Pontificia Universidad Católica de Valparaíso

Facultad de Ingeniería

Escuela de Ingeniería Informática

PREDICCIÓN DE CAPTURA DE ANCHOVETAS UTILIZANDO REDES NEURONALES

Autor:

Rodrigo Cristian Pérez Galleguillos

Informe final del Proyecto para optar al Título profesional de

Ingeniero Civil en Informática

Profesor guía:

Nibaldo Rodríguez Agurto

Profesor Co-referente:

Broderick Crawford Labrín

Diciembre, 2007

A mis padres y hermana, por creer en todo momento en mí.

AGRADECIMIENTOS

A mis amigos y profesor guía, por todo el apoyo.

GLOSARIO

ADN	Ácido Desoxirribonucleico.
ADALINE	Adaptative Linear Elements.
BI	Business Intelligence.
BP	Algoritmo de aprendizaje Backpropagation.
ECM	Error Cuadrático Medio.
ENIAC	Electronic Numerical Integrator and Computer.
IA	Inteligencia Artificial.
LM	Algoritmo de aprendizaje Levenberg Marquardt.
MATLAB	Matriz Laboratory.

MC	Simulación de Monte Carlo
MLP	MultiLayer Perceptron.
MSE	Mean Squared Error.
RLC	Regresión Lineal Clásica.
RN	Red Neuronal.
RNA	Red Neuronal Artificial.
RNA's	Redes Neuronales Artificiales.
RNB's	Redes Neuronales Biológicas.
RNC's	Redes Neuronales Computacionales.
RNR's	Redes Neuronales Recurrentes.
RNS	Redes Neuronales Sigmoidales.
RTRL	Real Time Recurrent Learning.
SAG	Servicio Agrícola Ganadero.
SERNAP	Servicio Nacional de Pesca.
SERNAPESCA	Servicio Nacional de Pesca.
STD	Desviación Estándar.
TRA	Teoría de Resonancia Adaptada.

RESUMEN

El problema de pronóstico de captura de anchovetas en la zona norte de Chile tradicionalmente ha sido modelado utilizando regresión lineal clásica (RLC). Sin embargo, la técnica de RLC no considera los fenómenos no lineales que puede exhibir el proceso de pronóstico. Por lo tanto, en esta memoria de título es propuesto un modelo no lineal de predicción de captura de anchovetas basado en redes neuronales sigmoidales (RNS). La arquitectura del modelo RNS está compuesta por una capa de entrada, una capa oculta no lineal y una capa de salida lineal y los pesos de la RNS son estimados utilizando el algoritmo de aprendizaje Levenberg Marquardt. La mejor topología encontrada durante la fase de evaluación ha sido una RNS con seis nodos de entrada, dieciséis nodos ocultos y un nodo de salida. Además, la varianza obtenida fue igual a un 91% con el modelo RNS propuesto.

ABSTRACT

The capture forecast problem of anchovy in the northern part of Chile traditionally has been shaped utilizing classical lineal decline (RLC). Nevertheless, the technique of RLC does not consider the not lineal phenomenon that can exhibit the process of forecast. Therefore, in this memory of title is proposed a not lineal model of prediction of capture of anchovy based on networks neural sigmoidal (RNS). The architecture of the model RNS is composed by an input layer, a not lineal hidden layer and the output lineal layer and the weights of the RNS are reckoned utilizing the algorithm of learning Levenberg Marquardt. The best topology found during the phase of evaluation has been a RNS with six nodes of input, sixteen hidden nodes and one node at the output. Besides, the variance obtained was equal to a 91% with the model RNS proposed.

Capítulo 1

Introducción

1.1 Introducción

En América del Sur, el sistema de corrientes de Perú-Chile sustenta uno de los ecosistemas marinos más productivos del mundo. En 1996 cerca de un 20% de las capturas de especies pelágicas mundiales se realizaron en esta región, lo cual representa un 0,09% de la superficie del océano mundial. El comportamiento y distribución de los recursos pelágicos en la zona norte de Chile están fuertemente ligados a las condiciones bio-oceanográficas existentes en el litoral de la XV, I y II regiones de nuestro país, por esta razón desde el año 1964 el país viene realizando cruceros bio-oceanográficos estacionales en la zona norte, cuyos resultados han contribuido con información actualizada y oportuna para explicar el comportamiento del recurso pelágico denominado Anchoveta, frente a anomalías oceanográficas, y han entregado datos sobre la abundancia y distribución mediante índices de densidad y cobertura geográfica.

Observaciones directas de corrientes medidas sobre el talud continental frente a Chile, han revelado una compleja estructura de variabilidad en la banda intraestacional, estacional e interanual [1]. Mucha de esta variabilidad aparece forzada remotamente desde el Pacífico ecuatorial, por las oscilaciones de Madden-Julian [2;3] y por los vientos ecuatoriales en la banda estacional [4] y por la corriente del Niño y la oscilación del sur en la banda interanual [1;4]. Esta variabilidad vinculada a la dinámica ecuatorial abre ahora un potencial de predictibilidad de las condiciones oceanográficas frente a Chile con varios meses de anticipación.

Chile es uno de los principales países pesqueros a nivel mundial, con aproximadamente 5 millones de toneladas de capturas anuales. Las especies pelágicas constituyen cerca de un 90% de las capturas anuales, donde la pesquería pelágica de la zona norte de Chile (18°21’S-24°00’S) contribuye con un 48% (valor promedio entre 1951-1998; SAG, 1957-77, SERNAP, 1978-1994 y SERNAPESCA, 1995-97) lo cual la posiciona como la segunda área más importante del país. Esta pesquería se realiza desde la década del 50, primero y esencialmente

sobre la Anchoveta (Engraulis ringens), especie que será utilizada para establecer el modelo predictivo de esta memoria de título. Esta pesquería colapsa a mediados de la década del 70, siendo reemplazada por la de la Sardina (Sardinops sagax). Después de 1975 también aumentan las capturas de Jurel (Trachurus murphy) y Caballa (Scomber japonicus), representando en el norte de chile el 15% y 4% de los desembarques del período 1976-2002; en tanto la Anchoveta y Sardina representan el 34% y 46% respectivamente (SAG, 1976-77; SERNAPESCA, 1978-2002).

El problema de la predicción de los recursos pesqueros es un gran desafío debido a la dificultad de llevarla a cabo de forma precisa y efectiva, como consecuencia de que dicha afirmación juega un papel central en el manejo de stocks de recursos. La predicción precede a la planificación que a su vez precede a la toma de decisiones. De esta forma, la política de gestión de cualquier pesquería establece en primer lugar las metas y objetivos a conseguir. Por lo general, el objetivo principal es establecer el esfuerzo de pesca a aplicar en un lugar concreto y durante un período de tiempo determinado. La consecución de este objetivo pasa inevitablemente por la predicción de eventos a priori incontrolables, como cambios en la abundancia y disponibilidad de ejemplares objeto de gestión, mediante la aplicación de las acciones y métodos apropiados.

La predicción de los cambios de abundancia, tan solo puede ser llevada a cabo, si se dispone de información cuantitativa/cualitativa del pasado de la pesquería y se asume que ciertos aspectos de la información disponible se repetirán en el futuro. En nuestro caso, se cuenta con una base de datos con información de más de 50 años, correspondiente al desembarco de Anchovetas en la zona norte de Chile, además de información adicional del país limítrofe Perú. Esta información, la cual veremos reflejada en el siguiente gráfico, es de suma importancia, ya que la consideraremos como base para poder construir un modelo predictivo adecuado para la solución de nuestra problemática, la cual es predecir la captura de Anchovetas.

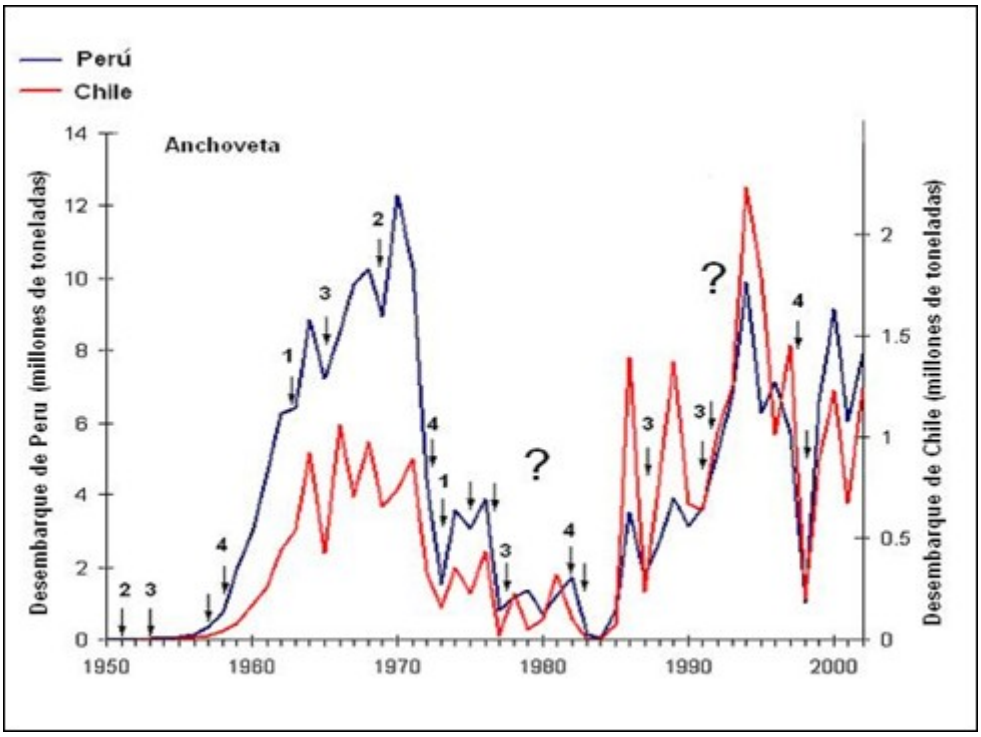

Figura 1.1: Gráfico desembarco de Anchovetas.

El diseño e implementación de las políticas, estrategias y medidas de administración, así como la evaluación de los impactos de las mismas, requiere de un conjunto integrado de información y conocimientos sobre aspectos ambientales, biológicos, técnicos, económicos, sociales, legales e institucionales que determinan en definitiva el desempeño de la actividad pesquera. Una de las necesidades prácticas de la autoridad pesquera, en este contexto, es la de generar la información y conocimiento para el diseño e implementación de los planes de manejo de las pesquerías chilenas. Se ha crecido bastante respecto a la década del 50 en información pesquera tanto en cantidad como en calidad, pero también se imponen nuevos y mayores desafíos para la administración pesquera, con respecto a esto, generar nuevos modelos predictivos con nuevas metodologías que aumenten la precisión de los mismos será una herramienta importante para la toma de decisiones en la administración pesquera.

Recientes investigaciones han puesto de manifiesto la utilidad, potencia y precisión de modelos que tradicionalmente se han incluido dentro del campo de la Inteligencia Artificial (IA). Entre estos modelos, también conocidos como modelos heurísticos, destacan las Redes Neuronales Artificiales o Computacionales (RNA's ó RNC's). La ventaja fundamental de este tipo de modelos frente a las técnicas de estimaciones clásicas, es que los modelos heurísticos permiten la modelación de sistemas altamente no lineales y no equilibrados (como las

pesquerías), mediante la transformación no lineal de los datos de entrada y salida, lo que supone importantes ventajas frente a metodologías estadísticas convencionales.

Por todo lo anteriormente mencionado, esta memoria de título plantea la búsqueda de modelos predictivos para la captura de Anchovetas en la zona norte de Chile, utilizando Redes Neuronales Sigmoidales con al Algoritmo de Aprendizaje Levenberg Marquardt.

1.2 Objetivos General

El objetivo general de esta memoria de título es:

Desarrollar e implementar un modelo de predicción de Anchovetas de la zona norte chilena utilizando Redes Neuronales Sigmoidales con el algoritmo de aprendizaje Levenberg Marquardt.

1.2.1 Objetivos Específicos

Comprender el funcionamiento de las arquitecturas de las Redes Neuronales.
Analizar e implementar los algoritmos de aprendizaje Backpropagation (BP) y Levenberg Marquardt (LM)
Evaluar y contrastar el rendimiento del modelo RNS basado en los algoritmos BP y LM.

1.3 Organización del Texto

La organización de la presente memoria de título se divide en capítulos. El Capítulo dos, dedicado a las Redes Neuronales, se presenta su reseña histórica, ventajas y las distintas clasificaciones, donde se destaca la topología utilizada en el predictor. En el Capítulo tres, se presentan los algoritmos de aprendizaje Backpropagation y Levenberg Marquardt, donde se explicará el funcionamiento detallado de cada uno. En el Capítulo cuatro se presenta una discusión de los resultados obtenidos en base a comparaciones de los rendimientos de los predictores propuestos. Por último, en el Capítulo cinco se mencionan las conclusiones.

Capítulo 2

Redes Neuronales

2.1 Introducción

El hombre se ha caracterizado siempre por la búsqueda constante de nuevas vías para mejorar sus condiciones de vida. Estos esfuerzos le han servido para reducir el trabajo en aquellas operaciones en las que la fuerza y la parte intelectual juegan un papel primordial. Los progresos obtenidos han permitido dirigir estos esfuerzos a otros campos, como por ejemplo, a la construcción de máquinas calculadoras que ayuden a resolver de forma automática y rápida determinadas operaciones que resultan molestas cuando se realizan a mano.

Muchos trataron infructuosamente de construir una máquina capaz de resolver problemas matemáticos. Posteriormente, otros tantos intentaron construir máquinas similares, pero no fue hasta la Segunda Guerra Mundial, cuando ya se disponía de instrumentos electrónicos, que se empezaron a recoger los primeros frutos. En 1946 se construyó la primera computadora electrónica, ENIAC. Desde entonces los desarrollos en este campo han tenido un auge espectacular.

Estas máquinas, permiten implementar fácilmente algoritmos para resolver multitud de problemas que antes resultaban difíciles de solucionar. Sin embargo, se observa una limitación importante: ¿qué ocurre cuando el problema que se quiere resolver no admite un tratamiento algorítmico, como es el caso, por ejemplo, de la clasificación de objetos por rasgos comunes?. Este ejemplo demuestra que la construcción de nuevas máquinas con un mayor desarrollo tecnológico requiere un enfoque del problema desde otro punto de vista. Los desarrollos actuales de los científicos, se dirigen al estudio de las capacidades humanas como una fuente de nuevas ideas para el diseño de las nuevas máquinas. Así, la inteligencia artificial es un intento por descubrir y describir aspectos de la inteligencia humana que pueden ser simulados mediante máquinas. Esta disciplina se ha desarrollado fuertemente en los últimos años teniendo aplicación en algunos campos como visión artificial, demostración de teoremas, procesamiento de información expresada mediante lenguajes humanos, etc. Las

redes neuronales son más que otra forma de emular ciertas características propias de los humanos, como la capacidad de memorizar y de asociar hechos. Si se examinan con atención aquellos problemas que no pueden expresarse a través de un algoritmo, se observará que todos ellos tienen una característica en común: la experiencia. El hombre es capaz de resolver estas situaciones acudiendo a la experiencia acumulada. Así, parece claro que una forma de aproximarse al problema consista en la construcción de sistemas que sean capaces de reproducir esta característica humana. En definitiva, las redes neuronales no son más que un modelo artificial y simplificado del cerebro humano, que es el ejemplo más perfecto del que disponemos para un sistema que es capaz de adquirir conocimiento a través de la experiencia. Una red neuronal es "un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona" [5].

Todos los procesos del cuerpo humano se relacionan de alguna forma con la actividad o inactividad de estas neuronas. Las mismas son un componente relativamente simple del ser humano, pero cuando millares de ellas realizan su sinapsis se hacen muy poderosas.

Lo que básicamente ocurre en una neurona biológica es lo siguiente: la neurona es estimulada a través de sus entradas (inputs) y cuando se alcanza un cierto umbral, la neurona se dispara o activa, pasando una señal hacia el axón. Posteriores investigaciones condujeron al descubrimiento de que estos procesos son el resultado de eventos electroquímicos.

Como ya se sabe, el pensamiento tiene lugar en el cerebro, que consta de billones de neuronas interconectadas. Así, el secreto de la "inteligencia", sin importar como se defina, se sitúa dentro de estas neuronas interconectadas y de su interacción. También, es bien conocido que los humanos son capaces de aprender. Aprendizaje significa que aquellos problemas que inicialmente no pueden resolverse, pueden ser resueltos después de obtener más información acerca del problema. Por lo tanto, las Redes Neuronales [5]:

Consisten de unidades de procesamiento que intercambian datos o información.
Se utilizan para reconocer patrones, incluyendo imágenes, manuscritos y secuencias de tiempo.

En el problema de estimación de variables, se pueden conseguir errores promedios menores al 5% para un sin fin de aplicaciones; dependiendo de una correcta selección de las variables influyentes en el proceso.

Su aplicación es sencilla.

Un adecuado y continuo entrenamiento, puede adaptar a la red a los cambios que pudieran presentarse en el sistema, indicando características dinámicas y adaptivas.

Esto lleva a tomar como punto de partida el uso de las redes neuronales como un método de solución al problema planteado anteriormente. La red neuronal como herramienta de predicción o estimación, se entrena con los datos históricos (patrones de entrada/salida, conocidos) que le permita ir ajustando una serie de números (memoria, pesos o constantes usadas para enlazar la entrada con la salida) hasta lograr la respuesta deseada.

2.2 Reseña Histórica

El deseo de emular el comportamiento del cerebro humano, con todo lo que esto involucra, llevó a que en 1936 Alan Turing fuera el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, los primeros teóricos que concibieron los fundamentos de la computación neuronal fueron Warren McCulloch, un neurofisiológico, y Walter Pitts, un matemático, quienes, en 1943, lanzaron una teoría acerca de la forma de trabajar de las neuronas. Ellos modelaron una red neuronal simple mediante circuitos eléctricos. Posteriormente en 1949, Donald Hebb fue el primero en explicar los procesos del aprendizaje (que es el elemento básico de la inteligencia humana) desde el punto de vista psicológico, desarrollando una regla de como el aprendizaje ocurría. Aún hoy, este es el fundamento de la mayoría de las funciones de aprendizaje que pueden encontrarse en una red neuronal. Su idea fue que el aprendizaje ocurría cuando ciertos cambios en una neurona eran activados. También intentó encontrar semejanzas entre el aprendizaje y la actividad nerviosa. Los trabajos de Hebb formaron las bases de la Teoría de las Redes Neuronales. Ya en la década de los 50’ Kart Lashley en su serie de ensayos, encontró que la información no era almacenada en forma centralizada en el cerebro, sino que era distribuida encima de él.

Pero estos hallazgos no podían quedar en el anonimato, es por eso que en 1956 se celebró el Congreso de Dartmouth, dando paso al nacimiento de la IA.

Estando aún en los años 50, Frank Rosenblatt en 1957, comenzó el desarrollo del Perceptrón. Esta es la red neuronal más antigua; utilizada hoy en día para aplicación como identificador de patrones. Este modelo era capaz de generalizar, es decir, después de haber aprendido una serie de patrones podía reconocer otros similares, aunque no se le hubiese presentado en el entrenamiento. Sin embargo, tenía una serie de limitaciones, por ejemplo, su incapacidad para resolver el problema de la función OR exclusiva y, en general, era incapaz de clasificar clases no separables linealmente. Luego en 1959, el mismo Frank Rosenblatt escribió: "Principios de Neurodinámica", en este libro confirmó que, bajo ciertas condiciones, el aprendizaje del Perceptrón convergía hacia un estado finito (Teorema de Convergencia del Perceptrón).

En 1960, Bernard Widroff y Marcian Hoff desarrollaron el modelo ADALINE (ADAptative LINear Elements). Esta fue la primera red neuronal aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas) que se ha utilizado comercialmente durante varias décadas. Un año más tarde, en 1961, Kart Steinbeck realiza: "Die Lernmatrix", una red neuronal para simples realizaciones técnicas (memoria asociativa).

Ocho años después, en 1969 casi se produjo la "muerte abrupta" de las Redes Neuronales, ya que Marvin Minsky y Seymour Papert probaron (matemáticamente) que el perceptrón no era capaz de resolver problemas relativamente fáciles, tales como el aprendizaje de una función no-lineal. Este demostró que el Perceptrón era muy débil, dado que las funciones no-lineales son extensamente empleadas en computación y en los problemas del mundo real.

Pero en 1974 Paul Werbos desarrolló la idea básica del algoritmo de aprendizaje de propagación hacia atrás (backpropagation), cuyo significado quedó definitivamente aclarado en 1985.

En 1977 Stephen Grossberg desarrolla la "Teoría de Resonancia Adaptada (TRA)". TRA es una arquitectura de red que se diferencia de todas las demás previamente inventadas. La misma simula otras habilidades del cerebro: memoria a largo y corto plazo.

Pasaron casi diez años, para que en 1985 John Hopfield provocara el renacimiento de las redes neuronales con su libro: "Computación neuronal de decisiones en problemas de optimización". Posteriormente en 1986 David Rumelhart y G. Hinton redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation).

2.3 Red Neuronal Biológica

El cerebro humano es uno de los órganos más complejos conocidos por el ser humano, se puede decir que es un dispositivo de cálculo casi perfecto. El cerebro no tan sólo puede resolver problemas, recordar, organizar, etc., además cuenta con la capacidad de aprender.

El cerebro humano contiene aproximadamente 12 billones de células nerviosas o neuronas [6]. Cada neurona tiene de 5.600 a 60.000 conexiones dendríticas provenientes de otras neuronas, (figura 2.1). Estas conexiones transportan los impulsos enviados desde otras neuronas y están conectadas a la membrana de la neurona. Cada neurona tiene una salida denominada axón. El contacto de cada axón con una dendrita se realiza a través de la sinapsis. Tanto el axón como las dendritas transmiten la señal en una única dirección. Cada neurona recibe de 10.000 a 100.000 sinapsis y el axón realiza una cantidad de conexiones similar.

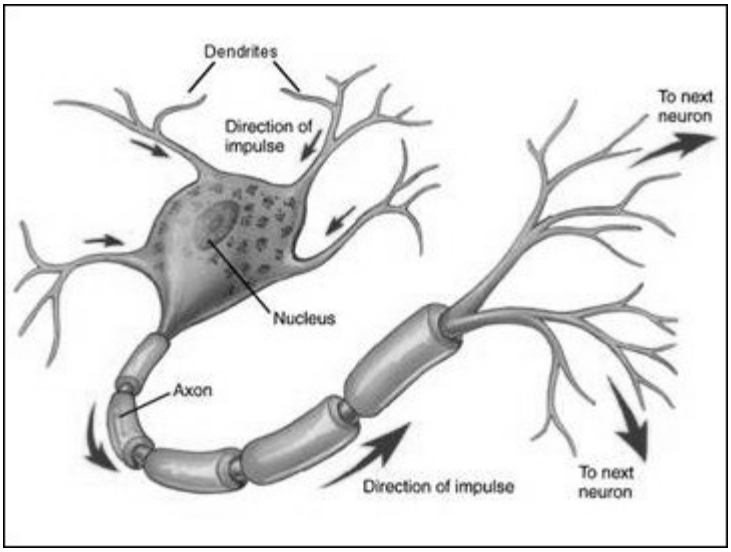

Figura 2.1: Neurona biológica.

