriormente serán contrastadas entre sí, según los resultados obtenidos.

La implementación de los algoritmos de optimización y de la arquitectura de red Wavelet fueron desarrollados en lenguaje de programación C/C++ estructurado por ser de fácil manejo y entendimiento, con lo cual se ha desarrollado un programa que integra estos conceptos realizando a la vez en una misma interfaz las fases de: entrenamiento y prueba con las configuraciones probadas, guardando los resultados y salidas de la red en archivos con extensión .csv para ser visualizados en Excel y poder así graficar el comportamiento de la red.

El entrenamiento de las versiones configuradas de la red con los distintos algoritmos de optimización de enjambre de partículas utilizados en la investigación se realizaron con las 2/3 partes de los datos que se poseen de las capturas de volúmenes en toneladas de Anchovetas por mes, normalizados [0,1] para su mejor manejo, para obtener en esta fase una calibración deseada de los pesos de la red Wavelet que posteriormente en la fase de prueba comparará los datos obtenidos con la data esperada que corresponde al 1/3 de los datos anteriormente no utilizados en el entrenamiento. Finalmente con los modelos de pronósticos obtenidos se realizaron evaluaciones de los resultados arrojados al aplicar las dos variantes de PSO propuestas, los cuales serán validados según algunos criterios a considerar.

4.2 Arquitectura de Red Neuronal

La definición de la arquitectura es un punto importante en el diseño de una red neuronal, porque en ella se restringe el tipo de problema que puede ser tratado y los tipos de datos que serán utilizados, en ella se formalizan el número de capas que va a tener la red (de una a múltiples capas), las que pueden ser clasificadas en tres grupos: capa de entrada, capas intermediarias u ocultas y capas de salida.

Nuestra arquitectura de red va a estar compuesta de tres capas: una capa de entrada, una capa oculta y una capa de salida. La cada capa de entrada va a considerar un número entre dos y doce neuronas [2,12], puesto que se ha estimado en estudios anteriores [3] y [22] que este número es suficiente para comenzar a obtener un pronóstico eficiente con la información que se posee de los meses previos de captura de Anchoveta y que se utilizarán en la red para predecir el próximo mes. Por su parte la capa oculta, que caracteriza a la red Wavelet, va a estar implementada con un número de neuronas a estimar posteriormente en la fase de implementación y pruebas, el cual será obtenido de un rango de entre diez y veinte [10,20]. Finalmente la capa de salida poseerá una única neurona, la cual entregará el pronóstico estimado del próximo período.

La Ilustración 4.1 representa la arquitectura de red Wavelet utilizada en la investigación, la cual posee un modelo de tres capas en la configuración interna utilizada.

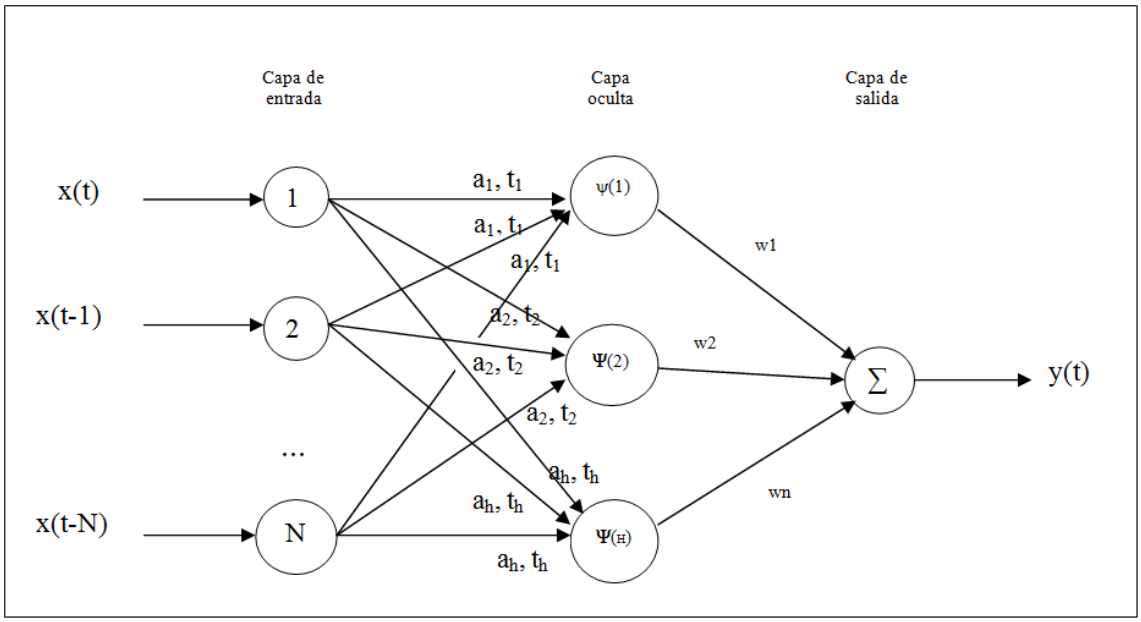

Ilustración 4.1: Arquitectura de la red Wavelet

Las entradas *x(t), x(t-1),* hasta *x(t-N)* de la Ilustración 4.1 corresponden a los datos de captura obtenidos los N meses anteriores al período que se quiere pronosticar *x(t+1)*, los cuales corresponden a los datos recibidos en la capa de entrada a la red de tamaño N. En la capa oculta se puede observar que existe H neuronas que participarán en el proceso, las cuales deben ser calibradas al momento de implementar la red y así poder determinar su número exacto. Es en esta capa oculta en donde los parámetro de la red wavelet se ven involucrados, puesto que cada neurona de la capa oculta responde a una función de activación que depende de los valores de dilatación a_i y traslación t_i de la función Wavelet descrita en la ecuación (4.1) para cada neurona *i* de la capa oculta que posee H neuronas.

$$\psi\left(\frac{x-t_i}{a_i}\right)\frac{1}{\sqrt{a_i}} \qquad \textbf{(4.1)}$$

Finalmente se puede apreciar en la Ilustración 4.1 la existencia de una única neurona en la capa de salida, la cual se ve relacionada con las distintas neuronas de la capa oculta según un peso w_i, las cuales según este peso tendrán un cierto grado de influencia sobre la salida *y(t)* pronosticada. La salida *y(t)*será el parámetro a depurar en la etapa de aprendizaje o entrenamiento de la red para que este sea lo más cercano posible al valor real esperado *x(t+1)*.

4.3 Aplicación de PSO en la Red Neuronal

En este punto se explicará la relación correcta de la arquitectura de red que se utilizará y el algoritmo que optimizará los parámetros utilizados, en nuestro caso este algoritmo de entrenamiento será PSO, el cual entrenará la red para conseguir los coeficientes de dilatación *a* y traslación *t* para cada neurona en el enlace de la entrada con las neuronas de la capa oculta y los pesos *w* de salida de la capa oculta.

Como se vio en el Capítulo 3 sección “3.3 Formulación del PSO”, se deben adecuar estos parámetros a la arquitectura de red. Se supondrá una población de *I* partículas (entre 20 y 50 inicialmente) donde cada partícula del enjambre se identifica con dos variables de estado inicializadas de forma aleatoria dentro del espacio N-dimensional a optimizar: un vector velocidad *Vi* = (*Vi1,Vi2,...,ViN*), y un vector de posición *Xi* = (*Xi1,Xi2,...,XiN*), este último corresponde a una solución potencial al problema de optimización, en donde *Xi* se compone de los parámetros *w*, *a* y *t* de las relaciones existentes entre las neurona de la red, es decir *Xi=[w , a , t]*, en donde *w* tiene entradas correspondientes a las H neuronas de la capa oculta (número aún no determinado), *a* y *t* poseen tantas entradas como relaciones existan entre la entrada N y la capa oculta H (NxH). Los límites de los parámetros a optimizar, *Xi* perteneciente al intervalo [0 , 1], conforman en su conjunto el espacio de búsqueda al cual debe restringirse el movimiento del enjambre. Como se ha de notar la

solución almacenada en *Xi*, de cada partícula, corresponde a una configuración completa de red y que cada partícula irá modificando en las iteraciones del algoritmo.

Adicionalmente, cada partícula *i* mantiene en memoria información de la posición espacial asociada con la mejor solución históricamente visitada por ésta *Pi*=(*Pi1,Pi2,...,PiK*) en el instante K, y también conoce la posición de la mejor partícula o mejor solución encontrada por sus congéneres, *G*=(*g1,g2,...,gK*), ambas informaciones estarán compuestas a su vez de *w*, *a* y *t*, que corresponde a los componentes de la configuración que manejará cada partícula. El movimiento del enjambre se realizará en pasos iterativos y consecutivos dentro del algoritmo.

En cada iteración k-ésima del método, en donde k será calibrado en las pruebas, cada una de las partículas de la población recorre el espacio de soluciones con una velocidad *Vi* hacia nuevas posiciones *Xi*, de acuerdo con su propia experiencia *Pi*, y con la experiencia aportada por el mejor de sus vecinos *G*.

Ahora lo importante del método de optimización recae en la función de fitness que determinará la calidad de la solución encontrada en cada iteración por la partícula y que actualizará los valores de los vectores *Pi* y *G*. Esta función que medirá la calidad de la solución se encontrará minimizando el error encontrado al aplicar la diferencia (ecuación 4.2) entre el valor esperado *x(t+1)* y el valor real *y(t)* entregado por la red al atravesar la red (forward en la red) según la configuración que entrega una solución *Xi* de una partícula *i* y los valores de entrada utilizados.

$$|e| = x(t+1) - y(t) \qquad \textbf{(4.2)}$$

Por lo tanto la función de fitness buscará minimizar el error definido en la ecuación (4.2) al hacer atravesar todas las partículas por la red y obtener un pronóstico *y(t)* por cada una de las partículas en una iteración, para posteriormente hacer converger este error al mínimo.

4.4 Estructuras y parámetros utilizados.

La forma en que se planteó la solución al problema de implementación fue a través del manejo de vectores y matrices de tamaño estático dentro de posibles tamaños para poder ser calibrarlos en las pruebas de las distintas variables que utilizará la red Wavelet y el algoritmo de PSO.

Luego teniendo los tamaños definidos de las distintas capas y variables a utilizar se definieron vectores en forma global para su fácil manejo en las distintas capas de la red: capa de entrada *X[N]* con *N* entradas, capa oculta *O[H]* con *H* neuronas y una salida *Y* para el valor pronosticado.

Para la representación de los parámetros de red, es decir los distintos pesos asociados *(a, t, w)* en las distintas relaciones internas, estos se encapsularon dentro de una misma estructura que representa una configuración completa de red, en donde el componente de dilatación *a* y el de traslación *t* entre la capa de entrada y la capa oculta están definidas por matrices según los tamaños de las capas, *Wa[N][H]* y *Wt[N][H]* respectivamente, y por su parte los pesos entre la capa oculta y la capa de salida están representados por un vector del tamaño de la capa oculta *Ww[H]*.

Una vez definida y representada la estructura de configuración de red, se definen las variables del PSO de forma global para optimizar la configuración de la red por arreglos de estructuras según el tamaño de partículas *P* a utilizar, obteniéndose así las posiciones de las partículas *Xi[P]*, la velocidad de cada partícula *Vi[P]*, la mejor solución encontrada por la partícula *Pi[P]* y la mejor solución global encontrada *G*. Se inicializan aleatoriamente las posiciones *Xi, Pi, G y Vi* de cada partícula y se comienza con las iteraciones del PSO.

En cada iteración del PSO las partículas *i* varían su posición *Xi* según la velocidad *Vi* que tomen debido a las soluciones encontradas por la propia partícula y la mejor de sus demás iguales, según se describió en la ecuación 3.1 y 3.2 del Capítulo 3. En esta variación de velocidad se considerarán los valores cognitivos *C1* y social *C2* según los valores propuestos por la literatura y a los valores *r1* y *r2* de forma aleatoria entre 0 y 1 para representar lo impredecible del comportamiento de una partículas.

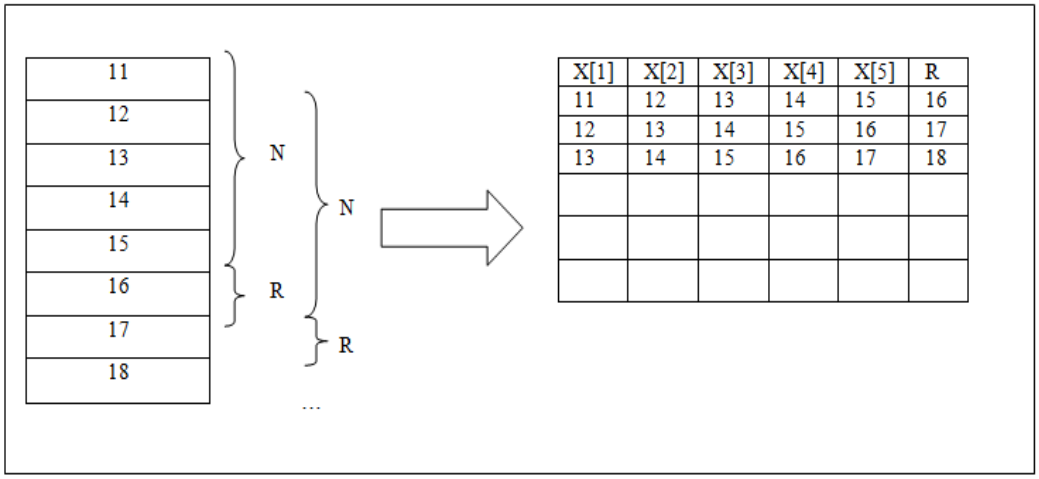

Ilustración 4.2: ejemplo de creación del archivo de N=5 entradas de red, guardando el valor real R

Para poder evaluar la calidad de las soluciones o posiciones *Xi* en que se encuentran las partículas es necesario evaluarlas según la función de fitness, definida en capítulos anteriores, en donde se desea minimizar la diferencia entre el valor pronosticado *y(t)* y el valor real *x(t+1)*, es decir minimizar el error $e = y(t) - x(t+1)$, para lo cual será necesario hacer un forward en la red y recorrerla según la configuración (posición) que haya encontrado la partícula en un instante dado y así obtener el valor pronosticado Y de la iteración.

El forward de la red comenzará con la lectura de los datos de entrada de la red *X[N]* desde un archivo que posee los volúmenes mensuales históricos de captura de Anchoveta, acomodado para las distintas lecturas de las entradas de la red y que será trasladado según el tamaño de la entrada y poder rescatar el valor real (Ilustración 4.2), luego en la capa oculta *O[H]* se hará la sumatoria de todas interconexiones que confluyen a una neurona *i* de la capa oculta según se describe en la ecuación 4.3.

$$O[i] = \sum_{j=1}^{N} X[j] * Wavelet(i, Xi[k].Wa[i][j], Xi[k].Wt[i][j]) \quad \textbf{(4.3)}$$

En donde el parámetro *i* de la ecuación 4.3 es la neurona de la capa oculta, *j* la entrada que influencia a la neurona *i*, *k* la partícula que estamos analizando, o sea, la configuración que estamos ocupando, *N* el tamaño de las entradas, *Wa* el valor de dilatación y *Wt* el valor de traslación de la Wavelet.

La función Wavelet madre que ocuparemos será una de las más recomendadas por la literatura investigada al respecto [23], y que en este caso será la segunda derivada de una Gaussiana, también llamada como sombrero mexicano por la forma que adquiere su gráfica y se encuentra definida según la ecuación 4.4.

$$\psi(t) = (1 - t^2) e^{-\frac{t^2}{2}} \quad \textbf{(4.4)}$$

Finalmente nuestro valor estimado *y(t)* obtenido por un forward en la red neuronal Wavelet será obtenido mediante la ecuación 4.5.

$$Y = \sum_{i=1}^{H} O[i] * Xi[k].Ww[i] \quad \textbf{(4.5)}$$

En donde el valor *O[i]* es el resultado o salida de la neurona *i* de la capa oculta, *Xi* es la posición actual de la partícula, *k* es la partícula que se está analizando o ejecutando, *H* es tamaño de la capa oculta y *Ww* es el peso de salida de la neurona de la capa oculta.

4.5 Pseudo Código

En esta sección del capítulo se presentan los pseudo códigos de los algoritmos más utilizados e importantes en la implementación del modelo de pronóstico para las Anchovetas, como lo son: el algoritmo de calibración de la red PSO Clásico y el forward de la red Wavelet.

4.5.1 PSO Clásico

```
Inicio
        InicializarParticulas()

        while  (no se termine el numero de iteraciones) do

                for i = 1 to tamaño_de_particulas do
                        evaluar la solución Xi de la partícula i
                        if fitness(Xi) es mejor que fitness(Pi) then
                                Pi = Xi
                        end if
                        if fitness(Pi) es mejor que fitness(G) then
                                G =Pi
                        end if
                end for

                for i = 1 to tamaño_de_particulas do
                        actualizar velocidad Vi de la partícula de acuerdo con Pi y Gi
                        mover la partícula pi de acuerdo con su nueva velocidad
                end for

        end while
Fin
```

4.5.2 Forward de la red

```
Inicio
        X=Leer_entradas()

        k=Obtener_configuración()
        for i=1 to numero de neuronas capa oculta do

                for j = 1 to numero de entradas do
                        O[i] = O[i] + X[j] * Wavelet (i, Xi[k].Wa[j][i], Xi[k].Ws[j][i])
                end for

        end for

        for i = 1 to numero de neuronas capa oculta do
                Y = Y + O[i] * Xi[k].Ww[i]
        end for

        retornar Y
Fin
```

4.6 Simulación

A continuación se explicará la elección de los distintos parámetros a utilizar en las diversas simulaciones planeadas en la investigación y de cómo éstas fueron llevadas a cabo para obtener los resultados estudiados, para luego ser discutidos y evaluados según algún criterio definido.

Para calibrar el modelo de pronósticos de volúmenes de Anchovetas y obtener los parámetros adecuados para la red neuronal, se desarrollaron simulaciones exhaustivas y consecutivas en la fase de entrenamiento y en la fase de pruebas, para todas las configuraciones posibles. La metodología consistió en integrar de forma automática en una sola prueba todas las combinaciones existentes entre el tamaño de la capa de entrada *N* [2, 3, 4,…12] y la capa oculta *H* [10, 11, 12,…, 20], en donde se mantuvo fijo el número de partículas *P* y el número de iteraciones *K* del método de optimización. Posteriormente, para completar todas las pruebas, se utilizaron las distintas combinaciones antes descritas con un número de partículas *P*=20,30,40 y con un número de iteraciones *K*=1500,2000,3000, obteniendo un número de 9 (*PxK)* pruebas exhaustivas para calibrar el modelo. Cabe destacar que los valores *N, H, P* y *K* posibles que se tomarán son valores recomendados por otros estudios [3] y [21], y que se suponen arrojarán resultados similares en nuestra calibración.

Con todo lo anterior, ya se tienen configuradas las distintas simulaciones y pruebas que se harán con la variante PSO Clásica, ahora bien, por cada una de estas simulaciones clásicas se obtendrá el promedio máximo y mínimo de las velocidades arrojadas por las partículas y se elegirá el valor mayor entre estos dos para ser asignado al valor *VMax* y acotar la velocidad de la partícula al rango [-VMax,VMax] en la variante PSO Max-Min del algoritmo, en donde se restringe la velocidad de la partícula a moverse dentro de estos valores y así realizar una prueba con los parámetros equivalente a la realizada con el PSO Clásico. Cabe en este punto destacar que no existe un criterio único y uniforme al momento de seleccionar el valor *VMax* a elegir en el PSO Max-Min, por lo cual esta es una propuesta e intento por restringir la velocidad de la partícula según el promedio de la velocidad observada en las partículas, para que así no se alejen demasiado del promedio de la velocidad alcanzada y así poder obtener mejores resultados.

Otro punto importante en las simulaciones lo será el valor cognitivo *C1* y social *C2* que adopte el PSO, tanto Clásico como en su variante Max-Min, el cual para ambos coeficientes *C1* y *C2* adoptará en todas las simulaciones realizadas el valor 1,49 manteniéndose constante y que según [24] es un valor que comienza a arrojar buenos resultados, sin desmerecer otros valores que aporta la literatura y que nos proponen que la suma de *C1* y *C2* sea menor a cuatro (*C1+C2<4*) y que también arroja buenos resultados.

Al finalizar una simulación configurada con coeficientes *C1=C2=1,49*, un tamaño *P* de partículas, un número *K* de iteraciones y un valor *VMax* (si corresponde al algoritmo), se obtendrán ciertos estimadores o métricas que nos harán posible la validación y comparación de los distintos modelos obtenidos en las simulaciones para los variados tamaños de capas de red estudiados.

Capítulo 5
Discusión de Resultados

En este capítulo se expondrán los resultados encontrados en las distintas simulaciones y pruebas, a los cuales se les hará un análisis comparativo en la versión Clásica de PSO y su variante Max-Min, según ciertos parámetros de validación escogidos, los que serán presentados en tablas comparativas de los valores obtenidos para ser ocupados en la justificación de la elección del mejor modelo.

5.1 Métricas

Para determinar el funcionamiento de los modelos durante las fases de validación, varias serán las métricas aplicadas pues no hay una prueba única y más conveniente en la evaluación del funcionamiento de un modelo.

A continuación se describen las métricas elegidas como parámetros de validación y comparación de los distintos modelos obtenidos en las simulaciones realizadas, las cuales nos servirán como justificativo al momento de discutir los resultados y poder emitir un juicio acerca de los valores obtenidos [21].

5.1.1 Raíz Cuadrada del Error Cuadrático Medio

La raíz del error cuadrático medio (square root of the mean square error - RMSE), nos sirve para cuantificar el error entre las estimaciones y las medicines reales, está dado por:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (R - Y)^2}{N}} \quad \textbf{(5.1)}$$

$$RMSE = \sqrt{MSE} \quad \textbf{(5.2)}$$

Donde N es el número total de validaciones o pruebas con la data, R es el valor real, Y es el valor estimado por la red y MSE es el error cuadrático medio (mean square error).

El $RMSE$ tomará valores cercanos a cero si la diferencia entre lo estimado y lo real es mínima, por lo cual entre más cercano a cero sea este valor mejor será nuestro modelo.

5.1.2 Porcentaje de Error Absoluto

El porcentaje de error absoluto (absolute percentage error - *APE*) también da cuenta de la medida del error en las estimaciones del modelo, pero en términos absolutos y porcentualmente.

$$APE = \left| \frac{R - Y}{R} \right| \tag{5.3}$$

En donde *R* es el valor real, *Y* es el valor estimado y el valor del *APE* esta expresado porcentualmente según el error encontrado en la medición realizada.

Al igual que el *RMSE*, el *APE* entre más pequeño mejores serán los resultados encontrados por el modelo y su valor se acercará a cero cuando esta diferencia sea muy pequeña.

5.1.3 Porcentaje de Error Medio Absoluto

El porcentaje de error medio absoluto (mean absolute percentage error - *MAPE*) da cuenta de la media del error encontrado en una muestra de estimaciones.

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{R_i - Y_i}{R_i} \right| \tag{5.4}$$

En donde R_i es el valor real, Y_i es el valor estimado, *i* es la instancia actualmente revisada y *n* es el tamaño de las validaciones a realizar.

Como se puede apreciar, el *MAPE* no es más que el promedio de todos los *APEs* de la muestra obtenida. Puede variar en rango dependiendo del modelo, sin embargo, un buen modelo sería un MAPE bajo el 20%

5.1.4 Coeficiente de Eficiencia

El coeficiente de eficiencia o de determinación R^2 describe la proporción total de variación en los datos observados que se pueden explicar por el modelo.

Un valor encontrado cercano a 1 o a 100% indica un buen modelo encontrado, por lo cual entre mayor sea este valor, mejor será nuestro modelo.

$$R^2 = 1.0 - \frac{\sum_{i=1}^{n} \left| R - Y \right|^2}{\sum_{i=1}^{n} \left| R - \overline{R} \right|^2} \tag{5.5}$$

En donde *n* es el numero de validaciones, *R* es el valor real, *Y* el valor estimado y $\overline{R}$ es el valor medio de los valores reales.

5.1.5 Criterio de Akaike Modificado

Se basa en el concepto de entropía, ofreciendo una medida relativa de la perdida de información cuando un determinado modelo se utiliza para describir la realidad. Se puede decir que describe la desventaja entre el sesgo y la varianza en el modelo construido, o bien, que describe de cierta medida la precisión y la complejidad del modelo encontrado.

Nosotros consideraremos la versión modificada de Akaike, pues se ajusta más a la realidad del problema modelado y se encuentra definido por:

$$AIC = \log(MSE) + \frac{2m}{N} \qquad \textbf{(5.6)}$$

En donde *MSE* es el error cuadrático medio, *m* es el número de parámetros del modelo y *N* el número de validaciones. En nuestro caso el número de parámetros será *m=2(n*h)+h*, donde *n* es el número de entradas de la red y *h* el número de neuronas de la capa oculta, esto debido a las componentes de dilatación *a*, traslación *t* y de los pesos de salida *w*.

El *AIC* no es una prueba a las hipótesis del modelo, sino que es una herramienta para seleccionar los modelos adecuados. Habiendo un conjunto de parámetros y varios modelos configurados, estos pueden ser ordenados de acuerdo a sus *AIC*, donde el modelo con el menor *AIC* será el mejor y este será el criterio para ordenar los resultados obtenidos.

5.2 Modelos Obtenidos

Las pruebas realizadas estarán configuradas según un número *P = [20 , 30 , 40]* de partículas y un número *K = [1500 , 2000 , 3000]* de iteraciones, existiendo nueve posibles combinaciones de los valores de *P* y *K*, tanto para la versión PSO Clásica como para la variante PSO Max-Min, para el cual se indicará el valor de *VMax* utilizado, el cual es obtenido luego de realizar un grupo de pruebas con el PSO Clásico configurado con *P* y *K* para la determinada prueba. Este valor *VMax* es obtenido con el promedio de las velocidades absolutas alcanzadas en la prueba, el que por lo general oscila entre 0,02 y 0,06. Por otra parte nuestro valor cognitivo C1 y social C2 que utilizará el PSO será constante y tendrá valor 1,49 para ambas versiones de PSO.

A continuación se mostrarán los resultados obtenidos según los parámetros configurados de *N = [2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12]* neuronas de entradas y *H=[10 , 11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20]* neuronas ocultas, las cuales son probadas exhaustivamente en su totalidad (N x H) y son elegidas para cada versión o grupo de pruebas, según los parámetros de validación, el mejor, el promedio y el peor de los resultados encontrados para posteriormente ser comparados los mejores resultados entre las dos versiones de PSO.

5.2.1 Prueba 1

La primera prueba consta de los valores primarios que adoptará el modelo, de ahí en adelante estos se irán incrementando para explorar todo el campo de combinaciones. En la tabla 5.1 se proponen los valores iniciales.

Tabla 5.1: Parámetros de la primera prueba.

P	= 20	(partículas)
K	= 1500	(iteraciones)
VMax	= 0,03	

5.2.1.1 PSO Clásico:

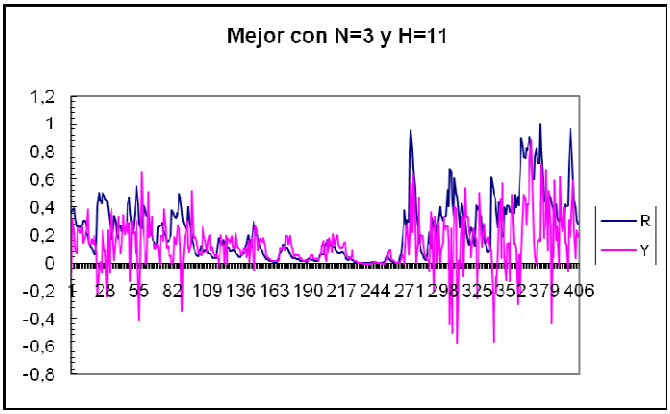

Gráfico 5.1: con N = 3 y H = 11

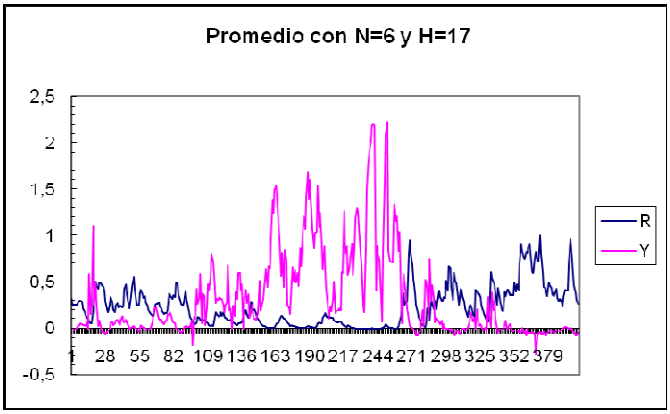

Gráfico 5.2: con N = 6 y H = 17

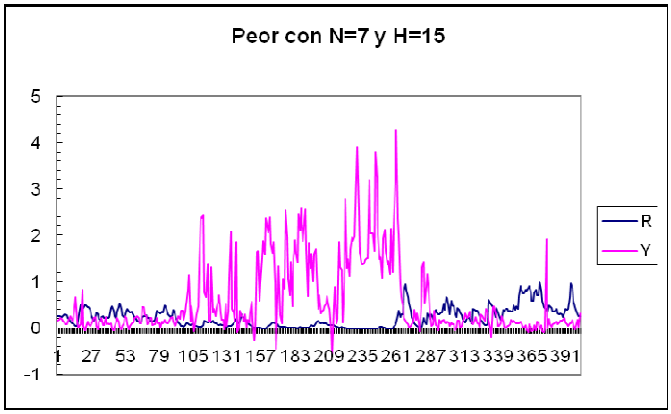

Gráfico 5.3: con N = 7 y H = 15

Tabla 5.2: Parámetros de validación de la primera prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
3	11	0,260041	0.067621	0,724141	0,481445	-0,844794
6	17	0,621632	0,386426	0,979824	0,430100	0,066863
7	15	0,986999	0,974167	0,992300	0,206944	1,03341

Como se puede apreciar en la prueba 1 con PSO Clásico los gráficos 5.1, 5.2 y 5.3 demuestran que el comportamiento de las gráficas se pueden justifica con los valores de los estimadores que estas configuraciones arrojan, así el mejor de estos tres nos arroja un RMSE=0,26 muy cercano a cero y a su vez un AIC= -0,844 muy pequeño según la tabla 5.2, pero un coeficiente de eficiencia muy pequeño R^2= 0,48.

5.1.1.2 PSO Max-Min:

Ahora se aplicará esta misma configuración a la variante Max-Min del algoritmo PSO, utilizando como parámetro restrictivo el valor VMax=0,03.

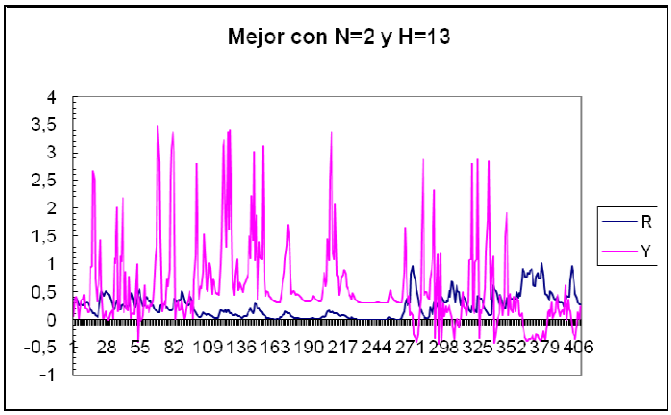

Gráfico 5.4: con N = 2 y H = 13

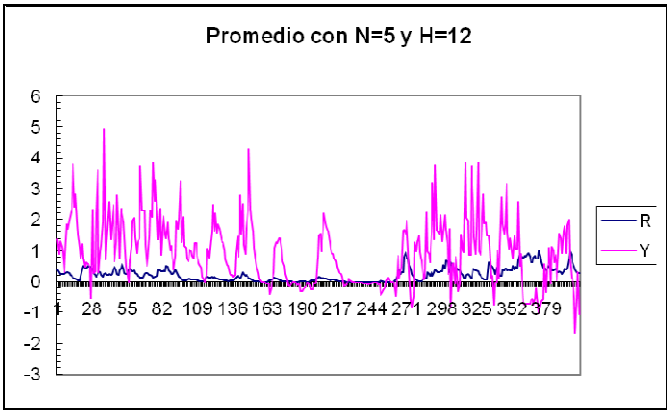

Gráfico 5.5: con N = 5 y H = 12

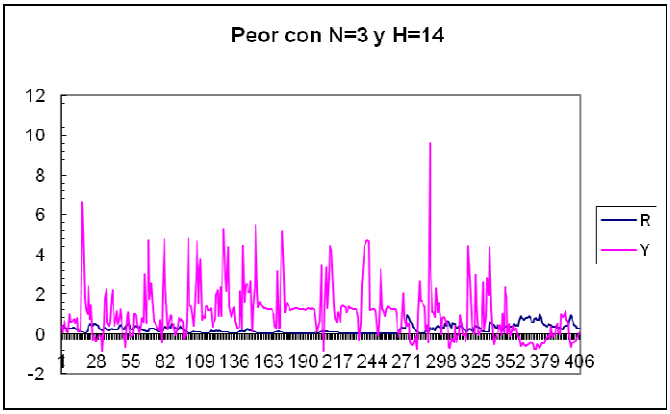

Gráfico 5.6: con N = 3 y H = 14

Tabla 5.3: Parámetros de validación de la primera prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
2	13	0,905144	0,819285	0,603431	0,581365	0,168963
5	12	1,237191	1,530641	0,786417	0,432770	0,778933
3	14	1,642935	2,699235	0,776202	0,134884	0,845034

En la versión Max-Min de la primera prueba, en los gráficos 5.4, 5.5 y 5.6 existe una clara tendencia a maximizar los valores altos de la data e incluso en el mejor de los resultados estos se alzan sobre la data original en gran medida. Esto se puede ver justificado por los valores de los parámetros RMSE y R^2 que son muy elevados a los valores que deberían tomar, mostrados en la tabla 5.3.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.1 con N=3 y H=11 y el gráfico 5.4 con N=2 y H=13.

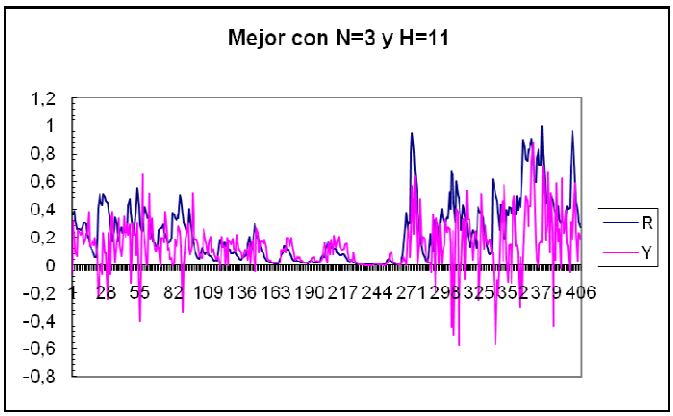

Gráfico 5.1: con N = 3 y H = 11 – PSO Clásico

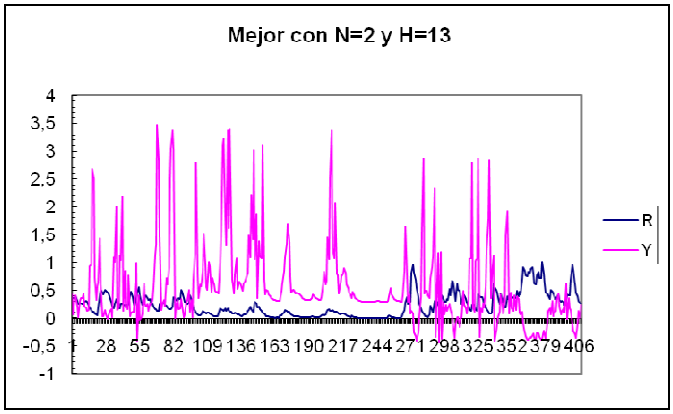

Gráfico 5.4: con N = 2 y H = 13 – PSO Max-Min

Se puede apreciar que el PSO Clásico tiene gran ajuste a la data e incluso en partes bajas de esta se intercepta con la real, sin embargo tiende a descontrolarse y a ser irregular con valores medianamente altos, arrojando incluso valores negativos. Por su parte Max-Min sigue la tendencia, pero arrojando valores muy altos en comparación a la data real.

5.1.2 Prueba 2

La segunda ronda de pruebas mantiene el anterior valor de P=20 partículas e incrementa el número de K de iteraciones en 500, buscando los mejores resultados y la velocidad promedio que adoptan las partículas.

Tabla 5.4: Parámetros de la segunda prueba

P	= 20	(partículas)
K	= 2000	(iteraciones)
VMax	= 0,026	

5.1.2.1 PSO Clásico:

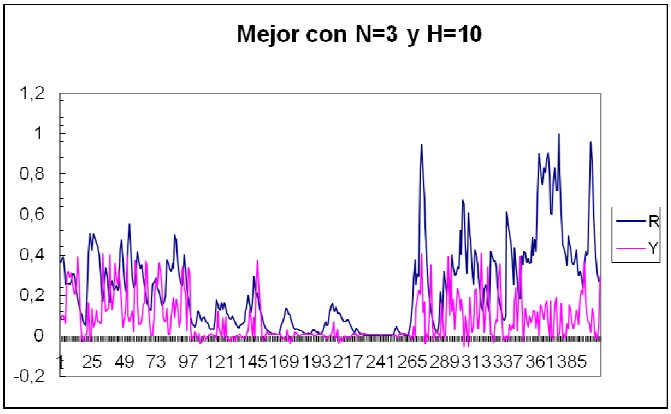

Gráfico 5.7: con N = 3 y H = 10

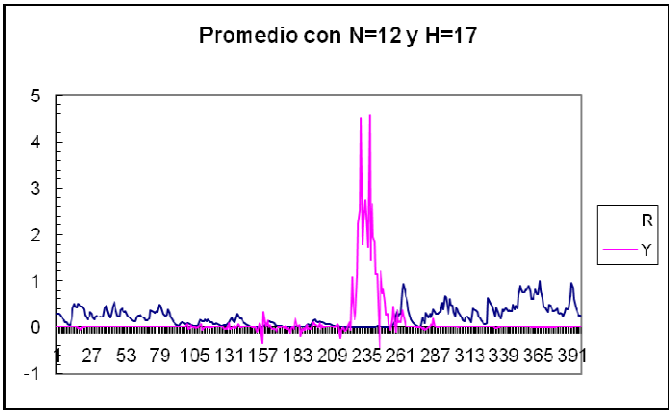

Gráfico 5.8: con N = 12 y H= 17

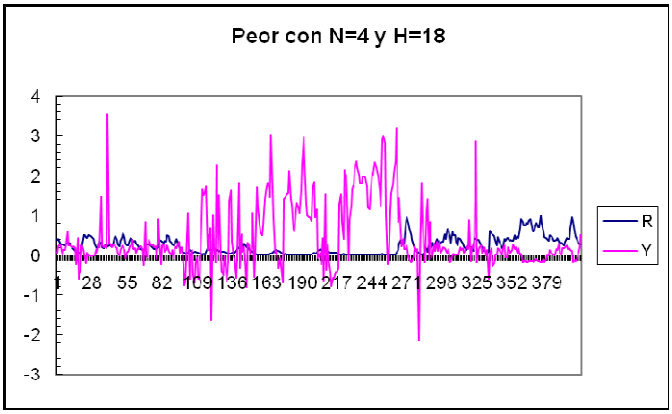

Gráfico 5.9: con N = 4 y H = 18

Tabla 5.5: Parámetros de validación de la segunda prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
3	10	0,262187	0,068742	0,842366	0,506005	-0,86721
12	17	0,592989	0,351635	0,955884	0,560738	1,601509
4	18	0,935716	0,875564	0,941518	0,150885	0,6534

En la segunda tanda de pruebas con PSO Clásico la tendencia en la data estimada del gráfico 5.7 es buena y refleja el comportamiento de la data real, sin embargo los gráficos 5.8 y 5.9 presentan una clara lejanía a medida que se aumenta en los parámetros N y H neuronas, centrándose en un punto de la data y arrojando valores muy alejados del real, es por esto que en la tabla 5.5 tenemos valores RMSE y MAPE muy altos.