Las neuronas son eléctricamente activas e interactúan entre ellas mediante un flujo de corrientes eléctricas locales. Estas corrientes se deben a diferencias de potencial entre las membranas de las neuronas. Un impulso nervioso es un cambio de voltaje que ocurre en una zona localizada de la membrana celular. El impulso se transmite a través del axón hasta llegar a la sinapsis, produciendo la liberación de una sustancia química denominada neurotransmisor, el cual se esparce por el fluido existente en el espacio sináptico. Cuando este fluido alcanza el otro extremo transmite la señal a la dendrita. Los impulsos recibidos desde la sinapsis se suman o restan a la magnitud de las variaciones del potencial de la membrana. Si

las contribuciones totales alcanzan un valor determinado (alrededor de 10 milivoltios) se disparan uno o más impulsos que se propagarán a lo largo del axón.

El efecto de los neurotransmisores sobre la neurona receptora puede ser excitatorio o inhibitorio, y es variable, de manera que podemos hablar de la fuerza o efectividad de una sinapsis. Las señales excitatorias e inhibitorias recibidas por una neurona se combinan, y en función de la estimulación total recibida, la neurona toma un cierto nivel de activación, que se traduce en la generación de breves impulsos nerviosos con una determinada frecuencia de disparo, y su propagación a lo largo del axón hacia las neuronas con las cuales hace sinapsis.

2.4 Red Neuronal Artificial

Las Redes Neuronales Artificiales son dispositivos o software desarrollados en base a modelos y estructuras neuronales del cerebro humano. Mediante las neuronas o unidades de procesamiento son capaces de almacenar conocimiento por experiencia y hacerlo disponible para su uso. Más bien, están diseñadas para desarrollar una determinada tarea de forma similar a la cual la ejecutaría el cerebro humano. Entre las principales tareas y funciones que realizan las RNA's se encuentran: la percepción, asociación, reconocimiento y clasificación de patrones, estimación de funciones, compresión de datos, entre otras.

Siguiendo la definición y funcionamiento de la neurona natural, se puede plantear un modelo en diagrama de bloques tal como se muestra en la siguiente figura:

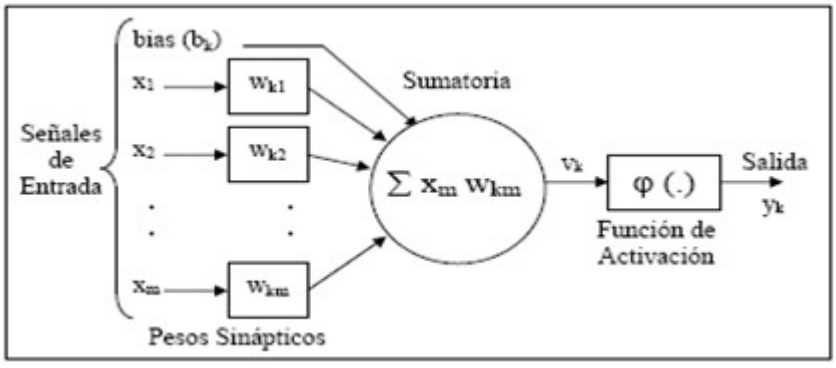

Figura 2.2: Modelo de una neurona artificial.

Este modelo conforma el diseño básico de una red neuronal artificial. En él se pueden apreciar tres elementos constitutivos:

- *Un grupo de sinapsis o enlaces de interconexión:* Que caracterizan cada entrada a través de la multiplicación de la misma con un valor, el cual se denomina peso. Este representa la eficiencia de la unión sináptica en la transmisión de la información. Los pesos se denotan con la letra "*w*". Específicamente, una señal X_j antes de ingresar a la neurona *k* será multiplicada por el peso sináptico W_{jk}.

- *Un sumador:* Que realiza la combinación lineal de las señales provenientes de las respectivas sinapsis de la neurona.

- *Una función de activación:* Cuya tarea es limitar la amplitud de la señal de salida de la neurona, a un rango de valores finitos. Típicamente, los rangos normalizados de las amplitudes de las señales de salida son los intervalos *(0,1)* y alternativamente *(-1,1)*.

El modelo de la figura anterior, incluye una entrada externa que se denomina tendencia o bias, y está denotada por B_k. Este parámetro adicional produce el efecto de añadir una cierta tendencia en los datos a la entrada de la función de activación, y permite trasladar dicha función a la región de interés según sea la aplicación.

En términos matemáticos, se puede describir una neurona *k* mediante las siguientes ecuaciones:

$$u_k = \sum_{j=1}^{m} w_{kj} \cdot x_j \qquad (2.1)$$

$$v_k = u_k + b_k \qquad (2.2)$$

$$y_k = \varphi(v_k) \qquad (2.3)$$

Donde:

$x_1, x_2, \ldots, x_m$ son las señales de entrada.

$w_{k1}, w_{k2}, \ldots, w_{km}$ son los pesos sinápticos que unen la salida de la neurona k con las entradas de las neuronas $1,2 \ldots m$.

u_k es una combinación lineal de las señales de entrada sopesadas por los respectivos pesos sinápticos.

b_k es el bias.

v_k es el potencial de activación de la neurona k.

y_k es la señal de salida de la neurona.

$\varphi(\cdot)$ es la función de activación.

La función de activación es uno de los parámetros más importantes dentro del modelo de la neurona, ya que define la salida de una neurona en términos de su potencial de activación. La función sigmoidal es el tipo de función de activación usada con mayor frecuencia para las aplicaciones de las RNA. Su nombre se debe a la forma de S que presenta la gráfica que se apreciará a continuación, y está definida por una función estrictamente creciente, no lineal y diferenciable, condiciones deseables para el proceso de entrenamiento de la red. Además la condición de diferenciabilidad, es necesaria para que se pueda aplicar la rutina de entrenamiento basada en las arquitecturas de retropropagación, base radial, etc.

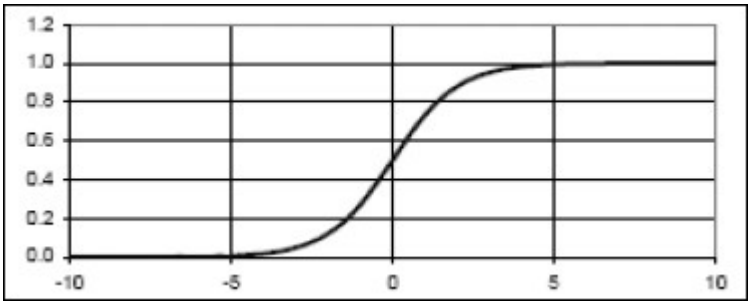

Figura 2.3: Función de Activación Sigmoidal.

Por otra parte, las funciones sigmoidales de mayor uso en la literatura son la función sigmoidal logarítmica, representada en la figura anterior, definida matemáticamente por

$$\varphi(\cdot) = 1/1 + e(-a \cdot v)$$

donde *"a"* representa la pendiente de la curva; y la función sigmoidal tangencial, definida matemáticamente por

$$\varphi(v) = aTanh(bv)$$

donde *"a"* y *"b"* son constantes positivas. Aunque estas dos funciones sólo difieren en la escala y en la simetría alrededor de cero, se ha comprobado experimentalmente que la sigmoidal tangencial ofrece una mejor estabilidad numérica en el entrenamiento de la RNA.

2.5 Ventajas de las Redes Neuronales

Debido a la semejanza en el funcionamiento de las Redes Neuronales Artificiales con el cerebro, las RNA's nos ofrecen las siguientes ventajas:

- *Aprendizaje:* Tienen la capacidad de aprender los datos que se le presentan, con lo cual pueden capturar desde las relaciones más sutiles, hasta las más complejas. Esta capacidad estará definida por la topología de las RNA's y el método de aprendizaje utilizado.

- *No-linealidad:* Una RNA puede ser lineal o no. Esta es una gran ventaja, que le permite a las RNA's capturar interacciones complejas entre las variables de entrada de un sistema. La no-linealidad es una propiedad extremadamente importante, si los sistemas responsables de la generación de información son no-lineales.

- *Pueden Generalizar:* Son capaces de manejar las imprecisiones e incertidumbre que aparecen al procesar información que conserva poco parecido con la disponible en su entrenamiento.

- *No son algorítmicas:* No se programan haciéndoles seguir una secuencia predefinida de instrucciones. Las RNA's generan sus propias reglas, para asociar la respuesta a su entrada, es decir, aprende por ejemplos y de sus propios errores.

- *Adaptabilidad:* Tienen la capacidad de ajustarse al cambio en las condiciones de operación. Por lo tanto, una red neuronal entrenada para operar en un ambiente específico puede ser reentrenable para adaptarse a cambios en las condiciones de operación.

- *Son estructuras altamente paralelas distribuidas:* Sus numerosas operaciones independientes pueden ser ejecutadas simultáneamente. Son ésta y la habilidad de generalización, las dos capacidades de procesamiento de información que hacen posible que las RNA's resuelvan problemas complejos que normalmente serían intratables.

Estas ventajas son resumen de algunas características del cerebro, tales como: la alta tolerancia a fallas, la flexibilidad, la capacidad de aprender con rapidez y generalizar, la alta velocidad de procesamiento y el buen manejo de información ruidosa e inconsistente. Todo esto ha hecho que las RNA's sean consideradas como una herramienta computacional multidisciplinaria, centrada principalmente en áreas como las matemática, estadística, física, neurociencia, ciencias de la computación e ingeniería, entre otras.

2.5.1 Comparación entre Redes Neuronales Biológicas (RNB's) y Redes Neuronales Artificiales (RNA's)

A continuación en la siguiente tabla se mostrará una comparación entre las Redes Neuronales Biológicas y las Redes Neuronales Artificiales:

Tabla 2.1: Comparación RNB's v/s RNA's.

Redes Neuronales Biológicas.	**Redes Neuronales Artificiales.**
Neuronas.	Unidades de Proceso.
Conexiones sinápticas.	Conexiones ponderadas.

Efectividad de las sinapsis.	Peso de las conexiones.
Efecto excitatorio o inhibitorio de una conexión.	Signo del peso de una conexión.
Efecto combinado de las sinapsis.	Función de propagación o de red.
Activación: Tasa de disparo.	Función de activación: Salida

2.6 Tipos de Redes Neuronales

Una RNA está organizada en capas, las cuales agrupan una serie de neuronas que se interconectan entre sí mediante las conexiones sinápticas. En general, tenemos tres tipos de capas: una capa de entrada donde se reciben las señales del ambiente, una capa de salida donde se emiten las señales al ambiente; y un conjunto de capas que se encuentran intermedias a la capa de entrada y salida, conocidas como capas ocultas y que no tienen contacto directo con el ambiente [7]. Así se pueden clasificar las RNA's en dos grupos:

2.6.1 Redes Estáticas

Son redes donde las neuronas de una capa se interconectan con las neuronas de la siguiente capa, desde la capa de entrada hasta la capa de salida. Además el flujo de información en las intercapas se propagan en una sola dirección o hacia delante (feedforward). En la siguiente figura se muestra un ejemplo de este tipo de arquitecturas.

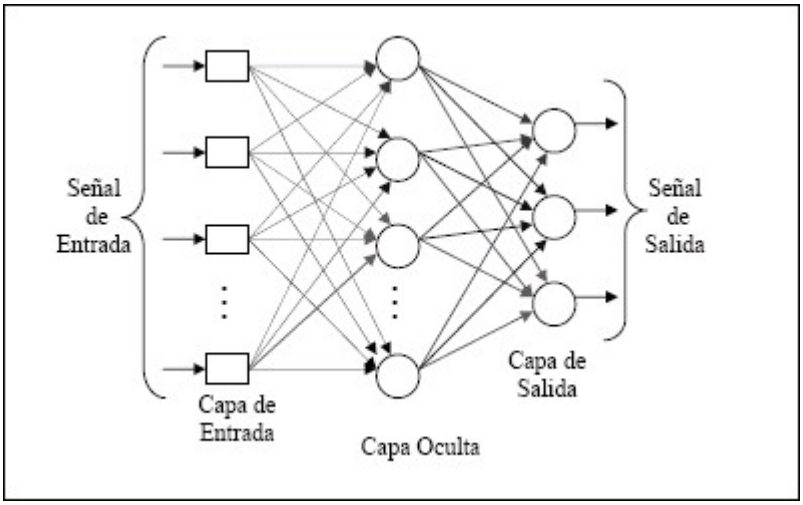

Figura 2.4: Esquema de una RNA Estática.

2.6.2 Redes Recurrentes

En víspera de la década de los 90', Williams y Zipser, desarrollaron el algoritmo de aprendizaje Real Time Recurrent Learning (RTRL), donde se utilizaba una red neuronal que sólo tiene una capa de entrada y de salida, pero que posee recurrencia o retroalimentación de sus salidas, esto da pie para la primera clasificación de redes neuronales. En segundo lugar, encontramos las redes tipo backpropagation, pero con la adición de retroalimentación desde la salida de la capa oculta hacia la entrada de esa misma capa oculta, cuyo desarrollador fue Elman en el año 1990. Dicha red también se denomina red parcialmente recurrente en términos de tiempo y espacio, es decir, tiene unidades de contexto que guardan los rendimientos de las unidades ocultas. Por último, y en tercer lugar, encontramos las redes neuronales de Jordan, estas son aquellas que realimentan la salida de la red a las neuronas de la capa oculta.

Toda esta clasificación cabe dentro de las denominadas Redes Neuronales Recurrentes: una Red Neuronal Recurrente (RNR), es aquella en la que existen lazos de realimentación en la red. Estos lazos pueden ser entre neuronas de diferentes capas, neuronas de la misma capa o, más sencillamente, entre una misma neurona, lo que le proporciona gran generalidad en su operación pues la salida de la red en un instante determinado, dependerá no sólo de las entradas actuales y de los pesos de conexión, sino también de las salidas ofrecidas por la red en instantes anteriores. Esta característica hace que las redes recurrentes muestren propiedades similares a la memoria humana, en la cual el estado de las salidas de las neuronas del cerebro depende, en parte, de entradas (estímulos) anteriores; dicho de otra forma, es un sistema capaz de reconstruir información exacta y completa a partir de unos datos de entrada incompletos, ruidosos o degradados. Esta estructura recurrente la hace especialmente adecuada para estudiar la dinámica de sistemas no lineales.

A continuación se presentan tres figuras, las cuales ilustran la clasificación de redes neuronales antes mencionada, estas son:

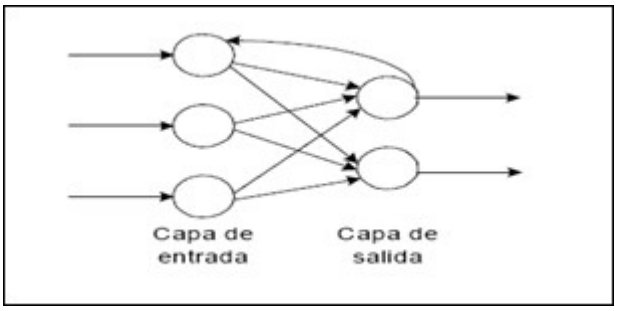

Figura 2.5: Primera clasificación: Red Neuronal Recurrente sólo con capa de entrada y salida, pero con recurrencia desde la salida hacia la entrada.

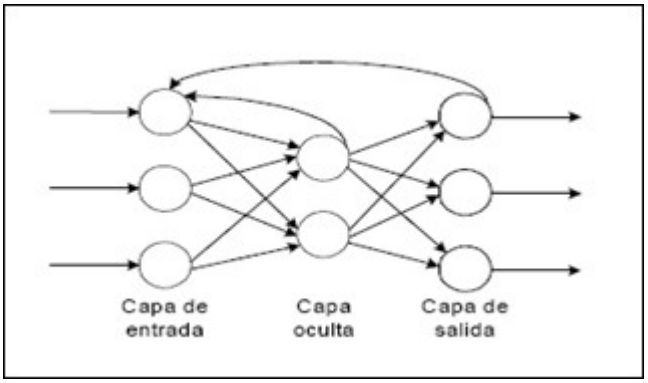

Figura 2.6: Segunda clasificación: Red Neuronal Recurrente con capa de entrada, oculta y de salida con recurrencia dentro de la capa oculta y desde la salida hacia la capa de entrada.

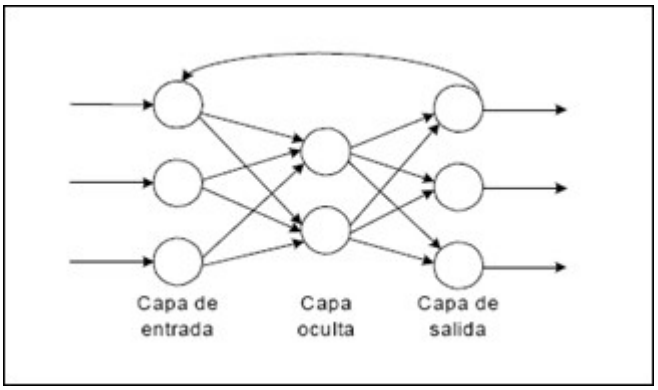

Figura 2.7: Tercera clasificación: Red Neuronal Recurrente con capa de entrada, oculta y de salida, con recurrencia desde la capa de salida hacia la entrada.

Asumiendo esta evolución no está demás destacar que el panorama sigue siendo alentador con respecto a las investigaciones y el desarrollo de las redes neuronales. En la actualidad, son

numerosos los trabajos que se realizan y publican cada año, las aplicaciones nuevas surgen (sobre todo en el área de control) y las empresas que lanzan al mercado productos nuevos, tanto hardware como software (sobre todo para simulación).

2.7 Perceptrones Multicapas

Las redes neuronales consisten de tres tipos de capas: una capa de entrada, una capa de salida y varias capas intermedias, las cuales contienen unidades o neuronas ocultas (estas neuronas no tienen contacto con el exterior). En la figura 2.8 podemos observar un perceptrón típico el cual posee una capa de entrada, una capa oculta y una de salida:

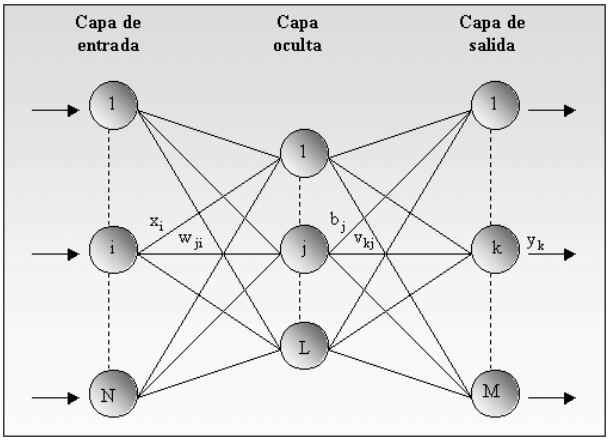

Figura 2.8: Perceptrón Multicapa.

Es a través de estas neuronas escondidas que el sistema puede presentar las abstracciones que no se pueden codificar directamente desde el ambiente por medio de los nodos de entrada. Esto significa que se puede realizar una representación interna en las unidades ocultas que no son parte de la entrada o salida.

Sin embargo, las entradas simplemente pasan a través de la primera capa hasta los nodos en la capa escondida sin ejecutar la función de activación y la función de salida que las neuronas normales se suponen llevan a cabo. Algunas personas consideran este tipo de red como una red de dos capas. Lo importante es notar que sólo los nodos de las capas ocultas y la capa de salida son los considerados como neuronas ordinarias, que ejecutan la función de activación y la función de salida. Se puede notar que en la estructura de la red “feedforward”, la señal de propagación se transmite solo desde la capa de entrada oculta, y desde la capa oculta a la capa

de salida. No se permiten las conexiones entre los nodos de una misma capa directamente o desde la capa de entrada a la de salida.

Para cada neurona *"k"* en la capa oculta y la neurona *"l"* en la capa de salida, la entrada a la red está dada por:

$$net_k = \sum_{g=1}^{Ng} w_{gk} O_g \qquad k = 1, \ldots, N_k \tag{2.4}$$

$$net_j = \sum_{K=1}^{Nk} w_{Kj} O_K \qquad j = 1, \ldots, N_j \tag{2.5}$$

Donde:

N_g: Número de neuronas en la capa de entrada.

N_k: Número de neuronas en la capa oculta.

N_j: Número de neuronas en la capa de salida.

O_i: Función de activación de la capa *i*.

j: Sub índice referido a las neuronas de la capa de salida.

k: Sub índice referido a las neuronas de la capa oculta.

g: Sub índice referido a las neuronas de la capa de entrada.

N_g: Número de neuronas en la capa de entrada.

La salida de las neuronas está dada por la entrada a la neurona pasada a través de la función de activación respectiva.

$$o_g = net_g \tag{2.6}$$

$$o_k = \frac{1}{1 + e^{-(net_k + b_k)}} = f_k(net_k + b_k) \tag{2.7}$$

$$o_j = \frac{1}{1 + e^{-(net_j + b_j)}} = f_j(net_j + b_j) \tag{2.8}$$

Donde

net_g

es la señal desde las fuentes externas al nodo *g* en la capa de entrada.

Los perceptrones multicapas han sido aplicados en un rango de tareas que van más allá de simples decisiones y reconocimiento de patrones. Por ejemplo, se puede entrenar una red para formar el tiempo pasado de los verbos en inglés, leer textos y manuscritos. Es el instrumento predilecto usado para problemas de predicción de una serie de datos en el tiempo; como es el caso de la medición de la demanda de electricidad. Además son capaces de predecir cambios en el valor de los instrumentos y mercados financieros.

Un Perceptrón multicapas posee tres características distintivas:

- *El modelo de una neurona incluye una función de activación no lineal:* Esta función debe ser continua, acotada y diferenciable; de forma que el ajuste de los pesos o entrenamiento se pueda realizar utilizando mecanismos de descenso de gradiente. Una función comúnmente utilizada y que satisface estos requerimientos es la función sigmoidal.

- *La red contiene uno o más capas de neuronas ocultas:* Estas neuronas ocultas capacitan a la red para aprender tareas complejas por la extracción progresiva de aspectos más significativos de los patrones de entrada (vectores). Además, permiten que la red sea capaz de extraer altos órdenes estadísticos, es decir, en un sentido bastante general, la red adquiere una perspectiva global (a pesar de sus conexiones locales) debido al grupo extra de conexiones sinápticas y a la dimensión extra de interacciones de las neuronas. La habilidad de las neuronas intermedias en extraer altos órdenes estadísticos es particularmente importante cuando el tamaño de la capa de entrada es grande.

- *La red exhibe un alto grado de conectividad,* determinada por las conexiones sinápticas (entre neuronas) de la red. Un cambio en la conectividad de la red requiere un cambio en las conexiones sinápticas o en los pesos.

Es a través de la combinación de estas características, junto con la habilidad de aprender por medio del entrenamiento, que el perceptrón multicapas posee una alta capacidad computacional. Sin embargo, son estas mismas características las responsables de complicar el análisis de esta topología para entender completamente el comportamiento de la red y así obtener el máximo provecho. La presencia de una forma no lineal distribuida en la alta conectividad de la red hace el análisis teórico de un perceptrón multicapa difícil de comprender. Además, el proceso de aprendizaje se hace aún más difícil porque la búsqueda, (del aprendizaje) debe ser conducida en un espacio más grande de funciones posibles, y se debe realizar una selección entre las representaciones alternativas del patrón de entrada.

Capítulo 3
Algoritmos de Aprendizaje

3.1 Algoritmo Backpropagation (BP)

El algoritmo backpropagation es un algoritmo de aprendizaje supervisado utilizado para entrenar redes neuronales artificiales, normalmente no retroalimentadas (aunque también puede usarse en éstas). Al ser un algoritmo de aprendizaje supervisado es necesario conocer las salidas consideradas correctas para el conjunto de valores de entrada que se usarán como prueba normalmente, los pesos iniciales de las entradas de cada neurona son inicializados aleatoriamente.

El BP estándar es un algoritmo de gradiente descendente, en el cual los pesos de la red son movidos a lo largo del negativo del gradiente de la función de error. En el podemos considerar, por un lado, una etapa llamada hacia delante donde se presenta, ante la red, un patrón de entrada y éste se transmite a través de las sucesivas capas de neuronas hasta obtener una salida y, por otro lado, una etapa hacia atrás donde se modifican los pesos de la red de manera que coincida la salida deseada por el usuario con la salida obtenida por la red ante la presentación de un determinado patrón de entrada o con un error aceptable.

El término backpropagation se refiere a la manera como el gradiente es calculado para redes multicapa no lineales. El error se propaga así hacia atrás corrigiendo los valores de los pesos hasta que se alcance el criterio de parada establecido, normalmente basado en un cierto número (elevado) de iteraciones o un error admisible. La superficie de error tiene varios mínimos locales lo que hace que su ejecución pueda quedar obstaculizada en ellos y por ende que el aprendizaje fracase, es frecuente repetir el aprendizaje o cambiar los valores de entrada para conseguir una convergencia. En la actualidad existen un cierto número de variaciones en el algoritmo básico, las cuales están basadas en otras técnicas de optimización, tales como el gradiente conjugado y los métodos de Newton.

Aunque el comportamiento de este algoritmo es bien conocido, éste requiere el cálculo de la primera derivada de la función de error cuadrático con respecto a todos los pesos de la red, lo cual, dada la configuración del perceptrón y otras redes neuronales implica el empleo de la regla de la cadena, ya que el error de salida no es una función explícita de los pesos de la capa de entrada, lo que, en general, representa una alta complejidad computacional dado el gran número de multiplicaciones y sumas requeridas, así como la evaluación de funciones sigmoidales presentes en cada nodo [8]. Esto sugiere la necesidad de desarrollar mecanismos que permitan la estimación de las derivadas del error cuadrático medio con respecto a los pesos de la red, requeridas durante la etapa de entrenamiento de las redes neuronales, con una complejidad computacional. La red neuronal artificial perceptrón multicapa usando el algoritmo de retropropagación ha sido aplicada a la solución de diversos problemas prácticos [8,9].

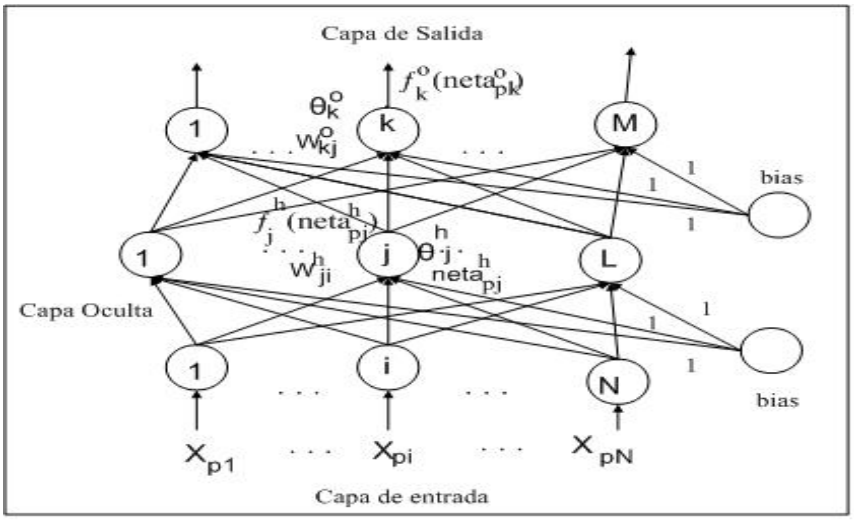

Figura 3.1: Arquitectura de una Red Backpropagation.

Para entender como el algoritmo obtiene la actualización de los pesos, comencemos revisando las ecuaciones para el procesamiento de la información que hay en la red de la figura 3.1. Se aplica un vector de entrada

$$X_p = (X_{1p}, X_{2p}, \ldots, X_{np})$$

en la capa de entrada de la red. Las unidades de entrada distribuyen los valores a las unidades de la capa oculta. La entrada neta de la j-ésima unidad oculta es:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h} \quad (3.1)$$

En donde w_{ji}^{h} es el peso de la conexión procedente de la i-ésima unidad de entrada y θ_{j}^{h} es el término de tendencia o bias. El índice h hace referencia a unidades de la capa oculta. La salida de este nodo es:

$$i_{pj} = f_{j}^{h}(net_{pj}^{h}) \quad (3.2)$$

En donde w_{ji}^{h} es el peso de la conexión procedente de la i-ésima unidad de entrada y θ_{j}^{h} es el término de tendencia o bias. El índice h hace referencia a unidades de la capa oculta. La salida de este nodo es:

$i_{pj} = f_{j}^{h}(net_{pj}^{h})$ (3.2)

Las ecuaciones para los nodos de salida son las siguientes:

$$net_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} y_{pj} + \theta_{k}^{0} \quad (3.3)$$

$$y_{pk} = f_{k}^{0}(net_{pk}^{o}) \quad (3.4)$$

En donde el índice “0” se refiere a magnitudes de la capa de salida.

La actualización es diferente dependiendo de si la capa es de salida u oculta. Para el caso de las unidades de la capa de salida, dado que en esta capa puede haber muchas unidades, definiremos el error de una sola unidad de salida en la forma $\delta_{pk} = (d_{pk} - y_{pk})$, en donde el subíndice *p* se refiere al k-ésimo vector de entrenamiento y *k* a la k-ésima unidad de salida. En este caso d_{pk} es el valor de salida deseado e y_{pk} es la salida obtenida. El error que se minimiza es la suma de los cuadrados de los errores de todas las unidades de salida:

$$E_{p} = \frac{1}{2} \sum_{k=1}^{M} \delta_{pk}^{2} \quad (3.5)$$

Si tomamos una variación proporcional al gradiente de una función de error *E(w)*, se tiene que:

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) - \alpha \nabla E(w) \tag{3.6}$$

Por lo tanto la variación de los pesos sinápticos será proporcional al gradiente de la función error

$$\Delta w_{kj}^{o} = -\alpha \frac{\partial E_p}{\partial w_{kj}^{o}} \tag{3.7}$$

Desarrollando el segundo término del lado derecho de la ecuación (3.7) se obtiene lo siguiente:

$$\frac{\partial E_p}{\partial w_{kj}^{o}} = -(d_{pk} - y_{pk}) \frac{\partial f_k^{o}}{\partial (net_{pk}^{o})} \frac{\partial (net_{pk}^{o})}{\partial w_{kj}^{o}} \tag{3.8}$$

El último factor de la ecuación (3.8) es:

$$\frac{\partial (net_{pk}^{o})}{\partial w_{kj}^{o}} = \frac{\partial}{\partial w_{kj}^{o}} \left(\sum_{j=1}^{I} w_{kj}^{o} I_{pj} + h_k^{o} \right) = i_{pj} \tag{3.9}$$

Combinando las ecuaciones (3.7) y (3.8) tenemos lo siguiente para el gradiente negativo:

$$-\frac{\partial E_p}{\partial w_{kj}^{o}} = (d_{pk} - y_{pk}) f_k^{o'} (net_{pk}^{o}) i_{pj} \tag{3.10}$$

Por consiguiente la ecuación (3.7) puede escribirse como:

$$\Delta w_{kj}^{o} = \alpha (d_{pk} - y_{pk}) f_k^{o'} (net_{pk}^{o}) i_{pj} \tag{3.11}$$

El factor α se denomina factor de aprendizaje, por el momento es positivo y menor que 1.

La función *f* debe ser derivable. En general disponemos de dos formas de función de salida:

La función lineal: $f_k^o(net_{jk}^o) = net_{jk}^o$

La función sigmoidal: $f_k^o(net_{jk}^o) = \dfrac{1}{1+e^{-net_{jk}^o}}$

Para el primer caso la función $f_k^{o'}$=1, en el segundo caso $f_k^{o'} = f_k^{o'}(1- f_k^{o'}) = y_{pk}(1- y_{pk})$ resultando la ecuación (3.11).

$\Delta w_{ji} = \alpha(d_{pk} - y_{pk})i_{pj}$ para la salida lineal.

$\Delta w_{ji} = \alpha(d_{pk} - y_{pk})y_{pk}(1 - y_{pk})i_{pj}$ para la salida sigmoidal.

Comúnmente se define una magnitud o sensitividad (de capa oculta o de salida):

$$\delta_{pk}^o = (d_{pk} - y_{pk}) f_k^{o'}(net_{pk}^o) \tag{3.12}$$

Entonces se puede escribir la ecuación de actualización de pesos en la forma:

$$\Delta w_{ji} = \alpha \delta_{pk}^o i_{pj} \tag{3.13}$$

Independiente de la forma de la función de salida.

Aún queda por determinar la actualización de los pesos de las capas ocultas. Surge un problema cuando se intenta determinar una medida del error de las salidas para las unidades de la capa oculta. Se sabe cuál es la salida obtenida, pero no tenemos forma de saber por anticipado cual debería ser la salida para estas unidades. La ecuación del error E_p demuestra que está relacionado con los valores de salida de la capa oculta.

$$\begin{aligned} E_p &= \frac{1}{2}\sum_{k=1}^{M}(d_{pk} - y_{pk})^2 \\ &= \frac{1}{2}\sum_{k=1}^{M}(d_{pk} - f_k^o(net_{jk}^o))^2 \\ &= \frac{1}{2}\sum_{k=1}^{M}(d_{pk} - f_k^o(\sum_{i=1}^{N} w_{kj}^o i_{pj} + \theta_k^o))^2 \end{aligned}$$

Sabemos que i_{pj} depende de los pesos de las capas ocultas a través de las ecuaciones (3.1) y (3.2). Para calcular el gradiente de E_p respecto de las capas ocultas se aprovecha de esta relación.

$$\frac{\partial E_p}{\partial w_{ji}^h} = \frac{1}{2}\sum_k \frac{\partial (d_{pk} - y_{pk})^2}{\partial w_{ji}^h}$$

$$= -\sum_k (d_{pk} - y_{pk}) \frac{\partial o_{pk}}{\partial net_{pk}^o} \frac{\partial net_{pk}^o}{\partial i_{pj}} \frac{\partial i_{pj}}{\partial net_{pj}^h} \frac{\partial i_{pj}}{\partial w_{ji}^h} \tag{3.14}$$

A partir de ecuaciones anteriores la ecuación (3.14) puede escribirse como:

$$\frac{\partial E_p}{\partial w_{ji}^h} = -\sum_k ((d_{pk} - y_{pk}) f_k^{o'}(net_{pk}^o) w_{kj}^o)\ f_j^{h'}(net_{pj}^h) x_{pi} \tag{3.15}$$

A partir de las ecuaciones anteriores, los pesos de las capas ocultas pueden ser actualizados proporcionalmente al valor negativo del gradiente como se muestra en la siguiente ecuación:

$$\Delta_p w_{ji}^h = \alpha f_j^{h'}(net_{pj}^h) x_{pi} \sum_k (d_{pk} - y_{pk}) f_k^{o'}(net_{pk}^o) w_{kj}^o \tag{3.16}$$

En donde α es la tasa de aprendizaje y podemos utilizar la definición de

$$\delta_{pk}^o$$

para reescribir la ecuación (3.16):

$$\Delta_p w_{ji}^h = \alpha f_j^{h'}(net_{pj}^h) x_{pi} \sum_k \delta_{pk}^o w_{kj}^o \tag{3.17}$$

En esta ecuación queda de manifiesto que las actualizaciones de pesos de la capa oculta dependen de todos los términos de error,

$$\delta_{pk}^o$$

de la capa de salida. Debido a este resultado surge la noción en el algoritmo de propagación hacia atrás, los errores conocidos de la capa de salida se propagan hacia atrás, hacia la capa

oculta, para determinar los cambios de peso adecuados en esa capa. Si se define un término de error para la capa oculta.

$$\delta_{pj}^{h} = f_j^{h'}(net_{pj}^{h}) \sum_k \delta_{pk}^{o} w_{kj}^{o} \qquad (3.18)$$

Y por lo tanto la actualización de los pesos de las capas ocultas se puede escribir como:

$$\Delta w_{ji}^{h} = \alpha \delta_{pj}^{h} x_{pi} \qquad (3.19)$$

A modo de resumen se describirán los pasos más importantes para actualizar los pesos del algoritmo:

Paso 1: Inicializar los pesos de la red con los valores pequeños aleatorios.

Paso 2: Presentar un patrón de entrada $X_p = (X_{1p}, X_{2p}, ..., X_{np})$ y especificar la salida deseada que debe generar la red.

Paso 3: Calcular la salida actual de la red. Para ello presentamos las entradas a la red y vamos calculando la salida que presenta cada capa hasta llegar a la capa de salida, ésta será la salida de la red. Los pasos son los siguientes:

- Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada. Para una neurona *j* oculta:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_j^{h}$$

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

- Se calculan las salidas de las neuronas ocultas:

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

- Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida:

$$net_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} y_{pj} + \theta_k^0$$

$$y_{pk} = f_k^0(net_{pk}^{o})$$

Paso 4: Calcular los términos de error para todas las neuronas. Si la neurona *k* es una neurona de la capa de salida, el valor de la delta es:

$$\delta_{pk}^{0} = (d_{pk} - y_{pk}) f_k^{o'}(net_{pk}^{o})$$

La función *f* debe ser derivable. En general disponemos de dos formas de función de salida:

La función lineal: $f_k(net_{jk}) = net_{jk}$

La función sigmoidal: $f_k(net_{jk}) = \dfrac{1}{1+e^{-net_{jk}}}$

La selección de la función depende de la forma que se decida representar la salida: si se desea que las neuronas de salida sean binarias, se utiliza la función sigmoidal, en otros casos, la lineal.

Para una función lineal, tenemos: $f_k^{o'} = 1$, mientras que la derivada de una función sigmoidal es: $f_k^{o'} = f_k^{o}(1 - f_k^{o}) = y_{pk}(1 - y_{pk})$ por lo que los términos de error para las neuronas de salida quedan:

$\delta_{pk}^{o} = (d_{pk} - y_{pk})$ para la salida lineal.

$\delta_{pk}^{o} = (d_{pk} - y_{pk}) y_{pk}(1 - y_{pk})$ para la salida sigmoidal.

Si la neurona *j* no es de salida, entonces la derivada parcial del error no puede ser evaluada directamente, por tanto se obtiene el desarrollo a partir de valores que son conocidos y otros que pueden ser evaluados. La expresión obtenida en este caso es:

$$\delta_{pj}^{h} = f_j^{h'}(net_{pj}^{h}) \sum_k \delta_{pk}^{0} w_{kj}^{0}$$

donde observamos que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término propagación hacia atrás (backpropagation).

Paso 5: Actualización de los pesos: Para ello utilizamos un algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la siguiente forma:

- Para los pesos de las neuronas de la capa de salida:

$$w_{ij}^{o}(t+1) = w_{ij}^{o}(t) + \Delta w_{ij}^{o}(t)$$

$$\Delta w_{ij}^{o}(t) = \alpha \delta_{pk}^{o} y_{pj}$$

- Para los pesos de las neuronas de la capa oculta:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \Delta w_{ji}^{h}(t)$$

$$\Delta w_{ji}^{h}(t+1) = \alpha \delta_{pj}^{h} x_{pi}$$

En ambos casos, para acelerar el proceso de aprendizaje se puede añadir el término *momento*.

Paso 6: El proceso se repite hasta que el término de error

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta_{pk}^{2}$$

resulta aceptablemente pequeño para cada uno de los patrones aprendidos.

Se debe tener en cuenta las siguientes consideraciones:

El algoritmo encuentra un valor mínimo de error (local o global) mediante una aplicación de pasos (gradiente) descendentes. Cada punto de la superficie de la función corresponde a un conjunto de valores de los pesos de la red. Con el gradiente descendente, siempre que se realiza un cambio en todos los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el sector más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local de error.

Uno de los problemas del algoritmo es que en busca de minimizar la función de error, puede caer en un mínimo local o en algún punto estacionario, con lo cual no se llega a encontrar el

mínimo global de la función de error. Sin embargo, no tiene porqué alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo preestablecido.

En las técnicas de gradiente decreciente es conveniente avanzar por la superficie del error con incrementos de pesos pequeños. Esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o cerca que se está del punto mínimo. Con incrementos grandes se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarda más en llegar, se evita que ocurra esto.

El incremento del paso adecuado influye en la velocidad de convergencia del algoritmo. La velocidad se controla con la tasa de aprendizaje α. Normalmente α, debe ser un número pequeño (del orden de 0,05 a 0,25), para asegurar que la red llegue a asentarse en una solución.

Lo habitual es aumentar el valor de α a medida que disminuye el error de la red durante la fase de aprendizaje. Así aceleramos la convergencia aunque sin llegar nunca a valores de α demasiado grandes, que hicieran que la red oscilase alejándose del mínimo.

En la práctica, si una red deja de aprender antes de llegar a una solución aceptable, se realiza un cambio en el número de neuronas ocultas o en los parámetros de aprendizaje, o simplemente, se vuelve a empezar con un conjunto distinto de pesos originales y se suele resolver el problema.

La popularidad de las redes backpropagation se debe principalmente a que es capaz de actuar como un aproximador universal de funciones [10]. Más concretamente, una red conteniendo al menos una capa oculta con suficiente unidades no lineales puede aprender cualquier tipo de función o relación continua entre un grupo de variables de entrada y salida. Esta propiedad convierte a las redes perceptrón multicapa en herramientas de propósito general, flexibles y no lineales; mostrando un rendimiento superior respecto a los modelos estadísticos clásicos en numerosos campos de aplicación.

Los problemas que aparecen al utilizar backpropagation son:

Mínimos locales: Surge cuando la solución ha dado un mínimo local y no un mínimo absoluto como sería de desear. Según los valores iniciales de los pesos y para varios

entrenamientos se obtendrán distintos resultados. En definitiva, obliga a repetir varias veces el entrenamiento.

Parálisis de la red: Se produce cuando las variaciones de los pesos tras sucesivas iteraciones son nulas. De esta forma, los valores de los pesos no son apenas modificados, aun cuando se está lejos de la solución óptima.

Otros problemas que pueden aparecer son los de estabilidad. Si la variación incremental en cada iteración del peso en cuestión es demasiado grande puede llegarse a la inestabilidad, pero una variación muy pequeña implicaría lentitud en la búsqueda del mínimo.

Otros problemas que deben ser tenidos muy en cuenta en el entrenamiento de las redes neuronales son:

Bajoentrenamiento (underfitting): Una red no es suficientemente compleja podría fallar para aprender la señal completa de un conjunto de datos complicado.

Sobreentrenamiento (overfitting): Una red demasiado compleja podría ajustar el ruido, no solamente la señal. La red debería aprender solamente la estructura general de los datos de entrenamiento.

3.2 Algoritmo Levenberg Marquardt (LM)

Como hemos mencionado anteriormente, existe otro tipo de algoritmos cuya operación se concentra en calcular matrices Hessianas que agreguen un segundo orden al algoritmo de propagación hacia atrás. Es decir que la matriz Hessiana es la matriz de derivadas de segundo orden de la función de costo, en los valores actuales de pesos para la búsqueda de la dirección óptima del gradiente.

Sin embargo, este tipo de procedimiento implica cálculos demasiado complejos, ya que el cálculo de la matriz Hessiana toma mucho tiempo de procesador, y es proporcional al número de parámetros de la red. Por lo que para redes de gran tamaño, es poco recomendable el uso de estos algoritmos. A estos algoritmos se les conoce como del tipo quasi-newton, debido a que se basan en el método de Newton para optimizar la dirección de cambio.

En base a estos algoritmos, se desarrolló también un algoritmo conocido como Levenberg Marquardt, el cual también busca hallar la dirección de cambio en base a derivadas de segundo orden, pero sin requerir el cálculo del Hessiano, sino de una aproximación de éste. De esta forma LM, puede aproximar al método de Newton en cuanto se encuentre avanzando en la dirección correcta, y por otro lado puede aproximar al método del gradiente descendiente para cuando viaja en dirección incorrecta permitiendo frenar al algoritmo, ya que como se sabe, el algoritmo del gradiente descendiente es más lento que los algoritmos de optimización de dirección y magnitud de cambio.

En la práctica este algoritmo obtiene el mejor desempeño, frente a los algoritmos antes mencionados, para los problemas de aproximación de funciones y ajustes de curvas, la razón es que este algoritmo está diseñado para trabajar con la función de costo de media de errores cuadrados que es una función aproximadamente lineal. No obstante, su desempeño es pobre para la clasificación de patrones al igual que los métodos quasi-newton.

LM es una modificación del método de Newton, el que fue diseñado para minimizar funciones que sean la suma de los cuadrados de otras funciones no lineales; es por ello, que tiene un excelente desempeño en el entrenamiento de redes neuronales donde el rendimiento de la red esté determinado por el error cuadrático medio.

Al igual que en los métodos quasi-newton, LM tiene el propósito de entrenar mediante un gradiente de segundo orden sin tener que computar la matriz Hessiana.

Cuando la función de minimización tiene la forma de una suma de cuadrados, como ocurre normalmente en el entrenamiento de redes feedforward, entonces la matriz Hessiana puede ser aproximada como:

$$H = J^T \cdot J \tag{3.20}$$

Y el gradiente puede obtenerse:

$$g = J^T \cdot e \tag{3.21}$$

Donde J es la matriz Jacobiana, la cual contiene las primeras derivadas de los errores de la red con respecto a los pesos y bias, y e es el vector de errores de la red.

El algoritmo usa la aproximación de dicha matriz para calcular los pesos y añade un incremento μ para evitar la singularidad de la matriz la cual podría causar la parálisis de la red:

$$X_{k+1} = X_k - \left[J^T \cdot J + \mu \cdot I\right]^{-1} J^T \cdot e \qquad (3.22)$$

Cuando el escalar μ es cero, el algoritmo es idéntico a un método Newton usando la aproximación de la matriz Hessiana. A medida que μ aumenta de valor, el algoritmo se parece más al típico de descenso de gradiente con un tamaño del peso pequeño. El objetivo es hacer que el algoritmo adquiera el carácter de un método de Newton, ya que es mucho más rápido y exacto que uno de descenso de gradiente. De esta forma, μ se reduce de valor en cada iteración si se produce un descenso en la función de minimización y es incrementado en caso contrario. Así la función a minimizar siempre se va reduciendo en su valor.

El algoritmo alcanza convergencia cuando la norma del gradiente de la suma de funciones cuadráticas sea menor que un valor predeterminado o cuando la suma de los errores cuadrados alcance el valor trazado como meta.

LM es el más rápido de todos los algoritmos de entrenamiento descritos, y especialmente indicado para redes neuronales de tamaño moderado (varios cientos de pesos). El principal inconveniente de este algoritmo es su elevado uso de memoria para su implementación. Las dimensiones de la matriz Jacobiana son de $Q \times N$, donde Q es el número de muestras utilizadas para el entrenamiento y N es el número total de pesos y bias de la red.

LM es una técnica de segundo orden para resolver problemas de optimización que suele ser más eficiente que el método clásico del descenso por gradiente aunque requiere más memoria. Hagan y Menhaj en [11] lo han aplicado a la determinación de los pesos sinápticos en un perceptrón multicapa.