5.1.2.2 PSO Max-Min:

Ahora los resultados de la variante Max-Min de PSO aplicando VMax=0,026.

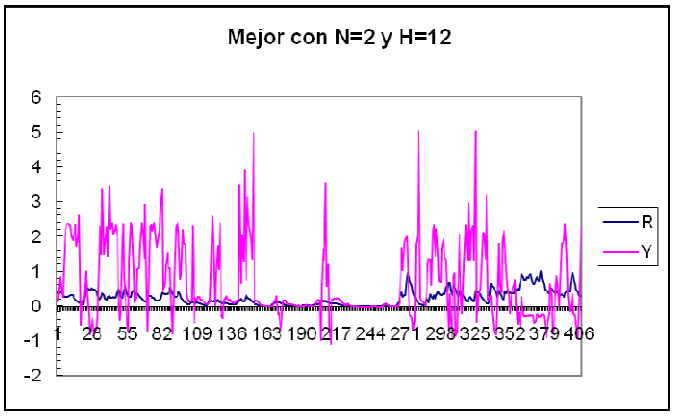

Gráfico 5.10: con N = 2 y H= 12

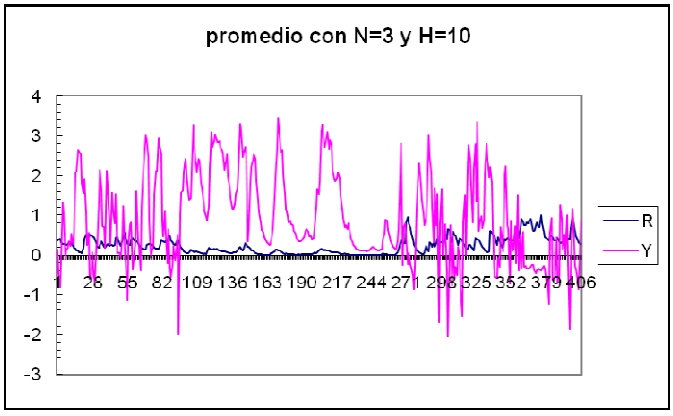

Gráfico 5.11: con N = 3 y H= 10

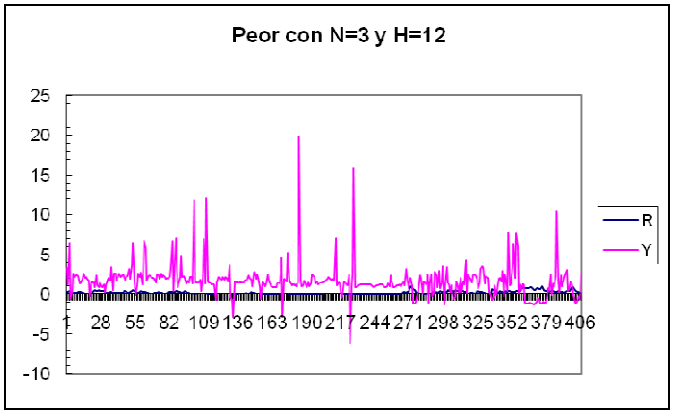

Gráfico 5.12: con N = 3 y H= 12

Tabla 5.6: Parámetros de validación de la segunda prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
2	12	1,186939	1,408824	0,682745	0,280357	0,384729
3	10	1,371727	1,881634	0,773344	0,222851	0,570102
3	12	2,557431	6,540453	0,961817	0,188315	1,170288

Como se puede apreciar en la tabla 5.6 de los estimadores, estos valores se alejan muchísimos de los ideales e incluso el mejor de ellos se puede desechar al poseer un R^2 muy pequeño para ser un modelo relativamente aceptable, estos quizás por el valor demasiado acotado de VMax=0,026.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.7 con N=3 y H=10 y el gráfico 5.10 con N=2 y H=12.

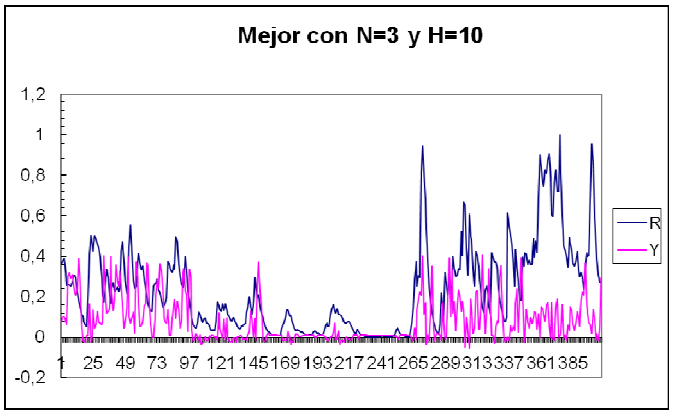

Gráfico 5.7: con N = 3 y H = 10 - Clásico

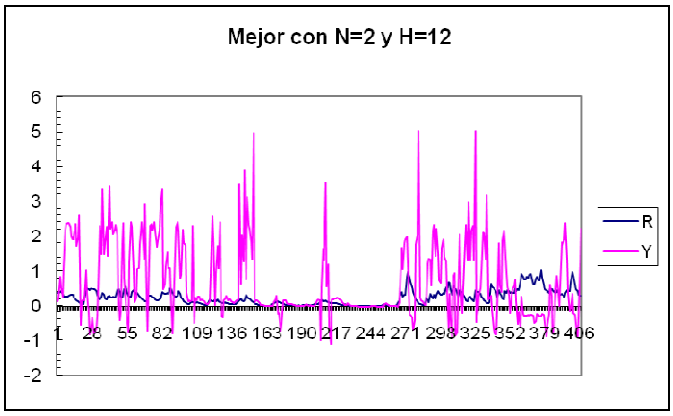

Gráfico 5.10: con N = 2 y H= 12 - Max-Min

Queda claro a simple vista que en este caso el modelo PSO Clásico es superior al PSO Max-Min, puesto que este último ni siquiera sigue la tendencia de la data en comparación a la versión clásica y además sus valores de estimación son muy elevados o muy mínimos según los ideales para las métricas, teniendo por una parte un R^2=50% en la versión clásica y un R^2=28% en la versión Max-Min.

5.1.3 Prueba 3

La tercera ronda de pruebas mantiene aún el valor de P=20 partículas constantes e incrementa el número de K de iteraciones en 1000 más, para luego encontrar la velocidad promedio que adoptan las partículas en este caso.

Tabla 5.7: Parámetros de la tercera prueba

P	= 20	(partículas)
K	= 3000	(iteraciones)
VMax	= 0,050	

5.1.3.1 PSO Clásico:

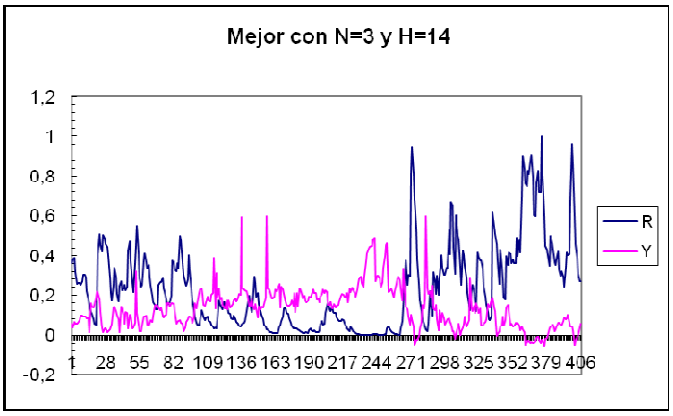

Gráfico 5.13: con N = 3 y H = 14

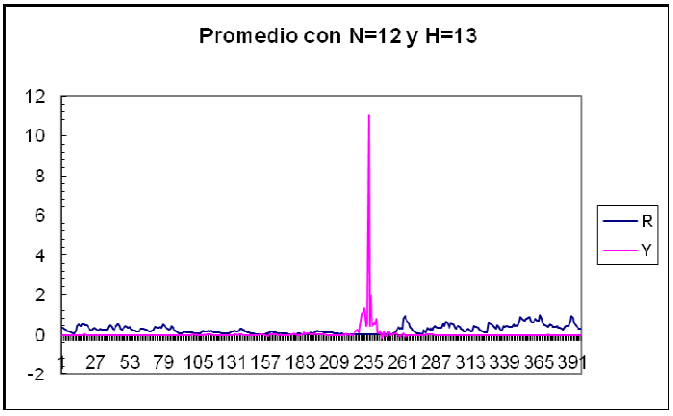

Gráfico 5.14: con N = 12 y H = 13

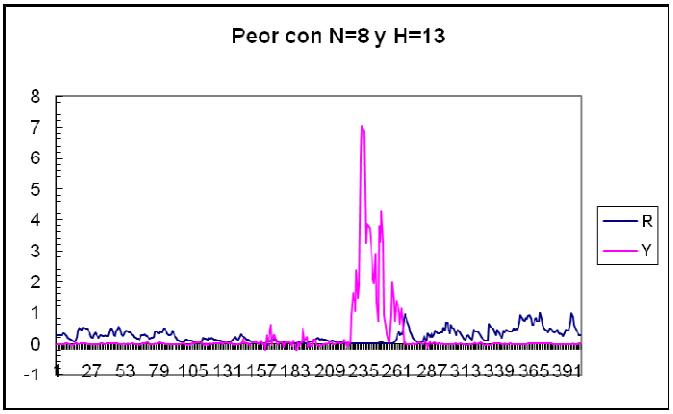

Gráfico 5.15: con N = 8 y H= 13

Tabla 5.8: Parámetros de validación de la tercera prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
3	14	0,312756	0,097816	0,752937	0,142960	-0,595795
12	13	0,658764	0,433997	0,818815	0,013104	1,209248
8	13	0,986138	0,972468	0,872947	0,011774	1,025282

Los resultados encontrados en la tercera tanda de pruebas con PSO Clásico no son muy satisfactorios, puesto que los valores de R^2 son muy pequeños a pesar del pequeño RMSE que presentan en la tabla 5.8. Es así como ni siquiera el mejor de los resultados (Gráfico 5.13) no se aproxima a una posible solución. Por otra parte se puede apreciar gráficamente que la data real y la data estimada toman formas muy dispares.

5.1.3.2 PSO Max-Min:

Ahora los resultados de la variante Max-Min, aplicando VMax=0,050.

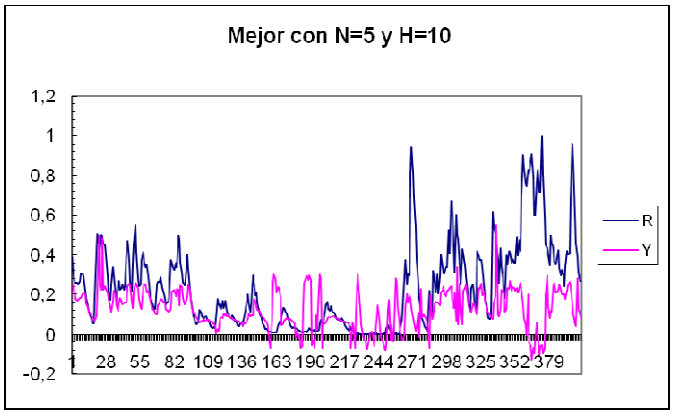

Gráfico 5.16: con N = 5 y H= 10

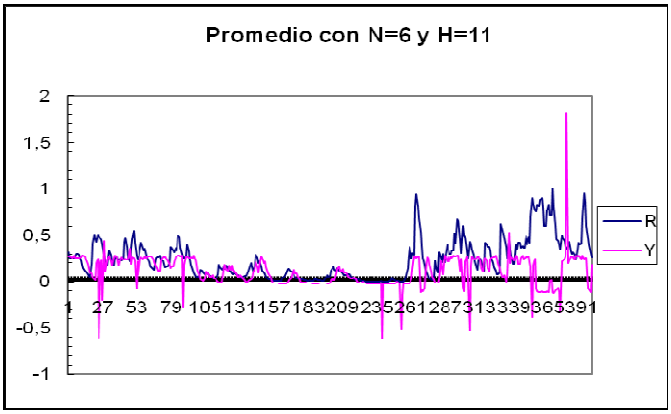

Gráfico 5.17: con N = 6 y H = 11

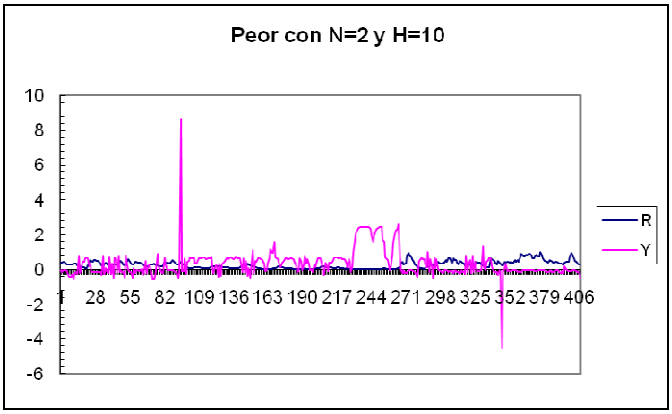

Gráfico 5.18: con N = 2 y H = 10

Tabla 5.9: Parámetros de validación de la tercera prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
5	10	0,242397	0,058756	0,697385	0,712833	-0,735897
6	11	0,555795	0,308908	0,721541	0,639007	0,144916
2	10	0,905365	0,819685	0,831617	0,019939	0,110207

En la versión Max-Min de la tercera prueba los valores de los estimadores del Gráfico 5.16 son valores muy cercanos a los óptimos con un R^2=71%, un RMSE=0,24 y se comporta bastante bien al momento de representar la data gráficamente. Estos valores son similares al Gráfico 5.17, pero el valor de R^2=63% es pequeño y posee poco ajuste gráficamente con la data real.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.13 con N=3 y H=14 y el gráfico 5.16 con N=5 y H=10.

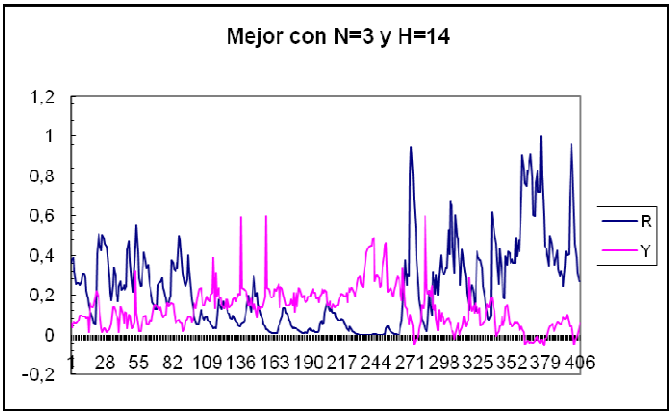

Gráfico 5.13: con N = 3 y H = 14 - Clásico

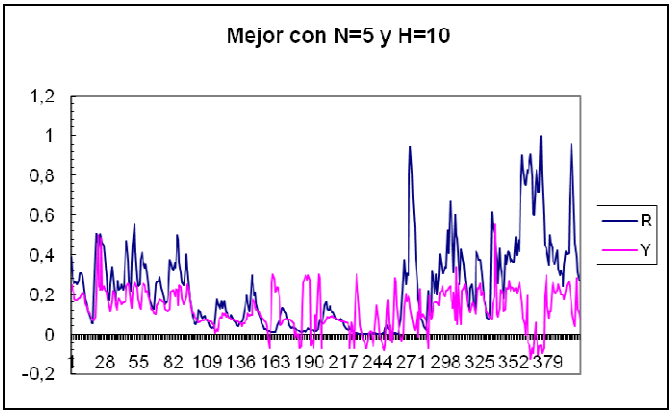

Gráfico 5.16: con N = 5 y H= 10 - Max-Min

En esta oportunidad la versión Max-Min del PSO mejora sustancialmente los resultados de la configuración y es mejor en varios aspectos al resultado encontrado con la versión clásica del algoritmo. Se obtiene un R^2=71% muy bueno en comparación a los valores ya encontrados y se puede apreciar gráficamente un cierto ajuste en data, tanto en la tendencia y en la coincidencia de los datos estimados con los reales.

5.1.4 Prueba 4

Con la cuarta ronda de pruebas se incrementa el valor de P partículas en 10 y el número de K de iteraciones vuelve a 1500, según la tabla 5.10.

Tabla 5.10: Parámetros de la cuarta prueba

P	= 30	(partículas)
K	= 1500	(iteraciones)
VMax	= 0,055	

5.1.4.1 PSO Clásico:

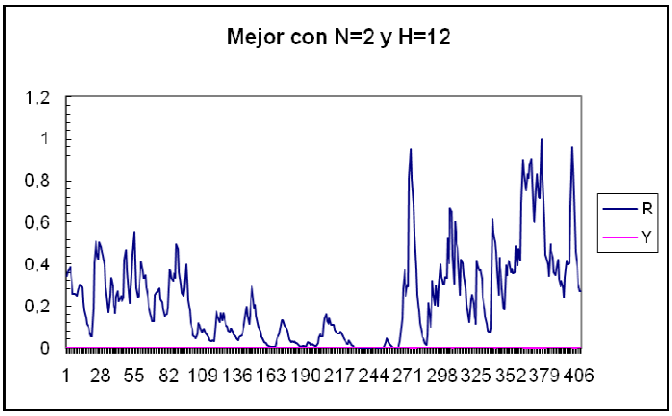

Gráfico 5.19: con N = 2 y H = 12

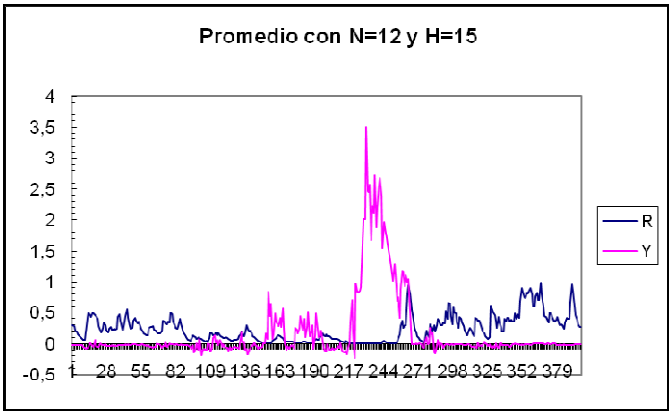

Gráfico 5.20: con N = 12 y H = 15

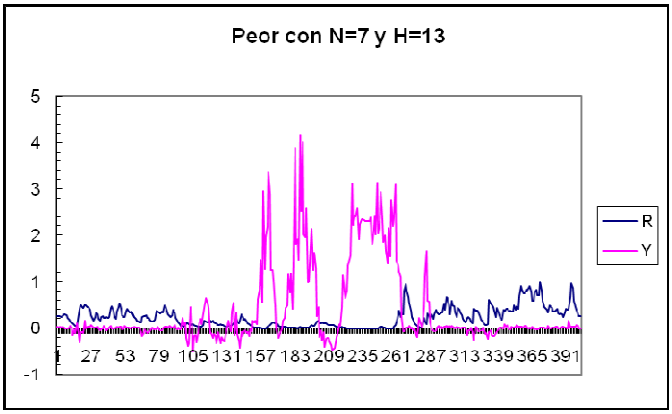

Gráfico 5.21: con N = 7 y H = 13

Tabla 5.11: Parámetros de validación de la cuarta prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
2	12	0,316094	0,099915	0,670116	0,192907	-0,764495
12	15	0,646097	0,417441	0,926858	0,135657	1,434197
7	13	0,983066	0,966418	0,566785	0,038254	0,890638

La cuarta prueba con PSO Clásico los valores de los estimadores no se representan claramente en el gráfico, ya que los valores de la data estimada son tan pequeños que a la vez arrojan pequeños errores, los cuales generan parámetros de validación (tabla 5.11) dentro de lo normal e incluso engañadores al momento de realizar comparaciones al tomar por buenos los modelos obtenidos, es así como el mejor valor de RMSE=0,31 del gráfico 5.19 nos induce a ser elegido como el mejor modelo, pero con el valor de R^2=19% nos demuestra lo contrario y a simple vista puede ser desechado.