Para entender como el algoritmo obtiene la actualización de los pesos, supongamos una red con q unidades de salida en la que se tratará de minimizar la función:

$$E(w) = \frac{1}{2}\sum_{i=1}^{q} e^2{}_i(w) = \frac{1}{2} e(w)^T e(w) \tag{3.23}$$

Donde:

$$e(w)^T = (e_1(w),..., e_q(w)) \tag{3.24}$$

$$e_1(w) = (d_1 - y_1) \qquad \forall\ i = 1,2,...,q \tag{3.25}$$

Puede ser demostrado según [15] que el gradiente de la función $E(w)$ tiene la siguiente forma:

$$\nabla E(w) = \sum_{i=1}^{q} e_i(w)\nabla e_i(w) \equiv J^T(w)e(w) \tag{3.26}$$

Y la matriz Hessiana viene dada por la siguiente ecuación:

$$\nabla^2 E(w) = \sum_{i=1}^{q}\left[\nabla e_i(w)(\nabla e_i(w))^T + e_i(w)\nabla^2 e_i(w)\right] \equiv J^T(w)J(w) + S(w) \tag{3.27}$$

Donde $J(w)$ es la matriz Jacobiana:

$$J(w) = \begin{bmatrix} \frac{\partial e_1(w)}{\partial w_1} & \frac{\partial e_1(w)}{\partial w_2} & \cdots & \frac{\partial e_1(w)}{\partial w_n} \\ \frac{\partial e_2(w)}{\partial w_1} & \frac{\partial e_2(w)}{\partial w_2} & \cdots & \frac{\partial e_2(w)}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_q(w)}{\partial w_1} & \frac{\partial e_q(w)}{\partial w_2} & \cdots & \frac{\partial e_q(w)}{\partial w_n} \end{bmatrix} \tag{3.28}$$

Y

$$S(w) = \sum_{i=1}^{q} e_i(w)\nabla^2 e_i(w) \tag{3.29}$$

El término *S(w)* es la parte de

$$\nabla^2 E(w)$$

que es una función de las segundas derivadas de *e(w)*. Para el método de Newton se asume $S(w) \approx 0$ [12], y la actualización del método de Newton se expresa en las siguientes ecuaciones:

$$\Delta w = -[\nabla^2 E(w)]^{-1} \nabla E(w) \tag{3.30}$$

$$\Delta w = [J^T(w) J(w)]^{-1} J^T(w) e(w) \tag{3.31}$$

Como LM es una aproximación al método de Newton, la actualización quedaría:

$$\Delta W = [J^t(w) J(w) + \mu I]^{-1} J^t(w) e(w) \tag{3.32}$$

Donde *I* es la matriz identidad y el parámetro μ se incrementa o disminuye en cada paso:

• Si $E_{k+1}(w) \leq E_k(w)$, entonces en la siguiente iteración el parámetro μ se divide por algún factor β.

• Si $E_{k+1}(w) \geq E_k(w)$, entonces en la siguiente iteración el parámetro μ se multiplica por el factor β.

Se suele tomar como punto de partida $\mu=0.01$ y $\beta=10$.

El paso clave en este algoritmo es el cálculo de la matriz Jacobiana. Los términos de esta matriz pueden ser calculados con una simple modificación de BP, el que calcula los términos de actualización según la siguiente ecuación:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial \sum_{i=1}^{q} e^2{}_i(w)}{\partial w_{ij}^k} \tag{3.33}$$

Para los elementos de la matriz Jacobiana que son necesarios para LM, necesitamos calcularlos como:

$$\frac{\partial e_i(w)}{\partial w_{ij}^k} \tag{3.34}$$

Esos términos pueden ser calculados usando el BP estándar con una modificación en la capa de salida:

$$\Delta w_{ij}^{o} = -F'^{q}(n^{q}) \tag{3.35}$$

Donde:

$$F'^{k}(n^{k}) = \begin{bmatrix} f'^{k}(n^{k}(1)) & 0 & \cdots & 0 \\ 0 & f'^{k}(n^{k}(2)) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & f'^{k}(n^{k}(q)) \end{bmatrix} \tag{3.36}$$

A modo de acotación es importante observar que cada columna de la matriz (3.36) es un vector de sensitividades que deben ser propagados a través de la red para producir una fila de la matriz Jacobiana.

La iteración del LM puede ser descrita como sigue:

Dados $\{p_1, t_1\}, \{p_2, t_2\}, \ldots\ldots, \{p_q, t_q\}$ donde p_q es la entrada a la red y t_q es la correspondiente salida deseada.

1.- Al presentar todas las entradas a la red, se calculan las correspondientes salidas y cada uno de los errores según:

La *net* de entrada a la unidad *i* en la capa *k+1* es:

$$net^{k+1}(i) = \sum_{j=1}^{S_k} w^{k+1}(i,j) y^{k}(j) + b^{k+1}(i) \tag{3.37}$$

Donde S_k es el número de neuronas que contribuyen a la entrada de la neurona *i*.

La salida de la unidad i de la capa k+1 será:

$$y^{k+1}(i) = f^{k+1}(net^{k+1}(i)) \tag{3.38}$$

Para una red de *M* capas el sistema de ecuaciones en forma de matriz será:

$$y^{0} = p_1 \tag{3.39}$$

$$y^{k+1} = f^{k+1}(w^{k+1} y^{k} + b^{k+1}) \quad \text{k=0,1……..M-1} \tag{3.40}$$

Donde y^0 es la salida de la capa de entrada y p_1 es la entrada a la red, y^{k+1} es la salida de la capa $k+1$, f^{k+1} es la función de transferencia de la capa $k+1$ y w^{k+1} es la matriz de pesos de la misma capa y b^{k+1} es el vector de bias de la capa $k+1$ e y^k es la salida de la capa previa.

Calcular los errores de la capa salida según:

$$e_q = t_q - y_q \tag{3.41}$$

Donde y es la salida obtenida en la última capa.

Calcular la suma de los cuadrados de los errores de todas las entradas según:

$$E(w) = \sum_{i=1}^{q} e_i^2(w) \tag{3.42}$$

2.- Se calculan las sensitividades individuales y con éstas, se calculan los elementos de la matriz Jacobiana.

3.- Resolver la ecuación (3.31) para obtener Δw.

4.- Se vuelve a calcular la suma de los errores cuadrados usando $w_{kj}(t+1)=w_{kj}(t)+\Delta w_{kj}(t)$. Si esta nueva suma es más pequeña que el valor calculado en el paso 1, entonces se divide μ por β, se calcula $w_{kj}(t+1)=w_{kj}(t)+\Delta w_{kj}(t)$ y se regresa al paso 1. Si la suma no se reduce entonces se multiplica μ por β y se regresa al paso 3.

El algoritmo debe alcanzar convergencia cuando la norma del gradiente (ecuación 3.26) sea menor que algún valor predeterminado, o cuando la suma de los errores cuadrados ha sido reducida a un error que se haya trazado como meta.

Capítulo 4

Discusión de Resultados

Una vez finalizado el estudio teórico se procede a la implementación de la arquitectura propuesta, la cual dará paso a la obtención de los resultados necesarios para demostrar la validez del modelo de predicción. Para esto se utilizará el software de análisis numérico denominado MATLAB (Matriz Laboratory), creado en el año 1984, el cual sirve para simular sistemas, como así también es de utilidad para áreas tales como: economía, lógica difusa, inteligencia artificial, etc, [13].

El pre-proceso de la data de aprendizaje no forma parte de esta memoria de título, debido a que los datos ya han sido suavizados utilizando una teoría Wavelet [14,15].

4.1 Arquitectura con suavizado

Teniendo como base el tamaño de la muestra o data que se consideró para la simulación fue de un 75%, manteniendo este parámetro constante a lo largo de todo el proceso de ajuste para la elección de la mejor topología, y considerando que la data de la simulación se encuentra suavizada, se inicia el proceso de ajuste con la elección de número de nodos en la capa oculta de la red.

Elección número de nodos capa oculta.

El primer método de selección de la arquitectura ideal tiene relación con el ajuste del número de nodos en la capa oculta. Para esto se trabajó con los siguientes parámetros de forma constante:

Tabla 4.1: Parámetros fijos para calibrar capa oculta.

Muestra	=	75%	(Porcentaje del total de la data).

Número de épocas	=	200.	
τ	=	5	(Desfase mensual).
γ	=	2E-3	(Factor de penalidad para el ajuste de los pesos).

El número de nodos de la capa oculta se fue ajustando según el siguiente conjunto {*2, 4,…….20}*. Cada ajuste en la elección de la topología ideal se realizó utilizando la simulación de Monte Carlo (MC), la cual es una técnica cuantitativa que hace uso de la estadística para imitar, mediante modelos matemáticos, el comportamiento aleatorio de sistemas reales no dinámicos (por lo general, cuando se trata de sistemas cuyo estado va cambiando con el paso del tiempo, se recurre bien a la simulación de eventos discretos o bien a la simulación de sistemas continuos) [13].

Una vez identificados las entradas o variables aleatorias, se lleva a cabo un experimento consistente en:

1. Generar con ayuda del computador muestras aleatorias (valores concretos) para dichas entradas.

2. Analizar el comportamiento del sistema ante los valores generados. Tras repetir *n* veces este experimento, dispondremos de *n* observaciones sobre el comportamiento del sistema, lo cual nos será de utilidad para entender el funcionamiento del mismo, obviamente nuestro análisis será tanto más preciso cuanto mayor sea el número de *n* experimentos que llevemos a cabo. Para nuestro caso *n*=10*,* lo que corresponde a un total de 10 iteraciones por cada arquitectura evaluada.

Cabe destacar que el método de simulación de MC, fue utilizado a lo largo de todas las etapas de ajuste para poder encontrar la topología ideal.

A continuación se detallan los resultados obtenidos al efectuar el ajuste de nodos. Considerando las siguientes métricas: desviación estándar *(STD),* máximo *(MAX),* mínimo *(MIN),* media, R^2, las cuales son definidas como:

Desviación Estándar. (STD): $$\sigma^2 = \frac{\sum_{i=1}^{n} \left(X_i - \overline{X} \right)}{n-1} \quad (4.1)$$

Media: $$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \quad (4.2)$$

Factor de correlación: $$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2}{\sum_{i=1}^{n} \left(y_i - \overline{y} \right)^2} \quad (4.3)$$

Tabla 4.2: Evaluación rendimiento del ECM con: Número de épocas=200, τ=5 y γ= 2E-3.

N° Nodos Métricas	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
STD	3,21	2,24	2,13	2,10	1,71	2,35	1,43	0,89	1,25	1,12
MAX	16,04	19,40	21,08	24,40	25,00	31,27	28,64	30,23	32,36	33,82
MIN	5,42	13,43	13,96	17,20	20,04	23,03	24,73	27,79	27,89	30,32
MEDIA	10,61	16,13	17,45	20,80	22,02	25,60	26,23	28,95	30,24	31,76

Tabla 4.3: Evaluación rendimiento de R^2 de Training con: Número de épocas=200, τ=5 y γ= 2E-3.

N°	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**

Nodos **Métricas**										
STD	30,49	20,13	3,91	7,31	2,18	1,23	17,61	1,39	1,22	5,19
MAX	94,03	95,34	95,68	95,97	96,03	96,02	96,06	96,22	96,59	96,47
MIN	12,90	33,87	83,89	77,48	88,92	92,12	38,27	92,66	92,21	83,17
MEDIA	55,16	78,94	92,90	90,04	94,47	95,01	84,11	94,77	95,57	92,13

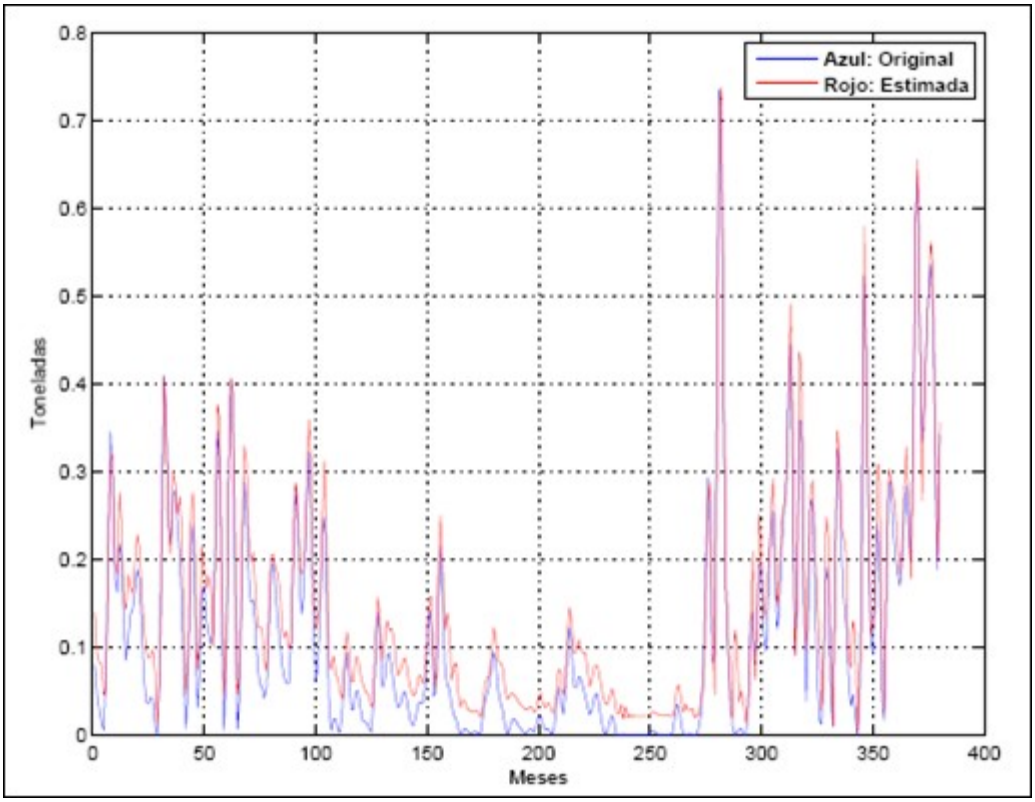

Figura 4.1: Rendimiento data estimada v/s data real de training MLP (5, 16, 1).

La figura 4.1 corresponde al gráfico de data estimada v/s data real para training en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 75% de la data total, en este caso el alrededor de 380 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología de 5 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida.

La figura muestra un desfase algo notorio al comienzo del período de aprendizaje, pero a medida que la red comienza a entrenarse ese desfase se hace cada vez menos notorio, acentuándose una gran cantidad de anchovetas capturadas alrededor del mes 280.

Tabla 4.4: Evaluación rendimiento de R^2 de Test con: Número de épocas=200, τ=5 y γ= 2E-3.

N° Nodos Métricas	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
STD	32,88	25,17	6,72	7,45	3,26	3,18	16,43	1,38	3,45	4,11
MAX	91,81	93,71	93,09	93,32	93,07	93,44	92,49	93,22	92,63	92,83
MIN	11,94	15,89	71,55	70,05	82,15	83,08	39,03	89,07	82,62	80,83
MEDIA	42,80	69,89	88,58	87,00	90,68	90,53	79,75	91,05	89,84	87,02

La figura 4.2 corresponde al gráfico de data estimada v/s data real para test en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 130 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología de 5 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura muestra un desfase algo notorio cercano al mes 10, pero de ahí en adelante se estabiliza para obtener un rendimiento estable.

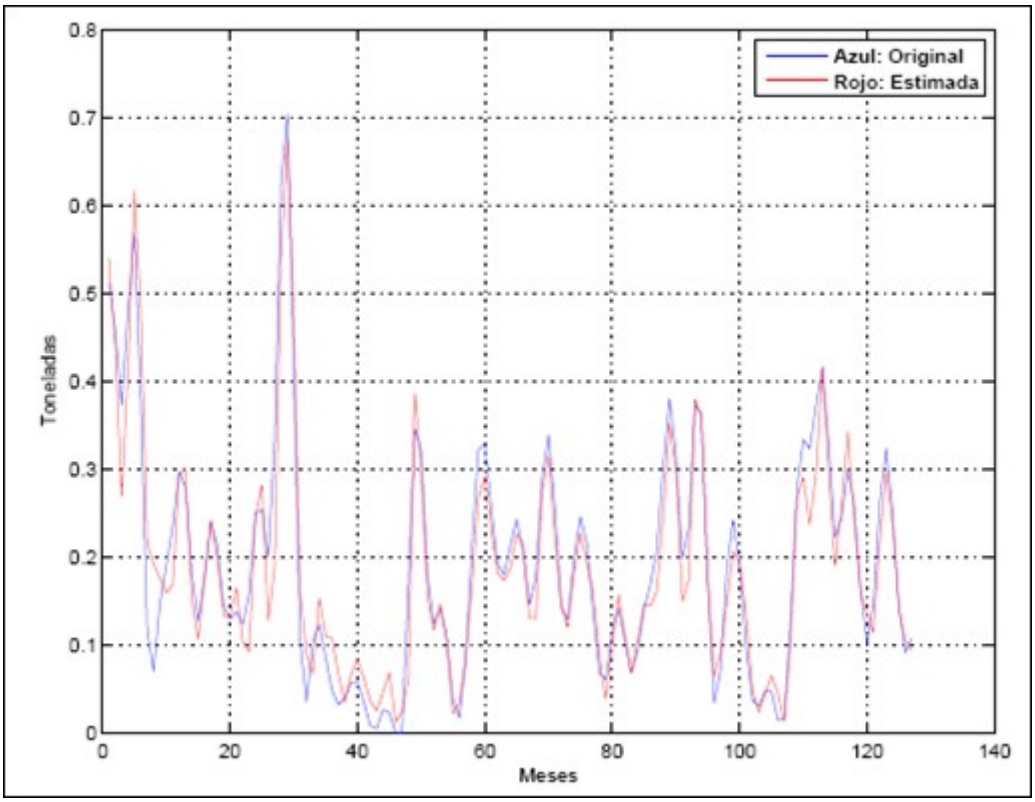

Figura 4.2: Rendimiento data estimada v/s data real de test MLP (5, 16, 1).

La figura 4.3 corresponde al gráfico de dispersión de la data estimada v/s data real para test en su valor promedio. El gráfico corresponde a una topología de 5 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura tiene un valor de R^2=91,05, lo cual es bastante bueno pero aun mejorable en medida que se realicen los ajustes restantes.

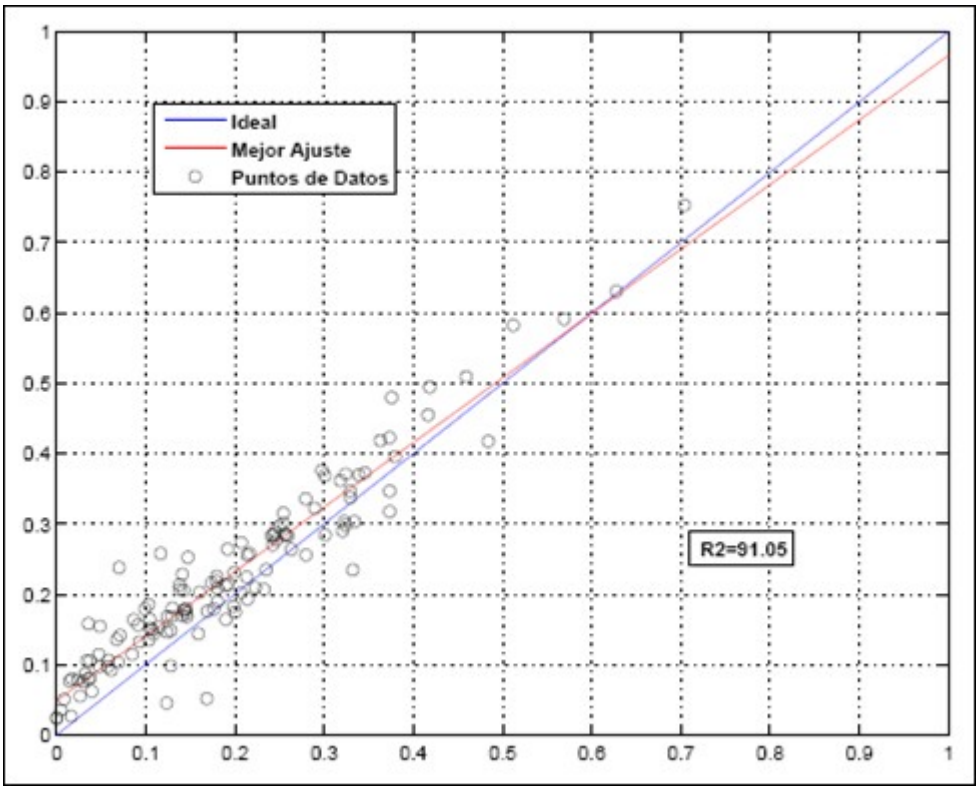

Figura 4.3: Dispersión data estimada v/s data real de test MLP (5, 16, 1).

Finalmente luego de analizar los resultados obtenidos en las tablas 4.2, 4.3, 4.4 y en las figuras 4.1, 4.2, 4.3, se determinó que la arquitectura más recomendable a seguir utilizando es la correspondiente a 16 nodos en su capa oculta, debido a sus bajos promedios de desviación estándar y a su alto promedio de R^2 para test. Por lo tanto, ahora se mantendrá fijo el número de nodos con ese valor y en la etapa siguiente se comenzará a modificar τ (desfase mensual).

Elección τ (Desfase mensual).

El segundo método de selección para encontrar la arquitectura ideal, es el de ajustar el valor de τ o desfase, el que se medirá en meses respecto a los valores históricos almacenados en la base de datos. Este valor es de suma importancia, ya que nos indicará la cantidad de nodos que tendrá la capa de entrada del predictor. Para esto se trabajó con el siguiente conjunto *{1, 2,....12},* ajustando de uno en uno el valor de τ y con los siguientes parámetros fijos:

Tabla 4.5: Parámetros fijos para calibrar desfase.

Muestra	=	75%	(Porcentaje del total de la data).
Número de épocas	=	200.	
Número de nodos capa oculta	=	16.	
γ	=	2E-3	(Factor de penalidad para el ajuste de los pesos).

A continuación se detallan los resultados obtenidos al efectuar el ajuste de τ:

Tabla 4.6: Evaluación rendimiento del ECM con: Número de épocas= 200, Número de nodos=16 y γ= 2E-3.

τ Métricas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
STD	1,11	9,06	5,94	1,12	0,84	0,89	0,62	0,63	0,57	0,49	0,36	0,32
MAX	5732,00	279,08	92,47	46,61	29,57	21,80	17,12	14,95	12,00	10,14	9,07	8,06
MIN	5728,63	252,54	76,54	43,05	26,79	19,56	14,95	13,08	10,23	8,93	7,90	7,03
MEDIA	5731,35	262,17	81,98	44,58	27,84	20,74	16,07	13,87	11,20	9,60	8,45	7,62

Tabla 4.7: Evaluación rendimiento de R^2 de Training con: Número de épocas= 200, Número de nodos=16 y γ= 2E-3.

τ Métricas	1	2	3	4	5	6	7	8	9	10	11	12
STD	0,29	1,63	9,86	1,25	2,40	2,29	2,60	5,82	4,87	2,64	5,00	2,68
MAX	73,68	85,73	93,12	94,59	96,17	96,80	96,58	96,91	96,99	97,40	96,47	97,26
MIN	72,90	80,34	61,55	90,57	88,84	88,94	88,63	78,18	84,66	89,61	82,13	89,93
MEDIA	73,37	84,93	88,99	93,54	94,46	94,90	94,97	94,54	93,82	94,87	93,28	94,32

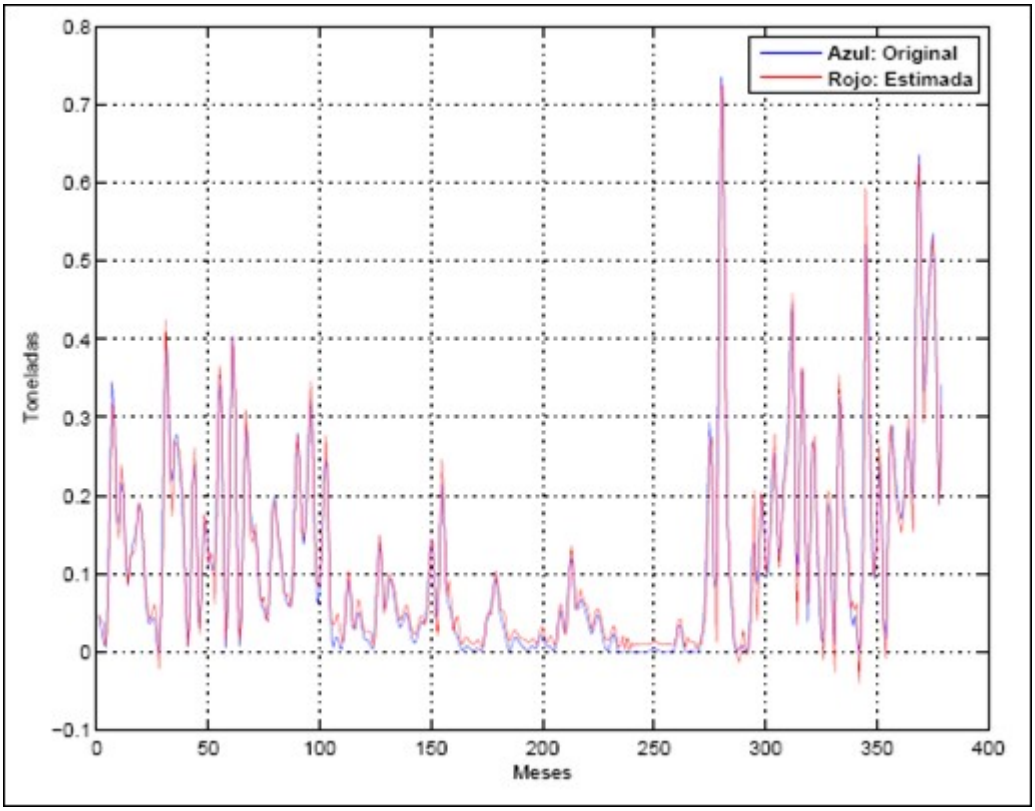

Figura 4.4: Rendimiento data estimada v/s data real de training MLP (6, 16, 1).

La figura 4.4 corresponde al gráfico de data estimada v/s data real para training en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 75% de la data total, en este caso alrededor de 380 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología

de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura muestra un desfase algo notorio entre los meses 110 y 240 en donde la captura de anchovetas se ve algo disminuida, lo que se ve superado posterior al mes 240.

Tabla 4.8: Evaluación rendimiento de R^2 de Test con: Número de épocas= 200, Número de nodos=16 y γ= 2E-3.

τ Métricas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
STD	3,34	1,80	13,48	1,58	3,73	2,08	4,73	6,44	7,54	4,99	6,46	3,82
MAX	59,61	76,14	86,34	91,33	93,58	93,37	92,63	94,37	92,68	92,04	92,48	89,49
MIN	50,57	70,69	41,63	86,75	81,93	87,16	78,26	71,76	66,73	77,14	74,38	76,23
MEDIA	56,86	74.06	76,39	89,62	90,36	91,39	88,50	88,62	86,05	87,25	84,52	84,71

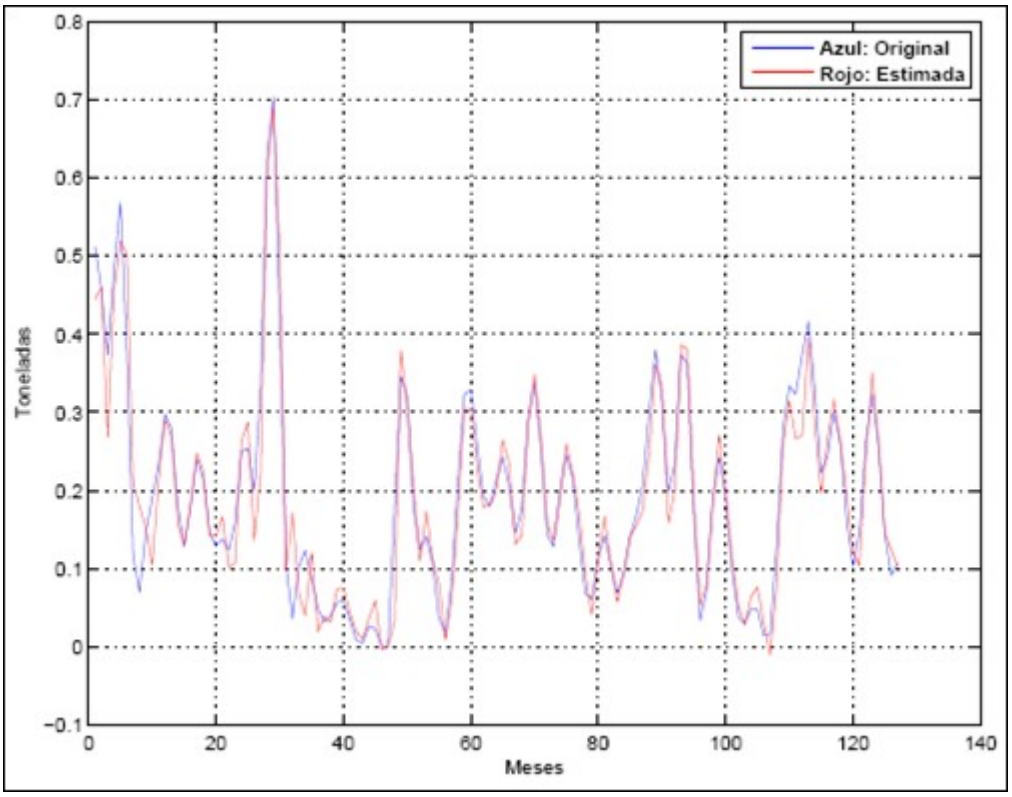

Figura 4.5: Rendimiento data estimada v/s data real de test MLP (6, 16, 1).

La figura 4.5 corresponde al gráfico de data estimada v/s data real para test en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 130 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura muestra un desfase algo notorio dentro de sus primero 20 meses, pero de ahí en adelante se estabiliza para obtener un rendimiento superior al ajuste anterior correspondiente al número de nodos.

La figura 4.6 corresponde al gráfico de dispersión de la data estimada v/s data real para test en su valor promedio. El gráfico corresponde a una topología de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura tiene un valor de R^2=91,39, lo cual mejora el resultado del ajuste anterior.

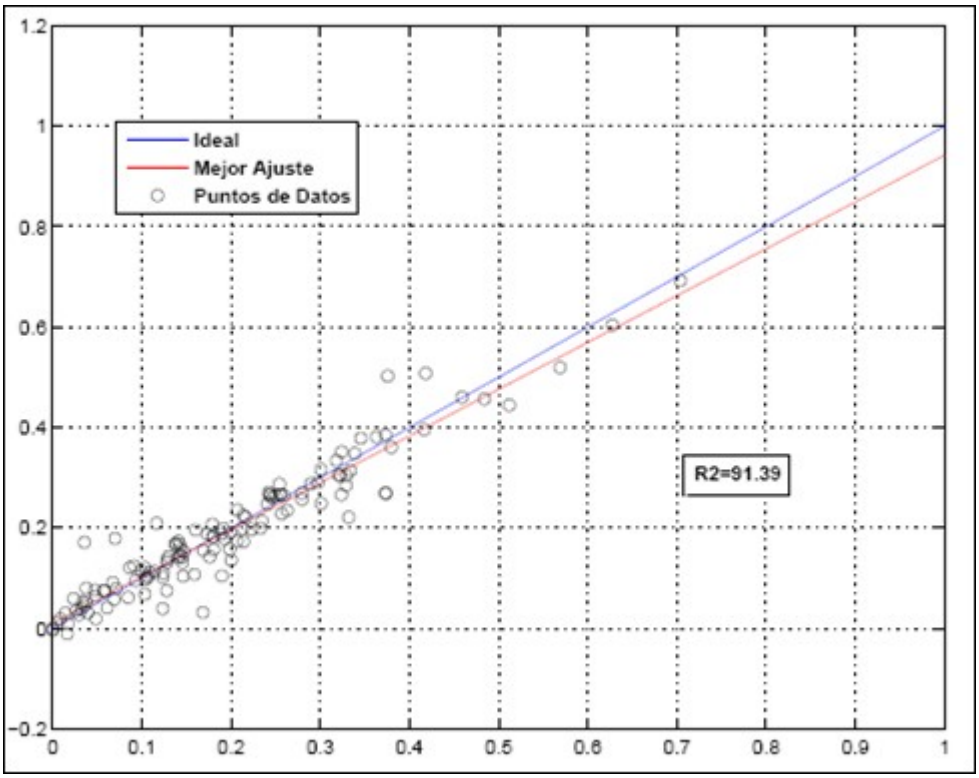

Figura 4.6: Dispersión data estimada v/s data real de test MLP (6, 16, 1).

Finalmente luego de analizar los resultados obtenidos en las tablas 4.6, 4.7, 4.8 y en las figuras 4.4, 4.5, 4.6, se determinó que la arquitectura más recomendable a seguir utilizando es la correspondiente a τ con valor 6, ya que su valor promedio de R^2 para test es el mayor dentro de todos los resultados observados. Por lo tanto, ahora se mantendrá fijo el valor de τ en 6 y en la etapa siguiente se comenzará a ajustar γ.

Elección Factor de Penalidad (γ).

Luego de calibrar el número de nodos en la capa oculta y el desfase del predictor, el ajuste que se realiza a continuación corresponde al factor de penalidad. Para esto se trabajó con el siguiente conjunto *{1E-3, 2E-3,……9E-3}*, ajustando de uno en uno y manteniendo fijos los siguientes parámetros:

Tabla 4.9: Parámetros fijos para calibrar factor de penalidad.

Muestra	=	75%	(Porcentaje del total de la data).

Número de épocas	=	200.	
Número de nodos capa oculta	=	16.	
τ	=	6	(Desfase mensual).

A continuación se detallan los resultados obtenidos al realizar el ajuste del factor de penalidad:

Tabla 4.10: Evaluación rendimiento del ECM con: Número de épocas= 200, Número de nodos=16 y τ= 6.

γ Métricas	**1E-3**	**2E-3**	**3E-3**	**4E-3**	**5E-3**	**6E-3**	**7E-3**	**8E-3**	**9E-3**
STD	1,11	1,11	0,95	1,18	0,95	1,22	1,05	0,89	1,01
MAX	22,21	22,95	22,39	23,58	22,39	23,33	22,34	21,88	23,06
MIN	18,96	19,40	19,68	19,00	19,80	18,50	18,65	19,32	19,81
MEDIA	20,71	21,16	21,02	20,79	20,90	20,84	20,96	20,59	21,48

Tabla 4.11: Evaluación rendimiento de R^2 de Training con: Número de épocas= 200, Número de nodos=16 y τ= 6.

γ Métrica	**1E-3**	**2E-3**	**3E-3**	**4E-3**	**5E-3**	**6E-3**	**7E-3**	**8E-3**	**9E-3**

s									
STD	24,51	4,57	4,37	3,12	3,74	12,03	1,11	2,07	2,20
MAX	96,49	96,88	96,76	96,85	97,32	96,43	97,10	96,25	96,50
MIN	18,67	81,58	81,82	86,52	84,62	57,67	93,62	89,00	90,67
MEDIA	79,28	93,86	93,61	95,06	94,73	88,96	95,94	94,36	94,37

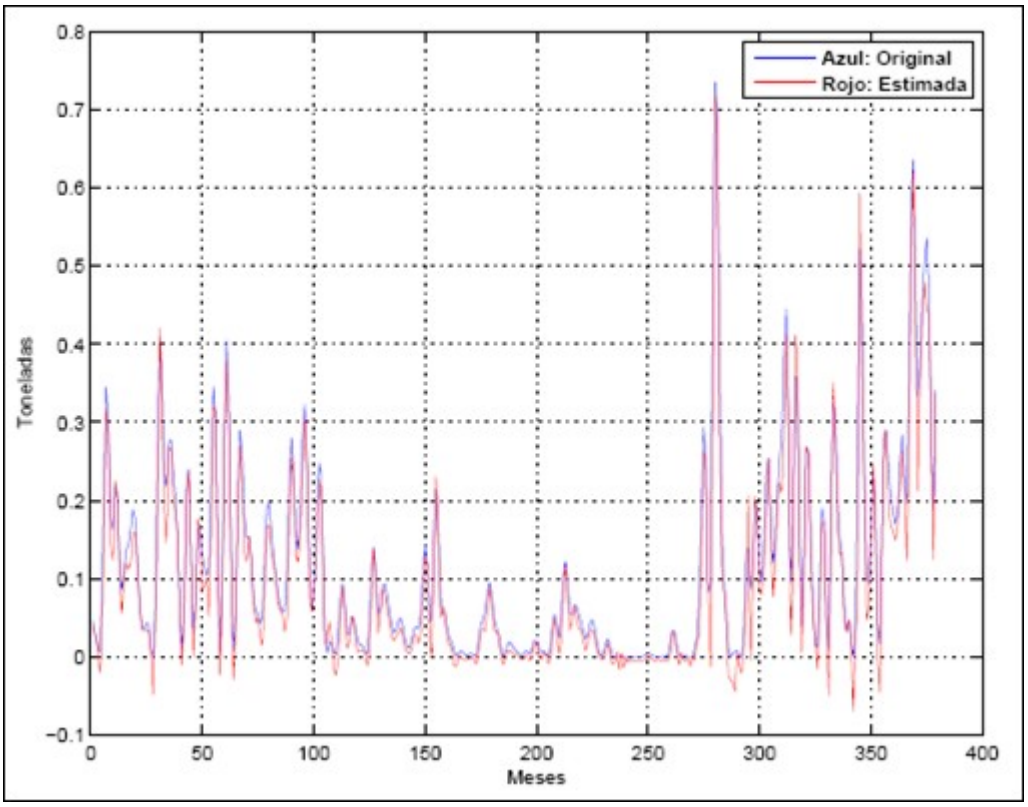

Figura 4.7: Rendimiento data estimada v/s data real de training MLP (6, 16, 1).

La figura 4.7 corresponde al gráfico de data estimada v/s data real para training en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 75% de la data total, en este caso alrededor de 380 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta pocos puntos en donde las curvas se vean notoriamente desfasadas,

producto que el ajuste de γ acelera el rendimiento del ajuste de los pesos en esta etapa de training.

Tabla 4.12: Evaluación rendimiento de R^2 de Test con: Número de épocas= 200, Número de nodos=16 y τ= 6.

γ **Métricas**	**1E-3**	**2E-3**	**3E-3**	**4E-3**	**5E-3**	**6E-3**	**7E-3**	**8E-3**	**9E-3**
STD	21,16	5,78	3,65	5,24	6,27	8,48	1,73	3,69	3,95
MAX	93,98	94,09	93,53	93,84	93,97	93,86	94,35	94,57	94,11
MIN	24,37	79,40	80,78	76,87	73,16	64,67	89,17	81,25	82,86
MEDIA	78,29	89,19	90,31	90,36	89,56	87,32	91,68	89,72	89,60

La figura 4.8 corresponde al gráfico de data estimada v/s data real para test en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso el alrededor de 130 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura muestra un desfase algo notorio cercano al mes 10, pero de ahí en adelante se estabiliza para obtener un rendimiento superior al ajuste anterior correspondiente a τ.

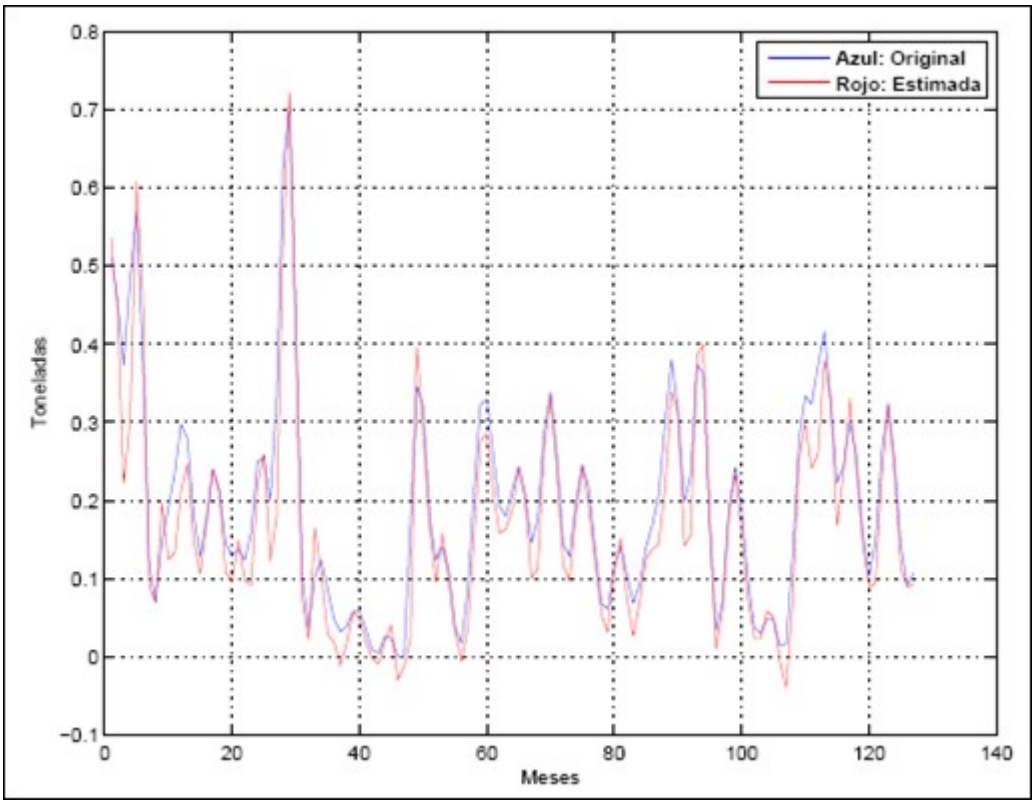

Figura 4.8: Rendimiento data estimada v/s data real de test MLP (6, 16, 1).

La figura 4.9 corresponde al gráfico de dispersión de la data estimada v/s data real para test en su valor promedio. El gráfico corresponde a una topología de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura tiene un valor de R^2=91,68, lo que mejora el resultado del ajuste anterior.

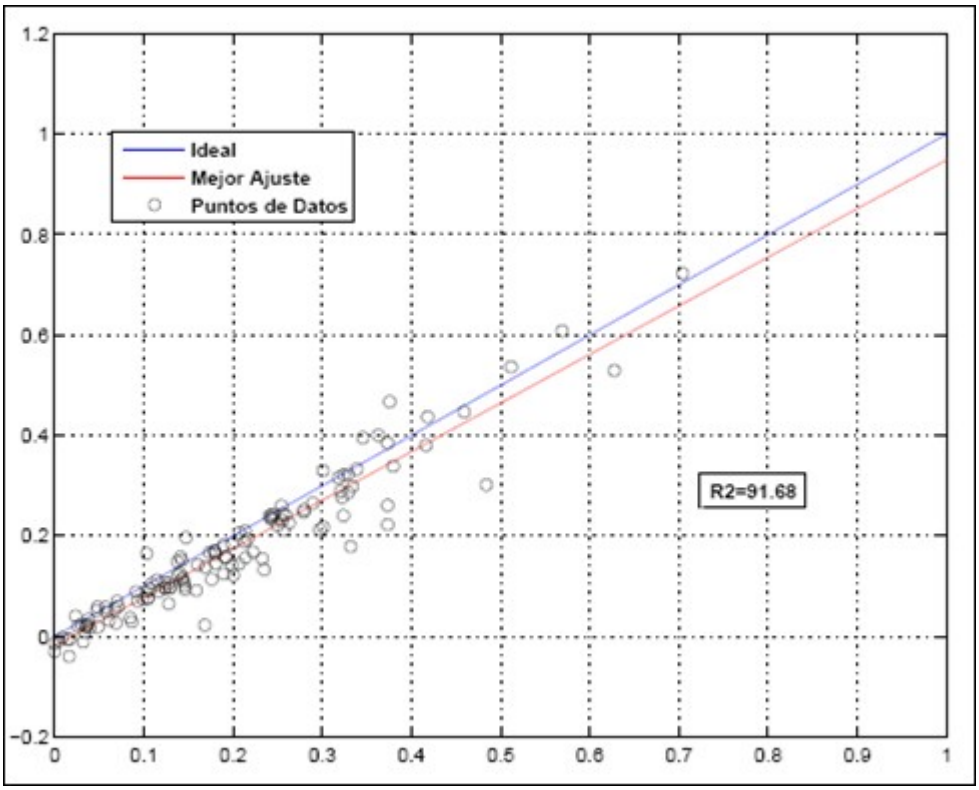

Figura 4.9: Dispersión data estimada v/s data real de test MLP (6, 16, 1).

Finalmente luego de analizar los resultados obtenidos en las tablas 4.10, 4.11, 4.12, y en las figuras 4.7, 4.8, 4.9 se determinó que la arquitectura más recomendable a seguir utilizando es la que tiene como valor γ= 7E-3, ya que su promedio de R^2 para test es el mayor dentro de todos los resultados observados, como así también los promedios de las desviaciones estándar son los menores. Por lo tanto, ahora se mantendrá fijo el valor de γ= 7E-3 y en la etapa siguiente se comenzará a modificar el número de épocas.

Elección número de épocas.

Luego de ajustar el número de nodos en la capa oculta, el desfase del predictor y el factor de penalidad, el último parámetro que resta por calibrar es el correspondiente al número de épocas, el que se realiza a continuación. El número de épocas es ajustado cada 200 iteraciones según el siguiente conjunto *{200, 400,……1000}*, y manteniendo fijos los siguientes parámetros:

Tabla 4.13: Parámetros fijos para calibrar número de épocas.

Muestra	=	75%	(Porcentaje del total de la data).
Número de épocas	=	200.	
Número de nodos capa oculta	=	16.	
γ	=	7E-3	(Factor de penalidad para el ajuste de los pesos).

A continuación se detallan los resultados obtenidos al realizar el ajuste del número de épocas:

Tabla 4.14: Evaluación rendimiento del ECM con: γ= 7E-3, Número de nodos=16 y τ= 6.

N° Épc **Métricas**	**200**	**400**	**600**	**800**	**1000**
STD	0,91	0,82	1,10	0,97	0,78
MAX	21,86	22,96	23,16	22,52	22,24
MIN	19,16	20,59	18,88	19,22	19,74
MEDIA	20,57	21,63	21,17	20,79	20,58

Tabla 4.15: Evaluación rendimiento de R^2 de Training con: γ= 7E-3, Número de nodos=16 y τ= 6.