5.1.4.2 PSO Max-Min:

Ahora los resultados de la variante Max-Min, aplicando VMax=0,055.

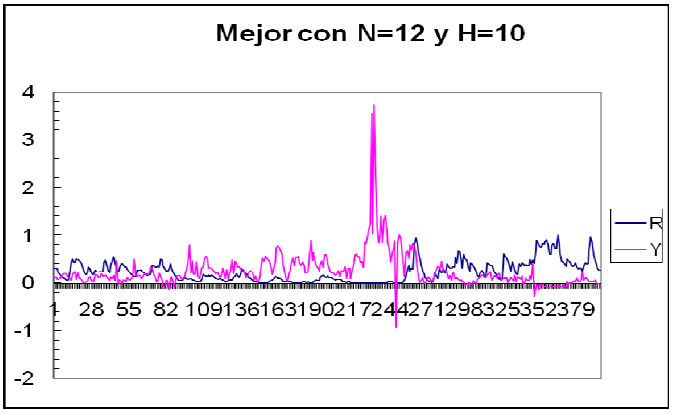

Gráfico 5.22: con N = 12 y H = 10

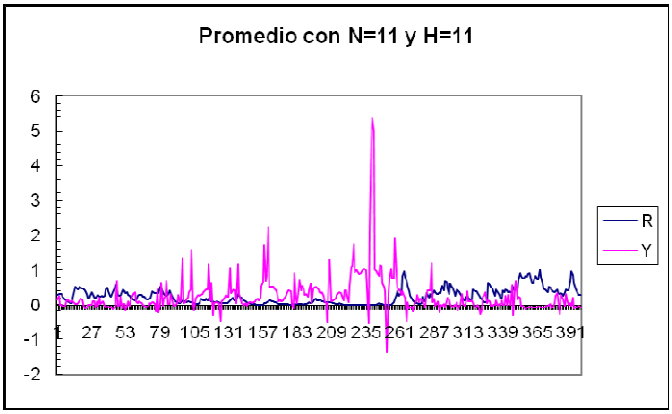

Gráfico 5.23: con N = 11 y H = 11

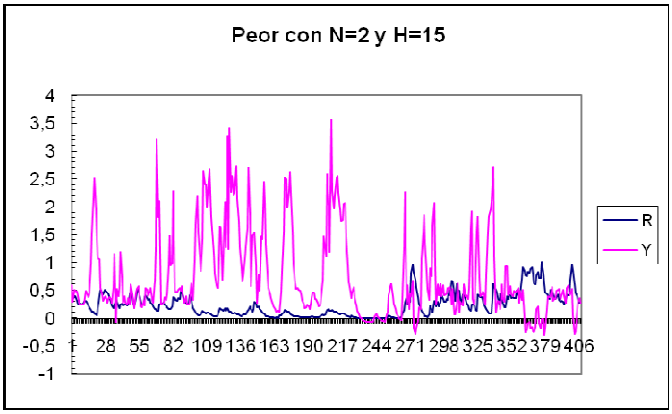

Gráfico 5.24: con N = 2 y H = 15

Tabla 5.12: Parámetros de validación de la cuarta prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
12	10	0,506483	0,256525	0,977183	0,515704	0,618198
11	11	0,662056	0,438318	0,910967	0,447724	0,857870
2	15	0,985672	0,971549	0,767061	0,323212	0,282306

El aplicar la variante Max-Min en el cuarto grupo de pruebas no aporta mayores beneficios, ya que los valores de RMSE se mantienen altos según la tabla 5.12 y esto se refleja en los gráficos 5.22, 5.23 y 5.24, que no demuestran tendencia o ajuste alguno de la data real con la data estimada.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.19 con N=2 y H=12 y el gráfico 5.22 con N=12 y H=10.

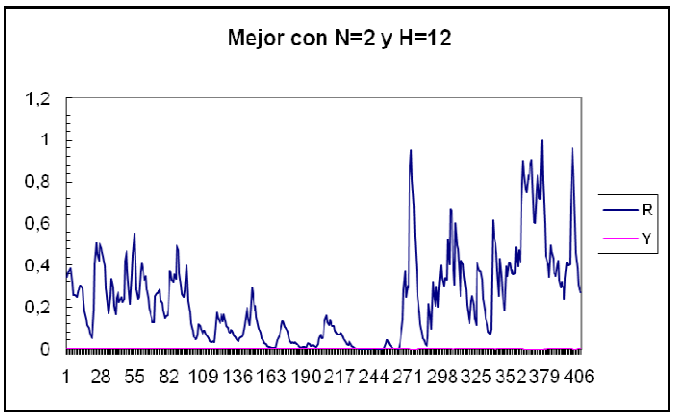

Gráfico 5.19: con N = 2 y H = 12 - Clásico

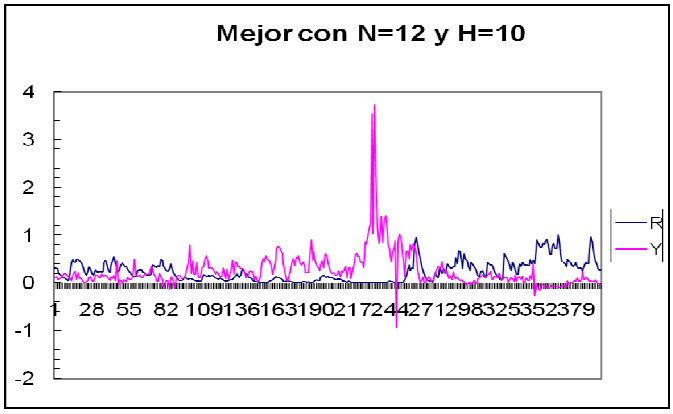

Gráfico 5.22: con N = 12 y H = 10 - Max-Min

Si bien es cierto, la configuración utilizada no arroja buenos resultados en la cuarta prueba y más bien entrega resultados engañosos que se alejan de la data real, se puede apreciar que la variante Max-Min mejora los resultados encontrados y que en ciertos tramos de la data esta sigue al menos una pequeña tendencia en los datos.

5.1.5 Prueba 5

En la quinta ronda de pruebas se mantiene el anterior valor de P=30 partículas y se aumenta en 500 el número de K de iteraciones.

Tabla 5.13: Parámetros de la quinta prueba

P	= 30	(partículas)
K	= 2000	(iteraciones)
VMax	= 0,060	

5.1.5.1 PSO Clásico:

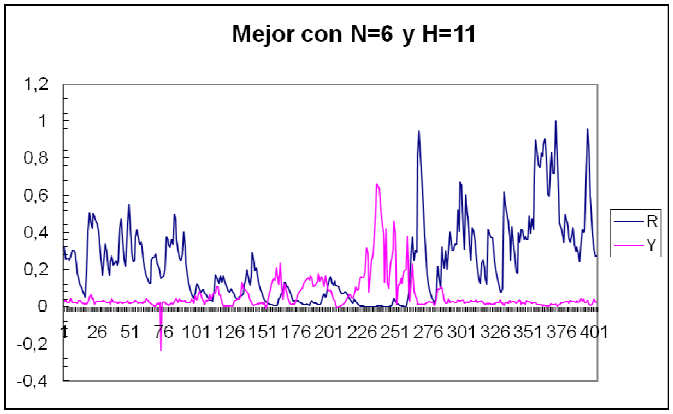

Gráfico5.25: con N = 6 y H = 11

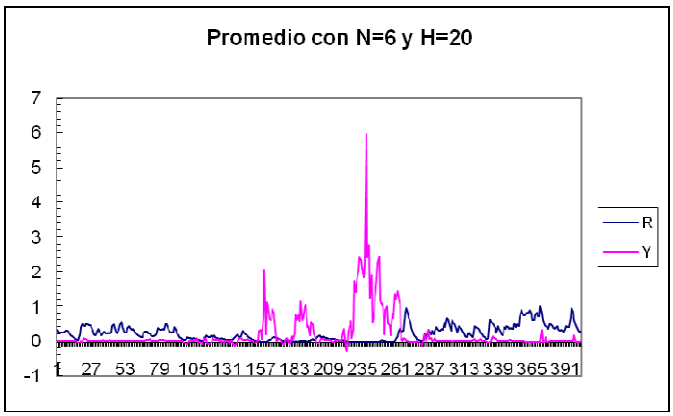

Gráfico 5.26: con N = 6 y H = 20

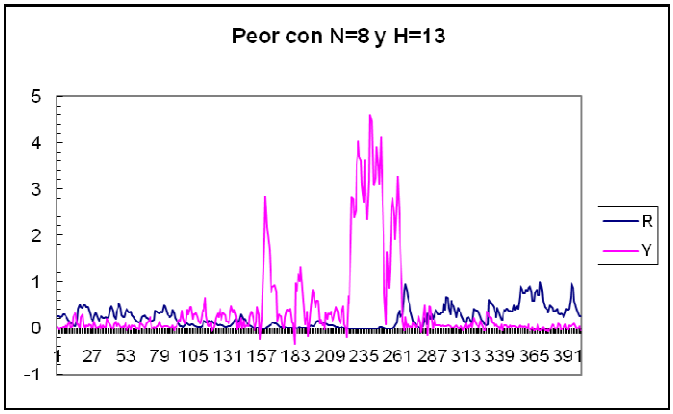

Gráfico 5.27: con N = 8 y H= 13

Tabla 5.14: Parámetros de validación de la quinta prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
6	11	0,317649	0,100900	0,924762	0,420121	-0,341017
6	20	0,652959	0,426355	0,937243	0,501198	0,820839
8	13	0,998120	0,996243	0,967193	0,340585	1,035772

Como se puede apreciar en los gráficos 5.25, 5.26 y 5.27 de la quinta prueba con PSO Clásico la data se vuelve muy irregular y presenta claros problemas con los valores extremos, aquellos que son muy elevados o muy bajos, a diferencia de los puntos de valores intermedios en que la predicción del modelo sigue al menos la tendencia de la data real.

5.1.5.2 PSO Max-Min:

Ahora obtendremos los resultados de la variante Max-Min, aplicando VMax=0,060.

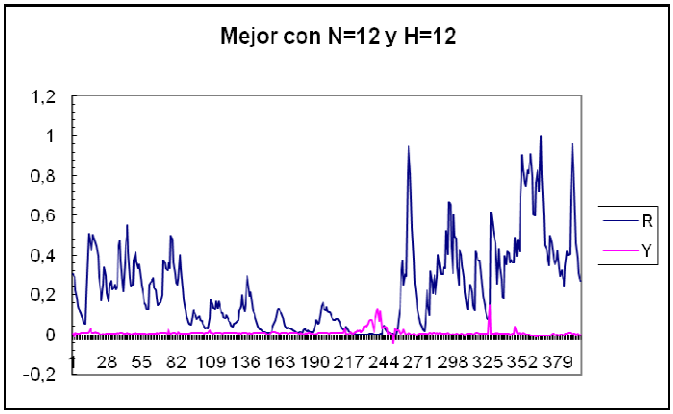

Gráfico 5.28: con N = 12 y H = 12

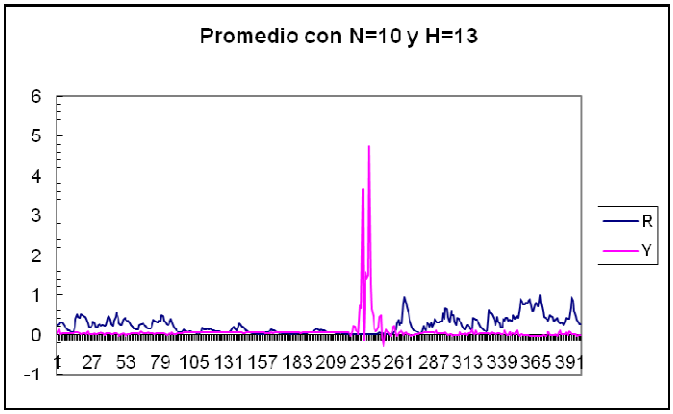

Gráfico 5.29: con N = 10 y H= 13

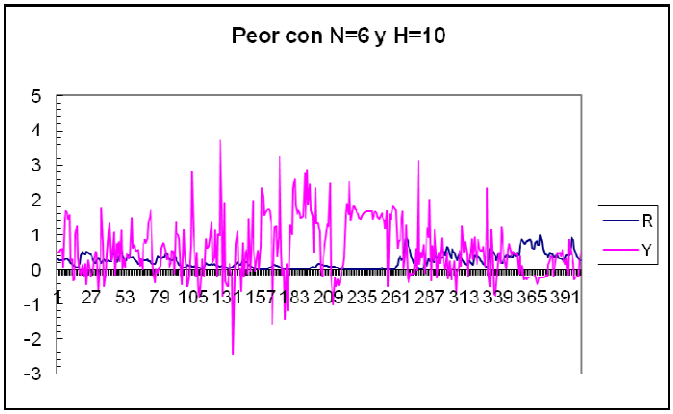

Gráfico 5.30: con N = 6 y H = 10

Tabla 5.15: Parámetros de validación de la quinta prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
12	12	0,314269	0,098765	0,868350	0,123600	0,445483
10	13	0,456713	0,208586	0,800790	0,150721	0,622544
6	10	0,990700	0,981486	0,882427	0,411690	0,587418

Los resultados obtenidos en la quinta prueba no mejoran sustancialmente con la variante Max-Min, manteniendo pequeños valores de R^2 según la tabla 5.15 en donde los valores fueron ordenados de mayor a menor según el RMSE. Esto se puede apreciar claramente en los datos resultantes en los gráficos 5.28, 5.29 y 5.30, en donde la data estimada es muy alejada de la real al ser muy mínima en el gráfico 5.28, catalogado como el mejor por su RMSE=0,31 y en cambio en el gráfico 5.30 la data estimada es mas cercana a la real obteniéndose un R^2=41%, pero con un alto error RMSE=0,99.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.25 con N=6 y H=11 y el gráfico 5.28 con N=12 y H=12.

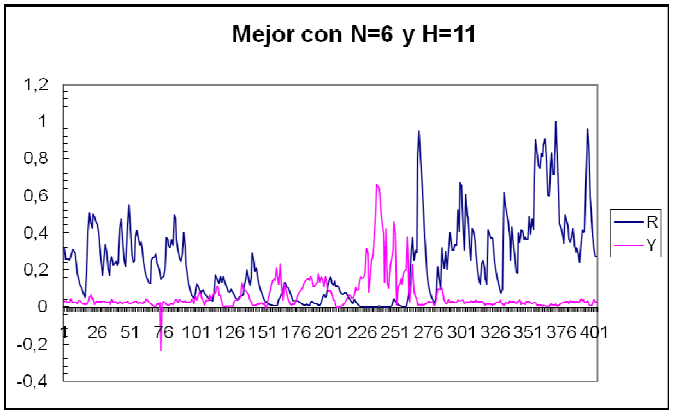

Gráfico5.25: con N = 6 y H = 11 - Clásico

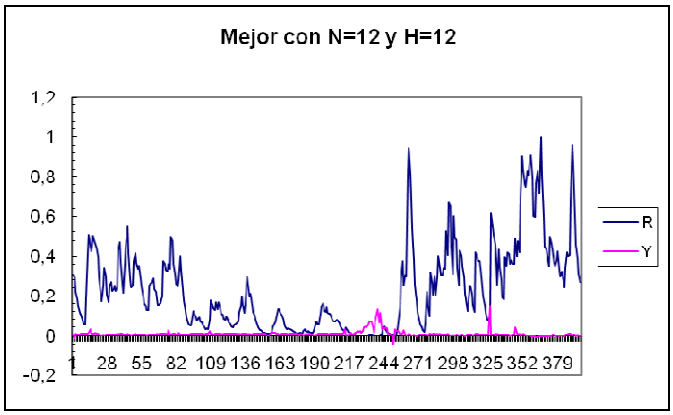

Gráfico 5.28: con N = 12 y H = 12 - Max-Min

Al comparar los resultados arrojados por los modelos en la quinta prueba, la versión clásica se presenta como la mejor, al mantener una mínima tendencia en un rango de valores pequeños, en comparación al Max-Min que arroja estimaciones muy pequeñas que no señalan tendencia alguna de la data estimada con la real, esto guiándonos por la magnitud de los errores encontrados en la quinta prueba.

5.1.6 Prueba 6

La sexta ronda de pruebas mantiene el valor de P=30 partículas en curso y aumenta en 1000 el número de las K de iteraciones.

Tabla 5.16: Parámetros de la sexta prueba

P	= 30	(partículas)
K	= 3000	(iteraciones)
VMax	= 0,058	

5.1.6.1 PSO Clásico:

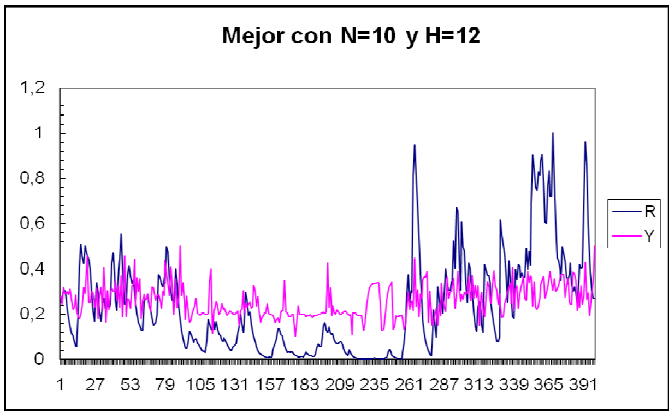

Gráfico 5.31: con N = 10 y H = 12

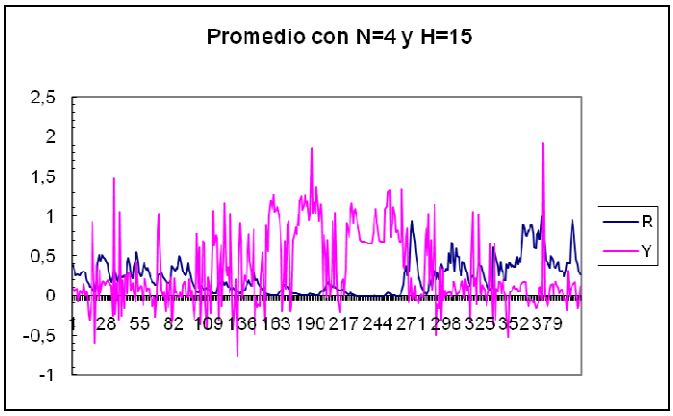

Gráfico 5.32: con N = 4 y H = 15

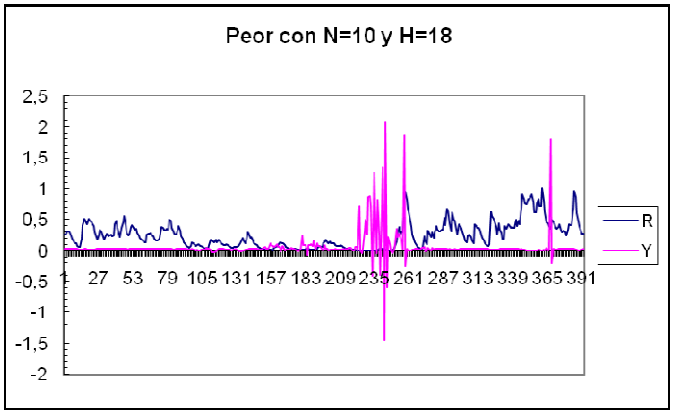

Gráfico 5.33: con N = 10 y H = 18

Tabla 5.17: Parámetros de validación de la sexta prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
10	18	0,392425	0,153997	0,751098	0,518710	1,177463
4	15	0,588099	0,345860	0,850618	0,564878	0,131494
10	12	0,196885	0,038763	0,665056	0,162381	-0,208568

En la sexta prueba con PSO Clásico, el Gráfico 5.31 sigue una cierta tendencia en la data, pero se escapa en la predicción de los valores menores. Los otros gráficos de los modelos analizados no entregan mayor información, ya que su comportamiento es muy irregular y se disparan sin justificación alguna, obteniendo predicciones muy alejadas de la real.