N° Épc **Métricas**	**200**	**400**	**600**	**800**	**1000**

STD	0,36	2,24	14,79	4,54	6,34
MAX	96,88	96,96	96,50	96,73	96,67
MIN	95,70	90,74	50,01	84,54	77,37
MEDIA	96,19	94,73	84,02	93,16	92,99

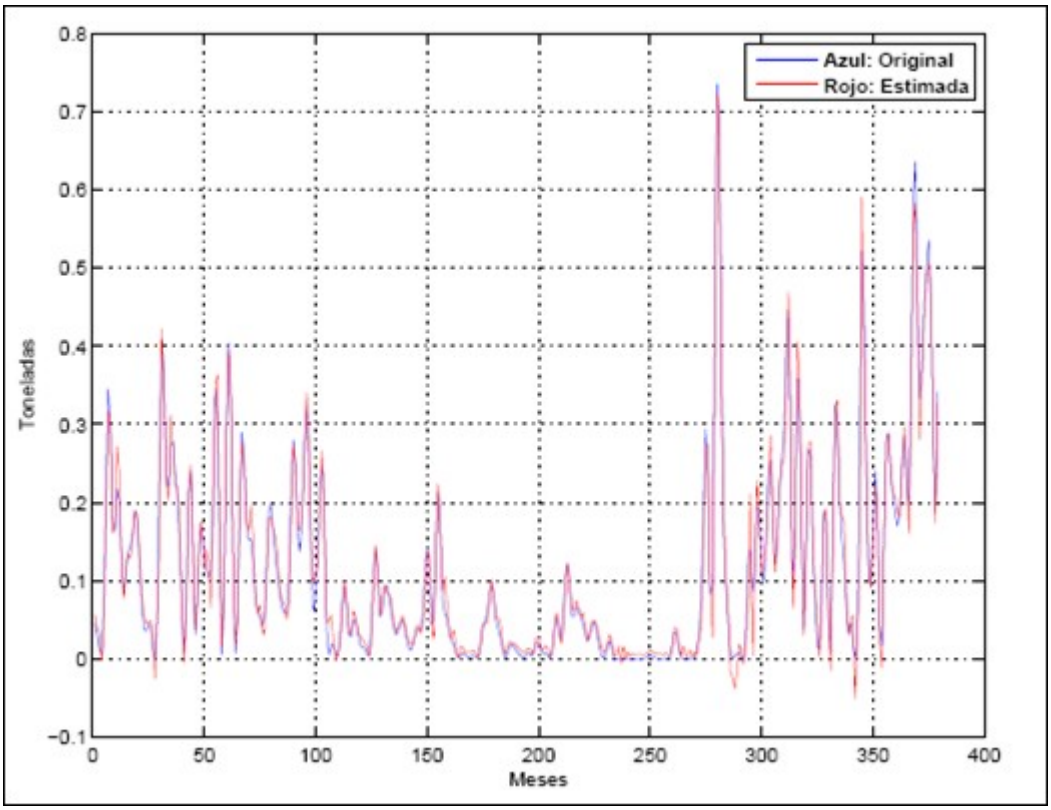

Figura 4.10: Rendimiento data estimada v/s data real de training MLP (6, 16, 1).

La figura 4.10 corresponde al gráfico de data estimada v/s data real para training en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 75% de la data total, en este caso alrededor de 380 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta pocos puntos en donde las curvas se muestren notoriamente desfasadas, producto que el ajuste del número de épocas corresponde a la última etapa de selección de la topología ideal y de sus respectivos parámetros.

Tabla 4.16: Evaluación rendimiento de R^2 de Test con: γ= 7E-3, Número de nodos=16 y τ= 6.

N° Épc **Métricas**	**200**	**400**	**600**	**800**	**1000**
STD	1,91	5,14	13,68	7,63	6,38
MAX	93,98	94,12	94,00	93,98	94,00
MIN	86,94	77,11	53,05	73,90	76,45
MEDIA	91,88	87,89	80,76	86,70	89,25

La figura 4.11 corresponde al gráfico de data estimada v/s data real para test en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 130 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura muestra un desfase algo notorio cercano al mes 10, pero de ahí en adelante se estabiliza para obtener un rendimiento superior al ajuste anterior correspondiente al factor de penalidad.

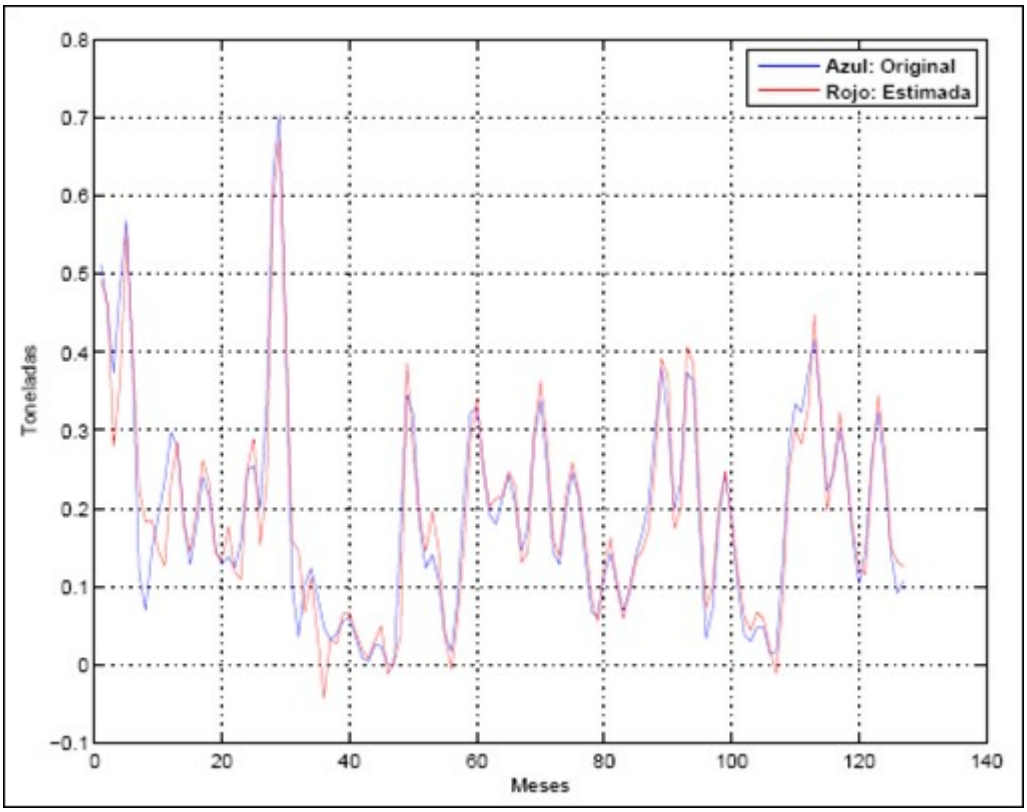

Figura 4.11: Rendimiento data estimada v/s data real de test MLP (6, 16, 1).

La figura 4.12 corresponde al gráfico de dispersión de la data estimada v/s data real para test en su valor promedio. El gráfico corresponde a una topología de 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura tiene un valor de R^2=91,88, lo que mejora el resultado del ajuste anterior.

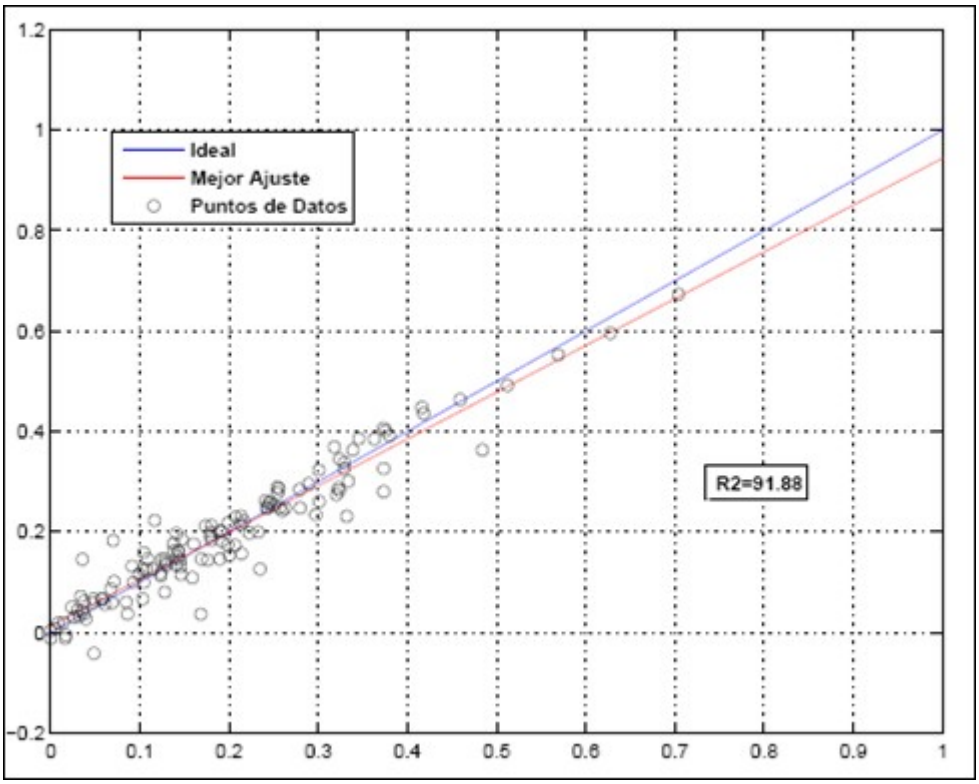

Figura 4.12: Dispersión data estimada v/s data real de test MLP (6, 16, 1).

Finalmente luego de analizar los resultados obtenidos en las tablas 4.14, 4.15, 4.16 y en las figuras 4.10, 4.11, 4.12 se determinó que la arquitectura más recomendable a seguir utilizando es la que tiene el número de épocas= 200, ya que su promedio de R^2 para test es el mayor dentro de todos los resultados observados, como así también los promedios de las desviaciones estándar son los menores.

Luego de realizar la etapa de ajuste del predictor se mostrará en la tabla 4.18 las 10 iteraciones realizadas con la topología ideal, la cual consta de los siguientes parámetros con sus respectivos valores finales ya ajustados:

Tabla 4.17: Parámetros de arquitectura ideal suavizada (6, 16, 1).

Muestra	=	75%	(Porcentaje del total de la data).
Número de épocas	=	200.	
Número de nodos capa oculta	=	16.	

γ	=	7E-3	(Factor de penalidad para el ajuste de los pesos).
τ	=	6	(Desfase mensual)

Tabla 4.18: Evaluación rendimiento con: γ= 7E-3, Número de nodos=16, τ= 6 y Número de épocas=200.

It **Métricas**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ECM	19,16	21,10	21,86	21,54	22,28	21,61	19,43	20,65	21,27	20,66
R^2 TR	96,06	95,70	96,31	96,88	94,18	96,03	95,86	96,49	96,12	96,85
R^2 TEST	91,67	92,96	91,65	92,30	87,29	92,00	93,22	92,77	91,46	94,40

En la tabla 4.18, donde se realizan las iteraciones se destacó en negrita el mínimo y máximo para test, de la topología ideal, los cuales serán representados a continuación.

La figura 4.13 corresponde al gráfico de data estimada v/s data real para test en su valor máximo. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 130 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a la topología ideal compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta muy pequeños desfases alrededor del mes 10, pero en general las curvas se superponen lo que demuestra que su correlación es alta y en este caso tiene un valor de R^2=94,40**,** el que fue conseguido en su décima iteración.

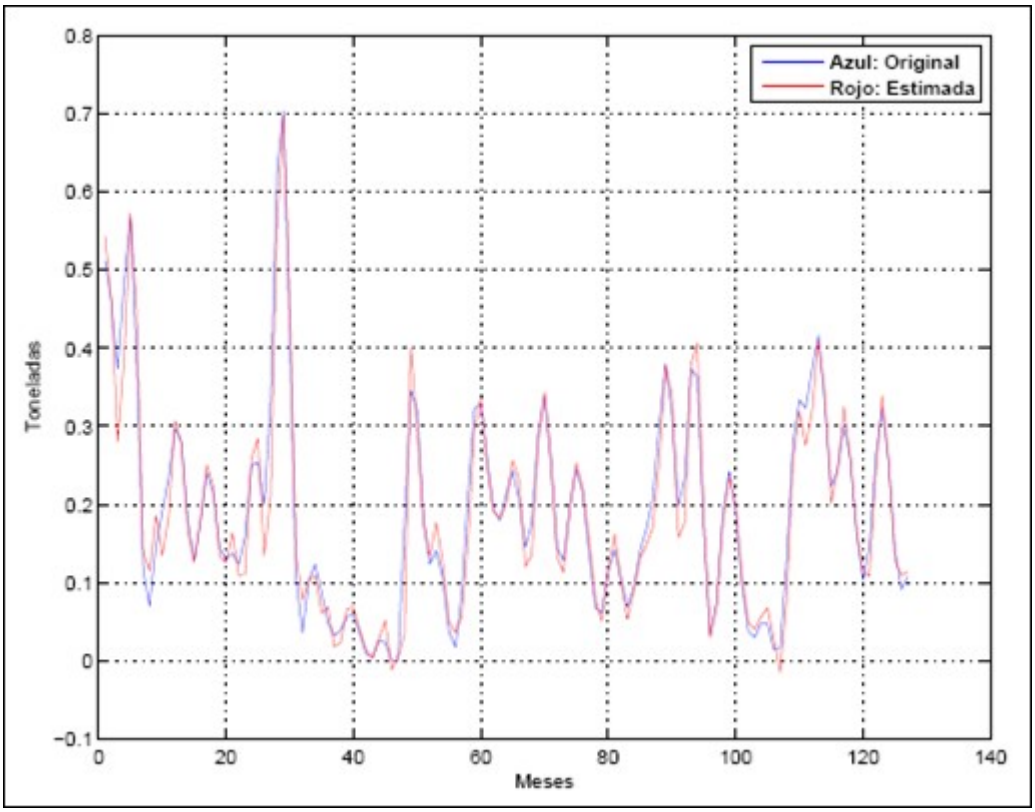

Figura 4.13: Rendimiento data estimada v/s data real de test MLP (6, 16, 1).

La figura 4.14 corresponde al gráfico de data estimada v/s data real para test en su valor mínimo. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 130 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a la topología ideal compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta desfases notorios en sus primeros 20 meses, pero de ahí en adelante las curvas se estabilizan alcanzando un valor de R^2=87,29, el que fue conseguido en su quinta iteración.

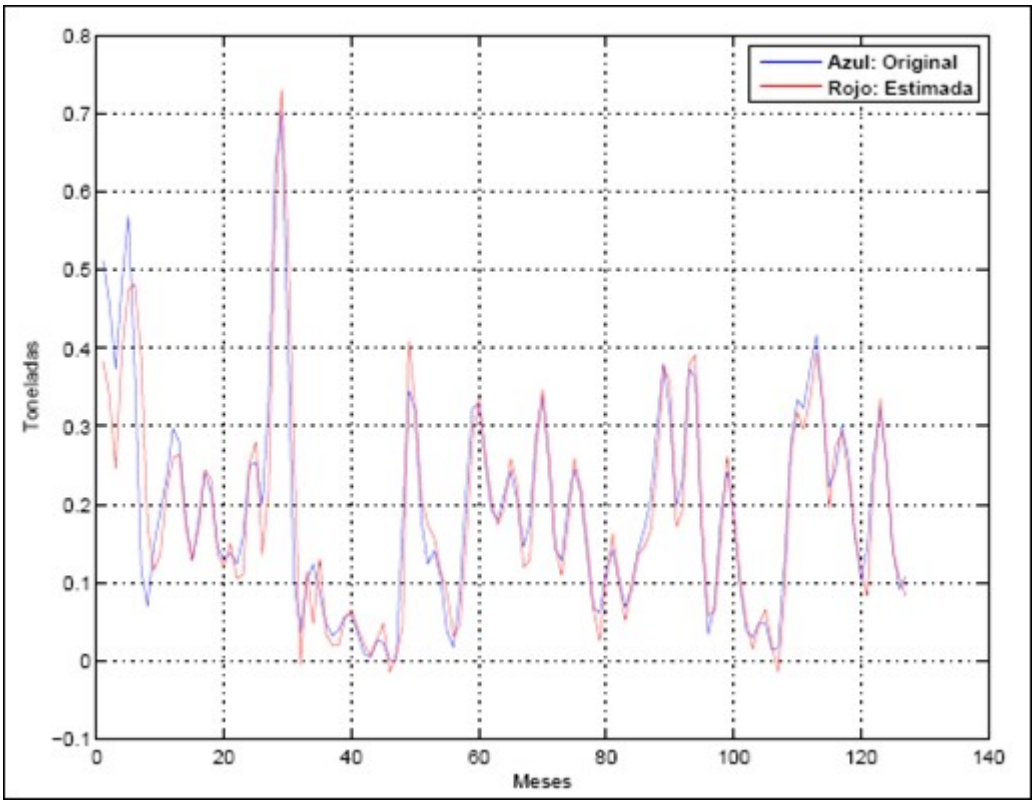

Figura 4.14: Rendimiento data estimada v/s data real de test MLP (6, 16, 1).

Tabla 4.19: Promedios con: γ= 7E-3, Número de nodos=16, τ= 6 y Número de épocas=200.

Métricas **Métricas**	**STAND**	**MAX**	**MIN**	**MEDIA**
ECM	0,99	22,28	19,16	20,83
R^2 TR	0,36	96,88	94,18	96,19
R^2 TEST	1,91	94,40	87,29	91,88

En la tabla 4.19 se observan los resultados finales de la topología ideal, es decir, los valores máximos, mínimos, desviación estándar y media correspondiente al ECM, R^2 para training y el R^2 para test. Cabe destacar que esta tabla no es más que un resumen de la tabla 4.18 anteriormente vista.

A continuación se graficarán los rendimientos para training y test de la data estimada v/s la data original junto al gráfico de dispersión para test.

La figura 4.15 corresponde al gráfico de data estimada v/s data real para training en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 75% de la data total, en este caso alrededor de 380 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a la topología ideal compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta pocos puntos en donde las curvas se muestren notoriamente desfasadas, producto que este gráfico corresponde a la topología ideal en su etapa de training.

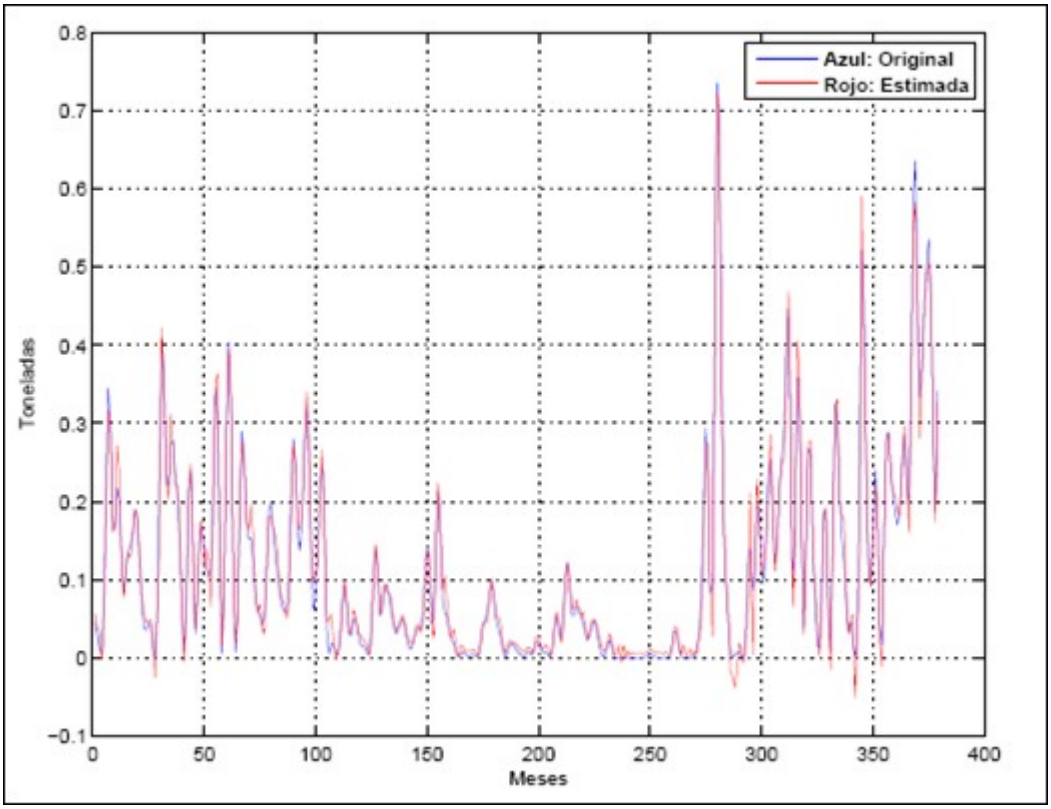

Figura 4.15: Rendimiento data estimada v/s data real de training MLP (6, 16, 1).

La figura 4.16 corresponde al gráfico de data estimada v/s data real para test en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 130 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a la topología ideal compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura muestra un desfase algo notorio cercano al mes 10, pero de ahí en adelante se estabiliza para obtener un rendimiento bastante aceptable.

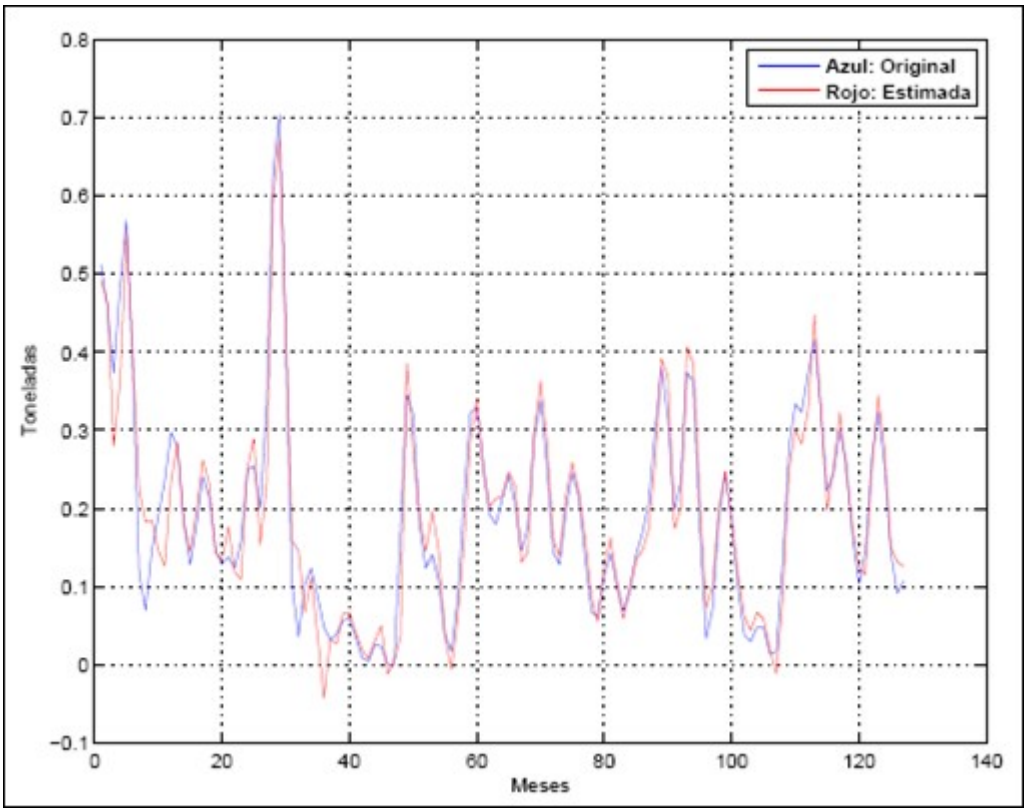

Figura 4.16: Rendimiento data estimada v/s data real de test MLP (6, 16, 1).