5.1.6.2 PSO Max-Min:

Ahora los resultados de la variante Max-Min, aplicando VMax=0,058.

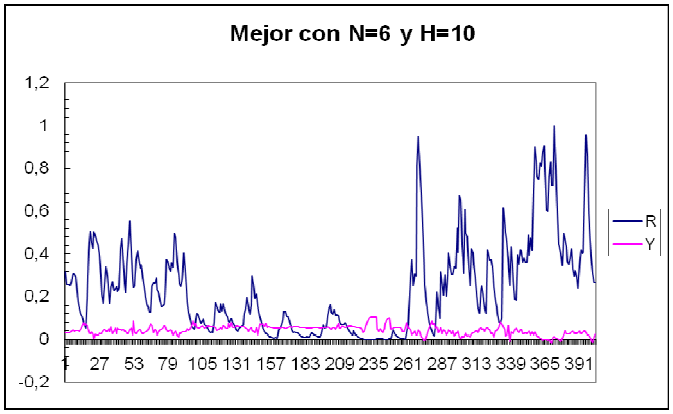

Gráfico 5.34: con N = 6 y H = 10

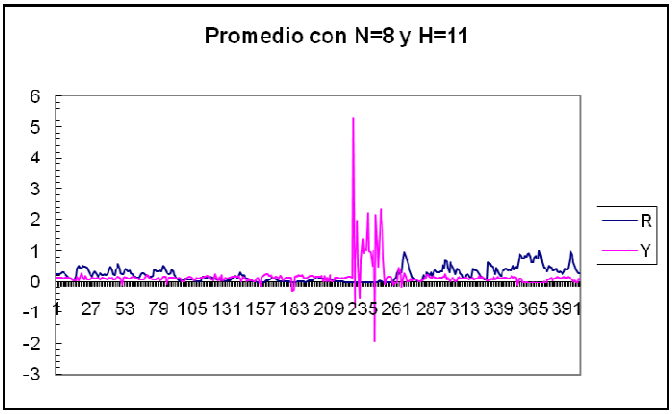

Gráfico 5.35: con N = 8 y H = 11

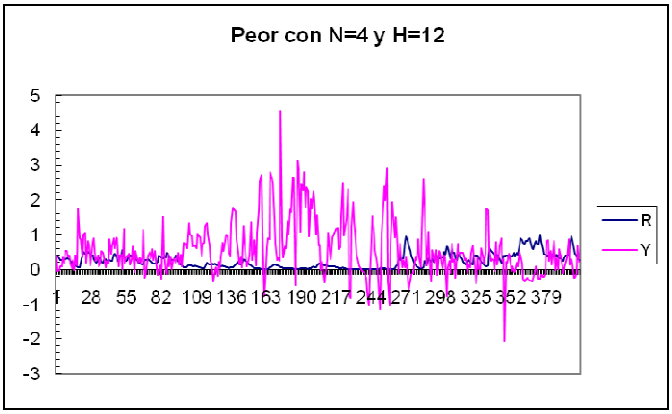

Gráfico 5.36: con N = 4 y H = 12

Tabla 5.18: Parámetros de validación de la sexta prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
6	10	0,298392	0,089037	0,923299	0,482408	-0,454892
8	11	0,493976	0,244012	0,906766	0,298887	0,265218
4	12	0,913810	0,835048	0,754147	0,264669	0,395785

Pese a obtener ciertos estimadores muy buenos en los valores encontrados en la sexta prueba con PSO Max-Min según la tabla 5.18, los resultados gráficos se alejan mucho de ser los deseados debido a que difieren mucho de la data real, es así como los valores de R^2 están lejos de ser los ideales.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.31 con N=10 y H=12 y el gráfico 5.34 con N=6 y H=10.

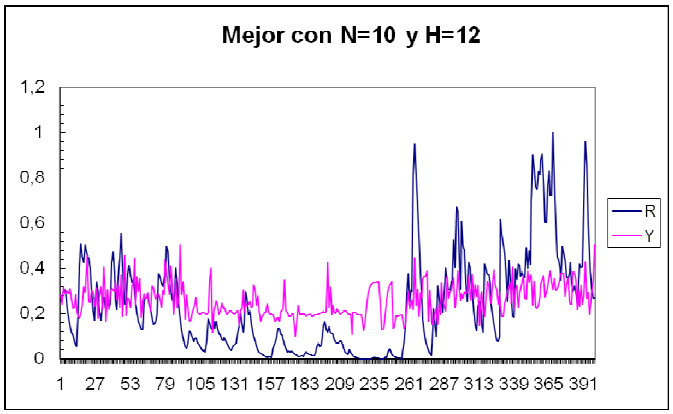

Gráfico 5.31: con N = 10 y H = 12 - Clásico

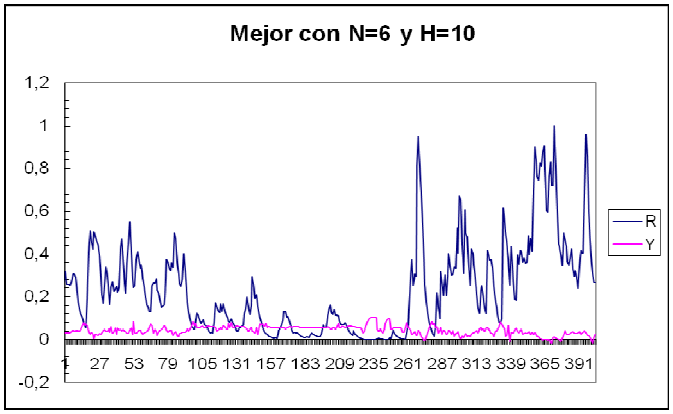

Gráfico 5.34: con N = 6 y H = 10 - Max-Min

En este grupo de pruebas el algoritmo clásico nos lleva a mejores resultados, ya que tiene cierto ajuste a la data real, en comparación a la versión Max-Min que no parece seguir, al menos, esta tendencia.

5.1.7 Prueba 7

La séptima ronda de pruebas aumenta a 40 el número de P partículas en curso y prueba nuevamente con 1500 K iteraciones

Tabla 5.19: Parámetros de la séptima prueba

P	= 40	(partículas)
K	= 1500	(iteraciones)
VMax	= 0,031	

5.1.7.1 PSO Clásico:

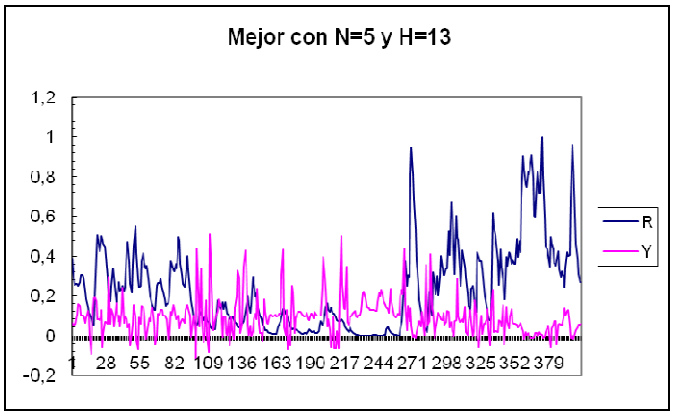

Gráfico 5.37: con N = 5 y H= 13

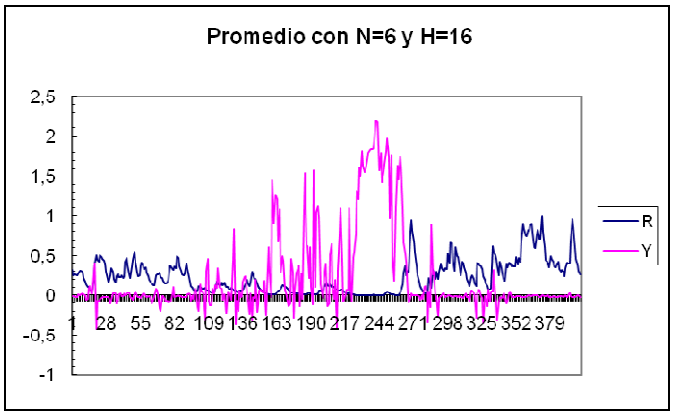

Gráfico 5.38: con N = 6 y H= 16

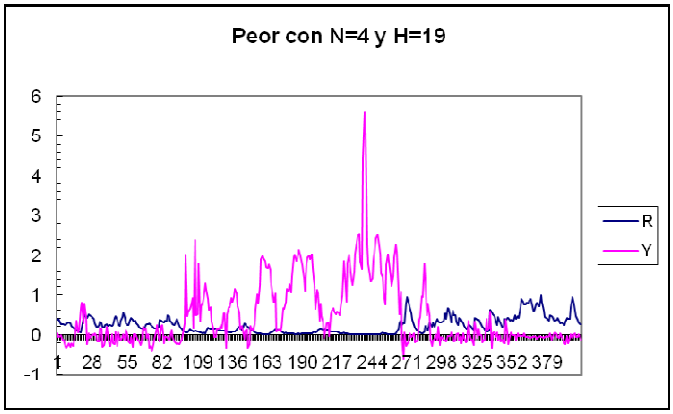

Gráfico 5.39: con N = 4 y H= 19

Tabla 5.20: Parámetros de validación de la séptima prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
5	13	0,297323	0,088400	0,698683	0,370878	-0,40998
6	16	0,642174	0,412387	0,760626	0,116462	0,568159
4	19	0,993767	0,987572	0,851525	0,140077	0,745186

Como se puede apreciar por los resultados entregados en la séptima prueba con PSO Clásico, el Gráfico 5.37 da un intento por arrojar buenos resultados gráficamente, lo que se justifica con los buenos estimadores obtenidos en la tabla 5.20 con un RMSE=0,29. Si bien es cierto, el mejor de estos resultados no da cuenta claramente que se siga una tendencia clara en comparación a la data original, manteniendo solamente los valores de predicción dentro del rango esperado, quizás esto justificado por los bajos valores del coeficiente de eficiencia R^2 en todos los casos.

5.1.7.2 PSO Max-Min:

Ahora los resultados de la variante Max-Min, aplicando VMax=0,031.

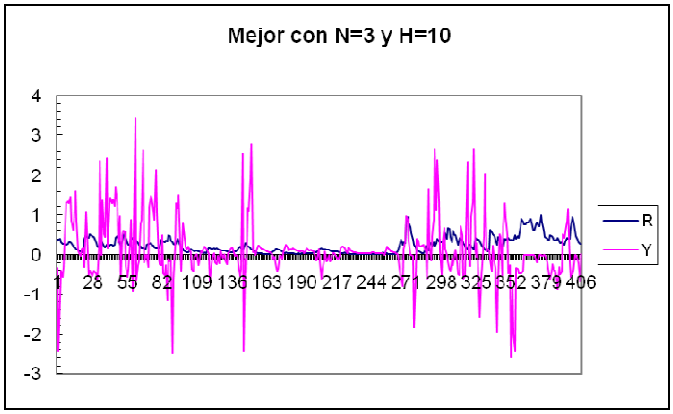

Gráfico 5.40: con N = 3 y H = 10

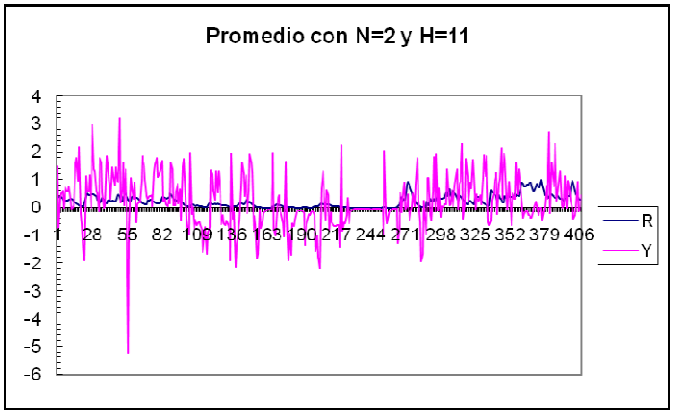

Gráfico 5.41: con N = 2 y H = 11

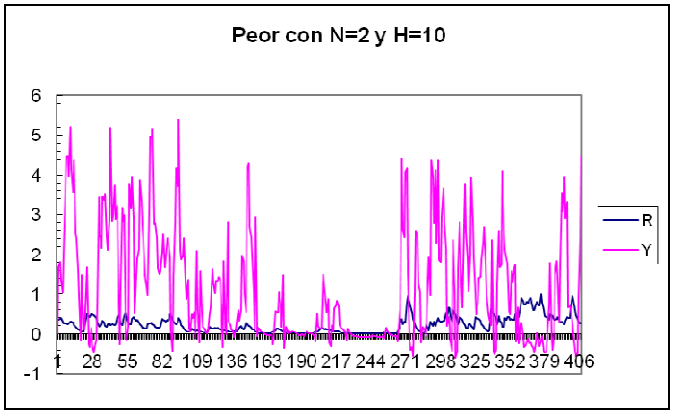

Gráfico 5.42: con N = 2 y H= 10

Tabla 5.21: Parámetros de validación de la séptima prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
3	10	0,796461	0,634350	0,847030	0,218973	0,097895
2	11	0,909034	0,826342	0,973616	0,136251	0,133376
2	10	1,660136	2,756051	0,994592	0,148877	0,636847

En los gráficos 5.40, 5.41 y 5.42 de la séptima prueba aplicando PSO Max-Min se aprecia una posible tendencia en seguir la data original, sin embargo se alejan demasiado en los valores estimados que arrojan en la tabla 5.21, esto claramente justificado por los valores de RMSE y R^2 que poseen todos los modelos estudiados.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.37 con N=5 y H=13 y el gráfico 5.40 con N=3 y H=10.

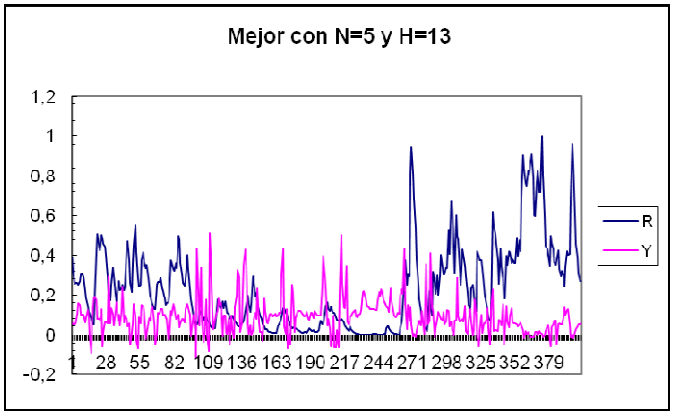

Gráfico 5.37: con N = 5 y H= 13 - Clásico

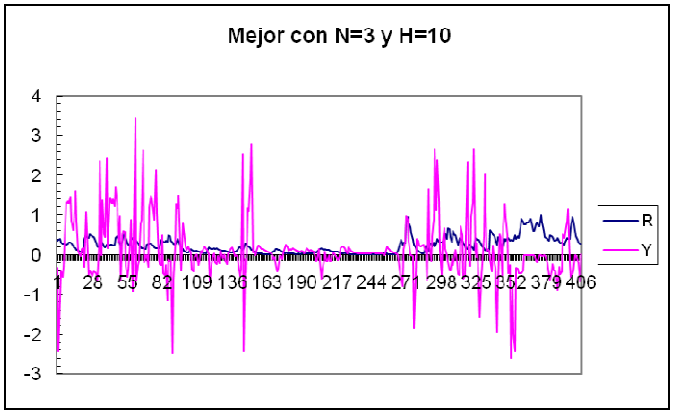

Gráfico 5.40: con N = 3 y H = 10 - Max-Min

La configuración con el algoritmo clásico a simple vista en el gráfico 5.37 es mejor, se justifica gráficamente y por los parámetros de validación obtenidos, ya que se mueven dentro de los valores válidos de la data, ahora bien, aún cabe por mejorar estos ajustes a la data real, ya que se mantienen enormes diferencias con lo estimado.

5.1.8 Prueba 8

En la octava ronda de pruebas se mantiene en 40 el número de P partículas y se aumenta en 500 el número de las K iteraciones

Tabla 5.22: Parámetros de la octava prueba

P	= 40	(partículas)
K	= 2000	(iteraciones)
VMax	= 0,035	

5.1.8.1 PSO Clásico:

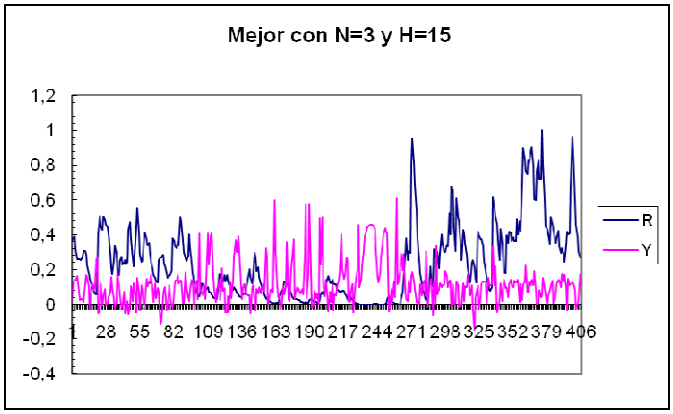

Gráfico 5.43: con N = 3 y H = 15

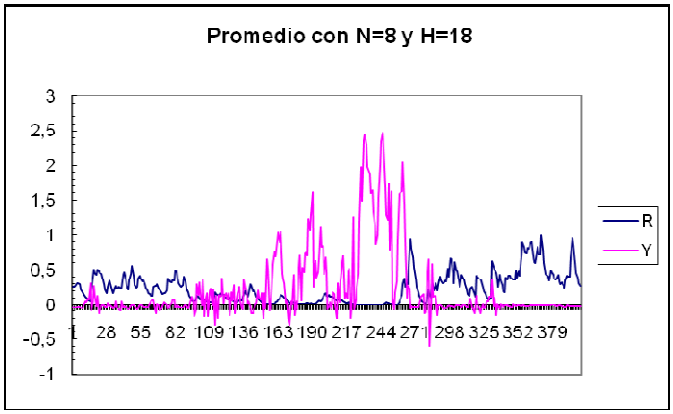

Gráfico 5.44: con N = 8 y H = 18

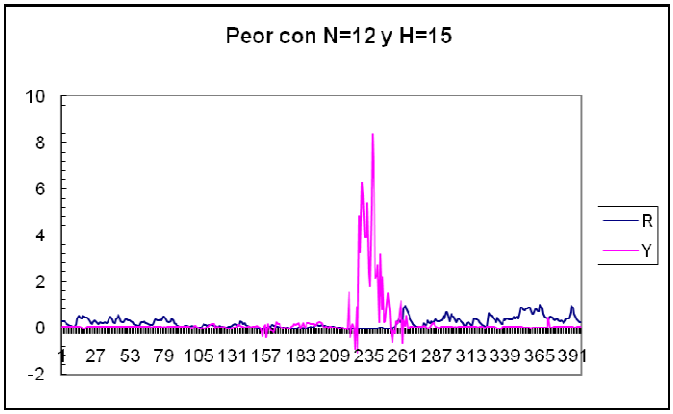

Gráfico 5.45: con N = 12 y H = 15

Tabla 5.23: Parámetros de validación de la octava prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
3	15	0,299985	0,089991	0,892750	0,521245	-0,602452
8	18	0,630874	0,398002	1,992136	0,287957	1,036295
12	15	0,980626	0,961627	24,134074	0,067548	1,796609

En la octava prueba aplicando PSO Clásico, tenemos a los gráficos 5.43, 5.44 y 5.45, en donde se puede destacar el primer gráfico de este grupo de pruebas ya que da mínimas señales de la tendencia en los datos estimados, justificados por los valores de RMSE=0,29 y R^2=52% descritos en la tabla 5.23, ya que los demás gráficos poseen métricas demasiado malas e incluso se escapan de los valores esperados comúnmente.