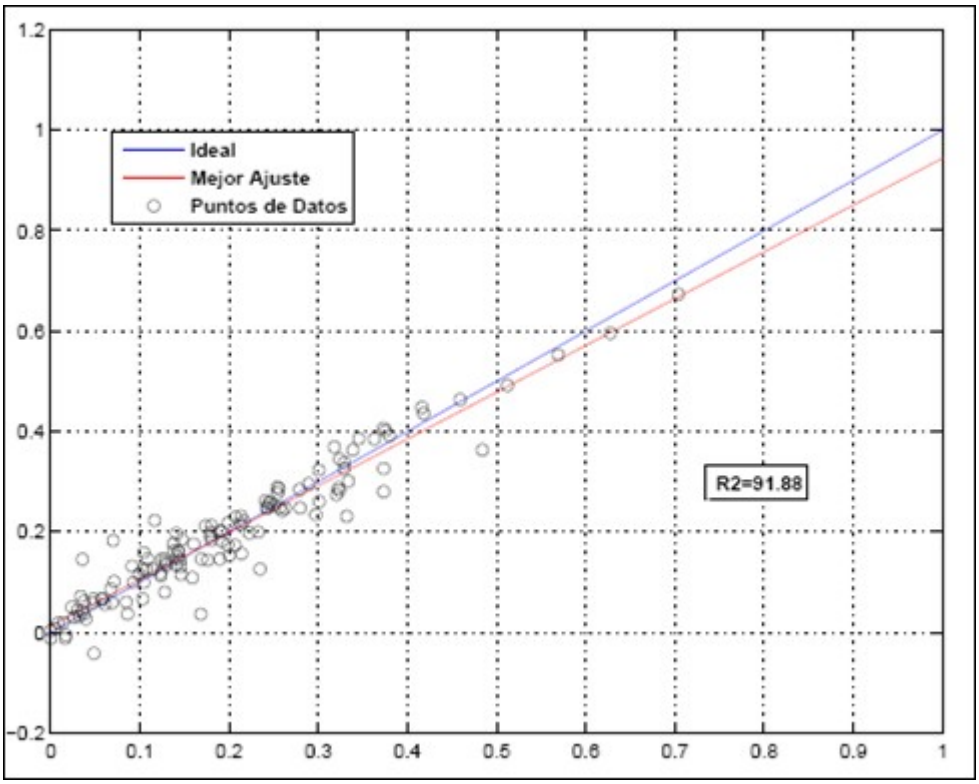

Figura 4.17: Dispersión data estimada v/s data real de test MLP (6, 16, 1).

La figura 4.17 corresponde al gráfico de dispersión de la data estimada v/s data real para test en su valor promedio. El gráfico corresponde a la topología ideal compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura representa un valor de correlación R^2=91,88, lo cual mejora los resultados obtenidos con otras arquitecturas y otros algoritmos de aprendizaje analizados en esta memoria de título.

Luego de conocer los resultados obtenidos por la arquitectura ideal con los datos suavizados, a continuación se analizan los resultados de la arquitectura ideal sin suavizado.

4.2 Arquitectura sin suavizado

En esta sección de la memoria de título, se detallan los resultados de tres arquitecturas, finalizando con la arquitectura ideal que fue encontrada en la sección anterior, pero con la diferencia que la simulación trabajará con los datos sin suavizar para todas estas arquitecturas, por lo tanto, los resultados obtenidos serán distintos en relación a la sección anterior, los cuales se verán reflejados en una tabla comparativa en el punto 4.3.

En el caso de las dos primeras arquitecturas se detallan los resultados mediante tablas y gráficos de rendimiento y dispersión para test, en el último tipo de arquitectura agregaremos gráficos de rendimiento máximo y mínimo para test, para de esta manera notar la diferencia de la arquitectura ideal con y sin suavizado de data.

La primera arquitectura que estará formada por los siguientes parámetros:

Tabla 4.20: Parámetros fijos para calibración de arquitectura no suavizada (2, 4, 1).

Muestra	=	75%	(Porcentaje del total de la data).
Número de épocas	=	200.	
Número de nodos capa oculta	=	4.	
γ	=	7E-3	(Factor de penalidad para el ajuste de los pesos).
τ	=	2	(Desfase mensual)

A continuación se detallan los resultados obtenidos para la arquitectura con los parámetros anteriormente mencionados:

Tabla 4.21: Evaluación rendimiento con: γ= 7E-3, Número de nodos=4, γ= 2 y Número de épocas=200.

It **Métricas**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ECM	43,92	25,95	35,38	43,23	39,97	46,10	41,38	42,65	39,56	34,54
R^2 TR	86,33	74,25	86,50	64,95	82,83	85,68	83,81	43,04	85,73	81,90
R^2	56,11	65,75	62,29	6,87	67,62	62,48	57,90	31,43	68,79	65,38

TEST										

Tabla 4.22: Promedios con: γ= 7E-3, Número de nodos=4, τ= 2 y Número de épocas=200.

Métricas **Métricas**	**STAND**	**MAX**	**MIN**	**MEDIA**
ECM	5,92	46,10	25,95	38,81
R^2 TR	13,89	86,50	43,04	76,05
R^2 TEST	19,89	68,79	6,87	47,18

La figura 4.18 corresponde al gráfico de data estimada v/s data real para test en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 100 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología sin data suavizada compuesta por 2 nodos en la capa de entrada, 4 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta un desfase entre ambas curvas bastante notorio a lo largo de toda la simulación, lo que demuestra que la topología no es la adecuada, como así también que la data no ha sido suavizada.

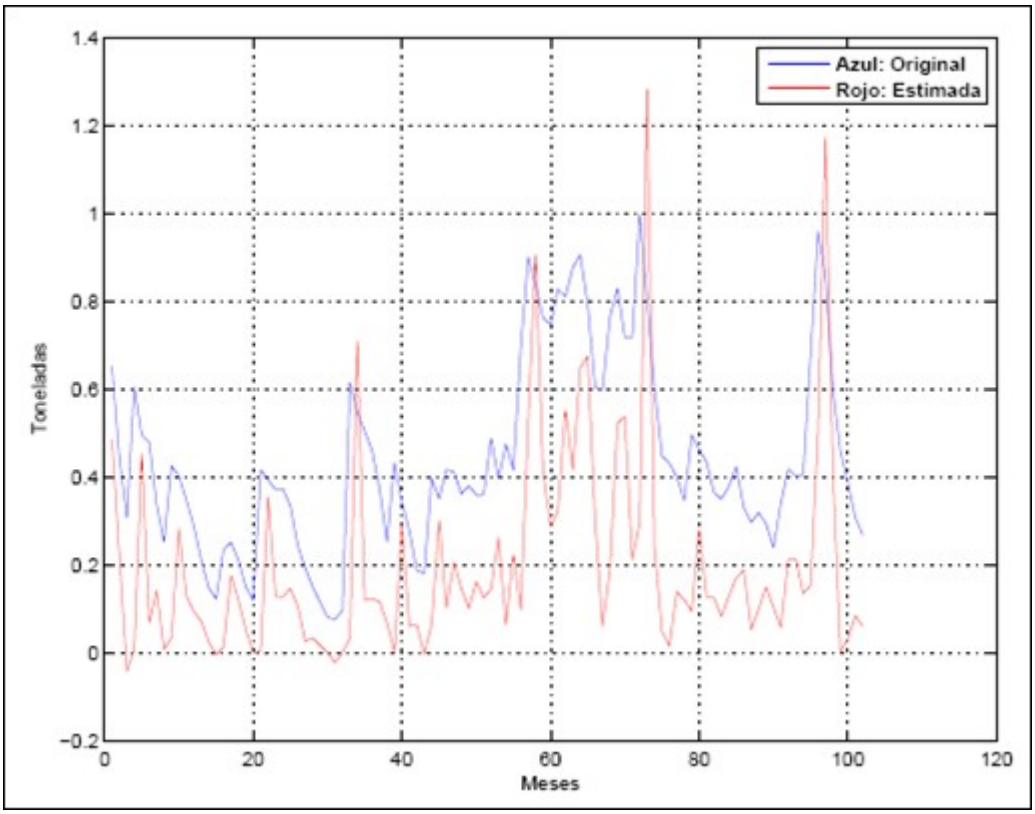

Figura 4.18: Rendimiento data estimada v/s data real de test MLP (2, 4, 1) con data no suavizada.

La figura 4.19 corresponde al gráfico de dispersión de la data estimada v/s data real para test en su valor promedio. El gráfico corresponde a una topología sin data suavizada compuesta por 2 nodos en la capa de entrada, 4 nodos en su capa oculta y un nodo en su capa de salida. La figura representa un valor de correlación R^2=47,18, lo cual es bastante bajo, por lo tanto se deberá intentar con otra topología mejor.

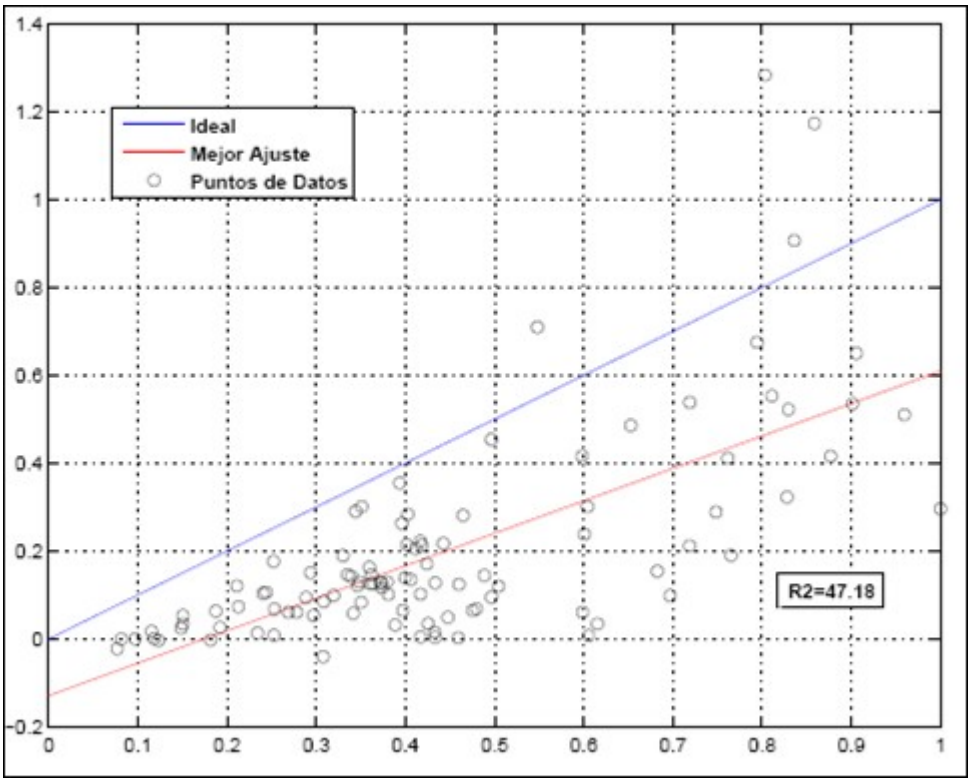

Figura 4.19: Dispersión data estimada v/s data real de test MLP (2, 4, 1) con data no suavizada.

La segunda arquitectura está formada por los siguientes parámetros:

Tabla 4.23: Parámetros fijos para calibración de arquitectura no suavizada (4, 8, 1).

Muestra	=	75%	(Porcentaje del total de la data).
Número de épocas	=	200.	
Número de nodos capa oculta	=	8.	
γ	=	7E-3	(Factor de penalidad para el ajuste de los pesos).
τ	=	4	(Desfase mensual)

A continuación se detallan los resultados obtenidos para la arquitectura con los parámetros anteriormente mencionados:

Tabla 4.24: Evaluación rendimiento con: γ= 7E-3, Número de nodos=8, τ= 4 y Número de épocas=200.

It **Métricas**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ECM	18,35	18,32	19,65	17,41	19,61	17,54	20,48	21,44	20,38	60,88
R^2 TR	86,96	70,38	86,44	87,79	87,03	87,53	87,47	87,11	85,66	87,07
R^2 TEST	59,28	56,77	67,22	54,31	63,55	43,75	59,61	65,93	55,78	61,98

Tabla 4.25: Promedios con: γ= 7E-3, Número de nodos=8, τ= 4 y Número de épocas=200.

Métricas **Métricas**	**STAND**	**MAX**	**MIN**	**MEDIA**
ECM	13,23	60,88	17,41	21,55
R^2 TR	5,29	87,79	70,38	85,18
R^2 TEST	6,79	67,22	43,75	58,44

La figura 4.20 corresponde al gráfico de data estimada v/s data real para test en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 100 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a una topología sin data suavizada compuesta por 4 nodos en la capa de entrada, 8 nodos en su capa oculta y un nodo en su capa de salida. La figura muestra un desfase entre ambas curvas notorio a lo largo de toda la simulación y es alrededor del mes 60 en donde se aprecia mejor, lo que demuestra que la topología no es la adecuada, como así también que la data no ha sido suavizada.

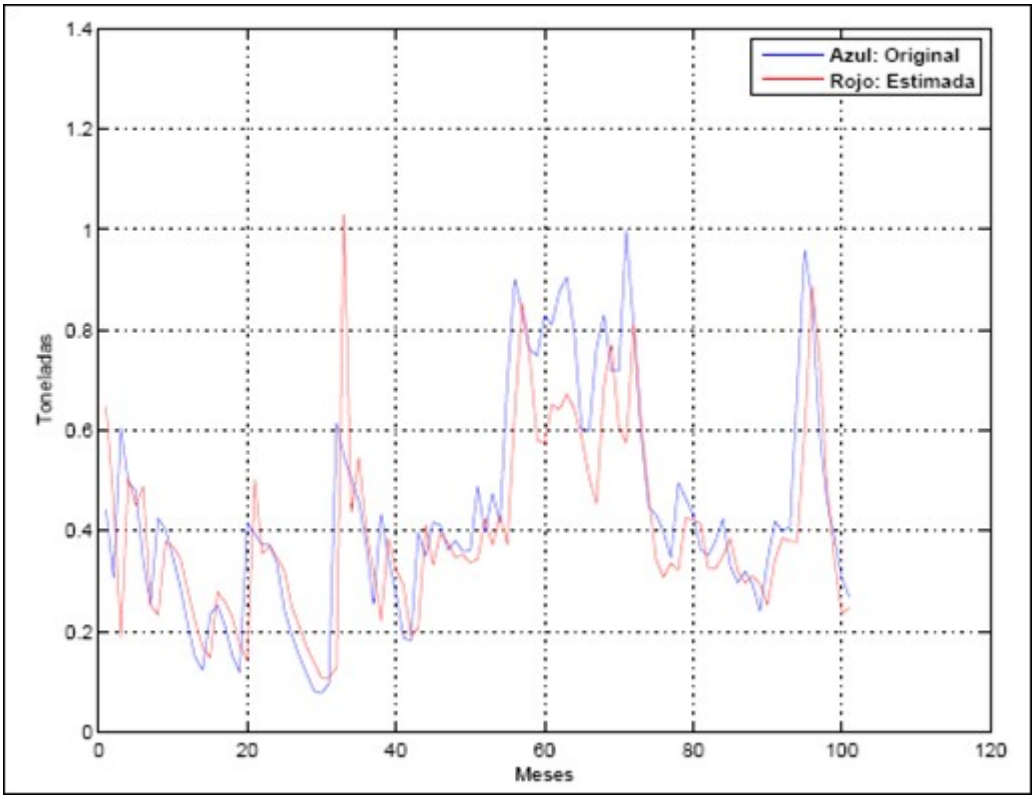

Figura 4.20: Rendimiento data estimada v/s data real de test MLP (4, 8, 1) con data no suavizada.

La figura 4.21 corresponde al gráfico de dispersión de la data estimada v/s data real para test en su valor promedio. El gráfico corresponde a una topología sin data suavizada compuesta por 4 nodos en la capa de entrada, 8 nodos en su capa oculta y un nodo en su capa de salida. La figura representa un valor de correlación R^2=58,44, lo cual es bastante bajo, pero aun así mejor que la topología que analizamos anteriormente, sin embargo, será necesario analizar la

topología ideal a continuación para poder distinguir de manera precisa la gran diferencia que el suavizado de la data realiza.

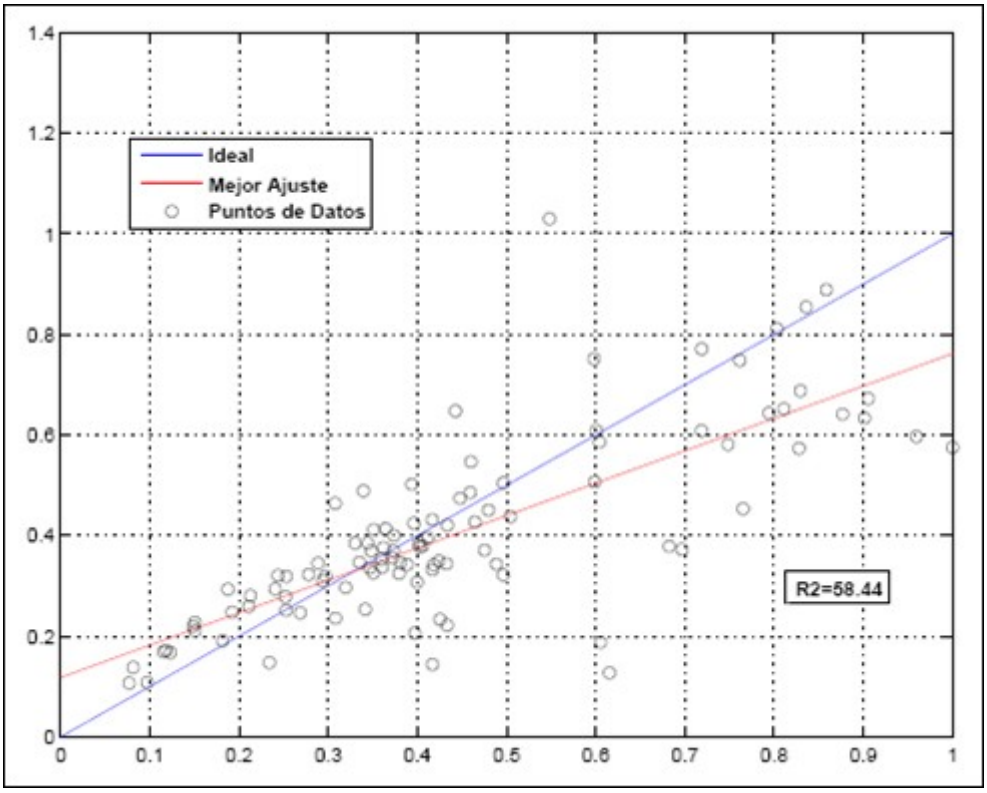

Figura 4.21: Dispersión data estimada v/s data real de test MLP (4, 8, 1) con data no suavizada.

La tercera arquitectura que se detalla en esta sección corresponde a la arquitectura ideal que se ajustó en la sección 4.1, pero la diferencia principal radica en que esta topología no tiene la data suavizada. A continuación se detallan los resultados obtenidos de la arquitectura ideal con la data sin suavizar.

Tabla 4.26: Parámetros fijos de arquitectura ideal no suavizada (6, 16, 1).

Muestra	=	75%	(Porcentaje del total de la data).
Número de épocas	=	200.	
Número de nodos capa oculta	=	16.	

γ	=	7E-3	(Factor de penalidad para el ajuste de los pesos).
τ	=	6	(Desfase mensual)

Tabla 4.27: Evaluación rendimiento con: γ= 7E-3, Número de nodos=16, τ= 6 y Número de épocas=200. (Arquitectura ideal sin suavizado)

It **Métricas**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ECM	14,08	14,72	14,78	14,21	13,19	14,25	14,49	13,71	14,96	13,66
R^2 TR	84,86	89,56	87,12	86,35	86,94	86,20	87,45	87,06	87,36	87,84
R^2 TEST	10,11	51,12	66,81	51,25	44,65	25,28	54,75	51,87	52,03	58,91

En la tabla 4.27, donde se realizan las iteraciones se destacó en negrita el mínimo y máximo para test, de la topología ideal sin suavizado, los cuales serán representados a continuación.

La figura 4.22 corresponde al gráfico de data estimada v/s data real para test en su valor máximo. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 100 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a la topología ideal sin data suavizada compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta desfases de sus curvas a lo largo de toda la simulación, pero alrededor del mes 70 se hace más notorio. Su valor de correlación es de R^2=66,81, el cual fue conseguido en su tercera iteración.

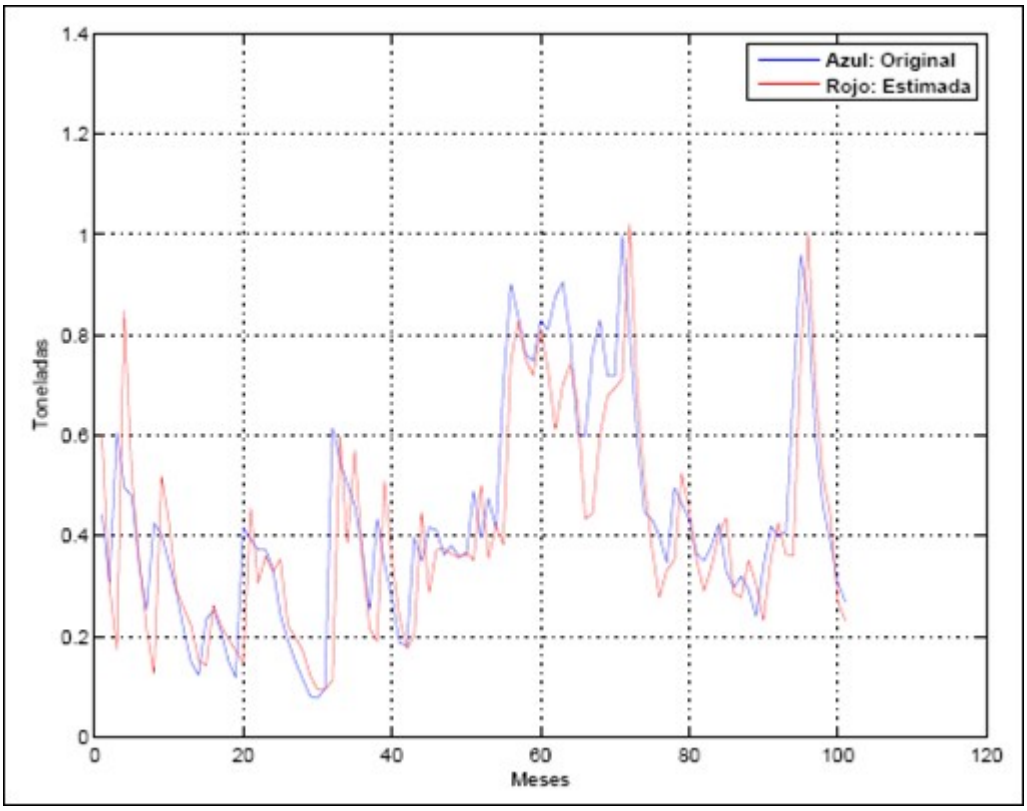

Figura 4.22: Rendimiento data estimada v/s data real de test MLP (6, 16, 1) con data no suavizada.