5.1.8.2 PSO Max-Min:

Ahora los resultados de la variante Max-Min, aplicando VMax=0,035.

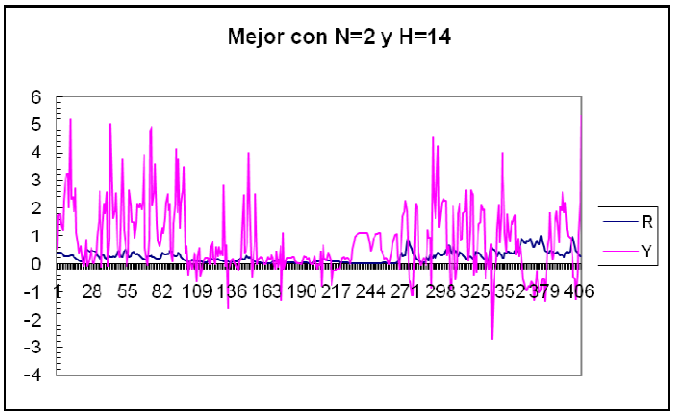

Gráfico 5.46: con N = 2 y H = 14

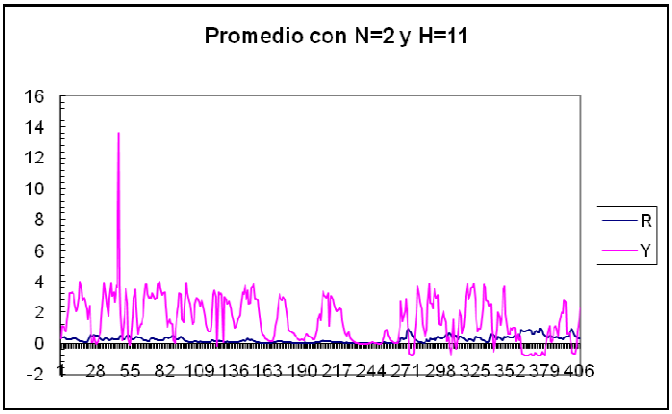

Gráfico 5.47: con N = 2 y H = 11

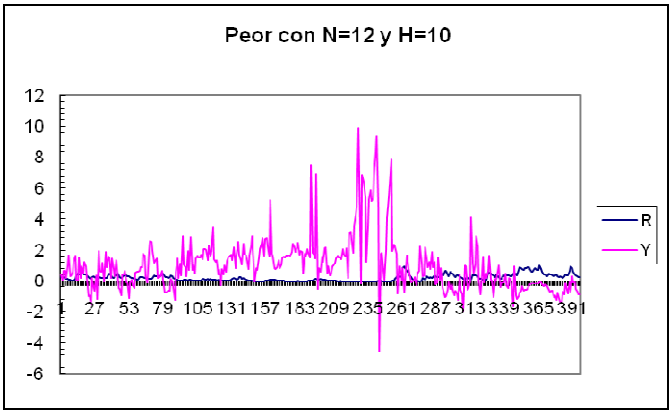

Gráfico 5.48: con N = 12 y H = 10

Tabla 5.24: Parámetros de validación de la octava prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
2	14	1,324468	1,754215	1,056978	0,050086	0,519267
2	11	1,983071	3,932570	21,614607	0,031053	0,810893
12	10	1,994904	3,979641	37,488461	0,005686	1,808912

Hasta el momento, como se puede apreciar en los resultados arrojados por la octava prueba con Max-Min, a medida que crecen los parámetros de P y K en las pruebas, se aprecia una mayor dispersión en los datos como es el caso de los gráficos 5.46, 5.47 y 5.48 que arrojan valores fuera de los rangos esperados, parámetros de validación mucho más elevados y alejados de lo óptimo según la tabla 5.24.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.43 con N=3 y H=15 y el gráfico 5.46 con N=2 y H=14.

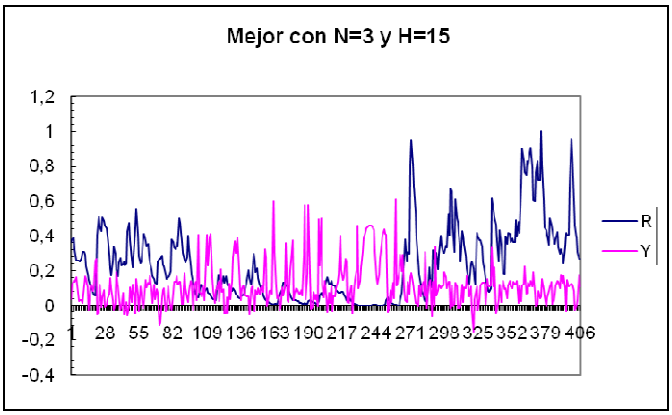

Gráfico 5.43: con N = 3 y H = 15 - Clásico

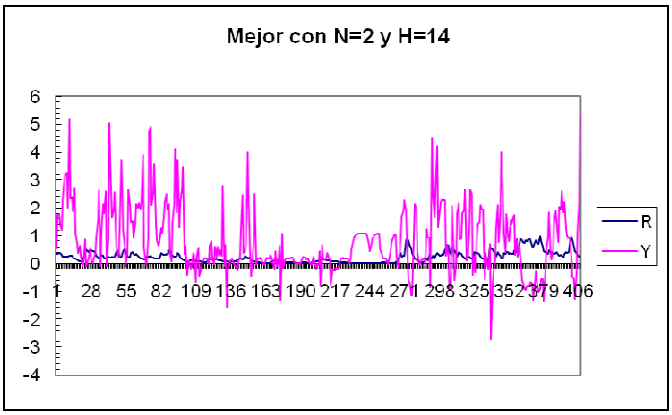

Gráfico 5.46: con N = 2 y H = 14 - Max-Min

Al observar solamente los valores de los estimadores en las tablas 5.23 y 5.24, claramente es superior el modelo clásico al modelo Max-Min, lo cual se confirma al representarlos gráficamente uno al lado del otro en los gráficos 5.43 y 5.46, aunque en ninguno de los casos se puede apreciar un modelo predictivo y que demuestre al menos la tendencia en los datos.

5.1.9 Prueba 9

En la novena y última ronda de pruebas se conjugan los máximos valores probados de P partículas y de K iteraciones según la tabla 5.25.

Tabla 5.25: Parámetros de la novena prueba

P	= 40	(partículas)
K	= 3000	(iteraciones)
VMax	= 0,034	

5.1.9.1 PSO Clásico:

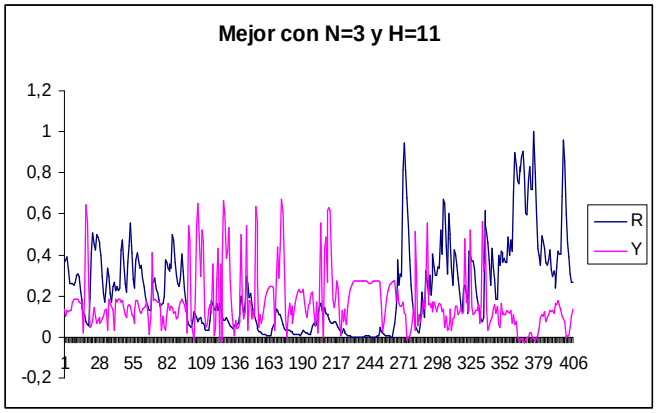

Gráfico 5.49: con N = 3 y H= 11

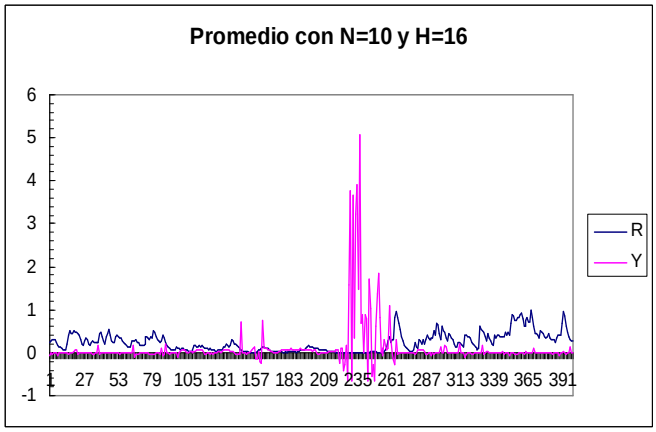

Gráfico 5.50: con N = 10 y H = 16

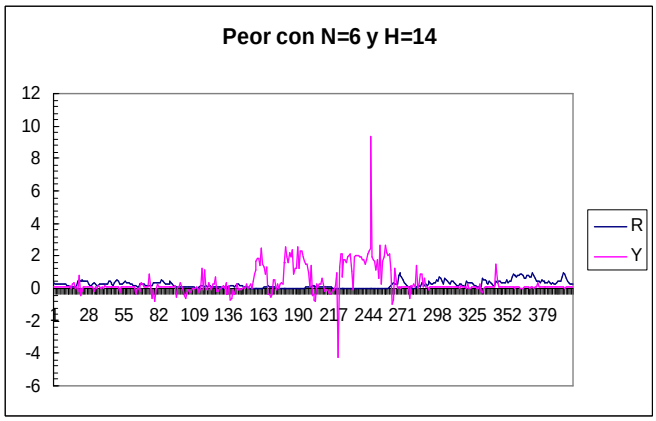

Gráfico 5.51: con N = 6 y H = 14

Tabla 5.26: Parámetros de validación de la novena prueba con PSO Clásico

N	H	RMSE	MSE	MAPE	R^2	AIC
3	11	0,303296	0,091988	0,630548	0,528012	-0,711144
10	16	0,594344	0,353244	10,913852	0,063306	1,152086
6	14	0,97806	0,956601	36,014833	0,086880	0,814478

Como se puede apreciar, la mejor configuración encontrada en la novena prueba con PSO clásico (Gráfico 5.49) sigue solamente la tendencia de la data y empeora bruscamente con los demás resultados encontrados (gráficos 5.50 y 5.51) al aumentar los valores de N y H neuronas.

5.1.9.2 PSO Max-Min:

Ahora los resultados de la variante Max-Min, aplicando VMax=0,034.

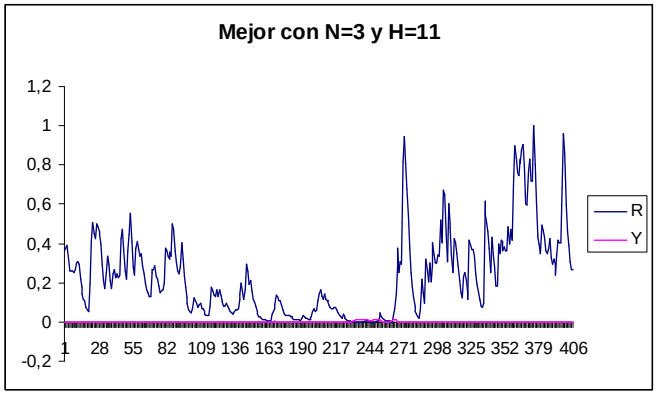

Gráfico 5.52: con N = 3 y H = 11

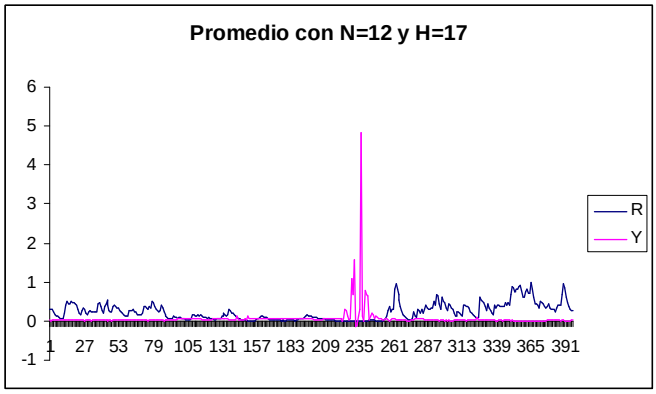

Gráfico 5.53: con N = 12 y H = 17

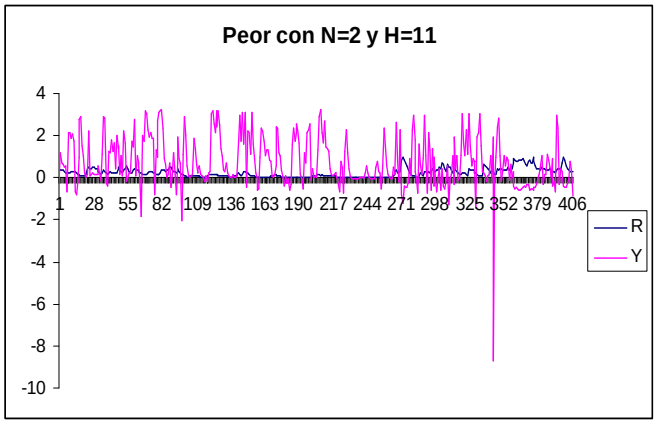

Gráfico 5.54:con N = 2 y H = 11

Tabla 5.27: Parámetros de validación de la novena prueba con PSO Max-Min

N	H	RMSE	MSE	MAPE	R^2	AIC
3	11	0,318233	0,101272	3,114900	0,218674	-0,669386
12	17	0,408635	0,166982	15,671920	0,059039	1,278087
2	11	1,327953	1,763459	21,619703	0,070370	0,462581

La versión Max-Min con la máxima configuración en la novena prueba presenta resultados bastante negativos, ya que o se aleja enormemente de los valores esperados (gráfico 5.54) o se acerca demasiado a valores prácticamente iguales a cero (gráfico 5.52), que dan cuenta de estimadores al parecer muy buenos de RMSE según la tabla 5.27 y que gráficamente demuestran lo contrario. A estas alturas, ya se puede adelantar como posible conclusión que al aumentar la topología los resultados ya no mejoran e incluso comienzan a empeorar, entonces nuestra topología ideal será menor a esta, ya que los últimos resultados han mantenido la tendencia de alejarse a los valores esperados.

Al comparar los mejores resultados arrojados en ambas versiones del algoritmo PSO tenemos los datos arrojados por el gráfico 5.49 con N=3 y H=11 y el gráfico 5.52 con N=3 y H=11.

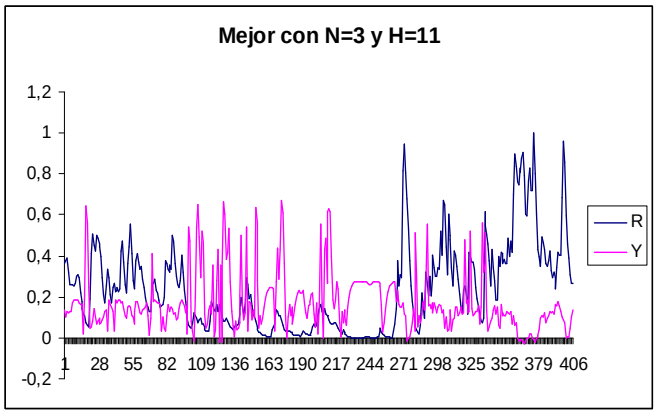

Gráfico 5.49: con N = 3 y H= 11 - Clásico

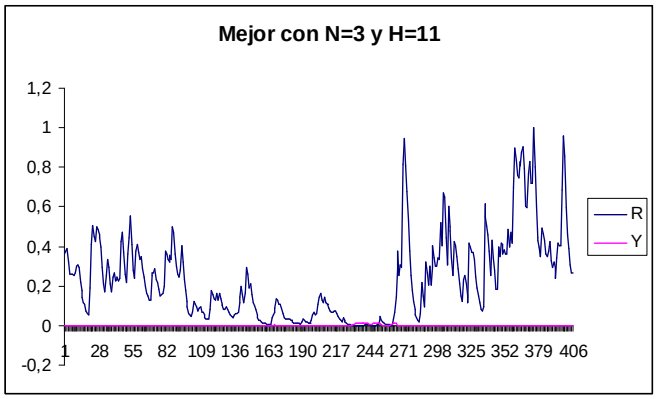

Gráfico 5.52: con N = 3 y H = 11 - Max-Min

En este último grupo de pruebas se puede apreciar en el gráfico 5.49 y 5.52, que el PSO Clásico encuentra una mejor aproximación a la solución deseada y además esta situación ocurre en la mayoría de las pruebas realizadas al ser comparados los resultados obtenidos de la variante Max-Min, salvo en pocas ocasiones en que la variante Max-Min mejora los resultados encontrados por la versión clásica.

Luego de terminados el conjunto de las distintas pruebas, podemos notar que la topología por más que aumente no mejora los resultados ya encontrados e incluso con topologías mas básicas y con menores parámetros se encuentran mejores resultados.

Por otra parte, se puede deducir que los resultados entregados por el PSO Max-Min fueron inferiores en casi todas las ocasiones al PSO Clásico, por lo cual se puede suponer que para este tipo de problemas acotar la velocidad no trae mayores beneficios o que simplemente el criterio de aplicación y selección del VMax no fue el adecuado para mejorar los resultados de gran manera. Sin embargo el mejor resultado lo encontramos aplicando la versión Max-Min y que por lo general estas mejoras se aprecian cuando VMax se acerca mas al valor 0,05.

Como resultado de las distintas pruebas realizadas se puede destacar que los valores estimados por las distintas pruebas suelen tener problemas cuando la data real es muy extrema, presentando valores muy altos o muy bajos, con lo que la data estimada se aleja obteniendo valores aún mas extremos y solo marcando la tendencia en los datos.

5.3 Elección del mejor modelo

Por los resultados obtenidos en las distintas simulaciones y apoyándonos en los parámetros de validación elegidos (RMSE, MSE, MAPE, R^2 y AIC), podemos decir que la configuración más confiable y elegida con esta investigación fue aquella obtenida con la versión de PSO Max-Min del algoritmo de entrenamiento, con N= 5 neuronas en la capa de entrada, H= 10 neuronas en la capa oculta, con K= 3000 iteraciones, utilizando P= 20 partículas y acotando la velocidad a VMax=0,050.

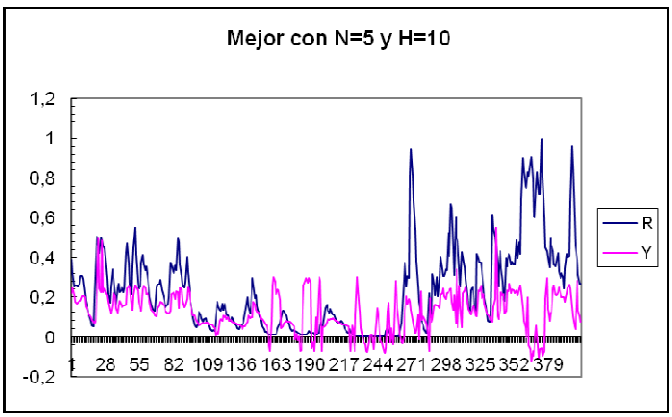

Gráfico 5.55: Modelo Elegido

Tabla 5.28: Parámetros de validación del Modelo Elegido

N	H	RMSE	MSE	MAPE	R^2	AIC
5	10	0,242397	0,058756	0,697385	0,712833	-0,73589

Cabe destacar que las mejores soluciones encontradas e incluso la mejor de todas ellas, solo aportan la tendencia que sigue la data y no son un reflejo exacto de lo que está ocurriendo en el tiempo, es cierto que por algunos tramos la data real y estimada se comportan de igual manera y se solapan en algunos tramos, pero aun existen altas diferencias en los puntos altos de la data, en donde la data estimada sigue la tendencia, pero no alcanza a igualarla. Es por esto que se puede concluir que el modelo de pronósticos obtenido no es exacto y solo marca la tendencia que seguirá la data estimada, no proporcionando un modelo de pronóstico muy eficiente.

¿Cómo mejorar estos resultados? Una tentativa de solución es refinar el modelo suavizando la data de entrenamiento y prueba, disminuyendo los puntos altos de la data, los cuales deben corresponder a outlayers causados por factores ambientales aislados que disparan el comportamiento de la data haciéndola difícil de predecir con exactitud. Con el suavizado se ha demostrado en otros estudios [3] que el ajuste de la data real y estimada aumenta considerablemente al aplicar distintos suavizados, mejorando los parámetros de validación utilizados. Otra forma de mejorar estos resultados es utilizar otro criterio en la elección del valor VMax en la variante PSO Max-Min, que en nuestro caso arrojo resultados no muy satisfactorios y muchas veces restringiendo de mayor manera el movimiento de las partículas.

Con este modelo de pronósticos encontrado se puede esperar manejar la tendencia de la data de los volúmenes de Anchovetas para poder entender a grandes rasgos el futuro valor en un mes próximo del volumen a capturar, siendo capaz de predecir este volumen con un cierto error conocido.

5.3 Mejoras al Modelo

Como mejora al modelo ya obtenido se propone suavizar la data para que al ingresar esta a la red el comportamiento obtenido sea mejor frente a posibles saltos en la data, probablemente provocados por factores ambientales e inconstantes.

La técnica utilizada en esta oportunidad es el Suavizado Promedio Móvil, a través de la cual se intenta eliminar o minimizar estos outlayers en la data, para que no posea grandes diferencias entre ciertos períodos o ventanas de tiempo.

5.3.1 Suavizado Promedio Móvil

El Suavizado Promedio Móvil es una técnica que se utiliza para suavizar los datos manteniendo la dirección y tendencia real de estos.

$$\tilde{x}(t) = \frac{\sum_{i=1}^{T} x(t-i)}{T} \qquad \textbf{(5.7)}$$

El valor $\tilde{x}(t)$ de la ecuación 5.7 es llamado el promedio móvil obtenido con los *T* períodos anteriores al real *x(t),* es así como se calcula un nuevo dato suavizado corriendo esta ventana de *T* períodos hasta suavizar toda la data.

Ahora para obtener una posible mejora en el modelo obtenido como el mejor de los estudiados, se hará ingresar los datos suavizados con un tamaño de ventana *T*=5, para posteriormente comparar nuestros resultados estimados con la data real y no con la suavizada.

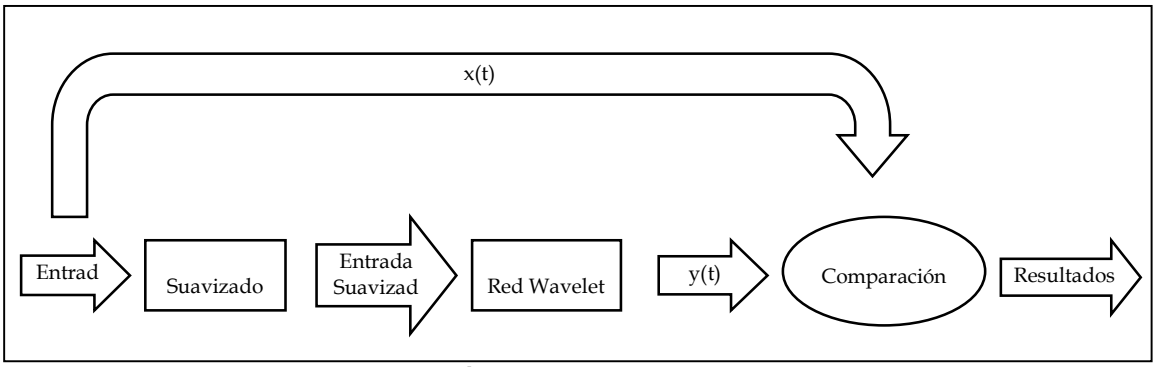

Ilustración 5.1: Esquema del suavizado

La Ilustración 5.1 nos muestra esquemáticamente como fue utilizado el suavizado en la mejora del modelo obtenido, en donde la validación del modelo se hace comparando el valor estimado con el real y no con el valor suavizado.

5.3.2 Evaluación de Resultados

El modelo elegido como el mejor por los parámetros de validación encontrados, descrito en la sección 5.2 de este capitulo, mejora parcialmente sus resultados al utilizar una data suavizada en el ingreso a la red neuronal Wavelet.

Tabla 5.29: Comparación del modelo con suavizado y sin suavizado

	RMSE	**MAPE**	**R^2**	**AIC**
Sin Suavizado	0,242397	0,697385	0,712833	-0,73589
Con suavizado	0,186298	0,654274	0,758140	-0,915024

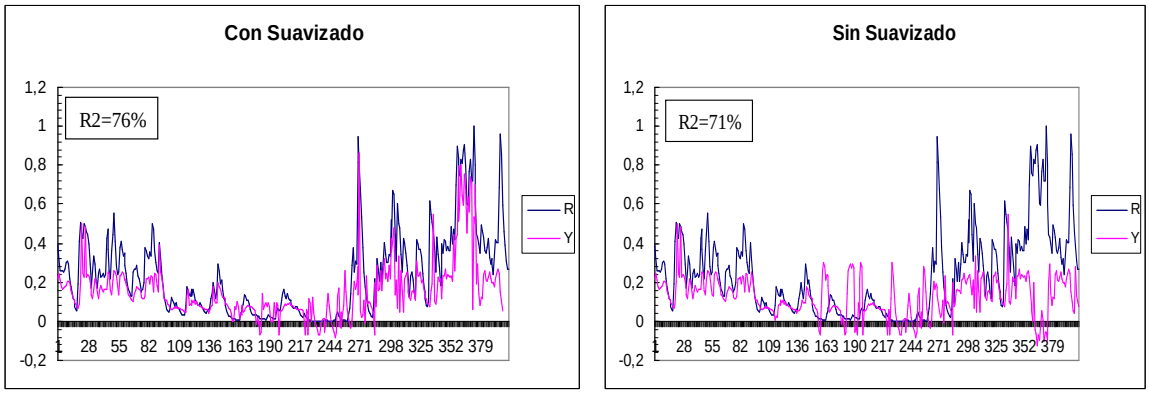

Gráfico 5.56: Comparación gráfica del modelo elegido con suavizado y sin suavizado

Como se puede apreciar en la tabla 5.29 los parámetros de validación mejoran al aplicar un suavizado en la data que se hace pasar por el modelo obtenido, en especial el valor de R^2= 71% para el modelo sin suavizado, que posteriormente se obtiene un R^2=76% para el modelo con suavizado.

$$GANANCIA(\%) = \frac{SS(\%) - CS(\%)}{SS(\%)} \qquad \textbf{(5.8)}$$

La ecuación 5.8 nos indica porcentualmente la ganancia entre una métrica sin suavizado (*SS*) y una métrica con suavizado (*CS*). Esta ganancia nos indica que tan mejorado ha sido el modelo sin suavizado en comparación al modelo con suavizado. La ganancia para el valor de R^2 es de un 7%, para el valor de RMSE la ganancia es de un 25% y para el MAPE la ganancia es de un 5%.

Gráficamente se puede apreciar una mejora a simple vista del ajuste de la data real con la data estimada (Gráfico 5.56), en especial en los valores extremos de la data, en donde el comportamiento de la red se comportaba de forma muy irregular y que ahora gracias al suavizado al menos mantiene la tendencia y dirección de los datos reales.

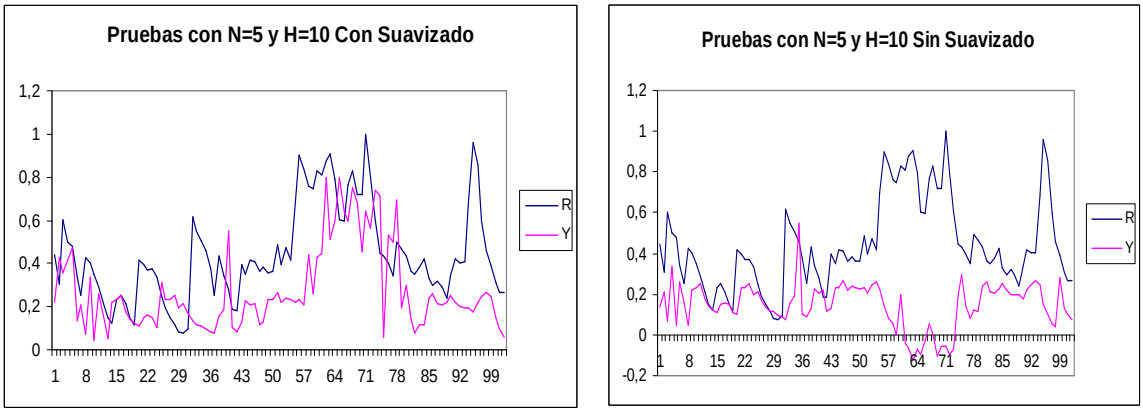

Gráfico 5.57: Comparación gráfica de la etapa de pruebas con ambos modelos.

El gráfico 5.57 nos muestra el comportamiento de los modelos con suavizado y sin suavizado, y como se puede apreciar el modelo con suavizado mejora su comportamiento predictivo con la data superior y se acerca mas a la tendencia real de los datos.

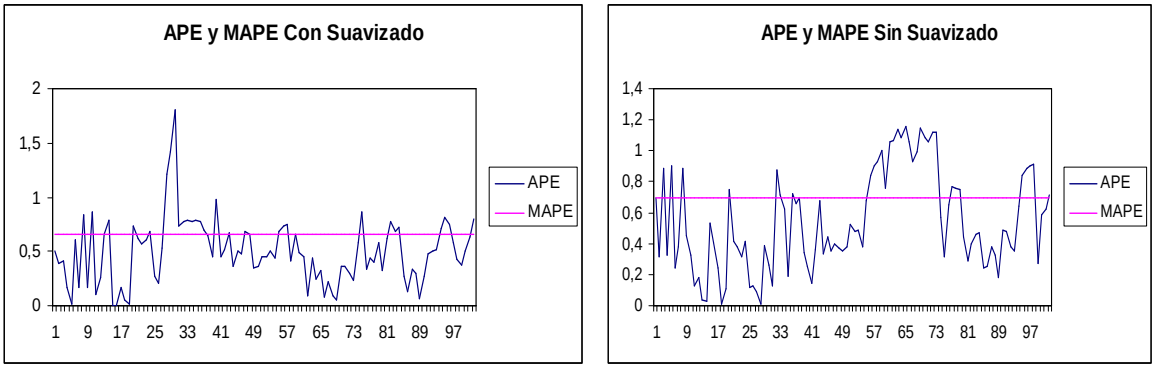

Gráfico 5.58: Comparación gráfica de los valores del APE y MAPE, con suavizado y sin suavizado

En el modelo sin suavizado obtenemos un MAPE=69% y los valores de APE se encuentran en su mayoría bajo este valor según el gráfico 5.58. Por su parte el modelo con suavizado obtiene un MAPE=65% y se mantiene la tendencia que los valores de APE están bajo este valor, sin embargo, existe un punto en el cual la data se encuentra muy alejada de la real y se alza sobre el valor del MAPE manifestando una clara diferencia entre los datos estimados y los reales.

Finalmente se puede afirmar que el modelo con suavizado mejora alrededor de un 5% su eficiencia con respecto a la diferencia entre los datos reales y los estimados, reduciendo así el error encontrado mejorando el modelo predictivo elegido en las pruebas realizadas exhaustivamente.

Si bien es cierto los resultados mejoraron al realizar un suavizado en la data, no se puede asegurar que el modelo encontrado sea un modelo predictivo muy eficiente, ya que la diferencia entre los datos reales y los estimados aun son muy grandes en algunos rangos de datos, como se puede apreciar en el gráfico 5.58 del APE, ya que esta pequeña diferencia que se ve gráficamente puede ser engañosa al significar una gran cantidad de toneladas de Anchovetas no consideradas en el problema real a solucionar, desaprovechando o sobreexplotando los recursos por la deficiencia del modelo a pronosticar con una menor exactitud.

Capítulo 6
Conclusiones

Muchas veces la utilización de redes neuronales se visualiza como un método complicado de solución, pero a medida que se desarrolla y profundiza en un problema a resolver se puede comprobar que la utilización de técnicas como las redes neuronales no están lejos de ser usadas para cualquier propósito, ya que basta con comprender y poder modelar correctamente el problema en cuestión para encontrar la aplicación correcta en el problema. Por otra parte la incorporación de nuevas técnicas en la calibración de las redes neuronales y en la utilización de nuevos métodos en el manejo de la data, han mejorado considerablemente su confiabilidad y por ende ha masificado su uso a distintas áreas de estudio. Es así como la presente investigación ha logrado definir la arquitectura a utilizar en una red neuronal Wavelet, además de haber podido definir y comprender con claridad el algoritmo de optimización de enjambre de partículas y su real dimensión al ser aplicado en el entrenamiento de una red neuronal.

Con la investigación se logró comprender por completo el problema en cuestión, relacionado con la optimización en la predicción de volúmenes de Anchovetas y por otro lado manejar las relaciones de las distintas técnicas a utilizar: redes neuronales, Wavelet y PSO, haciendo la correcta intersección entre estas técnicas.

A pesar de todos los beneficios descritos en la literatura estudiada y que se pudieron apreciar a lo largo de los capítulos desarrollados con la investigación de las distintas técnicas más utilizadas, las redes neuronales Wavelet no presentan una real solución a problemas de pronósticos de esta naturaleza, puesto que solo entregan un modelo de tendencia y no un modelo de predicción lo suficientemente confiable.

PSO es un algoritmo evolutivo independiente de la derivada, por lo cual puede manejar datas tan discontinuas y dinámicas como la estudiada, que reduce el riesgo de llegar a soluciones locales pero que tiene menor velocidad de convergencia en comparación a otros algoritmos de entrenamiento, es quizás esta característica lo que nos hizo llegar solo a una aproximación de la solución esperada, la cual debería tener mejores valores en los parámetros de validación para ser aceptable como modelo predictivo eficiente. Además, en el mismo sentido, nuestro modelo elegido fue encontrado con el máximo de iteraciones a probar K = 3000, lo cual nos da un indicio de que la velocidad de convergencia es lenta con este algoritmo.

Los modelos de red neuronal Wavelet en todas las pruebas realizadas presentaron claros problemas con los valores extremos de la data, obteniéndose valores aun más extremos y muchas veces solo marcando la tendencia en los datos estimados con diferencias muy grandes con el dato real esperado. Esta situación pudo ser mejorada, en

cierta forma, al suavizar la data y eliminar algunos outlayers de la data, presuntamente creados por efectos ambientales impredecibles por el modelo.

Un punto a destacar es el hecho que los mejores resultados se encontraron con configuraciones que poseían un número pequeño de parámetros y que al aumentar estos parámetros muchas veces los resultados empeoraban. Por lo tanto para obtener resultados más óptimos en el estudio de redes neuronales Wavelets, este debiese centrarse en configuraciones con combinaciones pequeñas de los parámetros utilizados, ya que modelos demasiado complejos no entregaron buenos resultados.

Los parámetros o métricas de validación fueron utilizadas para seleccionar el mejor, el intermedio y el peor modelo que arrojaba la configuración estudiada en un grupo de pruebas, es así como el criterio inicial fue encontrar el menor RMSE, posteriormente validarlo con su valor de R^2 y frente a posibles disyuntivas en la elección del modelo se consideró el menor AIC para ordenar los modelos.

Se propone la siguiente configuración y modelo de pronósticos, descrito en la tabla 6.1, para ser evaluado y refinado con nuevas técnicas para reducir la varianza en los datos. El mejor modelo fue aquel obtenido con la versión de PSO Max-Min del algoritmo de entrenamiento de la red neuronal Wavelet, 5 neuronas en la capa de entrada, 10 neuronas en la capa oculta, con 3000 iteraciones, utilizando 20 partículas y acotando la velocidad a 0,050, obteniéndose sin suavizado una eficiencia de R^2=71% y con suavizado un R^2=75%.

Tabla 6.1: Configuración propuesta.

N	= 5	(neuronas de entrada)
H	= 10	(neuronas capa oculta)
K	= 3000	(iteraciones)
P	= 20	(partículas)
VMax	= 0,05	(velocidad máxima)

Finalmente se puede decir que la variante PSO Max-Min en comparación al PSO Clásico no presentó grandes diferencias e incluso arrojó peores resultados en la mayoría de las ocasiones y sólo fue considerado como una versión válida del algoritmo al encontrarse después de varios intentos un modelo mejorado de la versión clásica con Max-Min, el cual finalmente resultó ser el modelo elegido con la investigación. Este resultado puede ser producto del criterio de acotación de velocidad que tomaba distinto valor para cada prueba dependiendo de las velocidades del PSO Clásico, puesto que los mejores resultados se vieron alrededor de VMax=0,050 y en los demás casos el acotamiento no mejoró el movimiento de las partículas. Es por esto que VMax=0,050 se nos presenta como un valor a ser estudiado y probado para el acotamiento de las velocidades de partícula, para que estas no generen una explosión del algoritmo generando valores fuera del campo de solución.

Referencias

[1] Amelia A. Catálfamo, *Redes Neuronales Aplicadas a Señales Sonoras Obtenidas de un Sonar,* Trabajo Final Especialidad en Ingeniería en Sistemas Expertos, ABTI 2006.

[2] Alfonso Pitarque, Juan Francisco Roy y Juan Carlos Ruiz, *Redes neurales vs modelos estadísticos: Simulaciones sobre tareas de predicción y Clasificación*, Universidad de Valencia 1998.

[3] *Global Aquaculture production highlights & estimated compound aquafeed use in 2005,* Extracto y adaptacion de Tacon, 2007, Aquafeed Int.

[4] Rodrigo Cristian Pérez Galleguillos, *Predicción de captura de Anchovetas utilizando Redes Neuronales,* Pontificia Universidad Católica de Valparaíso, 2007.

[5] Luis Cubillos, Patricia Ruiz, Gabriel Claramunt, Santiago Gacitúa, Sergio Núñez, Leonardo Castro, Katty Riquelme, Carolina Alarcón, Ciro Oyarzún, Aquiles Sepúlveda, *Spawning, daily egg production, and spawning stock biomass estimation for common sardine (Strangomera bentincki) and anchovy (Engraulis ringens) off central southern Chile in 2002,* Universidad de Concepción, Instituto de Investigación Pesquera Talcahuano, Universidad Arturo Prat Iquique, 2002.