La figura 4.23 corresponde al gráfico de data estimada v/s data real para test en su valor mínimo. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 100 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a la topología ideal sin data suavizada compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta desfases notorios principalmente entre los meses 60 y 80 y su factor de correlación llega a un valor de R^2=10,11, el cual fue conseguido en su primera iteración.

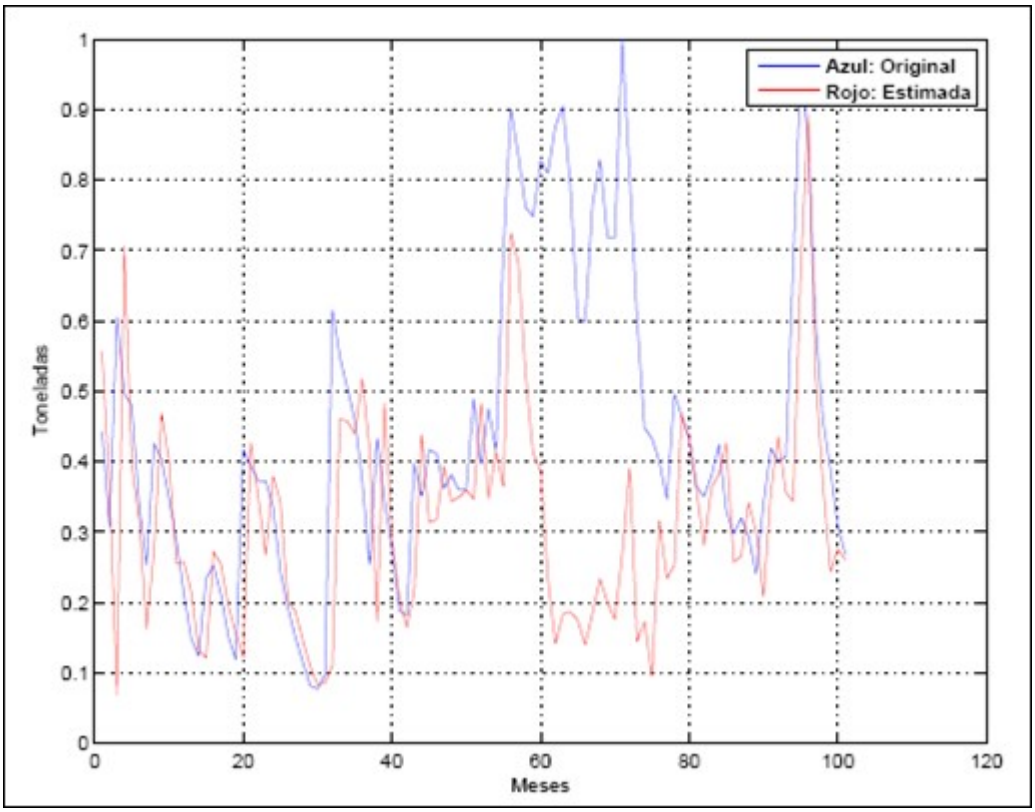

Figura 4.23: Rendimiento data estimada v/s data real de test MLP (6, 16, 1) con data no suavizada.

Tabla 4.28: Promedios con: γ= 7E-3, Número de nodos=16, τ= 6 y Número de épocas=200. (Arquitectura ideal sin suavizado)

Métricas **Métricas**	**STAND**	**MAX**	**MIN**	**MEDIA**
ECM	0,56	14,95	13,19	14,19
R^2 TR	1,21	89,56	84,86	87,07

R^2 TEST	16,72	66,81	10,11	42,07

En la tabla 4.28 se observan los resultados finales de la topología ideal sin suavizado, es decir, los valores máximos, mínimos, desviación estándar y media correspondiente al ECM, R^2 para training y el R^2 para test. Cabe destacar que esta tabla no es más que un resumen de la tabla 4.27 anteriormente vista.

A continuación se graficarán los rendimientos para training y test de la data estimada v/s la data original junto al gráfico de dispersión para test.

La figura 4.24 corresponde al gráfico de data estimada v/s data real para training en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 75% de la data total, en este caso alrededor de 300 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a la topología ideal sin data suavizada compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta desfases a lo largo de toda la simulación, siendo más significativa entre los meses 150 y 250.

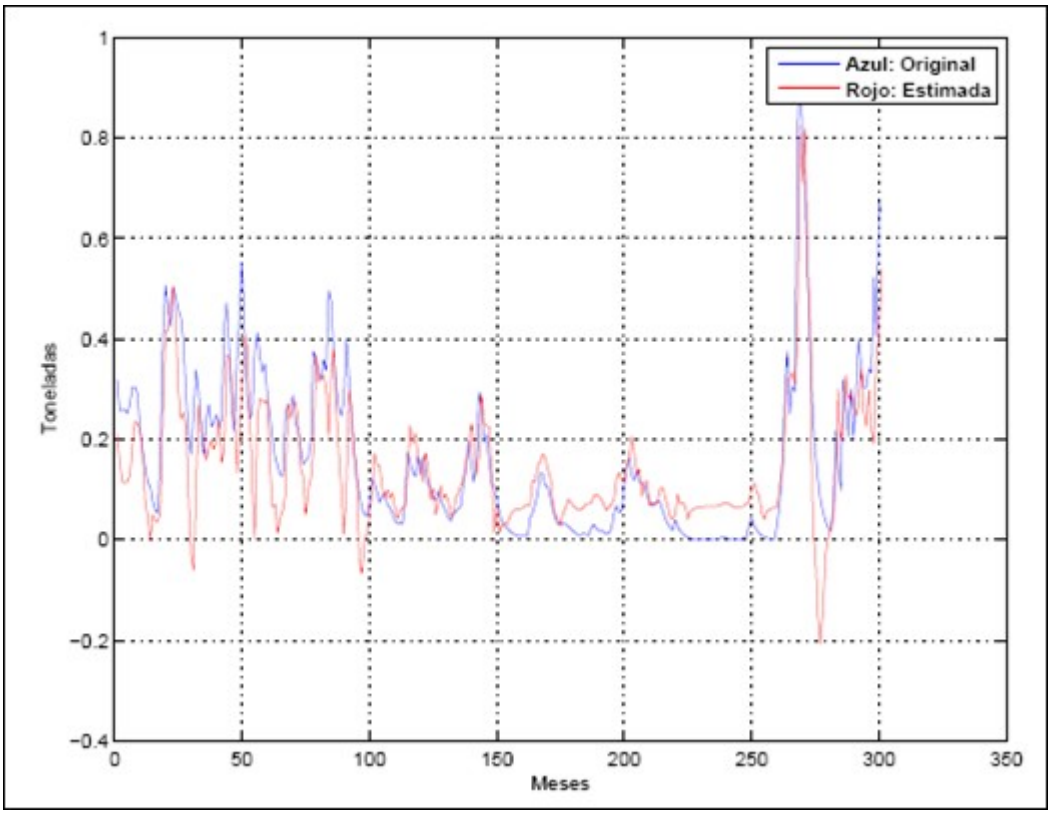

Figura 4.24: Rendimiento data estimada v/s data real de training MLP (6, 16, 1) con data no suavizada.

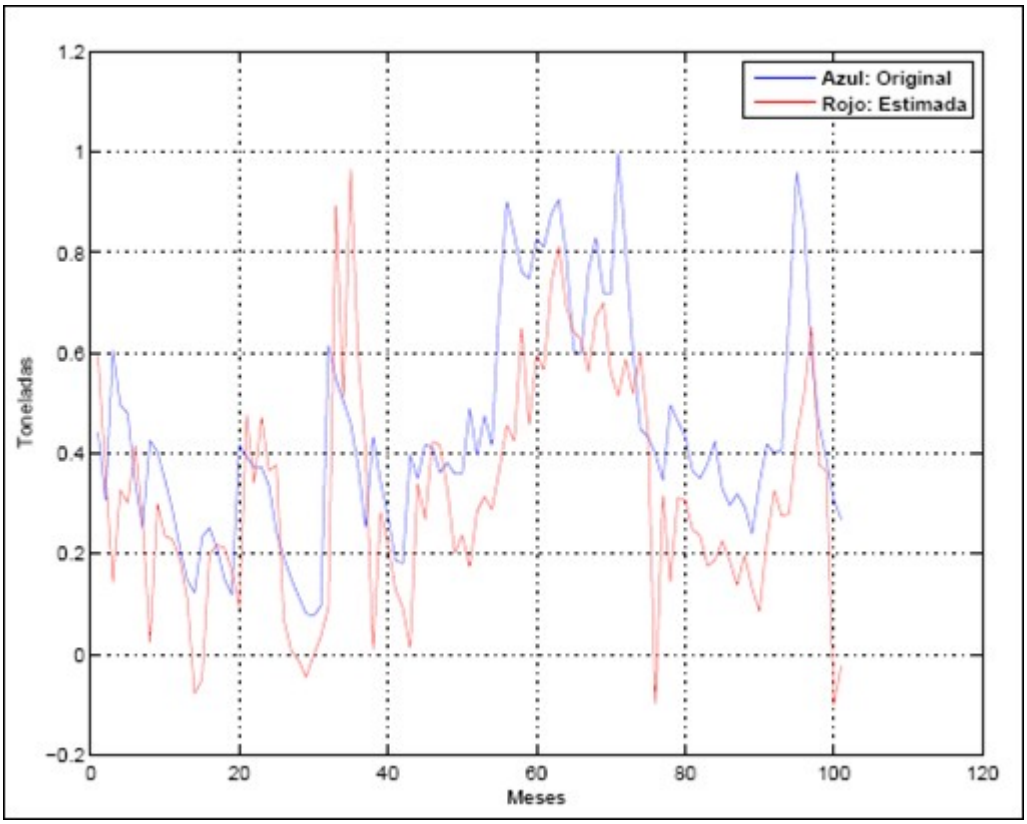

Figura 4.25: Rendimiento data estimada v/s data real de test MLP (6, 16, 1) con data no suavizada.

La figura 4.25 corresponde al gráfico de data estimada v/s data real para test en su valor promedio. Su coordenada *X* representa la cantidad de meses equivalentes al 25% de la data total, en este caso alrededor de 100 meses y su coordenada *Y* representa las capturas de anchovetas medidas en toneladas normalizadas [0,1]. El gráfico corresponde a la topología ideal sin data suavizada compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta un desfase bastante notorio durante toda la simulación producto de que la data no fue suavizada.

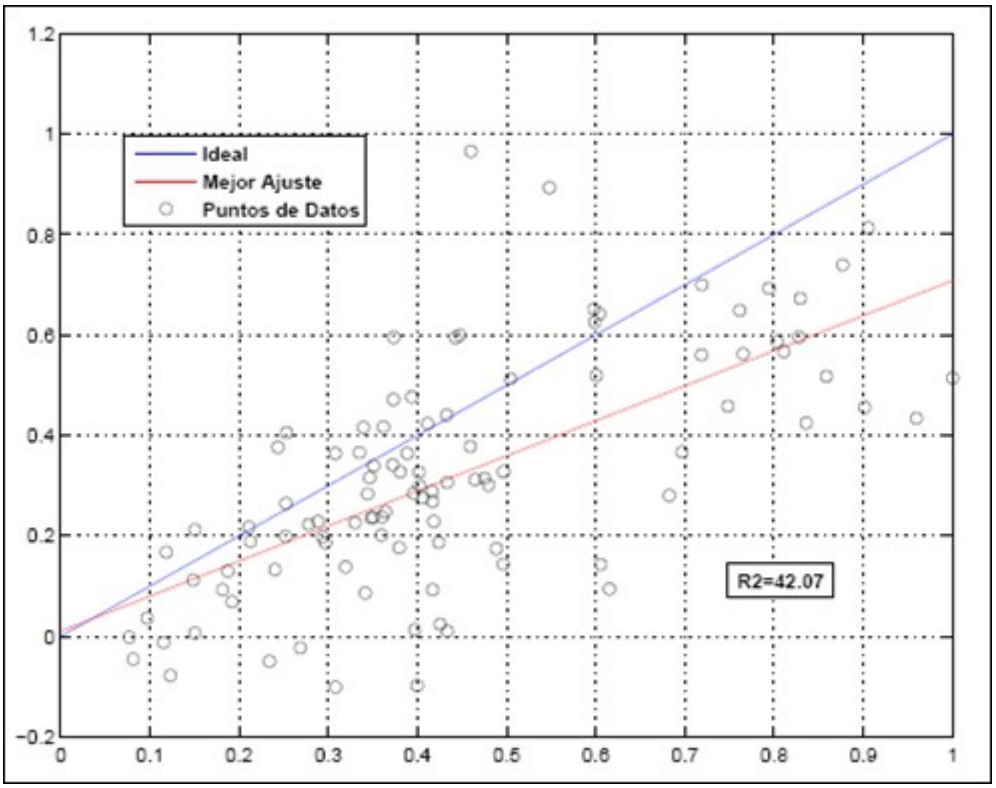

Figura 4.26: Dispersión data estimada v/s data real de test MLP (6, 16, 1) con data no suavizada.

La figura 4.26 corresponde al gráfico de dispersión de la data estimada v/s data real para test en su valor promedio. El gráfico corresponde a la topología ideal sin data suavizada compuesta por 6 nodos en la capa de entrada, 16 nodos en su capa oculta y un nodo en su capa de salida. La figura presenta un valor de correlación R^2=42,07, lo que es bastante bajo y esto radica en el hecho de que la data se encuentra sin suavizar.

Luego de analizar los resultados obtenidos por la arquitectura ideal sin suavizado, en la siguiente sección se establece una comparación respecto a la sección 4.1 en donde se analiza la misma topología ideal, pero con la data suavizada.

4.3 Comparación Arquitectura ideal con suavizado v/s Arquitectura ideal sin suavizado

Luego de realizar la calibración de los parámetros para obtener la arquitectura ideal con data suavizada en la sección 4.1 y de la simulación con data sin suavizar en la sección 4.2, se obtuvieron los rendimientos de los predictores en ambas secciones, los cuales se detallan a

continuación en la siguiente tabla resumen, teniendo como base la arquitectura ideal (con y sin data suavizada) y como métricas el Error Cuadrático Medio, Desviación Estándar y Factor de Correlación.

Tabla 4.29: Comparación arquitectura ideal con suavizado v/s arquitectura ideal sin suavizado.

Métricas **Arq. Ideal**	**ECM**	**STD**	**R^2**
Con Suavizado	20,83	1,91	91,88
Sin Suavizado	14,19	16,72	42,07

La tabla 4.29 presenta los resultados de los rendimientos promedios de test, para error cuadrático medio, desviación estándar y R^2 de la arquitectura ideal con y sin suavizado de la data.

El rendimiento promedio de R^2 para test de la arquitectura ideal con suavizar corresponde a 91,88%, valor que es muy superior al rendimiento promedio de la arquitectura ideal sin suavizar el que alcanza a un 42,07%, por consiguiente es mucho más conveniente utilizar en nuestro predictor la data suavizada, ya que de esta forma los resultados obtenidos por la simulación, los cuales se ven reflejados en la tabla anterior, nos servirán para mejorar los resultados anteriormente obtenidos por otros predictores.

El 91,88% de R^2 para test, correspondiente a la arquitectura ideal ajustada en la sección 4.1, nos indica que nuestro predictor cumple con lo esperado, es decir, el valor de su factor de correlación demuestra que el predictor podría ser útil en cualquier pesquería y además supera a los rendimientos anteriores de otros predictores.

Capítulo 5

Conclusión

En el capítulo 2 de esta memoria de título han sido presentadas las Redes Neuronales en general. Dejando en claro que las Redes Neuronales Artificiales han demostrado ser un modelo computacional capaz de resolver una gran cantidad de problemas en distintas áreas. En nuestro caso en particular, las Redes Neuronales Autoregresivas han demostrado ser un aproximador universal al momento de pensar la solución del problema, el cual es la predicción de la captura de Anchovetas en la zona norte de nuestro país.

En el capítulo 3 se presentaron los algoritmos de aprendizaje. Debemos señalar que el algoritmo de aprendizaje Backpropagation, el cual ha sido utilizado en diversas aplicaciones, presenta serias limitaciones, ya que su ajuste de sus pesos es muy lento. Es por esta deficiencia que se decidió que el algoritmo de aprendizaje Levenberg Marquardt sea el seleccionado para la implementación del predictor finalmente, ya que según la evaluación de las métricas Error Cuadrático Medio y Coeficiente de Determinación mejora las deficiencias del algoritmo anteriormente mencionado.

En el capítulo 4 se encontró la topología ideal, la cual consta de tres capas. Su capa de entrada compuesta por 6 nodos, su capa oculta por 16 nodos y finalmente su capa de salida por 1 nodo. La simulación se dividió en dos etapas, la primera etapa de training, en donde fue considerado un total del 75% de la data de captura de anchovetas, utilizando el algoritmo de aprendizaje Levenberg Marquardt, y una segunda etapa de test en donde fue considerado el restante 25% de la data, en donde el modelo obtuvo un coeficiente de determinación de un 91% de varianza explicada encontrado con la topología ideal de la red neuronal propuesta RN(6,16,1), valor que mejora el mejor resultado obtenido anteriormente por otro predictor en [16], el cual estaba compuesto por 6 nodos en su capa de entrada, 15 nodos en sus dos capas ocultas y un nodo en su capa de salida.

Los resultados obtenidos demuestran que se pueden solucionar muchos inconvenientes que presentan las pesquerías en la actualidad, debido a que este tipo de modelo predictivo permite

la transformación no lineal de los datos de entrada y salida, lo que mejora los problemas que presentan técnicas de estimaciones clásicas en el día de hoy. También se debe destacar que la solución de problemas a través de modelos predictivos de este tipo es de vital importancia en la actualidad en el mercado de Business Intelligence (BI).

Finalmente, se debe destacar la satisfacción personal conseguida luego de obtener los resultados anteriormente mencionados en base al trabajo realizado. Sin duda alguna este modelo predictivo puede perfeccionarse aun más, sin embargo, para ello se necesita un mayor conocimiento de algoritmos de aprendizaje para realizar de una mejor manera la etapa correspondiente al training de la red, factor que puede ser considerado como desventaja dentro de la implementación de una red neuronal.

REFERENCIAS

[1] Shaffer, G., S. Hormazabal, O. Pizarro y S. Salinas. "Seasonal and interannual variability of currents and temperature over the slope of central Chile". J. Geophys. Res. 104, C12, 29,951-29,961, 1999.

[2] Shaffer, G., O. Pizarro, L. Djurfeldt, S. Salinas y J. Rutllant. "Circulation and low-frequency variability near the Chilean coast: Remotely forced fluctuations during the 1991-92 El Niño". J. Phys. Oceanogr. 27, 217-235, 1997.

[3] Hormazabal, S., G. Shaffer y O. Pizarro. "Tropical Pacific control of intraseasonal oscillations off Chile by way of oceanic and atmospheric pathways". Geophys. Res. Lett. 29(6), doi: 10.1029/2001GL013481, 2002.

[4] Pizarro, O., G. Shaffer, B. Dewitte y M. Ramos. "Dynamics of seasonal and interanual variability of the Peru-Chile Undercuurent". Geophys. Res. Lett. 29(12), 10.1029/2002GL 014790, 2002.

[5] Yoh-Han Pao. "Adaptive Pattern Recognition and Neuronal Networks", Addison – Wesley Publishing Company, Inc, 1989.

[6] Enciclopedia Británica, Macropedia Knowledge in Depth, 15 th Edition, Vols. 15, 16, 1980.

[7] Thomas, D.W.P., Woolfson, M.S. "Voltage and current phasor estimation during abnormal conditions for transmission line protection schemes". Sixth International Conference on Developments in Power Protection, 1997.

[8] Mehra, P., Wah, B., Artificial Neural Networks, IEEE Computer Society Press, Piscataway, USA, 1997.

[9] Haykin, S., Neural Networks: A Comprehensive Approach, IEEE Computer Society Press, Piscataway, USA, 1994.

[10] Minsky, M., Papert, S., Perceptrons: An introduction to Computational Geometry. MIT Press, Cambridge, Mass., Second Edition, 1988.

[11] Hagan, M., Menhaj, M.: "Training Feedforward Networks with the Marquardt Algorithm", IEEE Transactions on Neural Networks, 1994.

[12] Marquardt, D.: "An algorithm for least squares estimation of non-linear parameters", J. Soc. Ind. Appl. Math., 1963.

[13] Gedam, S.G., Beaudet, S.T.: "Monte Carlo Simulation using Excel Spreadsheet for Predicting Reliability of a Complex System", Proceedings Annual Reliability and Maintainability Symposium, 2000.

[14] Coifman, R.R., Donoho, D.L.: "Translation-invariant denoising, Wavelets and Statistics", Springer Lecture Notes in Statistics 103, pp. 125-150, Springer-Verlag, 1995.

[15] Nason, G., Silverman, B.: "The stationary wavelet transform and some statistical applications, Wavelets and Statistics", Springer Lecture Notes in Statistics 103, pp. 281-300, Springer-Verlag, 1995.

[16] Gutiérrez-Estrada, J.C., Rodríguez, N.: "Monthly catch forescasting of anchovy Engraulis ringens in the north area of Chile: Non-linear univariate approach", Fisheries Research 86, pp 188-200, 2007.

ANEXO A – CÓDIGO FUENTE

```
clear
clear all
load D_smooth

%Training data

L=floor(length(x)*0.75);
u=y;
entrada=x(1:L,:);
y=u(1:L);
```

```
%Test data

xv=x(L+1:end,:);
yv=u(L+1:end);
PR=minmax(entrada');

rand('seed', sum(100*clock));

net = newff(PR,[16 1],{'logsig'  'purelin' },'trainlm');

net.trainParam.epochs=                                  200;
net.trainParam.goal=                                    1e-6;
net.trainParam.max_fail=                                5;
net.trainParam.mem_reduc=                               1;
net.trainParam.min_grad=                                1e-10;
net.trainParam.mu=                                      0.001;
net.trainParam.mu_dec=                                  0.1;
net.trainParam.mu_inc=                                  10;
net.trainParam.mu_max=                                  1e10;
net.trainParam.show=10;
net.performfcn='msereg';
net.performParam.ratio = 7e-3; %2e-3;

% Training the network

net                     =                       train(net,entrada',y);
z = sim(net,entrada');

figure(1)
t=1:length(z);
plot(t,y,'-b',t,z,'-r')
legend('Azul:                   Original','Rojo:                    Estimada')
%title('Training')
xlabel('Meses')
ylabel('Toneladas')
grid                                                    on
```

```
save red net
clear net

%Test

load red
z1 = sim(net,xv');
t=1:length(z1);

figure(2)
tt=1:length(yv);
plot(tt,yv,t,z1,'-r')
legend('Azul: Original','Rojo: Estimada')
%title('Testing')
xlabel('Meses')
ylabel('Toneladas')
grid on

figure(3)
[xx,yy,zz]=postreg(yv,z1)
grid on

se=z1-yv;
se=sum(se.^2)/length(se);
fprintf('\n R2 TEST: %e',zz*zz)
fprintf('\n MSE TEST: %e',se)
[a,b]=rlin(yv',z1');
xlin = 0 : 0.001 : 1;
ylin = a*xlin+ b;

figure(4)
plot(xlin,xlin,'-b',xlin,ylin,'-r',yv,z1,'ok')
legend('Ideal','Mejor Ajuste','Puntos de Datos')
% axis([0.5 1 0.5 1.0])
grid on
```

```
% print -depdf –r500 fig4
```