[6] Winston Palma, Ruben Escribano, Sergio Rosales, *Modeling study of seasonal and inter-annual variability of circulation in the coastal upwelling site of the El Loa River off northern Chile,* Universidad de Concepción, Universidad Arturo Prat Iquique, Center for Oceanographic Research in the South Eastern Pacific (COPAS), 2006

[7] Jorge Valdés, Luc Ortlieb, Dimitri Gutiérrez, Luis Marinovic, Gabriel Vargas, Abdel Sifeddine, *250 years of sardine and anchovy scale deposition record in Mejillones Bay, northern Chile,* Universidad de Antofagasta, Instituto del Mar de Perú, Universidad de Chile, Universidade Federale Fluminense Brazil, 2008

[8] Gordon Swartzman, Arnaud Bertrand, Mariano Gutiérrez, Sophie Bertrand, Luis Vasquez, *The relationship of anchovy and sardine to water masses in the Peruvian Humboldt Current System from 1983 to 2005,* University of Washington, Instituto del Mar del Perú (IMARPE), 2008

[9] McCulloch and Pitts, *A logical calculus of the ideas immanent in nervous activity*, 1943.

[10] John Von Neumann, *The Computer and the Brain,* Yale University Press, 1958.

[11] Marvin Minsky, Seymour Papert, *Perceptrons,* MIT Press, 1969.

[12] Frank Rosenblatt, *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*, Spartan Books, 1962.

[13] S. Y. Kung, *Digital Neural Networks,* PTR Prentice Hall Inc., 1993.

[14] Fernando Izaurieta, Carlos Saavedra, *Redes Neuronales Artificiales,* Universidad de Concepción, 2007.

[15] Emiliano Aldabas-Rubira, *Introducción al reconocimiento de patrones mediante redes neuronales,* UPC-Campus Terrassa-DEE-EUETIT Colom Barcelona.

[16] Haar A., *Zur Theorie der orthogonalen Funktionensysteme,* Mathematische Annalen, pp 331-371, 1910.

[17] J. Peña, *Espectrogramas Basados en Funciones Wavelets,* Museo Nacional de Ciencias Naturales Madrid.

[18] Hugo Adrián García Elías, José Federico Ramírez Cruz, *Método Basado en Redes Neuronales Wavelet para Eliminar Ruido en Espectros Estelares,* Instituto Tecnológico de Apizaco México, 2007.

[19] Josué Andrés Durán Ledesma, *Optimización con Enjambre de Partículas para Reducción de Distorsión No Lineal en Sistemas OFDM*, Pontificia Universidad Católica de Valparaíso, 2007.

[20] James Kennedy, *Particle Swarm Optimization,* Purdue School of Engineering and Technology Indianapolis, 1995.

[21] José Manuel García Nieto, *Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos*, 2006.

[22] Juan Carlos Gutiérrez Estrada, Claudio Silva, Eleuterio Yáñez, Nibaldo Rodríguez, Inmaculada Pulido Calvo, *Monthly catch forecasting of anchovy Engraulis ringens in the north area of Chile: Non-linear univariate approach*, Universidad de Huelva Spain, Pontificia Universidad Católica de Valparaíso Chile, 2007.

[23] L.-K. Shark and C. Yu, *Design of matched wavelets based on generalized Mexican-hat function,* University of Central Lancashire Preston, 2004.

[24] Nadia Nedjah y Luiza de Macedo Mourelle, *Swarm Intelligent Systems*, State University of Rio de Janeiro, 2006.

[25] Burnham, K.P. and Anderson, D.R., *Model selection and inference: a practical information-theoretic approach,* Springer-Verlag New York, 2002.

[26] Chapman and Hall, *Statistical evidence: a likelihood paradigm*, Royall R.M. New York, 1997.

[27] Diego Andina, *Computational Intelligence for Engineering and Manufacturing*, Pham, Duc Truong (Eds.), 2007.

[28] J. Kennedy, R.C. Eberhart, *Swarm Intelligence,* Morgan Kaufmann Publishers San Francisco, 2001.

Anexo

Código PSO Clásico:

```
/*****************************************************************************/
/*                    PSO CLASICO                                            * /
/*                   Entrenamiento                                            */
/* Autor: Manuel Mena P.                                                      */
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <string.h>

#define RAND_MAX 1000000   // maximo aleatorio
#define XnMax 1            // maximo del espacio de busqueda
#define XnMin 0            // minimo en el espacio de busqueda

//Definicion de tamaños
int P;          // numero de particulas
int N;          // numero de neuronas de entrada
int H;          // numero de neuronas capa oculta
int K;          // numero de iteraciones
float C1;       // valor cognitivo
float C2;       // valor social

float RMSE;
float MAPE;
float R2;
float AIC;

float VMax;
float VMin;
int v1;
int v2;

//Definicion de las capas de la red
float X[20];    // entradas de la red
float O[20];    //capa oculta
float Y;        //salida
float R;        //valor real


//Estructura de configuracion
struct config {
    float Wa[20][20];   //valor de dilatacion
    float Wt[20][20];   //valor de traslacion
    float Ww[20];       //peso de salida
};
```

```
struct config Xi[50];       //posicion de particula
struct config Pi[50];       //mejor posicion
struct config G;            //mejor posicion global
struct config Vi[50];       // velocidad de las particulas

FILE *fp;

/*****************************************************************************/
/* Funcion Forward                                                           */
/*****************************************************************************/
void forward(struct config C){

    int i,j;
    //capa oculta de la red
    for(j=0;j<H;j++){
        O[j]=0;
        for(i=0;i<N;i++){
            O[j] = O[j] + (1.0/sqrt(C.Wa[i][j])) * ( 1.0 - pow(((X[i]-C.Wt[i][j])/C.Wa[i][j]),2) ) * exp(-
                    (pow(((X[i]-C.Wt[i][j])/C.Wa[i][j]),2))/2.0); //wavelet
        }
    }

    //capa de salida, calculo del valor de Y estimado
    Y=0;
    for(j=0;j<H;j++)
        Y = Y + O[j]*C.Ww[j];
}

/*****************************************************************************/
/* Funcion Fitness                                                           */
/*****************************************************************************/
float fitness(struct config C){

    int i;
    float fit;

    //lectura de entradas a la red
    for(i=0;i<N;i++){
      if(feof(fp)){
        fclose(fp);
        fp = fopen("dataanchovy.csv", "r");
        if (fp == NULL)
          printf ("Abrir archivo data para lectura1\t..... ERROR\n");
        i=0;
        fscanf( fp, "%f;", &X[i]);
      }
      else
        fscanf( fp, "%f;", &X[i]);
    }

    //lectura del valor real a comparar
    if(feof(fp)){
      fclose(fp);
      fp = fopen("dataanchovy.csv", "r");
      if (fp == NULL)
```

```
            printf ("Abrir archivo data para lectura2\t..... ERROR\n");
        for(i=0;i<N;i++)
          fscanf( fp, "%f;", &X[i]);
        fscanf( fp, "%f;", &R);
        }
      else
        fscanf( fp, "%f;", &R);

      forward(C);          //forward de la red con la configuracion

      fit = fabs(Y - R);   //calculo del valor de fitness
      return fit;
}


/*****************************************************************************/
/* Funcion Inicializar Variables del PSO                                     */
/*****************************************************************************/
void inicializarPSO(void){
      int i,j,l;
      float k;

      //inicializando Xi, Pi aleatoriamente y Vi a cero
      for(i=0;i<P; i++){

          for(j=0;j<N;j++){                //primero Wa
              for(l=0;l<H;l++){
                  k = rand() % 100000;
                  Xi[i].Wa[j][l] = k/100000;
                  k = rand() % 100000;
                  Pi[i].Wa[j][l] = k/100000;
                  k = rand() % 100000;
                  Vi[i].Wa[j][l] = k/100000;//0;
              }
          }
          for(j=0;j<N;j++){                //luego Wt
              for(l=0;l<H;l++){
                  k = rand() % 100000;
                  Xi[i].Wt[j][l] = k/100000;
                  k = rand() % 100000;
                  Pi[i].Wt[j][l] = k/100000;
                  k = rand() % 100000;
                  Vi[i].Wt[j][l] = k/100000;//0;
              }
          }
          for(l=0;l<H;l++){                //finalmente Ww
              k = rand() % 100000;
              Xi[i].Ww[l] = k/100000;
              k = rand() % 100000;
              Pi[i].Ww[l] = k/100000;
              k = rand() % 100000;
              Vi[i].Ww[l] = k/100000;//0;
          }
      }
```

```
    //inicializando G
    for(j=0;j<N;j++)        //primero Wa y Wt
       for(l=0;l<H;l++){
          k = rand() % 100000;
          G.Wa[j][l] = k/100000;//Xi[0].Wa[j][l];
          k = rand() % 100000;
          G.Wt[j][l] = k/100000;//Xi[0].Wt[j][l];
       }
    for(l=0;l<H;l++){       //finalmente Ww
       k = rand() % 100000;
       G.Ww[l] = k/100000;//Xi[0].Ww[l];

    }
}

/*****************************************************************************/
/* Funcion PSO Clasico                                                       */
/*****************************************************************************/
void PSOClasico(void){

    int k=0;                  //contador de iteraciones
    int i,j,l;
    float r1,r2,k1,k2;        //valores estocasticos
    float f;
    FILE *fp2;                //para la configuracion obtenida

    //abriendo las lecturas
    fp = fopen("dataanchovy.csv", "r");
    if (fp == NULL)
       printf ("Abrir archivo data\t\t..... ERROR\n");

    v1=v2=0;
    VMax=VMin=0.0;

//comienzan las iteraciones
    while(k<K){
       for(i=0;i<P;i++){
          f=fitness(Xi[i]);              //actualizando Pi
          if( f < fitness(Pi[i])){
             for(j=0;j<N;j++){
                for(l=0;l<H;l++){
                   Pi[i].Wa[j][l] = Xi[i].Wa[j][l];
                   Pi[i].Wt[j][l] = Xi[i].Wt[j][l];
                }
             }
             for(l=0;l<H;l++){
                Pi[i].Ww[l] = Xi[i].Ww[l];
             }
          }
          if(fitness(Pi[i]) < fitness(G)){  //actualizando G
             for(j=0;j<N;j++){
                for(l=0;l<H;l++){
                   G.Wa[j][l] = Pi[i].Wa[j][l];
                   G.Wt[j][l] = Pi[i].Wt[j][l];
                }
             }
```

```
            for(l=0;l<H;l++){
                G.Ww[l] = Pi[i].Ww[l];
            }
        }
    } //fin actualizacion de Pi y G

    //movimiento de particulas
    for(i=0;i<P;i++){
        k1=(rand()%100000); //valores estocasticos
        k2=(rand()%100000);
        r1=k1/100000;
        r2=k2/100000;
        //printf("%f %f \n",r1,r2);  // viendo valores

        //variando la velocidad
        // primero las componentes Wa  y Wt
        for(j=0;j<N;j++){
            for(l=0;l<H;l++){
                Vi[i].Wa[j][l] = Vi[i].Wa[j][l] + (C1 * r1 * (Pi[i].Wa[j][l]-Xi[i].Wa[j][l])) + (C2 * r2 *
                                 (G.Wa[j][l]-Xi[i].Wa[j][l]));
                Vi[i].Wt[j][l] = Vi[i].Wt[j][l] + (C1 * r1 * (Pi[i].Wt[j][l]-Xi[i].Wt[j][l])) + (C2 * r2 * (G.Wt[j][l]-
                                 Xi[i].Wt[j][l]));
                //tomando el promedio
                if(Vi[i].Wa[j][l]>0){
                  VMax=VMax+Vi[i].Wa[j][l];
                  v1++;
                }
                if(Vi[i].Wt[j][l]>0){
                  VMax=VMax+Vi[i].Wt[j][l];
                  v1++;
                }
                if(Vi[i].Wa[j][l]<0){
                  VMin=VMin+Vi[i].Wa[j][l];
                  v2++;
                }
                if(Vi[i].Wt[j][l]<0){
                  VMin=VMin+Vi[i].Wt[j][l];
                  v2++;
                }

            }
        }

        //despues la componente Ww
        for(l=0;l<H;l++){
            Vi[i].Ww[l] = Vi[i].Ww[l] + (C1* r1 * (Pi[i].Ww[l]-Xi[i].Ww[l])) + (C2 * r2 * (G.Ww[l]-
                          Xi[i].Ww[l]));
            //obteniendo promedio
            if(Vi[i].Ww[l]>0){
              VMax=VMax+Vi[i].Ww[l];
              v1++;
            }
            if(Vi[i].Ww[l]<0){
              VMin=VMin+Vi[i].Ww[l];
              v2++;
            }
```

```
            }

            //variando la posicion
            // primero las componentes Wa  y Wt
            for(j=0;j<N;j++){
               for(l=0;l<H;l++){
                  Xi[i].Wa[j][l] = Xi[i].Wa[j][l] + Vi[i].Wa[j][l];
                  if(Xi[i].Wa[j][l]>XnMax)
                     Xi[i].Wa[j][l]=XnMax;
                  if(Xi[i].Wa[j][l]<XnMin)
                     Xi[i].Wa[j][l]=XnMin;

                  Xi[i].Wt[j][l] = Xi[i].Wt[j][l] + Vi[i].Wt[j][l];
                  if(Xi[i].Wt[j][l]>XnMax)
                     Xi[i].Wt[j][l]=XnMax;
                  if(Xi[i].Wt[j][l]<XnMin)
                     Xi[i].Wt[j][l]=XnMin;
               }
            }

            //despues la coponente Ww
            for(l=0;l<H;l++){
               Xi[i].Ww[l] = Xi[i].Ww[l] + Vi[i].Ww[l];
               if(Xi[i].Ww[l]>XnMax)
                  Xi[i].Ww[l]=XnMax;
               if(Xi[i].Ww[l]<XnMin)
                  Xi[i].Ww[l]=XnMin;
            }

         }//fin movimiento de particulas

      k++;
      }//fin iteraciones

      VMax=VMax/v1;
      VMin=VMin/v2;
      printf("VMax: %f  VMin: %f\n",VMax,VMin);


      fclose(fp);

}


/*****************************************************************************/
/* Funcion de Prueba                                                         */
/*****************************************************************************/
void prueba(){
FILE *fp, *fp6;
      int i,j,k;
      float f1,f2,f3,f4,mse;
      f1=0.0;
      f2=0.0;
      f3=0.0;
      RMSE=MAPE=R2= AIC=0.0;
      fp = fopen("dataanchovy.csv", "r");
```

```
if (fp == NULL){
    printf ("Abrir archivo data\t\t..... ERROR\n");
}

//creando salida de prueba
char salidap[30] = "";
char *n,*h;
strcat(salidap, "datos\\salidaprueba");
switch(N){
        case 2:
            n="2";
            break;
        case 3:
            n="3";
            break;
        case 4:
            n="4";
            break;
        case 5:
            n="5";
            break;
        case 6:
            n="6";
            break;
        case 7:
            n="7";
            break;
        case 8:
            n="8";
            break;
        case 9:
            n="9";
            break;
        case 10:
            n="10";
            break;
        case 11:
            n="11";
            break;
        case 12:
            n="12";
            break;
        default:
            break;
        }

strcat(salidap,n);
strcat(salidap, "y");
switch(H){
        case 10:
            h="10";
            break;
        case 11:
            h="11";
            break;
        case 12:
```

```
            h="12";
            break;
        case 13:
            h="13";
            break;
        case 14:
            h="14";
            break;
        case 15:
            h="15";
            break;
        case 16:
            h="16";
            break;
        case 17:
            h="17";
            break;
        case 18:
            h="18";
            break;
        case 19:
            h="19";
            break;
        case 20:
            h="20";
            break;
        default:
            break;
        }

strcat(salidap, h);
strcat(salidap, ".csv");

fp6 = fopen(salidap, "w");
if (fp6 == NULL){
   printf ("Abrir crear archivo salida de prueba\t..... ERROR\n");
}
//primera lectura
for(i=0;i<N;i++){
   fscanf( fp, "%f", &X[i]);
}


k=0; //validaciones
//leyendo configuracion desde G
fprintf(fp6,"R;Y;R-Y\n");
while(!feof(fp)){
   fscanf( fp, "%f", &R);
   k++;
   forward(G);
   fprintf(fp6,"%f;%f;%f\n",R,Y,R-Y);

   for(j=0;j<N-1;j++)
      X[j] = X[j+1];
   X[j] = R;
   f1=f1 + ((R-Y)*(R-Y));
```

```
        f2=f2 + fabs(R-Y);
        f3=f3 + ((R-RP)*(R-RP));
    }
    mse=f1/k;
    RMSE=sqrt(mse);
    MAPE=f2/k;
    R2=1.0 - (f1/f3);
    f4=4*N*H;
    f4=f4/k;
    AIC=log10(mse) + f4;

fclose(fp);
fclose(fp6);
}


/*****************************************************************************/
/* Funcion Principal                                                         */
/*****************************************************************************/
int main(){

    int i,n;
    float l;
    float valor;
    FILE *fp7;
    fp7 = fopen("dataanchovy.csv", "r");
    if (fp7 == NULL){
        printf ("Abrir leer datos promedio\t..... ERROR\n");
    }
    n=0;
    RP=0.0;
    Rtl=0.0;
    l=0.0;

    while(!feof(fp7)){
        n++;
        fscanf( fp7, "%f", &valor);
        Rtl=Rtl+((valor-l)*(valor-l));
        RP=RP + valor;
        l=valor;
    }
    RP=RP/n;
    fclose(fp7);

    FILE *fp8;
    fp8 = fopen("resultados.csv", "w");
    if (fp8 == NULL){
        printf ("Abrir crear archivo resultados\t..... ERROR\n");
    }

    printf("\n\nPrograma Automatico de Entrenamiento y Pruebas\n\n");
    P=20;
    K=2000;
    C1=1.49;
    C2=1.49;
```

```
fprintf(fp8,"Particulas:;%d\n",P);
fprintf(fp8,"Iteraciones:;%d\n",K);
fprintf(fp8,"Cognitivo:;%f\n",C1);
fprintf(fp8,"Social:;%f\n\n",C2);
fprintf(fp8,"N;H;RMSE;MAE;SEP;PI;ARV;AIC\n");


for(N=2;N<=12;N++){
  for(H=10;H<=20;H++){
    printf("Inicio entrenamiento y prueba con N= %d y H= %d\n",N,H);

    inicializarPSO();
    PSOClasico();
    prueba();
    fprintf(fp8,"%d;%d;%f;%f;%f;%f;%f;%f\n",N,H,RMSE,MAE,SEP,PI,ARV,AIC);
    printf("Termino\n\n");
  }
}
printf("\n\nPruebas Terminadas\n\n");

fclose(fp8);

//para hacer los entrenamientos manualmente
/*

do{

  printf("\n\n\t******* MENU ********\n\n");
  printf("\t1. Ingresar Parametros.\n\n");
  printf("\t2. Salir.\n\n\n");
  printf("Ingrese Opcion: ");
  scanf("%d",&op);

  switch(op){
  case 1:
      printf("\n\nNumero de particulas P:\t\t");
      scanf("%d",&P);
      printf("\nNumero de iteraciones K:\t");
      scanf("%d",&K);
      printf("\nNumero de entradas N:\t\t");
      scanf("%d",&N);
      printf("\nNumero de neuronas ocultas H:\t");
      scanf("%d",&H);
      printf("\nCoeficiente cognitivo C1:\t");
      scanf("%f",&C1);
      printf("\nCoeficiente social C2:\t\t");
      scanf("%f",&C2);
      printf("\nENTRENANDO... ");
      inicializarPSO();
      PSOClasico();
      printf("\n\nEntrenamiento Finalizado!!\n\n\n\n");
      printf("Inicio de la Prueba");
      prueba();
      fprintf(fp8,"%d;%d;%f;%f;%f;%f;%f;%f\n",N,H,RMSE,MAE,SEP,E2,ARV,AIC);
      printf("\n\nTermino de la prueba\n\n");
```

```
            getch();
            break;
        case 2:
            break;
        default:
            printf("\n\nOpcion no existe!!\n");
            getch();
            break;

        }

    }while(op!=2);

    */

    fclose(fp8);

    getch();

    return 0;

}
```